Section 4
Bidder's Products, Methodology, and Approach to the Project

4.1 FACTS II Requirements Summary	4.11 Interfaces
4.2 Functional Requirements	4.12 System Development
4.3 Technical Requirements	**4.13 System Testing**
4.4 Customer Relations Management Tools	4.14 System Training
4.5 Project Initiation and Management	4.15 Conversion
4.6 System Hardware	4.16 System Implementation
4.7 System Planning and Analysis	4.17 Post Implementation Support
4.8 Requirements Verification	4.18 Support Federal Review
4.9 System Design	4.19 Security
4.10 Reports	

DE_SACWIS-002m_4

4.13 System Testing

RFP reference: 6.13 System Testing, Page 52

System Testing is composed of all phases of testing, including Unit/System, Integration, User Acceptance testing and the test plan (including test strategy and scripts) will tie back to the requirements using the traceability matrix. Additionally, as applicable to the design and implementation methodology used, a full Regression Test will be conducted at the conclusion of each major phase of testing to verify that the application is ready to move to the next level of testing. Finally, depending upon the implementation strategy proposed, Pilot Testing may be conducted following User Acceptance Test. Bidders may also propose additional types of testing that could be conducted, such as Load testing or Disaster Recovery testing, and the advantages to the State of including these types of tests. Bidders should describe, in detail, their best practice approaches to each phase of testing and the criteria used to measure the success of each level of testing prior to moving to the next level. The selected Vendor will be required to develop a test plan for each level of testing for each phase of development. This plan will be submitted to DSCYF project staff for approval. The Vendor must provide the State access to the test results as each test is performed. The testing phases are described in detail below.

At the States discretion the code may be submitted to a third party for code efficiency and security vulnerability testing. The results of these tests will be available to the State and any necessary modifications will be tracked. Code testing may occur multiple times during the project life cycle.

Our approach to system testing is based on our lessons learned on SACWIS implementations similar in size and complexity to Delaware FACTS II, which allows us to create a business driven and technically sound application. We support our approach to testing and reduce overall project risk with CMMI certified processes and are compliant with the Project Management Book of Knowledge (PMBOK) best practice standard for project management.

Our objective for System Testing is to provide Unit, System, Integration, Regression, and User Acceptance testing that produces a high-quality, high-performing application that meets your system requirements and design. Our System Testing approach and methodology is detailed and includes multiple levels of testing of the functionality and application. Each layer of testing achieves a higher level of application stability.

Deloitte's functional and testing teams consists of experienced staff in testing processes and methodologies for social services implementations. Our approach to System Testing is based on our national HHS experience and is designed to provide early issue identification, a more business-oriented system, and greater state stakeholder buy-in.

Table 4.13-1 below summarizes the features of Deloitte's approach to Delaware FACTS II System Testing and the benefits to DSCYF.

Features of Our Testing Approach and Methodology	Benefits to DSCYF
Employ an established testing approach that: • Applies CMMI mature testing processes tailored to Delaware FACTS II • Leverages Deloitte's prior experience with SACWIS implementations to assess system compliance with requirements	• Lowers cost and schedule risk due to better validation and avoids need for rework • Users obtain modifications with improved system acceptance
• Realistic, business-oriented scenarios • Early identification and discovery of system anomalies in lower test environments • Stable code graduated to higher test and production environments	• Lowers downstream risk by discarding the majority of system issues in early lower environment level testing • Produces a stable Delaware FACTS II system that scales to meet end-user usage demands
Close coordination with DSCYF Business and Testing Teams to: • Provide solutions based on input from DSCYF SMEs • Communicate early identified system anomalies	• Improves DSCYF management awareness of testing progress and obstacles • Reduces testing cycle time and lowers risk and downstream impacts through better solution identification
Use of *SACWISMate*, a Systems Development Life Cycle (SDLC) tracking tool: • Produces repeatable test scripts • Increases code coverage • Produces clear and concise test result reporting	• Lowers deployment risk • Decreases cycle time • Improves resource productivity • Improves system acceptance by users • Robust regression tested solution

Table 4.13-1. Features and Benefits of Deloitte's Testing Approach for Delaware FACTS II.

Our System Testing Approach and Methodology

Deloitte understands that System Testing for Delaware FACTS II is comprised of all phases of testing, including Unit/System, Integration, Regression, and User Acceptance testing.

We extend our time-tested FACTS II Playbook methodology for System Testing to a deliver quality implementation and meet DSCYF's testing requirements. We also bring strong testing experience, tools, robust testing processes and methodology, deep industry knowledge, and demonstrated insight into DSCYF's business operations and systems. In

addition, our SACWISMate test script tracking module documents and tracks test scripts, test runs, and testing results through each testing cycle. (For additional detail on SACWISmate, see *Section 4.0 – Bidder's Products, Methodology and Approach to the Project.*).

Table 4.13-2 below shows the various test phases and their description.

Test Phase	Test Phase Description
Unit Testing	Unit testing is used to verify the input and output for each module. Successful testing indicates the validity of the function or sub-function performed by the module and confirms traceability to the design. During unit testing, the developer tests each module individually and verifies the module interfaces for consistency with the design specification. During unit testing, actual results of the important processing paths through the module are compared with expected results. In addition, the developer tests error handling paths.
System Testing	System testing is carried out by the Functional Track Leads to validate the functionality and confirm that all business requirements are met as expected. System test confirms that the system performs properly, both from a functional and technical perspective.
Integration Testing	Integration testing is the responsibility of the Functional Track Leads following the successful completion of system testing and the integration of newly created or modified code with the existing code base. We perform end-to-end testing of the code changes in relation to the business process and technical coordination of individual units or modules with the larger system. Performance and Disaster Recovery testing is conducted parallel with Integration Testing: Performance Testing: The objective of performance testing is to exercise Delaware FACTS IIto observe and analyze performance characteristics, and to identify load-related problems. Disaster Recovery Testing: A business continuity management process is defined to minimize the impact on DSCYF in the event of a system outage. It defines the process to recover from the loss of information assets through a combination of preventive and recovery controls. Consideration is given to the consequences of disasters, security failures, loss of service, and service availability.
Regression Testing	Regression testing verifies that system modifications have not caused unintended effects and that the existing software or system components still comply with specified requirements. Regression testing occurs in every stage of testing pre-implementation as well as post implementation.
User Acceptance Testing	Acceptance testing is performed by the DSCYF prior to accepting for the code promotion and the overall system readiness confirming that the system meets mutually agreed-upon requirements. The results of these tests give confidence to DSCYF as to how the system performs in production. This is a fully functional version of the application, where scenarios of great complexity are tested, as code is promoted.

Table 4.13-2. Phases of Deloitte's Testing Approach for Delaware FACTS II.

Testing of the Delaware FACTS II solution for all testing phases is aligned with our FACTS II Playbook methodology and is fully compliant with Department of Technology and Information (DTI) project management methodology. Our well-defined process of verification at all stages of the life cycle confirms that artifacts and deliverables are reviewed and verified by respective leads or experts.

Our iterative testing approach (Figure 4.13-1 below) illustrates important testing steps for each testing phase within the overall Systems Development Life Cycle (SDLC):

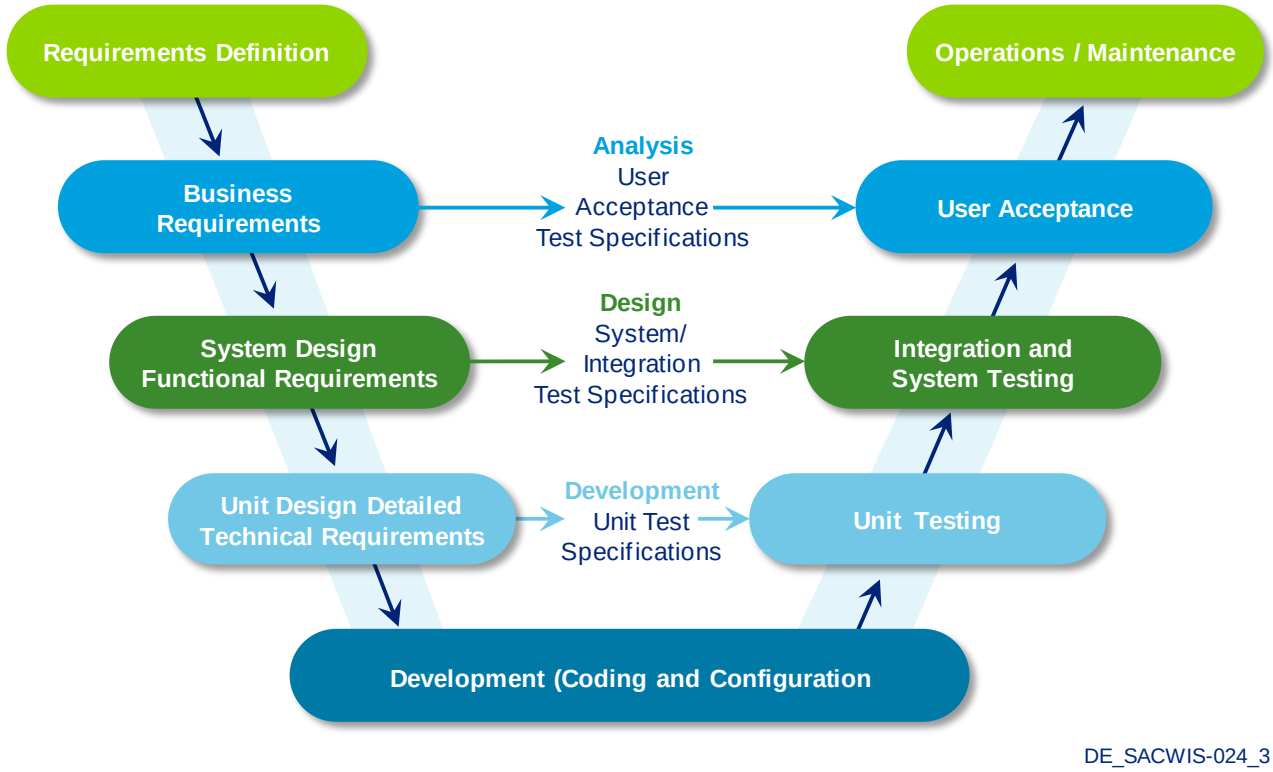


Figure 4.13-1. Software Testing Approach.
Deloitte's Testing Strategy is Incorporated into the Software Development Life Cycle.

Deloitte's Testing Team is experienced in testing processes and methodologies, and brings deep experience in child welfare integration implementations to Delaware FACTS II. Our Integrated Case Management Subject Management Expert (SME) serves as the Delaware FACTS II Test Manager, and participates in requirements and design sessions. The Case Management SME has worked in Delaware since 1993 and brings his experience and understanding of Delaware's culture to validating that Delaware FACTS II is designed per DSCYF's business objectives.

Deloitte's Functional and Testing Teams participate in requirements and design sessions to understand the true attributes of functionality, underlying business processes, and system interactions, and to provide quality deliverables and mitigate risks. In addition, we provide DSCYF with an orientation of each phase of testing with test plans early in the project to allow you sufficient review time, which mitigates risk to development and overall project timelines. The feedback process emphasized by Deloitte's testing approach

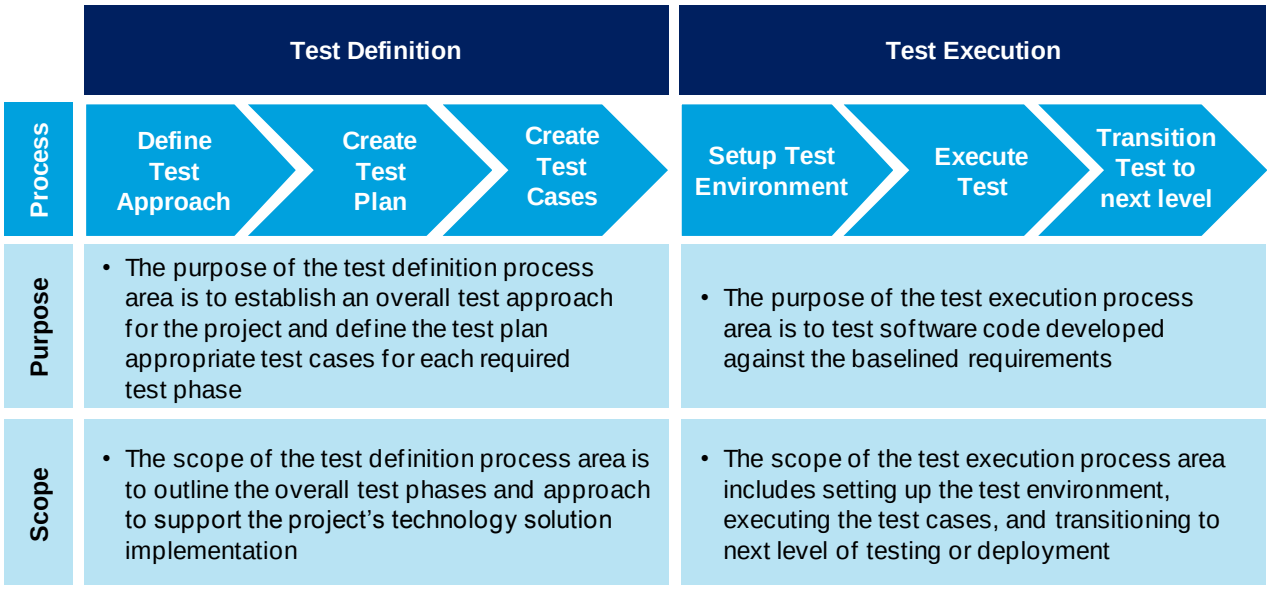
enables development aligned with business intent, improving the quality of code delivery to testing teams.

