An Information Systems Design Product Theory for Integrated Requirements, Test and Defect Management Systems

Yikun Lu & Timo Käkölä
University of Jyväskylä, Finland
{yilu, timokk}@jyu.fi

Abstract

Software companies need to develop high-quality products fast and with low cost. Development activities are typically distributed in different places and involve multiple stakeholders in different countries. Testing is an important way to ensure product quality but it is costly and takes ample time and resources. Sourcing testing services from specialized service providers is often the most viable option for software businesses. Flexible and effective test process management is especially vital to support the testing life-cycle when external testing service providers are involved. Yet, the extant literature provides little theoretical guidance for managing the testing life-cycle, including requirements, test, and defect management. This paper develops an Information Systems Design Product Theory for the class of integrated Requirements, Test and Defect Management Systems based on a case study and a literature review. This theory helps clients and testing service providers to manage and control the testing process, make the process transparent and seamless, and improve service effectiveness.

1. Introduction

Software has become ubiquitous in society and it is a key asset for high technology products. Therefore, critical concerns for software business are high software quality and competitiveness. Increasingly, complex high quality systems are produced with constant, or even diminishing, human resources [19]. According to Owens and Khazanchi [14], software quality assurance plays a critical role in the software development life-cycle and can impact a project’s overall success. If quality management is not effective, the projects will fail no matter how advanced the tools and techniques are [7].

Software testing is vital to reach software quality objectives [6; 8]. It is an empirical investigation conducted to provide stakeholders with information about the quality of the products and/or services under test with respect to the contexts in which the products and services are intended to operate [2; 9; 11]. Software testing can be divided in static and dynamic testing. Static testing is the process of reviewing documents and detecting errors without executing the software. Dynamic testing is the process of detecting faults in the software by executing the software with appropriate test materials such as automated scripts or test-specific software components [3].

Software testing is one of the most time- and labor-intensive development activities, consuming between 30% and 50% of the total development cost [1; 10; 17]. Software businesses are often faced with time and resource constrains, limiting their abilities to complete testing efforts effectively [11]. It is thus challenging to execute testing with low costs and high quality.

Sourcing testing services from specialized service providers is often the most viable option for software businesses. Testing service providers can make the necessary investments to effectively deal with the complexity involved in software testing. However, with more and more service providers entering the markets, competition within this industry has become increasingly fierce and time-based. To meet the competitive pressures, testing service providers need to improve management for the whole testing service life-cycle and provide mass-customizable services that address the needs of their clients rapidly and profitably.

To succeed in the testing market, testing service providers need to co-operate with geographically distributed clients to manage the testing life-cycle effectively. Sophisticated information systems are needed to support this collaboration throughout the life-cycle. Yet, most commercially available systems focus only on parts of the testing life-cycle and provide limited support for clients. They are used separately for requirements management, test execution management, or defect management. Standardized data transfer between the different information systems supporting specific life-cycle phases is difficult, reducing service effectiveness and raising the risks of failure. HP Quality Center is, to our knowledge, the only
commercially available integrated requirements, test, and defect management product.

This paper proposes that integrated Requirements, Test and Defect Management Systems (hereafter, RTDMS) should be designed to help testing service providers manage the testing life-cycle from test requirements management through test execution and defect tracking and management to project closure. A project database supports each RTDMS instance. It is needed to store automated test scripts and other artifacts for all testing projects. It includes reusable artifacts and other information produced during all the projects that the service provider has served. RTDMS instances use the associated repositories to help projects manage newly created test artifacts and reuse test artifacts from previous projects. RTDMS instances also support test team members’ work and help the clients monitor and track the testing service process. Clients and service providers can obtain timely information from RTDMS instances, making the service life-cycle transparent and seamless.

To achieve such integration, there are several factors to consider. Client requirements and system requirements need to be transformed to detailed and executable test requirements. Specific test executions may uncover defects and other problems needing quick communication between clients and test teams and between test team members, but test teams may be globally distributed, making it difficult to conduct face to face meetings with clients. Changing client requirements or revised test plans may require revising contracts between clients and testing service providers and raise project risks.

