Journal of Information Technology Management

ISSN #1042-1319

A Publication of the Association of Management

MONITORING FOR TESTING THROUGHOUT THE DEVELOPMENT LIFECYCLE

ROBIN POSTON
UNIVERSITY OF MEMPHIS
rposton@memphis.edu

JIGNYA PATEL
UNIVERSITY OF MEMPHIS
jmpatel@memphis.edu

JASBIR DHALIWAL
UNIVERSITY OF MEMPHIS
jdhaliwl@memphis.edu

ABSTRACT

Given software release deadlines, the early stages of a structured software development life cycle (SDLC) project can run behind schedule, shrinking the time allowed for performing adequate testing. This situation urges the need to start testing early and manage the testing effort efficiently. Our research examines how to assess the ways activities in the earlier stages of a project are progressing relative to their effect on the efficiency and effectiveness of the latter SDLC stage of testing. We build on the design for testability perspective by introducing the manage for testability perspective, where software testability reflects whether the activities of the SDLC process are progressing in ways that support the testing team with the appropriate software project information and testable designs to enable finding software product problems if they exist during the testing stage. To address this challenge, we develop a software testing assessment to manage project testability during the earlier stages of the SDLC and we propose using the assessment as part of a testability measurement system. The software testing assessment is designed to provide testing managers information they need: (1) to influence pre-testing activities in ways that ultimately increase testing efficiency and effectiveness, and (2) to plan testing resources that facilitate an efficient and effective testing stage. We developed specific software testing assessment measures through several rounds of interviews with key informants (i.e., testing managers at a global transportation company). To support the assessment's usefulness and application, we present data collected for the measures for large-scale structured software development projects and post-data collection debriefing sessions with senior testing leaders.

Keywords: testability, software project, design for testability, manage for testability, qualitative

INTRODUCTION

Waterfall development remains the most commonly used methods in large organizations [12], where inadequate quality can result from insufficient testing activities which are often relegated and compressed into the last stages of the life cycle limiting the time available for finding and fixing problems [11]. With pre-set release deadlines, the early stages of planning, analysis, design, and development within a structured software development life cycle (SDLC) can run behind schedule, shrinking the time allowed for performing adequate testing [9], [19]. One solution would be to better plan the testing process to be more efficient, while another would be to improve how activities in the earlier stages of the SDLC affect downstream testing activities, e.g., by ensuring less-ambiguous requirements are documented during the analysis stage. This article examines how to assess how activities in the earlier stages of a project are progressing relative to their effect on the latter SDLC stage of testing to inform testing management how to take remedial actions.

Many activities within the early stages of the SDLC influence the amount and type of software testing performed at the end of the SDLC [1], [7]. While progressive software development teams include members of the testing team in document walkthroughs and inspections early in the SDLC [17], most testing teams lack a software testing assessment to evaluate how the activities of the early stages are progressing relative to their influence on tasks performed during testing. Armed with such an assessment, testing managers could use assessment data to attempt to facilitate positive changes in the initial stages of the SDLC, as well as gain early warning of the amount and type of testing resources needed prior to the beginning of the testing stage.

Software testing assessment frameworks (Testing Maturity Model, Test Process Improvement, The Testability Fishbone [5]) currently exist that inform software development teams on ways to design software code to be more testable and the means of estimating the testing effort. These frameworks offer a design for testability (DFT) perspective, where software testability reflects whether code has been designed in such a way that the testing team will be able to find software problems if they exist [3], [6], [16]. From a DFT perspective, testability is a cumulative measure of design attributes of a developing software product that reflect the team's ability to assess the amount of testing effort that will be needed to proficiently test the product. The less testable a software product, the more testing effort will be needed to ensure its quality prior to its release. Proposed DFT measures have focused on improving test cases [2], class diagram interactions [4], input and output states of the code, and state transitions of the program [10]. These measures support utilizing design and code methodologies to help ensure more testable software products enter the testing stage. The DFT research addresses ways to improve a product's testability at the software design phase, with little attention given to ways to improve a product's testability through better managing the activities performed throughout all the phases of the SDLC process.

