Presented to the Interdisciplinary Studies Program:

UNIVERSITY OF OREGON
APPLIED INFORMATION MANAGEMENT

Applied Information Management
and the Graduate School of the
University of Oregon
in partial fulfillment of the
requirement for the degree of
Master of Science

The Role of Testers in an Agile Software Development Life Cycle within B2B Companies

CAPSTONE REPORT

Dwayne Thomas
Statistical Data Quality Assurance Analyst
Rentrak Corporation

University of Oregon
Applied Information
Management
Program

February 2013

Continuing Education
1277 University of Oregon
Eugene, OR 97403-1277
(800) 824-2714

Approved by

_____

Dr. Linda F. Ettinger
Senior Academic Director, AIM Program

The Role of Testers in an Agile Software Development Life Cycle within B2B Companies

Dwayne Thomas

Rentrak Corporation

Abstract

High software quality is a very important outcome of software development practices for business customers (Mairiza, Zowghi, & Nurmuliani, 2010). This annotated bibliography is developed for software testers who want to improve the quality of software and customer satisfaction in the Agile development cycle. Selected references published between 2006 to 2013 are reviewed to examine software quality requirements, appropriate amounts of software tester readiness, test planning, verification of business test cases, and additional testing activities.

Keywords: Agile, software quality, testing, business-to-business

Table of Contents

List of Figures

List of Tables

Introduction

Problem

High software quality is a very important outcome of software development practices for business customers (Mairiza, Zowghi, & Nurmuliani, 2010), who often desire quality of service guarantees (Filieri, Ghezzi, & Tamburrelli, 2012). Blaine and Cleland-Huang (2008) define software quality as the product's value to its stakeholders. Mairiza, Zowghi, and Nurmuliani (2010) claim that nonfunctional requirements are often more critical than individual functional requirements (FRs) in the determination of a software's perceived success or failure. The authors note that functional requirements are specific requirements by business customers. Non-functional requirements (NFRs) entail *how* the specific requirements by the business customers are delivered: specific NFRs are particular to software application however they often include items such as software performance, usability, integrity, interoperability, security, safety, and testability.

Software development companies are tempted to rely on meticulous programming to produce quality software that meets software functional requirements (Roseberry, 2012). For example, one of the more vibrant discussion threads on the professional network Linked-In asks, "if you want better software quality, [should you] get rid of the testing department" (William, 2012)? However, Roseberry (2012) and Beck (*InfoWorld*, 2007) suggest that software project team members are beginning to realize that meticulous programming [alone] frequently does not produce high quality software and is a dangerous practice of software development. Ahamed (2009) notes that for every

100 lines of well-written code there are typically one to three defects. This annotated bibliography refers to all the members of the software project teams as *software developers*; they all should help at the various stages of the development projects (Crispin & Gregory, 2009).

Agile software development methods have successfully evolved to produce high quality software on short software development cycles (Ambler, 2007 & *InfoWorld*, 2007). Agile software development consists of completing software development iterations in several weeks, as opposed to six months to a year (*InfoWorld*, 2007). Agile software development life cycles (SDLC) deliver small chunks of business value in extremely short release cycles (Crispin & Gregory, 2009, p. 3). *Business to business* (b2b) relationships, which by definition include business customers (Wikipedia, 2012), have been a beneficiary of the agile software development approach (Bavani, 2012). Without agile development processes, any software development company might be devoured by its competitors (Watkins, 2009).

A large share of the quality of the software produced by the software developers in agile SDLC teams necessarily hinges on software team members who fulfill the role of software quality testers (Crispin & Gregory, 2009, p. 6). Software testers execute software to evaluate the functional and non-functional requirements of the software. Testers use specific sets of data, called test cases, which the software must handle appropriately to be considered successful (Ambler, 2007). Software testers also develop high quality software from doing many interesting applications of the software, termed *exploratory testing* (Crispin & Gregory, 2009; Roseberry, 2012).Unlike the engineers for a car for example, software programmers rarely get a chance to directly use their products (Spillner, Linz, & Schaefer, 2011). Software testers are often titled as Software Quality Assurance Analysts.

Quality assurance is the set of activities testers perform, which includes test cases, that give stakeholders confidence that software [at least] meets the specified business requirements (CSTE, 2006).

Citing an authoritative automotive quality manufacturer's text (W. Edward Demings's Out of Crisis, 1986) technology consultant Matt Heusser recommends that testers study software development as a system, visualize the process, and suggest methods for improvement that are grounded in the nature of the work at their business (Heusser, 2012). In the journal *Information & Software Technology* Andrade et al. (2013) also note that software testing has a large intellectual capital component. According to Crispin and Gregory (2009), the main function of testers should be exploratory testing, which uses the tester's understanding of the system, along with critical thinking, to define focused, experimental tests, which can be run and documented in short time frames (Crispin & Gregory, 2009; Heusser & Kulkarni, 2011).

Purpose

Mauldin, Nicolaou, and Kovar (2006) suggest that testers should be at least as important as programmers in b2b software design life cycles. The purpose of this annotated bibliography is to identify literature that examines the role of the software tester (Spillner, Linz, & Schaefer, 2011) in an agile software development life-cycle (Ambler, 2007) of Business-to-Business (b2b) relationships (Wikipedia, 2012, Business-to-Business). The goal is to develop a set of suggested practices for agile software testers who work in b2bs. As suggested by Heusser (2012), each tester must decide which practices are best for his or her own workplace. However, this study avoids

discussing software automation usability testing tools because though these tools are often an important part of any testing operation, the tools are only valuable as companions to in-person tests (Harty, 2011).

Significance

Technical debt is a term for software development tradeoffs, which satisfy current business, needs but that must be met with longer subsequent iterations to improve maintainability (Bavani, 2012). Software companies in b2b relationships are highly susceptible to accumulating technical debt because technical debt shortens the software delivery time to customers in the short run (Ambler, 2007). Barkley states that completing software correctly the first time is inexpensive; however, completing software incorrectly is costly (Heusser & Kulkarni, 2011). Crispin and Gregory (2009) suggest that testers explain to managers that accumulating technical debt subsequently reduces the ability of software development teams to deliver products (p. 487).

Because software can exhibit dynamic behaviors, it is very important that it is tested as a way to reduce behaviors that do not conform to the requirements (Spillner, Linz, & Schaefer, 2011). It is important for software professionals to note that testing does not prove the absence of faults; in practice, exhaustive tests are impossible because of the combination possibilities of software features and data inputs. Along with helping remove software defects, software tests increase confidence in the software system—although no complex software system is defect free (Spillner, Linz, & Schaefer, 2011).

Molinari, Abratt, and Dion(2008) find that business experts especially rely on product

satisfaction and interpersonal satisfaction when renewing their contracts. Mauldin, Nicolaou, and Kovar (2006) note that b2b experts are also more likely to recommend a service when they know that service receives ongoing quality assurance. The authors suggest redesigning existing assurance services to provide continuous assurance even over other factors, such as web site design.

Audience

Software-development companies are moving away from traditional long product development cycles because the long product development cycles do not lend themselves to high product quality (Harter, Krishnan, & Slaughter 2000). Traditional manual software testing requires laborious work to aid in the delivery of complete products or software updates and takes a relatively long period of time (Crispin & Gregory, 2009). This annotated bibliography is framed to especially help agile software testers who may not have the knowledge, skills, or the necessary tools to support software testing in more fast paced agile software development environments of business- to-business companies. This annotated bibliography focuses on suggested practices related to (a) the role of software testers in agile software teams in a b2b context; (b) appropriate amounts of software tester readiness and planning; and (c) testing activities in addition to verifying business test cases.

Research Questions

What is the role of the software quality tester in an agile development team, situated in a b2b context? The bibliography also raises sub-questions:

- “how much of the test plan should testers develop prior to receiving the testable code in the

agile environment (Crispin & Gregory, 2009)?"

- which activities, in addition to business test cases, might testers be responsible?

Delimitations

Focus. Three fields of information are combined in this annotated bibliography: (a) the role of software testers in agile software teams in a b2b context; (b) appropriate amounts of software tester readiness and planning; and (c) testing activities in addition to verifying business test cases. Each field is related to the software industry, but all three fields are rarely addressed concurrently.

Exclusion. This study avoids comprehensive examination of automated testing tools, which all need to be monitored by humans (Harty, 2011). This study does not address testing functions which could be assigned to any other member of the software-development team.

Timeframe. The references provided in this study, with few exceptions, are published between 2006 and 2013 in order to be most relevant to testing professionals. agile software development practices have especially achieved success within this timeframe (Ambler, 2007; *InfoWorld* 2007).

Sources. Reference selection relies on peer-reviewed academic journals and published commercial textbooks. As well, the professional literature from the Pacific Northwest Software Quality Conference (PNSQC) provides some cutting-edge and software industry vetted topics in the subject matter. Sources are cited as a credible based on features outlined by Bell and Frantz (2012): authority, objectivity, quality of work, coverage of work, and currency.

Reading and Organization Plan Preview

Reading plan. The in-depth reading of the references selected for use in this study is performed through conceptual analysis methods suggested by Busch et al. (2012). Conceptual analysis

searches for the existence of key concepts which in this case include: (a) the role of software testers in agile software teams in a b2b context; (b) appropriate amounts of software tester readiness and planning; and (c) testing activities in addition to verifying business test cases.

Organization plan. To support the construction of ideas in this annotated bibliography, the references in this annotated bibliography are organized thematically around the three research questions, rather than chronologically ("Literature reviews," 2013). This annotated bibliography offers nuanced perspectives on the themes and refers the reader to its references for further research.

Definitions

This annotated bibliography examines literature concerning software quality testing practices within an agile software-development team, developing for businesses in b-2-b contexts. The definitions of terms and phrases are derived from the selected literature so as to be meaningful to testing practitioners in similar development teams and business contexts.

Agile software development life cycles consist of completing software development iterations in several weeks as opposed to six months to a year (*InfoWorld*, 2007). Agile software development life cycles (SDLC) deliver small chunks of business value in extremely short release cycles (Crispin & Gregory, 2009, p. 3). Agile software development life cycles methods include: Scrum, Pinball, Extreme Programming (XP) and Dynamic System Development Method (DSDM) (Ambler, 2007).

Business-to-business (b2b) and business-to-consumer (b2c) relationships are two business structures that might require different methods for software improvement. Both structures facilitate the selling process of goods and services. However, while b2b products and services are sold from a business to a business expert, b2c products and services are sold from a business to a consumer (Wikipedia, 2012, Business-to-Business).

Code coverage are software coding processes that show exactly which parts of the product software code, expressed as source code, were hit during a software programmer's code check (Roseberry, 2012).

Customer teams include business experts, product owners, domain experts, product managers, business analysts, subject matter experts—everyone on the "business" side of a project. They provide the examples that will drive coding (Crispin & Gregory, 2009, p.7)**.**

Exploratory testing uses the tester's understanding of the system, along with critical thinking, to define focused, experimental "tests" which can be run and documented in short time frames (Crispin & Gregory, 2009; Heusser & Kulkarni, 2011).

Functional requirements are specific requirements by business customers. Non-functional requirements (NFRs) entail how the specific requirements are delivered: specific NFRs are important for a particular application however they include items such as software integrity, interoperability, performance, security, safety, usability, and testability (Mairiza, Zowghi, & Nurmuliani, 2010).

Non-functional requirements (NFRs) entail how the specific requirements are delivered: specific NFRs are important for a particular application. However, they include items such as software integrity, interoperability, performance, security, safety, usability, and testability (Mairiza, Zowghi, & Nurmuliani, 2010).

Quality assurance (QA) is the set of activities, which includes test cases, that gives stakeholders confidence that software meets the specified business requirements (CSTE, 2006).

Software developers are all software project team members. Software developers help deliver software functional and non-functional requirements to customers: including programmers, testers, system administrators, architects, database administrators, technical writers, security specialists and members who take on multiple activities (Crispin & Gregory, 2009, p.7).

Software development life cycles (SDLCs) are the processes that convert business experts' requirements to tangible software features. The agile software development life cycle is the particular form of SDLC emphasized by this literature review (CSTE, 2006).

Software testers execute software in order to compare its actual behavior to its expected behaviors. They help improve the quality of the software by helping remove some unexpected behaviors from the software (Spillner et al., 2011).

Test cases are specific sets of data, which the software must handle appropriately to be considered successful (Ambler, 2007).

Technical debt is the term for software programming design tradeoffs which satisfy current business needs but that must be met in subsequent design cycles to improve maintainability (Bavani, 2012).

Research Parameters

Ambler (2007) declares that agile software development methods have evolved to produce high quality software on short software development cycles; in response this annotated bibliography examines literature from 2007 through 2013. Specific research objectives aim at helping practitioners who need to hone their craft as agile development practices become more prominent in b2b settings (Crispin & Gregory, 2009). Mauldin, Nicolaou, and Kovar (2006) suggest that testers should be at least as important as programmers in b2b software design life cycles. The research parameters in this bibliography include: a search strategy report, key words, rreference collection procedures**,** a documentation approach, an evaluation criteria for selection of references, and full description of the reading and organizing plan.

Search Strategy

Searches for information draw from three main bodies of literature: (a) the role of software testers in agile software teams in a b2b context; (b) appropriate amounts of software tester readiness and planning; and (c) testing activities in addition to verifying business test cases.

Key Words

Key words used for searches in this study are collected from several sources: (a) terms found in peer-reviewed articles and journals, of the University of Oregon online business and computer science databases, and through the University of Oregon online portal at IEEE (technology leadership resource center) (b) Google Scholar, across many disciplines for literature; (c) online textbook reseller Amazon suggests textbooks relevant to this study, and (d) Wikipedia and LinkedIn provide some background of current software testing discussion topics.

The key words used in various combinations include:

- b2b quality
- Agile testing
- software productivity
- software quality
- lean software management
- software tester

Reference Collection Procedures

Qualifying references for this study cover the areas of software testing, agile software development, and b2b relationships in various combinations. The University of Oregon online library facilitates the download of most of the texts relevant to the study. Many of the texts are in portable digital file (PDF) format and readily available for in depth reading. Parallel searches are also done via Google Scholar, the online academic search site. Google Scholar indicates which texts are the most commonly cited sources. The most commonly cited sources are sometimes also guides for further investigation. The online textbook reseller Amazon suggests textbooks relevant to this study. QA conference papers from the PNSQC Conference of 2012 provide predictive snapshots of current QA discourses. Wikipedia and LinkedIn provide minimal but necessary professional background of additional topics.

Documentation Procedures

Zotero Google Chrome plugin and standalone programs suggested (Cheslack-Postava et al.,

2012) is helpful for documentation of most online sources helps to document most of the American Psychological Association (APA) bibliographic entries for this study. As well, the University of Oregon databases offers APA entries. First the abstract of a bibliographic help determine a source's for relevance to this study if the addresses the role of software testers in agile software teams in a b2b context. Next, the bibliography entry is copied and organized alphabetically around this study's two research subquestions: (a) how much of their tests should testers plan prior to receiving the testable code in the agile environment (Crispin & Gregory, 2009)? (b) which activities, in addition to business test cases, might testers be responsible?

Evaluation Criteria

Text material from disparate references that are directly related to the study are evaluated using criteria below (Busch et al., 2012):

- Per Ambler's (2007) declaration that agile projects are successful most sources in this annotated bibliography are published in or after 2007.
- HTMLs, Adobe Acrobat PDFs, and Microsoft Word search features assess the presence of themes of a reference and variations on that theme are identified and recorded.
- Abstracts of peer reviewed research articles help determine the relevance of references for this study. Peer-reviewed journal articles often have higher methodological and reporting quality compared to articles published in non-peer-reviewed journals (Rochon et al., 2002). However only references which are available in full text version through their DOI information are included in this study. The full texts of the articles are further

reviewed for reference content that is directly applicable to this study.

- Journal articles, professional conference papers, and some textbooks are less often relied on as peer-reviewed sources.

In general, the evaluation criteria help narrow the sources from references to the core subject matter of this paper: b2b testing in agile software development environments. Exceptions to these criteria include the 2006 QAI training book, a popular reference in software-development quality assurance and a highly relevant, though University specific, master's thesis of Farooq and Azmat at the Blekinge Institute of Technology in Sweden.

