Reliability

May 2015

System and Software Assurance

IEEE

Reliability Society

Reliability Society Administrative Committee (AdCom) Members

Officers (Excom)

President
Christian Hansen

Sr. Past President
Jeffrey Voas

Jr. Past President
Dennis Hoffman

Vice Presidents

Technical Activities
Shiuhpyng Winston Shieh

Publications
W. Eric Wong

Meetings and Conferences
Alfred Stevens

Membership
Marsha Abramo

Secretary
Scott Abrams

Treasurer
Bob Loomis

Elected Members-at-Large (with vote)

TERM EXPIRES 2015 (DEC 31)
Marsha Abramo
Loretta Arellano
Lon Chase
Shiuhpyng Winston Shieh
Alfred Stevens
Rex Sallade

TERM EXPIRES 2016 (DEC 31)
Lou Gullo
Christian Hansen
Pradeep Lall
Zhaojun (Steven) Li
Bob Loomis
Pradeep Ramuhalli

TERM EXPIRES 2017 (DEC 31)
Joseph A. Childs
Pierre Dersin
Lance Fiondella
Carole Graas
Samuel J. Keene
W. Eric Wong

Standing Committees and Activities/Initiatives

Web Master
Lon Chase

Standards
Lou Gullo

Chapters Coordinator
Loretta Arellano

Professional Development
Marsha Abramo

Constitution and Bylaws
Dennis Hoffman

Fellows
Sam Keene

Finance
Christian Hansen

Academic Education/Scholarship
Sam Keene

Meetings Organization
Alfred Stevens

Membership Development
Marsha Abramo

Transactions Editor
Way Kuo

Newsletter Editor
Joe Childs

Video Tutorials
Christian Hansen

Nominations and Awards
Jeffrey Voas

Life Science Initiative
Peter Ghavami

Transportation Electification
Sam Keene
Pradeep Lall
Michael Austin

IEEE Press Liaison
Dev Raheja

IEEE Government Relations Committees

Energy Policy
Jeff Voas

Transportation and Aerospace Policy
Scott Abrams

Medical Technology Policy
Jeff Voas

Critical Infrastructure Protection
Sam Keene

Career and Workforce Policy
Christian Hansen
(corresponding only)

Intellectual Property Committee
Carole Graas
(corresponding only)

Research and Development Committee
Pradeep Lall

Combined TAB Ad Hoc Committee on Attracting Industrial Members
Dennis Hoffman

2013 TAB Awards and Recognition Committee (TABARC)
Dennis Hoffman

Technical Committees

Vice President for Technical Activities

Shiuhpyng Winston Shieh, National Chiao Tung University

Email: ssp@cs.nctu.edu.tw

Technical Committee on Internet of Things (IoT)

Chair: **Jeffrey M. Voas**, National Institute of Standards and Technology

Email: jeff.voas@nist.gov

Co–chair: **Irena Bojanova**, Program Director, Telecommunications and Management, and Collegiate Professor, University of Maryland University College, USA

Email: irena.bojanova@umuc.edu

Committee Member:

1. George F. Hurlburt: CEO of Change Index

Technical Committee on System and Software Assurance

Chair: **Eric Wong**, University of Texas at Dallas

Email: ewong@utdallas.edu

Technical Committee on Prognostics and Health Management (PHM)

Chair: **Rex Sallade**, Sikorsky Aircraft Co.