This article compliments the DFT perspective by examining how to assess SDLC process activities relative to their effect on the testing stage. We call this approach the manage for testability (MFT) perspective. From a MFT perspective, testability reflects whether SDLC activities are progressing in ways that are informing and supporting the testing team with the project information needed to improve the efficiency and effectiveness of finding software product problems if they exist during the testing stage. Following MFT, we develop a software testing assessment, which focuses on the process and product characteristics of how the activities of the SDLC are progressing relative to their influence on tasks that will be performed during the testing stage, with the goal of improving testing capabilities. Projects with low software testability assessments indicate greater testing effort will be needed as the activities of earlier stages of the SDLC are progressing in ways that make finding defects more challenging.

In support of the MFT perspective, prior research has acknowledged the need for assessing testability at the SDLC process level [5], however details of assessment criteria have not been offered. While the recognition for the need for MFT persists, little guidance exists as to how to develop a software testing assessment to help testing managers evaluate how activities performed earlier in the SDLC influence a software project's testability and the testing effort that will ultimately be required during the testing stage. This article develops a project testability assessment for testing managers to utilize during the earlier stages of the SDLC before software testing begins and proposes using the assessment as part of a testability measurement system. The software testing assessment is designed to provide managers information they need: (1) to influence pre-testing activities in ways that ultimately increase testing outcomes, and (2) to plan testing resources that facilitate an efficient and effective testing phase. Thus, in this article, we move beyond the software product DFT research [4], [10], [15] to explore how to manage activities earlier in the SDLC process to improve testing activities. Based on the insights learned here, we

propose best practices for testing managers to follow in order to increase the testability of their SDLC projects.

To create the assessment, we developed specific software testing assessment measures for structured projects through several rounds of interviews with key informants (i.e., testing managers at a global transportation company). Our aim was to discover and define measures of MFT. We next validated the testability measures with member-checking and peer-debriefing and updated our assessment accordingly. Finally we collected data for the measures for structured software development projects at the original global transportation company, as well as at other large companies, and then gathered post data-collection debriefing feedback from senior testing leaders.

CREATING SOFTWARE TESTABILITY MEASURES

Producing high-quality software is not only a function of creating high-quality software product designs, but also managing high-quality software development processes. Binder [5] uses fishbone diagrams to illustrate the myriad facets of the testing process which influence testability, and emphasizes that "testability cannot be considered apart from the [SDLC] process" (p. 88). However, the paper fails to define measures of the activities of the SDLC process that influence testability. Voas and Miller [18] focus on random black-box testing DFT and suggest repeated measures of testability are needed throughout the SDLC, however, they also do not define measures. Our research was designed to develop these measures.

Given these gaps, to create the software testability measures, we followed three main steps: interviewing key informants to draft assessment measures; gathering feedback from additional key informants to determine the clarity, comprehensiveness, and accuracy of the measures; and collecting data from testing managers across multiple Fortune 500 level companies as a proof of concept. In the interview step, four testing leads and two testing auditing managers at a global transportation company were asked questions about the attributes of work performed across the SDLC that affected the ability of the testing team to find problems in the software and that influenced the work activities performed in the testing stage. While the key informant sample represents a convenience sampling, people were selected based on the recommendation of senior testing leaders and were chosen on the basis of their considerable experience and expertise.

To validate the assessment, we used triangulation accomplished through the use of multiple data sources and multiple researchers [14]. Iterative comparison and contrasting and cross-examination of our work across multiple key informant interviewees allowed us to ensure that the outcomes of this assessment are well developed. The presence of multiple researchers allowed us to systematically recognize, discuss, and debate different interpretations and improve our understanding of defining the testability measures. To further improve the merit of the assessment we also employed member-checking and peer-debriefing [8]. We presented drafts of the assessment to the upper management team (senior managing directors and vice president) at the global transportation company (member-checking) as well as with researchers and practitioners at a research workshop, a separate research colloquium, and international conference to gather additional input (peer-debriefing). In addition, key informants from a global aviation company reviewed and provided input on each measure. Together, these steps serve to support the assessment's credibility, validity, and generalizability.

