|  |
| --- |
| Requirement driven testing |
| Test Plan for |
| Irrenhaus |
|  |
| **Requirement Driven Testing** |
| **11/6/2014** |

|  |
| --- |
| This document contains the break down of testing project Irrenhaus. It includes component testing, system testing and a testing schedule. |

# Approval

|  |  |  |  |
| --- | --- | --- | --- |
| AUTHOR | | | |
| Cassandra Siewert | Lead Designer/Programmer |  | |
| APPROVED BY | | | |
| Cassandra Siewert | Lead Designer/Programmer |  |  |
| ACKNOWLEDGMENT | | | |
|  | | | |

# Related Documents

|  |  |
| --- | --- |
| **Ref #** | **Document Name** |
| 01 | High level system architecture |
| 02 | Project brief |
| 03 | Infrastructure design.. etc |

Glossary

|  |  |
| --- | --- |
| Term | Definition |
| Bug | See Defect |
| Defect | Software function does not work as per specification |
| Defect Owner | The person who created the defect |
| Issue | Software function does not work as expected or is not specified |
| RDT | Requirement Driven Testing |
| SDLC | Software Development Life Cycle |
| SME | Subject Matter Expert |
| TDD | Test Driven Environment |
| UAT | User Acceptance Testing |

Contents

[Approval 2](#_Toc272783525)

[Related Documents 2](#_Toc272783526)

[Glossary 2](#_Toc272783527)

[Introduction 5](#_Toc272783528)

[Purpose 5](#_Toc272783529)

[Project Overview 5](#_Toc272783530)

[Testing objectives 6](#_Toc272783531)

[Features to be tested 6](#_Toc272783532)

[Features Not to be tested and constraints 6](#_Toc272783533)

[Testing Approach 6](#_Toc272783534)

[Static Testing 6](#_Toc272783535)

[Component Testing 6](#_Toc272783536)

[Entry Criteria 6](#_Toc272783537)

[Suspension Criteria 6](#_Toc272783538)

[Resumption Criteria 7](#_Toc272783539)

[Exit Criteria (Test Completeness) 7](#_Toc272783540)

[System Testing 7](#_Toc272783541)

[Entry Criteria 7](#_Toc272783542)

[Suspension Criteria 7](#_Toc272783543)

[Resumption Criteria 7](#_Toc272783544)

[Exit Criteria (Test Completeness) 8](#_Toc272783545)

[User Acceptance Testing (UAT) 8](#_Toc272783546)

[Defect Management 9](#_Toc272783547)

[Defect Status 10](#_Toc272783548)

[Defect Severity Levels 10](#_Toc272783549)

[Test Activities and Schedules 11](#_Toc272783550)

[Week 1 (date from – date to) 11](#_Toc272783551)

[Week 2 (date from – date to) 11](#_Toc272783552)

[Test Deliverables 12](#_Toc272783553)

[Test Reporting 12](#_Toc272783554)

[Test Environment Control 13](#_Toc272783555)

[Summary 13](#_Toc272783556)

[Release versioning 13](#_Toc272783557)

[Testing Resources 13](#_Toc272783558)

[Cassandra Siewert - Project Manager 13](#_Toc272783559)

[Cassandra Siewert - Business Analyst 13](#_Toc272783560)

[Testing Tool 13](#_Toc272783561)

[Appendix 14](#_Toc272783562)

# Introduction

This test plan document describes the scope, approach, resources and schedule of intended testing activities to be undertaken for Irrenhaus. This document should be read in conjunction with the Test Strategy.

## Purpose

This document provides the following guidance:

* Testing Scope;
* Entry and exit criteria for each test level
* A description of resources and tools to be used to conduct testing;
* An overview of test schedules per development cycle;
* An overview of the types of testing that is to be conducted;
* Defect management work flow.