Email: Rex.Sallade@SIKORSKY.COM

Co–chair: **Pradeep Lall**, Auburn University

Email: lall@eng.auburn.edu

Technical Committee on Big Data

Chair: **David Belanger**, Stevens Institute of Technology

Email: david.belanger@stevens.edu

Technical Committee on Trustworthy Computing and Cybersecurity

Chair: **Wen-Guey Tzeng**, National Chiao Tung University

Email: wgtzeng@cs.nctu.edu.tw

Committee Member:

1. Raul Santelices: Assistant Professor, Department of Computer Science and Engineering, University of Notre Dam
2. Brahim Hamid: Associate Professor, IRIT Research Laboratory, University of Toulouse, France

Technical Committee on Reliability Science for Advanced Materials & Devices

Chair: **Carole Graas**, Colorado School of Mine & IBM Systems and Technology Group

Email: cdgraas@mines.edu

Technical Committee on Systems of Systems

Chair: **Pierre Dersin**, Alstom Transport

Email: pierre.dersin@transport.alstom.com

Technical Committee on Resilient Cyber-Systems

Chair: **Pradeep Ramuhalli**, Pacific Northwest National Laboratory

Email: pradeep.ramuhalli@pnnl.gov

Standards Committee

Chair: **Louis J Gullo**, IEEE RS Standards

Email: Lou.Gullo@RAYTHEON.COM

Committee Member:

1. Ann Marie Neufelder: Softrel, LLC – Owner; IEEE P1633 Standard Working Group Chair
2. Lance Fiondella: Assistant Professor, Dept. of Electrical and Computer Engineering, University of Massachusetts Dartmouth; IEEE P1633 Standard Working Group Vice Chair
3. Steven Li: Assistant Professor, Industrial Engineering and Engineering Management, Western New England University; IEEE P61014 Standard Working Group Chair
4. Diganta Das: Research Staff at the Center for Advanced Life Cycle Engineering, University of Maryland
5. Sony Mathews: Engineer at Halliburton, IEEE P1856 Standard Working Group Chair
6. Mike Pecht: Chair Professor and Director of Center for Advanced Life Cycle Engineering, University of Maryland
7. Arvind Sai Sarathi Vasan: Research Assistant at Center for Advanced Life Cycle Engineering , University of Maryland; IEEE P1856 Standard Working Group Vice-Chair
8. Joe Childs: Staff Reliability/Testability Engineer at Lockheed Martin

Working Group on Education

Chair: **Zhaojun (Steven) Li**, Western New England University

Email: zhaojun.li@wne.edu

Committee Member:

1. Emmanuel Gonzalez, Jardine Schindler Elevator Corporation

Editorial Board

Editorial Staff

Special Issue Editor's Message

Software today plays a significant role in controlling the behavior of many systems, including mission- and safety-critical applications. A hazard caused directly or indirectly by software operation can have catastrophic consequences, including property damage, financial loss, and serious injury or death. Preventing such disasters from occurring is thus of paramount importance.

Yet the disappointing truths are that software is far from defect-free and that very large sums of money are spent each year only to fix and maintain defective software. In fact, it is estimated that more than 60% of the cost of software development is spent on testing and debugging. This situation has become even more challenging because not only is software becoming larger and more complicated, but the industry also suffers from tighter scheduling and budget requirements.

Thus, in order to produce dependable and trustworthy systems with reduced code, it is absolutely essential to develop advanced cutting-edge technologies that can assure software quality by incorporating cost-effective procedures and tools tailored to various scenarios.

This need for cutting-edge technologies is also the reason to have this issue dedicated to Software Quality Assurance. The first paper is "Intersecting Definitions of V&V" by Dr. Mark Paulk that provides in-depth review of what verification and validation are and their respective objectives. The second paper by Dr. Mehra Nouroz Borazjany discusses the performance testing of a real-world system and lessons learned from that project.

We would like to thank both authors for sharing their ideas and experiences on this important topic. Special thanks also go to Professor Shiuhpyng Shieh and his editorial team for all their support.