In the data collection step, testing managers not involved in the steps above selected and assessed one of their development projects. A total of 15 projects that were either in testing phase or release phase were assessed across five Fortune-500 level companies: five projects from the global transportation company, three from a global business-to-business supply chain company, three from a global retail company, two from a major utility company, one from a large non-profit healthcare company, and one from a worldwide manufacturer of engineering solutions. The experience of the testing managers handling the project ranged from 4 – 25 years. The testing manger's experience ranged from 0-25 years and the number of man hours needed to complete the project ranged from 30-100,000. The purpose of the data collection was to offer a proof of concept illustrating how the testability measures would be evaluated. The following sections describe the software testability assessment and its application in more detail.

SOFTWARE TESTING ASSESSMENT

The software testing assessment is provided in the Appendix. As illustrated in the assessment, measures were developed for the following information technology components: software, hardware, documentation, security, data, and facilities. These components were derived from the COBIT framework used by the IT professionals in the implementation, review, administration, and monitoring of an IT environment including the software development process [13]. Within these components each area was fur-

ther broken down into its subcomponents. For example, under the heading of software, the subcomponents include: critical applications, where and how applications are executed, if patches are up-to-date, input and output controls, and error messages. For each subcomponent, specific testability measures were developed. For example, for error messages, measures include: error messages provide clear description of the problem, error handling processes are efficient, and ability to perform fail-over and recovery testing. When using the software assessment for software projects, testing managers rate each testability measure for the project on a scale where 7 = highest in testability to 1 = lowest in testability. For example, if error messages provide clear descriptions of the problem, the associated measure would be rated 7. In this case, being rated at the highest testability level means this activity within the SDLC process provides insightful information about software defects in error message coding improving the efficiency and effectiveness of the testing team to test for and find problems if they exist.

As a proof of concept, to illustrate how the testability measures would be evaluated and their usefulness, we provide data collected for large-scale software development projects. We asked each respondent to assess a project in the testing or release stage of the SDLC in order to encourage participants to consider how each measure influenced the ability to find problems if they existed. Limiting our data collection to projects in the final stages of development helped us validate the newly-created testability measures. Some specific testability measures were irrelevant to a specific project. We remove the effects of the irrelevant measures to compare projects by calculating the percentage of the total possible score for each project. Table 1 summarizes these percentages for all projects assessed which range from 37% to 82%, and average 64%. The testability scores can be used by testing managers to highlight issues in projects. For example, a project scoring 37% testability score would imply that there are more SDLC issues to monitor and manage compared to a project scoring 82% as its testability score.

Table 1: Summary of Project Software Testing Assessment

Project Name	**Total Score**	**Total Possible**	**Testability Score (Total Score/ Total Possible)**
Ink and Toner Saver	183	224	82%
Lab data management	228	287	79%
ePrint	192	245	78%
Management GUI	250.5	336	75%
JRB Conversion	238	323	74%
New service introduction	232	315	74%
I Roads	266	371	72%
International Returns	219	315	70%
Pricing enhancements	200	315	63%
Vendor Conversion	169	294	57%
DSO Process Improvement	206	364	57%
Global Tax Engine	197	357	55%
ILS	155	315	49%
Event Report	161	371	43%
Plant Metric Dashboard	131	350	37%

TESTABILITY MEASUREMENT SYSTEM

Next we held post data-collection debriefing sessions with additional software testers with project oversight responsibilities: two managing directors and a vice president at the global transportation company and a senior project manager and two senior testing managers at the major utility company. The goal was to determine the uses and benefits of implementing the assessment to manage for testability. Table 2 summarizes the proposed uses and benefits. The senior testing leaders interviewed agreed that the testability scores as evaluated by their testing managers reflected their knowledge of the testing challenges encountered with each project. The senior testing leaders confirmed the order of projects from highest to lowest testability scores did reflect the relative amount of challenges and testing effort that incurred within each of the projects evaluated for their respective companies. This feedback supports the usefulness of the testability assessment.