## Project Overview

The Irrenhaus test plan is a structured guideline which will break down testing into different stages which will help minimize and eliminate various bugs in the program. The goal of this test plan is to have a finished shippable product without game breaking bugs. In order to achieve this the game is being broken down into basic components and tested. Once the initial testing is successfully completed (components work on their own) then the components will be assembled and system testing will start. This will eliminate any minor bugs that could become a huge problem when the components are assembled. The project will then be tested by users or potential players/investors to see if it is meeting their requirements. This will ideally result in a very functional and desirable game

The goal for the testing of the Irrenhaus prototype is to have the basic features of the game working that way more time can be spent on the design and story.

# Testing objectives

## Features to be tested

Tile Map Engine

Sprite Animation

Player Input and Collision Detection

Basic AI

Player Powers

## Features Not to be tested and constraints

Lighting

Sound

Story Content/Dialogue output

Menu

Testing Approach

The purpose of the testing is to verify the functionality of all components, ensuring they satisfy the defined and agreed technical and business requirements. Requirement Driven Testing[[1]](#footnote-1) is the preferred approach focussing on the following:

1. Building business requirement list where test cases will be derived from;
2. Requirement is used to select which test case(s) to execute and;
3. Report on business requirements instead of test cases.

## Static Testing

Static testing is testing of a component or specifications without execution of that software. This is usually done as soon as acceptance criteria or business requirements are ready for review before code implementation such as conflicting rules, invalid data types, redundant process just to name a few.

## Component Testing

Component level testing focuses on the functionality of each component being developed. This is crucial where different components are being developed before they are integrated together as one system.

### Entry Criteria

Component Testing may commence when the following criteria have been satisfied:

1. All codes have been unit tested and passed.
2. Test environment including software have been setup and configured correctly.
3. Business requirements and test cases are up to date as per user story.
4. Code includes error handling to output error messages that are easily understood

### Suspension Criteria

Component testing will be suspended under the following condition:

1. Critical error(s) found preventing test completion.
2. Change of business requirements.
3. Change of environment components or technology including different version.
4. Change in component requirements

### Resumption Criteria

Component testing will resume when the following criteria are met:

1. All issues in suspension criteria have been resolved or mitigated
2. New software build has been redeployed or;
3. New build with fixed Critical and Medium severity defects has been deployed into Test.
4. Changes for requirements have been completed

### Exit Criteria (Test Completeness)

Component testing can be considered complete when the following conditions have been met:

1. All High and Medium priority requirements have been tested without Critical or Medium severity defects.
2. Any remaining bugs can easily be fixed in a short time period

## System Testing

The purpose of the system testing is to validate that the complete and integrated system complies with functional requirements and business requirements.

### Entry Criteria

System testing may commence when the following criteria have been satisfied:

1. Component Testing has been completed.
2. No change to business requirements and test cases are up to date.
3. Scenario based test cases have reviewed by business owners or business users.
4. Code has been thoroughly commented to improve legibility

### Suspension Criteria

System Testing will be suspended under the following condition:

1. Critical error(s) found affecting functionality of the whole system.
2. Change of business requirements
3. Refactoring code to include different feature

### Resumption Criteria

System Testing will resume when the following criteria have been satisfied:

1. All issues in suspension criteria have been resolved or mitigated
2. New build with fixed Critical and Medium severity defects has been deployed into Test.
3. There are no additional features/modifications to make

### Exit Criteria (Test Completeness)

System Testing will be considered complete when the following conditions have been met:

1. All High and Medium priority requirements have been tested without Critical or Medium severity defects.
2. Business owner(s) and/or business user(s) have been notified with any remaining defects and understand the risks or limitations of current release.
3. All defects found during testing have been recorded in defect management tool.
4. Remaining defects have been ordered by priority to fix

## User Acceptance Testing (UAT)

UAT is a formal testing with respect to user needs, business requirements and expectations. The idea here is to gain confidence from business owner on the software being developed. Although it is not mandatory business owner(s) and/or business user(s) are expected to produce his/her own test scenarios.