Reading and Organization Plan

Reading plan. The reading plan enables the researcher to read through the selected references in this study in order to explore the concepts embedded in the main research question and the sub questions. Construction of the reading the plan is framed in relation to the conceptual analysis strategy, presented by Busch et al. (2012). The references are first selected based on the criteria described in the literature evaluation criteria section located in three topics: (a) the role of software testers in agile software teams in a b2b context; (b) appropriate amounts of software tester readiness and planning; and (c) testing activities in addition to verifying business test cases. Using the guidelines described in Busch et al. (2012), the references are subjected to a more thorough reading by conceptual analysis. According to Busch et al. (2012), conceptual analysis involves an eight step coding process designed to examine literature through the use of key words and phrases. The eight

steps, as applied in this study, are detailed below.

1. **Level of analysis.** The level of analysis during coding is based on single words or sets of words such as *tester, agile software development, or b2b*. These words and phrases often serve as the search terms used to collect literature.

2. **How many concepts to code for.** Three larger concepts are coded initially, based on the themes of the organizational plan. As other relevant concepts emerge during the coding process, these are noted.

3. **Existence or frequency of a concept.** For this study coding focuses on existence of concepts as opposed to frequency because (a) the focus is on understanding the meaning of the concepts, and (b) some literature may focus on one concept in great detail.

4. **Level of generalization.** Terms with similar meanings such as *testers* and *quality assurance* and are coded similarly, as long as they are in the same context. Terms with different context and/or range of meaning such as *software quality* and *software programming* are coded separately.

5. **Coding rules.** *Translation rules* help the researcher decide what to code for, as suggested by Busch et al., 2005. For example, *software tester* and *quality assurance analyst* are the same concept.

6. **Irrelevant Information.** Concepts that are irrelevant that would not benefit this study are ignored. For example, *business-to-consumer* practices are not included.

7. **Text Coding.** The coding of text is accomplished by search features within each document type. The "find" feature of HTMLs, Adobe Acrobat PDFs, and Microsoft Word search through available text format. Results from these searches are recorded in a Microsoft Word table, organized by the predetermined three thematic concepts. Manual coding of textbooks is performed using the index of the text, to also identify the existence of important concepts in the text. Microsoft Word helps in the quick leveraging of relevant texts, as repeated searches of texts can be quite time consuming (Busch et al., 2005).

8. **Analyze results.** The Annotated Bibliography presents the analyzed findings of the coding process. The conclusion of the study places the findings of the study in the context of the three main research questions.

Organizational plan. To support the construction of ideas in this annotated bibliography, the references in this annotated bibliography are organized thematically around three themes, rather than chronologically ("Literature reviews," 2013). Each theme is directly related to one of the research questions: (a) the role of the software tester in an agile software team, in a b2b environment (b) how much of their tests should testers plan prior to receiving the testable code in the agile environment (Crispin & Gregory, 2009), and (c) which business activities, in addition to business test cases, might testers be responsible?

Anticipated topics for theme one include (a) high quality needs of b2b customers in agile SDLC (Filieri, Ghezzi, & Tamburrelli, 2012; Mairiza, Zowghi, & Nurmuliani, 2010) and (b) assigning the responsibility for the quality of the software to the tester instead of to programmers (Roseberry,

2012). Theme two places QA'ing in the context of other project development practices; anticipated topics for discussion in theme two include (a) identifying areas of risk that affect test readiness (Heusser & Kulkarni, 2011) and test planning (Crispin & Gregory, 2009). Topics in theme three stem from the question: which activity activities, in addition to verifying business requirements and test cases, might testers be responsible? The theme discusses the Crispin and Gregory (2009) proposal that exploratory testing is the main testing activity, but other testing practices are discussed as well.

Annotated Bibliography

This annotated bibliography organizes 30 references in relation to three themes: (a) the role of software testers in agile software teams in a b2b context; (b) appropriate amounts of software tester readiness and planning; and (c) testing activities in addition to verifying business test cases. The summaries of the references for the annotated bibliography are paraphrased from full readings of the original text. The references in this study help describe the unique context of agile testing in a b2b context. As noted by two references (Crispin & Gregory, 2009; Heusser 2012), it is always most important for a QA to identify their own unique context rather than to experiment, as a way to judiciously choose practices that best suit their environment. This study does not attempt a comprehensive list of best practice prescriptions but identifies some useful ones identified in peer-reviewed sources. References are cited as a credible based on features outlined by Bell and Frantz (2012): authority, objectivity, quality of work, coverage of work, and currency.

Theme 1: The Role of Software Testers in Agile Software Teams in a b2b Context

Ahamed, S. S. R. (2009). Studying the feasibility and importance of software testing: An analysis. Retrieved from http://arxiv.org/abs/1001.4193

Abstract. Ahamed describes the purpose of software testing. The author notes that several testing strategies are appropriate for each programming effort. Typically for every 100 lines of well-written code there are one to three defects. The author focuses on black-box or functional

testing and whitebox or structural testing. White-box testing enters inputs into every line of the code in order to verify that the code has no structural errors. The full text of the Ahamed article treats various testing topics.

Credibility. Prof. Dr. S.S.Riaz Ahamed holds several academic degrees in the computer sciences, including a PhD degree in Information Technology & Computer. During the course of his 15-year career, he has taught and authored 20 books on a number of the latest topics in computer science and engineering and information technology. The "International Journal of Engineering Science and Technology" (IJEST™) is an international online journal in English published monthly. The editors of the journal read all submissions and some are sent for peer review, as necessary. (Universidad Azteca International Network System, 2013; IJEST™, 2013)

Summary. The author's testing progression diagram (see Figure 1) is useful for explaining the two phases of testing throughout software development; the downward sloping line shows test planning prior to receiving completely testable code and the upward slope shows testing with completely testable code. The author is also effective at describing the advantages of different test types. Programmed tests, like unit tests, occur at the beginning of testing and throughout the process. However system tests are most effective in the late phases of testing. The testing descriptions by the authors offer a foundation for the role of a tester on software development teams.

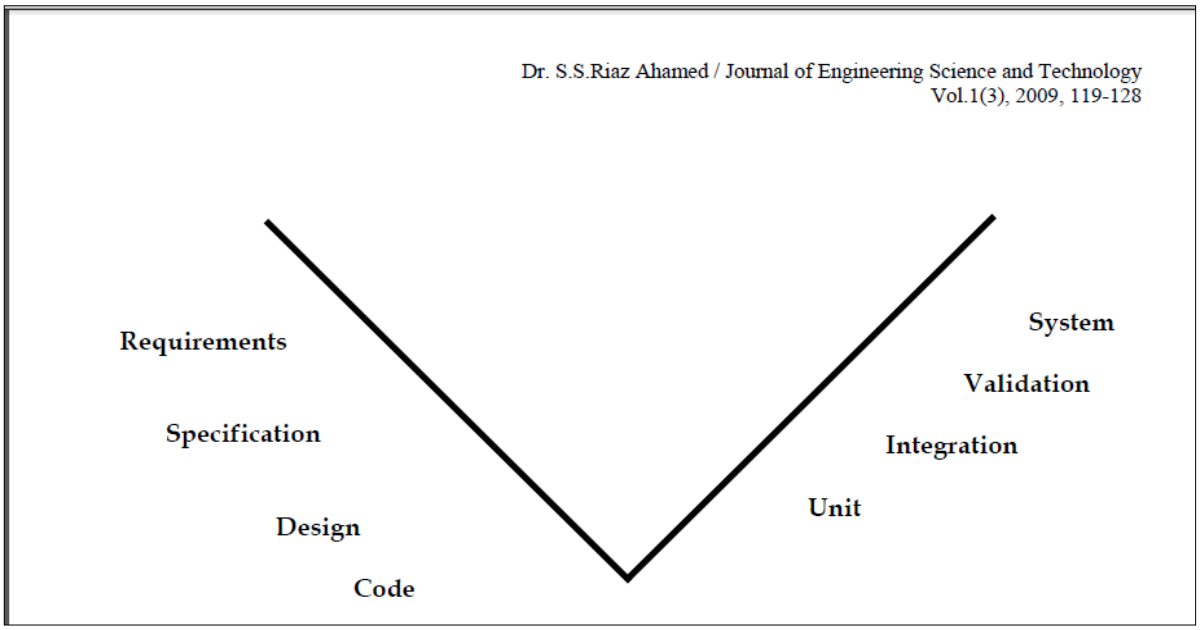

Figure 1. Testing progression diagram.

Ambler, S. (2007). Survey says...Agile has crossed the chasm. *Dr. Dobb's Journal: The World of Software Development*, *32*(8), 59-61. Retrieved from Computer Source database: http://search.ebscohost.com.libproxy.uoregon.edu/login.aspx?direct=true&db=cph&AN=25734588&login.asp&site=ehost-live&scope=site

Abstract. The article examines a survey about the effectiveness of agile software development practices. The survey includes responses of 781 software personnel about their software development practices and discusses their adoption of agile software development practice, formally a suggested practice by Beck in 2001. Results of the survey show that techniques are adopted by a majority of commercial and government organizations. Approximately 69% of projects harness agile development practices on project teams of mainly less than 10 people. The survey shows the success rate of agile projects at 77%. Most participants, (51.7%) used agile practices for team members that were located close to each other. Agile development practices include 27 items, with participants self scoring highest on these five items: iterative development, regular delivery of working software; configuration management; whiteboard modeling; and customer tests. Software professionals admitted they had the most room for improving their agile practices in five other items: pair programming, database testing, database refactoring, model reviews, and data modeling.

Credibility. Ambler has authored several books focused on the Disciplined Agile Delivery process decision framework, the Unified process, agile software development, the Unified Modeling Language, and CMM-based development. Ambler has a BSc in computer science and a MA in information science from the University of Toronto. Since 1990 he has worked in

various roles: Software Engineer, Business Architect, System Analyst, System Designer, Project Manager, Smalltalk programmer, Java programmer, and C++ programmer. Further he has led the development of several software processes, including Agile Modeling (AM), Agile Data (AD), Enterprise Unified Process (EUP), and Agile Unified Process (AUP) methodologies. Ambler has written columns for various magazines including: Dr. Dobbs Journal, Software Development, Object Magazine, and Computing Canada. Dr. Dobbs Journal is an online magazine, which caters to software engineers by providing practical solutions to real-world problems.

Summary. Ambler addresses the first theme tackled in this annotated bibliography, Agile development. Ambler explains that a high percentage of practitioners have adopted the new agile practice successfully and he supports the notion that agile development is worth studying as a best practice of software development companies of various sizes. Ambler lists several software test activities that are desirable throughout the development cycle including: (a) customer tests, (b) independent testing, (c) database testing, and (d) test plan.

Filieri, A., Ghezzi, C., & Tamburrelli, G. (2012). A formal approach to adaptive software: Continuous assurance of non-functional requirements. *Formal Aspects of Computing*, *24*(2), 163–186. Doi: 10.1007/s00165-011-0207-2

Abstract. Filieri, Ghezzi, and Tamburrelli conduct research on software systems that are vital to external companies. The authors note that these same software systems are increasingly configured to adapt to changes in the environment in which they are embedded. Moreover, adaptation often needs to be performed automatically, through self-managed reactions enacted

by the application at run time. Off-line, human-driven changes should be requested only if self-adaptation cannot be achieved successfully. To support this kind of autonomic behavior, software systems must be empowered by a rich run-time support that can monitor the relevant phenomena of the surrounding environment to detect changes, analyze the data collected to understand the possible consequences of changes, reason about the ability of the application to continue to provide the required service, and finally react if an adaptation is needed. This paper focuses on non-functional requirements, which constitute an essential component of the quality that modern software systems need to exhibit.

Credibility. Antonio Filieri is a student at Politecnico di Milano. His main research area is Software Reliability. In 2007 he received an MS in Computer System Engineering from Politecnico di Milano and in 2009 a MS in Computer Science from the University of Illinois. The Tamburrelli is a Computer Science Engineer. He is currently a PostDoc (Marie Curie Fellow IEF) in Software Engineering at the Università della Svizzera Italiana (USI) in Lugano, Switzerland. His research interests include: Software Models and Models at Run-time, Service-Oriented Architectures, Mobile Computing, Performance and Reliability, and Non-Functional Requirements. He has been a PostDoc and PhD Student at Politecnico di Milano under the supervision of Prof. Carlo Ghezzi. Ghezzi has co-authored over 180 papers and 8 books. His papers appeared on prestigious journals like the Journal of the ACM, Information and Control (now Information and Computation), ACM Transactions on Software Engineering and Methodology, ACM Transactions on Programming Languages and Systems, IEEE Transactions on Software Engineering.

Summary. Although the proposed approach is quite general, it is mainly exemplified in the paper in the context of service-oriented systems, where the quality of service (QoS) is regulated by contractual obligations between the application provider and its clients. Their research has important implications for quality guarantees among b2b parties. The authors analyze the case where an application, exported as a service, is built as a composition of other services. Non-functional requirements—such as reliability and performance—heavily depend on the environment in which the application is embedded. Thus changes in the environment may ultimately adversely affect QoS satisfaction. We illustrate an approach and support tools that enable a holistic view of the design and run-time management of adaptive software systems. The approach is based on formal (probabilistic) models that are used at design time to reason about dependability of the application in quantitative terms. Models continue to exist at run time to enable continuous verification and detection of changes that require adaptation.

The authors assert that streaming data and other dynamic software expose their business customers to some risks. The authors propose that software models could help software engineers adapt to changes in the performance of their products for example. The authors suggest that business customers may need concrete guarantees about the reliability of internet software.

Heusser, M. (2012, November). *A brief history of the quality movement and what software should do about it.* 2012 Pacific Northwest Software Quality Conference Portland, OR. Retrieved from http://www.pnsqc.org/2012-conference/papers-and-presentations

Abstract. Heusser attempts to reach four goals in this conference keynote presentation: (a) he

presents a quick introduction to the automotive quality movement of the 20th century, (b) he relates manufacturing quality to software quality, and examines the ways in which software is different, (c) he identifies some of the common ways in which the analogy fails, and (d) he connects these ideas to the modern agile and lean software movements, and suggests some refinements. Heusser provides an important entry for testers hoping to improve their practices by learning from an established professional in the field. He notes that there are many project management techniques and metrics that could be relevant for software testing, but the techniques and metrics often do not hold up in practice. He notes that total quality management has important strengths such as crossfunctional teams and early feedback. As well, he discusses the capability maturity model, which has the same strengths (QAI, 2006). However, he also lists significant weaknesses of the capability maturity model and total quality management, which often rely too much on standardization and metrics. Heusser expresses the need for an empowering development framework for the software testing field instead of top-down implementations. He lays a viable foundation for business context of agile testing.

Credibility. Matthew Heusser is a consulting software tester and software process naturalist who has spent career developing, testing, or managing software projects. Heusser is a contributing editor for Software Test & Quality Assurance Magazine, and writes a personal blog, which is consistently highly-ranked among software writing. He is the lead editor for "How to Reduce the Cost of Software Testing" (2011). He was a keynote speaker at the 2012 Pacific Northwest Software Quality Conference (PNSQC). His writing attracts a diverse

audience consisting of software product managers, quality professionals, Agilists, managers, contractors, consultants, customers, developer-testers, tester-developers, and maintenance engineers.

Summary. Heusser elevates software testing to more of a professional and academic discipline. His citations help flesh out other references used in this paper. He supplies open-ended suggestions on testing that his audience is encouraged to develop within their own customized testing practices. He deconstructs some of the jargon around testing and the various software development practices, reducing the most important ones to their core of shorter, more efficient life cycles. The most important suggestion he proffers in relation to this study is that programmers, testers, and product owners work together to even before coding begins.

Mairiza, D., Zowghi, D., & Nurmuliani, N. (2010). An investigation into the notion of non-functional requirements. In *Proceedings of the 2010 ACM Symposium on Applied Computing* (pp. 311–317). New York, NY, USA: ACM. Doi:10.1145/1774088.1774153

Abstract. Although Non-Functional Requirements (NFRs) are recognized as very important contributors to the success of software projects, studies to date indicate that there is still no general consensus in the software engineering community regarding the notion of NFRs. This paper presents the result of an extensive and systematic analysis of the extant literature over three NFRs dimensions: (1) definition and terminology; (2) types; and (3) relevant NFRs in various types of systems and application domains. Two different perspectives to consider NFRs are described. A comprehensive catalogue of NFRs types as well as the top five NFRs that are frequently considered are presented. This paper also offers a novel classification of NFRs based

on types of systems and application domains. This classification could assist software developers in identifying which NFRs are important in a particular application domain and for specific systems.

