**Testing GUI**

GUI testing is the process of testing the system's Graphical User Interface of the Application under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars - toolbar, menu bar, dialog boxes and windows, etc.

GUI is what user sees. Say if you visit guru99.com what you will see say home page it is the GUI (graphical user interface) of the site. A user does not see the source code. The interface is visible to the user. Especially the focus is on the design structure, images that they are working properly or not.

GUI testing is the process of ensuring proper functionality of the graphical user interface ([GUI](http://searchwindevelopment.techtarget.com/definition/GUI) ) for a given application and making sure it conforms to its written specifications.

In addition to functionality, GUI testing evaluates design elements such as layout, colors, [fonts](http://whatis.techtarget.com/definition/font), font sizes, labels, text boxes, text formatting, captions, buttons, lists, icons, links and content. GUI testing processes can be either manual or automatic, and are often performed by third -party companies, rather than developers or end users.

GUI testing can require a lot of programming and is time consuming whether manual or automatic. Usually the software author writes out the intended function of a [menu](http://whatis.techtarget.com/definition/menu) or graphical button for clarity so that the tester will not be confused as to the expected outcome. GUI testing also tends to test for certain program behaviors that users expect, like an hourglass when the program is busy, the F1 key bringing up the [help system](http://whatis.techtarget.com/definition/help-system-help-file) and many other common details.

User interface testing, a testing technique used to identify the presence of defects is a product/software under test by using Graphical user interface [GUI]

**GUI Testing - Characteristics:**

* GUI is a hierarchical, graphical front end to the application, contains graphical objects with a set of properties.
* During execution, the values of the properties of each objects of a GUI define the GUI state.
* It has capabilities to exercise GUI events like key press/mouse click.
* Able to provide inputs to the GUI Objects.
* To check the GUI representations to see if they are consistent with the expected ones.
* It strongly depends on the used technology.

**GUI Testing - Approaches:**

* Manual Based - Based on the domain and application knowledge of the tester.
* Capture and Replay - Based on capture and replay of user actions.
* Model-based testing - Based on the execution of user sessions based on a GUI model. Various GUI models are briefly discussed below.

**Model Based Testing - In Brief:**

* Event-based model - Based on all events of the GUI need to be executed at least once.
* State-based model - "all states" of the GUI are to be exercised at least once.
* Domain model - Based on the application domain and its functionality.

**GUI Testing Checklist:**

* Check Screen Validations
* Verify All Navigations
* Check usability Conditions
* Verify Data Integrity
* Verify the object states
* Verify the date Field and Numeric Field Formats

### GUI Testing Tools

* Selenium
* QTP
* Cucumber
* SilkTest
* TestComplete
* Watir

**GUI Testing Test Cases**

* **GUI Testing basically involves**

1) Testing the size, position, width, height of the elements.

2) Testing of the error messages that are getting displayed.

3) Testing the different sections of the screen.

4) Testing of the font whether it is readable or not.

5) Testing of the screen in different resolutions with the help of zooming in and zooming out.

6) Testing the alignment of the texts and other elements like icons, buttons, etc. are in proper place or not.

7) Testing the colors of the fonts.

8) Testing the colors of the error messages, warning messages.

9) Testing whether the image has good clarity or not.

10) Testing the alignment of the images.

11) Testing of the spelling.

12) The user must not get frustrated while using the system interface.

13) Testing whether the interface is attractive or not.

14) Testing of the scrollbars according to the size of the page if any.

15) Testing of the disabled fields if any.

16) Testing of the size of the images.

17) Testing of the headings whether it is properly aligned or not.

18) Testing of the color of the hyperlink.

**What do you Check in GUI Testing?**

* The following checklist will ensure detailed GUI Testing.
* Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
* Check you can execute the intended functionality of the application using the GUI
* Check Error Messages are displayed correctly
* Check for Clear demarcation of different sections on screen
* Check Font used in application is readable
* Check the alignment of the text is proper
* Check the Color of the font and warning messages is aesthetically pleasing
* Check that the images have good clarity
* Check that the images are properly aligned
* Check the positioning of GUI elements for different screen resolution.

**Advantages of GUI**

1) Easy to Operate

2) No Need of Remembering Commands

3) Easy to Learn

4) Improved Speed and Efficiency

5) Convenient and user friendly

6) User Customization

7) Accessibility Features

**Disadvantages of GUI**

1) Difficulty of cursor movements

2) Difficulty of displaying all necessary controls because of limited screen space

3) Difficulty of searching items that are hidden in nested containers

4) Ambiguity of pictorial symbols

5) Incompatibility between different standards and different platforms

**Software Testing Documentation**

**Software Testing Documentation: What’s that?**

We all read various articles on testing technologies and methods, but how many of us have seen articles on documentation? No doubt there are few, Is it that documents are not important? No, but its’ because we have not yet realized importance of documents.