W. Eric Wong
Professor &
Director of Advanced Research Center on Software Testing and Quality Assurance
Department of Computer Science
University of Texas at Dallas
Richardson, Texas, USA

Reliability

IEEE Reliability Society

Special Issue on System and Software Assurance

Performance Testing of a Real-World System

Mehra Nouroz Borazjany
Advanced Research Center on Software Testing and Quality Assurance
Department of Computer Science
The University of Texas at Dallas
Richardson, Texas
mehra@utdallas.edu

***Abstract*—One major concern for most companies is to assure that the capacity of their system meets their customers' performance requirements. In this short paper we describe our approach to performance testing of an asset tracking system that provides asset tracking to very large number of customers. For the system under test, we collect data from the available production system and then evaluate the capacity of the system under existence and future expected load. We used jmeter to simulate the communication of tracking devices with the system platform and gather performance metrics such as system response time, resource utilization, and throughput. This helped us to validate whether the system can scale to handle the future expected capacity. This being done by identifying the bottlenecks and allocating enough resources. Additionally, performance testing enabled us to verify the system recovery time after a spike load**

***Keywords*—Performance Testing, Capacity Planning, System Assurance**

I. Introduction

Companies measure the Quality of their Services (QOS) in terms of resource utilization, throughput and response time. Expected QOS assures more satisfied customers which leads to more business opportunities. We need to identify how a system responds to a specified set of conditions and inputs, which is the process of performance testing.

The main goal of performance testing is to identify how well your system performs in relation to your performance objectives. These objectives are often specified in a service level agreement. Additionally, performance testing helps define the conditions under which the system will fail, how it will fail, and what indicators should be monitored to notify us of an impending failure.

There are three major types of performance testing that share similarities yet accomplish different goals. Load testing, verifies system behavior under normal and peak load conditions. Stress testing, leads to evaluate our system's behavior when it is pushed beyond the normal or peak load conditions. Capacity testing helps us identify a scaling strategy to determine whether we should scale up or scale out based on our future growth.

It is important to have a performance testing plan for our system to ensure that it can perform under expected and peak load conditions. Ideally, the result of such testing can determine when and how to upgrade resources and scale the system sufficiently to handle increased capacity. Thus, we allocate adequate resources where they will generate the most benefit.

II. QOS Measurement

Quality of service (QOS) is the overall performance of the system under test. In order to measure the QOS, several quantitative aspects of the system are studied, such as response time, resource utilization, and throughput.

One of the key quality factors of a system is response time. We need to measure end-to-end response time and find the speed of the system under load.

Another key factor is resource utilization. We measure the resource utilization in terms of the amount of CPU, RAM, network I/O, and disk I/O that system consumes under normal or peak loads.

Throughput of a system is also another key factor. Throughputs are rate of services that a system can handle or process per unit of time.

III. TOOLS

There are several tools available to simulate load. We can simulate load in terms of users, connections, data, and etc. Also, they help us gather performance-related metrics such as response time, requests per second, and performance counters from remote server computers. In our project, we used jmeter [1] to simulate the traffic and gather metrics.

In addition, we need monitoring tools to monitor the status of various network services, servers, and other hardware. In this project, all of our servers are monitored by Zabbix[2].

IV. The System Under Test

The system under test provides asset tracking to very large number of customers. The front-end of the system is a web application that our customers can login and track their assets. They can use geofencing, over speed alerts and other exception notifications to ensure their assets are where there should be and safe even when they are not actively managing them.

The back-end is Apache ActiveMQ [3] server, which communicate with a Gateway. Asset tracking devices send information such as IMEI, device type, gps location, service type, and event type to the Gateway. The Gateway transfers the messages to the relevant ActiveMQ server. (Fig. 1)

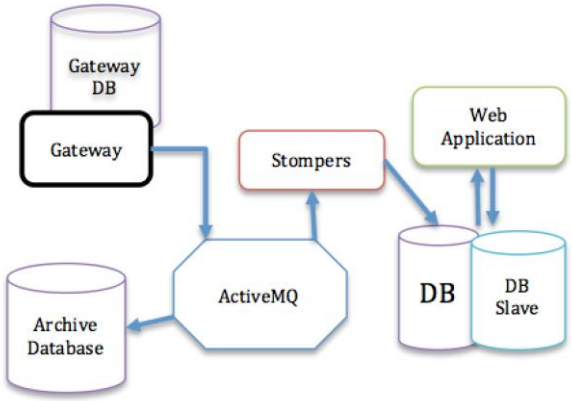


Figure 1: Overall System Architecture

Like any other testing, we need to have a testing plan and strategy based on the nature of the test and the architecture of the system under test. In the plan, we need to specify the goal and scope of the test, our approach, all the possible preparations, all risks, and also some technical details.