Table 2: Proposed Uses and Benefits of Using the Software Testing Assessment

Proposed Uses	**Proposed Benefits**
1. Library of project assessments	Software testing assessment scores can be used for project baselines and comparisons to highlight issues
2. Within project checkpoints	Comparison of scores across the SDLC can highlight progress
3. Communication mechanism	Scores assessed by a variety of project stakeholders can be used to reconcile differing opinions, compare scores to reality, educate the project manager, and identify project issues
4. Risk and resource indicator	Scores can be used to discuss with upper management resource allocations and realistic goal setting
5. Troubled project indicator	Scores can aid in determining which projects should be abandoned

We propose the following testability measurement system, illustrated in Figure 1. The measurement system offers managers a way to assess how activities in the earlier stages of a project are progressing relative to their effect on the latter SDLC stage of testing. Our assessment illustrated in the Appendix provides measures of work performed throughout the SDLC including project documentation, tester's product understanding, software product attributes, etc. As an example of assessing and managing testability during the initial planning stage of an SDLC project, one of the testability measures evaluates the quality or number of problems with prior version of the product in cases involving major modifications. If the quality of the original software is low, more problems existed in prior versions increasing the likelihood that the problems persist. In this situation, testing teams will have greater difficulty finding defects that exist and will want to encourage SDLC stakeholders to be more diligent before testing begins or to be prepared for great testing expense given the additional effort that will be needed to test the product. The dark black arrow illustrates that the information gathered by a testing manager from the assessment throughout the SDLC can then be used to plan the amount of testing resources that will be needed to test to the product.

In the analysis state, one of the testability measures offers insights to the level of involvement that testing representatives have in document walkthroughs. Less involvement suggests the testing team has less input as well as less understanding of the project leading to greater challenges in finding problems and more testing needed during the testing stage. In the design stage, one of the testability measures addresses how well data mapping between systems has been documented. Poor documentation makes it more difficult to find software problems. In the development stage, one of the testability measures assesses the ability to decouple code between interfacing systems. Reduced ability to decouple code leads to more difficulty finding problems and more testing work required to define issues and determining solutions. In these situations, using a software testing assessment throughout the SDLC would provide useful information to testing managers needed to influence their SDLC counterparts to improve the software product or process activities throughout the SDLC, ultimately improving the efficiency and effectiveness of the testing stage. The assessment would also offer testing managers early warning about the testing challenges and effort needed in planning testing resources prior to the beginning of the testing stage.

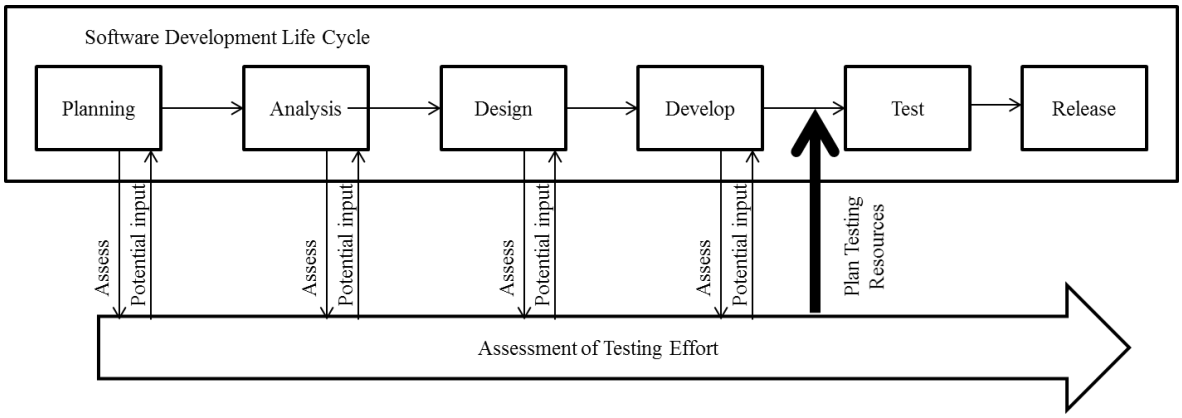

Figure 1: Testability Measurement System

In a survey of software development professionals from other large organizations following structured development processes, we asked 24 project team leaders questions about how the assessment would benefit their projects, its stakeholders, and the organization. As shown in Table 3, the assessment would be most useful for stakeholders to identify project issues (95.8%), for managers to assess project risks (91.7%), and for the quality assurance team to improve testing process (91.7%).