Credibility. Mairiza, and Zowghi are PhD students at the School of Software, University of Technology Sydney, Australia. Mairiza is a member of the Human Centered Technology Design (HCTD) Research Center and the Requirements Engineering (RE) Laboratory University of Technology, Sydney. She is also an academic staff at the Faculty of Computer Science of University of Indonesia and an academic staff of the Faculty of Engineering and IT, University of Technology, Sydney. Zowghi is a professor of Software Engineering at the University of Technology, Sydney. The paper is published by the ACM symposium.

Summary. Mairiza, Zowghi, and Nurmuliani make a compelling case for the prioritization of non-functional requirements by industry. They further narrow the focus of quality testers in various business environments, including banking, education, energy, government, insurance, medical, telecommunication, and transportation. However, this aspect of their study is slightly less applicable for business to business development environments. Nevertheless, they provide cautionary tales and note two catastrophic instances from ignored nonfunctional requirements: the London Ambulance System and the New Jersey Department of Motor Vehicles Licensing System. They also note that performance and usability are the most commonly considered NFRs in various types of systems and application domains. The definition and attributes of performance and the top five most considered NFRs are presented in Table 1.

Table 1

The Five Most Commonly Considered NFRs

Performance	requirements that specify the capability of software product to provide appropriate performance relative to the amount of resources needed to perform full functionality under stated conditions	response time, space, capacity, latency, throughput, computation, execution speed, transit delay, workload, resource utilization, memory usage, accuracy, efficiency compliance, modes, delay, miss rates, data loss, concurrent transaction processing
Reliability	requirements that specify the capability of software product to operates without failure and maintains a specified level of performance when used under specified normal conditions during a given time period	completeness, accuracy, consistency, availability, integrity, correctness, maturity, fault tolerance, recoverability, reliability, compliance, failure rate/critical failure
Usability	requirements that specify the end-user-interactions with the system and the effort required to learn, operate, prepare input, and interpret the output of the system	learnability, understandability, operability, attractiveness, usability compliance, ease of use, human engineering, user friendliness, memorability, efficiency, user productivity, usefulness, likeability, user reaction time
Security	requirements that concern about preventing unauthorized access to the system, programs, and data	confidentiality, integrity, availability, access control, authentication
Maintainability	requirements that describe the capability of the software product to be modified that may include correcting a defect or make an improvement or change in the software	testability, understandability, modifiability, analyzability, changeability, stability, maintainability compliance

Mauldin, E., Nicolaou, A., & Kovar, S. (2006). The influence of scope and timing of reliability assurance in b2b e-commerce. *International Journal of Accounting Information Systems, 7*(2),

115-129. Doi: 10.1016/j.accinf.2005.09.002

Abstract. This study investigates potential demand for third-party assurance reports in business-to-business electronic commerce (b2b e-commerce). We experimentally analyze 95 purchasing professionals'' decisions to recommend using a b2b exchange. The experiment is a 2X2 between-participants design varying the scope and timing of an assurance report with an additional control condition of no assurance. The results suggest that purchasing professionals are more likely to recommend use of the exchange when general assurance over the reliability of the exchange's system is present than when specific assurance over the reliability of transaction information is present. Purchasing professionals are also more likely to recommend using the exchange when the assurance report is continuous than when it is static, issued at a point in time. However, the results also suggest that participants are less likely to recommend using the exchange when specific information assurance or static assurance is present than when assurance is not present at all. Further, other factors besides assurance, especially trust in the trading partner and propensity to trust, have a stronger influence on the decision to use a b2b exchange than the presence of either continuous or systems assurance. Potential implications for practice include redesigning existing assurance services to provide continuous assurance, de-emphasizing formalized reports and considering assurance services over other factors, such as web site design of the exchange's system is present than when specific assurance over the reliability of transaction information is present. Purchasing professionals are also more likely to recommend using the exchange when the assurance report is continuous than when it is static,

issued at a point in time. However, the results also suggest that participants are less likely to recommend using the exchange when specific information assurance or static assurance.

Credibility. Elaine Maudlin is an Associate Professor and BKD Professor of the University of Nebraska at Omaha. Andreas Nicolaou is a professor at Bowling Green State University. He has a PhD in accounting information systems. Stacy Kovar has a PhD from Oklahoma State University. Her current research involves the decision by managers to select internal versus external sources for information assurance, student time management and performance in accounting programs, and student perceptions of key learning objectives. The paper is published in the *International Journal of Accounting Information Systems,* which peer reviews its submissions on the 'double-blind' system by two or more specialists selected from a panel of referees (Guide for Authors, January 26, 2013).

Summary. The authors' scientific experiment has important implications for the types of quality assurance services that might be necessary in b2b electronic transactions, possibly in software development. They conclude that information system quality assurance guarantees are more important than specific information assurance guarantees. Customers also desire assurance about their most pressing concerns. Business customers may desire assurance related to the performance and usability of reports, as suggested by Mairiza, Zowghi, and Nurmuliani (2010). Perhaps customers only trust specific information quality assurance guarantees if system guarantees are already in place. As well, business customers prefer ongoing quality assurance practices instead of discrete ones. Business customers were not more likely to trust assurances by CPAs, suggesting a distrust in some formal practices.

The authors state that the existence of trust between business participants is more important than any quality assurance guarantee. Trust may be confirmed by the performance of the software or independently imbued within a relationship on a referral basis. They also suggest that consulting engagements may enhance the quality of web sites especially in b2b contexts.

Molinari, L. K., Abratt, R., & Dion, P. (2008). Satisfaction, quality and value and effects on repurchase and positive word-of-mouth behavioral intentions in a b2b services context. *Journal of Services Marketing, 22*(5), 363-373. Doi: 10.1108/08876040810889139

Abstract. This article provides an understanding of how satisfaction, quality, and value affect repurchase and positive word-of-mouth in a business-to-business (b2b) setting. Most previous studies in this area apply to business-to-consumer (b2c) situations. The authors also note that b2b service transactions are commonly after purchases. Managers are also given guidelines on how to increase customer satisfaction in b2b services.

Credibility. Lori K. Molinari is Assistant Professor of Business Management at the School of Business at Point Park University in Pittsburgh, Pennsylvania. Dr. Molinari is also a sales manager for a global freight forwarding company. Russell Abratt is the Associate Dean and Professor of Marketing in the Huizenga School of Business and Entrepreneurship at Nova Southeastern University, Florida, USA. His PhD and MBA are from the University of Pretoria, South Africa. His research interests are in Marketing Strategy and Corporate Identity and Reputation. Abratt is the corresponding author. Paul Dion received his PhD in Management Studies, with a major in marketing and a minor in statistical methods, from the University of

Toronto in 1986. He has been an Associate Professor at Susquehanna University in Selinsgrove, Pennsylvania since 1992.

Summary. While quality factors may be important to business customers in b2bs, the authors provide evidence that suggests service value is important to business customers. Satisfaction occurs when a service is delivered correctly the first time. Exceeding customer expectations is the key criterion for increasing satisfaction, quality, and value. However the link of quality to repurchase intentions is low, likely due to the availability of lower-price substitutes in the marketplace. The most important factors to getting customers to repurchase are value, satisfaction, exceeding customer expectations, and word of month. Positive word of mouth works directly and indirectly through value. Customer experiences with a product should be at least as good as it is promoted to be. As well b2b companies should understand customer expectations for their product.

Pantouvakis, A. (2011). Internal service quality and job satisfaction synergies for performance improvement: Some evidence from a b2b environment. *Journal Of Targeting, Measurement & Analysis For Marketing, 19*(1), 11-22. Doi:10.1057/jt.2011.2

Abstract. The article assesses the performance of business units in a business-to-business (b2b) environment, by presenting a framework including tangible (hard) and intangible (soft) elements. The intangible part encapsulates internal service quality and job satisfaction, whereas the tangible part includes quantifiable elements. In this study, the dimensionality of the INTSERVQUAL instrument is tested in a b2b environment through

Confirmatory Factor Analysis (CFA), and its ability to explain job satisfaction is explored with regression analysis, both in isolation and together with other tangible elements. An optimization framework is then proposed, with respect to the satisfaction – internal-service quality – performance triad in a b2b environment, based on a two-phase data-envelopment analysis model. ‘ Interactive ’ and ‘ physical ’ quality, as extracted from INTSERVQUAL, assesses internal service quality (ISQ) sufficiently and appropriately in a b2b environment. In addition, the results suggest that internal customers ’ job satisfaction, which depends on the soft (interactive and physical) ISQ dimensions, as well as the ‘ hard ’ ISQ dimensions, succeeds in accounting for measurable effects on the outcomes (performance) of the businesses. Managers of service firms should focus on both soft and hard dimensions of internal service quality, as they influence job satisfaction and, as a consequence, business performance. Moreover, the benchmark method (DEA) provides useful information about the efficiency of the set of decision-making units.

Credibility. Angelos Pantouvakis holds a Civil Engineering degree (Meng) from National Technical University of Athens, an MBA from the Nottingham Business School, UK, and a PhD in Performance Measurement and Services Marketing from the Judge Business School of the University of Cambridge, UK. His research interests are in the area of services marketing and consumer behavior, specifically in the services sector. He is senior lecturer in the Department of Maritime studies at the University of Piraeus, Greece after having served in managerial positions at Deloitte. The *Journal of Targeting, Measurement and Analysis for Marketing* is a peer-reviewed professional journal for Marketers.

Summary. The article suggests that the job satisfaction of a company's internal stakeholders transfers to the quality of its products. The article suggests that the job satisfaction of each member of the software development team is important for the profit of the software product.

Patcha, K. K. (2009). Agile EDI Framework for b2b Applications. In *International Conference on Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09* (pp. 1–3). Presented at the International Conference on Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09. doi:10.1109/ARTCom.2009.13

Abstract. The unified software development process or unified process is a popular iterative and incremental software development process framework. The best-known and extensively documented refinement of the unified process is the rational unified process (RUP) created by the Rational Software Corporation, a division of IBM. Agile unified process is a simplified version of the RUP developed by Scott Ambler, the Practice Leader for agile development at IBM. It describes a simple, easy to understand approach to developing business application software using agile techniques and concepts yet still remaining true to the RUP. This paper explains the need of applying agile methodologies and agile unified process framework for developing b2b, EDI applications. This paper presents a refined agile unified process framework tailored for EDI (electronic data interchange) and b2b (business-to-business) software projects. The International Conference on Advances in Recent Technologies in Communication & Computing, ARTCom 2013, is an international conference that presents and discusses the practices of communication, computational engineering, and computer technology.

Credibility. Patcha is a software engineer for IBM in India. International Business Machines Corporation or IBM, is an American multinational technology and consulting corporation, with headquarters in Armonk, New York, United States. IBM manufactures and markets computer hardware and software, and offers infrastructure, hosting and consulting services in areas ranging from mainframe computers to nanotechnology. The International Conference on Advances in Recent Technologies in Communication & Computing, ARTCom, is an international conference where theory, practices, and applications of Communication, Computational Engineering, Computer Technology and related topics are presented and discussed.

Summary. Patcha explains that a significant cost in data transfer systems in b2b applications is the cost in time and money of the initial setup. His findings suggest an increased need in b2b applications for higher quality usability and documentation. Implementation, customization, and training may dissuade some companies from accepting data transfer engagements. Simple systems also allow reuse across partners. Data exchanges may require integration with other applications as well. The author asserts that agile is the leading development method for global companies and is highly suggested for b2b software projects.

Quality Assurance Institute. (2006). Guide to the CSTE common body of knowledge. N.p.

Abstract. The Certified Software Tester (CSTE) program is developed by software testing professionals to recognize software testers who demonstrate a predefined level of testing competency by studying the CSTE textbook. The CSTE program is directed by an independent Certification Board and administered by the Quality Assurance Institute (QAI).

Changes in the certification text should occur approximately every three years.

Credibility. The CSTE is one of the software tester education programs valued by some American companies. The testing principles are grounded in the capability maturity model (CMM) software development principles, which are not completely accepted as necessary for good software development. Wikipedia for example asserts that CMM is irrelevant to many software products.

Summary. The CSTE program has some suggestions for software testing. These suggestions are not specifically supported with specific scientific evidence. However, their suggestions are guides for additional research on software testing. The text mainly lays the foundation for a few testing terms, such as SDLC and quality assurance.

Zhang, Y., Fang, Y., Wei, K., Ramsey, E., McCole, P., et al. (2011). Repurchase intention in b2c e-commerce—a relationship quality perspective. *Information & Management*, *48*(6), 192-200. Doi:10.1016/j.im.2011.05.003

Abstract. Information systems professionals must pay attention to online customer retention. Drawing on the relationship marketing literature, they formulated and tested a model to explain b2c user repurchase intention from the perspective of relationship quality. The model, empirically tested through a survey in Northern Ireland, showed that online relationship quality and perceived website usability positively impacted customer repurchase intention.

Credibility. Zhang is an Assistant Professor in the School of Management and Economics, Beijing Institute of Technology. He received his PhD from City University of Hong Kong and University of Science and Technology of China. His current research is focused on knowledge

management, and electronic commerce. He has published papers in International Journal of Information Management. The International Journal of Information Management (IJIM) is an international, peer-reviewed journal, which aims to bring its readers the very best analysis and discussion in the developing field of information management.

Summary. The authors suggest variables that could be important for maintaining the quality of any software customers. Online customers behave differently than in-person ones. The authors confirm that factors such as website usability and reputation become more important in online transactions. This study corroborates others on the importance of usability for b2b customers of internet-based software companies. Moreover, online relationship quality was positively influenced by perceived vendor expertise in order fulfillment, and perceived vendor reputation, whereas distrust in vendor behavior negatively influenced online relationship quality.

Theme 2: Appropriate Amounts of Software Tester Readiness and Planning

Bertolino, A. (2007). Software testing research: Achievements, challenges, dreams. In *Future of Software Engineering, 2007. FOSE '07* (pp. 85 –103). Presented at the Future of Software Engineering, 2007. FOSE '07. doi:10.1109/FOSE.2007.25

Abstract. Bertolino provides a consistent roadmap of the most relevant challenges addressed in testing. The routes from the achievements to the dreams are paved by the outstanding research challenges, which are discussed in the paper along with interesting ongoing work. She identifies several brooding questions of testing: why, how, how much, what, where, and when. The most ambitious aspirations for testers lie in universal test theory, model-based testing, 100% automated testing, and efficacy-maximized test engineering. Domain-specific software development languages, that are developed to meet the particular demands of a company, quickly become problematic for the larger software-development community that would like to share testing solutions.

Credibility. Antonia Bertolino is a researcher at the Italian National Research Council and the leader of the Software Engineering Research Laboratory at the Instituto di Scienza e Technologie dell'Informatzione in Pisa, Italy. She is area editor for software testing of the Journal of Systems and Software, published by Elsevier. Bertolino has authored over 80 papers. She offers a needed academic perspective of the field of software testing. She isolates the practice of software testing as an academic one. The paper is edited and published by the Future of Software Engineering symposium of 2007.

Summary. While noting several decades academic progresses on software testing goals, Bertolino explains the dichotomy between academic and industrial testing. The testing aspirations she lists are less relevant for business settings. Still, her software testing research roadmap (see Figure 2) suggests that testing should be further studied as a discipline so that testers can understand some of the academic theory behind their practices. She asserts that testers are inadequately educate for their roles, or perhaps software systems are too complex for testers. Bertolino confirms this annotated bibliography's intention to further educate testers. In her early accolades to the testing field she notes that rapid release cycles, similar to agile development are increasingly being trusted in the discipline.