The goal of our performance testing for asset tracking system is to validate that the system can scale to handle the expected capacity planned for the coming year. We want to protect our system against crashes and slowdowns on instantaneous peak and verify the system recovery time after the spike load. Thus, we need to answer the two most common questions. First, how many concurrent users are going to use this system. Second, how many devices will communicate with the platform, which basically means how many messages per second needs to be handled.

The first approach is to simulate the activity of devices communicating with the platform. The second approach is to simulate the activities of the end-users logging in and using the web application. The third approach is to simultaneously simulate both devices and the end-users to measure capacity levels at key points within the architecture. In this paper we will only describe the first approach.

Some preparations are also needed. First, because we don't want to run our testing on production system, we need to prepare a staging system, which mirrors the production system. Therefore, tests against it can confirm the capacity of production environment.

Second, we need to characterize the message flow and user activity. We can do this by observing the production archive data. If the system is new, then we can characterize it based on different use case scenarios.

Next, we need to populate the database according to our test plan. For example, to simulate the activities of a certain number of devices, users or accounts, we need to create test accounts/devices within the application infrastructure.

Next, we will prepare the test scripts according to the plan and run them using simulation tools. We then capture the capacity levels using operational monitoring tools.

The risk involved in this project is that some of the network and system resources are shared between our production and our staging systems. As a result, if any of these shared resources prove to be a capacity bottleneck, the production system could be impacted during the testing.

Last but not least is to analyze the results. Depends on our results and intermediate findings, we may require multiple runs. Finally we are ready to report our findings.

V. Results

As it is discussed, according to the testing plan, the goal is to verify whether the system can scale to handle the expected capacity planned for the coming year. The first approach is to simulate the activity of devices communicating with the platform. As it is shown in Fig 2, we sent 74k messages/sec (spike load) to the ActiveMQ server. As a result, system was able to receive all the messages and hold them in the queue without any loss or crash.

Also as it is shown in the system architecture (Fig. 1), stompers are reading the messages from the queue, process them and store them into the database. It is shown in the Fig. 2, that stompers are able to process 70 messages per second. Moreover, it shows the recovery rate of the system. For example, it took the system 1000s to process all the messages and recover.