Table 3: Testability Assessment's Usefulness by Project, Stakeholder, or Organization-Related Uses

Survey Items	**% of respondents agreeing with assessment's usefulness (n = 24)**
Project-related	
Identifying project issues	95.80%
Assessing risks in the project	91.70%
Improving project testing	91.70%
Comparing projects	87.50%
Managing project	87.50%
Comparing a project across lifecycle stages	83.30%
Determining if project should be abandoned	54.20%
Stakeholder-related	
Highlighting stakeholders with accurate perceptions	83.30%
Reconciling stakeholder opinions about the testability of the project	75.00%
Organization-related	
Raising awareness to upper management of risks	87.50%
Becoming part of the organization culture	83.30%
Discussing with upper management about resources needed	79.20%
Note: Each survey item was evaluated using a 7 point Likert scale with 1 signifying "Strongly Disagree" and 7 signifying "Strongly Agree".	

DISCUSSION AND CONCLUSION

This study collects data from actual projects. Given our findings and discussions with the respondents, we recognize project or testing managers may not necessarily be the one and only respondent needed to assess the testability of projects. Project and testing managers bring a project and testing perspective, respectively, which may bias their viewpoints in evaluating projects across the testability measures. Thus, multiple project stakeholders, such as developers and designers, should also assess project testability. How and why assessments provided by various project stakeholders converge or diverge in their opinions of project testability could provide insightful evidence regarding the challenges of MFT. A project manager could assess the level of bias by comparing assessments to actual outcomes and by obtaining data from more members of the SDLC team. This would also highlight which stakeholders provide more accurate insights about project testability.

Another important implication of this study is the need to use the software testing assessment as a benchmark-type tool to determine what types of projects are more versus less testable. A database of projects could be gathered and used to determine patterns of project factors that drive testability. Factors could include project size, project manager style, whether offshoring was involved or not, criticality of the software to the user base, etc. As benchmark data builds, best practices in software testability can be derived and baseline measures could assess if future projects exhibit signs of improvement. With a database, comparing measures across and within SDLC stages may also provide useful insights. Through statistical analysis of the data, researchers could determine which measures are driving testability the most and which criteria are most critical to the testability of software projects. Utilizing such a repository of projects, demographics and testability scores could be analyzed to assess what factors drive testability across types of projects, people, and organizations.

As a contribution to practice, testing managers lack the means to systematically assess how the activities of the SDLC are progressing in their relationship to a software product's testability, which ultimately impacts the ability to find software problems if they exist and the amount of testing effort required in the testing stage. In order to provide managers with the ability to influence pre-testing activities in ways that reduce testing efforts, and to plan testing resources that facilitate an efficient and effective testing phase, we propose managers utilize the software testing assessment, as developed and illustrated through our research, throughout the SDLC. Testing managers and other SDLC stakeholders should utilize the software testing assessment as an audit tool, benchmark baseline, or a checklist. By assessing software development projects starting at the beginning of the SDLC, development teams can learn to build testability into their projects along the way and testing managers can gain forewarning of testing issues prior to the beginning of the testing phase.

This also gives the top testing management team a communication mechanism to open discussions with SDLC stakeholders about areas of improvement. As the findings of Table 2 illustrate, issues that affect project testability are pervasive, as all projects scored below 85%, with 7 scoring at or below 70% signifying projects that have 'significant challenges' in testability. Armed with the assessment data, the score can encourage discussions among SDLC stakeholders to find ways to improve the process, and ultimate improve testing performance. An efficient approach would be for practitioners to monitor which measures are most strongly associated with testability issues and identify the 'top 10'. Then the most important testability indicators could be adopted as part of standard development practices.