The Deloitte teams responsible for conducting all phases of system testing are outlined in Table 4.13-3 below.)

Test Phase	Responsibility
Unit Testing	Deloitte Development Team
System Testing	Deloitte Functional Team Leads
Integration Testing	Deloitte Testing Team
Regression Testing	Deloitte Testing Team
User Acceptance Testing	DSCYF with Deloitte Testing and Functional Team support

Table 4.13-3. Deloitte Team responsibility for each phase of System Testing.

Our overall testing approach includes the activities described in the Figure 4.13-2 below.



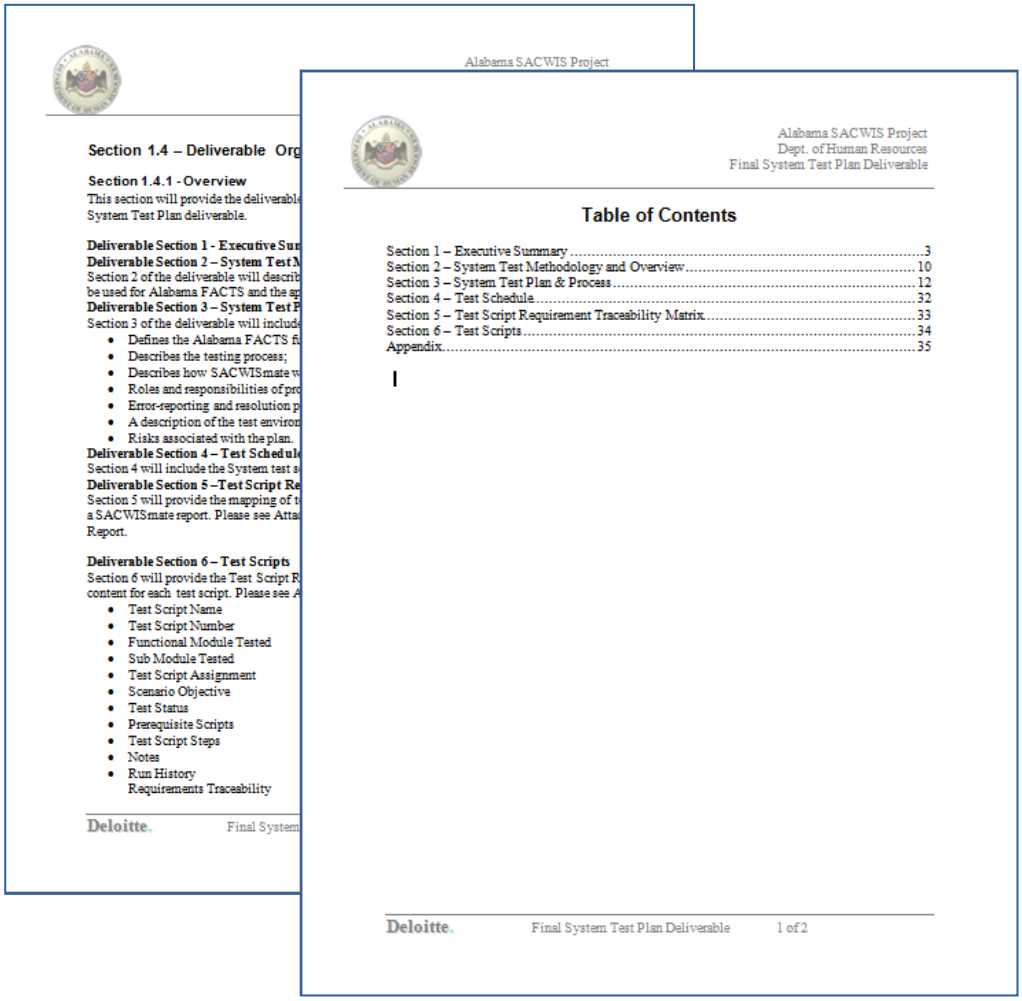
DE_SACWIS-025

Figure 4.13-2. Testing Techniques and Methodology.
Illustration of Test planning, test designing and execution.

Each phase of system testing – Unit/System, Integration, Regression, and UAT -- follows the following four phases:

- **Planning for System Testing**. Deloitte develops System Test Plans for Delaware FACTS II for each phase of Systems Testing. The System Test Plan is submitted to DSCYF project staff for approval.
- **Preparing for System Testing**. In collaboration with DSCYF, Deloitte establishes a separate testing environment for each phase of System Testing.

- **Conducting System Testing**. Testing for each testing phase is conducted, deficiencies are recorded and corrected, and test cases are then re-tested.
- **Report Test Results**. The System Test Results Report tracks and summarizes the test case results, including identified defects and their corrective action. Deloitte documents the results of system testing activity in the System Test Results Report deliverable. We work with DSCYF to define an appropriate format for this document. Sample table of content for the System Test Plan are found below:



Alabama SACWIS Project
Dept. of Human Resources
Final System Test Plan Deliverable

Table of Contents

Section 1 – Executive Summary 3
Section 2 – System Test Methodology and Overview 10
Section 3 – System Test Plan & Process 12
Section 4 – Test Schedule 32
Section 5 – Test Script Requirement Traceability Matrix 33
Section 6 – Test Scripts 34
Appendix 35

Deloitte. Final System Test Plan Deliverable 1 of 2

DE_SACWIS-1516

Figure 4.13-3. Sample Table of Contents for System Test Plan.
The System Test Plan provides a description of System Test methodology, processes, test schedule, test script requirements and test scripts.

In the sections that follow, we describe our approach and strategies for each testing cycle following the above mentioned four phases.

4.13.1 Unit/System Testing

RFP reference: 8.13.1 Unit/System Testing, Page 52

Unit testing is defined as the first level of testing of the application and is completed at the component level. For example, a unit test might be conducted to verify that data entered at the application layer is correctly saved to the data layer. Unit testing ensures that graphical user interface (GUI) standards are met, that component functions work as expected, and that the presentation, business logic, security, and data layers perform the discrete function as designed. Unit testing must be successfully completed before the code is migrated to the System test environment. Unit testing will be completed by the Bidder's development team, using standard methods described in the Unit Test Plan as well as a standard template, form, or checklist to ensure that testing is done consistently and thoroughly. It also ensures that the application meets the expected criteria as defined in the Application Standards document.

In their proposal responses, Bidders should identify their Unit and System test strategies, best practices, and tools used. Inclusion of sample checklists or scripts is desirable.

Deloitte's Testing practice provides service offerings delivered through extensive testing capabilities, covering the entire testing solutions landscape from strategy, plan, execution, and support across leading industry sectors and technologies. While constructing the FACTS II testing approach, we leverage leading practices from our national network of health and human service implementations, the Deloitte Testing Center of Excellence (CoE), and most importantly our experiences and lessons learned on other SACWIS projects. The figure below illustrates Deloitte's overall System testing methodology.

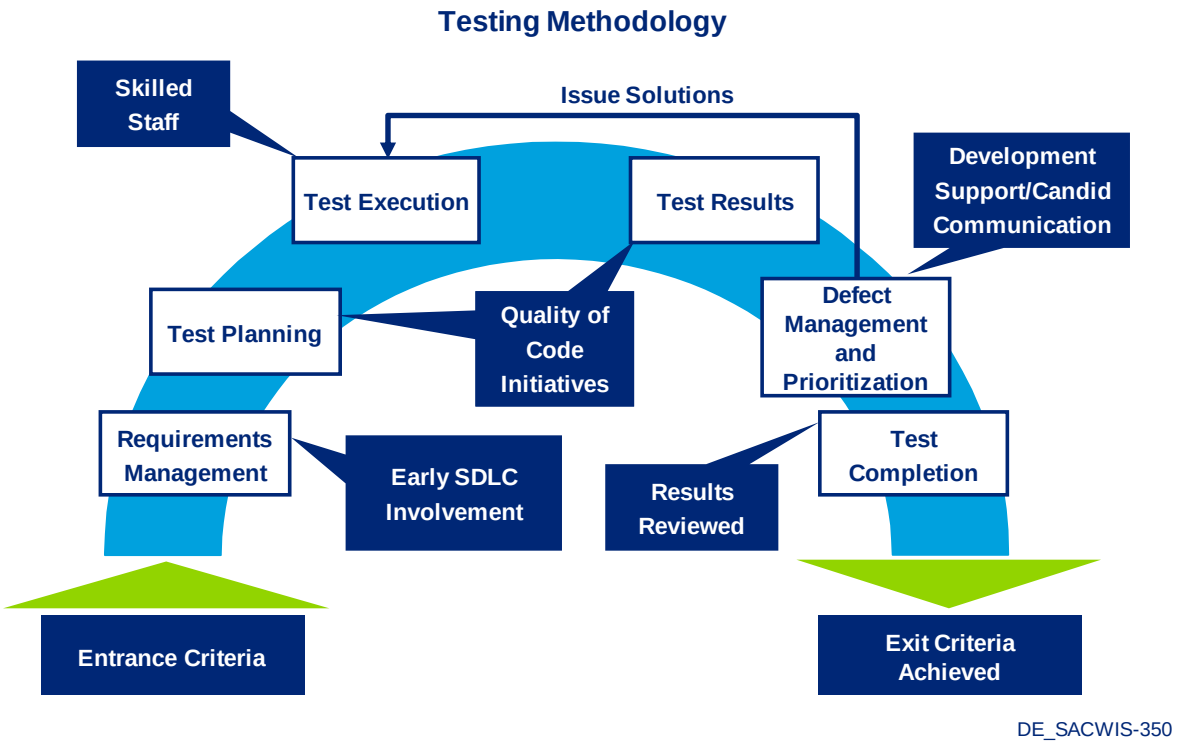


Figure 4.13-4 Overall SystemTesting Methodology.
Our overall system testing methodology focuses on achieving entry and exit criteria for each test to produce stable software, lower risk, and improve the quality of delivered software.

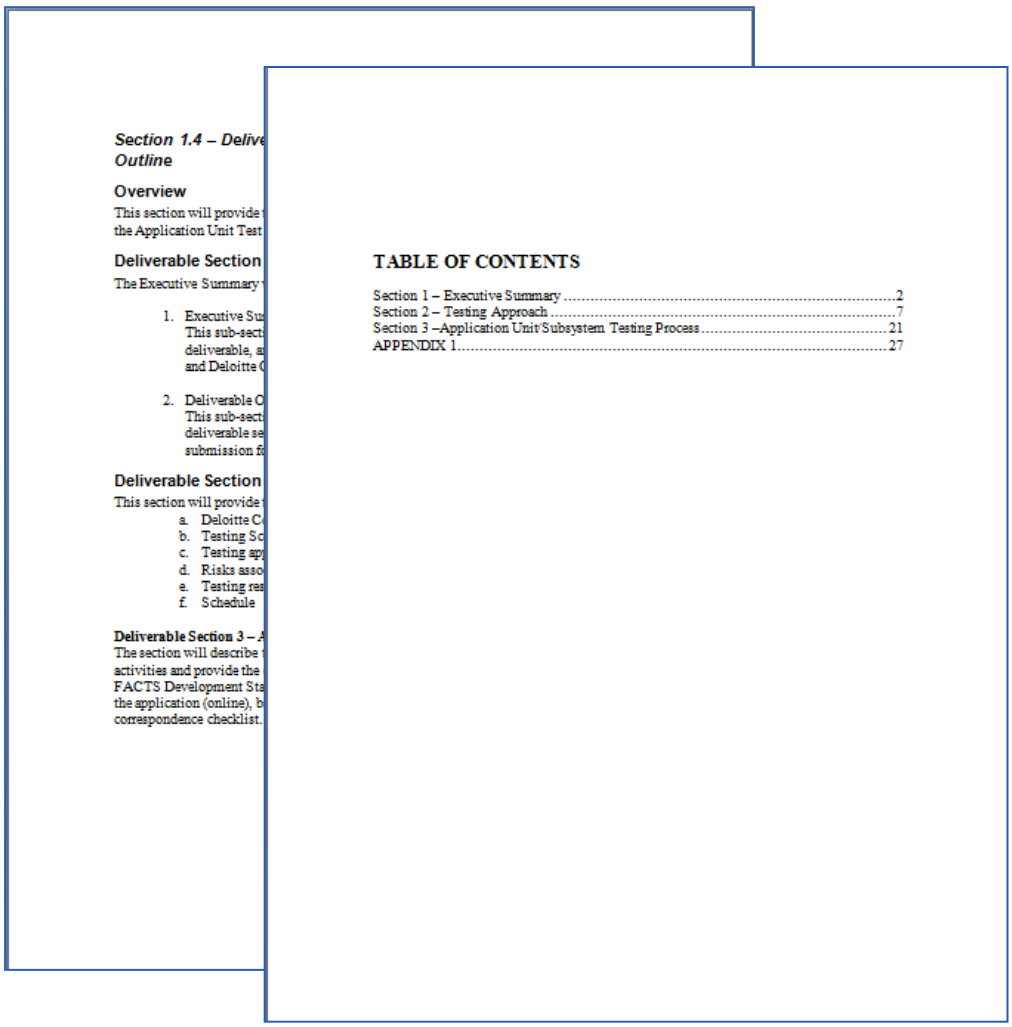
We provide you with test plans early in the project to afford sufficient review time, thus mitigating risk to development and overall project timelines. The feedback process emphasized by Deloitte's testing approach enables development aligned with business intent, improving the quality of code delivery to testing teams.

Unit Testing

The first level of testing Delaware FACTS II, the objective of unit testing is to test single units of code in isolation. Unit testing ensures that the graphical user interface (GUI) standards are met, that component functions work as expected and that the presentation, business logic, security and data layers perform the discrete functions as designed.