But, if we observe then the fact is, **projects that have all the documents have high level of maturity.** Most companies do not give even a little importance to the documentation as much they give to software development process. If we search on web then we can find various templates on how to create various types of documents. But how many of them are really used by organizations or individuals?

Fact is that, **careful documentation can save an organization’s time, efforts and money.** While going for any type of certification, why there is stress given on documentation, it’s just because it shows importance of client and processes to individual and organization. Unless you are able to produce document that is comfortable to user no matter how good your product is, no one is going to accept it.

It’s my experience, we own one product, which is having a bit confusing functionality. When I started working on that I asked for some help documents to Manager and I got answer “No, we don’t have any documents” Then I made an issue of that, because as a QA I knew, no one can understand how to use the product without documents or training. And if user is not satisfied, how we are going to make money out of that product?

**“Lack of documentation is becoming a problem for acceptance” – Wietse Venema**  
Even same thing is applicable for User manuals. Take an example of Microsoft, they launch every product with proper documents, even for Office 2007 we have such documents, which are very explanatory and easy to understand for any user. That’s one of the reasons that all their products are successful.

**Let’s organize all discussion in few points in quality perspective:**  
– Clarify quality objective and methods  
– Ensure clarity about tasks and consistency of performance  
– Ensure internal co-ordination in client work  
– Provide feedback for preventive actions  
– Provide feedback for your planning cycle  
– Create objective evidence of your quality management system’s performance

There are hundreds of documents used in software development and testing life cycle. **Here I am listing few important software testing documents that we need to use/maintain regularly:**

1) [Test plan](http://www.softwaretestinghelp.com/category/test-plan-template/)  
2) Test design and [Test case specification](http://www.softwaretestinghelp.com/how-to-write-effective-test-cases-test-cases-procedures-and-definitions/)  
3) Test Strategy  
4) Test summary reports  
5) [Weekly Status Report](http://www.softwaretestinghelp.com/how-to-write-software-testing-weekly-status-report/)  
6) User Documents/ manuals  
7) User Acceptance Report  
8 ) [Risk Assessment](http://www.softwaretestinghelp.com/types-of-risks-in-software-projects/)  
9) Test Log  
10) [Bug reports](http://www.softwaretestinghelp.com/how-to-write-good-bug-report/)  
11) [Test data](http://www.softwaretestinghelp.com/tips-to-design-test-data-before-executing-your-test-cases/)  
12) Test analysis

🡪Documentation is an important not only to QA people, it is also important to Development team and organization, because it is a proof to client if we get any issues and important for future reference. Every document will have a particular piece of information.

## Software testing of real-time systems

**Introduction**

**What is a real-time system?**

A real time system is any system that “is subject to time constraints” meaning that it has to complete a task or tasks in a certain amount of time. The response to the input for this task is considered correct if the result is valid and if it was delivered in the right amount of time. Furthermore, this type of system is not only subject to deadlines but must also be predictable with its responses and should not be affected by external factors, for example, such as network traffic that can affect the performance of a system.

Examples of real-time systems include air traffic control systems, medical monitoring systems, weapons delivery systems and space navigation and guidance systems. All of which are considered to be safety critical systems.

There are also degrees of real-time systems that are classified as follows:

Hard real-time system – In which it is absolutely imperative that operations are completed within their deadline in this case if an operation is completed after the time in which it is supposed to then it is considered useless e.g. systems in space shuttle. In which case if the system did not react within the strict deadline it could result in serious issues such as the loss of life or damage to the equipment such as the shuttle crashing. As can be seen in this situation a late response would be pointless.

Soft real-time system – Although it is emphasised that deadlines are important they are not as strict as in hard real-time systems. Deadlines in this case can be missed occasionally but the important thing in this type of system is to try and reduce the amount of deadlines that are missed. Examples of soft real-time systems include operating systems.

It is also important not to confuse real-time systems with high performance systems a real-time system is one that is designed to meet timing constraints placed on it and once this has been achieved no further increases in performance are required unlike a high performance system.