In this study, the software testing assessment that was developed covers a variety of activities and facets of the SDLC. The assessment in the Appendix illustrates the documentation category has the highest number of measures. . This finding suggests testing leads should monitor how documentation is being managed as the project progresses through the SDLC. Documentation comprises business requirements, system requirements, system architecture specifications, de-coupling and back-out plans, detailed test plan specification, and defect log files. This finding highlights the need for testing managers to pay special attention to the level of testability based on the level of involvement and understanding of the testing team in document walkthroughs; how well documentation is begin developed and maintained; and information traceability, version control, storage location, and open issue resolution.

REFERENCES

[1] Adrion, W. R., Branstad, M. A., and Cherniavsky, J. C., "Validation, Verification, and Testing of Computer Software," *ACM Computing Surveys (CSUR)*, Volume 4, Number 2, 1982, pp.159-192.

[2] Bache, R., and Mullerberg, M., "Measures of testability as a basis for quality assurance," *Software Engineering Journal*, Volume 5, Number 2, 1990, pp. 86-92.

[3] Basili, V. R., Briand, L. C., and Melo W. L., "A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE Transactions on Software Engineering*, Volume 22, Number 10, 1996, pp. 751-761.

[4] Baudry, B., Traon, Y., and Sunye, G., "Testability Analysis of a UML Class Diagram," *Proceedings of the IEEE Symposium on Software Metrics*, 2002, pp. 54-63.

[5] Binder, R., "Design for Testability in Object Oriented Systems," *Communications of the ACM*, Volume 37, Number 9, 1994. pp. 87-101.

[6] Chidamber, S.R., and Kemerer, C.F., "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, Volume 20, Number 6, 1994. pp. 476-493.

[7] Cohen, C., Birkin, S., Garfield, M., and Webb, H., "Managing Conflict in Software Testing," *Communications of the ACM*, Volume 47, Number 1, 2004, pp. 76-81.

[8] Corbin, J., and Strauss, A., "Basics of Qualitative Research", Sage Publications, Newbury Park, California, 2008, pp. 299-300.

[9] Everett, G., and McLeod, R., "Software Testing: Testing Across the entire Software Development Lifecycle," John Wiley & Sons Inc., Hoboken, New Jersey, 2007.

[10] Freedman, R., "Testability of Software Components," IEEE *Transactions on Software Engineering*, Volume 17, Number 6, 1991, pp. 553-564.

[11] Gelperin, D., and Hetzel, B., "The Growth of Software Testing," *Communications of the ACM*, Volume 31, Number 6, 1988, pp. 687-695.

[12] Gillenson, M.L., Racer, M.J., Richardson, S.M., and Zhang, X., "Engaging Testers Early and Throughout the Software Development Process: Six Models and a Simulation Study," *Journal of Information Technology Management*, Volume 22, Number 1, 2011, pp. 2011-2027.

[13] Lindeberg, T., "The Ambiguous Identity of Auditing," *Financial Accountability and Management*, Volume 23, Number 3, 2003, pp. 4267-4424.

[14] Mason, J., "Qualitative Researching," Sage Publications, London, London, 2002, pp 190-191.

[15] Mouchawrab, S., Briand, L., and Labiche, Y., "A Measurement Framework for Object-Oriented Software Testability," Information *and Software Technology*, Volume 47, Number 15, 2005, pp. 979-997.

[16] Offutt, J., "Quality Attributes of Web Software Applications," *IEEE Software*, Volume 19, Number 2, 2002, pp.25-32.

[17] Singh, Y., and Shivani, G., "Role of Testing in Phases of SDLC and Quality," *International Journal of Information Technology*, Volume 2, Number 2, 2009, pp. 343-346.

[18] Voas, J., and Miller, K., "Improving the Software Development Process using Testability Research," *Proceedings of Software Reliability Engineering*, Research Triangle Park, North Carolina, October 7-10, 1992, pp 114-121.