There is little theory-based guidance to help service providers and clients design and leverage such integrated systems. This paper draws upon a case study to create an information system design product theory (hereafter, design product theory) for the class of RTDMS. A complete information systems design theory (ISDT) prescribes both the product and process aspects of a class of information systems, that is, what are the meta-requirements and the meta-design for all the products within the class and how the products should be built [20]. This paper focuses on prescribing the product aspects for the class of RTDMS because the existing literature does not provide such a theory. Moreover, RTDMS instances can be built in many ways and it is thus not as fruitful to prescribe the process aspects as the product aspects. The design product theory addresses the following research question: What are the meta-requirements and the meta-design of the design product theory for the class of RTDMS in order to enable comprehensive testing life-cycle management?

The paper is organized as follows. Section 2 presents the research methodology and the case company. Section 3 presents the meta-requirements for RTDMS, that is, the specific practices and test team roles and tasks in each phase of the testing life-cycle that must be supported by the RTDMS instances. Section 4 describes the meta-design for the class of RTDMS. The last section concludes the paper and suggests topics for future research.

2. Research methodology and the case organization

This research was conducted in the context of the Chinese testing services market, offering services for international and domestic clients that leverage information and communication technology (ICT) enabled sourcing (eSourcing). Testing sourcing services are widely delivered by Chinese providers. To understand the international eSourcing life-cycle holistically from both clients’ and providers’ viewpoints, the eSourcing Capability Model for Service Providers (eSCM-SP) was chosen as the reference model. It has been demonstrated to help various types of providers to improve their capabilities related to both ongoing, phase-specific, and engagement-specific sourcing practices throughout the sourcing life-cycle [18]. The eSCM-SP life-cycle (Table 1) involves three phases from the provider’s viewpoint: initiation, delivery, and completion. Ongoing practices are run throughout the life-cycle to perform management functions. The three phases and the ongoing practices cover ten capability areas (e.g., knowledge management, threat management, and performance management). This investigation collected data and compared the practices of the case organization to eSCM-SP based on the three phases and specific practices. This paper presents the testing life-cycle from the viewpoint of the three phases.

This research uses a single qualitative case study to provide a holistic, systemic understanding of the phenomenon of testing services provisioning in the context of eSourcing [5]. It collected data covering the complete sourcing life-cycle for testing service providers, including the most important testing practices, the artifacts reused in these practices, and the people responsible for test assignments.

The first author conducted two rounds of investigation in the case company. For the first round, he spent over three weeks observing life in the case organization, analyzing documents and memoranda, and interviewing key personnel. The in-depth interviews involved the CEO, all testing managers, and a number of test analysts to uncover the routine
practices and information systems associated with testing work and major breakdowns disrupting work. Each interview was started by following a questionnaire and was concluded with an open discussion to address emerging issues. Interviews were summarized and sent to the interviewees, who verified them and provided feedback as necessary. If the analysis indicated that major deviations existed or information was missing, clarifications were re-requested from informants through email. Data collection and analysis continued for several months using the internet to collaborate with the case organization. A year after the first round of interviews, the first author performed a second round in the case organization to collect supplementary data related to breakdowns and workarounds. This time, the quality assurance manager, the measurement process manager, and other people supporting the test teams were also interviewed.

<table>
<thead>
<tr>
<th>Title</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Manager</td>
<td>Test managers are responsible for test project planning, management, risk evaluation, and report review. During project completion, test managers summarize the projects and the lessons learnt. They have at least five years of relevant work experience.</td>
</tr>
<tr>
<td>Test Analyst</td>
<td>Test analysts analyze test requirements; design test plans (together with the test manager); and design test cases. They have three to four years of relevant work experience.</td>
</tr>
<tr>
<td>Tester</td>
<td>Testers conduct the specific test assignments. They have more than one year of work experience.</td>
</tr>
<tr>
<td>Coordinator</td>
<td>Coordinators communicate with clients, acting as bridges between clients and service providers. They need to have comprehensive testing knowledge, because they attend the testing service life-cycle from early bidding and negotiation through to service completion.</td>
</tr>
</tbody>
</table>