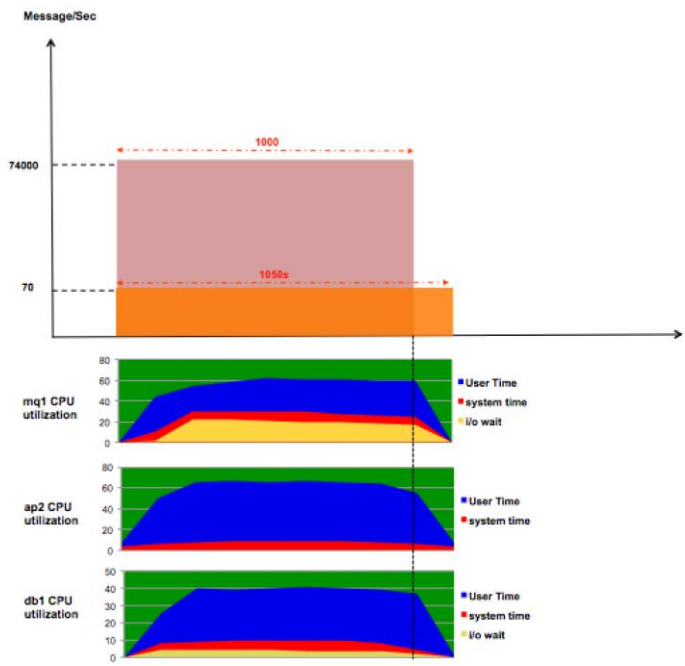


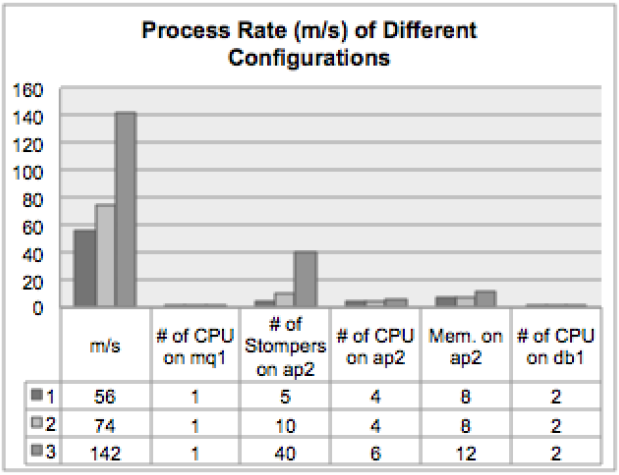
Figure 2: Capacity testing

The actual capacity of the system was 50 m/s. In order to increase the capacity of the system to 150 m/sec, we need to make some changes to the system configuration. For example,

we could increase the number of stompers from 5 to 10, which would increase the process rate to 70 m/s.

Fig. 3 displays some of the modifications with their process rates. As it is shown, in order to increase the capacity to 150 m/s, we need to increase the number of stompers to 40, and allocate more CPUs and memories to the server.

The second approach is to simulate the activity of the end-users logging in and using the web application. We believe that there are more studies describing the web application performance testing such as Kunhua Zhu et al.[9] and Yanyan Lu et al.[10]; therefore, in this short paper we only explained the back-end performance testing.



	m/s	# of CPU on mq1	# of Stompers on ap2	# of CPU on ap2	Mem. on ap2	# of CPU on db1
1	56	1	5	4	8	2
2	74	1	10	4	8	2
3	142	1	40	6	12	2

Figure 3: Configurations

References

[1] http://jmeter.apache.org/

[2] http://www.zabbix.com/about.php

[3] http://activemq.apache.org/

[4] D.A. Menascé and V.A.F. Almeida, "Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning", Prentice Hall, Upper Saddle River, N.J., 2000.

[5] P.J. Denning and J.P. Buzen, "The Operational Analysis of Queuing Network Models," ACM Computing Surveys, vol. 10, no. 3, Sept. 1978, pp. 225-261.

[6] D.A. Menascé and V.A.F. Almeida, "Capacity Planning for Web Services: Metrics, Models, and Methods", Prentice Hall, Upper Saddle River, N.J., 2002.

[7] D.A. Menascé et al., "A Methodology for Workload Characterization of E-commerce Sites," Proc. First ACM Conf. Elec- tronic Commerce, ACM Press, New York, 1999, pp. 119-128.

[8] S. Vinoski, "Web Services Interaction Models: Part I: Current Practice," IEEE Internet Computing, May/June 2002, pp. 89-91.

[9] Kunhua Zhu, Junhui Fu, and Yancui Li, "Research the performance testing and performance improvement strategy in web application", 2nd international Conference on Education Technology and Computer (ICETC), 2010.

[10] Yanyan Lu, Haiyan Wu, Yingxue Wang, "Web Application Performance Analysis Based on Comprehensive Load Testing", ICWMMN2006 Proceeding

Author Biography

Dr. Mehra Nouroz Borazjany is a Senior Lecturer in the Department of Computer Science at the university of Texas at Dallas. Additionally Dr. Borazjany is a member of the Software Testing and Quality Assurance Center. She received her Ph.D. degree in May 2013 from the Computer Science and Engineering department at the University of Texas at Arlington. She had the opportunity to be supervised by Prof. Jeff Lei in the Software Engineering Research Center (SERC) Group.

