Vision 2020: The Future of Software Quality Management and Impacts on Global User Acceptance

Robin Poston[1(✉)] and Ashley Calvert[2]

[1] University of Memphis, Memphis, USA
rposton@memphis.edu
[2] System Testing Excellence Program, Canada, USA

Abstract. This paper explores the future evolution of software quality management (SQM), testing, and global user acceptance approaches keeping in mind the evolution in software and technology quality management in general, including new technologies and the increasing adoption of new software development life cycle methodologies such as Agile and Scaled Agile. These evolutions are forcing quality organizations to change the way they approach software quality processes, including increased outsourcing of development, and the need to update traditional testing and user acceptance testing approaches which lag behind with manual and invasive techniques. User acceptance as we know it today must evolve.

This discussion about evolution should deemphasize the role of the end user in "testing" and emphasize the end user's role in acceptance, adoption, and ability to influence the quality and usability of software much further upstream in the development life cycle. User acceptance teams should increase their role in user experience, the development of usability standards, non-invasive automation techniques gathering usage data, etc. All of these mechanisms should increase the ability of the end user to influence product quality and enhance the user experience and acceptance. Perceptions of end user's participation in user acceptance events need to be transformed away from just another cycle of software testing. User acceptance should not be about testing, but about validating that the end user needs and expectations have been met. All testing and other quality processes should be completed and defects corrected before the end user is engaged in the process of "accepting" the deliverable.

Thus, this effort will explore the future of SQM and its impacts on global user acceptance. We will discuss how organizations involve users throughout the development life cycle to facilitate adoption, user experience, and usability of new technologies. This discussion will be embedded in the use of futuristic new technologies and development methodologies. To explore these notions, this study gathers input from technology visionaries about best practices approaches for facilitating SQM and user acceptance throughout the development life cycle.

Keywords: Global user acceptance · User acceptance testing · Software development life cycle · Software quality management

F.F.-H. Nah and C.-H. Tan (Eds.): HCIB 2015, LNCS 9191, pp. 748–760, 2015.
DOI: 10.1007/978-3-319-20895-4_70

1 Introduction

Software quality management (SQM) has been a key driver enabling teams to deliver software at the user-accepted quality levels. The focus of these management approaches is to adopt specific quality assurance activities throughout the development life cycle with input from developers, customers, and users (Poth and Sunyaev 2014), that ultimately improve customer's acceptance of software quality at release time. Prior research has examined various supporting mechanisms including the adoption of total quality management principles and process knowledge management (Shang and Lin 2009), among others. While software testing activities tend to remain relegated to the final stages of development work, SQM brings a more holistic approach to ensure user acceptance, keeping in mind today's global reach of enterprise user groups.

Development life cycles have evolved over time increasing in quality management focus. Starting with structured formalized methodologies in the 1960s and1970 s, early approaches required each stage of the life cycle from inception of the product to release of the final system performed under a rigid, sequential timeline. Still in use today, these plan-driven methods typically have hierarchical organization structures, formalized project manage processes (Agarwal and Sambamurthy 2002), and tend to be characterized by predefined process phases, pre-approved designs, and agreed-to requirements documentation. More recently, industry has moved toward agile methodologies in the 1990s and 2000s, characterized by flexibility and openness to embrace and learn from changing work routines, requirements, and designs being responsive to shifting customer demands (Conboy 2009, Ramesh et al. 2006). Today, methodologies, such as Scaled Agile Framework (SAFe), offer guidelines for adopting agile in large enterprises, by addressing challenges with architecture designs, integration activities, funding mechanics, governance oversight, and role assignments. However, quality management and development life cycles will continue to evolve as innovative, cutting-edge technologies are introduced, e.g., artificial intelligence, visualization tools, and augmented reality.