Planning and Preparation

As the Deloitte Development Team completes work on individual software components, the unit testing effort begins. Deloitte develops a Unit Test Plan for DSCYF staff approval. In collaboration with DSCYF, establishes a separate testing environment for Unit Testing. A sample table of contents for the Unit Test Plan is included in the figure below.



DE_SACWIS-1514

Figure 4.13-5. Sample Table of Content for Unit Test Plan.
Unit testing is used to verify the input and output for each module. The Unit Test Plan includes the methodology, scope, approach and process, associated risks, testing resources, roles and responsibilities and the test schedule.

Unit Testing Strategies

Each developer is responsible for the unit testing of their modules to begin the software quality assurance process, attempting to identify defects early in the process. Our goal is to focus on delivering a testable application to the functional team for System Testing. Unit Testing must be completed for the code to be moved into the System Testing environment.

Within our DC FACES.NET application, a definition of a unit of work for screens, batch processes, and reports is simple – each screen, process, or report generally maps to a single unit of work. Given the distributed and diverse nature of components that make up the online application (the presentation and interaction layer, the application layer and the business rule layer) the definition of a single unit of work is not so clear cut. For this reason, unit testing is further divided into white box testing and black box testing.

Upon completion of a compliable unit of work, the programmer begins white box testing. This approach allows the programmer to use their knowledge of the internal structure of their module ("inside the white or transparent box") to guide their testing activities and the selection of test data. Using this approach, the programmer can test each of the logic branches contained within the module and verify that the expected results are generated. This white box approach is necessary for the first round of testing because it allows programming staff to manually enter input parameters into their module. This overcomes the limitations placed on testing caused by interdependencies between modules. For example, in the child welfare side of the application, the foster parent licensing functionality provides input to the child placement functionality. In the absence of the white box approach, testing of child placement must wait until foster parent licensing has been completed. Given this focus on the internal workings and conditions and boundaries of individual modules, white box testing alone cannot deliver a comprehensively tested module.

Therefore, we employ black box testing to measure the performance of the module against the business requirements. Although black box testing is first introduced as a component of Unit Testing, the concept of black box testing is carried throughout the entire testing process. As its name implies, this approach treats the functionality to be tested as a "black box." the programmer cannot use their knowledge of the internal workings of the module to shape their testing activities. This mode of testing has two main goals:

did you KNOW?

Over 500 Unit Test Checklists were completed for Alabama.

- Confirm that the module behaves as expected under normal operating conditions, rather than the "laboratory" conditions of the white box step. For example, a hidden dependency on code residing within the programmer's development environment is unlikely to be discovered during white box testing
- Employ unit test cases to confirm that the module meets each of the functional requirements specified during the design phase

The table below gives a snapshot of the different dimensions/components tested during Unit Testing.

Test Component	Description
User Interface Unit Test	Our unit testing approach for online screen functionality focuses on navigational paths to follow, GUI standards, error handling for negative testing, data handling etc. Deloitte developers subject every change to the user screens to this testing.
Business Layer Unit Test	This testing focuses on positive as well as negative testing of the component. The component is tested to confirm appropriate handling of both valid as well as invalid data. Test scenarios address testing with valid as well as invalid data and Deloitte developers perform this kind of thorough testing.

Test Component	Description
Data Layer Unit Test	Test data transactions for updates, inserts and deletion against a set of expected results. Deloitte developers thoroughly test the data component to maintain data integrity and avoid any data issues.
Data Security Unit Test	This test is to validate that the component allows access to only users that have the appropriate security credentials.
Unit Integration Test	This is a final test to validate the overall functioning with a specific focus on testing the user interface, business layer and data layer together. Deloitte developers are responsible for performing testing to deliver quality code for the next phase of testing.

Table 4.13-4. Unit Test Components.

Unit Test Checklist

Deloitte developers document unit testing activity using the Unit Testing Checklist feature in SACWISMate's Incident Tracking Module. Results are used to generate the Unit Test Results Deliverable directly from SACWISMate. Following a testing checklist aids in the common testing practices for all members of the development team to follow, as well as to document the results. Figure 4.13-4 shows the Checklist in the Incident Tracking module. The Deloitte team works with DSCYF to make any modifications to this checklist if required.

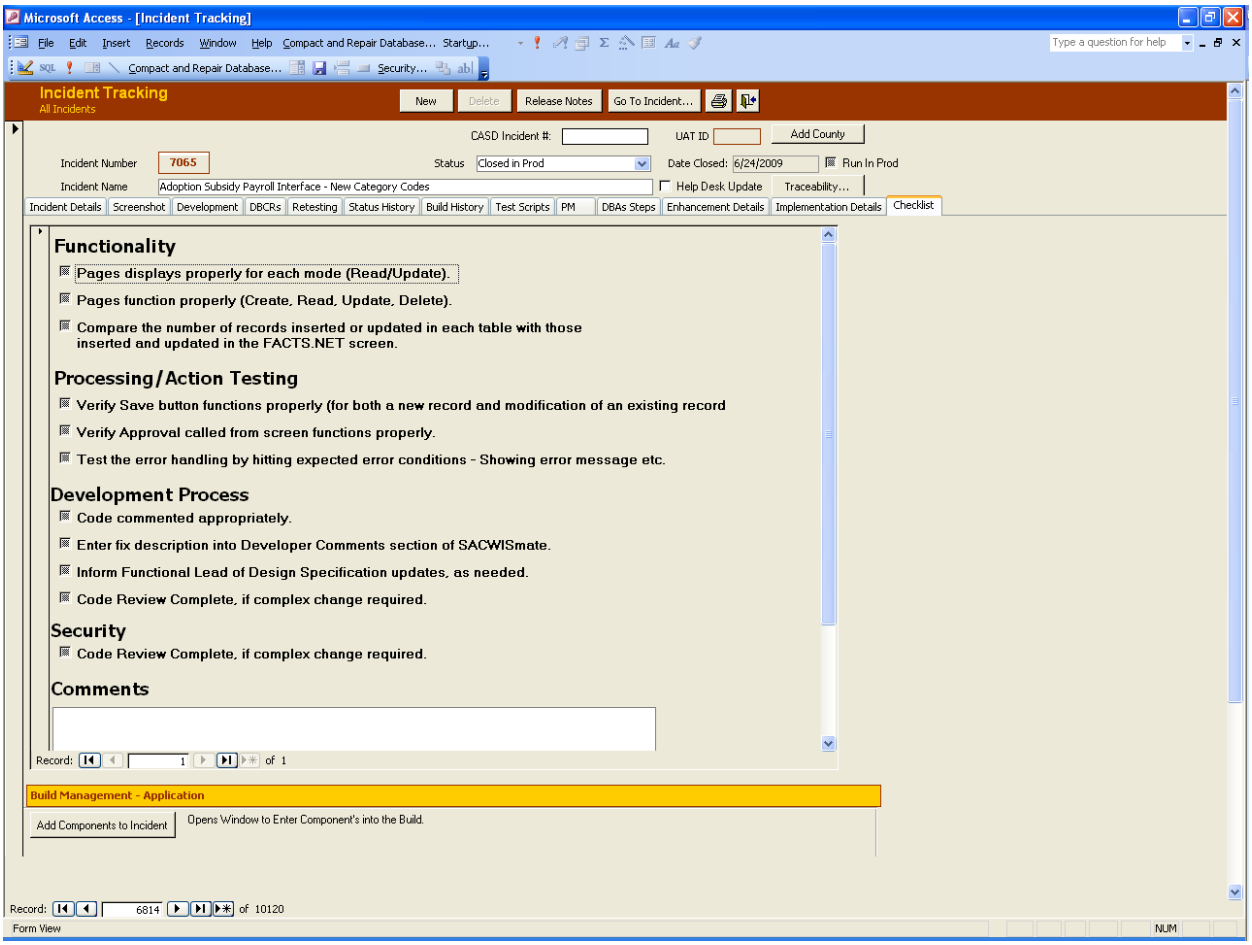
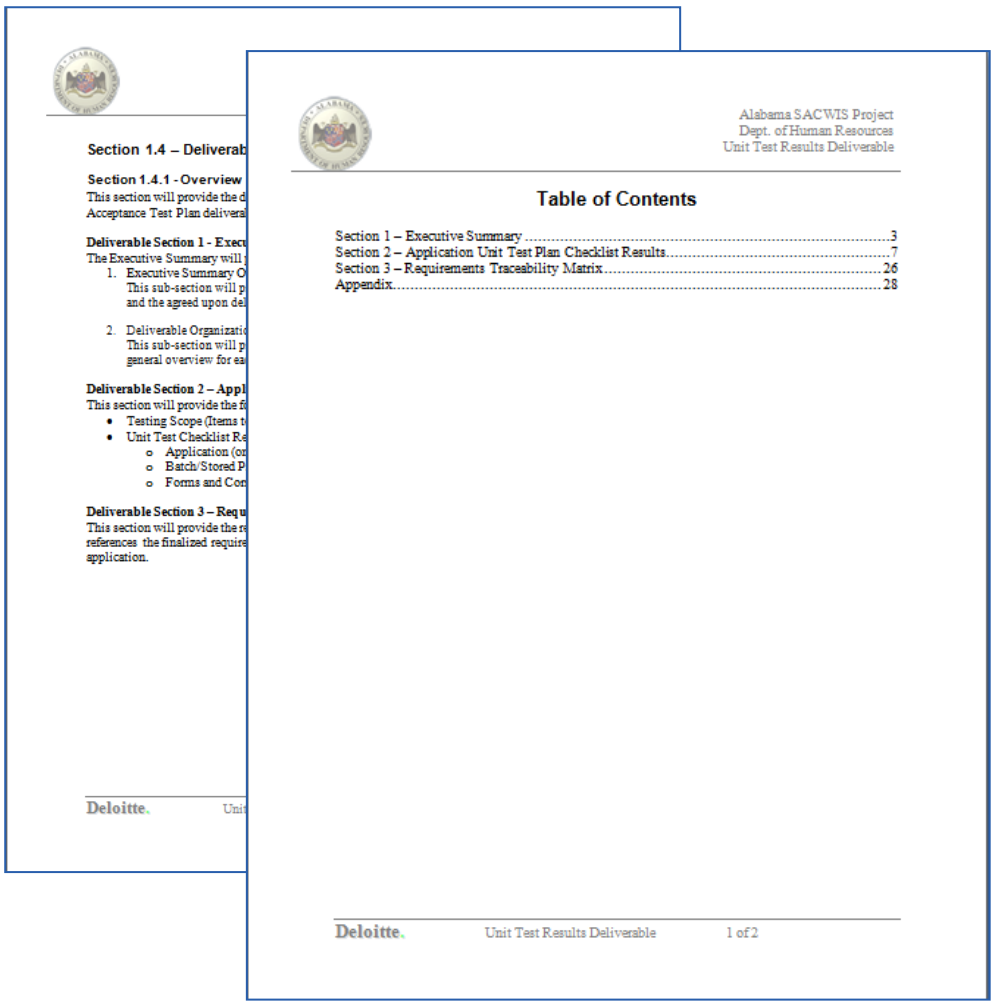


Figure 4.13-6. Sample Snapshot of a Test Scenario with Detailed Steps which form the Test Scripts Executed by our System Testers.
The Deloitte system test team documents test scripts within test scenarios created in SACWISMate.

If defects are found, the developer documents the defect, reviews the design specification, fixes the defect, and re-runs the unit test. This cycle continues until the unit of code has successfully complied with the checklist. The Functional Team Lead monitors the development unit test cycle, reviews and signs off on the checklist, and determines that the code is accepted for migration to the system test environment.

The SACWISmate tool allows DSCYF to monitor the development/unit test process as the Deloitte team progresses through this process. Unit test checklists are compiled, reviewed, and submitted to DSCYF as the Unit Test Results Report deliverable. A sample table of contents of a Unit Test Results Report is found below.



DE_SACWIS-1515

Figure 4.13-7. Sample Table of Contents for a Unit Test Results Report.
The Unit Test Results Report includes a detailed presentation of the testing scope (items tested and unit checklist results.

System Testing

RFP reference: 8.13.1 Unit/System Testing, Page 52

System testing is defined as functional testing and is completed at the function level. For example, system testing could verify that the Inquiry and Screening Module works as designed and that it supports the business processes used by intake workers. System testing ensures that the application meets the functional requirements of the system and is usually completed through the execution of test scripts adapted from typical business scenarios.

System testing will be completed by the Bidder's functional team, using standard methods described in the System Test Plan. Bidders may propose either manual or automated testing processes, or a combination of both. If automated processes are recommended, Bidders must provide information about the proposed testing tool(s).

In their proposal responses, Bidders should identify their Unit and System test strategies, best practices, and tools used. Inclusion of sample checklists or scripts is desirable.

Planning and Preparation

Following successful unit testing, code changes are promoted from the development environment to the system test environment and undergo system testing. Deloitte develops the Systems Test Plan for DSCYF staff approval and, in collaboration with DSCYF, establishes a separate testing environment for System Testing.

System Testing Strategies

System testing is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. In this environment is where the Functional leads evaluate if the specified standards of quality are met. Any defects found within the system test environment in that system functionality can only be modified by code changes promoted to that environment. This restriction confirms that necessary components of a change, including cross-functional items, have been completed and promoted. If system testing is not successful, then the code changes are not promoted beyond the system test environment, maintaining the stability of higher environments.

We conduct manual testing during the system test phase. The Deloitte team practices the above discussed testing processes and approach for each of the following test components. Our testing process analyzes the functionality of software along the following dimensions:

- **Presentation and Interaction**. How does the software look and feel? More importantly, how well do the models and metaphors employed by the software correspond to the user's internal model of the organization of a business task?
- **Data Validation**. Can the software cope with any combination of possible user interaction and input data?
- **Business Rules ad Requirements**. Does the software accomplish the tasks set out in the design specifications? Can it accomplish these tasks for any combination of user interaction and data entry that passes data validation criteria?
- **Security**. Does the system prevent access to data or functionality to users not authorized through system security infrastructure? Does the system permit access to all data or functionality for authorized users?