The topic of her PhD research was in the area of automated software analysis, testing and verification. Before joining UTD, she worked as a Performance Test Engineer.

Intersecting Definitions of V&V

Mark C. Paulk
Advanced Research Center on Software Testing and Quality Assurance
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75080
Mark.Paulk@utdallas.edu

***Abstract*—Verification and validation are two terms with three overlapping definitions in the closely related fields of systems and software engineering: 1) fulfill the conditions imposed by the previous phase; 2) reflect the requirements; and 3) fulfill the intended use. All three concepts are important and address distinct needs. One can argue that the first, the traditional software engineering definition of verification, has been largely superseded by agile methods.**

***Keywords*— verification, validation**

When the Capability Maturity Model Integration for Development (CMMI-DEV) was being developed, I noticed something odd about the Verification and Validation process areas. Coming out of a software engineering background, I was familiar with the definitions from IEEE 610 [1]:

- **verification.** (1) The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.
- **validation**. The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements.
- **verification and validation (V&V).** The process of determining whether the requirements for a system or component are complete and correct, the products of each development phase fulfill the requirements or conditions imposed by the previous phase, and the final system or component complies with specified requirements.

For verification, "conditions imposed at the start of that phase" implied, for example, that the code implemented the design. The Capability Maturity Model for Software (Software CMM) [2], which was one of the source documents for CMMI, used the IEEE 610 definitions. In CMMI-DEV [3], however, the purposes of the process areas and the definitions of the terms are:

1. The purpose of **Validation** (VAL) is to demonstrate that a product or product component fulfills its intended use when placed in its intended environment.
   - **Validation**. Confirmation that the product or service, as provided (or as it will be provided), will fulfill its intended use. In other words, validation ensures that you built the right thing.
2. The purpose of **Verification** (VER) is to ensure that selected work products meet their specified requirements.
   - **Verification**. Confirmation that work products properly reflect the requirements specified for them. In other words, verification ensures that you built it right.

These definitions were consistent with those used in the Systems Engineering CMM [4] and EIA 731 [5] but differed, perhaps significantly, from those used in software engineering. They intersected, albeit with different terms for the same concept, but there were really three concepts in systems and software engineering captured by these two terms.

Concept	Software Engineering	Systems Engineering
fulfill the conditions imposed by the previous phase	verification	--
reflect the requirements	validation	verification
fulfill the intended use	--	validation

One might consider these differences to reflect the different foci of systems engineering and software engineering. Traditionally systems engineering, at least for hardware/software systems, has had the lead on interfacing with the customer and users. Such an interpretation of responsibilities leaves a gap for software-only systems, which also need systems thinking even if systems engineering may not have a formal role.

A challenge for any project is the gap between what the user needs and the requirements that supposedly express those needs. The systems engineering sense of validation attacks the problem of missing and misunderstood requirements.

One might argue that a system's fulfilling its intended use in its operational environment is primarily a requirements elicitation problem – what is the role of "validation" against such an amorphous target? In requirements inspections, however, we expect the developers to ask probing questions into whether a correct and complete set of requirements has been elicited, using checklists and scenarios to confirm that the stakeholders have articulated their needs and desires effectively. Teams using agile methods, for example, maintain a dialog with the customer and users throughout the development process to ensure that the project gets timely feedback on how accurately the system is addressing the customer's (evolving) needs.

The mechanisms we use are not necessarily constrained by our jargon. The systems engineering sense of validation could be addressed by requirements inspections; design inspections could address software engineering validation and systems engineering verification; code inspections could address systems and software engineering verification. One can easily conclude that all three concepts are important and should be addressed, even if the terminology may be context dependent. Unfortunately many of the organizations that adopt inspections focus on code inspection rather than requirements and design inspection, where the greatest benefit accrues.