Table 1. Job Descriptions of Key Roles in Test Teams

Test teams are responsible for testing. Usually the teams have four roles: test manager, test analyst, tester, and coordinator (Table 1). Coordinators serve as boundary spanners between clients and providers. Coordinators are especially important for solving communication challenges in international sourcing when clients and vendors use different languages and have different cultures [13; 15]. Test teams can be organized flexibly based on the project characteristics, personnel workloads, and client requirements. For example, when the projects are small, testers need not be involved in test teams because test managers and test analysts can do their work.

The case organization deployed HP Quality Center for managing requirements, test, and defect information throughout the testing services lifecycle.

“All the test projects will use QC or other similar test platforms, but the other platforms mainly focus on test management. QC is better than them, because it can also offer complete requirements management and defect management services. The specific test tools can run on this platform, but they can also run on other platform, so based on QC we can better control the whole service life-cycle.” -CEO

The design product theory for RTDMS was created based on a literature review and the analysis of the case organization and its use of the Quality Center. The case organization is a leader in its business in China, so its practices are likely to create an adequate baseline for theory creation.

The design product theory for RTDMS has been designed to be abstract and generic enough so testing services providers can use it to improve their processes and information systems regardless of their current practices and systems. It may be possible for the providers to benefit from the theory for RTDMS even without replacing any existing systems. Service providers can thus use even separate requirements management systems, test management systems, and defect management systems and use the theory to better integrate and organize these systems for enabling the end-to-end life-cycle. For example, a RTDMS instance can track the test execution process against the test plan and report test results and defects. It does not need to help execute specific test assignments but it needs to trace and report the results of the assignments. A specific test assignment can be run by using other test tools. Therefore, the analysis of the practices and information systems of the case organization has helped us to scope the design product theory for RTDMS appropriately.

### 3. Meta-requirements of the design product theory for RTDMS

This section describes the meta-requirements for the design product theory of RTDMS, that is, what services integrated requirements, test, and defect management systems must provide to enable stakeholders to streamline the end-to-end testing life-cycle (Figure 1). RTDMS instances shall offer three categories of services: (1) requirements management, (2) test management, and (3) defect management (Table 2). Requirements management deals with, for example, test requirements prioritization and management. Prioritized requirements, in turn, establish a baseline to measure test progress and trace related test artifacts. For example, specific test assignments are based on test requirements. Execution results and found defects are linked with the specific
test assignments. The interdependencies between requirements are recorded and presented to manage test requirements and arrange test efforts and resources. Aligning test requirements and test cases helps ensure the test cases meet the test requirements. Test execution refers to creating and running manual or automated tests and reporting execution results and defects. During the execution process, version control and baseline management are used to ensure that each version of the system is tested using the right, baselined set of test artifacts. Whenever a client changes requirements, the RTDMS instance generates impact analysis reports, so the stakeholders can evaluate the impacts and decide whether to revise the test plans or not. Defect management refers to identifying defects and tracking them from initial detection to resolution and verification.

<table>
<thead>
<tr>
<th>Requirements management</th>
<th>Test management</th>
<th>Defect management</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Prioritize requirements based on business priorities and risks</td>
<td>1. Monitor the progress against the test plan</td>
<td>Identify defects and track them from initial detection to resolution and verification</td>
</tr>
<tr>
<td>2. Collect requirements from previous projects to reduce duplication</td>
<td>2. Report execution results and defects</td>
<td></td>
</tr>
<tr>
<td>3. Collect test cases from previous similar projects</td>
<td>3. Version control and baseline management</td>
<td></td>
</tr>
<tr>
<td>4. Manage interdependencies between requirements and align test requirements and test cases</td>
<td>4. Generate impact analysis report</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. A framework for categorizing the services of the design product theory for RTDMS

Test artifacts from previous projects should be reused throughout the testing life cycle to improve productivity and software quality [12], reduce development costs [16] and shorten the time to market [4]. RTDMS instances thus store the artifacts and make them available for reuse. However, there are factors that reduce the possibilities for artifact-level reuse. For example, clients typically use different development processes and tools. The systems or products to be tested are thus associated with diverse sets of models, documents, and other artifacts that do not follow any single standard or guideline. Therefore, it would be too risky and laborious for service providers to store all the artifacts in one format, for example, using the Unified Modeling Language (UML). The artifacts are typically stored in formats in which they were produced during the service delivery projects.