With the focus on quality management and more rapid flexible software development capabilities, user acceptance testing will inherently need to adapt. The purpose of user acceptance testing is to gather software product feedback from actual system users, those who have experience with the business processes and will be using the system to complete related tasks (Klein 2003, Larson 1995). Actual users bring knowledge of processes and work activities and are able to assess how the system meets what is required of it. UAT typically occurs after development is complete but before the product is released. As business systems become more complex and decentralized, UAT on a global scale becomes more complicated to perform (Poston et al. 2014). The execution of UAT events needs effective participation of geographically distributed actual system users. The evolution toward increasing global dynamics, quality focus, and more rapid development life cycles is forcing UAT organizations to change the way they approach software quality processes. Organizations are beginning to see the need to update traditional UAT approaches which lag behind with manual, invasive, and time-consuming techniques (Poston et al. 2014). Thus, how will UAT evolve?

This study examines the future evolution of UAT through an assessment of the future of SQM and impacts on global user acceptance. The evolution should deemphasize the role of the end user in "testing" and emphasize the end user's role in acceptance, adoption, and ability to influence the quality and usability of software much further up-stream in the software development life cycle. User acceptance teams should increase their role in user experience (UX), the development of usability standards, non-invasive automation techniques gathering usage data, etc. All of these mechanisms serve to increase the ability of the end user to influence product quality and enhance the user experience and acceptance. Perceptions of end user's participation in UAT events needs to be transformed away from just another phase of software testing to become about validating that the end users' needs and expectations have been met. All testing and other quality processes should be completed and defects corrected before the end user is engaged in the process of "accepting" the deliverable. The idea is to move toward an 'operational readiness' mindset.

This study set out to describe the future of SQM and its impacts on global user acceptance. We will discuss how organizations involve users throughout the software development life cycle to facilitate adoption, user experience, and usability of new technologies. This discussion will be embedded in the use of futuristic new technologies and development methodologies. To explore these notions, this study gathers input from technology visionaries about best practices forward-thinking approaches for facilitating SQM and user acceptance throughout the development life cycle.

2 Literature Background

The goal of SQM is to place the focus on how software is progressing in quality as it is being developed. A quality software product will meet its requirements as set by the users or how much it satisfies user needs. The idea is to create a culture of quality within the organizational environment holding everyone responsible for evolving a product's quality throughout the development life cycle. Keeping this in mind, testing and global user acceptance approaches are well aligned to help support SQM. While regulated to the end of the waterfall life cycle, testing and user acceptance activities have been migrating toward the earlier stages of software development. With the introduction of agile methodologies, testing and user acceptance team members become equally important on software development teams. With the move to greater quality management, the infusion of new technologies, such as artificial intelligence and virtualized automated autonomous testing, becomes even more promising. See Table 1 in the Appendix A for advances in these technologies and their potential implications for quality management. As the Table illustrates, research has advanced our understanding of how smarter tools can be created to support SQM goals. This paper explores the use of such technologies in advancing how user acceptance techniques might evolve in the 21st century based on interviews with top visionary of the field.

3 Exploratory Study Methodology

The research methodology follows an exploratory approach in gathering case study data on automated testing and artificial intelligence practices in order to provide descriptive and explanatory insights into the management activities in software development work. This approach has been used successfully in prior research (Pettigrew 1990, Sutton 1997) and allows us to induce a theoretical account of the activities found in empirical observations and analysis of team member's viewpoints. This approach is also known to lead to accurate and useful results by including an understanding of the contextual complexities of the environment in the research analysis and outcomes. This approach encourages an understanding of the holistic systematic view of the issues and circumstances of the situation being addressed, in this case the issues of managing development projects from team member perspectives about their testing practices (Checkland et al. 2007, Yin 1984). To identify the future of user acceptance practices and tools, we interviewed experts in artificial intelligent systems, software testing, user experience, and automated systems.

