Hindawi Publishing Corporation
Advances in Human-Computer Interaction
Volume 2014, Article ID 805307, 15 pages
http://dx.doi.org/10.1155/2014/805307

Research Article

Orchestrating End-User Perspectives in the Software Release Process: An Integrated Release Management Framework

Simon Cleveland and Timothy J. Ellis

Nova Southeastern University, 3301 College Avenue, Fort Lauderdale-Davie, FL 33314, USA

Correspondence should be addressed to Simon Cleveland; sc1674@nova.edu

Received 2 June 2014; Accepted 12 August 2014; Published 16 November 2014

Academic Editor: Anthony Savidis

Software bugs discovered by end-users are inevitable consequences of a vendor's lack of testing. While they frequently result in costly system failures, one way to detect and prevent them is to engage the customer in acceptance testing during the release process. Yet, there is a considerable lack of empirical studies examining release management from end-users' perspective. To address this gap, we propose and empirically test a release framework that positions the customer release manager in the center of the release process. Using a participatory action research strategy, a twenty-seven-month study was conducted to evaluate and improve the effectiveness of the framework through seven major and 39 minor releases.

1. Introduction

This paper extends prior research on customer-driven release models [1] by expanding scope of the release problem areas and proposing an integrated release framework with a specified role of the customer release manager.

A considerable portion of software bugs contributing to the largest system failures are discovered by customers in their production environments [2, 3]. A recent survey revealed that 46% of software developers do not perform thorough testing due to lack of time and 36% do not believe that their companies perform enough prerelease testing. Additionally, 88% of software development companies do not use fully automated test systems, opting for less reliable manual testing instead [4]. Reference [5] reported a system experiencing nearly 15,000 software bugs throughout a 12-year span. Reference [6] reported over 60% of organizations discovering major software errors in production.

Common service level agreements between the vendor and the users have provisions for resolution of software bugs; however, these bugs are more likely to require several fixes. Reference [7] found that software bugs discovered by customers are more likely to be reopened for fixing (up to 1.33 times more) due to a poor or incorrect fix in the first place, requiring more time to roll out new fixes and increased productivity and financial losses for the end-users. Causes for the undiscovered bugs include (1) difficulty for software developers to reproduce failures due to differences in environment, (2) customer privacy concerns regarding the type of data shared with vendor for off-site diagnosis, and (3) lack of automated feedback regarding the root cause of each failure [2, 8]. According to [9], companies sell buggier software products as a trade-off for earlier market entry, while 53% of software developers cite computing resources for their lack of software testing [4]. According to [10], unit testing, considered essential for software quality, is frequently omitted due to costs associated with personnel, automation [11], and insufficient testing infrastructure [6]. In the public sector, shrinking IT budgets account for the lack of computing resources. While software as a service (SaaS) or cloud models ease cost pressures by allowing customers to outsource some applications [12–14], most large information systems do not support these models. Instead, vendors of these systems follow the traditional model of software updates, by delivering patches and new versions at various intervals. As a result, it is imperative to find ways to minimize occurrences of software bugs in live production environments [15].

A proven cost effective method to reduce software bugs and improve service delivery is through the adoption of release management framework (Table 1), such as the

Table 1: Release Activity and Corresponding Articles.

Release Management Activity	Article and Description
Release Policy	[15, 34]
Release Planning	[35, 37]
Design and Develop Software	Variety of methods; handled by software or IT service vendor. Also see release acceptance [38, 39]
Build and Configure Release	Variety of methods; handled by software vendor.
Fit-for-Purpose Testing	[40–42]
Release Acceptance	[38, 39]
Roll-out Planning	[22, 23]
Communication, Preparation and Training	[23, 25]
Distribution and Installation	[24, 45]

Information Technology Infrastructure Library (ITIL) framework [16]. ITIL represents a set of documents created by United Kingdom's Central Computer and Telecommunications Agency (CCTA) in the 1980s with the goal of providing a common language between IT and business units and streamlining the improvement of IT service management processes. ITIL consists of a comprehensive collection of procedures designed to increase the value of IT operations. Organizations adopting ITIL define the structure and skills of their IT departments for a business-driven, top-down method of IT service management, improving end-to-end service availability [17–19]. Research on release management focuses on software vendor challenges associated with release planning [20, 21], open source release support [22], service provision [23], release automation [24], and patch management [25]. Reference [26] proposed a software product development framework incorporating aspects of release management and addressing the process from the perspective of the software vendor. Reference [27] proposed an EM3 framework based on two industrial models (Microsoft's and Oracle's). It addressed customer (acceptor) release management processes but was not tested in a "typical" acceptor organization. Instead, their framework was studied in an organization that developed and maintained software systems, with level-three support from a software vendor.

Empirical studies examining the release management process from an end-users' perspective are lacking. To address this gap, this paper presents a case study of the release management process from the customer's point of view. The goal of the study is to develop a method for better integrating end-user perspectives in the release process. The main research question is how can organizations improve their release management process while minimizing the amount of software bugs in the production systems? Based on the study's results, a framework is proposed to accomplish this goal and to improve the overall release management process, minimizing the occurrence of software bugs. The framework is unbounded by industry specific models and instead combines the traditional ITIL release management methodology with the project management methodology developed by the Project Management Institute (PMI). To our knowledge, this is the first empirical study to propose specific details of the customer release manager (CRM) role in each of the release management processes.

The balance of this paper will first review relevant literature in the area of release management, with a particular focus on each release component identified in the ITIL framework and articles addressing challenges in each activity. The research methods employed will then be detailed, followed by a review of the results. The proposed framework for incorporating the customer's perspective in the release management is presented with a description of the CRM's role for each process. The paper concludes with an overview of the study, limitations, and recommendations for future research.

2. The Literature Review

Release management, a process associated with the delivery of high quality software to users [28], consists of building, testing, packing, collaborating, and arranging tasks between the software vendor and the final users and deploying software and hardware releases effectively into production. Release management is executed in incremental steps involving both technical and management tasks covering the initial gathering of requirements, software development, and deployment into production with the ultimate goal of rolling out the final release packages into the end-users' production environments without issues and within a specified schedule [20, 23, 29]. Release management is a challenging process for software vendors who use it frequently as a mechanism to maintain the value of their software product [30]. Vendors provide release packages to multiple customers who have different environments and various needs. Some customers require new versions of the software to fix software bugs, while others require new enhancements. Depending on the complexity and component integration of the software application, the release may involve multiple resources and tasks, increasing the chance for errors [23, 31]. Only 45% of companies use specific frameworks for their release management process [20]. Several process frameworks have been developed to address release management, including ITIL, Controlled Objectives for Information and related Technology (COBIT), and ISO 20,000. ITIL, the most popular, structures software releases based on reported user-incidents, analyzed by software developers, and deployed as release packages into the end-users' environments [23]. The ITIL framework has been shown to increase customer satisfaction and improve operational performance [32].

The ITIL release management involves determining, acquiring, releasing, and deploying changes (such as bug fixes and enhancements) in an IT environment. The following table outlines the key release management activities according to the ITIL framework [33] and articles addressing challenges/solutions.

2.1. Release Policy. Release policy, the first activity of the release management process, represents a formal agreement on the strategic approach and includes (1) infrastructure used in the release; (2) acceptable schedule; (3) definition of major versus minor releases; (4) deliverables for each release; (5) roll-out and back-out plans; (6) documentation of releases; and (7) roles, responsibilities, escalation steps, contacts for vendors, and end-users [34]. This activity also ensures that each release has its unique number by which it can be tracked throughout the release management process. Versioning is used for the purpose of release stability and for the purpose of soliciting responses for an upcoming release. From the customer's perspective, the release policy is generally outlined in the contractual maintenance agreements negotiated between the vendor and the customer's contracts/procurement department at the time of the initial procurement of the system. Reference [15] considered the release policy activity in the scenario between a software vendor and a customer. They developed a release policy and a cost model that reduces costs for buggy software by extending the testing time after release. In this scenario, the vendor testing teams are not dissolved after a release but instead continue to test the system once it is in production. Any detected bugs are resolved with patches or new releases before they cause significant impact for the larger user community. Although this model can support a software vendor's early market entry, the risks are transferred to the end-users who accept the release earlier with the possibility of placing undiscovered critical bugs into production.