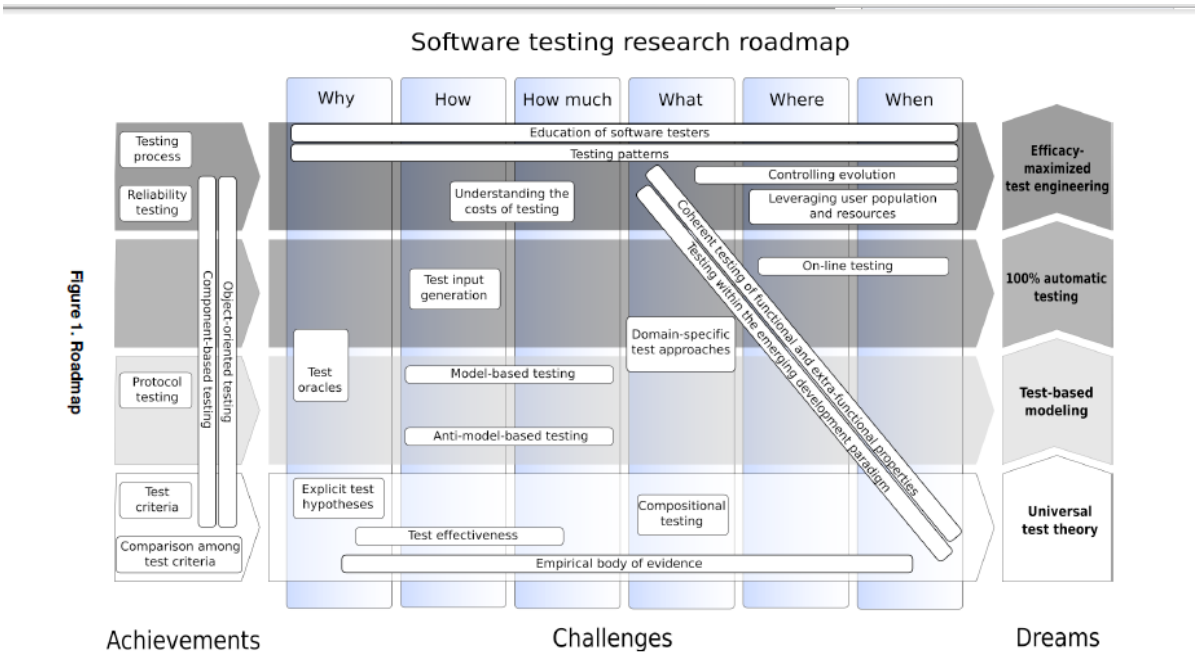

Figure 1. Roadmap

Figure 2. Achievements, challenges, and the theoretical goals (dreams) of software testers and researchers.

Bruns, A., Kornstädt, A., & Wichmann D,. (Sept.-Oct 2009). Web application tests with Selenium, IEEE Software, *26(*5), (pp. 88-91) doi:10.1109/MS.2009.144

Abstract. Web applications tend to continuously evolve and thus need thorough, yet lean and automatic, regression testing. Bruns and his colleagues describe automatic regression testing for

Web applications that uses the Selenium testing framework. Bruns and his colleagues also provide many testing hints for practitioners.

Credibility. Andreas Bruns is a senior software engineer who focuses on consulting and managing in the areas of advanced software architecture, software transformation, and software engineering. Andreas Kornstädt is a senior software architect and carries out research at Stanford University's Center for Computer-Assisted Research in the Humanities. Dennis Wichmann is also a software engineer. The IEEE Computer Society is a peer-reviewed source for technology information, and collaboration.

Summary. Certain automated testing tools are particularly suited for the tests within b2b contexts. Selenium is an open-source tool that reflects business structure and mimics customers. Selenium is portable open source software available for software development companies that use Windows, Linux, or Macintosh operating systems. Selenium allows testers to write tests in are written as HTML tables in popular programming languages, including C#, Java, Groovy, Perl, PHP, Python and Ruby. The tests can then be run against many modern web browsers.

Crispin, L. & Gregory, J. (2009). *Agile testing a practical guide for testers and agile teams*. Addison Wesley.

Abstract. In *Agile Testing*, Crispin and Gregory define agile testing and illustrate the tester's role with examples from real agile teams. They teach how to use the agile testing quadrants to

identify what testing is needed, who should do it, and what tools might help. The book chronicles an agile software development iteration from the viewpoint of a tester and explains the seven key success factors of agile testing. They also supply a dynamic agile testing quadrant that should be useful for the teaching in most organizations that perform testing (p.242).

Credibility. Lisa Crispin and Janet Gregory are two of the software industry's most experienced agile testing practitioners and consultants. Since 1982, Crispin has worked as a programmer, analyst, technical support engineer, tester, and QA director with various organizations (About Lisa, 2013). Gregory has helped introduce agile practices into companies as tester, and QA manager. Gregory has a degree in Computing Science from the University of Alberta, an Information Management Certificate from the University of Calgary, Scrum Master Certification, as well as Quality Management Certification from the ASQ (Janet Gregory, 2013).

Summary. This text addresses all three themes of this annotated bibliography in some depth. In 2009, they reaffirm the success of agile software development practices. They describe the contexts for organizations with successful agile testing. Crispin and Gregory emphasize the collaboration opportunities among software roles. They note that testers are both part of customer teams and development teams (see Figure 3). The authors also begin to outline the appropriate business-facing tests of agile testers (see Figure 4). Importantly, they identify that automated tests are not an elixir for quality software production; rather every tool for testing serves a purpose among the 4-part testing matrix. The authors posit that skilled testers might consult talented programmer colleagues for automated verification of certain kinds of tests.

Vice-versa programmers might consult knowledgeable testers on the relevant use cases and/or expected behavior of graphic user interface products that should be tested manually. Unit and component tests should be automated, by programmers or with the help of programmers. Exploratory testing, usability, and user acceptance testing should be done manually, by testers or likely with the help of testers.

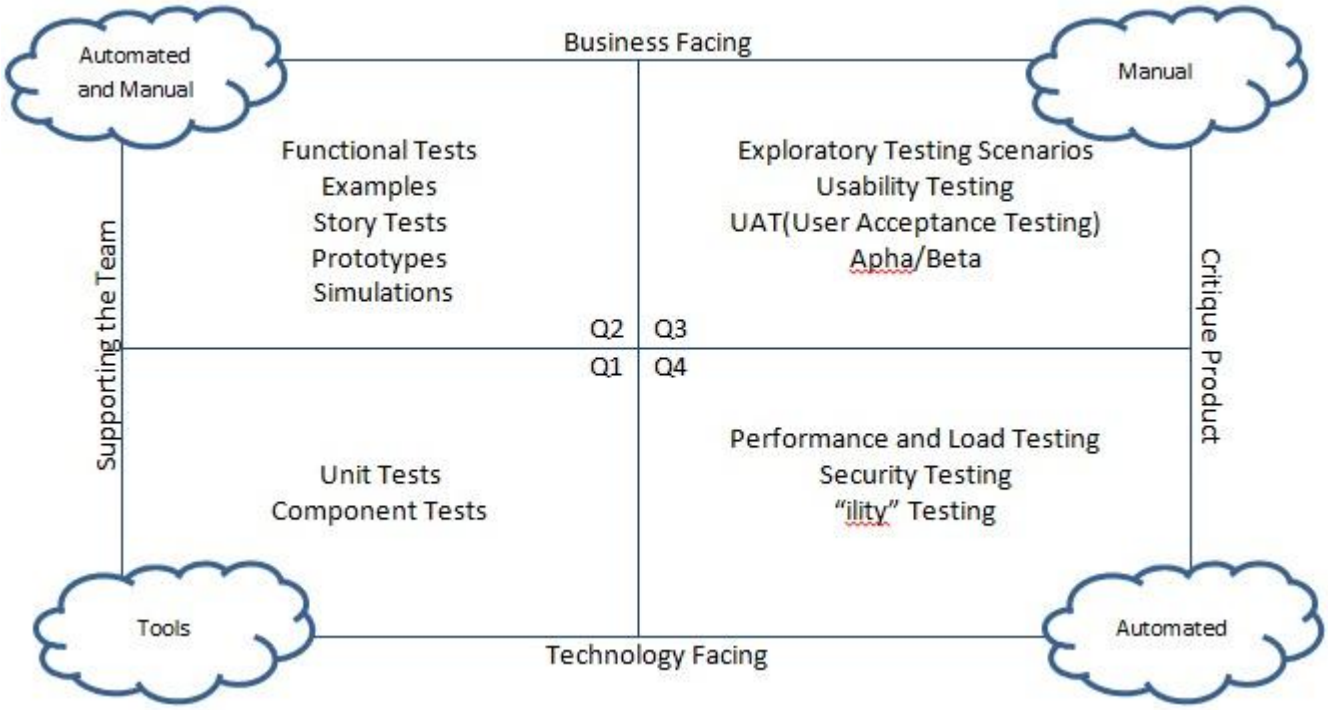

Figure 3. Testing quadrants.

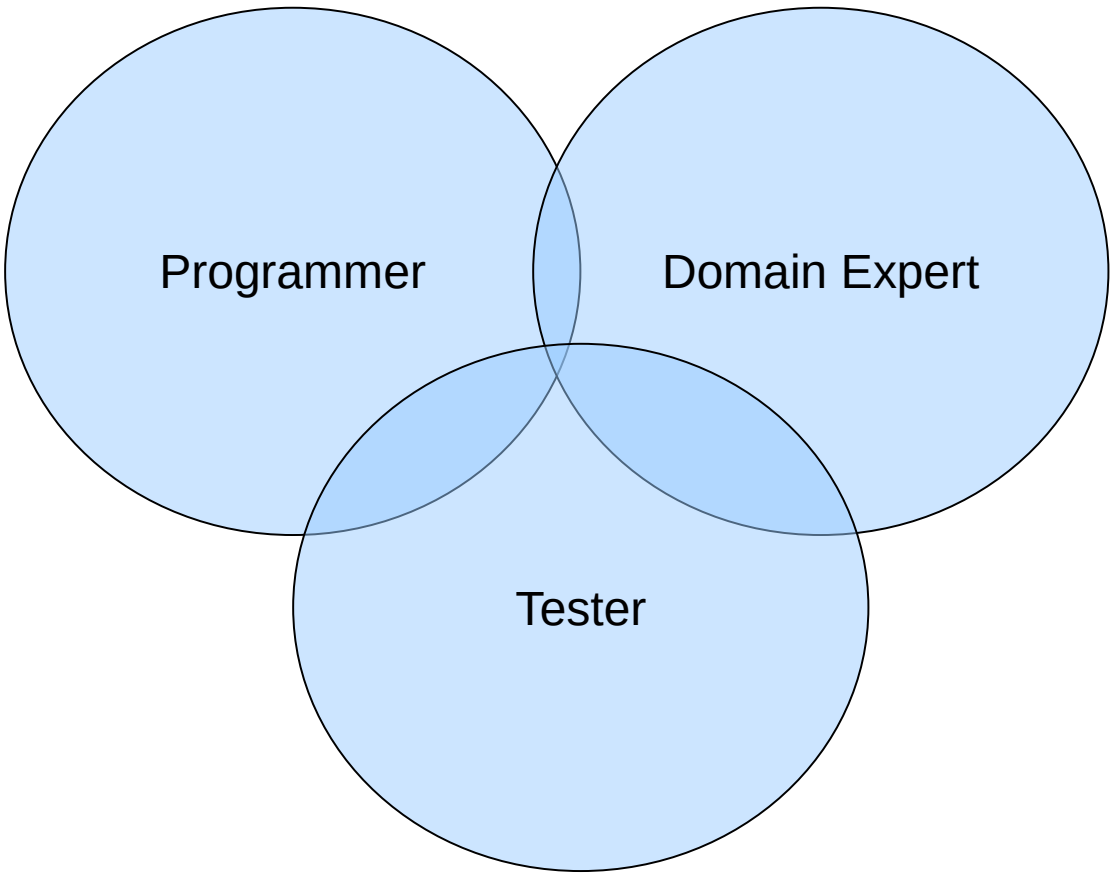

Figure 4 . Interaction of roles

Heusser, M., & Kulkarni, G. (Eds.). (2011). *How to reduce the cost of software testing (1st ed.).* Auerbach Publications.

Abstract. Heusser and Kulkarni edit the essays of various software development professionals. How to Reduce the Cost of Software Testing provides tips, tactics, and techniques to help readers accelerate the testing process, improve the performance of the test teams, and lower costs. The distinguished team of contributors—that includes corporate test leaders, best paper authors, and keynote speakers from leading software testing conferences—supply concrete suggestions on how to find cost savings without sacrificing outcome. Detailing strategies that testers can immediately put to use to reduce costs, the book explains how to make testing nimble, how to remove bottlenecks in the testing process, and how to locate and track defects efficiently and effectively.

Credibility. Matthew Heusser is a consulting software tester and software process naturalist who has spent career developing, testing, or managing software projects. Heusser is a contributing editor for Software Test & Quality Assurance Magazine, and writes a personal blog (http://xndev.com/creative-chaos/), which is consistently highly-ranked among software writing. He is the lead editor for "How to Reduce the Cost of Software Testing" (2011). He is a keynote

speaker of the 2012 PNSQC and attracts a diverse audience consisting of software product managers, quality professionals, Agilists, managers, contractors, consultants, customers, developer-testers, tester-developers, and maintenance engineers. Govind Kulkarni has spent seventeen years in software quality assurance and management. He is a Project Management Professional, Certified Quality Auditor (CQA), and Tick IT professional. He is one of the reviewers of the test maturity model integrated (TMMi), actively researching model-based testing. He has written more than 25 technical papers and frequently speaks at testing conferences. He manages his own testing website http://www.enjoytesting.com.

Summary. Of the 16 chapters in this book, the relevant sections and chapters for this bibliography are test readiness (Chapter 7), session-based management (Chapter 8), and a nimble test plan (Chapter 12). The discussion of science-based test design (Chapter 14) suggests that the design of testing could be refined by a scientific process called pairwise testing. QA testers can only perform their testing competently within a project management context. There are at least seven categories of project management risks to testing tasks, and knowing these may help improve the likelihood of the testing success of the project. Additionally, testers should structure their own work to reduce test risks. Session-based testing tips should help testers improve their documentation among themselves. The nimble test plan provides a simple way of thinking about a test plan. The test plan should have a relevant strategy and minimal requirements but enable dynamic testing/navigation of the software application. The suggested science-based pairwise testing may be less relevant to business applications where the test cases require multiple parameters.

Jaffar-ur Rehman, M., et al (2007). Testing software components for integration: a survey of issues and techniques: Research Articles. *Software Testing Verifiability and Reliability. 17*(2), 95–133. doi:10.1002/stvr.v17:2

Abstract. Component-based development has emerged as a system engineering approach that promises rapid software development with fewer resources. Yet, improved reuse and reduced cost benefits from software components can only be achieved in practice if the components provide reliable services, thereby rendering component analysis and testing a key activity. This paper discusses various issues that can arise in component testing by the component user at the stage of its integration within the target system. The crucial problem is the lack of information for analysis and testing of externally developed components. Several testing techniques for component integration have recently been proposed. These techniques are surveyed here and classified according to a proposed set of relevant attributes. The paper provides a comprehensive overview which can be useful as introductory reading for newcomers in this research field, as well as to stimulate further investigation.

Credibility. Jaffar-ur-Rehman is a Pakistani computer scientist. He is a professor and the Dean of the Faculty of Engineering and Sciences at Mohammad Ali Jinnah University in Pakistan. Jaffar-ur-Rehman is the founder of the Center for Software Dependability (CSD), which pioneered research in the domain of software dependability and reliability in Pakistan. Software Testing, Verification and Reliability (STVR) is a peer-reviewed international journal, publishing eight issues per year. It publishes papers on theoretical and practical issues of software testing, verification and reliability. The goal of the journal is to

publish high-quality papers that help researchers, educators and practitioners understand cutting edge results.

Summary. The authors suggest that there are five categories of tests used by testers to verify software components: (a) built-in testing (BIT); (b) testable architecture; (c) metadata-based; (d) certification strategy; (e) user's specification-based testing. Testers obtain direction for their testing from the software programmers and from the software users. The component-based development creates new risks where developers have too much confidence in the work of third-parties. The authors assert that reuse of components do not reduce the need for testing as application domains are dynamic.

Kadry, S. (2011). A new proposed technique to improve software regression testing cost. The *International Journal of Security & Its Applications (IJNSA) 5(*3), 46-58 Retrieved from http://arxiv.org/abs/1111.5640.

Abstract. Kadry describes the regression test process to test and verify the changes made on software. Automated testing is prone to being run too frequently as a means of detecting defects. Kadry offers the automated tests viability model (ATVM) in Table 2, which helps testers decide when to automate a test. Automated tests help reduce manual mistakes, allow parallel executions, and offer convenient analysis. A second model, the regression test selector, only runs test cases that are the least costly.