4 Data Collection

The results reported in the present study are based on interviews with identified experts in the fields predicted to influence the software development life cycle, e.g., artificial intelligence, visualization tools, and augmented reality. Our data gathering began with the creation of semi-structured interview protocols which comprised both closed and open-ended questions. To inform our interview question development, we reviewed documentation about the technology, and the relevant scholarly literature. The data collection methods focused on interviewees' perspectives on visualized automated testing and artificial intelligence issues, roles played by various stakeholders involved, and the challenges of UAT. Face-to-face and phone interviews of approximately 1 to 1.5 h were conducted. In total, we held 33 interviews, conducted between November 2014 and January 2015, with additional follow-up clarification Q&A sessions conducted over e-mail. The expertise of those interviewed is available upon request. Each expert provided input on their field of expertise and extrapolated ideas to proposed future scenarios of SQM.

By collecting and triangulating data across a variety of methods, we were able to develop robust results because of the perspectives we gained about testing and user experience technologies and issues. This approach provides in-depth information on emerging concepts, and allows cross-checking the information to substantiate the proposed future of SQM (Eisenhardt 1989, Glaser and Strauss , 1967, Pettigrew 1990).

5 Findings

In this research, we gathered and analyzed interview data from a panel of experts. Our findings offer insights into the future of SQM and the impacts on global user acceptance. The future will entail better tools and techniques for user centric design and

development of software systems. Future process will infuse end user perspectives in the design and validation of new software at the front end of the development life cycle. Similar to the Boeing's 777 design where significant user input was used (Birtles and Boeing 1998, Weiner 1990), enterprise systems development will incorporate input from more and more end users or employees via the use of automation. Boeing was able to coordinate over 200 design teams with about 40 members each by automating the process by using three-dimensional CAD software systems enabling team members to assemble and simulate a virtual aircraft to verify that the thousands of components would work together properly (Norris and Wagner 1999). Boeing also successfully used visualization systems for design reviews and production illustrations (Abarbanel and McNeely 1996). The Boeing 777 is successfully being used by dozens of airlines today. Figure 1 illustrates the vision of the future of SQM best practices.

5.1 Scaled Agile Frameworks

Agile software development is increasing in prevalence and should be considered in any vision of the future. Agile is being found across settings to deliver faster time to market, improved productivity, better quality, and greater morale. Agile methods are increasingly being used in high assurance and regulated environments where the cost of errors is high. For large scale software development, Scaling Software Agility (Leffingwell 2007) and Agile Software Requirements (Leffingwell 2011) have been shown to apply at enterprise scales. Agile methods, when properly implemented, enable teams to focus on better understanding the end user needs, and building high-quality software, including highly reliable and safe systems. While there will always be the need to address defects, Agile methods are being usefully applied across industries and project

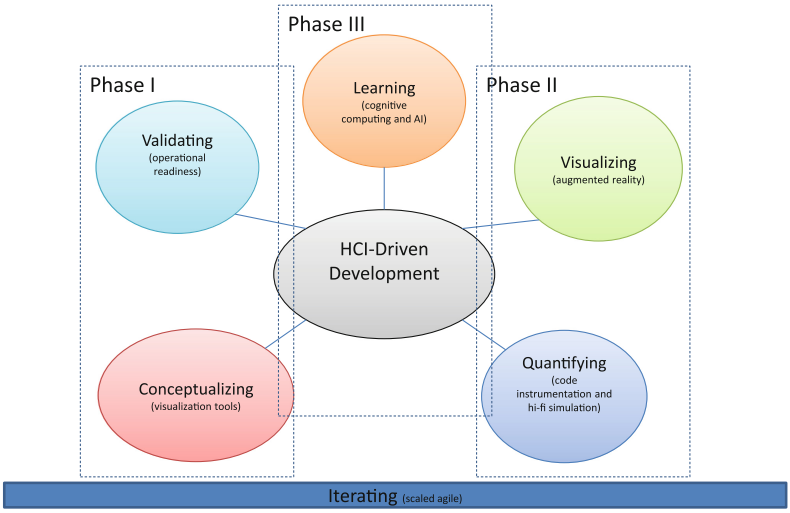

Fig. 1. Conceptual model of HCI-driven software development

sizes. User acceptance validation becomes an integral part of agile frameworks given the method calls for active, continuous user, i.e., user proxy, participation or operational readiness input.