Perhaps, however, software engineering is evolving in the direction of the systems engineering perspective. Consider the growth of agile methods over the last twenty years. In the agile world, design documentation is de-emphasized. Design is emphasized as a verb rather than a noun. As iterative and incremental development move to very short increments where requirements analysis, design, code, and test are performed in a compressed manner, does it make sense to worry about IEEE 610's 1990 view of verification?

As a side note, I agree with agile's de-emphasis on detailed design documentation because my own experience with detailed design documents is that they quickly move out of synch with the code as the requirements change. Even maintaining an up-to-date requirements document can be a challenge. Detailed design documents are rarely useful in maintenance because they don't reflect what the code is actually doing. I am, however, an advocate for architecture as useful top-level design documentation that can and should be maintained as a system evolves.

If one takes an agile perspective, then fulfilling the conditions imposed by the previous phase becomes a meaningless concept. The "previous phase" is compressed down to an activity in a short iteration. Maintaining traceability from requirements, perhaps in the form of user stories, to code and then test cases remains a worthwhile exercise, but arguably the systems engineering sense of V&V should dominate.

Not all projects, however, are good candidates for agile methods. In life critical or high reliability systems, the need for thoughtful analysis of requirements changes implies that the project cannot be agile in the sense of responding rapidly to changing requirements even if it has adopted agile practices elsewise. The individual practices of the agile world derive for the most part from good software engineering and management practices that have been cranked up to an extreme implementation, so a project could in principle adopt an agile methodology even if it is not "agile" in the sense of responding rapidly to change. Projects in a high reliability environment are likely to use a plan-driven approach along with good practices inspired by their agile counterparts.

If we accept that many projects will continue to use a plan-driven style, does the IEEE 610 kind of verification continue to add value? Arguments can be made on both sides of this question; my point is that we should understand the terms that we use and impose on others. If V&V evolves to a more useful set of terms, that should be a conscious decision on the part of those defining and using those terms.

In discussing this point with the authors of various standards and models, I have so far found no one who was concerned about these intersecting definitions. If a project applies peer reviews (in particular inspections) throughout the life cycle, then all three senses of V&V can be addressed via a single mechanism, rendering the distinction moot.

CMMI-DEV characterizes verification as an inherently incremental process occurring throughout product development. By implication, peer reviews will take into consideration the inputs to each engineering process, e.g., design as an input to code, in their search for defects. Thus the model that originally sparked this question implicitly addresses the intersecting definitions.

In recent years standards writing bodies have tended to follow the path laid out by CMMI; entities originally focused on software engineering standards have evolved to address both systems and software engineering. The current IEEE glossary, for example, is IEEE 24765 [6], which addresses the vocabulary of both software and systems engineering. It includes five definitions of validation from various standards, including one from ISO 12207 (Software Life Cycle Processes) [7]: validation is, in a life cycle context, the set of activities ensuring and gaining confidence that a system is able to accomplish its intended use, goals and objectives.

Thus we see that software and systems engineering have become thoroughly intertwined. Once useful distinctions have

become blurred and/or evolved. We should not, however, allow jargon to cloud our consideration of these three intersecting concepts. It is our responsibility as professionals to consider whether and how each of these three should be appropriately addressed on our projects. This can become a particularly pointed question if our organizations are undergoing appraisals against CMMI-DEV, assessments against ISO 15504, or other kinds of "certification". Understanding the relevant definitions of V&V can prevent unwelcome surprises.

References

[1] "IEEE Standard Glossary of Software Engineering Terminology," IEEE 610.12, September 1990.

[2] M.C. Paulk, C.V. Weber, B. Curtis, and M.B. Chrissis, The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley, Boston, 1995.

[3] M.B. Chrissis, M.D. Konrad, and S. Shrum, CMMI for Development: Guidelines for Process Integration and Product Improvement, Third Edition, Addison-Wesley, Upper Saddle River, NJ, 2011.

[4] R. Bate, D. Kuhn, C. Wells, J. Armitage, G. Clark, K. Cusick, S. Garcia, M. Hanna, R. Jones, P. Malpass, I. Minnich, H. Pierson, T. Powell, and A. Reichner, "A Systems Engineering Capability Maturity Model, Version 1.1," Carnegie Mellon University, Software Engineering Institute, CMU/SEI-95-MM-003, November 1995.