2.2. Release Planning. This activity is concerned with the designation of resources, the roles and responsibilities of these resources, agreement on policies and procedures to be used during the release, decision on deliverables, and features [35]. The key step is a predefined acceptance criterion specifying whether a release should be accepted in the production environment or paused and reverted to a prior release stage [34]. Release planning has been termed a "wicked" problem not guaranteeing stable software for the end-users regardless of frequency [22, 35, 36]. Reference [37] found that 69% of release planning challenges of software development companies were associated with the priorities and needs of the system stakeholders and the interdependencies of requirements. Additionally, they noted that the release planning process takes approximately 60% of a release manager's time. Reference [35] also approached the release planning process from the perspective of the software vendor. They developed a model heavily involving stakeholders (external customers) in the process, allowing them the opportunity to vote on features and assign priority to requirements.

2.3. Design and Develop Software. This activity involves the process of designing the software based on requirements and developing the software included in the release. End-users engage with the software developers in writing the acceptance test cases as defined in the release acceptance activity [38, 39]. Presently, this process has the least customer involvement, as the majority of customers do not get involved with the building of the software.

2.4. Build and Configure Release. This activity includes the compilation of modules stored in the software library to create the derived objects from the source objects. Reference [34] recommends the use of build procedures, tools, and checklists during the assembling of the release package to ensure repeatable practice with anticipated outcome. This activity is dependent on the software vendor's available tools. This process also has some of the least customer involvement, is very environment specific, and is not technology or process agnostic.

2.5. Fit-for-Purpose Testing. This activity includes functional, operational, performance, and integration testing of the release. Lack of resources (including testers and test environments) impedes testing, resulting in releases with software bugs [23]. Research heavily addresses this activity from the perspective of the software vendor. Vendors are concerned with the reliability of their software and use a variety of models to determine the probability of their product's failure-free operation for a given time in a given environment [10]. These models include reliability modeling [40], causal prediction [41], and good enough to release (GETR) [42]. During the fit-for-purpose testing activity, customers are not engaged in the testing of the software. According to [43], vendors can gain confidence in the operations of the system if the testing is performed to mimic the actual product use. As a result, software should be tested based on typical usage scenarios with feedback provided by the end-users.

2.6. Release Acceptance. This activity includes testing software by the end-users and obtaining approval for the release to proceed. During this activity, the release package is deployed to the customer's test environments with the coordination of the customer's technical team. Typically, testing is performed by end-users, and the release acceptance is based on specific conditions which the released package must satisfy [44]. Reference [25] found that release package quality was poor due to lack of quality control acceptance procedures established by the customer during the release acceptance activity. To streamline this activity, [38] proposed a tool allowing end-users to collaborate with developers earlier in order to write, automate, and execute acceptance tests during the software development cycle. Similarly, [39] proposed a method for generating user acceptance testing (UAT) test cases from behavioral use cases created during the requirements analysis phase. This study implies that the requirements and use cases should be confirmed by the end-users prior to the generation of UAT test cases.

2.7. Roll-Out Planning. This activity involves the creation of a time table, resources, responsibilities, and events during the distribution and installation of the release. Reference [22] discussed a time-based strategy which follows a predefined schedule with a cut-off date for inclusion of additional features. Additionally, a release checklist was discussed as a method to ensure that no steps were missed during new releases. Reference [23] recommended timetables and action plans for the release installation. Both studies addressed this activity from the perspective of the software vendor, expecting to communicate with various customer parties to coordinate the release (e.g., customer's IT department, the end-users, etc.).

2.8. Communications, Preparations, and Training. This activity, involving notifications to release stakeholders (including software development, release, and end-user teams), roll-out meetings, and training sessions, heavily involves the end-users in preparation of the release deployment. Clearly defined procedures are necessary to keep end-users informed of the upcoming release changes, and trainings must be organized before the release delivery [23, 25].

2.9. Distribution and Installation. This activity concerns the deployment of the final changes into the live environments. Since only 9% of companies use a proper release automation tool [20], the distribution and installation activity are essentially a manual process typically handled by the vendor's release manager [45]. Ramakrishnan proposed a tool addressing common distribution and installation issues such as files already existing in the environments and file permissions changes from prior releases. Reference [24] considered the distribution and installation activity as part of the end-users' responsibility for locating, retrieving, and assembling the software components. They proposed a release management tool ensuring distribution transparency and consistency of dependencies among component based software.

2.10. Release Manager. The release manager role is not defined in the ITIL framework, yet it plays a key figure in the software release management process. The release manager is expected to possess all the qualifications of experienced project managers such as judgment, community building, attention to detail, and communications and management skills. While 60% of a release manager's time is spent on release planning, 30% of the remaining time is spent on replanning activities due to changes in requirements and stakeholder priorities [37]. According to [20], 97% of software development organizations depend largely on the release manager to ensure successful releases. Reference [23] noted that a designated release managed with specified assignments and responsibilities can increase the quality of release and reduce the number of the incidents occurring after installation into the customer's production environment. Reference [22] briefly addressed the key responsibility of the software release manager by noting that he or she should be directing the development team, assessing the risks of each change proposed in the release, examining each software line of code, writing the release notes, and interacting with users. The release manager has complete control over the releases [15, 24, 37, 44]. However, while software vendors use release managers to coordinate their internal release activities, coordinating the customer resources (e.g., end-user testers, end-user test environments, and release dates) could be challenging for someone who is external to the customer organization. Vendor release managers may not be familiar with the customer's organizational and reporting structure. If releases provide fixes or enhancements for various departments, the release manager may need to coordinate availability of testing resources from various departments with their managers. At the same time, organizational priorities may deter promised resources and impact directly the release schedule. The customer's IT department resources need to be coordinated for the distribution of the release into the customer's testing and then production environments.

The ITIL framework, as implemented, does not fully address the issues that could cause software failure, because that framework is essentially focused on release management from the vendor's perspective. Our position is that expanding the ITIL framework to include the customer perspective in the form of a customer release manager (CRM) function would reduce those errors.

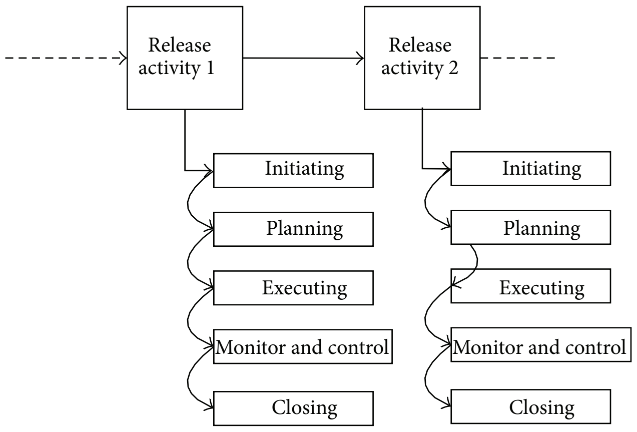


Figure 1: Release activity and project process group integration.

3. Proposed Framework

Based on the literature review, a customer-driven release management framework is developed that combines the ITIL's best practices framework for releases with the PMI process group methodology for project delivery. The framework considers each ITIL release management area as an unique subproject with incorporated PMI's structured process groups and respective tasks. Figure 1 demonstrates the integration points between the release activities and project management process groups.

This new framework aims to address the common release issues associated with communication, coordination of vendor, and customer involvement in the release management activities.

Releases can be considered to be projects, as they are temporary endeavors with predefined start and finish dates [46]. The PMI methodology is universally scalable and incorporates initiating, planning, executing, monitoring, controlling, and closing of temporary endeavors (e.g., releases).

The following section describes the role of the CRM in the framework's release and process group activities.

3.1. Release Activities and CRM

3.1.1. CRM Role in Release Policy. During the release policy activity, the CRM should formulate and document optimal acceptable release criteria for the customer based on discussions with internal stakeholders. These criteria should be included in the contract and negotiated with the software vendor at the time of software procurement. The criteria should include triggers for each release (e.g., change requests, entry and exit conditions for each activity, staff skills, service and operations level agreements, UAT criteria, release design option such as phased approach, or push or pull of new software [34]).

3.1.2. CRM Role in Release Planning. The release planning activity should also occur on the customer's end. It should include a comprehensive planning effort on behalf of the release manager who should address the scope, time frame, possible costs, resources which will test the release, quality control, risks, communications strategy, and procurement of required hardware and software. The planning effort should be coordinated with the vendor's release manager to confirm that planning estimates coincide with the vendor's resource availability and that all stakeholders' expectations are managed.