5.2 Operational Readiness

Utilizing all or some of the concepts and tools above, user acceptance testing teams can implement quality assurance plans covering the entire development life cycle. By measuring technical debt along the life cycle, acceptance teams can measure a technical debt ratio as exit criteria for input to go/no go decisions. This is essentially a risk profile of weighted features that becomes an operational readiness score, customized for that release for that system, and acceptance teams become readiness evaluators. This process can be built into scaled agile frameworks and application life cycle management systems.

Creating a quality management plan at the beginning of a project, quality teams can expand acceptance criteria beyond simply defect counts and classifications, to develop a new "Risk Profile" model. Using a "Technical Debt" ratio (e.g., configuration control or software versioning) becomes a rated criteria), test environment availability becomes a rated criteria. Continuous integration, automated testing and other forms of testing also become rated criteria. In this way, "user acceptance testing" becomes "operational readiness evaluation"; test cases become "operational readiness scenarios"; and exit criteria become "operational readiness score".

5.3 Visualization Tools

The next generation of automating testing activities will involve requirements visualization tools, e.g., iRise (see: www.irise.com/). These tools enable user input into requirements through visual formats, enabling more descriptive communications and knowledge transfer of software product development. Enabling all of the software development stakeholders, including users, to virtually 'see' product designs, these tools create visualizations that look and act almost identical to the real end product. End users may not be necessary if proxy combinations of experts are involved, e.g., those trained in user experience, architectures, development, operations, and quality analysis. Visualization helps users and those who create new software to illustrate what the product will do before a single line of code is written, helping people understand what the product is going to be like.

Visualization tools with high-fidelity representations have been known to reduce costs and decrease development time. They create and communicate innovative, revenue-enhancing ideas without coding, helping global teams to deliver apps that delight the first time, virtually eliminating the re-work that many companies have been forced to expect. Through simulations, software designs will be refined with the end users' needs and interests in mind.

5.4 Augmented Reality, Code Instrumentation, and High-Fidelity Simulation

Along with instrumentation, future generations of automating testing activities will involve augmented reality, which will be used to support user-system interactions offering additional informative overlays of explanation of system features. Instrumented code provides the ability to monitor the software's performance, diagnose errors, and track information values and usage. Special code instructions monitor specific parts of the system. Operational scenarios then can be provided via augmented reality while enterprise users are performing daily activities, while instrumented code tracks how users respond to the overlays. Video snippets, sensory awareness, and other tools can offer users more informative direction for testing actions. This approach can be modelled by companies such as User Testing (see: www.usertesting.com/) or Applause (see: www.applause.com/) supporting distributed testing based on concepts of cloud-sourced testing. Building on systems such as LARIAT, the Lincoln Adaptable Real-time Information Assurance Testbed, which uses automated high-fidelity real-time evaluations of system usage (Rossey, et al. 2002), artificial intelligence techniques can be used to learn then mirror and simulate the real-time user experience as input to continuous automated testing systems. With applications embedded into production systems, intelligence systems can monitor user interactions with technology and model users and their behaviors. Artificially intelligent testing systems then mimic real users and be used to realistically evaluate new software development designs and coded products.

5.5 Cognitive Computing and Artificial Intelligence

Cognitive computing systems learn and interact naturally with people to extend what either humans or machine could do on their own. Future generations of automating testing activity will involve cognitive chips, e.g., the latest SyNAPSE chip by IBM (see: www.research.ibm.com/cognitive-computing/neurosynaptic-chips.shtml#fbid=h9hs1Q-C8Ra). This chip is expected to initiate a new type of software applications that can respond to sensory information. The IBM chip is built based on a brain-inspired computer architecture with 1 million neurons and 256 million synapses. This technology has the potential to utilize a cognitive hardware and software ecosystem within a business enterprise that learns user work habits and software usage anomalies. Utilizing technology that can gain an automated profound understanding of the actual facets of how employees use enterprise software will enable real-time virtualized testing and quality management to be performed throughout the SDLC. The technology will be able to mimic a greater breadth and depth of user experiences as it monitors everyday use to understand how, when, and where users get work done across the enterprise network.