Table 2

Indicator questions for test automation.

Identifier	Topics	Related Questions
1	Frequency	How many efforts is this test supposed to be executed?
2	Reuse	Can this test or parts of it be reused in other tests?
3	Relevance	How would you describe the importance of this test case?
4	Automation effort	Does this test take a lot of effort to be deployed?
5	Resources	How many members of your team should be allocated or how expensive is the equipment needed during this test's manual execution?
6	Manual Complexity	Is this test difficult to be executed manually? Does it have any embedded confidential information?
7	Automation Tool	How would you describe the reliability of the automation tool to be used?
8	Porting	How portable is this test?
9	Execution effort	Does this require a lot of effort to be executed manually?

Credibility. Dr. Seifedine Kadry is an associate professor at the American University of the Middle East, Faculty of Engineering. He got his Master Degree in Computer Science and Applied Math from AUF-EPFL-Inria, Lebanon in 2002. He received the Doctor degree from the

Clermont Ferrand II University, France in 2007. His Research interests include software testing and security. The International Journal of Security and Its Applications is a peer-reviewed bi-monthly publication, which supports research related to security technology and the applications.

Summary. Kadry identifies software maintenance as one of the main costs in the software development life cycle. Regression tests are one common resource that development teams use to maintain their software applications. Kadry suggests an approach that lowers the cost of maintaining the software while being more effective than the dominant regression testing models. Kadry develops a hybrid regression and automated technique that he applies to a practical case and the result shows its improvement. The proposed technique relies on regression test selection based on risk analysis and automation. The proposed hybrid technique costs the least testing time and has cost-effective rates better than the individual risk analysis models or the test automation models.

Li, J., Stephanie, T., Conradi, R., & Kristiansen, J. M. W. (2012). Enhancing defect tracking systems to facilitate software quality improvement. *IEEE Software*, *29*(2), 59–66. Doi:10.1109/MS.2011.24

Abstract. The authors study nine Norwegian companies and find little usage of the detection tracking information systems for software quality assessments or process improvement initiatives. This study illustrates that goal-oriented changes or extensions to the existing data of projects' respective defect tracking systems could provide valuable and prompt information to

improve software quality assessment and assurance. The nine defect tracking systems had many common attributes including: the description, time stamp and person involved, impact, and trace of the status. However the inputs did not always trace to a goal, question, or metric.

Credibility. Li is a senior researcher at DNV Research & Innovation. His research interests include software process improvement, empirical software engineering, and software reliability. Li has a PhD in software engineering from the Norwegian University of Science and Technology. Stålhane is a full professor of software engineering at the Norwegian University of Science and Technology (NTNU). Stålhane has a PhD in applied statistics from NTNU. Conradi is a full professor in the Department of Computer and Information Science at the NTNU. Conradi has a PhD in software engineering from NTNU. Kristiansen has a master's degree in computer science from the Norwegian University of Science and Technology. Jan M.W. Kristiansen is a software engineer with Steria AS. His research interests include agile methods for software development, software process improvement, and open source software.

Summary. Defect tracking systems are a concern of testers in B2Bs because defect-tracking systems address the concerns of business stakeholders. The authors suggest that even business critical software often does not have appropriate defect tracking practices.

Because the defect tracking systems did not have specific quality assessments and software process improvements in mind the collected information often did not prove very useful. The article suggests that there are important best practices at implementing adequate defect tracking systems. For example, it might be important not to have a default value for the defect levels as it might not be obvious if stakeholders have changed the field. As well, the

identification of the source of defects might have functional and software names. Therefore it might be important to have different stakeholders verify a defect. Multiple choice options might be better than single select ones for accurately describing defects. The authors' findings are very useful because they show that: "Simple goal-oriented changes to existing data in defect tracking systems provides valuable and prompt information to improve software quality assessment and assurance (p. 59)."

Roseberry, W. (2012, November). *Code coverage isn't quality, it isn't even coverage.* Paper presented at Pacific Northwest Software Quality Conference Portland, OR. Paper retrieved from http://www.pnsqc.org/2012-conference/papers-and-presentations

Abstract. Roseberry asserts that there really is a weak relationship between code coverage and quality because code coverage is ineffective at telling whether or not tests have been thorough. Good test coverage only comes from doing as many interesting things as possible that are good at exposing flaws in code. Code coverage reports, when used as a metric of quality, hide the useful tests.

Credibility. Roseberry is a Principal Software Design Engineer in Test at Microsoft. Microsoft Corporation is an American multinational software corporation headquartered in Redmond, Washington that develops, manufactures, licenses, and supports a wide range of products and services related to computing. The company was founded by Bill Gates and Paul Allen on April 4, 1975. Microsoft is the world's largest software maker measured by revenues. It is also one of the world's most valuable companies. The PNSQC attracts a diverse audience consisting of

software product managers, quality professionals, Agilists, managers, contractors, consultants, customers, developer-testers, tester-developers, and maintenance engineers.

Summary. Roseberry helps redirect the software development community away from more intensive programming/coding towards more exploratory testing approaches to improve the quality of software. His paper shows examples of code coverage reports where 100% coverage were completed, and nevertheless ineffective. Directing one's attention in the opposite direction implied by code coverage actually yields better test generation and more confidence in what the test suite addresses.

Spillner, A., Linz, T., & Schaefer, H. (2011). *Software testing foundations: A study guide for the certified tester exam*. O'Reilly Media, Inc.

Abstract. Much time and effort is wasted both within and between industry, commerce, government and professional and academic institutions when ambiguities arise as a result of the inability to differentiate adequately between such terms as 'statement coverage' and 'decision coverage'; 'test suite', 'test specification' and 'test plan' and similar terms which form an interface between various sectors of society. Moreover, the professional or technical use of these terms is often at variance, with different meanings attributed to them.

Credibility. The International Software Testing Qualifications Board (ISTQB) was founded in November 2002 and is a not-for-profit association legally registered in Belgium. ISTQB (International Software Testing Qualifications Board) has defined the "ISTQB Certified Tester" scheme and is a leader in the certification of competences in software testing. ISTQB is an

organization based on volunteer work by hundreds of international testing experts.

Summary. The ISTQB Certified Tester exam helps establish an international standard for the growing professional software testing field. The ISTQB includes 47 international testing boards. Today more than 130,000 people have taken the International Software Testing Qualifications Board (ISTQB) Foundations Level exam. Professional testing of software has become an increasingly important task that requires a profound knowledge of testing techniques. The ISTQB has developed a universally accepted, international qualification scheme aimed at software and system testing professionals, and has created the syllabi and the tests for the "Certified Tester." With authors who are among the founders of the Certified Tester Syllabus, this thoroughly revised and updated third edition covers the "Foundations Level" (i.e., entry level) and teaches the most important methods of software testing. It is designed for self-study and provides the knowledge necessary to pass the Certified Tester: Foundations Level exam as defined by the ISTQB. Additionally, in this new edition, technical terms have been stated more precisely according to the revised and updated ISTQB glossary.

Stolberg, S. (2009). Enabling agile testing through continuous integration. In *Agile Conference, 2009. AGILE '09.* (pp. 369 –374). Presented at the Agile Conference, 2009. AGILE '09. Doi:10.1109/AGILE.2009.16

Abstract. A continuous integration system is often considered one of the key elements involved in supporting an agile software development and testing environment. For a traditional software tester transitioning to an agile development environment, improved development practices are necessary in order to make the transition to agile testing possible. Stolberg reports a continuous

integration implementation. The initial motivations for implementing continuous integration are discussed and a pre and post-assessment using Martin Fowler's" practices of continuous integration" is provided along with the technical specifics of the implementation. The report concludes with a retrospective of his experiences implementing and promoting continuous integration within the context of agile testing.

Credibility. Stolberg is a software quality tester at Pacific Northwest National Laboratory. Located in Richland, Washington, PNNL is one among ten U.S. Department of Energy (DOE) national laboratories managed by DOE's Office of Science. The PNNL research is designed to strengthen the U.S. foundation for innovation, and helps find solutions for not only the DOE, but for the U.S. Department of Homeland Security, the National Nuclear Security Administration, other government agencies, universities and industry.

Summary. The author identifies critical steps for a tester to transition from a traditional software-development approach to an agile one. He specifies automating acceptance tests at the application program interface level as being one of the most important practices possible by testers and programmers. Acceptance tests should be repeated at various stages of code integration and provide email notifications to testers. Otherwise, Stolberg has experienced increasing technical debt, whereby more manual tests have to be run at each stage of the code iteration and integration. He suggests several software tools to facilitate acceptance testing throughout the different testing stages.

XP inventor talks about agile programming. (2007, November 8). *InfoWorld*. Retrieved from http://www.infoworld.com/d/developer-world/xp-inventor-talks-about-agile-programming-565

Abstract. Kent Beck explains to *InfoWorld* the intentions of agile development. Beck emphasizes social skills of agile teams: integrity, transparency, accountability. Trends that drive agile programming include reliability, low cost of change, and an increased return on investment. He laments programmers and testers who accept high amounts of defects in their products especially during traditional software development. As well, Beck posits that agile development enables some projects to deservedly fail sooner. He also elucidates the concept of cowboy coding, where programmers obscure themselves and their work, perform heroically because they assume there's no one else that could have completed similar coding. In the short run cowboy coding may succeed but in the longrun it has huge risks and huge costs, huge hidden costs. Beck's informal definition of agile development is that "You [software developers] accept input from reality and respond to it. (*InfoWorld* 2007)"

Credibility. Beck is often credited with being a leading participant in the creation of the agile Manifesto, a foundational document in agile development. The document introduced the term in 2001 and he has refined his definition of the term in peer-reviewed and industry journals in the intervening years. He is highly credible on the topic although there were about 20 other contributors to the Agile Manifesto. Perched on the edge of Silicon Valley, InfoWorld has

chronicled and analyzed the development of new technology and the people who create it since 1978. Today, InfoWorld is targeted to IT decision makers seeking to modernize their operations using the latest technologies, architectures, and strategies (About InfoWorld, 2013).

Summary. Beck restates his belief in the momentum and validity of agile development. He helps lay the context for agile development and is a foundational source for many of the elaborations in this annotated bibliography. Beck's emphasis on team development has many applications for testers on agile teams. Beck notes that testers and programmers have the majority of the responsibility for creating, identifying, and fixing defects in software. He notes that high quality software is itself an important motivator for testers and programmers to improve their software practices.

Theme 3: Testing Activities in Addition to Verifying Business Test Cases

Ambler, S. (2007). Agile testing strategies. *Dr. Dobb's Journal: The World of Software Development*, *32*(1), 59-61. Retrieved from http://search.ebscohost.com.libproxy.uoregon.edu/login.aspx?Direct=true&db=cph&AN=23543448&login.asp&site=ehost-live&scope=site

Abstract. This article explores several strategies for independent testing (specifically, not done by programmers) on agile software development projects. Agile projects undergo an often short initiation phase where the foundation for the project is set, a construction phase where the system is developed in an evolutionary manner, an end game phase where the system is transitioned into production and a production phase where the system is operated and users are

supported. The majority of testing takes place during construction iterations on agile projects. There are two aspects to confirmatory testing, (a) agile acceptance testing and (b) developer testing, both of which are automated to allow continuous regression testing throughout the life cycle.

Credibility. Ambler has authored several books focused on the Disciplined Agile Delivery process decision framework, the Unified process, agile software development, the Unified Modeling Language, and CMM-based development. Ambler has a BSc in computer science and a MA in information science from the University of Toronto. Since 1990 he has worked in various roles: Software Engineer, Business Architect, System Analyst, System Designer, Project Manager, Smalltalk programmer, Java programmer, and C++ programmer. Further, he has led the development of several software processes, including Agile Modeling (AM), Agile Data (AD), Enterprise Unified Process (EUP), and Agile Unified Process (AUP) methodologies. Scott is a contributing editor with Dr. Dobbs Journal, and has written columns for Software Development, Object Magazine, and Computing Canada. Dr. Dobbs Journal is an online magazine, which caters to software engineers by providing practical solutions to real-world problems.

Summary. Ambler notes some of the important differences between testing in traditional SDLCs versus testing in Agile SDLCs. He confirms that testing is even more important in agile SDLCs. He states that though there are four stages of development, the majority of the testing takes place in the construction phase (see Figure 5). The testing effort like the software should evolve throughout construction. He makes the important suggestion that

functional and unit testing should be automated in order to be easily repeated throughout the evolution of the software. Agilists perform planning, and write documentation, but they focus on high-value activities such as actual testing. He states that independent testing is highly desirable on development projects and makes four suggestions to an independent testing team:

- Test as early as the potential impact of a defect often rises exponentially over time
- Test as often and effectively as possible, to increase the chance that you'll find defects. Early testing increases your costs in the short term, but studies have shown that greater investment in testing reduces the total cost of ownership of a system due to improved quality.
- Test enough for your situation: business software testing will require more testing than testing for a local Girl Scouts group.
- Pair testing is an exceptionally good idea. His general philosophy is that software development is a lot like swimming—it is very dangerous to do it alone.

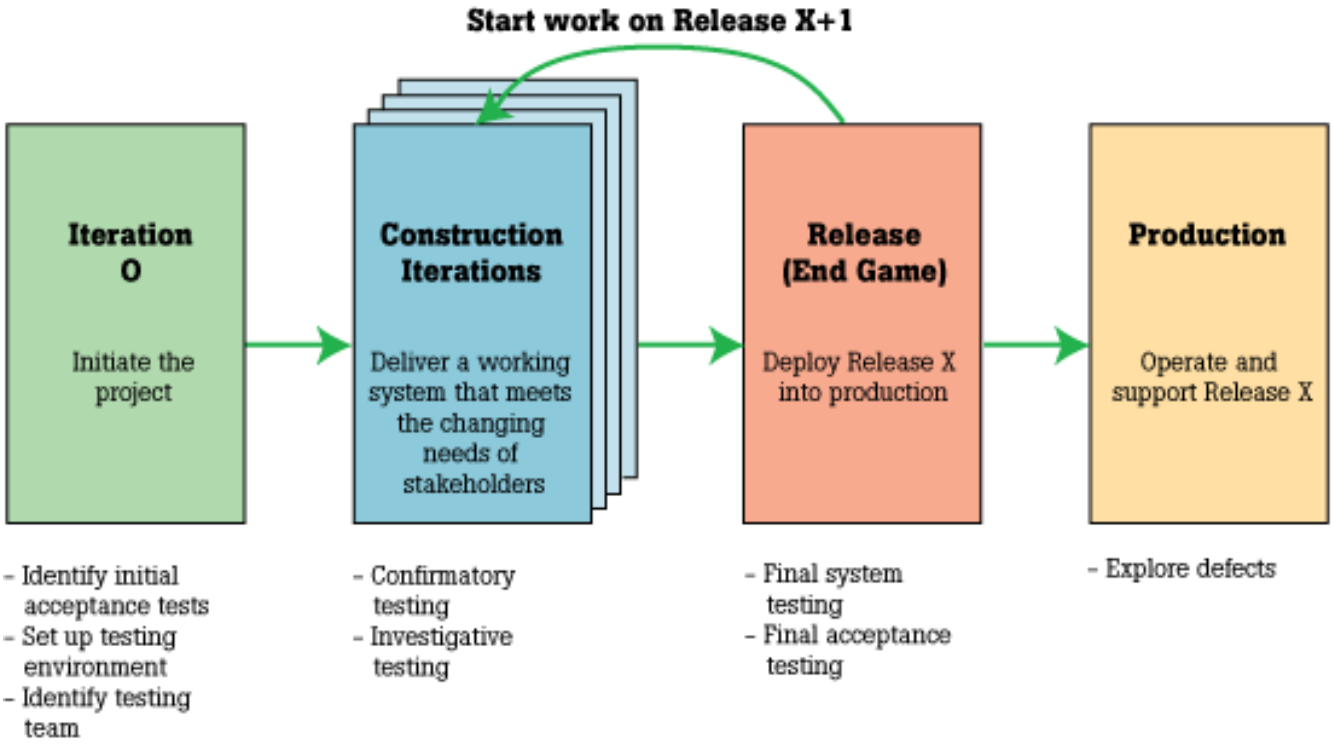

Figure 5. Independent testing throughout the agile development lifecycle.