We assess these aspects of the system by following our system development methodology described in the table below.

Test Component	Description
User Interface System Test	Our system testing approach for online screen functionality focuses on navigational paths to follow, GUI standards, error handling for negative testing, data handling etc. Deloitte lead analysts and developers subject every change to the user screens to this testing.

Test Component	Description
Business Layer System Test	This testing focuses on positive as well as negative testing of the application. The application is tested to confirm appropriate handling of both valid as well as invalid data. Test scenarios address testing with valid as well as invalid data and Deloitte lead analysts and developers perform thorough testing of this kind.
System Integration Test	This is a final test to validate the overall functioning with a specific focus on testing the user interface, business layer, and data layer together. Deloitte lead analysts and developers are responsible for performing integration testing to deliver quality code for the next phase of testing.
Regression Test	This is a test to confirm that any changes/new functionality does not affect existing functionality. This is part of every release that Deloitte implements.

Table 4.13-5. System Testing Components.

Scripts/Scenarios

Once the work item passes their first level of testing, a System Test build occurs, bringing the new code to the System Test Environment. During the development phase, the functional team documents System Test Scripts within the SACWISMate Test Script Tracking module. These scripts document in detail exactly how each screen functions. All business rules and validations, both on the front and back ends, are documented in a step-by-step script. We use the same tool for creating these test scenarios and documenting the test scripts as that used for tracking development and defects, SACWISMate. This provides a high level of traceability between the business requirements, design, and scenarios throughout SDLC.

Conduct System Testing

Once the test scripts have been developed and entered into SACWISMate, the test strips are executed by following the step-by-step description of a functional process. Test team members are assigned to test scenarios by functional area based upon the schedule in the Test Plan. As testers run test scripts, they enter the test script results into SACWISMate. SACWISMate reporting capabilities include summary and statistical reports of the unit test progress.

The Functional Team Lead and Testing Team review the results of the scenario run for discrepancies between expected and actual results. For any discrepancies which cannot be explained, testers log a defect in SACWISMate, linking the defect to the test scenario that resulted in the defect. This link between defect and test scenario enables any member of the testing team or development team to understand the background and context for any new defect and required corrective action.

As defects are logged in SACWISMate, a core set of test team members, including the Functional Team Lead, evaluate the nature of the problem. For each defect identified, the Functional Team Lead or other test team members assign a priority level. Setting priority levels is essential to differentiating between critical problems and non-critical issues. For

example, an error preventing the saving of Intake records significantly affects a core function of solution. Problems such as these are classified with a priority of “High.” Conversely, a spelling error in an error message has little impact on work processes. This type of bug is classified as a “Low” priority.

When a test script fails, the testing team runs the test scenario again once the development team addresses the defect. Depending upon the severity and nature of the defect, the testing team may run related test scenarios for further quality assurance. If the defect passes re-test, the tester describes the defect resolution and closes the defect in SACWISMate. Throughout this testing process, the Functional Team Lead and other core team members run reports from SACWISMate for the purpose of status reports and deliverables, as well as on an as-needed basis. Ultimately, the project team uses these reports on current defect and test scenario results to evaluate the stability of the system.

The figure below is an example of a System Test script that takes a tester through targeted tests for work items.

Test Script # Clothing Inventory

Functional Module	Case Management	**Sub Module**	Case
Objective	This scenario will test the Clothing Inventory Screen	**Assigned To**	Jon Cholak
Test Script Result	Tests the Clothing Inventory Screen		

Prerequisite Test Scripts
This test script has no prerequisite test scripts.

Security
User Profile's are defined in the login step of the test scenarios.

Associated Requirements

ID	*RFP ID*	*Requirement*
SR1.9.16		The system must allow providers to document Clothing Inventory details when a child enters placement and when a child is discharged from placement.
SR1.9.19		The system must allow a provider to view a printable Clothing Inventory report.
SR1.9.18		The system must notify the assigned caseworker when a clothing inventory has been submitted by the caseworker.
SR1.9.17		The system must allow providers to document Clothing Inventory Details such as the Date of Clothing Inventory, a Clothing Inventory Summary, Clothing Type, Number of Clothing, and Comments for each type.

Test Script # Clothing Inventory

Step #	Program	Description	Expected Results
1			
		Log on to the KIDS application as a user having the position title Provider caseworker and the additional security codes.	The user is directed to the KIDS application home page.
2			
		Set case in Focus. Make sure that case has Client in it.	Case is set to Focus as a Primary entity
3			
		Verify that the Providers can only access the cases for which they were providing the services.	The Providers can only access the cases for which they were providing the services.

Deloitte. AL SACWIS Project - Test Script Detailed Report 1/11/2011 Page 10 of 316

Figure 4.13-8. Sample System Test Script from SACWISMate
The first section of the test script displays information that helps the tester trace the script back to the original design and requirements

Once requirements are linked to the script **(traceability),** the step-by-step process is laid out for the tester. A description of the action needed and the expected results aids the tester through the script. At the end of the script is a record of all test runs. Testers document reasons why they've passed or failed the script.

See below (Figure 4.13-6) for a sample of the system testing defect log.

We maintain system test defects in the SACWISMate incident management module which increases awareness and reporting on defects providing analysis over time of open versus closed defects during the project duration.

SACWIS Defect Log

Issue Log #	Date Identified	Resolution Date	Status	Description	Short Term Resolution (if applicable)	Long Term Resolution	Assigned Functional Team	Severity
1	4/23/10	5/20/10	Closed	Interface file transmittal on failure	Restarted the database server to release excess transaction locking requests.	Online redo and archival strategy is being reviewed by	Intake	High
2	7/11/10	8/18/10	Closed	Several database instances remained in read-only mode after completion of nightly batch. One of two database instances restarted at 4AM causing approximately 50% of the application instances to remain in read-only mode after 7AM.	Instances that remained in read-only mode were restarted manually.	Procedures have been established to monitor production instances and determine if they are read-only; steps are being taken to avoid the accidental restart of production	Reports	Medium

DE_SACWIS-351

Figure 4.13-9 Sample of the FACTS II Test Defect Log.
Sample System Test defect log.

4.13.2 Integration Testing

RFP reference: 8.13.2 Integration Testing, Page 53

Integration testing ensures that all facets of the application work together as a cohesive whole, particularly when one (or more) COTS component(s) is included as part of the solution. Integration testing will be conducted as each component or function is developed and added to the baseline code. Integration testing will be conducted by the Bidder's technical and functional teams and will be executed in a production-like environment.

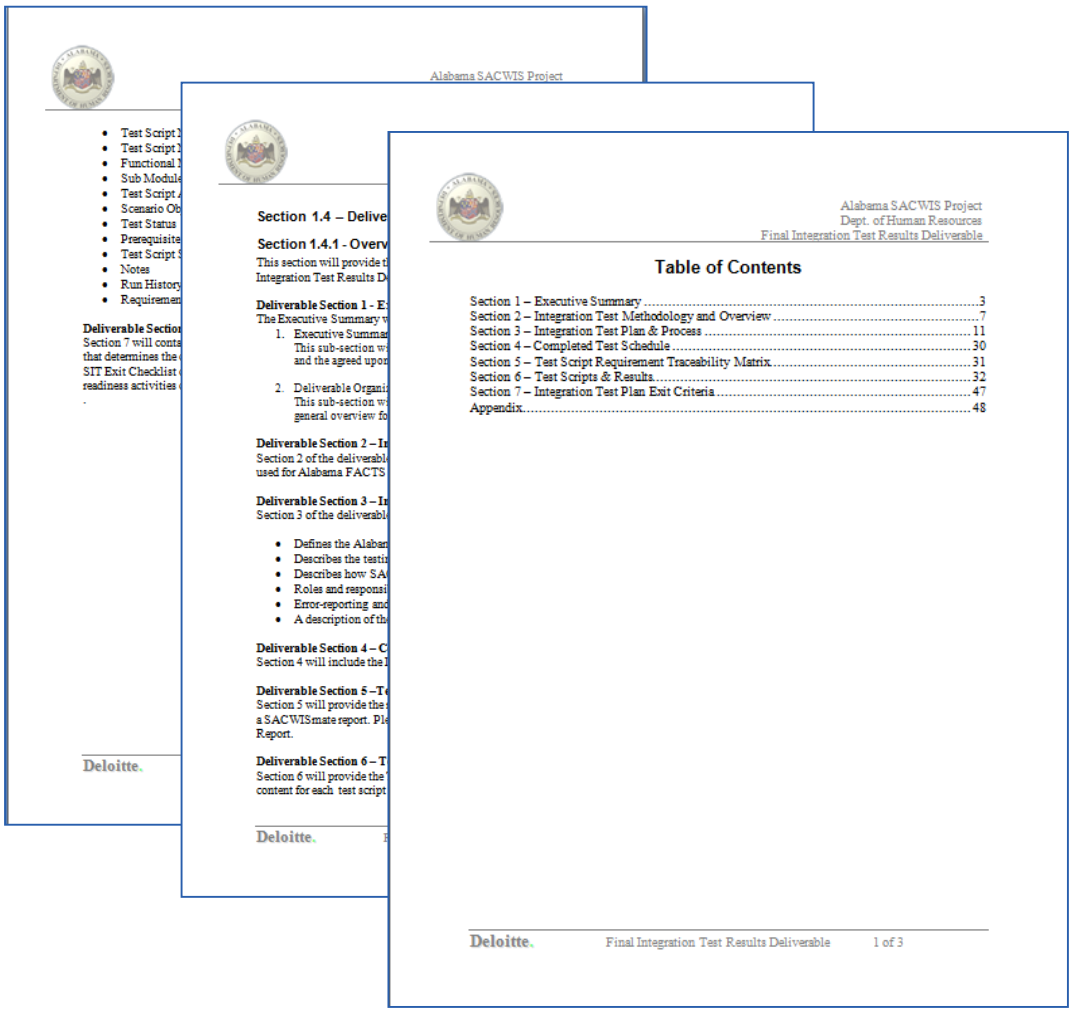
In their proposals, Bidders should identify their methodologies for conducting Integration testing, with particular emphasis on best practices approaches related to their proposed solution. Examples of previous successful methods of Integration testing, using the application and COTS components being proposed, are desirable. This testing should include the interfaces with other systems.

We have a clear understanding based on our experience with implementing SACWIS nationally, as the expert in the functionality of our transfer solution – DC FACES.NET, and the impact of how module changes impact overall functionality. Therefore, as part of the Delaware FACTS II Integration test phase, we execute a set of integration test scripts.

These scripts are identified and developed in collaboration with DSCYF and encompass the common functionality of the application across the different modules. The results help confirm the stability of the application with introduction of the new functionality being tested.

Planning and Preparation

Following the System Testing, as each component or function is developed and added to the baseline code, Integration Testing is conducted. Deloitte develops the Integration Test Plan for DSCYF staff approval and, in collaboration with DSCYF, establishes a separate testing environment for Integration Testing. A sample table of contents for the Integration Test Plan is found below.



DE_SACWIS-1518

Figure 4.13-10. Sample Table of Contents for Integration Test Plan.
The Integration Test Plan includes an overview of the Integration Test methodology, processes, test schedule, test script requirement traceability matrix and test scripts.

Methodology and Best Practices

Integration testing is performed as part of the test components incorporated into our FACTS II Playbook methodology.

Integration testing validates that the interaction between the various components work according to the specifications when they are integrated and that the various sub-system/package interfaces are interacting correctly with the system. The scope includes verification of critical functional sub-systems and sub-system interfaces and the verification of requirements related to interaction between the critical component layers. The test scenarios are mapped to the requirements determined and approved for the specific module/interface that is being tested.

We conduct manual testing during the integration testing phase. The Deloitte team practices the above discussed testing processes and approach for each of the following test components. Our testing process analyzes the functionality of software along the following dimensions:

- **Business Processes ad Requirements**. Does the software accomplish the tasks set out in the design specifications? Can it accomplish these tasks for any combination of user interaction and data entry that passes data validation criteria?
- **COTS Tools (COTS) Components**. Do components used by the application work correctly?
- **Interfaces**. Do all external facing interfaces work correctly and exchange correct data with the application?
- **Security**. Does the system prevent access to data or functionality to users not authorized through system security infrastructure? Does the system permit access to all data or functionality for authorized users?
- **Performance**. Does the system respond in a timely manner?
- **Technical**. Can the hardware infrastructure handle the expected usage and anticipated future growth?

For Integration Testing to successfully start, unit and system testing must be completed. The work items are validated per the design specs and tested to ensure each item is fully functional. Integration test scripts are documented in SACWISMate and reviewed for completeness. Test data is seeded and all test scenarios are then ready to be executed.