5.6 Vision 2020: Future of Completely Automated User Testing

Combining cognitive chips, augmented reality, visualization tools, agile frameworks, and operational readiness measures, tomorrow's end user acceptance activities will take

on a new look. In the future, virtual end users based on artificially intelligent autonomous systems will interact with simulated versions of new software to utilize reality-driven operational scenarios to perform user acceptance validation before code is written. More efficient and faster than humans, these systems will deploy more scenarios continuously augmented with every day usage behaviors. Neural networks will continuously collect the activities of real-time global users to update scenarios. An event that occurred yesterday will be replicated in simulation and testing scenarios today. Problems will be addressed in real-time. This approach eliminates the need to incur the expense of flying globally located users to a central location to review and provide input on software requirements and manually test software products. Figure 2 delineates the environmental components needed to make HCI-driven software development a reality.

Autonomous testing systems with automated learning will not be constrained by one person's understanding of system use, but will incorporate the mission of many user types, understand a more holistic environmental scope, and potentially outthink the devices under test. These test vehicles will be fed in real-time the artificial intelligence gathered about users around the globe to continuously build and update test scenarios. This approach eliminates the issue of test team members joining user acceptance teams as user proxies with operational knowledge which becomes obsolete with time. The goal is to shift user validation left as a virtual activity before software is coded, through the combined use of augmented reality, cognitive computing, and artificial intelligence. To further support this approach, code and simulations can be modified with instrumentation to measure usage patterns.

UAT events will change based on advances being made in code instrumentation and simulation. The focus is to design requirements visually within simulation to allow real end users to 'test drive' the software before it is created. Using operational

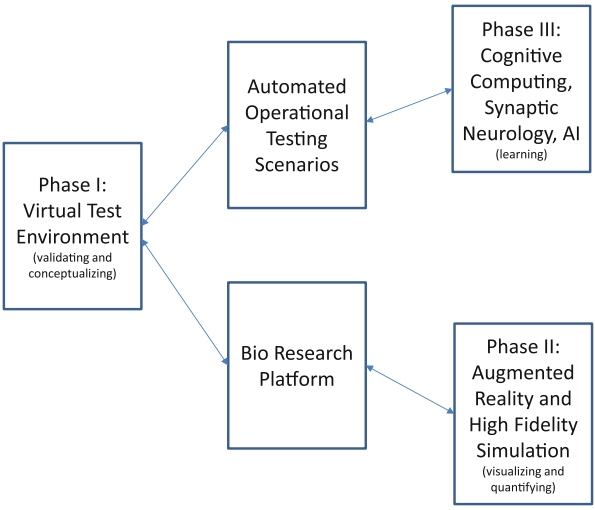

Fig. 2. Vision 2020: HCI-driven development environment

scenarios in the simulations based on known usage patterns, autonomous artificial intelligence systems can use neural networks to scan actual live production systems for real user activities. The operational scenarios can be continuously updated based on how users change how they use the software and anomalies in usage. These systems can provide real world operational scenarios to the simulations. Thus, the scenarios are updated in real-time to correctly map to actual system user usage activities and patterns. Eventually the simulations become autonomous testing devices that represent end user actions. Consistent with the concepts behind the Internet of Things, autonomous devices can be programmed and updated to think like end users and change as users change their usage activities. Using autonomous devices updated in real-time eliminates reduces the need for human involvement, removing issues of a non-representative sample of end users or outdated knowledge regarding current business processes. Autonomous UAT devices will perform mission-based scenarios with real environmental parameters, representing the current user experience, and be able to outthink humans as UAT participants, e.g., a faster supercomputer processes 8.2 billion megaflops (million operations per second) and the human brain only 2.2 billion megaflops (Fischetti 2011).