[5] "Systems Engineering Capability Model," Electronic Industries Alliance, EIA 731.1, August 2002.

[6] "Systems and Software Engineering – Vocabulary," ISO/IEC/IEEE 24765, December 2010.

[7] "Systems and Software Engineering - Software Life Cycle Processes," ISO/IEC/IEEE 12207, January 2008.

Author Biography

Dr. Mark Paulk teaches software engineering and is a member of the Software Testing and Quality Assurance Center in Computer Science Department at the University of Texas at Dallas. He was a Senior Systems Scientist at the Institute for Software Research at Carnegie Mellon University from 2002 to 2012. From 1987 to 2002, Dr. Paulk was with the Software Engineering Institute at Carnegie Mellon, where he led the work on the Capability Maturity Model for Software. He was co-project editor of ISO/IEC 15504-2 (Software Process Assessment: Baseline Practices Guide) and a contributor to ISO and IEEE standards. He is a co-author of the eSourcing Capability Model for Service Providers. Dr. Paulk received his PhD in industrial engineering from the University of Pittsburgh. He is a Fellow of the ASQ and a Senior Member of the IEEE.

Submission Instructions

Authors should use the designated IEEE Reliability Manuscript Central Website to summit their papers. Please refer to the following steps to submit your papers:

1. Login to IEEE Reliability Manuscript Central. If you have no account, sign up for one.
2. Click "Authors: Submit an article or manage submissions".
3. Please click "CLICK HERE" at the bottom of this page, and you will be brought to the five-step submission process.
4. You need to 1) choose the section that you are going to submit your paper to; 2) complete the submission checklist; 3) enter the comments for the editor, which is optional; 4) save and continue.
5. If you have any supplementary files, please upload them in step 4.

Manuscript Types

Manuscripts for regular issues fit within the scope of the magazine, but are not intended for a special issue. Special issue manuscripts cover a specific topic scheduled on our editorial calendar. Please select the appropriate issue (manuscript type) when uploading your manuscript. For more information and to see upcoming special issue topics, see our Editorial Calendar at http://rs.ieee.org/reliability-digest/author-guidelines.html.

Typing Specifications

The manuscript should be written in Times New Roman in a double-column format. The typical length of the submitted manuscript is 4 single-spaced pages. The text portion of the manuscript should be in 10-point font and the title should be in 24-point font, bold.

Manuscript Length

The typical length of the submitted paper is 4 pages, including text, bibliography, and author biographies. Please note that proper citations are required.

Illustrations

The illustrations in the articles must be cited in the text and numbered sequentially. Captions that identify and briefly describe the subject are needed as well. In order to avoid dense and hard-to-read illustrations, graphs should show only the coordinate axes, or at most the major grid lines. Line drawings should be clear. To prevent potential layout problems from happening, related figures described within the same section of text should be grouped together as parts (a), (b), and so on.

References

All manuscript pages, footnotes, equations, and references should be labeled in consecutive numerical order they are mentioned in the text. Figures and tables should be cited in text in numerical order.

Biographical Sketch

A brief biographical sketch should contain the full title of the paper and complete names, affiliations, addresses, and electronic mail addresses of all authors. The corresponding author should be indicated.

Please provide a short biography and a picture for each author of your paper at the end of your paper. The short biography should contain no more than 150 words

Copyright

IEEE Reliability Society owns the copyrights of the articles published in Reliability. If you wish to reproduce the copyrighted materials, please contact us and seek the permissions. The contents on this website can be referenced with proper citation.

Special Issue Proposal Submissions

For a special issue in Reliability, experts are welcome to serve as our guest editors. To know more information, please contact Editor-in-Chief on Reliability, Shiuhpyng Shieh: ssp@cs.nctu.edu.tw.