3.1.3. CRM Role in Design and Developing Software. During the design and developing software activity, the CRM should coordinate the development of user acceptance test cases between the end-users and the vendor's software team. This proactive effort will ensure that the vendor discovers issues during the fit-for-purpose testing activity and will not delay the release acceptance activity.

3.1.4. CRM Role in Build and Configuration Release. During the build and configuration activity, the CRM should communicate with the vendor release manager to address any issues associated with versioning of the release, security, and access rights that could jeopardize the release process.

3.1.5. CRM Role in Fit-for-Purpose Testing. During the fit-for-purpose testing activity, the CRM should organize a preliminary end-user testing with the vendor release manager into the vendor's test environment. Reference [34] recommended the use of pilots to test the release prior to the actual user acceptance testing activity. Any changes or issues should be communicated with the vendor release manager for resolution prior to the release acceptance activity.

3.1.6. CRM Role in Release Acceptance. During the release acceptance activity, the CRM should ensure that the new build is deployed to the customer's test environment and should communicate with the UAT team to initiate the testing process. He or she will also report any bugs discovered to the vendor release manager and ensure that fixes are provided by the vendor and retested by the UAT team. During this activity, the CRM would adjust the release plan based on release progress, document, and enforce any changes that are reported.

3.1.7. CRM Role in Roll-Out Planning. During the roll-out planning activity, the CRM should prepare a detailed roll-out checklist that outlines key tasks, start and end times, key resources and their responsibilities, system backups, and back-out procedures should the installation fail. Testing of the release in production should also be planned at this time. The checklist should be coordinated with the vendor release manager to include vendor resources.

3.1.8. CRM Role in Coordination, Communication, and Training. During the coordination, communication, and training activity, a decision on go or no go for the release should be made by all internal and external stakeholders. The CRM should direct end-users during the training sessions and host several trial runs of the upcoming installation with the roll-out team.

3.1.9. CRM Role in Distribution and Installation. Finally, during the distribution and installation activity, the CRM and the vendor release manager should coordinate the delivery of the release. The CRM should monitor the installation, the final production testing of the release by key end-users, and make a decision if the release should be stopped based on the reported production bugs.

3.2. Project Management Processes and CRM. The proposed release management framework provides a flexible and innovative methodology that could be utilized by many individuals, departments, and vendors for the effective release of software. Moreover, a definition of the CRM role within each release process group from the proposed framework is provided where the CRM is appointed to coordinate the release process on the customer's end. The CRM should be skilled in the PMI's project management methodology and possess sufficient technical skills to be able to execute certain engineering and database operations (such as rolling out application installers and performing database backups and restorations for roll-back purposes). The CRM should get involved in the formulation of the customer release team and work with the vendor release manager to facilitate the planning, delivery, testing, and installation of the final software packages. He or she would accomplish this by applying the project management process groups through the release management activities.

From the very inception of each release, the CRM will embark on a thorough planning effort that integrates careful consideration of the enterprise's environmental factors influencing the release's success (e.g., organizational culture, end-user testing availability, IT hardware and software resource administration, and established organizational communication channels). Table 2 outlines proposed release management activities and process group mapping. It demonstrates key tasks for the CRM.

Table 2: Proposed Customer Release Manager Activities and Process Group Mappings.

Release Activity	Project Management Process				
	Initiating	Planning	Executing	Monitoring and Controlling	Closing
Release Policy	Determine optimal policy for the organization with appropriate stakeholders	Develop the release policy and add to procurement documentation	Negotiate policy with vendor at software procurement time. Enforce changes	Ensure policy is followed by vendor and document changes if necessary	Document and archive lessons learned
Release Planning	Integrate lessons learned from prior releases. Review prioritized request for changes (RFC). Identify impacted end-users and address end-user enhancements needs	Plan release's scope (RFC, enhancements), schedule, potential costs, UAT (use cases, quality control testing), resource roles, communication strategy, risks, hardware/software procurement needs, roll-out strategy, training, distribution, and installation strategy. Re-plan based on changes	Coordinate plans with the vendor's release manager. Communicate plans with release stakeholders. Procure hardware or software. Enforce changes	Document changes based on vendor's release manager's feedback. Track and document any changes	Document and archive lessons learned
Design and Develop the Software	Review the release scope, schedule, end-user resources plans, and integrate lessons learned on UAT cases from prior releases	Plan required use cases for the RFCs and enhancements with end-users and vendor release manager. Re-plan if there are new changes	Direct and manage the documentation effort of end-users. Coordinate progress and distribution of use cases with vendor's release manager. Enforce changes	Verify all use cases have been completed and delivered. Track and document any changes	Document and archive lessons learned
Build and Configure the Release	Integrate lessons learned on build and configuration from prior releases	Re-plan if there are changes	Address potential issues associated with the compilation and configuration with vendor release manager (version, access permissions, and security). Enforce changes	Track and document any changes	Document and archive lessons learned
Fit-for-Purpose Testing	Integrate lessons learned on fit-for-purpose testing from prior releases	Plan the testing of a pilot. Re-plan if there are changes	Direct and manage the pilot testing. Enforce changes	Document and communicate issues and potential changes with end-users and vendor release manager	Document and archive lessons learned
Release Acceptance	Integrate lessons learned on release acceptance from prior releases. Review release plans	Re-plan if there are changes	Direct and mange the deployment of release into test environment, the acceptance testing and communications of issues with vendor release manager, and the deployment of fixes to issues. Adjust release plans based on progress. Enforce changes	Document and track reported issues, changes, and report on progress with stakeholders	Document and archive lessons learned

Table 2: Continued.

Release Activity	Project Management Process				
	Initiating	Planning	Executing	Monitoring and Controlling	Closing
Roll-out Planning	Integrate lessons learned on roll-out from prior releases. Identify stakeholders involved in the roll-out (e.g. IT, key end-users, vendor resources)	Prepare distribution and installation time table checklist including back out procedures, production testing, each roll-out activity, and each participant's role and responsibility. Re-plan if there are changes	Coordinate roll-out plan with vendor release manager. Enforce changes	Track and document any changes	Document and archive lessons learned
Communication, Preparation, and Training	Integrate lessons learned from prior releases. Identify stakeholders who will perform the training	Review release plan for training, levels of support after the release installation. Re-plan if there are changes	Document any procedures, and communicate with end-users and vendor release manager to address any improvements. Direct and manage training. Make final decision on go/no go for release with vendor release manager and stakeholders. Manage trial runs for the installation. Enforce changes	Track and document any changes	Document and archive lessons learned
Distribution and Installation	Integrate lessons learned on distribution and installation from prior releases	Re-plan if there are changes	Direct and manage the roll-out. Ensure backups are completed. Direct production testing and make decision of go-live versus stopping the release. Enforce changes	Track progress, document issues, and communicate with stakeholders	Document and archive lessons learned

3.2.1. CRM Role in Initiating Process. During the initiating process group, the CRM will determine release policy, define the initial scope of the release (including RFCs), schedule, costs, and identify both external stakeholders (e.g., vendor release manager and vendor resources) and internal stakeholders (organization's end-users, managers, and IT resources). Authorization will be obtained to proceed with the release and release charter will be completed. This process group feeds from lessons learned in prior releases and requires a different review of archived documentation by CRM.

3.2.2. CRM Role in Planning Process. During the planning process, the overall release plan will be defined along with comprehensive plans for the tasks and end-user involvement during each of the remaining release processes (the design and development of the software, build and configuration of the release, fit-for-purpose testing, release acceptance, roll-out, communication, preparation, training, distribution, and installation of the release).

3.2.3. CRM Role in Execution Process. During the execution process, the release manager will coordinate, direct, and manage the work done by the end-user release team as well as the internal and external stakeholders (including vendor). He or she will also enforce any changes discovered during the monitoring and controlling process.

3.2.4. CRM Role in Monitoring and Controlling Process. During the monitoring and controlling process, the release manager will track the progress of the release, monitor, and use corrective actions to realign the release back in line with the plan and document any changes. This process involves frequent communication with end-users to determine and inform users of pending changes.

3.2.5. CRM Role in Closing Process. During the closing process, the release manager will ensure that all deliverables have been released and lessons learned have been documented and archived.