Bavani, R. (2012). Distributed agile, agile testing, and technical debt. *IEEE Software, 29*(6), 28-33. Doi:10.1109/MS.2012.155

Abstract. Agile teams create business value by responding to changing business environments and delivering working software at regular intervals. While doing so, they make design tradeoffs to satisfy business needs such as meeting a release schedule. Technical debt is the result of such decisions or tradeoffs. When this happens, agile teams must pay off the accumulated debt by improving designs during subsequent iterations in order to improve maintainability. This must happen in a systematic way so that technical debt does not swell up

and damage the project. Accomplishing this is one of the major challenges in distributed agile projects. The scope of technical debt in software projects is spread across all areas including architecture, design, code, and test scripts.

Credibility. Johanna Rothman, an author and speaker, helps companies to improve how they manage their product development to maximize management and technical staff productivity and to improve product quality. She has authored several books, white papers, and articles. Crispin is an agile testing coach, author, and speaker. She specializes in showing testers and agile teams how testers can add value and how to guide development with business-facing tests. She has authored several books, white papers, and articles. Their article is published in the Business Intelligence Journal, which is not affiliated with any software vendors but provides business and IT consultants with a collection of IT information and resources.

Summary. Rothman and Crispin note that distributed agile teams are becoming more prominent in agile development because there are "smart people all over the world. However the authors warn about the potential risks of distributed teams. For example, if the developer in India checks in some code and the developers in Denver feel that the code isn't designed well then they might have to wait another day for the Indian developer to fix the code. Rothman and Crispin note that distributed teams should be cross-functional and should use face to face interactions at least at the beginning of their projects to establish acceptance criteria especially on technical debt. The authors suggest that all members on software development teams should be aware of technical debt and assist in ranking the debt. As well, spending short development

cycles exclusively on technical debt should increase the success of future business projects. As demonstrated in Figure 6, acceptable technical debt should be prioritized and noted in order to maintain predictability on future projects.

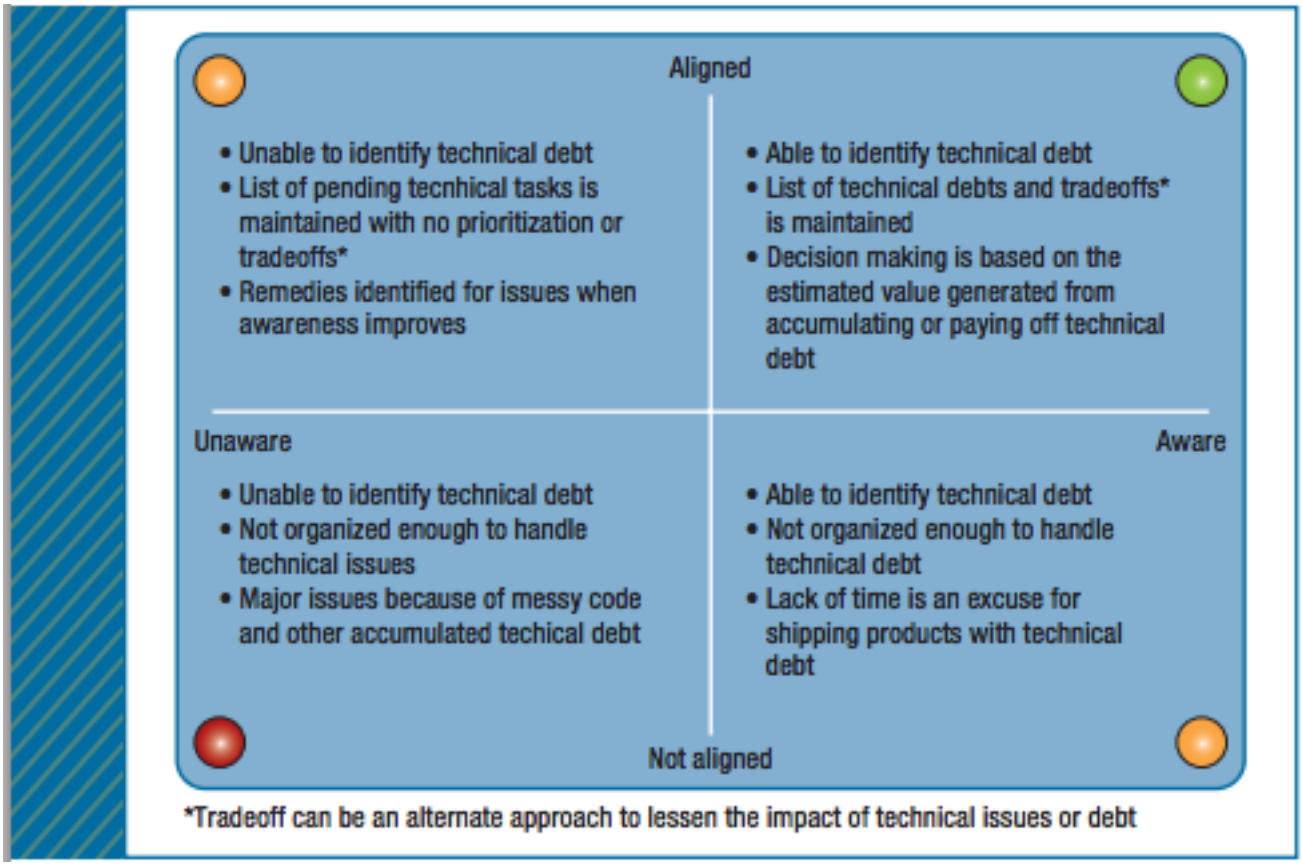

Figure 6. Technical debt quadrants.

Blaine, J. D., & Cleland-Huang, J. (2008). Software quality requirements: How to balance competing priorities. *IEEE Software*, *25*(2), 22–24. doi:10.1109/MS.2008.46

Abstract. The authors highlight the software industry's confusion around the quality demands of its products. The terminology around quality requirements is unclear. The elicitation, analysis, and specification of quality requirements involve careful balancing of a broad spectrum of competing priorities. Developers must therefore focus on identifying qualities and designing solutions that optimize the product's value to its stakeholders.

Credibility. Blaine is an independent software quality improvement consultant specializing in project management and quality, with specific expertise in value planning, requirements engineering, and software measurement. He received an MA in mathematics from Arizona State University and an MS in electrical and computer engineering from the University of California, Santa Barbara. His professional accreditations include the Project Management Institute's Project Management Professional certification and the American Society for Quality's Certified Software Quality Engineer certification. Jane Cleland-Huang is an assistant professor at DePaul University's College of Computing and Digital Media. Her research interests include requirements engineering with an emphasis on traceability and automated prioritization and triage. She coauthored Software by Numbers: Low-Risk, High-Return Development (Prentice Hall, 2003). She received her PhD in computer science from the University of Illinois at Chicago.

Summary. The article notes the general software development interest in quality. They suggest a hypothetical software that meets all the functional requirements of its customers might not be perfect. The software might be much slower than the customers expect it to be and there are other failures that the software might have. They note that integrity, interoperability,

performance, security, safety, usability, and testability are some the many possible quality requirements. Yet software quality attributes are often not balanced against functional requirements in software development. They note that software-development requirements lie along a range of quality attributes and should be considered in software design and analysis.

Farooq, U., & Azmat, U. (2009). Testing challenges in web-based applications with respect to interoperability and integration. Blekinge Institute of Technology, Sweden. Retrieved from http://www.bth.se/fou/cuppsats.nsf/all/c854314848360436c125754600502b83/$file/Testing_challenges_in_web-based_applications_with_respect_to_interoperability_and_integration.pdf

Abstract. Testing is one of the critical processes in software development life cycle. It plays a key role in the success of software product by improving its quality. Web-based applications are emerging and evolving rapidly; their importance and complexity is also increasing.

There are many testing challenges involved in Web-based applications. But most importantly interoperability and integration are the most critical testing challenges associated with Web-based applications. There are number of challenging factors involved in both integration and interoperability testing efforts. These integration and interoperability factors have almost 70 percent to 80 percent impact on overall quality of web-based applications. In the software industry different kinds of testing approaches are used by practitioners to solve the issues associated with integration and interoperability, which are due to ever increasing complexities of Web-based applications. Both integration and interoperability are inter-related and it is very helpful to cover all the possible issues of interoperability testing that will reduce the integration

testing effort. It will be more beneficial if a dedicated testing team is placed to perform the both integration and interoperability testing.

Credibility. Farooq and Azmat submitted the paper to the Department of Interaction and System Design, School of Engineering at Blekinge Institute of Technology (BTH) in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The BTH has ranked fifth in the world within Systems and Software Engineering.

Summary. Based on an examination of ten companies, the authors show that the most critical challenging factors of integration testing are inconsistent infrastructure and environment, and performance and reliability issues due to heterogeneity. Most of the companies addressed in this study do both integration and interoperability testing manually and not using automated testing tools. The results and their analysis show that the existing techniques are not enough to solve the challenging factors of integration and interoperability testing. The authors suggest that there should be a separate sub-testing team consisting of about three testers (depending upon the size and type of enterprise or organization) with multiple skills (multi-lingual expertise, standards, methodologies and tools) to handle the issues of interoperability and integration. Their research shows another opportunity for testers to improve the quality of software in the SDLC. Software testers should revisit testing software after the software has been integrated with other branches of software code. After the software codes have been integrated there may be coding conflicts that can be vetted out by a combination of automated and manual tests.

Green, R. R., Mazzuchi, T. T., & Sarkani, S. S. (2010). Communication and quality in distributed agile development: an empirical case study. *World Academy Of Science, Engineering & Technology*, *61*322-328.

Abstract. Through inward perceptions, we intuitively expect distributed software development to increase the risks associated with achieving cost, schedule, and quality goals. To compound this problem, agile software development (ASD) insists one of the main ingredients of its success is cohesive communication attributed to collocation of the development team. The following study identified the degree of communication richness needed to achieve comparable software quality (reduce pre-release defects) between distributed and collocated teams. This paper explores the relevancy of communication richness in various development phases and its impact on quality. Through examination of a large distributed agile development project, this investigation seeks to understand the levels of communication required within each ASD phase to produce comparable quality results achieved by collocated teams. Obviously, a multitude of factors affects the outcome of software projects. However, within distributed agile software development teams, the mode of communication is one of the critical components required to achieve team cohesiveness and effectiveness. As such, this study constructs a distributed agile communication model (DAC-M) for potential application to similar distributed agile development efforts using the measurement of the suitable level of communication. The results of the study show that less rich communication methods, in the appropriate phase, might be satisfactory to achieve equivalent quality in distributed ASD efforts.

Credibility. Green is a System Engineering doctoral student at The George Washington University, Washington, DC. Mazzuchi is the Chair of the Department of Engineering Management and Systems Engineering at The George Washington University, Washington, DC .Sarkani is a Professor of Engineering Management and Systems Engineering at The George Washington University, Washington, DC. The *World Academy Of Science, Engineering & Technology* is an open access journal of peer-reviewed scientific research.

Summary. Agile software development generally advocates the collocation of team members in order to increase effective communications. Communications with programmers, especially in the sprint phase for testers, should help meet cost, schedule, and quality goals. However to meet global software development opportunities, agile teams need to responsibly deal with distributed teams. The authors tracked project media richness as high, medium, and low including: face-to-face, video teleconference, teleconference/phone/skype/instant messenger, podcast/recorded webcast/web-based tracking tool, and email/documentation/wikis. Distributed teams introduce cultural and time zone issues. However, the authors demonstrate that face-to-face communications at the beginnings of agile products are most important for the success of software development processes. In Table 3 the authors summarize the defect rate, and software lines of code found in different phases of development, with higher communication scores of 5 going to the collocated teams. As well, distributed talents may, at times, surpass face-to-face teams. Testers might do well to keep these communication practices in mind, especially for phase 1 and phase 3 (See Figure 7): in phase 1, the requirements are documented and communicated and help testers be ready for confirming

functional requirements; in phase 3, programmers and testers most likely develop the nonfunctional requirements of the software.

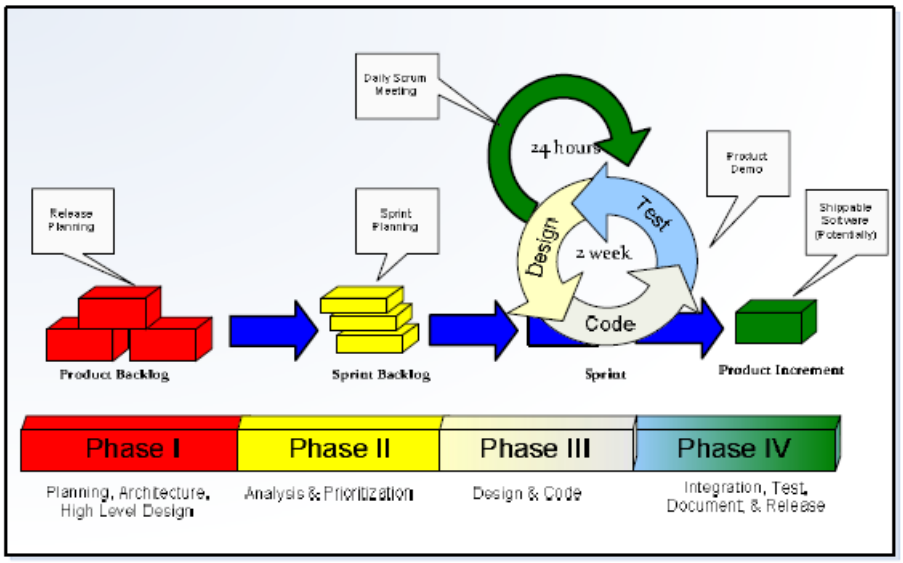

Figure 7. Phases of agile software-development.

Table 3

Defect quantities of collocated and distributed teams and their software lines of code.

Method	Phase 1	Phase 2	Phase 3	Phase 4	Defects	SLOC
Collocated	5	5	5	5	713	700,000
Distributed	2.86	2.85	2.78	2.30	635	837,375

Hong, W., Thong, J. Y., Chasalow, L. C., & Dhillon, G. (2011). *User acceptance of agile information systems: A model and empirical test. Journal of Management Information Systems, 28*(1), 235-272.

Abstract. Agile Information Systems are characterized by frequent upgrades with a small number of new features released periodically. The existing research on agile Information Systems (IS) has mainly focused on the developers' perspective with little research into end users' responses to these agile IS. The authors investigate data from 477 users with focus not only users' intentions to continue using the agile IS but also their intentions to use new features.

Credibility. Weiyin Hong is an associate professor in the Department of Management Information Systems, University of Nevada, Las Vegas. Her work has appeared in Information Systems Research, Journal of Management Information Systems, International Journal of Human–Computer Studies, etc. James Y.L. Thong is a professor of information systems .Lewis C. Chasalow is an assistant professor of business at the University of Findlay, Ohio. Gurpreet Dhillon is a professor of information systems. The Journal of Management Information Systems

is a peer-reviewed publication.

Summary. The study suggests that software testers should more exhaustively consider the comfort level of customers when approving new upgrades to software. Citing psychological theory, the authors note that customers place higher costs on learning a system than developers typically recognize. Developers tend to focus on the virtues of new features. A consistent system layout and consistence with user knowledge help customers with upgrades and increases the likelihood of customers adopting new features. The authors note that past system upgrades determine the likelihood of the customers to continue using a service.

Itkonen, J., Mäntylä, M., & Lassenius, C. (2012) The role of the tester's knowledge in exploratory software testing. *Software Engineering, Ieee Transactions.* PP *99.*
doi: 10.1109/TSE.2012.55

Abstract. The authors present a field study on how testers use knowledge while performing exploratory software testing in industrial settings. The authors video recorded 12 testing sessions in four industrial organizations, having their subjects think aloud while performing their usual functional testing work. Testers recognize failures based on their personal knowledge without detailed test case descriptions.

Credibility. Juha Itkonen works as a post-doc researcher at the Department of Computer Science and Engineering, Aalto University School of Science, Finland. Mantyla is a post-doc researcher at Aalto University, Finland. Casper Lassenius is a professor at Aalto University

School of Science, Department of Computer Science and Engineering. Their article is published in the peer-reviewed IEE Transactions on Software Engineering.