**Introduction to real-time testing**

Real-time testing involves testing a system when it is running at its normal operating speed to ensure that operations are completed within their deadlines. When is comes to testing real time software there is not one single method for testing it because of the nature of the software and the time constraints placed on it. As a result of this the software is quite difficult to test, meaning that we have to be exhaustive with our testing, which is why there isn’t one single method in which to test it. Therefore, in order to test such software several methods need to be employed in order to do this, these could include both black and white box testing, sequence testing and integration testing.

**How do we test real-time systems?**

Real time software cannot be tested with a single type of analysis because of the nature of the software and the constraints placed upon it. Therefore, in order to test such systems we need to not only be exhaustive with the testing to ensure that each part of the software complies with the time constraints placed on it. But also that we conduct low-level and high-level testing, for example, it may be necessary to test single lines of code in terms of low-level testing if that line is significant and may affect time constraints that are placed on the software. High-level testing may include testing functions and methods in order to ascertain that they are valid and do not respond unpredictably.

This means that one method that is used to test real-time software, is to generate test sequences by applying time constraints in which it must be completed. As discussed above these sequences can be used to test that single lines of code produce their expected output or can be used in order to test whole functions and methods. Testing these sequences means that data must be sent or received during a time interval that is relevant to the beginning of the test. If the data is sent or received in this time interval it is said that the sequence has passed. If not then it has failed and if the result is unexpected but valid it is inconclusive. If the response if unexpected then the element being tested has to be reconsidered as unexpected responses are undesirable.

Furthermore, when testing such systems as well as considering both low and high-level testing testers must also consider both black and white box testing.

From the research that has been conducted a 3-stage approach has been developed that can be used to structure the testing of real-time software:

1. Identification of tasks

Output testing of tasks

Performance testing of tasks

2. Incorporation of tasks

Output Testing

Performance testing in a real environment

3. Integration Testing

The first stage should involve identifying the main tasks and functions of the software that may potentially affect deadlines. Once we have done this the output of the tasks must to be tested to ensure that the deadlines are met. The performance of these tasks needs also to be considered and tested which is usually achieved through the use of CASE tools to analyze behavior.

In the next stage each task should be tested once it has been incorporated into the software to ensure that each component works with the software as a whole and again in this situation the outputs of the incorporated components should be tested to make sure that they are still being completed within their deadlines. Also for this stage the performance of the software should be tested in a real environment before it is used.

Finally the software should be integrated into the system and fully tested to ensure that there are no errors with relation to the integration of the hardware and software interfaces.

Difficulties that have been encountered with testing real-time software

There are a number of difficulties that accompany testing real-time systems. These being that it is difficult to develop methods in which to test a system that is running with a real time constraint in which responses maybe too early or too late unlike other systems that may not be subject to such deadlines. Furthermore, because of these constraints it can be difficult to predict what the outputs may be, meaning that the behavior of this type of software is non-deterministic.

As a result of these difficulties of time constraints and there being no single method of testing it means that testing must be exhaustive particularly depending on what the software is being used for (e.g. a space shuttle mistakes result in loss of life). Therefore, this means that to carry out thorough testing it could potentially be very expensive in terms of time and cost, and perhaps be one of the most expensive parts of the project. More often than not in the case of real-time software because a lot of them are safety critical testing can often take months or longer in order to verify and validate responses to ensure that they are correct and as predicted.

Not only is it difficult to devise methods to test singular deadlines it is difficult to test this kind of software because there will be a degree of concurrency as it may not only be subject to time constraints but there may be multiple deadlines that that require a response at the same time which would mean that each individual deadline would need to be tested as well as the deadline as a whole. Conventional methods are not applicable and there are no commercial tools currently available for testing real-time software this can lead to unpredictable outcomes which if we consider the definition of a real-time system, that it should be predictable we do not want unpredictable outcomes.

**Summary**

In summary the main points to be considered in this report are that a real time system in one that is subject to teal time constraints and should be predictable. Of which there are also two main types of real-time system known as hard real-time and soft real-time systems. When testing this type of software many different methods of testing should be considered as there is not one specific method in which to test this type of system. As a result of this there are many difficulties that accompany attempts to test such systems.

**Conclusion**

In conclusion it would seem that there is not one specific way in which to test real-time software but it is simply the case that critical components and functions are tested fully. Like any testing it is important to find the balance between testing every line of code to testing units and whole systems. Furthermore, it is important to work out when it is useful to test single lines of code and units in order to find a balance between time, cost and the nature of the system should also be considered. As although testing this type of software should be exhaustive it is not feasible to test for example, every line of code therefore, a judgment should be made as to what should be tested with regards to what part of the software could affect any and all deadlines that have been placed upon it. Also if there are any factors that could affect the response and give unpredictable results.