Scripts/Scenarios

Integration Test Scripts represent a business process or set of function(s) or a function. This business process may include one or many business functions, and in most cases, require multiple steps to achieve specific goals and expected results. For example, processing an Intake involves the following steps:

- Collecting information about the referral

- Screening the intake
- Having the intake screening decision approved by the worker's supervisor
- In the event the intake is screened in, assigning the intake to an investigation or assessment worker
- Printing a copy of the intake report

As showcased in Figure 4.13-7 below, each test script includes:

1. Unique identifier and name for each script – ID and Headline fields on Main Tab
2. Tester name(s) – Place for multiple tester names (Policy Tester, PCS Tester, Accounting Tester and Field Testers) on Main Tab
3. Start and End Dates – Target Start and Target End Date on Main Tab
4. Dependency Data that must be loaded prior to script execution – Execution Tab
5. Step Numbers and detailed instructions on what the tester must perform – Requirements/Execution Tab
6. Expected results documented in detail to provide the tester with the exact results they should view when completing each test step – Expected Results Tab
7. Actual results to document the results of each step and document any associated defect number - Attachments and Notes as well as Expected Results Tab
8. Place for the system version to be documented – Scheduled Release field on Main Tab

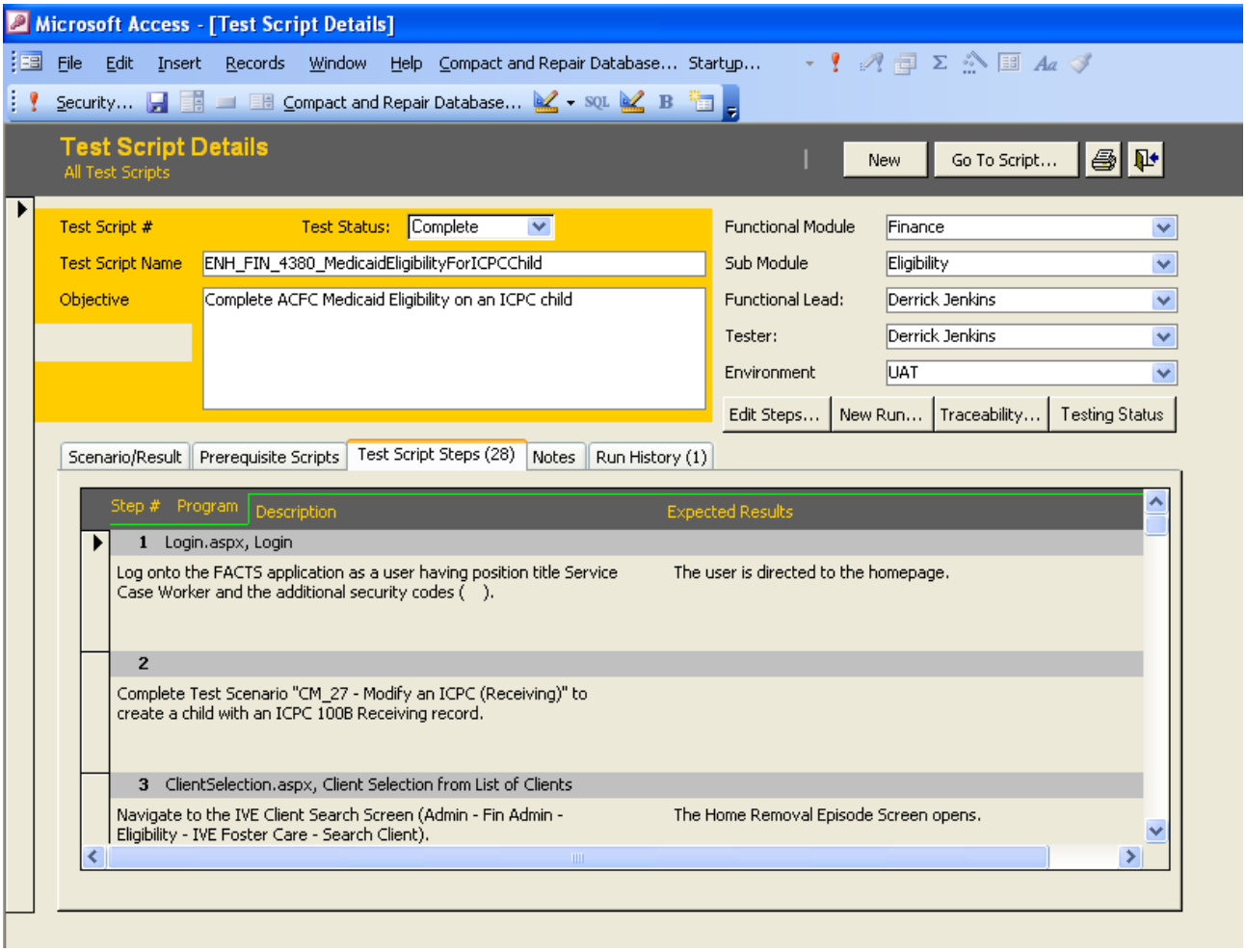


Figure 4.13-11. Sample Snapshot of a Test Scenario with Detailed Steps which form the Test Scripts Executed by our System Testers.
The Deloitte test team documents test scripts within test scenarios created in SACWISMate.

Conduct Testing

Integration Testing spans multiple programs and modules. The testing team uses a “black box” approach to design test scripts, as opposed to a “white box” approach. “Black box” Integration Testing through test scripts or scenarios is also a best practice for confirming the correct and efficient communication and interface of objects across technologies, hardware components, and software components. Interfaces between the SACWIS application and other systems are important features that are tested as part of the overall project effort. As such, interface testing follows the same methodology and approach as what is described here.

To create test scripts spanning multiple objects, more than one module, and eventually multiple systems through interfaces, the testing team continues to use the validated requirements specifications as source material. Workflow rules, business rules and non-functional requirements provide the basis for many test scripts. In addition, our team

continues to work alongside the Delaware FACTS II project team members to help confirm that certain types of situations or transactions of interest or concern are included in the test scripts.

A collaborative approach to designing and then executing test scripts is the best way for the DSCYF staff to become application experts. Continual knowledge transfer throughout the development life cycle lessens the need for extensive formal transition time at the end of the project. This approach of hands-on involvement also is more efficient and effective than an extended transition of classroom-style training.

Once the test scripts have been developed and entered into SACWISMate, the identified testers execute the test scripts by following the step-by-step description of a functional process. Test team members are assigned to test scenarios by functional area based upon the schedule in the Test Plan. As testers run test scripts, they enter the test script results into SACWISMate. For each step the tester needs to document whether the step passed or failed.

Testers review the results of the scenario run for discrepancies between expected and actual results. For any discrepancies that cannot be explained, testers log a defect in SACWISMate, linking the defect to the test scenario that resulted in the defect. This link between defect and test scenario enables any member of the testing team or development team to understand the background and context for any new defect and required corrective action.

Below is an example of the Incident Tracking module where testers log all defects found while running scenarios:

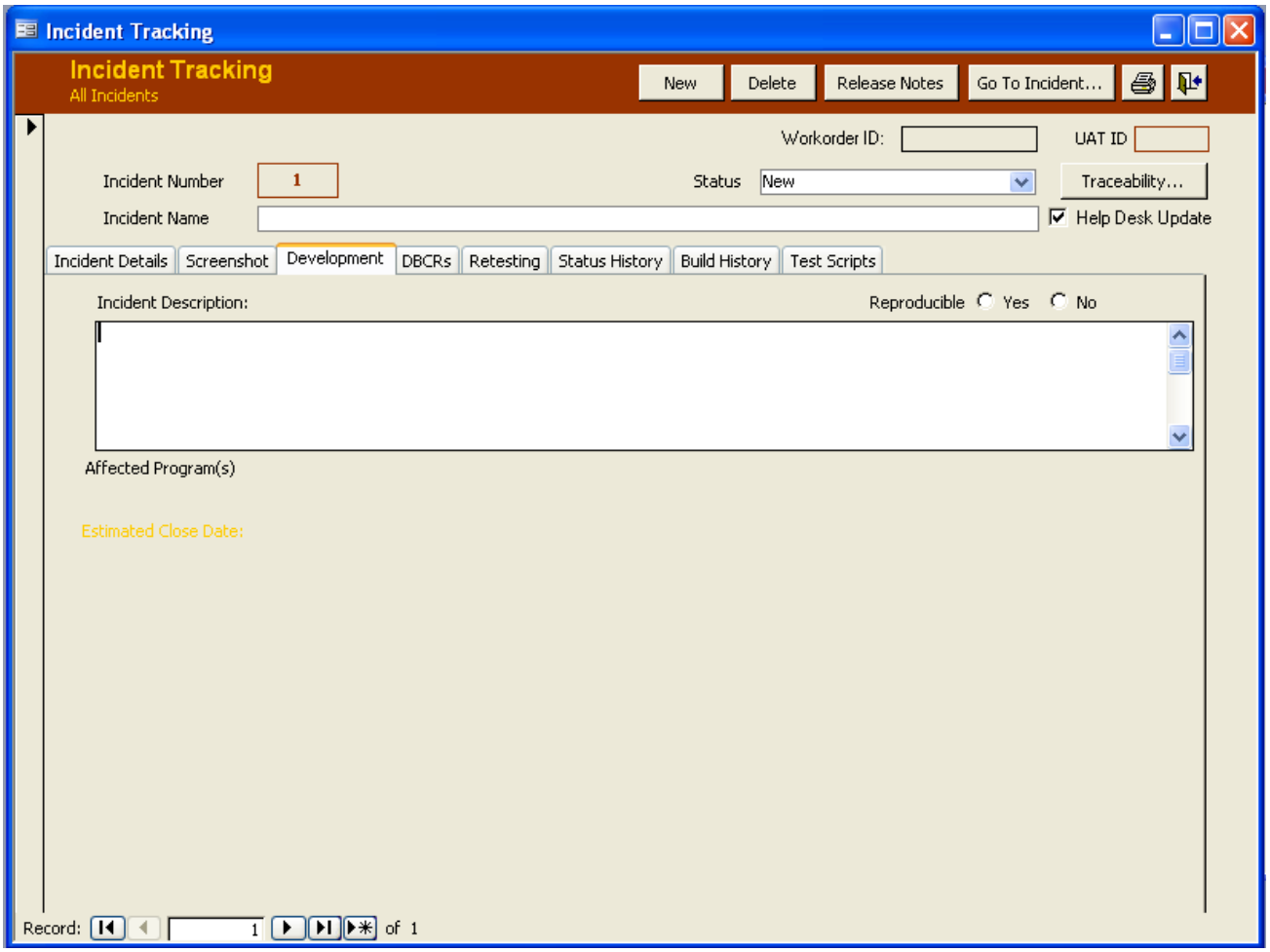


Figure 4.13-12. Sample Snapshot of a Work Request created for tracking an identified defect.
The Deloitte team recommends the use of SACWISMate defect management and prioritization.

For each defect identified, a priority level is assigned. For example, an error preventing the save of Intake records significantly affects a core function of solution. Problems such as these are classified with a priority of “High.” Conversely, a spelling error in an error message has little impact on work processes. This type of bug is classified as a “Low” priority. We recommend a three-tier priority hierarchy for defects found in the system such as high, medium, and low priority.

The development team assesses the problem and triages all defects in order of priority and the Functional Team Lead and development team assign bugs to a developer based on developer workload and expertise in the specific business or technical nature of the problem. If any defect results in significant change to the data model, business rule specifications, or any other deliverable, the detailed design deliverables are updated to reflect the change. For example, in the course of testing, it might be determined that a client can serve as his or her own provider. In past projects, this situation has arisen when reviewing participants in the independent living program.

The testing team runs the test scenario again once the development team addresses the defect. Additionally, depending upon the severity and nature of the defect, the testing team may run related test scenarios for further quality assurance. If the defect passes re-test, the tester describes the defect resolution and closes the defect in SACWISMate. The project team uses these reports on current defect and test scenario results to evaluate the stability of the system for the next project phase.

Reports

When the Integration testing is complete, the Deloitte prepares the Integration Test Results report and the team leads project team members in a detailed review of the format and contents of the documents. The periodic and final reviews of this test report familiarize DSCYF personnel with the most significant errors encountered. This collaborative approach from the beginning of the testing phase helps to avoid any “surprises”, thereby expediting the coding and testing process. A sample table of contents for the Integration Test Results Report is found below.

Previous Successful Integration Testing

All of our previous SACWIS solutions have undergone our integration testing methods and have been proven to be stable, successful solutions. For example, our Alabama SACWIS FACTS integration testing included the testing of all functional modules, converted data, interfaces, and reports. Testing all functionality included the testing of COTS tools like the integrated search tool, workflow, and document management features.

Using the SACWISMate tool, Deloitte effectively managed the traceability between requirements to the end testing phases. Our test scripts effectively covered each one of our business rules and requirements. Once the code was promoted to start integration testing, work item defects were identified, categorized, and resolved prior to statewide go-live. By testing all integrated functionality gave the state the confidence that the application and all critical integration components were working as designed to meet their requirements.

We have used this integration test methodology on all of our SACWIS engagements nationally. We are proud of our track record and proven ability to “Go-Live” in production was planned with full functionality.