‘UML is good, but it is still too risky to store all the artifacts in this format. For example, if the clients’ development processes do not use UML but we use it, it would be too risky for the projects.’ -A test manager

“There are many differences in individual projects, so reusing the test methods and test knowledge is more important than reusing specific artifacts.” - A test manager

The following subsections show the meta-requirements in the context of the three phases of the testing services life-cycle: initiation, delivery, and completion of a testing sourcing engagement.

3.1. Initiation

In the testing initiation phase, the client provides a request for proposal (RFP) and documented system requirements, including specific testing service requirements. Test manager and test analyst need to analyze the proposal and requirements to create a business case to estimate the profitability of the project. If the project is profitable and meets the objectives of the provider, and resources are available, they will draft a brief test plan and bid for the project. Coordinator will assist the test manager and the test analyst to analyze the RFP and test requirements especially in an international sourcing situation when the client is from a different country and/or the RFP uses a different language. Sometimes, it is impossible to form an executable test plan because the test requirements are unclear or incomplete. If the applicable laws for tendering allow it, providers can request the client to elaborate on some requirements to ensure they understand what the client wants. Elaborations must be made available to all providers to afford fair bidding. If the provider wins the contract, the test team needs to draft a detailed test plan, which includes the test cases to be used. After the client reviews and approves the detailed test plan, the client will transfer the system to be tested and other test resources to the provider. Both sides will work together to set up a test environment and arrange training for stakeholders. Test manager will select appropriate testers, make them available to the project, and assign them specific test assignments.

Requirements analysis and management are the main activities throughout the testing life-cycle. The project database enables a RTDMS instance to manage multiple requirement types and provides real-time visibility of requirements coverage and associated defects to evaluate quality and business risks. Furthermore, multi-dimensional traceability is supported between requirements, tests, and defects across releases and test cycles.
3.1.1. Prioritize the requirements based on business priorities and risk

Test manager and test analyst will analyze the RFP and client requirements. The RTDMS instance helps them prioritize test requirements based on business priorities and risks. Test manager can then arrange suitable resources (e.g., human resources, time).

3.1.2. Collect requirements from previous projects to reduce duplication

The RTDMS instance collects requirements from previous projects and traces them to respective test plans and other test artifacts. Test team analyzes the system requirements and searches the earlier similar projects for relevant artifacts to reduce the time and human resources based on previous experiences. Clients review the bids, including brief test plans, from several providers and select the proper provider. After contracting has been completed, the service provider will draft a detailed test plan.

3.1.3. Collect test cases from previous projects

Test analysts will design test cases for the detailed test plan. All the test cases should be reviewed by the test manager before they are added to the detailed test plan and approved by the client before they are used in test execution. To help design test cases, the RTDMS instance collects the test cases from all projects and makes them available for reuse in order to shorten development times and reduce costs.

Test teams can use the RTDMS instance to draft test plans or import test plans from Microsoft Word or Excel. The case company usually imports test plans from Word or Excel because clients typically revise the test plans before approving them and submit the final versions to the case company as Word or Excel files.

3.1.4. Manage interdependencies between requirements and align test requirements and test cases

When the test cases have been designed, the RTDMS instance manages interdependencies between requirements and between test requirements and test cases. Tracing test progress is easy, because it is known which requirements have been tested and what the results of the tests have been. The test cases corresponding to test requirements are also easy to manage and reuse. Test team uses the RTDMS instance to ensure detailed test plans are bidirectionally traced to test requirements, which, in turn, are traced to test cases. The RTDMS instance stores this test project information to the project database. Defining dependencies among test can help test team realistically emulate real-life business process and easily maintain and reuse tests.