## Defect Management[[2]](#footnote-2)



Figure 1 - Defect workflow

### Defect Status

Every defect must be assigned a status to identify its place in the defect management workflow.

|  |  |
| --- | --- |
| **Status** | **Description** |
| New | Defect found in most recent testing |
| Active | Defect that was found in previous testing but hasn't been resolved |
| Resolved | The defect has been eliminated by coding a fix for it |
| Investigate | A defect that needs to be looked into to find out why the problem is occurring |
| Closed | The reason for the defect has been discovered |

**Table**  **- Defect Status**

### Defect Severity Levels

Every defect must be assigned a severity level according to the following table. If the tester is unsure what level to assign to a defect, then advice must be taken from the business owners or business users.

| **Level** | **Type** | **Description** |
| --- | --- | --- |
| 1 | High | Game breaking defect. Nothing works, can't play the game. eg. game won't load |
| 2 | Medium | Game can be played but 1 or more features don't work eg. collisions aren't being detected |
| 3 | Low | Game can be played, but some values and results are unpredictable  eg. player sprite animation doesn't always play |

**Table**  **- Defect Severity Levels**

## Test Activities and Schedules

### Week 1 (date from – date to)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Activities** | **Entry criteria** | **Exit Criteria** | **Priority** | **Status** |
| Develop Test Strategy | High level architecture documentation and project scope. | Document completed and reviewed | High | Done |
| Develop Test Plan | High level architecture and scope for user story. | Document completed and reviewed | High | Done |
| Static test and test case for Tile Map Engine | Business requirements have been documented. | Develop functional and negative test cases for each acceptance criteria. | Medium | Done |
| Component tests for Texture Manager, Tile, TileMap | Coding requirements laid out in UML have been met | Texture Manager- keeps track of textures, Tile object can be created, TileMap loads xml | High | Done |
| System tests for TileMap Engine | Component tests have been successfully completed | TileMap loads the levels, stores textures using Texture Manager, and stores each pieces attributes as a tile | High | Done |

### Week 2 (date from – date to)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Activities** | **Entry criteria** | **Exit Criteria** | **Priority** | **Status** |
| Static test and test case for Player | Player class requirements have been documented | Develop functional and negative test cases for each acceptance criteria. | Medium | Done |
| Component tests for Player | Coding requirements laid out in UML have been met | Loads player texture, updates player movement based on user input | High | Done |
| System tests for player interaction with tile engine | Collision handling needs to be coded | Player is able to interact with tiles eg. walk on walkable, bump into non walkable | High | In-Progress |
| **The following tasks are carried over from last week or added** | | | | |
|  |  |  |  |  |
|  |  |  |  |  |

Done – task complete

In progress – currently working on it

Pending – ready to start but waiting for requirements

Not started – planned tasks

Removed – task no longer required

## Test Deliverables

Testing Team will provide specific deliverables during the project. These deliverables fall into the following basic categories:

1. Documents,
2. Test Cases / Bug Write-ups, and
3. Reports.
4. Potential Bug Solutions

## Test Reporting

The following test reports will be used to monitor and manage test progress:

* The number of requirements (passed and failed results);
* The number of defects (bugs) identified over the test cycle sub-categorized by severity level as shown in the following example.