Summary. The authors observe that three main kinds of knowledge help testers in their exploratory testing: domain knowledge, system knowledge, and general software engineering knowledge. The authors find that testers often applied their knowledge either as a test oracle (see Table 4) to determine whether a result was correct or not, or for test design, to guide them in selecting objects for test and designing tests. Interestingly, a large number of failures, are found outside the actual focus areas of testing as a result of exploratory investigation. The authors conclude that the way exploratory testers apply their knowledge for test design and failure recognition differs clearly from the test-case based paradigm and is one of the explanatory factors of the effectiveness of the exploratory testing approach.

Table 4

Categories of knowledge used for recognizing failures in software.

Knowledge category and perspective		Knowledge type and how it was applied
Domain knowledge	Users' perspective	Episodic knowledge of usage procedures and context • Simulating realistic usage tasks (design) • Using the system to prepare tests and data (design)
		Conceptual knowledge of the information content and presentation in usage context • Evaluating test results and outputs (oracle)
		Knowledge of problems in customer cases • Testing for specific risks (design)
	Application domain perspective	Conceptual knowledge of the subject matter • Evaluating test results and outputs (oracle)
		Practical knowledge of the subject matter and tools • Creating reference results for tests (oracle)
System knowledge	Interacting features and system perspective	Knowledge of system's working mechanisms, logic, and interactions • Simulating realistic usage tasks (design) • Observing the overall response of the system to changes in configuration, state, or data (oracle) • Recognizing failures outside the actual testing scope, e.g., in other features (oracle) • Comparing to similar features (oracle)
		Knowledge of past failures • Recognizing familiar symptoms (oracle) • Testing for frequently occurring failure types (design)
	Individual features and functional perspective	Knowledge of features and views of the system • Visual inspection of GUI or a report (oracle) • Comparing to earlier behavior (oracle) • Recognizing failures outside the actual testing scope, e.g., in other features (oracle)
		Knowledge of the detailed technical aspects • Systematic searching for errors in logs (design) • Investigating error messages or suspicious results and log messages (oracle)
Generic knowledge	Generic correctness perspective	Knowledge of software user interfaces and presentation • Visual inspection of GUI or a report (oracle) • Evaluating test results and outputs (oracle) • Recognizing failures outside the actual testing scope, e.g., in other features (oracle)
	Usability perspective	Practical knowledge of usability of software systems • Recognizing failures outside the actual testing scope, e.g., in other features (oracle)
	Direct failure perspective	Practical knowledge to recognize crashes and error messages (oracle)

Nommesen, P., & Macfie, A. (2012). Using pragmatic testing to ensure project success. *Business*

Intelligence Journal, *17*(3), 40-47. Retrieved from http://search.ebscohost.com.libproxy.uoregon.edu/login.aspx?direct=true&db=cph&AN=80178600&login.asp&site=ehost-live&scope=site

Abstract. The authors of *Pragmatic Testing* are concise in their explanation of simple testing strategies for business intelligence environments. They believe successful business intelligence organizations should perform basic data tests routinely. They note that business stakeholders often wrongly neglect testing in software development cycles. They emphasize the need for high quality software. "BI teams must deliver business intelligence applications that meet the needs of the users and key stakeholders. Identifying errors as early as possible is the least expensive way to ensure program quality, and pragmatic testing is valuable because it is used at the beginning of development, at regular intervals throughout a BI project, and to continually mirror the end users' goals" (Nommesen, & Macfie, p. 40, 2012)."

Credibility. Peter Nommesen, is a senior consultant with BusinessMinds consulting company Australia. He has worked in the IT industry for 24 years in various roles including project manager, data modeler, business intelligence consultant, system architect, and software developer. Alison Macfie is a marketing and communications manager with BusinessMinds Australia.

Summary. The authors state that well-thought out quick tests (sprints) could alleviate important defects in delivered products. Pragmatic tests employ Structured Query Language (SQL) to compute: individual record counts, combining records to groups and inspecting the counts,

counting null data values, checks the history of data records do not conflict, and comparing information in single--instead of linked--data tables. A few other important practices include: (a) establishing a framework from which to run tests; (b) developing fast tests; (c) documenting test findings; (d) reusing test scripts; (e) cordoning testing from other network resources; and (f) advocating a testing mind-set among all stakeholders.

Roberts, N. (2012). Leveraging information technology infrastructure to facilitate a firm's customer agility and competitive activity: an empirical investigation. *Journal Of Management Information Systems, 28*(4), 231-270.

Abstract. This paper investigates how information technology (IT) facilitates a firm's customer agility and, in turn, competitive activity. Customer agility captures the extent to which a firm is able to sense and respond quickly to customer-based opportunities for innovation and competitive action. Drawing from the dynamic capability and IT business value research streams, the authors propose that IT plays an important role in facilitating a "knowledge creating" synergy derived from the interaction between a firm's web-based customer infrastructure and its analytical ability.

Credibility. Nicholas Roberts is an assistant professor in the Johnson College of Business and Economics at the University of South Carolina Upstate. He received his PhD in management information systems from Clemson University. His research interests include IT value, organizational learning, and health IT. The *Journal of Management Information Systems* is a peer-reviewed quarterly journal for field of managing of

information systems.

Summary. The paper highlights an area for software testing opportunities. The authors show that a web-based customer infrastructure facilitates a firm's customer-sensing capability; furthermore, analytical ability positively moderates this relationship. They find that internal systems integration positively moderates the relationship between interfunctional coordination and a firm's customer-responding capability. They show that action efficacy is highest when sensing and responding capabilities are both high**.** Agile team members, including testers, help develop a sense of customer expectations.

Conclusion

This annotated bibliography synthesizes 31 selected references on b2b software quality roles for agile testing professionals. Agile software development practices have the allure of responding to customer requests and they are a popular SDLC of software development companies (Ambler, 2007). It is important to note that software updates are a critical opportunity to ensure continued customer engagement and to increase customer satisfaction (Ambler, 2007). Still, software-development organizations might be well served to lower the learning costs of customers in order to increase the satisfaction of the customers with release offerings (Molinari et al., 2008). As well, development teams should be aware of the customer expectations in the marketplace (Molinari et al., 2008). In *Conceptual Foundations of Human Computer Interaction and Software Engineering,* the authors advocate digital technologies which people interact with through sensitive, considerate human-centered design (Imaz & Benyon 2006).

Ambler (2007) and Heusser (2012) note the importance for context specific testing. Bertoloni (2007) explains that academic hopes for testing, many of which involve leveraging programming, simply are not feasible in businsess SDLC contexts. Research questions of this study are designed to address the needs of software testers in organizations transitioning from traditional SDLCs to agile SDLCs. This discussion is built around three main themes: (a) the role of software testers in agile software teams in a b2b context (Crispin & Gregory, 2009) (b) appropriate amounts of software tester readiness and planning (Heusser & Kulkarni 2011); and (c) testing activities in addition to verifying business test cases (Kadry, 2011; Nommesen & Macfie, 2012).

Table 5 provides an overview of the findings in the selected references. The fields marked "Present" indicate that an aspect of one of the three themes is addressed in the reference. The themes are broken into eight components. The table is presented to help testers and other business-to-business stakeholders locate specific content within the selected literature. The information provides a thematic map, meant to display the distribution of variables but not locate them with any degree of precision in the references. Most of the literature in this annotated bibliography emphasizes the role of the tester (see p. 86).

Table 5

Coding Table: Findings in the selected references.

Legend
P: Present
N: Not present

	Theme (a) the role of software testers in Agile software teams in a b2b context			Theme (b) appropriate amounts of software tester readiness and planning;		Theme (c) testing activities in addition to verifying business test cases.		
Author	business-to-business	Agile	tester	readiness	planning	quality	testing	test cases
Ahamed, S. S. R. (2010).	N	N	P	N	P	P	P	P
Ambler, S. (2007).	N	P	N	N	P	N	P	N
Filieri, A.,et al (2012).	N	P	N	N	N	N	N	N
Heusser, M. (2012, November).	N	P	P	N	N	P	P	N
Mairiza, D., et al (2010).	N	N	N	N	N	P	N	N
Mauldin, E., et al (2006).	P	N	N	N	N	P	N	N
Molinari, L. K. Et al (2008).	P	N	N	N	N	P	N	N
Pantouvakis, A. (2011).	P	N	N	N	N	P	N	N
Patcha, K. K. (2009).	P	P	P	N	N	P	P	P
Quality Assurance Institute. (2006).	N	P	P	P	P	P	P	P
Zhang, Y. et al. (2011).	P	N	N	N	N	P	N	N
Bertolino, A. (2007).	N	P	P	N	P	P	P	P
Bruns, A., et al. (2009, September).	N	N	P	N	N	P	P	N
Crispin, L. & Gregory, J. (2009).	N	P	P	P	P	P	P	P
Heusser, M., & Kulkarni, G. (Eds.). (2011).	N	P	P	P	P	P	P	P
Jaffar-ur Rehman, M., et al (2007).	N	N	P	N	N	P	P	P
Kadry, S. (2011).	N	N	P	N	N	P	P	P
Li, et al. (2012).	N	N	P	N	P	P	P	P
Roseberry, W. (2012, November).	N	N	P	N	N	P	P	P
Spillner, A., Linz, T., & Schaefer, H. (2011).	N	P	P	N	P	P	P	P
Stolberg, S. (2009).	N	P	P	N	P	P	P	P
Beck (2007)	N	P	P	N	P	N	P	N
Ambler, S. (2007).	N	P	N	N	P	P	P	N
Bavani, R. (2012).	N	P	P	N	P	P	P	N
Blaine, J. D., & Cleland-Huang, J. (2008).	N	P	N	N	N	P	P	N
Farooq, U., & Azmat, U. (2009).	P	N	N	N	P	P	P	N
Green, R. R. Et al (2010).	N	P	N	P	N	P	P	N
Hong, W. Et al (2011).	N	P	N	N	P	N	P	N
Itkonen, J. Et al. (2012)	N	P	P	P	P	P	P	P
Nommesen, P., & Macfie, A. (2012).	N	P	N	N	N	P	P	N
Roberts, N. (2012).	N	P	N	N	N	P	N	N

Theme 1: The Role of Software Testers in Agile Software Teams in a b2b Context

Blaine et al. (2008) assert that software quality demands are often ignored in the design and implementation of software, partly because it is hard to prioritize the importance of nonfunctional requirements. Because, software applications are more dynamic and context specific than manufactured goods, such as automobiles, the software applications can benefit significantly from independent testers. Authors cited in this annotated bibliography find that testers are at least as important as programmers in agile SDLCs (Crispin & Gregory, 2009) because testers are important in b2b traditional software design life cycles (Mauldin et al. 2006). Molinari et al. (2008) state that satisfaction occurs for b2b customers when a service is delivered correctly the first time.

The tester's knowledge includes domain knowledge, system knowledge, and general software engineering knowledge. Testing experience could be gained through testing, attending seminars, and working with external consultants. Itkonen et al. (2012) states that testers detect software defects about 70% of the time, which is higher than the alternatives. The authors suggest that software teams include testers of long and short tenures, because testers of different experiences levels report different classes of software defects. Using checklists reduced the number of missing categories and all types of mistakes.

Crispin and Gregory (2009) suggest that interpersonal skills are very important for the role of software testers. They suggest that software testers strategically leverage the programming expertise of their coworkers in order to provide added value to the customer team's goals. As well, Crispin and Gregory (2009) note in Figure 4 (see p. 52) that tester roles frequently overlap with both domain experts and programmers. Beck (2007) further emphasizes the team work component of agile team

members and asserts that agile teams are most efficient through integrity, transparency, and accountability.

Web-based software often has the ability to provide high quality data about customers and increase the customer-sensing capability of testers (Roberts, 2012). The costs of firing the testers include additional technical debt and slowing development lifecycles (Heusser & Kulkarni, 2011). Addressing technical debt requires a matrix (see Figure 6, p. 70) of communication among various development stakeholders (Bavani, 2006). Testers are well-positioned to navigate the complex communication because testers are members of both customer teams and development teams (Crispin & Gregory, 2009). Jaffar-ur Rehman et al. (2007) assert that reuse of programming components does not reduce the need for testing as application domains are dynamic. Customer tests in b2b development require higher quality of service requirements than b2cs (Filieri et al. 2012). Filieri et al. (2012) also note that internet-based software exposes business customers to extra risks. Further, the authors suggest that business customers often need concrete guarantees about the reliability of internet software. Crispin and Gregory (2009) note that most business facing tests should include some manual testing (see Figure 2, p. 48) Despite the ambiguity of these nonfunctional quality requirements, software testers are tasked with the responsibility for these quality goals.

Software programmers tend to believe that all the new software features (functional requirements) will be welcomed and applied by business customers (Hong et al., 2011). However, business customers are wary of the time and learning costs of software updates, which nonfunctional requirements can ameliorate (Patcha, 2009). According to Mairiza et al. (2010) performance and usability are the main nonfunctional quality factors in business software systems. Interoperability and

data integrity are also paramount to business customers (Patcha, 2009). Testers and programmers would be well served to communicate their strategies, as Beck (2007) notes that testers and programmers have the majority of the responsibility for creating, identifying, and fixing defects in software. According to Mairiza et al. (2010) performance and usability are the main nonfunctional quality factors in business software systems. Interoperability and data integrity are also paramount to business customers (Patcha, 2009).

Quality assurance testers and computer hardware can help evaluate performance attributes, measurable by factors such as response time and latency. However, for the second most important nonfunctional requirement of usability, quality assurance testing is most efficient in improving the software's learnability, understandability, operability, attractiveness, memorability, efficiency etc. (Crispin & Gregory 2009; Mairiza et al., 2010). As the testing quadrant of Crispin and Gregory (2009) and the automation decision table of Kadry (2011) show, programming and automation decisions should be compared to business goals in order to help testing teams. Kadry (2011) cautions developers that automated tests have time costs for development, maintenance, and execution.

Theme 2: Appropriate Amounts of Software Tester Readiness and Planning

Software test planning with many test cases by themselves is not among the most critical aspects of testing goals (Bertolino, 2007; Crispin & Gregory 2009; Heusser & Kulkarni, 2011). Software test planning does not reduce the severity of internal defects and only sometimes prevents the release of external defects (Heusser & Kulkarni, 2011). However, it is important that testers are ready to test when they receive testable code. Heusser (2012) suggests that testers participate in software development brainstorming sessions to raise the development team's awareness of risks from system

and user perspectives. Heusser and Kulkarni (2011) suggest testers create nimble test plans which rely on brief software functional requirements and the knowledge of the testers.

Test readiness also entails some project-management and coordination among various software development resources (Heusser & Kulkarni, 2011). Farooq and Azmat (2009) show that the most challenging factors of integration testing are inconsistent infrastructure and environment. Heusser & Kulkarni (2011) further prioritizes the five resources that affect test readiness:

1. Environment—The specific system hardware and software that are important to the project's testing effort.

2. Configuration and change management—The regulation of changes made to hardware, software, firmware, data, databases, documentation, test tools/fixtures, and test documentation throughout the project's life cycle.

3. Documentation—The sum of requirements, test plans, use cases, user stories, test scenarios, cases and scripts that make up the bulk of any test team's documents.

4. Data—The two areas of "data" to be concerned with (a) databases and (b) the data referenced in the test scripts.

5. Test tools-These can be scripting tools such as Excel macros, or capture/playback tools used for regression testing, load and stress testing software, simulators, and tracking tools.

System specific documentation may especially help software development teams communicate more efficiently. Rothman and Gregory (see Figure 6, p. 70) suggest that business stakeholders, testers, and programmers must communicate efficiently in order to be able to identify technical debt, list tradeoffs of technical debt, and conduct decisions based on the value of releasing software versus the

cost of accumulating technical debt. Otherwise, software development lifecycles slow and development efforts encounter major issues that cause customer dissatisfaction. These issues might include slow performing websites, confusing graphical user interfaces, messy coding, inaccurate data tables, or hardware failures (Bavani, 2012). Green (2010) suggests that in-person communications are most helpful at the beginning of software projects. High quality software practices weed out the most severe defects at this stage of development (Bavani, 2012). Much of the interaction between programmers and testers that occurs in the construction phase of the development is later concerned with the less severe defects (Ambler, 2007).

Barkley states that completing software correctly the first time is inexpensive; however, completing software incorrectly is costly (as cited in Heusser & Kulkarni, 2011). Moreover, Mauldin et al. (2006) suggest that quality assurance services to customers increase their trust in the software. According to Pantouvakis (2011), the products of b2b companies are even more profitable when companies increase the job satisfaction of each member of the service delivery team. Ultimately each b2b agile tester hones their skills for their specific contexts (Heusser, 2012).