4. Methodology

4.1. Participatory Action Research. To test the framework, a participatory action research (PAR) strategy was adopted [47, 48]. Action research has been shown to address real life issues by allowing researchers to implement frameworks in empirical settings and observe, document, and make changes based on the results [47]. This specific type of research allows researchers active involvement and insights to the experiment that otherwise may not be observed by external researchers [49]. Reference [50] was the first who coined the term "action research," while [51] introduced the term "participant action research" as a method to diagnose a problem, create an action plan, and implement it to solve the problem in collaborative ways between the researcher and the organization's system.

Reference [52] expanded on the applicability of action research by designating it as a collaborating method for addressing an organization's practical issues and the researcher's scientific goals via the use of ethical framework. Reference [53] formulated the key characteristics of action research by defining it as future oriented, collaborative, implying systems development, generating theory grounded in action, agnostic, and situational. They proposed that action research can be conducted as a cyclical process in five phases: diagnosing, action planning, action taking, evaluating, and specifying learning.

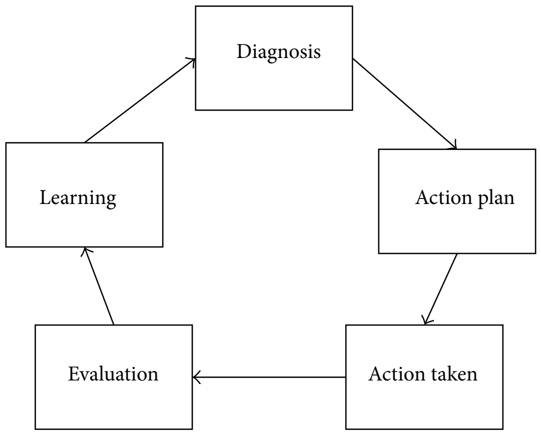


Figure 2: Adopted participatory action research strategy.

During the diagnosing stage, the researcher and participants identify an organizational problem and settle on a framework that will be used to address it. Next, the organization designates a team and the intervention as part of the planning stage. Research occurs during the action taking phase, followed by data collection, analysis, and summarization during the evaluating phase. In the final phase, specific lessons are documented for improvement during the following cycles. According to [54], structure and rigor should be kept in every phase.

In this study, the testing of the framework followed the same PAR strategy and stages (Figure 2).

4.2. Project Background. The unit of analysis for this study was a two-year period of participation by one of the authors following the five phases of action research. The author, a certified PMI project management professional, was assigned as the leading project manager for the implementation of an ERP system at a large county in the southern quadrant of the United States between February 2010 and June 2012. The ERP system, a non-US developed system, was procured in 2007 with the goal of replacing an unsupported and rapidly aging system, which lacked vendor support and was entirely maintained by the county's IT department. The goal of the new system was to improve the department's business operations by delivering customized business workflows and streamlined data access for over 200 employees with fully dedicated support provided by the software vendor. The new system also included a new public web portal to handle citizen inquiries as well as an integrated voice response (IVR) system for permits and inspections status via phone and fax requests.

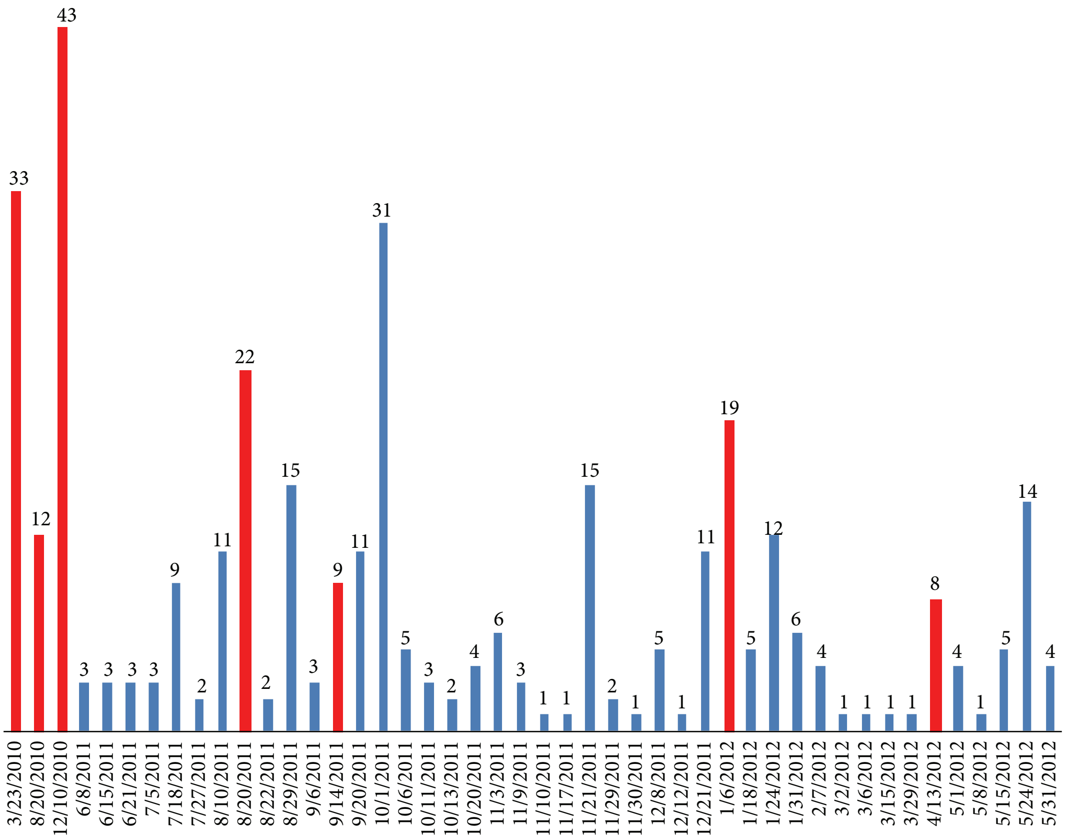


Figure 3: Resolved bugs and corresponding releases (red = major releases; blue = minor release).

It was heavily customized to fit the county's business processes. Results for the success of the framework were determined by measuring the number of releases and release types, reported and resolved software bugs, number of system crashes, average release acceptance durations, and average installation durations prior to and after intervention of each cycle. The data was evaluated using tables, graphs, and charts.

The project implementation was divided into three phases. Phase 1, completed in 2008, included the delivery of the code enforcement module. Phase 2 included the planning and engineering modules and was delivered in 2009. Schedule delays and lack of acceptance testing during these phases resulted in the accumulation of a number of software bugs. The release management process was performed ad hoc with the vendor delivering the final module resulting in the customer accepting it into production without rigorous testing. The final phase, which included the permitting and inspections module, the public web portal, and the IVR system, was completed in 2011. This study covered the twenty-seven-month time frame required to implement the final phase and resolve accumulated software bugs from prior phases via the use of the ITIL release management process.

At the time of the study, the system had accumulated 45 major unresolved software bugs from the prior phases. Some of these errors caused six system crashes with significant user down time. Furthermore, critical functionality was missing on already delivered components. As a result, the project was halted until the vendor resolved the system performance issues and delivered the missing functionality. The author and the project team performed an analysis of the existing release management processes based on the ITIL framework and identified the following critical problem areas that were addressed in two action research cycles: (1) no existing release acceptance criteria; (2) inefficient customer roll-out planning process; (3) inefficient distribution and installation; (4) no existing release planning process in place; and (5) no existing release policy in place to address the process of release management.

5. Results and Discussion

Between March 2010 and June 2012, one of the researchers, the project team, and the software vendor worked to resolve 361 software bugs via seven major and 39 minor releases. Figure 3 shows the number of software bugs resolved, the type of release, and the date of the releases. Major releases are defined as those that included more than one customer resource and took more than three man hours to complete (not including vendor hours). Minor releases included not more than one customer resource that took no longer than three man hours to complete.

The following section identifies the action research cycles, identified problems, and the application of the framework to resolve these problems.

5.1. Cycle 1

5.1.1. Problem 1: No Release Planning

(1) Diagnosis. No release planning existed between the vendor and the customer. Releases were performed ad hoc with the vendor primarily setting expected delivery dates without customer requested software fixes. There were no release acceptance criteria specified, roll-out and back-out plan, or release notes provided to customers prior to the release. The customer did not have a designated release manager to coordinate the resources. No user acceptance testing was performed prior to the release delivery.