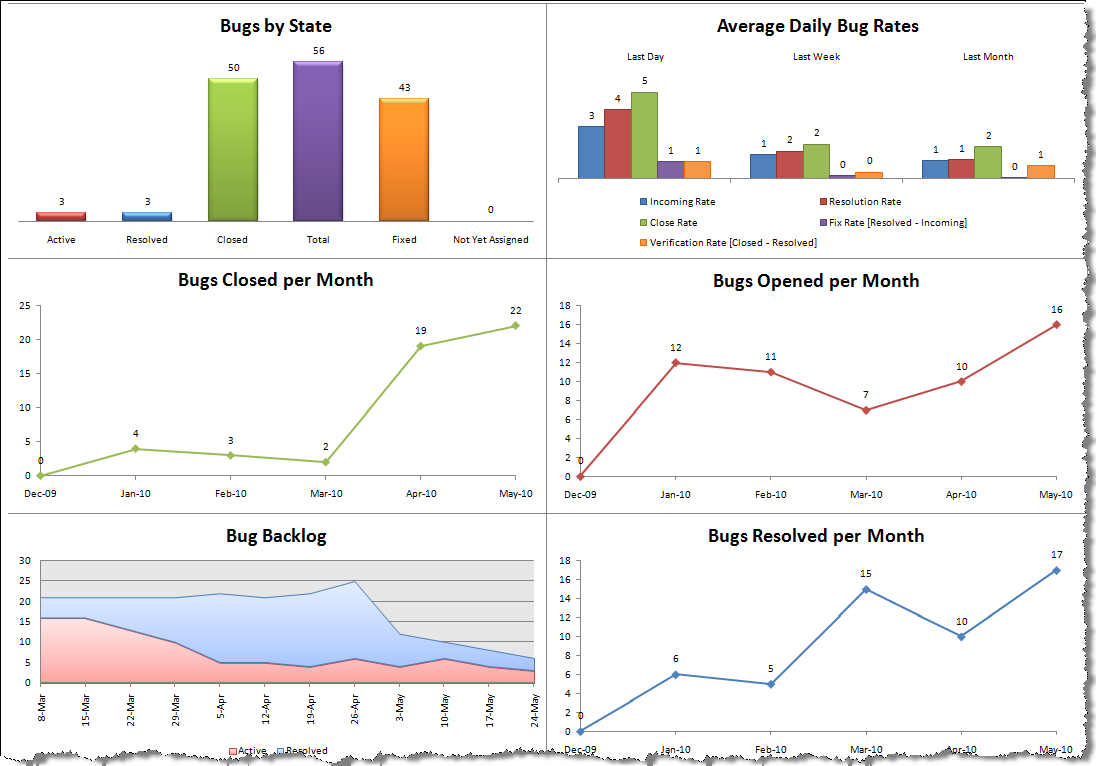
****

Figure 2 - Defect Report Example using Team Foundation Server (TFS)

* A Final Test Summary Report will be issued by the Test Manager. It will certify the extent to which testing has been completed, and provide an assessment of the product readiness for Program End-to-End Testing <add more details>

## Test Environment Control

### Summary

I will be managing the testing environment. I will create a separate project in the solution to test various components.

### Release versioning

|  |  |
| --- | --- |
| **Version #** | **Description of version** |
| 0.0.1 | Some basic features implemented eg. Tiles loading - Alpha |
| 0.0.2 | More basic features implemented getting closer to beta version eg. Tiles and player loading - Alpha |
| 1.0.0 | After or still beta, full version of playable game all basic features are implemented - Beta or possible Release Candidate |
| 1.0.1 | Full version of playable game, some minor additions -Beta or possible Release Candidate |

**Table**  **- Version numbering example**

## Testing Resources

### Cassandra Siewert - Project Manager

Responsible for:

* keeping track of defects and priority

### Cassandra Siewert - Business Analyst

Responsible for:

- making sure basic features that were pitched and appeal to investors or future players are being coded and tested

-feature creep

## Testing Tool

The game is being coded in C++ using SFML so testing will be done using a rendering window and a console to output variable information and states. Tests will be recorded on a table to keep track of defects and their priority.

# Appendix

Testing Table Example

**Version: 0.0.2 Date: October 20, 2014**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action/function being tested** | **Expected result** | **Actual Result** | **Pass/ Fail** | **Defect Priority** |
| Loading tiles | Tile texture will load | White screen | Fail | High |
|  |  |  |  |  |
|  |  |  |  |  |



1. [↑](#footnote-ref-1)
2. In Requirement Driven Testing it is recommended that [each defect must be linked to at least one requirement and test case](http://www.requirementdriventesting.com/software-test-methodology-rdtrule4/). [↑](#footnote-ref-2)