When the client has reviewed and approved the detailed test plan, the client will transfer the system and other artifacts to be tested and the needed resources to the service provider. Both sides still need to set up a test environment and train the stakeholders. Test manager ensures appropriate staff attends the training, assigns test assignments to testers, and prepares the test team for initiating test execution.

During the process, coordinator needs to review the brief test plan and the detailed test plan to ensure these plans have been translated correctly and no misunderstandings exist between the parties involved. The plans are only sent to clients after they have been approved by the coordinator. In the end of the initiation phase, coordinator assists the test manager in transferring test resources and in organizing training.

3.2. Delivery

In the delivery phase, testers execute manual and automated test assignments according to the test plan and report results and found defects. Test manager and test analyst need to analyze the test results, identify which tests failed and which steps caused the failure, determine whether one or more defects have been detected in the application, and send defect reports to the client. If no defect caused the test failure, the expected results of the test may need to be updated. Before any new defect is submitted, the RTDMS instance is used to check the project database for similar defects, reducing the documentation of duplicate defects and removing the need for manual checking.

3.2.1. Monitor test process against the test plan

During the test execution process, the RTDMS instance will monitor test progress against the test plan and report progress. Project managers can review quality metrics and decide whether an application is ready for release. Stakeholders can use this information to check the progress against the schedule.

3.2.2. Report execution results and defects

After testers run manual or automated tests, the RTDMS instance will report execution results and defects. The automation scripts and related artifacts are stored into the project database.
3.2.3. Version control and baseline management

The client and the service provider may be geographically distributed. Still they need to work together iteratively in creating software versions, detecting defects, revising the software to correct defects, and so on. Software versions thus change rapidly and clients and providers can easily become confused and work with wrong versions of software and test artifacts. Version control and baseline management are thus especially crucial features of the RTDMS instance in international sourcing to manage multiple versions of software and test artifacts while maintaining data integrity. In addition, clients’ test requirements change. Version control can be used to manage change requests, which should detail the version numbers of test requirements, tests, test scripts, and software components. Versioning allows distributed teams to collaborate, while providing an audit history of changes throughout the sourcing lifecycle. Baselining allows the test team and the client to capture groups of requirements, tests, test artifacts, and defects at strategic points in the project lifecycle to mark specific milestones. The test team can compare baselines to assess the impacts of changes and enable rollback of artifacts, if required.

3.2.4. Identify defects and track them from initial detection to resolution and verification

The main purpose of the delivery phase is to locate application defects efficiently. Defect reports should include detailed defect information such as related requirements, run steps, and related defects. Clients need to repair defects and draft new system versions to continue testing. The client and the service provider can use an online testing platform, which allow developers to review and correct defects logged into the platform and the project database. Test manager assigns the tests to testers to start the testing of the newly formed version. During the execution process, the RTDMS instance will identify defects and offer comprehensive defect tracking from initial detection to resolution and verification. In addition, the instance will search for defects from previous similar projects to help developers and testers figure out the reasons for defects and reduce repair time. In addition, analysis of earlier defects can help service providers probe the steps or factors that might have caused the defects and thus reduce test time and efforts.

3.2.5. Generate impact analysis report

If the client changes requirements, the RTDMS instance will report the affected requirements. The test team needs to evaluate the impacts and risks and adjust the test plan accordingly. If the changes affect the schedules and/or test resources heavily, the test manager may need to renegotiate with the client and revise the test contract. When both parties agree with the changed situation, the test manager organizes a new round of testing.

“Clients try to avoid making major requirements changes because they can be troublesome. If the required changes are great, leading to renegotiations, the clients generally have to choose between costs and quality. If the changes are small, the generated reports can help us quickly analyze the impacts and go forward.” - A test manager

3.3. Completion

After the testers have completed all the predefined test assignments or the service status meets the project closure conditions (e.g., the number of found defects has declined for a specified period), the test engagement between the client and the provider can be closed. The test manager sends the final test report and delivers the tested system and related resources to the client who reviews the test results and makes sure that all the test requirements have been met. When the client has approved the deliverables, it will pay for the service. The test team will summarize the project and archive artifacts to the project database for further reuse.