(2) Action Plan. Develop a release plan that includes specific roles and responsibilities of the CRM, user acceptance team, content of each release, and acceptance criteria. Identify impacted end-users and address requests for change (RFC). Plan the upcoming release scope, schedule, cost, resources, acceptance tests, roll-out, training, distribution, and installation strategy. Coordinate plans with the vendor's release manager. Document lessons learned.

(3) Action Taken. The release plan was developed by the CRM with feedback from the vendor, the customer's IT department, and the business unit stakeholders. Resources, costs, and overall release strategy were addressed before commencement of the release. Change requests were documented in an RFC document, tracked, and communicated with all internal and external stakeholders.

(4) Evaluation. The release plan went into effect immediately after it was accepted by both the customer and vendor. The vendor provided the CRM with release notes for the upcoming release. The notes were used as the agenda for discussion between the CRM, the newly formed customer user testing team, and internal stakeholders. The group determined that some of the proposed fixes were not of significant importance to the users, while the ones that were had not been considered for release. A decision was made to create a bug prioritization scale. All software bugs reported by the users were prioritized via vote by all stakeholders and communicated to the vendor. The scheduled release was cancelled. Instead, a new release was planned to address the proposed high impact bugs.

(5) Learning. This cycle ended with the creation of a new process for software bug prioritization, which became the driving factor for subsequent release planning processes. Stakeholders voted on what key software bugs needed to be addressed. A release schedule determined start and end dates and informed management of when key customer resources were required to participate in the testing process and therefore could not be absent. The release plan also provided the vendor team with expected deliverables.

5.1.2. Problem 2: No Release Planning

(1) Diagnosis. Since no release acceptance criteria existed between the customer and the vendor, releases were not rigorously tested by the customer and no conditions were specified when a release should or should not be accepted. The vendor relied exclusively on its internal fit-for-testing process prior to delivery and installation of the release into the customer's production environment. As a result, a number of bugs appeared after release that were related to the differences in the vendor versus live customer environments. Furthermore, because the vendor team was not testing the application in the manner it was utilized by end-users, the total count of software bugs discovered in production continued to climb, while the production system frequently crashed two to three times per week for a period of one month due to performance problems.

(2) Action Plan. Determine the cause and resolve the performance issue leading to system crashes. Develop a rigorous user acceptance testing process that includes the development and execution of test cases for each of the application's business functionality, prioritization, and reporting of each bug discovered during testing, retesting of fixed software, and a go/no go approval decision for the acceptance of the tested release. Manage the deployment of release into test environment, the acceptance testing, and communications of issues with vendor release manager. Track reported changes and communicate often the progress with stakeholders. Document lessons learned.

(3) Action Taken. The performance issue was addressed first. The CRM and the UAT team prepared several test cases and distributed them to all the 150 application users. At a specified date and time, the organization ceased business operation, and, for 30 minutes, all users executed the test cases in the live production environment while the CRM and the vendor team monitored and recorded the performance of the system. The test yielded enough information to pinpoint a memory leak occurring with the live system. The vendor began working on a fix to address the memory leak. A new test environment was created to match the live production environment. The CRM worked with the UAT team to develop test cases based on the departmental business process maps. When the emergency release was deployed to the test environment, the UAT team executed the test cases and discovered additional bugs that were able to create other system performance issues. The release was also tested for memory leaks.

(4) Evaluation. The creation and execution of test cases and acceptance criteria added 40 days to the final schedule. As a result, the release schedule was adjusted to incorporate sufficient lead time in the future to allow UAT, resolution, and retesting.

(5) Learning. The release acceptance process was addressed first because of the urgency to stop accepting more bugs in production that could impact normal business operations. The production testing of the system proved successful in discovering and eliminating the performance related issue. The business unit management was convinced of the need for rigorous acceptance testing before production deployment and dedicated additional users to participate in the UAT. The UAT team uncovered additional bugs with each new release

Table 3: Release details.

Release date	Release type	Release version	Bugs discovered
3/23/2010	Major	3.3.3	17
6/8/2011	Minor	3.8.4.11	68
7/18/2011	Minor	3.8.4.12	77
8/20/2011	Major	3.8.8	62
8/29/2011	Minor	3.8.8	51
9/20/2011	Minor	3.8.9	68
1/6/2012	Major	3.8.11	26

(Table 3). These were prioritized and resolved in subsequent releases.

5.1.3. Problem 3: No Release Policy

(1) Diagnosis. No release policy existed prior to the cycle. Releases were performed ad hoc with the vendor primarily setting expected delivery dates without being agreed upon software fixes. There were no release acceptance criteria specified, roll-out and back-out plan, or release notes provided to customers prior to the release.

(2) Action Plan. Determine optimal policy for the organization with internal stakeholders. Propose, negotiate, and implement a release policy with the vendor that defines acceptable schedule, deliverables for each release, definition of major (merge build) versus minor (point) releases, test and production environments used for each release, roll-out plan, roll-back plans in case of issues with the release, roles, responsibilities, escalation steps, procurement needs, and contacts for vendor and customers.

(3) Action Taken. The release policy was developed with feedback from the vendor, the customer's IT department, and the business unit users. The final policy was negotiated with the vendor via the customer's procurement department and was incorporated into the maintenance agreement of the contract.

(4) Evaluation. The release policy included provisions for unlimited telephone support services concerning the use of the software. Prior to the policy, bugs were reported via email or restricted for reporting to specific times. The policy specified criteria for response to all errors found in the software within a defined timeframe. The policy specified technical documentation that the vendor should provide for each subsequent update, revision, release, or new version of the software. The policy identified an installation process by the vendor of each update, revision, release, or new version of the software. This included installation in the customer's test environment of every release, customer testing criteria, and vendor deployment in the live production environment. Additionally, the release policy outlined a semiannual review of customer's software and database by the vendor to

(i) uncover any patterns and recommend process refinements;

(ii) propose improvements to the database backup logs and procedures;

(iii) run performance checks on the application server to check for memory, CPU, and disk capacity issues with recommended configuration refinements;

(iv) review primary processes with customer's representative from each department to help identify bottlenecks in processes; and

(v) review application configuration to identify administrative errors and provide written documentation of all findings and recommendations.

(5) Learning. The new release policy was formulated with feedback from the customer's IT department and the business unit management. This joint collaboration between the two groups contributed to the documentation of technical and operational considerations necessary for inclusion into the release policy. The final product was a broad release policy that focused on both technology and business process improvements in order to minimize customer productivity due to system performance or software bugs introduced with each release. The policy was negotiated with the vendor over a period of several months.

5.2. Cycle 2

5.2.1. Problem 1: Inefficient Customer Roll-Out Planning

(1) Diagnosis. There was minimal involvement from end-users during the roll-out planning process. A list of tasks for the delivery and installation was prepared by the vendor; however, coordination of customer IT resources for the back-out procedures was frequently omitted or handled at the last minute.

(2) Action Plan. Develop a release time table that clearly outlines

(i) who is involved from each team;

(ii) what tasks are delegated to each participant;

(iii) what are the contact phone numbers and email addresses for each person;

(iv) what is the conference bridge number which members can use to join and remain on for the duration of the installation;

(v) what is the start and end time for each task on the release;

(vi) who and what testing will be performed in the production system; and

(vii) what is the roll-out procedure plan in case the installation fails?

Distribute the time table and host a preroll-out meeting with all stakeholders involved with the release to address questions and provide clarifications. (Integrating) Identify stakeholders involved in the roll-out (e.g. IT, key end-users, and vendor resources). Track and communicate changes and document lessons learned.

(3) Action Taken. The CRM developed a detailed release checklist and provided it to all stakeholders during the preroll-out meeting. Tasks included

- (i) names of servers to be updated,
- (ii) dates, times, and responsible party for
    - (1) release notifications to the entire user pool;
    - (2) disabling of the production system;
    - (3) server snapshots;
    - (4) primary and secondary database backups;
    - (5) database restores;
    - (6) installations of the new release by vendor;
    - (7) UAT testing in production;
    - (8) desktop client roll-out to all users' PC workstations; and
    - (9) certification of release completion.