Software development processes will focus more on design and prototypes, with great user involvement. The next generation high-fidelity simulations, e.g., iRise, will offer developers the means to involve greater user input. With instrumentation, augmented reality, and artificial intelligence, user proxies will eventually replace actual users. In addition augmented reality can be added to production systems to introduce future system modifications to users prior to their development in order to collect feedback. As these systems autonomously gather usage information, operational scenarios can be augmented by the feedback and used in test lab environments.

Appendix A Table. Recently-Published Select Studies

Citation	Type of artificial intelligence	Potential implications on SQM
Khan(2014)	Case-based reasoning	Helps to reduce the knowledge availability bottleneck
Mims(2014)	Schedule group meetings	Software produces marketing e-mail messages for clients
Rusli(2014)	Analysis of customer feedback language	Software in the development of their business strategies and marketing
Han et al. (2014)	Using multiple classification ripple down rules based agile approach	Used agile development for overcoming difficulty of analysis, and business rules approach for reducing issues in maintenance
Zapf (2013)	Software agent platforms	Categories of software agent systems and their properties
Padgham et al. (2013)	Oracle generation method for unit testing	Oracle Generation for Automated Unit Testing of Agent Systems
Šerić et al.(2013)	Intelligent forest fire monitoring system	Artificial perception system whose aim is early detection of forest fires

(*Continued*)

(*Continued*)

Citation	Type of artificial intelligence	Potential implications on SQM
Simonite (2012)	Artificial chat partner	Development of chat software for instant messaging
Powell (2011)	People are the only solution to software problems	Requirements gathering and software modeling through artificial intelligence
Oprea (2011)	University knowledge management system	Software tools that assist the decision making process
Henderson-Sellers (2011)	Meta-models and ontologies	Contribution of meta-models and ontologies for software engineering
Omoteso et al. (2010)	Information and communications technology (ICT) tools	Software development to help auditors match the complexity of their clients' information systems
Chang 2010	Artificial slow intelligence systems	Applications of slow intelligence Systems in software engineering
Cohen et al. 2010	Herbal toolset	Development of intelligent agents using established software-engineering principles
Singh et al., 2010	Predicting software development effort using artificial neural network	Software effort estimation of cost, time and manpower using feed forward network trained using back-propogation algorithm using training and validation data of 650 projects
Schneidewind (2010)	Applying neural networks to software reliability assessment	Neural networks to assess the reliability of software, employing cumulative failures, etc., method proved superior for reliability.
Farah(2009)	Techniques developed in artificial intelligence (AI)	Artificial intelligence techniques in software engineering
Kapur et al. (2008)	Software reliability assessment using artificial neural network	Apply neural networks to build software reliability growth models. Logistic function provides improved goodness-of-fit.
Sagarna and Lozano 2008	Dynamic search space transformations for software test data generation	Propose test data generation with definition of the initial search space using static information extracted from source code. Grid search method is promising option for test data generation
Citation	Type of Autonomous Systems	Potential Implications on SQM
Warwick (2014)	Automatic control systems	Use of autonomous and nondeterministic control systems in aeronautics
Safiullah (2014)	Methodology techniques	Strategy for testing the autonomous system integrations domain
Garcia et al. (2011)	Multiagent systems	Evaluation and comparison of MAS software engineering techniques
Jiao (2011)	Autonomous software entities	Autonomous component to model independent software entity
Jiao et al. (2010)	Automated assembly of internet-scale software systems with autonomous agents	Systems are modeled by dynamic trial-and-evaluation strategy to select high quality agents to facilitate the interoperations among autonomous agents

(*Continued*)

(*Continued*)

Citation	Type of artificial intelligence	Potential implications on SQM
Trivino et al. (2009)	Semiautonomous robot tele-control systems	Role of operator and autonomous behavior of the robot
Citation	Type of Augmented Reality	Potential Implications on SQM
Sangani (2013)	Developing applications	Benefits & obstacles associated with the augmented reality GPS
Ferran and Salim (2012)	Distributed cognition supported by technology for knowledge sharing	Minimize knowledge bottlenecks with virtual reality and internet-based distributed cognition.