Performance Testing

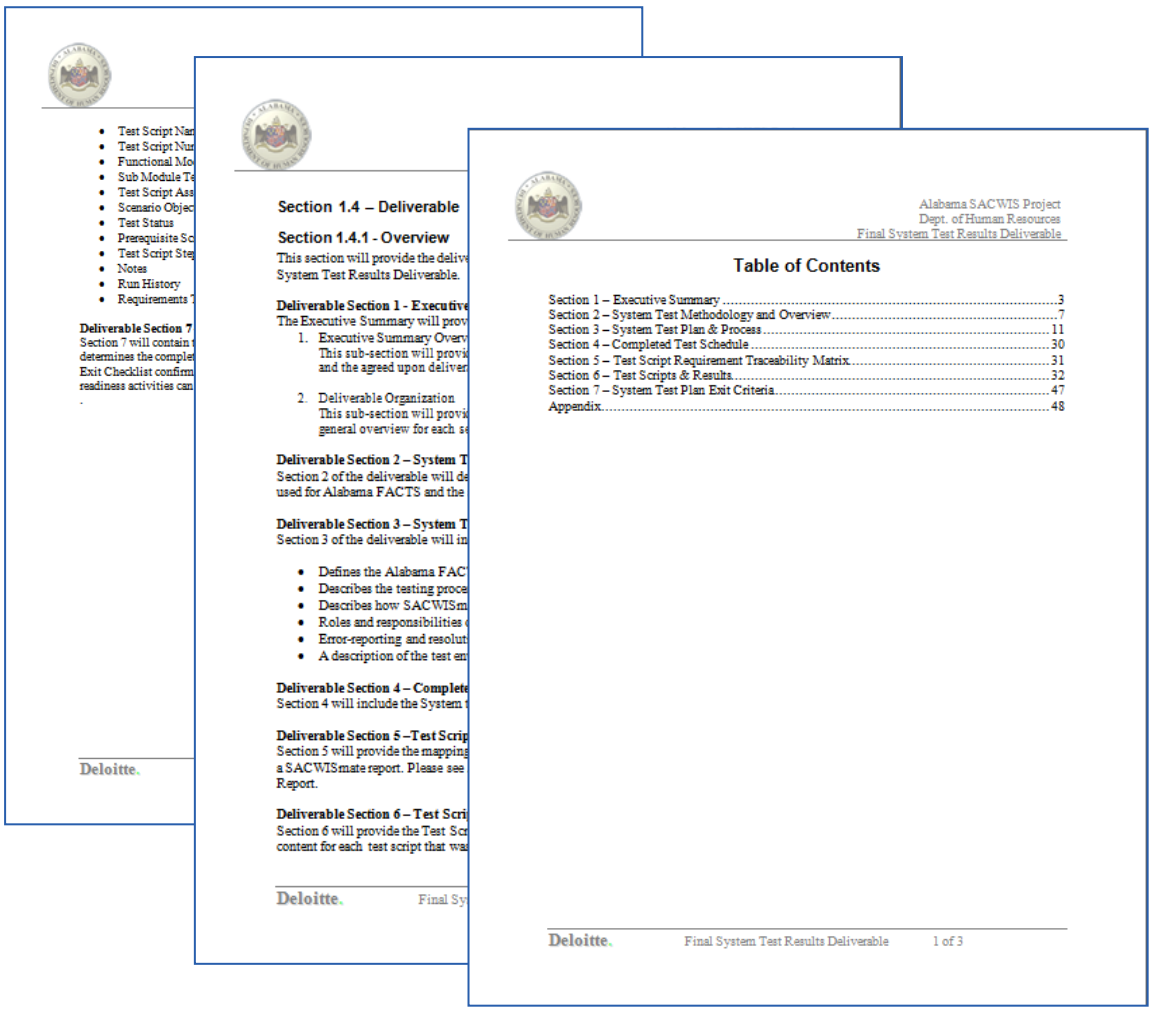
Performance Testing occurs parallel with the User Acceptance Testing phase. Performance Testing evaluates whether or not the system can perform at the levels specified in the requirements for the possible user activity in the system and data. Deloitte proposes to use Microsoft Visual Studio Ultimate, an industry leading performance testing tool that we have used on our transfer solution project, DC FACES.NET. In order to execute a performance test, we configure a test scenario according to defined test

objectives. Microsoft Visual Studio Ultimate provides the framework required to orchestrate scripts, monitor system errors/events and govern the time frame within which the system is to be tested. We recommend starting this testing during the user acceptance testing (UAT) environment, since this application code at this stage most closely mimics the targeted production code. The Performance Test Results provide a summary of the Performance Testing activities, including the results of executed test scripts and any revisions to the testing documentation or application as result of the testing findings. Web Performance test scripts and results are generated by the Visual Studio 2010 Ultimate which are then documented in SACWISMate. The status of each web performance test is discussed and distributed along with all the testing phase results.

Disaster Recovery Testing

Disaster Recovery Testing occurs parallel with the Integration Testing phase. A business continuity management process is defined to minimize the impact on integrated case management system in the event of a system outage. It defines the process to recover from the loss of information assets through a combination of preventive and recovery controls. Consideration is given to the consequences of disasters, security failures, loss of service, and service availability.

By conducting Performance testing and Disaster Recovery testing, Deloitte prepares DSCYF to gauge FACTS II system availability, reliability and continue to serve DSCYF customers without any interruptions due to any unforeseen reasons.



DE_SACWIS-1519

Figure 4.13-13. Sample Table of Contents for the Integration Test Results Report

The Integration Test Results Report includes an overview of integration test methodology, processes, completed test schedule, test script requirement traceability matrix, test scripts and results, and integration test plan exit criteria.

4.13.3 Regression Testing

RFP reference: 8.13.3 Regression Testing, Page 53

Regression testing must be conducted by the Bidder each time a significant component is added to the application or a defect is corrected to ensure that the addition or correction does not break some component of the application that worked previously. Bidders must describe the proposed approach to regression testing for all phases of the project, including the post-implementation maintenance/enhancement phases.

Regression testing (i.e. re-runs of previous tests) verifies when code changes have not adversely impacted existing functionality. It also confirms that no additional defects are introduced when repairing defects or adding new functionality to 'approved' code. The decision to perform regression testing is based on the risk introduced by the change, the size or impact of the change, or the criticality of the business functions impacted.

did you KNOW?

Deloitte achieved "near zero" critical defects during the end of testing phases at the Go-Live phase at the Alabama FACTS project.

Planning and Preparation

Deloitte develops the Regression Test Plan for DSCYF staff approval and, in collaboration with DSCYF, establishes a separate testing environment for Regression Testing.

Our Approach to Regression Testing by Phase

Adding in regression testing to each phase of Testing confirms a fully functional application prior to go-live. Deloitte works with DSCYF to identify business critical functionality and test scripts related to achieving the required end-to-end business functionality to create the basis for regression test scenario repository. Performing the regression testing is an integral part of Deloitte's testing approach. As new enhancements are added or significant code changes are performed, the regression test scripts repository grows to cover the additional functionality for regression testing.

Regression Scripts/Scenarios

Test scripts are reviewed and shared between the Deloitte and DSCYF functional teams, in coordination with SMEs, to ensure compliance with policy and business processes. The same scripts and scenarios are executed in all environments, confirming consistent regression testing.

Regression testing for new scenarios does not need to be documented in SACWISMate. Instead, scenarios are indentified that confirm the most critical components of the system are functioning as expected. The Functional team decides which sub-set of scenarios is used and executes those scripts as a group to test.

Documentation of test runs mirror the process used for system and integration test documentation. Using the testing module of SACWISMate, testers note all results of test runs. Figure 4.13-9 below is an example of how system test runs are tracked and documented including a description of the issue, expected and actual results, and if the test passed or failed.

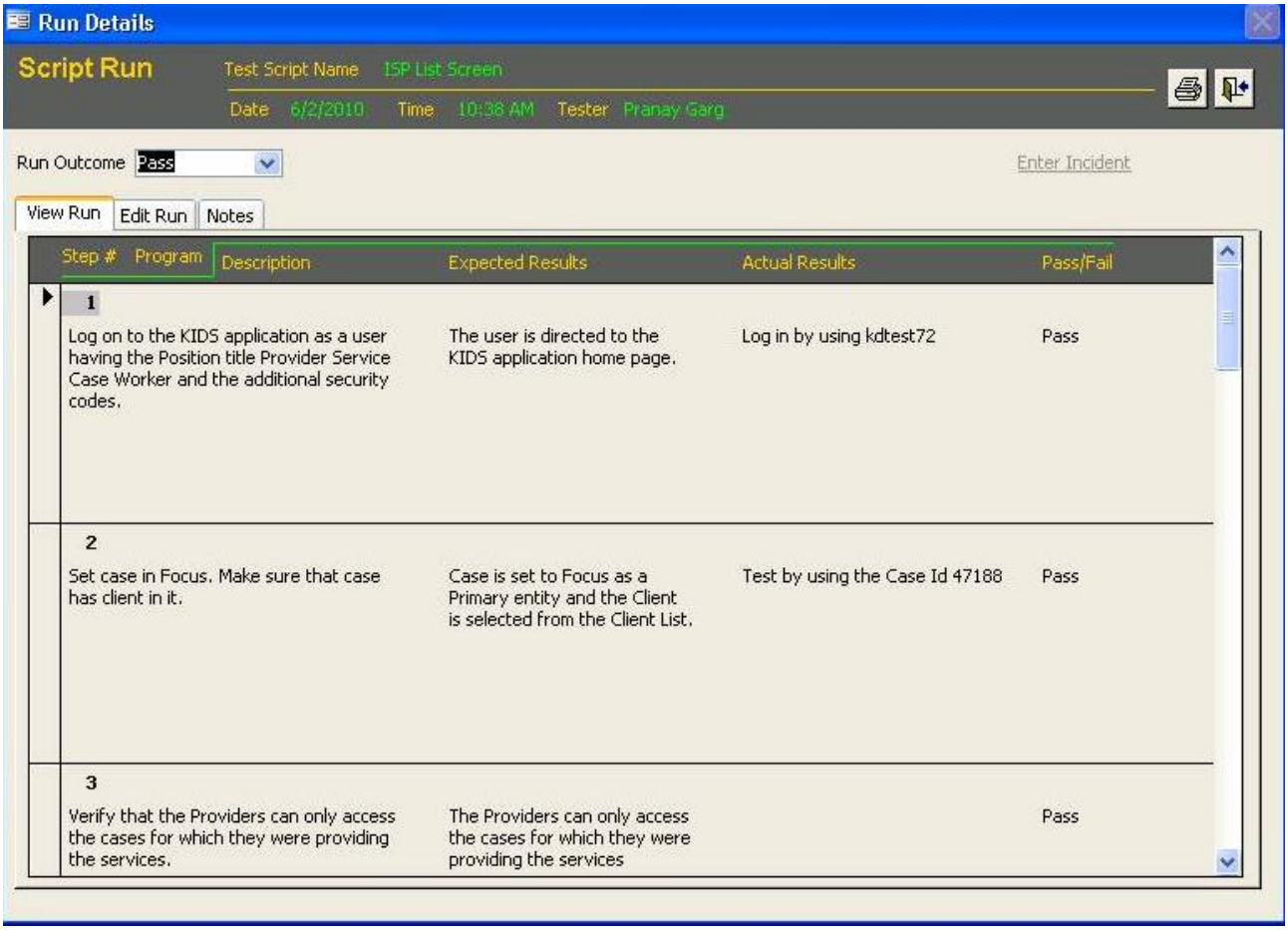


Figure 4.13-14. Sample Snapshot of a Test Run in SACWISMate.
The Deloitte team recommends the use of SACWISMate to document all test runs and results.

Testers review the results of the scenario run for discrepancies between expected and actual results. For any discrepancies that cannot be explained, testers log a defect in SACWISMate, linking the defect to the test scenario that resulted in the defect. (Additional information about SACWISMate is found in *Section 4*). This link between defect and the test scenario enables any member of the testing team or development team to understand the background and context for any new defect and required corrective action.

As defects are logged in SACWISMate, a core set of test team members, including the Functional Team Lead, evaluates the nature of the problem. For each defect identified, the Functional Team Lead or other test team members assigns a priority level. Setting priority levels is essential to differentiating between critical problems and non-critical issues. For example, an error preventing the save function for intake records significantly affects a core function of a solution and are therefore classified with a priority of “High.” Conversely, a spelling error in an error message has little impact on work processes. This type of bug is classified as a “Low” priority.

Regression Testing during Enhancement

Once an enhancement is approved through the Change Control process it is created as a work item and follows the normal development life cycle. Although regression testing occurs within the system and integration test phases, for enhancements the testing team must carefully consider all functionality impacted by this new piece of code. Small enhancements within the application can have adverse effects on already established processes. Ensuring that the code functions properly with the new enhancement is imperative prior to releasing in a maintenance build.

Regression Testing during Post Implementation/Maintenance

Just as defects can be identified during system test at each phase prior to go-live, end users may also run into defects during every day use of the application. Once these work items are prioritized and corrected they are fully tested prior to their release in a maintenance build. The work items undergo the same unit, system, and integration test process, but it is especially imperative at the post implementation/maintenance level that regression testing be taken into careful consideration. Simple defect fixes can have a detrimental downstream effect if things are not tested properly.

Reports

The incidents created through the Regression testing are documented in the SACWISMate incident management module. The SACWIS reporting module provides DSCYF staff the capability to generate the incidents report for review and tracking.

4.13.4 User Acceptance Testing

RFP reference: 8.13.4 User Acceptance Testing, Page 53

User Acceptance testing will be conducted by the Department, with support from the Bidder's team. The purpose of User Acceptance testing is to ensure that the application is working according to the approved Detailed Design. Although the testing will be completed by Department staff, the Bidder is expected to develop a User Acceptance plan, to draft test scripts, to assist staff in the preparation of the User Acceptance test environment, to provide training on testing tools or processes for the User Acceptance test team, and to provide ongoing support during the User Acceptance test phase, both from a functional as well as a technical perspective.

In their proposals, Bidders should describe their approaches to User Acceptance testing, and may propose alternative methods for the Department's verification and validation of the system.

We provide fulltime support to DSCYF throughout the preparation, support, and defect resolution phases of User Acceptance Testing (UAT). Our history of similar successful implementations provides DSCYF with a proven approach to UAT. We bring the tools, techniques, and approaches to support end-to-end traceability to confirm your requirements are met and automated testing tools to provide confidence as releases are sent to you for testing.

We provide full-time support to the DSCYF throughout the preparation, support, and defect resolution phases of User Acceptance Testing (UAT). Our history of similar successful implementations provides a proven approach to UAT. We bring the tools, techniques and

approaches to support end to end traceability to confirm your requirements are met and automated testing tools to provide confidence as releases are sent to you for testing.