During the preroll-out meeting, the CRM informed the stakeholders of their role in the release, including the fact that he would be calling out the completion of each task during the installation and letting each party know when to start their respective tasks. Next, the CRM called out each participant and asked them to read out their task. He solicited participants' feedback about the task and addressed any inconsistencies or misunderstandings. At the end of the preroll-out meeting, each stakeholder was made aware of the expectations of his or her role, including start and end times and contact information for all participants and the CRM. The UAT team was also informed of the test cases to be executed in the production environment after the completion of installation.

(4) Evaluation. The use of a release checklist with start and end times, designated stakeholders, and solicitation for feedback from each participant helped the team to minimize confusion. By asking the stakeholders to provide their understanding of each task, the CRM was able to evaluate the roll-out plan, address any inconsistencies, and update the release checklist where necessary.

(5) Learning. The preroll-out meeting served as a dry run for the real installation similar to the training and visualization techniques used by sportsmen before a competition. This exercise also helped the CRM to determine tasks that were missed in the initial release planning.

5.2.2. Problem 2: Inefficient Distribution and Installation

(1) Diagnosis. Prior to the study, the delivery and installation process frequently experienced a number of issues due to the lack of coordination between the vendor and customer. On one occasion, the vendor account permissions were not enabled due to lack of communication between the vendor and the customer IT department. This caused a delay of the installation until the security team activated the vendor's account and elevated its permissions to allow installations on the production server. On another occasion, the vendor failed to provide the software installation files. When the files were made available for download from the vendor's FTP site, the downloading process took over two hours to complete, which delayed the start of the installation. During the installation, production server security updates commenced, resulting in further delays due to lack of coordination between the customer IT team and the vendor. As a result, the release was delivered over a period of 16 hours during the weekend. Similar issues caused other installations to also last over entire weekends. Furthermore, on the next business day, end-users would discover configuration issues with the application that could have been avoided by performing production testing after the installation.

(2) Action Plan. Ensure the installation process follows a roll-out checklist with specific roles, time table for tasks, details about vendor account permissions, and disabling of backups and security updates on the servers. Ensure installation files are provided by the vendor at least one day prior to the installation. Ensure an open channel of communication is established between all stakeholders, including the vendor at all times during the installation to address any unforeseen issues that may occur. Ensure the customer release team is available at all times during the delivery and installation process. Ensure the release is tested by the UAT team after the installation in the production environment. Document any lessons learned.

(3) Action Taken. During the preroll-out meeting, all release participants were made aware of their specific tasks and roles associated with the upcoming installation. During the delivery, a conference bridge was made available to all stakeholders. The CRM used the bridge to obtain status on each specific task and direct the execution of the next task on the list. All installers were downloaded from the vendor site prior to the installation. Regular backups were disabled from the production server at the start of the installation. Production testing was performed immediately after the installation to ensure that no errors with the release existed in the live system.

(4) Evaluation. The first delivery and installation that used a roll-out checklist were completed within six hours of initiation. There were no issues reported during the installation, but three issues were discovered by the UAT team during production testing. The issues were related to server configuration settings and were resolved by the vendor. The CRM certified the release as successful at the end of the sixth hour and informed all stakeholders. During the next business day, the CRM and the UAT team responded to calls from users for general information on the application. No installation issues were reported by the end-users.

(5) Learning. The use of a time table, checklist with tasks, a conference bridge for communication, leadership and direction on when a task was completed, and when a task should be started, eliminated stakeholder confusion. Production system uptime data was collected for normal business hours to

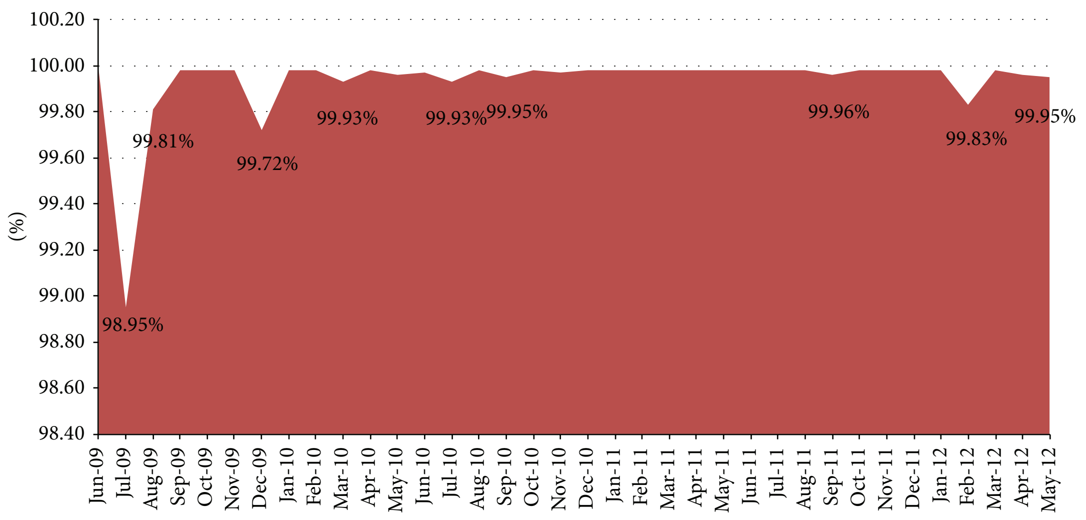


FIGURE 4: Production system uptime.

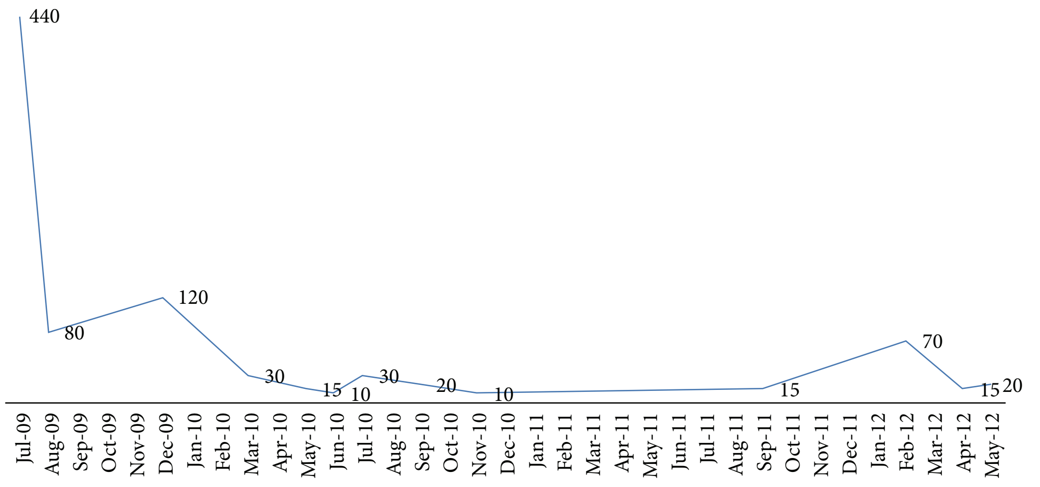


FIGURE 5: Actual system downtime (minutes per month).

determine release successes based on the newly delivered builds. As a result of the improvements, uptime of the production server improved by an average of 0.15% or nearly one hour and five minutes per month compared to the period prior to the changes (Figure 4), while system downtime was reduced from 440 minutes in July of 2009 to only ten minutes in May of 2012 (Figure 5).

6. Conclusions

This appears to be the first study that proposes an integrated release management framework with a predefined CRM role. The paper presents many opportunities for future research on the release management process. For example, in release management, it would be valuable to examine the impact on software quality via use cases written by the customer team and used by the vendor software development team during the software coding process. This process was not investigated in this case but can prove valuable in order to understand not only how use cases impact software development but also how the interaction between the customer UAT team and the vendor's software development team can work together better to produce higher quality software.

This research has begun the investigation of how the ITIL framework can be applied from the customer's perspective within an organization. By viewing the impact of the release management process through the lens of the proposed integrated release management framework, it illustrates how release management can be accomplished electively through establishing coordination methods by the CRM with the vendor release manager and how problems are resolved through the centralization of the activities. Future research can examine the use of social media systems to capture lessons learned during releases [55], optimization of on-the-job learning practices for release teams [56], and customer privacy protection when sharing issues with vendors [57].

Little research exists on the application of project management concepts for release management from the customer's perspective. Frameworks like ITIL, COBIT, and ISO

20,000 examine software release from a single perspective, most commonly the vendor's. Further research through a multicase study approach on the use of the proposed integrated release framework and the results achieved with it could answer questions on its advantages to other systems across a broad range of industries. This research will hopefully set the stage for investigating these and other important questions and for more effective and efficient release management processes.