The artifacts to be stored include test requirements, summarized test cases, and the found defects. According to the industry convention, all the test cases used during the engagement belong to the client. Therefore, service provider has to summarize or generalize the useful test cases and store only the summaries or the generalized test cases into the project database. The found defects can be reused by both parties to reduce repair time and improve service quality. In addition, lessons learned from the testing engagement need to be accumulated to improve services in the future. Testers provide valuable feedback concerning the test artifacts they have reused (e.g., generalized test cases) that can be deployed to improve the designs of the artifacts and the usefulness of the test artifact repositories.

4. A meta-design of the design product theory for RTDMS

This section outlines a generic meta-design for RTDMS based on the analyses of interview transcripts, the testing life-cycle, and the literature review. The meta-design covers the entire testing life-cycle outlined
in Section 3 and visualized in Figure 1. The section concludes by explicating the linkages between requirements management, test management, and defect management subsystems to validate the metadesign and to justify its scope.

The most central classes of artifacts managed by RTDMS instances are requirements, test executions, and defects. Test execution artifacts are associated with all other test artifacts such as test cases and automated test scripts. The relationships between these artifacts are explained next. Test requirements are based on client requirements and RFP; each test requirement needs at least one test execution to verify and validate it; each defect is detected by a specific test execution; and a defective feature or subsystem does not meet one or more specific test requirements.

This section introduces generic structures and attributes of the three classes of artifacts presented above. According to the design product theory, RTDMS instances should include at least these structures and attributes to be effective.

4.1. Requirements

Table 3 presents the generic structure of requirements artifacts. In the following, each class within the structure is explained.

*Description* describes what a requirement is about and the purpose of the requirement. Version indicates the version number of the requirement. Name and ID are used for identification and traceability.

*Origin* describes which client requirement the test requirement comes from. Test requirements are based on specific client requirements and several test requirements may be needed to meet one client requirement.

*Analysis* is used to probe the implications of the requirement. Priority is used to rank requirements and arrange suitable resources and efforts. During the service delivery phase, status can be used to check whether the requirement has been met.

*Workflow* describes what should be done next to this requirement and by whom. Test manager needs to allocate the requirement to specific test execution assignment and testers.

*History* is used to provide information about all prior edits of various requirement attributes and the responsible editors. As a result, the stakeholders can be held accountable for their actions and unexpected service breakdowns can be dealt with effectively. Changed requirements may necessitate unexpected revisions of test plans and raise service risks. History information helps test teams to proactively eliminate many breakdowns and to recover from breakdowns to continue test execution.

4.2. Test execution

Table 4 presents the generic structure of test artifacts. In the following, each class within the structure is explained.

*Description* describes the purpose of test execution. Version indicates the version number of the test execution. Name and ID are used for identification and traceability.

*Origin* describes the test requirement(s) the test execution refers to. One test requirement may need more than one test execution.

<table>
<thead>
<tr>
<th>Class</th>
<th>Questions</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>What is the requirement about?</td>
<td>Name, ID, Description, Rationale, Version</td>
</tr>
<tr>
<td>Origin</td>
<td>Which client requirements does the requirement refer to?</td>
<td>Author, Source, Date of creation</td>
</tr>
<tr>
<td>Analysis</td>
<td>What are the implications of the requirement?</td>
<td>Status, Required effort, Priority</td>
</tr>
<tr>
<td>Workflow</td>
<td>What should be done to this requirement next? By whom?</td>
<td>Allocation to tester</td>
</tr>
<tr>
<td>History</td>
<td>What has been done to the requirements? When?</td>
<td>Information about all prior edits, editors, and changes</td>
</tr>
</tbody>
</table>