[19] Whittaker, J., "What is Software Testing? And Why is it so Hard? *IEEE Software*, Volume 17, Number 1, 2000, pp. 70-79.

AUTHOR BIOGRAPHY

Jasbir Dhaliwal is Professor of Information Systems and Associate Dean for Research and Academic Programs at the University of Memphis. As the Director, he also leads the Systems Testing Excellence Program at the FedEx Institute of Technology that seeks to advance the science of software testing to provide a stronger theoretical base for industry practice. He has a PhD. from the University of British Columbia

Jignya Patel is currently working toward her PhD degree in the Department of Management Information Systems at the University of Memphis. Her research interests focus on both technical topics such as software testing and non-technical topics with managerial implications such as knowledge management and effective ways to integrate technology into the workplace.

Robin Poston received a B.S. in Computer Science from the University of Pennsylvania and a Ph.D. in Information Systems from Michigan State University. She is an Associate Professor at the University of Memphis and Associate Director of the Systems Testing Excellence Program. She has published articles in journals such as *MIS Quarterly, Journal of Management Information Systems, Decision Sciences Journal, IEEE Transactions on Engineering Management, Communications of the ACM, IEEE Computer,* and others, including international conference proceedings.

APPENDIX: SOFTWARE TESTING ASSESSMENT

Note: Each measure is assessed on a Testability Scale of 7 = highest to 1 = lowest

Testability Categories	**Testability Measures**
Critical applications	Quality of original software before testing starts specifically, unit test results along with build and known issues are available
	Quality of original software before testing starts specifically, first cycle of integration (end-to-end) testing results are good
Where and how applications are executed	Visibility to data mapping to input and output of interfacing systems
	Ability to control business rule parameters (e.g., modify data retention periods)
Are patches are up-to-date?	All patches been applied within the test environment before the start of testing
Input and output controls	Data dependencies are documented
	Changes that affect other systems are documented
Error messages	Error messages provide clear description of the problem
	Error handling processes are efficient
	Ability to perform fail-over and recovery testing
System fileservers: fileserver integrity	All fileservers are operational
Business Requirements (BRS)	Level of involvement of testing representative(s) in the document walkthrough
	Understanding of BRS by testing team members
	Comprehensive assumptions and constraints have been included
	Detail business scenarios and examples have been included
	High level specifications for de-coupling have been included
	Stakeholder review and approvals exist
	Version control in place and followed
	Open issues are tracked and addressed
System Requirements (SRS)	Level of involvement of testing representative(s) in the document walkthrough
	Understanding of SRS by testing team members
	Comprehensive assumptions and constraints have been included
	Detail scenarios and examples have been included
	Traceability to BRS has been documented
	Stakeholder review and approvals exist
	Version control in place and followed
	Open issues are tracked and addressed
System Architecture Specification (SAS)	Completed and provided with entire system flow
	Visibility of all interface changes
	Defined data mapping between systems
De-coupling/ Back-out Plan	Document is complete and provided
	Ability to decouple specific functions within a

	project
	Degree of ability to decouple the code between interfacing systems /domains (more data/switch driven less code driven)
Detail Test Plan Specification (DTPS)	Stakeholder review and approvals exist
	Understanding of DTPS by testing team members
	Version control in place and followed
	Known location of organized repository of project files
	Mitigation and contingency plan known risks
	Well defined test strategy
	Well defined test cases
	Well defined test data plan
Log files: Defect log files	All defects and their remedies are logged in an easily accessible manner by the testing group
Overall	Defined process for tracking and resolving testing issues/concerns/queries
Access controls	Access rights to all impacted systems have been set up before the start of testing
	Access rights have been completely defined before the start of testing
Internal controls on key applications	Ability to test software compliance (e.g., HIPPA, SOX, PCI)
Data security policies: Is there any formal written data security policy?	Test data is locked down and secure
	Production data is efficiently cleansed of sensitive information
Data files / database access	Updates and database files are accurate and available
Data encryption	Ability to simulate sensitive data
	Ability to simulate encrypted data
	Level of complexity in decrypting encrypted data
Test environment	Separate testing environment from the remaining software development team