The ITIL framework holds great promise for leveraging a structured method of delivery and acceptance of release; however, the lessons from this case study show that if the framework is only applied from the vendor's perspective, the customer can incur a number of disadvantages. As a result, the proposed framework addresses the gaps that exist in literature to address the release management process from the customer's perspective. Whether or not these lessons can be duplicated in other scenarios is an open question, as each organization offers unique challenges.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] S. Cleveland and T. J. Ellis, "Toward a model for customer-driven release management," in *Proceedings of the 19th Americas Conference on Information Systems (AMCIS '13)*, pp. 3570–3577, August 2013.

[2] S. K. Sahoo, J. Criswell, and V. Adve, "An empirical study of reported bugs in server software with implications for automated bug diagnosis," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE '10)*, vol. 1, pp. 485–494, Cape Town, South Africa, May 2010.

[3] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance computing systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 337–350, 2010.

[4] E. Cloud, *Electric Cloud Survey: 58% of Software Bugs Result from Test Infrastructure and Process, not Design Defects*, 2012, http://www.electric-cloud.com/news/2010-0602.php.

[5] G. E. Bryan, "Not all programmers are created equal—Redux," in *Proceedings of the IEEE Aerospace Conference*, pp. 1–10, March 2012.

[6] G. Tassey, "The economic impacts of inadequate infrastructure for software testing," RTI Project 7007, National Institute of Standards and Technology, Gaithersburg, Md, USA, 2002.

[7] T. Zimmermann, N. Nagappan, P. J. Guo, and B. Murphy, "Characterizing and predicting which bugs get reopened," in *Proceedings of the 34th International Conference on Software Engineering (ICSE '12)*, pp. 1074–1083, June 2012.

[8] J. Tucek, S. Lu, C. Huang, S. Xanthos, and Y. Zhou, "Triage: diagnosing production run failures at the user's site," in *Proceedings of the 21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP '07)*, pp. 131–144, Stevenson, Wash, USA, October 2007.

[9] A. Arora, J. P. Caulkins, and R. Telang, "Research note—sell first, fix later: impact of patching on software quality," *Management Science*, vol. 52, no. 3, pp. 465–471, 2006.

[10] A. Bertolino, "Software testing research: achievements, challenges, dreams," in *Proceedings of the Future of Software Engineering (FoSE '07)*, pp. 85–103, May 2007.

[11] O. Taipale, K. Smolander, and H. Kalviainen, "Cost reduction and quality improvement in software testing," in *Proceedings of the International Conference on Software Quality Management*, p. 63, 2006.

[12] A. Pande, "Innovations in cloud computing: the way ahead," *International Journal of Multidisciplinary Research*, vol. 2, pp. 197–205, 2012.

[13] J. Hota and S. Mishra, "Implementation of ERP SaaS option for HRIS reporting practices," in *Proceedings of the International Conference on Technology and Business Management*, p. 28, 2012.

[14] B. Aslan, M. Stevenson, and L. Hendry, "An assessment of the applicability of enterprise resource planning systems to make-to-order companies," in *Proceedings of the 6th European and Mediterranean Conference on Information Systems (EMCIS '09)*, July 2009.

[15] Z. Jiang, S. Sarkar, and V. S. Jacob, "Postrelease testing and software release policy for enterprise-level systems," *Information Systems Research*, vol. 23, no. 3, pp. 635–657, 2012.

[16] W.-G. Tan, A. Cater-Steel, and M. Toleman, "Implementing it service management: a case study focussing on critical success factors," *Journal of Computer Information Systems*, vol. 50, no. 2, pp. 1–12, 2009.

[17] J. Zeng, "A case study on applying ITIL availability management best practice," *Contemporary Management Research*, vol. 4, pp. 321–332, 2008.

[18] J. van Bon, *IT Service Management Guide*, Addison-Wesley, New York, NY, USA, 2002.

[19] J. Van Bon, G. Kemmerling, and D. Pondman, *IT Service Management: An Introduction*, Van Haren Publishing, San Antonio, Tex, USA, 2002.

[20] A. S. Danesh, M. R. Saybani, and S. Y. S. Danesh, "Software release management challenges in industry: an exploratory study," *African Journal of Business Management*, vol. 5, no. 20, pp. 8050–8056, 2011.

[21] D. Greer and G. Ruhe, "Software release planning: an evolutionary and iterative approach," *Information and Software Technology*, vol. 46, no. 4, pp. 243–253, 2004.

[22] M. Michlmayr, F. Hunt, and D. Probert, "Release management in free software projects: practices and problems," in *Open Source Development, Adoption and Innovation*, vol. 234, pp. 295–300, Springer, Berlin, Germany, 2007.

[23] A. Lahtela and M. Jäntti, "Challenges and problems in release management process: a case study," in *Proceedings of the IEEE 2nd International Conference on Software Engineering and Service Science (ICSESS '11)*, pp. 10–13, July 2011.

[24] A. van der Hoek and A. L. Wolf, "Software release management for component-based software," *Software—Practice and Experience*, vol. 33, no. 1, pp. 77–98, 2003.

[25] H.-M. Sihvonen and M. Jäntti, "Improving release and patch management processes: an empirical case study on process challenges," in *Proceedings of the 5th International Conference on Software Engineering Advances (ICSEA '10)*, pp. 232–237, Nice, France, August 2010.

[26] K. Rautiainen, C. Lassenius, J. Vahaniitty, M. Pyhajarvi, and J. Vanhanen, "A tentative framework for managing software

product development in small companies," in *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS '02)*, pp. 3409–3417, 2002.

[27] M. Kajko-Mattsson and P. Meyer, "Evaluating the acceptor side of EM3: release management at SAS," in *Proceedings of the International Symposium on Empirical Software Engineering*, pp. 315–324, November 2005.

[28] K. Levin and O. Yadid, "Optimal release time of improved versions of software packages," *Information and Software Technology*, vol. 32, no. 1, pp. 65–70, 1990.

[29] M. Moreira, Wrangling a Release: The Role of Release Manager, 2010, http://www.cmcrossroads.com/article/wrangling-release-role-release-manager.

[30] A. Baethge and T. Rigotti, "Interruptions to workflow: their relationship with irritation and satisfaction with performance, and the mediating roles of time pressure and mental demands," *Work and Stress*, vol. 27, no. 1, pp. 43–63, 2013.

[31] G. Ballintijn, "A case study of the release management of a health-care information system," in *Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM '05)*, pp. 34–43, Budapest, Hungary, September 2005.

[32] B. Potgieter, J. Botha, and C. Lew, "Evidence that use of the ITIL framework is effective," in *Proceedings of the 18th Annual Conference of the National Advisory Committee on Computing Qualifications*, pp. 160–167, Tauranga, New Zealand, 2005.

[33] P. Elephant, *ITIL IT Service Management Essentials*, Pink Elephant, Burlington, Canada, 2006.

[34] G. Rasa, S. J. Kumar, and D. R. Banu, "Release and deployment management using ITIL," *Global Journal of Computer Science and Technology*, vol. 10, no. 15, 2010.

[35] G. Ruhe and M. O. Saliu, "The art and science of software release planning," *IEEE Software*, vol. 22, no. 6, pp. 47–53, 2005.

[36] P. Carlshamre, "Release planning in market-driven software product development: provoking an understanding," *Requirements Engineering*, vol. 7, no. 3, pp. 139–151, 2002.

[37] J. Momoh and G. Ruhe, "Release planning process improvement—an industrial case study," *Software Process Improvement and Practice*, vol. 11, no. 3, pp. 295–307, 2006.

[38] D. Connolly, F. Keenan, and F. M. Caffery, "Acceptance test-driven development by annotation of existing documentation," in *Proceedings of the European Systems & Software Process Improvement and Innovation Conference (EuroSPI '10)*, 2010.

[39] C. Ieamsaard and Y. Limpiyakorn, "On integrating user acceptance tests generation to requirements management," in *Proceedings of the International Conference on Information Communication and Management*, pp. 248–252, 2011.

[40] C.-Y. Huang and M. R. Lyu, "Optimal release time for software systems considering cost, testing-effort, and test efficiency," *IEEE Transactions on Reliability*, vol. 54, no. 4, pp. 583–591, 2005.