Table 3. Generic Structure of Requirements

<table>
<thead>
<tr>
<th>Class</th>
<th>Questions</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>What is the test execution about?</td>
<td>Name, ID, Description, Rationale, Version</td>
</tr>
<tr>
<td>Origin</td>
<td>Which test requirement does the test execution refer to?</td>
<td>Author, Source requirement, Data of execution</td>
</tr>
<tr>
<td>Analysis</td>
<td>What are the implications of the test execution?</td>
<td>Status, Required effort, Priority</td>
</tr>
<tr>
<td>Related test artifacts</td>
<td>Which test cases and other test artifacts are involved in this test execution?</td>
<td>IDs of test artifacts to be used</td>
</tr>
<tr>
<td>Workflow</td>
<td>What should be done to this execution next? By whom?</td>
<td>Allocation to testers</td>
</tr>
<tr>
<td>History</td>
<td>What has been done to the test execution documentation? When?</td>
<td>Information about all prior edits, editors, and changes</td>
</tr>
</tbody>
</table>

Table 4. Generic Structure of Tests
Analysis is used to probe the implications of test execution. Priority describes the priority of the execution and status refers to the execution results. Required effort describes the test execution costs, time, and resources, which can be used to calculate the total service cost of an engagement. This information can be reused to estimate the profitability and feasibility of future engagements.

Related test artifacts provides traceability links to the artifacts (e.g., test cases) that are involved in the test execution.

Workflow describes the steps of test execution and who is responsible for the steps.

History is used to document the prior edits of test execution information.

4.3. Defect management

Table 5 presents the generic structure of defect artifacts. In the following, each class within the structure is explained.

Description explains the defect. Name and ID are used for identification and traceability.

Origin describes who detected the defect when and using which version of the test execution.

Analysis probes the implications and risks of the defect, the severity of the defect, and the amount of effort needed to repair it in order to prioritize and schedule defect fixing. This information can also be reused in future to help the service provider and the client reduce repair time and improve service effectiveness.

Workflow describes the steps that should be taken to fix the defect and by whom. The developers need to repair defects and form a new system version.

History provides the information about all prior edits of the defect information.

4.4. Validating and scoping the design product theory for RTDMS

RTDMS instances help test managers and test analysts to prioritize and valuate requirements. The requirements and requirements interdependencies can be stored in the project database. The prioritization and valuation methods are beyond the scope of this paper.

Most testing resources are allocated to deal with the highest priority requirements. RTDMS instances make it easy to trace test executions and defects, because all test artifacts and defects are bidirectionally linked to requirements. Each test execution clearly indicates which test requirements provide the purpose for the execution and which defects have been detected by the specific test execution. Version management and baselining help identify the stakeholders involved with different versions of different artifacts and the actions the stakeholders have taken. Therefore, stakeholders can control requirements changes, analyze the impacts of requirements changes, and revise the test processes and test plans as necessary to meet the most important requirements and service breakdowns.

<table>
<thead>
<tr>
<th>Class</th>
<th>Questions</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>What is the defect about?</td>
<td>Name, ID, Description, Rationale, Version</td>
</tr>
<tr>
<td>Origin</td>
<td>Who created the defect information? Who detected the defect?</td>
<td>Author, Source, test execution, Date of defect discovery</td>
</tr>
<tr>
<td>Analysis</td>
<td>What are the implications of the defect?</td>
<td>Status, Priority, Risk, Required repair effort</td>
</tr>
<tr>
<td>Workflow</td>
<td>What should be done to this defect next? By whom?</td>
<td>Allocation to developers</td>
</tr>
<tr>
<td>History</td>
<td>What has been done to the defect? When?</td>
<td>Information about all prior edits, editors, and changes</td>
</tr>
</tbody>
</table>

Table 5. Generic Structure of Defects

5. Conclusions and further research

This research established the meta-requirements and the meta-design of the design product theory for RTDMS. RTDMS instances can be used by any testing service provider for testing projects. The validity of the theory was enhanced by a case study, which involved a leading Chinese testing service provider, and the analysis of HP Quality Center, which has been widely used by testing service providers, including the case organization, all over the world.