DSCYF requires a dedicated period of time to validate that Delaware FACTS II is meeting your business processes. You require a stage to experience the system hands-on and provide feedback prior to the rollout of Delaware FACTS II to end users. Also during this time, DSCYF requires up-to- date system documentation and development of training materials to support testers and users. To provide the best possible Delaware FACTS II solution, DSCYF requires a proven plan and methodology to execute UAT. From our prior projects, Deloitte brings DSCYF a baseline set of test plans, scenarios, scripts, and tools. The tests used and tools ultimately become yours, enabling DSCYF to efficiently and effectively support Delaware FACTS II post-implementation.

distinguishing FACTORS

Deloitte brings the following attributes to the UAT task:

- Established knowledge of social services projects to support DSCYF staff through issue resolution
- Existing repository of test cases based on systems similar to FACTS II

Deloitte understands the need to integrate with and support DSCYF throughout the UAT process. Our team has a successful track record with large-scale system implementations, and we bring our best practices and lessons learned from previous experiences to DSCYF.

Planning and Preparation

Deloitte develops the User Acceptance Test Plan for DSCYF staff approval and, in collaboration with DSCYF, establishes a separate testing environment for User Acceptance Testing. A sample table of contents for the User Acceptance Test Plan is found below:

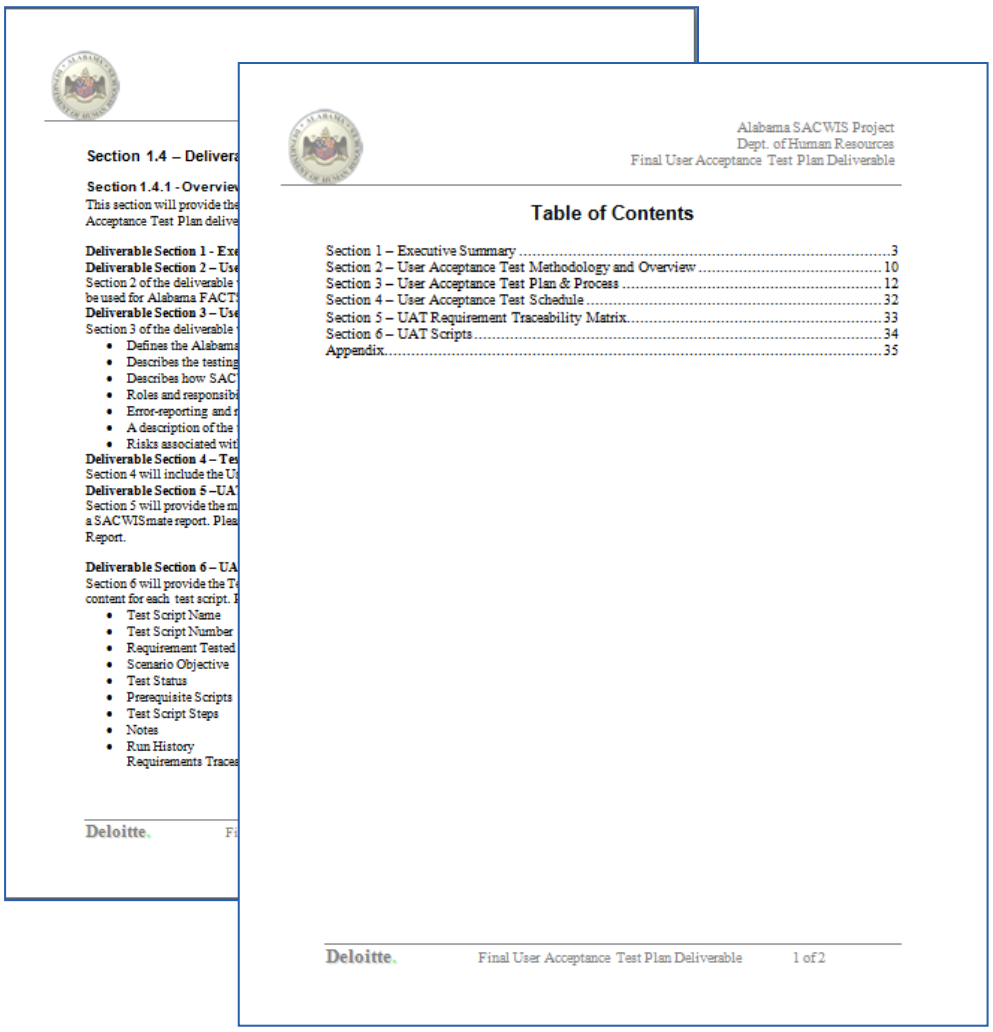

Alabama SACWIS Project
Dept. of Human Resources
Final User Acceptance Test Plan Deliverable

Table of Contents

Section 1 – Executive Summary 3
Section 2 – User Acceptance Test Methodology and Overview 10
Section 3 – User Acceptance Test Plan & Process 12
Section 4 – User Acceptance Test Schedule 32
Section 5 – UAT Requirement Traceability Matrix 33
Section 6 – UAT Scripts 34
Appendix 35

Deloitte. Final User Acceptance Test Plan Deliverable 1 of 2

DE_SACWIS-1521

Figure 4.13-15. Sample Table of Contents for the User Acceptance Test Plan.
The User Acceptance Test Plan includes UAT methodology, processes, test schedule, requirements traceability matrix, and UAT test scripts.

Our Approach to User Acceptance Testing

The objective of UAT is to ensure that the system is working according to the approved design. The following methods provide the tools for DSCYF to verify and validate the system.

Scripts/Scenarios

Per DSCYF's requirement, Deloitte develops the UAT scenarios to be executed by UAT testers. UAT scripts are created using portions of the end to end integration scenarios used during the System and Integration test phases. These scenarios are targeted at the specific tester's functions to validate that the business requirements were fulfilled.

We leverage our test script creation process to establish consistency in scripts and create test scenarios. Our test script template captures details of the test business scenario, tester details, execution times, test script with step-by-step instructions for executing the test and corresponding test results. The test scripts are assigned to tester(s) through the SACWISMate and after execution the tester can pass/fail each test script and document actual results. We also create proper test data prior to test execution and these are the test cases which are also documented in the test scenario in SACWISMate. The test data requirement and testability of any requirement is identified by us early during the test planning stage and this ensures a smoother test execution.

Data seeding

Data created by automated test tools and data entered into the system by UAT testers provide functional level test data that combine to permit the verification that the application meets the system functional requirements. The type and amount of data along with the corresponding data parameters needed for the UAT environment are discussed with DSCYF to confirm that the most appropriate data is available. The UAT environment setup should mirror the Production environment. We use our experience gained from previous SACWIS implementations to build the environment so that it is similar to production. The data and setup should be of a quality that allows for meticulous testing of the multiple components of Delaware FACTS II to verify that the system is ready for implementation.

We understand your need for a rollback of the database to a previous checkpoint. This is sometimes due to the quality of the data after repeated testing in the UAT environment or to facilitate more accurate functionality, environment, or data elements. We work with the DSCYF to determine and create periodic checkpoints that allow for the UAT database to be restored to a particular point in time.

Entry Criteria

Before UAT can begin, the test environment must be configured and prepared. Among the activities that we are responsible for, the main items include:

- Establish (create and initialize) and test UAT databases.
- Establish UAT application and web servers.
- Confirm that the correct executable software, as it is promoted from the (accepted) tested Quality Assurance environment, is installed in the UAT environments.
- Monitor that the system remains stable and refresh data as greater levels of refinement and stability are achieved.
- Establish test data which includes data created via the conversion process and confirms conversion sources are available for subsequent use if required.
- Verify interfaces, data set-up, tables and user acceptance materials are set up and ready for the DSCYF before testing begins.

Once the UAT environments are established, we maintain environmental stability by running regression tests as changes are promoted into the UAT environments. Additionally, from an operational perspective, we support the User Acceptance Testing environment in the following capacity:

- Monitoring system performance
- Monitoring computer resource usage
- Creating and running batch schedules
- Applying conversion data to obtain a true-to-life system response - this gives the UAT authenticated and measurable reactions to the scenario tasks
- Investigating problems and identifying potential problems
- Providing appropriate user access to the system in the UAT environment

It is important that the functionality that is promoted to the UAT environment goes through various stages of testing (Unit, Integration, System, and Regression testing) to validate that Delaware FACTS II is at a high quality and ready for user acceptance testing. Our robust testing process uses our experience from previous SACWIS implementations to make sure that functionality is rigorously reviewed and that any major issues are identified prior to it reaching the UAT environment.

Conduct User Acceptance Testing

Deloitte supports the testing effort by preparing for and conducting UAT sessions and provides guidance to the DSCYF testing team as needed during the testing effort. We provide training on how to use the SACWISMate tool to run reports for incidents ready for testing, create incidents and track and resolve the incidents. Although we are not directly responsible for executing user acceptance tests, we serve in a management, advisory and technical support role by answering questions about the Delaware FACTS II system and when necessary, helping users execute tests and review results. At the beginning of each UAT session, we give a brief overview of the testing scope of the session and review the Delaware FACTS II application components to be tested. Throughout the UAT process, we provide ongoing support by maintaining and monitoring testing environments and our production support team resolves defects in accordance with the timeliness standards agreed upon by Deloitte and DSCYF.

As testers run test scripts, they enter the test script results into SACWISMate. As you can see, the tool lays out each step of the process. For each step the tester documents whether the step passed or failed (see Figure 4.13-10 below).

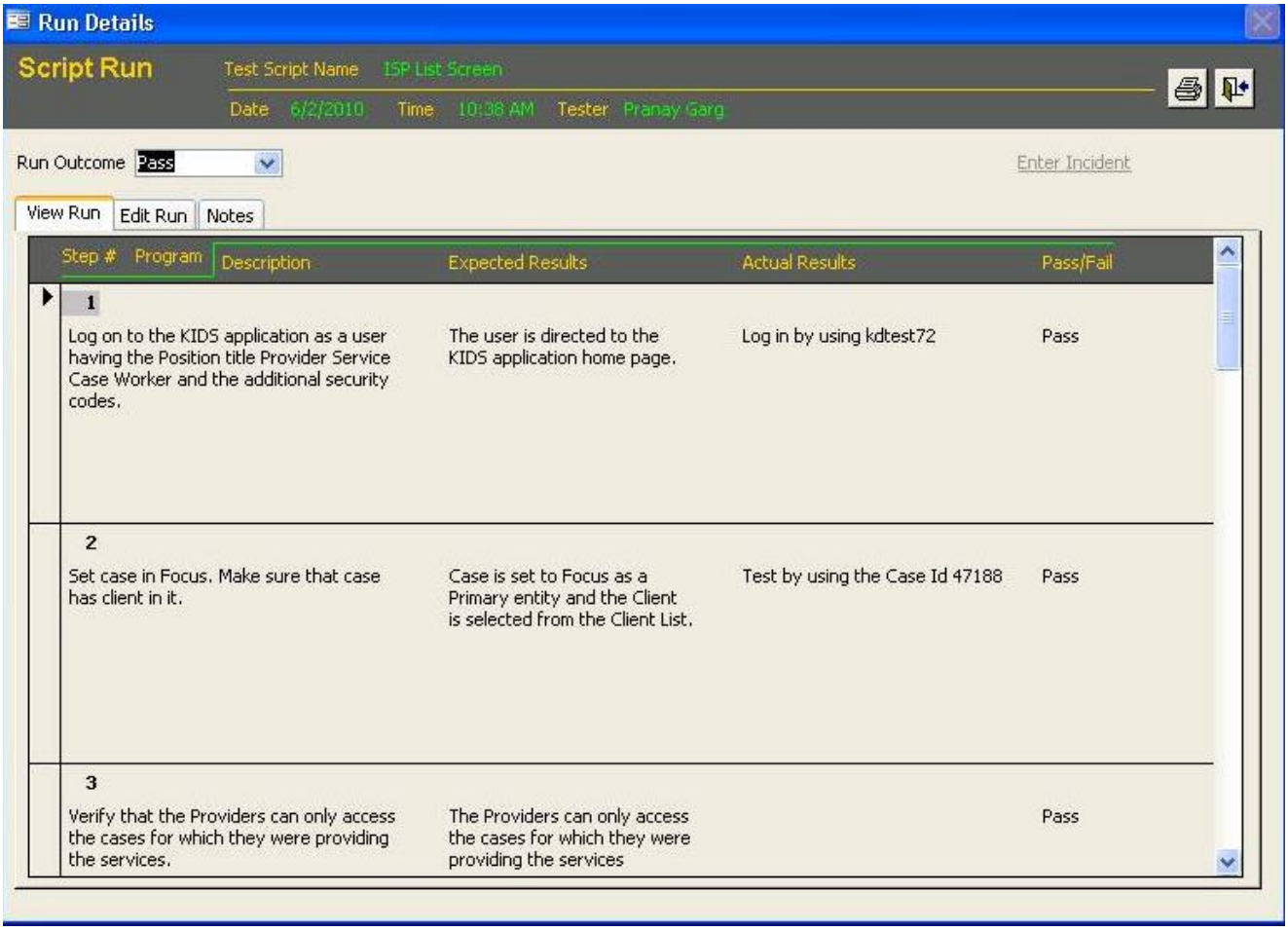


Figure 4.13-16. Sample Snapshot of a Test Run in SACWISMate
The Deloitte team recommends the use of SACWISMate to document all test runs and results.

We understand that if defects or barriers are found during UAT testing that prevent a test scenario from being completed, once we have resolved the issue, the scenario starts from the beginning or can proceed from the point where it was left off. If required, in addition to the defect resolution process, we also monitor the test environment so that it remains stable throughout the duration of UAT, in accordance with the test plan.

Defect Resolution

Adequate tracking of defects is necessary to correct the current defects and to identify trends in development and testing so that the process is improved. We work with you to establish a formal change control process to manage the defect resolution process.

Defects found during testing are captured through SACWISMate. This tool provides us the ability to log defects, track their progress to completion, and create reports detailing the overall status of the testing effort.