[41] N. Fenton, M. Neil, W. Marsh et al., "Predicting software defects in varying development lifecycles using Bayesian nets," *Information and Software Technology*, vol. 49, no. 1, pp. 32–43, 2007.

[42] S. K. Donohue and J. B. Dugan, "Is my software "good enough" to release?—a probabilistic assessment methodology," in *Proceedings of the 29th Annual IEEE/NASA Software Engineering Workshop (SEW '05)*, pp. 5–13, Greenbelt, Md, USA, April 2005.

[43] S. U. Farooq and S. M. K. Quadri, "Evaluating effectiveness of software testing techniques with emphasis on enhancing software reliability," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 2, no. 12, pp. 740–745, 2011.

[44] J. R. Erenkrantz, "Release management within open source projects," in *Proceedings of the 3rd Open Source Software Development Workshop*, pp. 51–55, 2003.

[45] M. Ramakrishnan, "Software release management," *Bell Labs Technical Journal*, vol. 9, no. 1, pp. 205–210, 2004.

[46] PMI, *Guide to the Project Management Body of Knowledge (PMBOK)*, Project Management Institute, White Plains, Md, USA, 2008.

[47] D. Grant and O. Ngwenyama, "A report on the use of action research to evaluate a manufacturing information systems development methodology in a company," *Information Systems Journal*, vol. 13, no. 1, pp. 21–35, 2003.

[48] A. McIntyre, *Participatory Action Research*, Sage, Thousand Oaks, Calif, USA, 2008.

[49] D. Yates and S. Paquette, "Emergency knowledge management and social media technologies: a case study of the 2010 Haitian earthquake," *International Journal of Information Management*, vol. 31, no. 1, pp. 6–13, 2011.

[50] K. Lewin, "Action research and minority problems," *Journal of Social Issues*, vol. 2, pp. 34–46, 1946.

[51] I. Chein, S. W. Cook, and J. Harding, "The field of action research," *The American Psychologist*, vol. 3, no. 2, pp. 43–50, 1948.

[52] R. N. Rapoport, "Three dilemmas in action research with special reference to the Tavistock experience," *Human relations*, vol. 23, pp. 499–513, 1970.

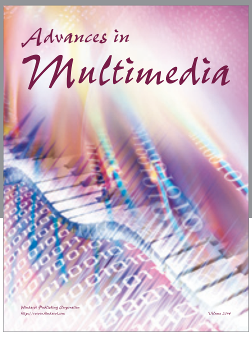
[53] G. I. Susman and R. D. Evered, "An assessment of the scientific merits of action research," *Administrative Science Quarterly*, vol. 23, no. 4, pp. 582–603, 1978.

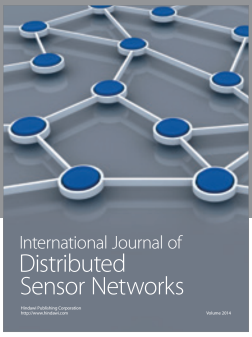
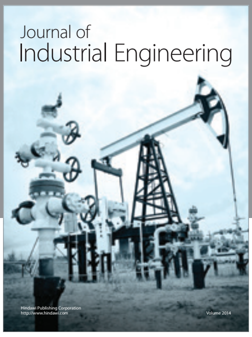
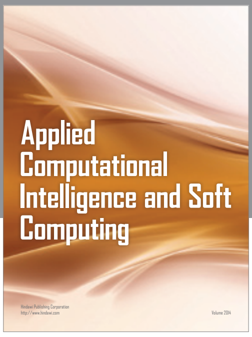
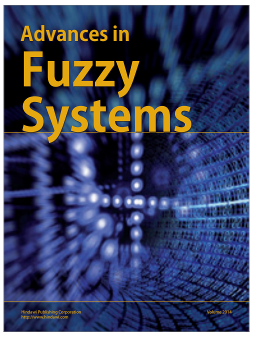
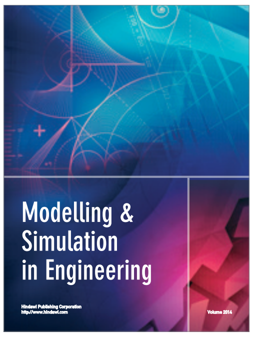
[54] D. DeLuca, M. J. Gallivan, and N. Kock, "Furthering information systems action research: a post-positivist synthesis of four dialectics," *Journal of the Association of Information Systems*, vol. 9, no. 2, pp. 48–72, 2008.

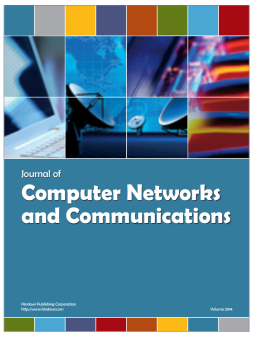
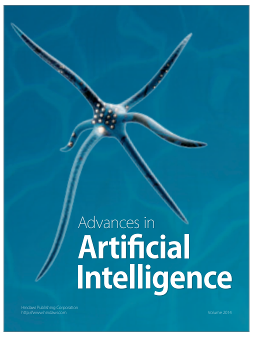
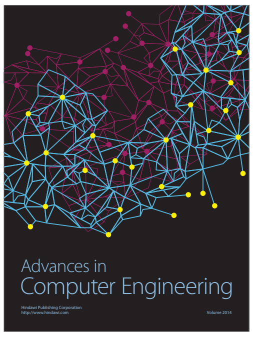
[55] S. Cleveland, "Using microblogging for lessons learned in information systems projects," in *Proceedings of the 7th International Research Workshop on Information Technology Project Management (IRWITPM '12)*, Orlando, Fla, USA, 2012.

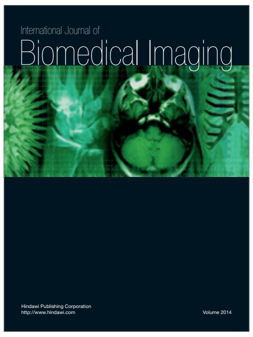
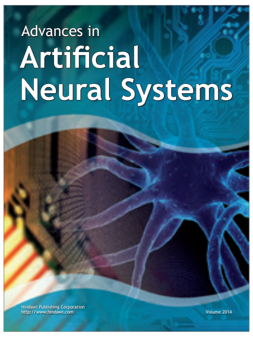
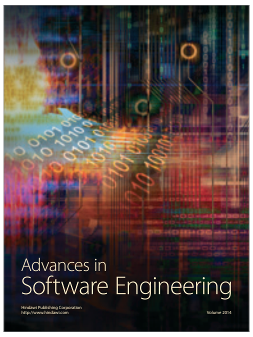
[56] S. Cleveland, "On-the-job informal learning practices for IS students," in *Proceedings of Southern Association for Information Systems Conference*, Savannah, Ga, USA, 2013.

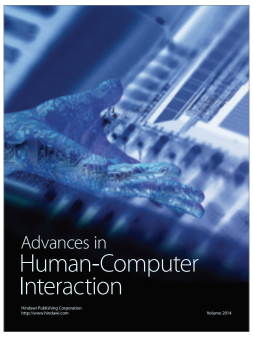
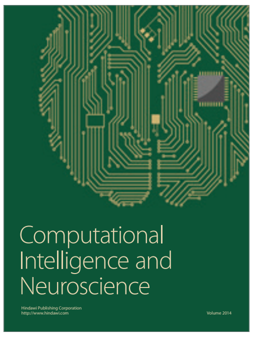
[57] S. Cleveland, "In search of user privacy protection in ubiquitous computing," in *Proceedings of the IEEE 13th International Conference on Information Reuse and Integration (IRI '12)*, pp. 694–699, August 2012.


Advances in Multimedia

The Scientific World Journal

International Journal of Distributed Sensor Networks

Journal of Industrial Engineering

Applied Computational Intelligence and Soft Computing

Advances in Fuzzy Systems

Modelling & Simulation in Engineering


Journal of Computer Networks and Communications

Advances in Artificial Intelligence

Advances in Computer Engineering

International Journal of Computer Games Technology

International Journal of Biomedical Imaging

Advances in Artificial Neural Systems

Advances in Software Engineering

Journal of Robotics

Advances in Human-Computer Interaction

Computational Intelligence and Neuroscience

International Journal of Reconfigurable Computing

Journal of Electrical and Computer Engineering