Theme 3: Testing Activities in Addition to Verifying Business Test Cases

Observed directly, testers in several companies did not rely on test cases or heavily documented test cases (Itkonen et al., 2012). Instead, several authors (Itkonen et al., 2012; Crispin & Gregory 2009; Heusser & Kulkarni, 2011) suggest that manual exploratory tests often have a high degree of success due to the tester's knowledge and questioning of implicit assumptions in software development. Exploratory testing also seems suited for agile development, which avoids the excessive planning and documentation of traditional software development lifecycles. As well, exploratory testing

accomplishes several objectives at the same time: it educates the tester and often ferrets out the most important functional and nonfunctional product defects. Technical knowledge is particularly important in the agile development context; defect clues could be found by using programming to examine the internal state of the software system (Itkonen et al., 2012). Exploratory testing should be differentiated from manual testing via a script, because manual testing via a script can be especially unproductive (as edited by Heusser & Kulkarni, 2011). Figure 1 (see p. 25) shows that integration, validation, and system tests lie in the purview of software testers and should occur throughout the SDLC.

Itkonen et al. (2012) note: "...test automation has been the focus of a great deal of research, manual testing is still widely utilized and appreciated in the software industry, and is unlikely to be replaced by automated testing in the foreseeable future (p. 1). " However, automated tests are best suited for any repeated tests Kadry (2011) after testers consider nine indicator questions for automation. Bertolino (2007) highlights the dream of total automation. As noted more recently by testing consultant James Marcus Bach (Twitter, 2013), "I write tools to help me test. That does not automate my testing any more than driving to work automates my job." Stolberg (2009) suggests that automated functional (acceptance) tests are most useful at the beginning of SDLCs before tests are repeated. The Selenium web-based software is an open-source tool that can be leveraged to mimic business customers (Bruns et al., 2009). Stolberg (2009) specifies that automating acceptance tests at the application program interface level is one of the most important practices of testers and programmers.

Automated regression tests are very helpful at protecting software systems, but there is the danger of automated tests becoming inefficient as well. Ahamed (2009) observes that automated system tests are most effective in the late phases of testing. Kadry (2011) suggests that test suites should be

prioritized in the order that they are likely to discover defects. Automated tests should also be strategically implemented according to the costs and benefits of each automated test

The literature selected for this study supports that notion that testing efforts are never exhaustive (Kadry, 2011; Nommesen & Macfie, 2012). Instead, certain testing practices identify more important defects and increase the development team's confidence in the software. Ahamed (2009) and Crispin & Gregory (2009) suggest testing should be repeated throughout the software development life cycle as soon as deliverable parts of the software are coded. These tests will in turn facilitate the delivery of agile software. Ambler (2007) and Beck (2007) offer pair testing as another effective testing strategy. Pragmatic testing, suggested by Nommesen and Macfie (2012), leverages data aggregation strategies to ensure the integrity of the data of software applications. Instead of exhaustive comparison of individual records for example, the authors suggest that testers compare the sum of all the records. Kelly (as edited by Heusser & Kulkarni, 2011) states that exploratory testing allows testers to interact dynamically with the software in timed and documented sessions to identify any risks that likely diverge from the functional requirements and test cases. Science-based testing also suggests that changing one or two features of a software application at a time increases the effectiveness of bug detection. Testers might be best served with highly responsive testing environments for exploratory tests.

References

About. (n.d.). *Dewi MAIRIZA's Homepage*. Retrieved January 22, 2013, from http://mairiza.wordpress.com/about/

About InfoWorld. (n.d.). Retrieved January 20, 2013, from http://www.infoworld.com/about

About Lisa Crispin - Agile Testing with Lisa Crispin. (n.d.). *Agile Testing with Lisa Crispin*. Retrieved January 20, 2013, from http://lisacrispin.com/about/

Ahamed, S. S. R. (2009). Studying the feasibility and importance of software testing: An analysis. Retrieved from http://arxiv.org/abs/1001.4193

Ambler, S. (2007). Agile testing strategies. *Dr. Dobb's Journal: The World of Software Development*, *32*(1), 59-61. Retrieved from http://search.ebscohost.com.libproxy.uoregon.edu/login.aspx?direct=true&db=cph&AN=23543448&login.asp&site=ehost-live&scope=site

Ambler, S. (2007). Survey says...agile has crossed the chasm. *Dr. Dobb's Journal: The World of Software Development*, *32*(8), 59-61. Retrieved from Computer Source database: http://search.ebscohost.com.libproxy.uoregon.edu/login.aspx?direct=true&db=cph&AN=25734588&login.asp&site=ehost-live&scope=site

Andrade, J., Ares, J., Martinez, M., Pazos, J., Rodriguez, S., et al. (2013). An architectural model for software testing lesson learned systems. *Information & Software Technology*, *55*(1), 18-34. doi:10.1016/j.infsof.2012.03.003

Armbrust, A., et al. (2009). Above the clouds: A Berkeley view of cloud computing. Technical report, Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley.

Bach, J. M. (2013, February 6). I also write tools to help me test. That does not automate my testing any more than driving to work automates my job. @jamesmarcusbach. microblog. Retrieved from https://twitter.com/jamesmarcusbach/status/299040292588380160

Bastek, N. (2012). Review essays for the biological sciences. Writing@CSU. Colorado State University. Available at http://writing.colostate.edu/guides/guide.cfm?guideid=79.

Bavani, R. (2012). Distributed agile, agile testing, and technical debt. *IEEE Software, 29*(6), 28-33. doi:10.1109/MS.2012.155

Bell & Frantz. (2012). Critical evaluation of information sources. (n.d.). Retrieved from http://library.uoregon.edu/guides/findarticles/credibility.html

Bertolino, A. (2007). Software testing research: Achievements, challenges, dreams. In *Future of Software Engineering, 2007. FOSE '07* (pp. 85 –103). Presented at the Future of Software Engineering, 2007. FOSE '07. doi:10.1109/FOSE.2007.25

Blaine, J. D., & Cleland-Huang, J. (2008). Software quality requirements: How to balance competing priorities. *IEEE Software*, *25*(2), 22–24. doi: 10.1109/MS.2008.46

Bruns, A., Kornstädt, A., & Wichmann D,. (Sept.-Oct 2009). "Web Application Tests with Selenium," IEEE Software, *26(*5), (pp. 88-91) doi:10.1109/MS.2009.144

Busch, C. et al. (2012). Content Analysis. Writing@CSU. Colorado State University. Retrieved from http://writing.colostate.edu/guides/guide.cfm?guideid=61

Business-to-business. (n.d.). In *Wikipedia*. Retrieved January 13, 2013, from

http://en.wikipedia.org/w/index.php?title=Business-to-business&oldid=532935943

Capability Maturity Model. (n.d.). In *Wikipedia*. Retrieved January 20, 2013, from http://en.wikipedia.org/w/index.php?title=Capability_Maturity_Model&oldid=532267195

Cheslack-Postava, F., Gibbs F., Kornblith, S., & Stillman, D. (2012). Zotero standalone for Windows (Version 3.0.3) [Software]. Available from http://www.zotero.org/support/3.0

Crispin, L. & Gregory, J. (2009). Agile testing a practical guide for testers and agile teams. Boston, MA: Addison Wesley.

Deming, W. E. (1986). *Out of the crisis*. Cambridge, Mass.: Massachusetts Institute of Technology, Center for Advanced Engineering Study.

Dr. Dobb's | Good stuff for serious developers: Programming Tools, Code, C++, Java, HTML5, Cloud, Mobile, Testing. (n.d.). *Dr. Dobb's*. Retrieved January 20, 2013, from http://www.drdobbs.com/aboutus

Farooq, U., & Azmat, U. (2009). Testing challenges in web-based applications with respect to interoperability and integration. Blekinge Institute of Technology, Sweden. Retrieved from http://www.bth.se/fou/cuppsats.nsf/all/c854314848360436c125754600502b83/$file/Testing_challenges_in_web-based_applications_with_respect_to_interoperability_and_integration.pdf

Few, S. (2009). *Now You See It: Simple Visualization Techniques for Quantitative Analysis* (1st ed.). Analytics Press.

Filieri, A., Ghezzi, C., & Tamburrelli, G. (2012). A formal approach to adaptive software: continuous assurance of non-functional requirements. *Formal Aspects of Computing*, *24*(2), 163–186. doi: 10.1007/s00165-011-0207-2

Guide for Authors . (n.d.) Retrieved January 26, 2013 from.http://www.elsevier.com/journals/international-journal-of-accounting-information-systems/1467-0895/guide-for-authors

Golden, B. (Jul 16, 2010) Cloud computing: Two kinds of agility. http://www.cio.com/article/599626/Cloud_Computing_Two_Kinds_of_Agility.

Green, R. R., Mazzuchi, T. T., & Sarkani, S. S. (2010). Communication and quality in distributed agile development: an empirical case study. *World Academy Of Science, Engineering & Technology*, *61*322-328.

Harter, D. E., Krishnan, M. S., & Slaughter, S. A. (2000). Effects of process maturity on quality, cycle time, and effort in software product development. *Management Science*, *46*(4), 451–466.

Harty, J. (2011). Finding usability bugs with automated tests. *Commun. ACM*, *54*(2), 44–49. doi:10.1145/1897816.1897836

Heusser, M. (2012, November). *A brief history of the quality movement and what software should do about it*. 2012 Pacific Northwest Software Quality Conference Portland, OR. Paper retrieved from http://www.pnsqc.org/2012-conference/papers-and-presentations

Heusser, M., & Kulkarni, G. (Eds.). (2011). *How to reduce the cost of software testing (1st ed.)*. Boca Raton, FL: Auerbach Publications.

Hong, W., Thong, J. Y., Chasalow, L. C., & Dhillon, G. (2011). *User acceptance of agile information systems: A model and empirical test. Journal of Management Information Systems, 28*(1), 235-272.

Imaz, M., & Benyon, D. (2006). *Designing with Blends: Conceptual Foundations of Human-Computer Interaction and Software Engineering*. The MIT Press.

International Journal of Engineering Science and Technology (IJEST™). Welcome to IJEST™! January 20, 2013 from http://www.ijest.info

Itkonen, J., Mäntylä, M., & Lassenius, C.. (2012) The role of the tester's knowledge in exploratory software testing. *Software Engineering, Ieee Transactions.* PP *99*. doi: 10.1109/TSE.2012.55

Janet Gregory, Consultant. (n.d.). Retrieved January 20, 2013, from http://www.janetgregory.ca/

Jyothi, V., & Nageswara Rao, K. (2012). Effective implementation of agile practices – in coordination with lean kanban. *International Journal on Computer Science & Engineering*, *4*(1), 87-91. Retrieved from http://www.enggjournals.com/ijcse/doc/IJCSE12-04-01-126.pdf

Kadry, S. (2011). A new proposed technique to improve software regression testing cost. The *International Journal of Security & Its Applications (IJNSA) 5(*3), 46-58 Retrieved from http://arxiv.org/abs/1111.5640.

Li, J., Stephanie, T., Conradi, R., & Kristiansen, J. M. W. (2012). Enhancing defect tracking systems to facilitate software quality improvement. *IEEE Software*, *29*(2), 59–66. doi:10.1109/MS.2011.24

Literature Reviews. (n.d.). *The Writing Center*. Retrieved January 11, 2013, from http://writingcenter.unc.edu/handouts/literature-reviews/

Maantay, J., & Ziegler, J. (2006). *Gis for the Urban Environment*. Esri Press.

Mairiza, D., Zowghi, D., & Nurmuliani, N. (2010). An investigation into the notion of non-functional requirements. In *Proceedings of the 2010 ACM Symposium on Applied Computing* (pp. 311–317). New York, NY, USA: ACM. doi:10.1145/1774088.1774153

Mauldin, E., Nicolaou, A., & Kovar, S. (2006). The influence of scope and timing of reliability assurance in b2b e-commerce. *International Journal of Accounting Information Systems, 7*(2), 115-129. doi: 10.1016/j.accinf.2005.09.002

Molinari, L. K., Abratt, R., & Dion, P. (2008). Satisfaction, quality and value and effects on repurchase and positive word-of-mouth behavioral intentions in a b2b services context. *Journal of Services Marketing, 22*(5), 363-373. doi: 10.1108/08876040810889139

Nikolik, B. (2012). Software quality assurance economics. *Information & Software Technology, 54*(11), 1229-1238. doi: 10.1016/j.infsof.2012.06.003

Nommesen, P., & Macfie, A. (2012). Using pragmatic testing to ensure project success. *Business Intelligence Journal*, *17*(3), 40-47. Retrieved from http://search.ebscohost.com.libproxy.uoregon.edu/login.aspx?direct=true&db=cph&AN=80178600&login.asp&site=ehost-live&scope=site

Oberheide, J. C. (2012). Leveraging the cloud for software security services. Retrieved from http://deepblue.lib.umich.edu/handle/2027.42/91385

Pantouvakis, A. (2011). Internal service quality and job satisfaction synergies for performance improvement: Some evidence from a b2b environment. *Journal Of Targeting, Measurement & Analysis For Marketing, 19*(1), 11-22. doi:10.1057/jt.2011.2

Pardo, C., Pino, F. J., García, F., & Piattini, M. (2011). Harmonizing quality assurance processes and product characteristics. *Computer*, *44*(6), 94–96.

Patcha, K. K. (2009). Agile EDI Framework for b2b Applications. In *International Conference on Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09* (pp. 1 –3). Presented at the International Conference on Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09. doi:10.1109/ARTCom.2009.13

Quality Assurance Institute. (2006). Guide to the CSTE common body of knowledge. n.p.

Roberts, N. (2012). Leveraging information technology infrastructure to facilitate a firm's customer agility and competitive activity: an empirical investigation. *Journal Of Management Information Systems*, *28*(4), 231-270.

Rochon, P. A., Bero, L. A., Bay, A. M., Gold, J. L., Dergal, J. M., Binns, M. A., . . . Gurwitz, J. H. (2002, June 5). Comparison of review articles published in peer-reviewed and throwaway journals. doi:10.1001/jama.287.21.2853

Roseberry, W. (2012, November). Code coverage isn't quality, it isn't even coverage. Pacific Northwest Software Quality Conference Portland, OR. Paper retrieved from http://www.pnsqc.org/2012-conference/papers-and-presentations

Scott Ambler. (n.d.). In *Wikipedia.* Retrieved January 4, 2013, from http://en.wikipedia.org/w/index.php?title=Scott_Ambler&oldid=531316802

Spillner, A., Linz, T., & Schaefer, H. (2011). *Software testing foundations: A study guide for the*

certified tester exam. O'Reilly Media, Inc.

Stolberg, S. (2009). Enabling agile testing through continuous integration. In *Agile Conference, 2009. AGILE '09.* (pp. 369 –374). Presented at the Agile Conference, 2009. AGILE '09. doi:10.1109/AGILE.2009.16

Universidad Azteca International Network System. Faculty. Retrieved January 20, 2013 from http://www.univ-azteca.edu.mx/europe/Academics/faculty.html

Watkins, J., & Mills, S. (2011). *Testing IT: An off-the-shelf software testing process*. New York: Cambridge University Press.

Wiegers, K. E. (2003). *Software requirements: Practical techniques for gathering and managing requirements throughout the product development cycle.(2nd ed)* Redmond, WA: Microsoft Press.

William, A. (2012, November). Re: If you want better software quality, get rid of the QA department. Does it really work or software quality become non-existent? [Online forum comment]. Retrieved from http://www.linkedin.com/groupItemview=&gid=55636&item=184142883&type=member&commentID=106027793&trk=hb_ntf_LIKED_GROUP_DISCUSSION_COMMENT_YOU_CREATED#commentID_106027793

XP inventor talks about agile programming. (2007, November 8). *InfoWorld*. Retrieved from http://www.infoworld.com/d/developer-world/xp-inventor-talks-about-agile-programming-565

Zhang, Y., Fang, Y., Wei, K., Ramsey, E., McCole, P., et al. (2011). Repurchase intention in b2c e-

commerce--a relationship quality perspective. *Information & Management, 48*(6), 192-200. doi: 10.1016/j.im.2011.05.003