References

Abarbanel, R., McNeely, W.: Fly Thru the Boeing 777. ACM Siggraph, New york (1996)
Agarwal, R., Sambamurthy, V.: Principles and models for organizing the information technology function. Manag. Inf. Syst. Q. Executive **1**(1), 1–16 (2002)
Birtles, P.: Boeing 777, Jetliner for a New Century. Motorbooks International, St. Paul, Minnesota (1998)
Chang, S.: A general framework for slow intelligence systems. Int. J. Softw. Eng. Knowl. Eng. **20**(1), 1–15 (2010)
Checkland, K., McDonald, R., Harrison, S.: Ticking boxes and changing the social world: data collection and the new UK general practice contract. Soc. Policy Adm. **41**(7), 693–710 (2007)
Cohen, M., Ritter, A., Haynes, F.E., Steven, R.: Applying software engineering to agent development. AI Mag. **31**(2), 25–44 (2010)
Conboy, K.: Agility from first principles: reconstructing the concept of agility in information systems development. Inf. Syst. Res. **20**(3), 329–354 (2009)
Eisenhardt, K.M.: Making fast strategic decisions in high-velocity environment. Acad. Manag. J. **32**(3), 543–576 (1989)
Engr.Farah Naaz Raza.: Artificial intelligence techniques in software engineering (AITSE), In: *International MultiConference of Engineers and Computer Scientists* (1). (2009)
Ferran, C., Salim, R.: Distributed cognition supported by information technology can help solve the knowledge management bottleneck. Acad. Bus. Res. J. **4**, 432–454 (2012)
Fischetti, M.: Computers Versus Brains, *Scientific American*, 12 October 2011
Garcia, E., Giret, A., Botti, V.: Evaluating software engineering techniques for developing complex systems with multiagent approaches. Inf. Softw. Technol. **53**(5), 494–506 (2011)
Glaser, B., Strauss, A.:The discovery grounded theory: strategies for qualitative inquiry (1967)
Han, S., Yoon, H., Kang, B., Park, S.: Using MCRDR based agile approach for expert system development. Computing **96**(9), 897–908 (2014)
Henderson-Sellers, B.: Bridging metamodels and ontologies in software engineering. J. Syst. Softw. **84**(2), 301–313 (2011)
Jiao, W.: Using autonomous components to improve runtime qualities of software. IET Softw. **5** (1), 1–20 (2011)
Jiao, W., Sun, Y., Mei, H.: Automated assembly of Internet-scale software systems involving autonomous agents. J. Syst. Softw. **83**(10), 1838–1850 (2010)

Kapur, P.K., Khatri, S.K., Basirzadeh, M.: Software reliability assessment using artificial neural network based flexible model incorporating faults of different complexity. Int. J. Reliab. Qual. Saf. Eng. **15**(2), 113–127 (2008)

Khan, M.J.: Applications of case-based reasoning in software engineering: a systematic mapping study. IET Softw. **8**(6), 258–268 (2014)

Klein, G.S.: Lims user acceptance testing. Qual. Assur. **10**(2), 91–106 (2003)

Larson, G.B.: The user acceptance testing process. J. Syst. Manag. **46**(5), 56–62 (1995)

Leffingwell, D.: Scaling Software Agility: Best Practices for Large Enterprises. Addison-Wesley Professional, Boston (2007)

Leffingwell, D.: Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional, Boston (2011)

Mims, C.: Artificial intelligence comes to your inbox. Wall Street J. – East. Ed. **264**(47), B1–B2 (2014)