RTDMS instances are expected to enforce standardized processes for service providers and the implementation of best practices across projects. Test analysts, developers, and testers can share and reuse artifacts from project database across projects to raise productivity and quality. Developers can share knowledge about defects across projects to increase efficiency and reduce risks. Test managers and clients can aggregate quality metrics across projects.

This research is limited to one case study but it must be noted that the case organization is not only a leader in its field in China but its key staff members interviewed for this study have more than ten years of work experience in testing services. As a result, the generalizability of the design product theory is expected to be significant. Future research needs to validate the theory using more case studies and help service providers to better reuse test artifacts. New case studies and action research projects are thus necessary.
to make the theory even more credible for testing 
service providers, information systems designers, 
testing professionals, and researchers.

6. References

Challenges, Dreams, Future of Software Engineering, IEEE, 
2007.
Development”, Information Systems Management, 17(1), 
2000, pp. 92-95.
Research”, The Academy of Management Review, 14(4), 
Strategic Issues”, Staffordshire University, 1996.
Management Revisited”, ACM Sigsoft Software Engineering 
Notes, 30(2), 2005, pp.1-4.
[8] Gupta, M. and M. Prakash, “Possibility of Reuse in 
Conference in India, 2006.
Institute Worldwide Annual Software Testing Conference, 
and the Test Development Process: A Combined Approach”, 
International Conference on Software Reuse (ICSR), LNCS 
[11] Li, Q., Y. Yang, M.S. Li, Q. Wang, B.W. Boehm, and 
Prioritization to Make Winners of Success-critical 
Stakeholders”, Journal of Software Maintenance and 
from the Engineering Department to the Executive 
85-103.
C.N. Liu, “A State of the Practice Study on Communication 
and Coordination Between Chinese Software Suppliers and 
Their Global Outsourcers”, Software Process Improvement 
Assurance”, IGI Global, 2009, pp.245-263
Communication Practices in Managing Relationships with 
Offshore Vendors of Software Testing Services”, 
Communications of the Association for Information Systems, 
[16] Poulin, J., “The Business Case for Software Reuse: 
Reuse Metrics, Economic Models, Organizational Issues, and 
Case Studies”, 9th International Conference on Software 
Management of Software Testing”, Value-based Software 
[18] The eSourcing Capability Model for Service Providers 
[19] Van der Linden, F., K. Schmid and E. Rommes, 
Software Product Lines in Action: The Best Industrial 
“Building an Information System Design Theory for Vigilant 
Figure 3: Testing Sourcing Life-Cycle enabled by Integrated Requirements, Test, and Defect Management Systems

Client
- Request for Proposal: Documented system requirements
- Review the test plan and ask the provider
- Select the provider and sign the contract
- Transfer the system to be tested and other resources to the service provider; Set up a test environment and train the service provider
- Change requirements
- Impact and accept the test results
- Close the engagement

Coordinator
- Assist the client analyze the requirements
- Verify the brief test plan
- Verify the detailed test plan
- Coordinate resource transfer and training
- Organize training and arrange test resources for assignments
- Allocate test assignments to testers
- Send the final test report and deliver the tested system and other resources
- Archive artifacts to Project Database for future reuse

Test Manager
- Analyze and prioritize the system requirements and create a business case
- Draft a brief test plan for bidding
- Draft a detailed test plan
- Review test cases
- Analyze test results and send defect reports
- Analyze risks and revise the test plan

Test Analyst
- Draft a test case
- Design test cases
- Design and run manual and automated tests
- Complete test assignments and meet the closure conditions

Tester
- Help prioritize the requirements based on business priorities and risks
- Find and reuse requirements from previous similar projects
- Find and reuse test cases from previous similar projects
- Track interdependencies between requirements and align test requirements and test cases
- Monitor the test progress against the test plan
- Version control and backup management
- Track defects from initial detection to resolution and verify, store, share, and reuse defect information
- Generate the impact analysis report

Project Database
- Requirements artifacts from previous similar projects
- Test artifacts from previous similar projects
- Tracability links between requirements, test, and defect artifacts
- Automated test scripts and related artifacts
- Defect artifacts
- Engagement related artifacts