With a project of this size and complexity, it is necessary to have a close and ongoing collaborative approach to address situations where valid Delaware FACTS II defects are

identified. Daily meetings are held with the application development team, the DSCYF test team lead, and key people from legacy systems to discuss, route, and prioritize incoming defects. Delaware FACTS II defects that are application code related are researched and combined with the other changes in the same components for efficiency. These components are retested and returned to the DSCYF testing team. If the correction requires extensive work to resolve, we communicate this to the DSCYF project director. UAT documentation, training, and design documentation are updated (as needed) to reflect the changes. Deloitte provides periodic statistics and status of defects to keep project management aware of the trends and issues. If issues found during UAT are out of scope agreed upon by Deloitte and DSCYF, we go through the change request process to determine the most appropriate way to address the requested change.

Reports

The Deloitte team develops the Weekly UAT Test Results Report to track the status of UAT testing, including test scenario progress and any identified issues related to a scenario. This serves as a mechanism for communication between the Deloitte team, the DSCYF, and other necessary stakeholders to verify that the involved parties understand the status and evolution of the UAT phase. A sample table of contents for the User Acceptance Test Results Report is found below:

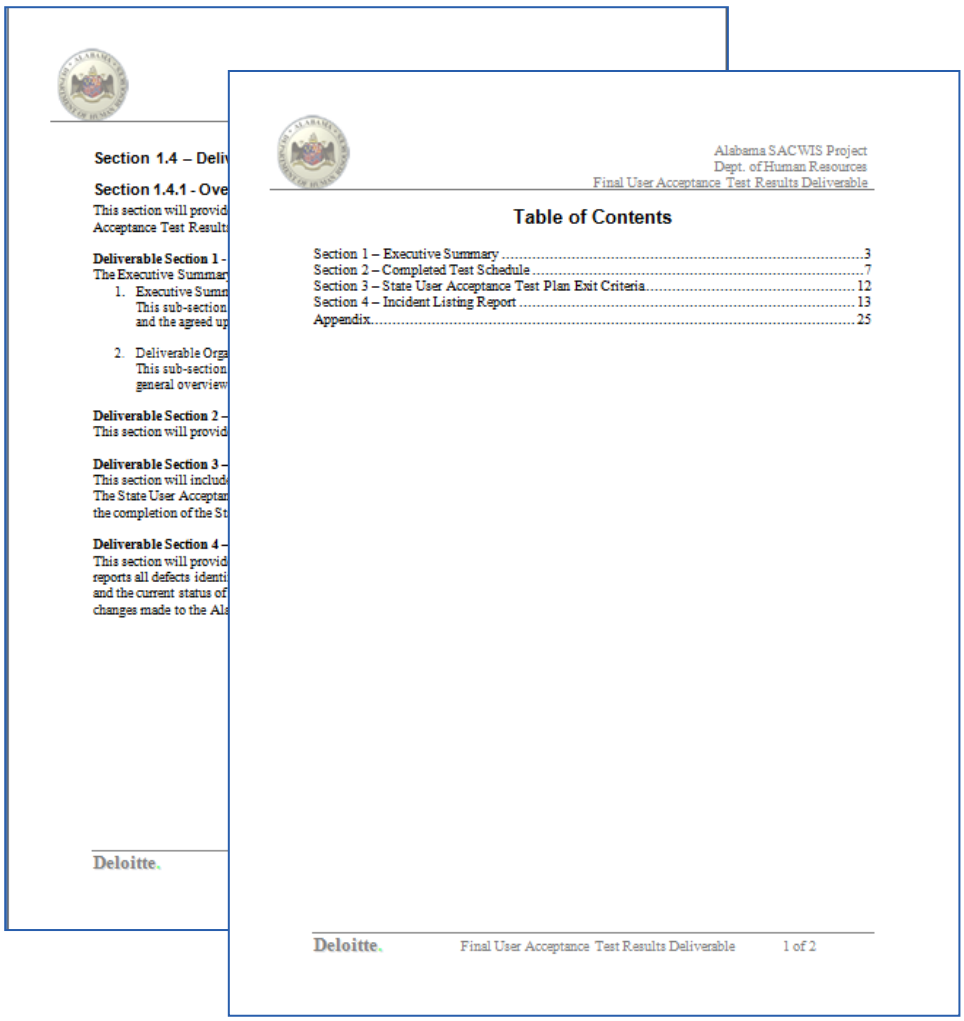

Alabama SACWIS Project
Dept. of Human Resources
Final User Acceptance Test Results Deliverable

Table of Contents

Section 1 – Executive Summary 3
Section 2 – Completed Test Schedule 7
Section 3 – State User Acceptance Test Plan Exit Criteria 12
Section 4 – Incident Listing Report 13
Appendix 25

Deloitte. Final User Acceptance Test Results Deliverable 1 of 2

DE_SACWIS-1520

Figure 4.13-17. Sample Table of Contents for User Acceptance Test Results Report
The User Acceptance Test Results Report includes the completed Test Schedule, State User Acceptance Test Plan Exit Criteria, and an Incident Listing Report.

Deloitte Assists DSCYF During User Acceptance Test

We recognize the need for our team to support the UAT team with various aspects of preparation and execution of testing scenarios, including generation of necessary transactions, data, and files. We collaborate with the UAT team as they complete the steps of each test to provide them with this information and confirm that the system functionality is performing as expected.

If for some reason an unfavorable testing result occurs, it is an essential task that the issue is analyzed and corrected in a timely manner to minimize impact to testing progress. The Deloitte team excels at this process because of our in depth knowledge from user acceptance testing for other SACWIS implementations as well as our understanding of the DSCYF of Delaware's policies and business procedures. Our functional team analyzes the testing results and tracks potential issues in SACWISmate. If an issues is identified that

needs to be corrected, it follows our development process. The main objective of UAT is to validate that FACTS II is ready for Statewide Go-Live, so it is crucial that potential issues are identified and corrected in order to maintain the quality of the system.

UAT Training

To facilitate proper testing during the UAT process, it is also essential that the DSCYF's UAT participants receive an overview of the technical and functional aspects of FACTS II along with the tools that are used during UAT to document and monitor test scenarios at the commencement of the UAT phase. The Deloitte team works with the DSCYF to provide training, documentation, and assistance with these components to the UAT testers. This confirms that the parties involved are confident with the direction and process of user acceptance testing.

Ongoing Support for UAT

We agree with DSCYF's requirement to have members of our team support the end users in both a functional and technical perspective. Based on prior experience, we have found locating team members in the location of the end users to be effective for the communication between UAT testers and our Deloitte team because of the ability for us to expedite responses to questions or requests. This close proximity also allows both the DSCYF and our team to monitor the UAT process and verify that it is progressing as planned.

Deloitte and DSCYF Responsibilities

Table 4-13-6 below describes our understanding of Deloitte's and DSCYF's roles and responsibilities in executing each phase of System Testing. The associated deliverable is also included. See *Section 1.500 Acceptance* for further details regarding our understanding of the review and approval process for project deliverables.

Deloitte Responsibilities	DYSCF Responsibilities	FACTS II Deliverables
	Unit/System Testing	
• Develop Unit Test and System Test Plans • Submit Unit Test and System Test Plans to DSCYF project staff for approval • Conduct transfer of knowledge sessions • Identify system test tools to be used and reports • Provide DSCYF with access to test results as each test is performed. • Provide Unit/System Test results to DSCYF	• Review and approve the Unit and System Test Plans • Attend deliverable walkthroughs • Provide input and clarifications to the Deloitte team regarding tools and reports	• Unit Test Plan • System Test Plan

Deloitte Responsibilities	DYSCF Responsibilities	FACTS II Deliverables
• At the States discretion, provide code for submission to a third party for code efficiency and security vulnerability testing		
• Develop the Unit Test Results Report • Submit Unit Test Results Report to DSCYF	• Review and approve the Unit Test Results Report • Approve the System Test Results Report	• Unit Test Results Report • System Test Results Report
	Integration Testing	
• Develop Integration Test Plan • Submit Integration Test Plan to DSCYF project staff for approval • Provide DSCYF with access to test results as each test is performed. • Provide Integration Test results to DSCYF • At the States discretion, provide code for submission to a third party for code efficiency and security vulnerability testing	Review and approve the Integration Test Plan	• Integration Test Plan
• Develop Integration Test Results Report • Submit Integration Test Results Report to DSCYF for approval	Review and approve the Integration Test Plan Results Report Approve the Integration Test Plan Results Report	• Integration Test Results Report
	Regression Testing	
• Develop Regression Test Plan • Submit Regression Test Plan to DSCYF project staff for approval • Provide DSCYF with access to test results as each test is performed • At the States discretion, provide code for submission to a third party for code efficiency and security vulnerability testing	Review and approve the Regression Test Plan	• Regression Test Plan
• Develop Regression Test Report • Submit Regression Test Report to DSCYF	Review and approve Regression Test Report	• Regression Test Report

Deloitte Responsibilities	DYSCF Responsibilities	FACTS II Deliverables
	User Acceptance Testing	
• Develop User Acceptance Test Plan • Submit User Acceptance Test Plan to DSCYF project staff for approval • Draft Test Scripts • Assist DSCYF staff in the preparation of the User Acceptance test environment • Provide training on User Acceptance testing tools and processes for the DSCYF User Acceptance Test Team • Provide ongoing functional and technical support during User Acceptance Testing phase • At the States discretion, provide code for submission to a third party for code efficiency and security vulnerability testing	Conduct User Acceptance Testing with support from Deloitte teams	• User Acceptance Test Plan
• Develop User Acceptance Test Results Report • Submit Regression Test Report to DSCYF		• User Acceptance Test Results Report
• Training/Mentoring for the DSCYF's technical staff on the environments and required rebuilds as new builds are released	• Receive training from Deloitte in the environments and required rebuilds • Act as students to assess the adequacy, accuracy and effectiveness of training materials	
• Appropriately configuring test environments to adequately emulate real world system use, including use of system from mobile devices • Configuring environments to interface/integrate with DSCYF legacy systems and data	• Review test environments • Configure legacy system environments and establish connectivity	
• Preparing and providing sample sets of program-specific structured test data, including converted data for use with test scripts	• Review test data	

Deloitte Responsibilities	DYSCF Responsibilities	FACTS II Deliverables
• Preparing and providing system test scripts and results and recommendations for use by UAT team	• Review test scripts	
• Preparing and providing appropriate versions of system documentation and training materials for processing and evaluation by UAT team	• Act as students to assess the adequacy, accuracy and effectiveness of training materials	
• Supporting the operation of the test system and delivery of system output to the DSCYF's UAT test team	• Support Deloitte in test system configurations • Provide legacy system support to support completion of testing	
• Provide adequate technical and other staff dedicated to testing support and problem resolution while the test is in progress	• Review the set of staff dedicated to testing support • Provide legacy system support to answer functional questions and resolve legacy issues	
• Develop test cases and scenarios for DSCYF's execution	• Form a User Acceptance Test team that supports developing test cases and scenarios	
• Training/Mentoring for the DSCYF's technical staff on the environments and required rebuilds as new builds are released	• Receive training from Deloitte in the environments and required rebuilds • Act as students to assess the adequacy, accuracy and effectiveness of training materials	

Table 4.13-6. Delaware FACTS II System Testing Roles and Responsibilities.

4.13.5 Associated Deliverables

RFP reference: 8.13.6 Associated Deliverables, Page 54

The deliverables listed below are required during the System Testing Phase:

- Unit Test Plan
- Unit Test Results Report
- System Test Plan
- System Test Results Report
- Integration Test Plan
- Integration Test Results Report
- Regression Test Plan
- Regression Test Report
- Pilot Test Plan (if applicable)
- Pilot Test Report (if applicable)
- User Acceptance Test Plan
- User Acceptance Test Results Report

If the Bidder indicates that pilot testing is an appropriate option for the proposed solution, a Pilot Test Plan must be prepared. Each of the test plans must include, at a minimum: (1) Sample standard test checklists or scripts; (2) descriptions of the test environments for each phase; (3) descriptions of all tools to be used during the test phases; (4) definitions of software defects, including descriptions, examples, priority rating, defect reporting process, and the resolution and retest process; (5) manual and/or automated testing approaches; (6) 508 compliance testing; (7) stress, load, and performance testing specific to the testing phase; (8) entrance and exit criteria for the testing phase; and, (9) metrics to be used to evaluate the testing phase.

Each of the test results reports must include, at a minimum: (1) copies of the executed test scripts, checklists, etc., for each phase; (2) reports of all defects, their priority assessments, and their resolution and retest results; (3) metrics of the test phase results; and, (4) lessons learned and recommendations to move to the next level.

The outcome of the System Testing phase is the creation and submission for DSCYF approval the following deliverables:

- Unit Test Plan
- Unit Test Results Report
- System Test Plan
- System Test Results Report
- Integration Test Plan
- Integration Test Results Report
- Regression Test Plan
- Regression Test Report
- User Acceptance Test Plan
- User Acceptance Test Results Report

Each of the test plans includes, at a minimum the following: (1) Sample standard test checklists or scripts; (2) descriptions of the test environments for each phase; (3) descriptions of all tools to be used during the test phases; (4) definitions of software defects, including descriptions, examples, priority rating, defect reporting process, and the resolution and retest process; (5) manual and/or automated testing approaches; (6) 508 compliance testing; (7) stress, load, and performance testing specific to the testing phase; (8) entrance and exit criteria for the testing phase; and, (9) metrics to be used to evaluate the testing phase.

Each of the test results reports also includes, at a minimum the following: (1) copies of the executed test scripts, checklists, etc., for each phase; (2) reports of all defects, their priority assessments, and their resolution and retest results; (3) metrics of the test phase results; and, (4) lessons learned and recommendations to move to the next level.