Norris, G., Wagner, M.: Modern Boeing Jetliners. Minnesota: Zenith Imprint, Minneapolis (1999)

Omoteso, K., Patel, A., Scott, P.: Information and communications technology and auditing: current implications and future directions. Int. J. Auditing **14**(2), 147–162 (2010)

Oprea, M.: A university knowledge management tool for academic research activity evaluation. Informatica Economica **15**(3), 58–71 (2011)

Padgham, L., Zhang, Z., Thangarajah, J., Miller, T.: Model-based test oracle generation for automated unit testing of agent systems. IEEE Trans. Softw. Eng. **39**(9), 1230–1244 (2013)

Pettigrew, A.M.: Longitudinal field research on change: theory and practice. Organ. Sci. **1**(3), 267–292 (1990)

Poston, R., Sajja, K., Calvert, A.: Managing user acceptance testing of business applications In: Proceedings of 16th International Conference on Human-Computer Interaction, Creta Maris, Heraklion, Crete, Greece, 22–27 June 2014

Poth, A., Sunyaev, A.: Efficient Quality Management: Risk- and Value-based Software Quality Management, IEEE Software (2014)

Powell, J.: Leadership by walking around: 21st century people solutions for software problems. Contract Manage. **51**(5), 56–65 (2011)

Ramesh, B., Cao, L., Mohan, K., Xu, P.: Can distributed software development be agile? Commun. ACM **49**, 41–46 (2006)

Rossey, L.M., Cunningham, R.K., Fried, D.J., Rabek, J.C., Lippmann, R.P., Haines, J.W., Zissman, M.A.: LARIAT: Lincoln Adaptable Real-time Information Assurance Testbed. IEEE (2002)

Rusli, E.M.: Deep dives into buyer minds. Wall Street J. – East. Ed. **263**(87), B5 (2014)

Safiullah, F.:Some Practical Considerations and a Methodology for Testing Autonomous System Integrations, Int. J. Softw. Eng. Appl. (IJSEA), 5 (2014)

Sagarna, R., Lozano, J.A.: Dynamics search space transformations for software test data generation. Comput. Intell. **24**(1), 23–61 (2008)

Sangani, K.: Developing AR apps. Eng. Technol. (17509637) **8**(4), 52–54 (2013)

Schneidewind, N.: Applying neural networks to software reliability assessment. Int. J. Reliab. Qual. Saf. Eng. **17**(4), 313–329 (2010)

Šerić, L., Štula, M., Stipaničev, D.: Engineering of holonic multi agent intelligent forest fire monitoring system. AI Commun. **26**(3), 303–316 (2013)

Shang, S., Lin, S., Wu, Y.: Service innovation through dynamic knowledge management. Ind. Manage. Data Syst. **109**(3), 322–337 (2009)

Simonite, T.: Artificial intelligence, powered by many humans. Technol. Rev. **115**(6), 14–16 (2012)

Singh, Y., Kaur, A., Bhatia, P.K., Sangwan, O.: Predicting software development effort using artificial neural network. Int. J. Softw. Eng. Knowl. Eng. **20**(3), 367–375 (2010)
Sutton, R.I.: Crossroads-the virtues of closet qualitative research. Organ. Sci. **8**(1), 97–106 (1997)
Trivino, G., Mengual, L., van der Heide, A.: Towards an architecture for semiautonomous robot tele control systems. Inf. Sci. **179**(23), 3973–3984 (2009)
Warwick, G.: Enabling autonomy. Aviat. Week Space Technol. **176**(20), 28–29 (2014)
Weiner, E.: New Boeing Airliner Shaped by the Airlines, The New York Times, 19 Dec 1990
Yin, R.K.: Case Study Research: Design and Methods. Sage Publications, Beverly Hills (1984)
Zapf, M.: Two decades of software agent platform engineering. PIK - Praxis der Informationsverarbeitung und Kommunikation **36**(4), 235–242 (2013)