System Testing

Contents

[System Testing 2](#_Toc447812626)

[Why System Testing 2](#_Toc447812627)

[Testing Against the Architecture 2](#_Toc447812628)

[Example Architecture 2](#_Toc447812629)

[Criteria for Motivating Architectural Test Cases 2](#_Toc447812630)

[Building Test Cases for the Architecture 3](#_Toc447812631)

[Functional System Testing 3](#_Toc447812632)

[Deployment Testing 3](#_Toc447812633)

[Beta Testing 4](#_Toc447812634)

[Certification, Standards, and Testing for Compliance 4](#_Toc447812635)

[Non-Functional Testing Areas 4](#_Toc447812636)

[Performance Testing 4](#_Toc447812637)

[Scalability Testing 5](#_Toc447812638)

[Reliability Testing 5](#_Toc447812639)

[Stress Testing 5](#_Toc447812640)

[Interoperability Testing 5](#_Toc447812641)

[Localization Testing 5](#_Toc447812642)

[References 6](#_Toc447812643)

# System Testing

* Used to evaluate the system’s compliance with the specified requirements
* Helps to identify defects that are fundamental to the design, architecture, and implementation
* Tests both the functional and non-functional aspects

# Why System Testing

* Provide an independent perspective in testing
* Bring a customer perspective in testing
* Test the product behavior in a realistic environment
* Analyze and reduce the risks with releasing the product
* Ensures all requirements are met and ready the product for acceptance testing

# Testing Against the Architecture

The architecture is the high-level design of the application. It specifies the internal and external high-level elements (systems/components/actors) and the interfaces between the elements.

## Example Architecture

|  |
| --- |
|  |

***What types of interface paradigms are represented here?***

## Criteria for Motivating Architectural Test Cases

* **Completeness**
  + A sufficient set of interfaces are defined to provide all the services needed for the application’s functionality. The relationship between the interfaces allows for the flow of control and data necessary to realize all of the uses described in the use case.
* **Correctness**
  + The architecture satisfies its constraints; uses the appropriate architectural patterns; represents the interactions between interfaces.
* **Consistency**
  + Each use of the system can be handled only in one set of interfaces.

## Building Test Cases for the Architecture

You use the scenarios (use-cases) to motivate the questions. Questions are typically of the form: ***How does component x send/receive data to/from component y?***

|  |  |  |
| --- | --- | --- |
| **Scenario** | **Question** | **Expected Interface** |
| 1. The instructor inputs the base score, student name, and student score. 2. The instructor issues the compute command. 3. The system computes the letter grade. 4. The system outputs the letter grade. | How does the application receive the student data? | The application provides a student data entry user interface. |
| How does the application report the result? | The application provides a letter grade user interface. |

|  |
| --- |
|  |

# Functional System Testing

* The focus should be on product features
* Create and document realistic scenarios to execute the product features
* There should be a clear expected results document
* Types
  + Deployment Testing
    - Offsite vs. Onsite
  + Beta Testing
  + Certification, Standards, and Testing for compliance

## Deployment Testing

* Offsite
  + Simulating the customer’s configuration in-house to ensure that the customer’s deployment requirements are met
* Onsite
  + Using the customer’s resources and ensuring that the deployment requirements are met
  + Part of acceptance testing
  + Two stages
    - Stage 1 – mirror the customer’s configurations and replicate actions made on the existing live system – record the transactions
    - Stage 2 – convert the mirror to a live system
      * Keep regular backups
      * Ensure that it is possible to revert to the old system

## Beta Testing

* Sending the product under test to the customer and receive feedback
  + Reduces a number of risks with the product not meeting requirements
* Activities
  1. Identify a list of users and their expectations
  2. Identify a beta program schedule
  3. Prepare the beta testers
  4. Ensure that the product meets testing entry criteria
  5. Send the beta product to the users
  6. Collect feedback periodically
  7. Respond to feedback
  8. Analyze and decide if the beta program met exit criteria
  9. Communicate progress with users and customers and close the beta program
  10. Incorporate appropriate changes

## Certification, Standards, and Testing for Compliance

* The product should be certified on popular hardware, operating system, and other popular infrastructure
* The product should implement identified standards – e.g. Open LDAP, ODBC
* The product should meet contractual, legal, and statutory compliance – e.g. ADA

# Non-Functional Testing Areas

* Performance/Load Testing
* Scalability Testing
* Reliability Testing
* Stress Testing
* Interoperability Testing
* Localization Testing

## Performance Testing

* Used to identify defects by checking that the system can:-
  + Process the required number of transactions in any given interval (throughput)
  + Be available when running under different load conditions (availability)
  + Respond fast enough for different load conditions (response time)

## Scalability Testing

* The goal is to find the maximum limits of the system
* The design may give an idea of these limits
* Helps to identify the major bottlenecks in the system
* Should be performed on different configurations

## Reliability Testing

* Looking for defects in the product’s ability to perform its required functions repeatedly
  + E.g. login 10,000 times
* Characteristics of a “reliability tested product”
  + No errors or very few errors from repeated transactions
  + Zero downtime
  + Optimum utilization of resources
  + Consistent performance and response time
  + No side-effects after the repeated transactions

## Stress Testing

* Looking for defects by evaluating the system beyond the limits of the specified requirements
* Guidelines for selected test cases
  + Repetitive tests
  + Concurrency
  + Magnitude
  + Random variation

## Interoperability Testing

* Looking for defects when other systems interact with the product
* Things to examine
  + Information flow across systems should be consistent
  + Data representation is changed as needed e.g. big endian to little endian
  + Standard protocols are adhered to e.g. FTP
  + Response to unrecognized messages

## Localization Testing

* Changing the different locales using the system settings or environmental variables, and testing the software functionality, number, date, time, and currency format
  + All identified locales should be tested
  + Hot keys, function keys, and help screens are tested with applicable locales
  + Date and time formats are in line with the locale e.g. British date dd/mm/yyyy
  + Currency is line with the locale
  + Number format is in line with the locale
  + Time zone information and daylight saving time calculations are consistent and correct
* All output is localized to reflect native users and the conventions of the country, locale, and language
* Sorting and case conversions are correct as per language
* Font sizes and Hot keys are working correctly
* Filtering and searching capabilities of the software work as per language and locale
* Addresses, phone numbers, numbers, and postal codes in the localized software as per conventions of the target user

# References

Naik, K., & Tripathy, P. (2008). *Software Testing and Quality Assurance Theory and Practice.* Wiley.