2015

MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

Poker Playing Agent

Software Test Documentation

REVISION: 1.0

STANDARD: IEEE STD 829 - 2008

A4

HÜSEYİN ALEÇAKIR

HÜSEYİN AYDIN

HEVAL AZİZOĞLU

ZÜLFİYE ARĞIN

PREFACE

This STD (Software Test Documentation) includes identification of system test cases for Poker Playing Agent Project. The document is prepared according to the "IEEE Standard for Software and System Test Documentation – IEEE Standard 829 – 2008".

This document provides a complete description of all the system test cases that are designed to verify and validate whether the "Poker Playing Agent" system meets its requirements.

The first section of the document includes problem definition, document identifier, scope, definition and abbreviations and references.

The second section provides test items and their identifiers, test traceability matrix, features to be tested and not be tested, approach adopted while testing, pass/fail criteria of tests and test deliverables.

The third section is about test management containing planned activities, environmental needs for test cases and schedule, estimates and costs of these cases.

The fourth section provides the details of test cases which can be categorized into 5 groups: unit tests, integration tests, system testing, validation tests and performance testing.

The fifth section describes the special steps and requirements of the test execution process.

The sixth section provides system test report details including overview of test results and detailed test results.

The last section contains quality assurance procedures, metrics that guides to evaluation of test items, test coverage, glossary and change history table for this document.

Record of Changes

Version Number	Date	Number of Section	A – Added M – Modified D - Deleted	Title or brief description
1.0	03.05.2015			First Release

Table of Contents

Table of Figures

1 INTRODUCTION

1.1 PROBLEM DEFINITION

The main purpose of Poker Playing Agent project is to find efficient and effective solutions to real life problems containing unreliable and incomplete information. Automatized problem solvers like agents may be handy while overcoming the challenge of the scope. To reflect all this constraints and to imitate the exact same conditions, poker game has been chosen as the main subject in the project. Poker is quite simple in rules but requires extremely complex strategically thinking. Hence in this project, it is aimed to build a successful poker agent in terms of making rational decisions, having good rate of win and meaningful opponent modelling in case of uncertainty.

1.2 DOCUMENT IDENTIFIER

This document is prepared by A4 team members with the purpose of documenting the testing process of the Poker Playing Agent project. The document explains design of test cases and procedure that should be conducted to verify and validate the system's features in detail so that any tester would be able to run and observe the outcomes. This document is the first version of Software Testing Documentation. Version changes can be traced by Record of Changes table in page 2.

1.3 SCOPE

The main purpose of this STD is to describe necessary test cases to verify Poker Playing Agent (PPA) system's functionality. All the test cases covering functional requirements mentioned in the Software Specification Document of this project will be given in this documentation in detail by stating their purposes, expected outcomes and success/fail criteria etc. Also testing procedures for system integration, validation and performance will be introduced within following sections. The PPA system will be tested under guidance of this document. However, this STD is a living document that it may need to be improved after further iterations in the development.

1.4 DEFINITIONS, TERMS AND ABBREVIATIONS

Abbreviations, Terms	Definitions
Busted Hand	A hand in poker which fails to fill a pair, straight and flush.
Call	To match a bet or raise.
CPT	Conditional Probability Table
Flush	Any five cards in same suit.
Fold	To discard one's hand and forfeit interest in the current pot.
IT	Integration Test
JDK	Java Development Kit
PPA	Poker Playing Agent
Pre-flop	A stage in Texas Hold'em Poker Game in which players decide according to their two cards only.
PT	Performance Test
Raise	To increase the previous bet.
SDD	Software Design Description
SRS	Software Requirements Specification
ST	System Test
STD	Software Test Documentation
Straight	Five consecutive cards of any suit.
UT	Unit Test
VT	Validation Test

Table 1: Definitions, Terms and Abbreviations

1.5 REFERENCES

- IEEE. IEEE STD 829-2008, IEEE Standard for Software and System Test Documentation. IEEE Computer Society, 2008.
- "Poker Playing Agent" Software Specification Requirements (SRS) Document, 2014.
- "Poker Playing Agent" Software Design Description (SDD) Document, 2014.

2 DETAILS FOR SYSTEM TEST PLAN

2.1 TEST ITEM AND THEIR IDENTIFIERS

Poker Playing Agent system is the only test item that will be considered in this documentation. All the functional requirements mentioned in the Software Requirements Specification (SRS) of PPA project will be tested by individual test cases as it is explained correspondingly in section "4. Test Case Details" of this STD. All test cases will be labeled with unique identifiers to ease the tracing.

To successfully run some integration tests, Poker Academy program which only available for Windows OS is needed to be installed in test environment. Also for unit and integration tests, Meerkat API versions 1.5 and 2.5, JavaBayes library version 0.346 and Java Development Kit 1.5 should be supplied in the test environment.

- Requirements: Software Requirements Specification of PPA
- Design: Software Design Description of PPA
- Plan: Planning, SRS of PPA
- Test: Software Test Documentation of PPA (this document)

2.2 TEST TRACEABILITY MATRIX

	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	Performance and System Requirements	Bayesian Network Component	Hand Evaluation Component
UT-1												X
UT-2												X
UT-3												X
UT-4											X	
UT-5											X	
UT-6											X	
UT-7											X	
UT-8											X	
UT-9										X		
IT-1											X	

IT-2												X
IT-3										X		
ST-1	X											
ST-2	X											
ST-3						X						
ST-4										X		
VT-1										X		
PT										X		
NOT TESTED		X	X	X	X		X	X	X			

Table 2: Test Traceability Matrix

2.3 FEATURES TO BE TESTED

The features to be tested and to be documented are mostly related with the PPA system's agent component's internal behaviour, agent's rationality and performance. Also since PPA requires to be integrated into Poker Academy to be run and tested, all interfaces between two components and compatibility factors will also be tested.

During the documentation of the PPA's testing phase, the following functionalities are highlighted:

- Algorithm and implementation issues in terms of time, accuracy and memory
- Platform requirements and dependencies
- Integration issues depend on the client application Poker Academy

2.4 FEATURES NOT TO BE TESTED

All the features obtained directly from client application Poker Academy and third party libraries will not be tested hence will not be listed in this document as test cases. These functional requirements specified in SRS of this project are listed below.

- UC2 - Choose Game Type
- UC3 - Choose Poker Type
- UC4 - Choose Table
- UC5 - Create New Table

- UC7 - Load Table
- UC8 - Choose Spectator mode
- UC9 - Deal Hand

2.5 APPROACH

Black box testing approach will be adapted while testing individual features and correctness of the internal unit calculations. Test codes will be written to be executed manually in case of need.

However, for general system testing white box testing approach will be used. This process is not depend on only the inputs and corresponding outputs but whole process needs to be traced to guarantee that everything goes fine. Also analysis process needs to be conduct for the requirements based on validation and performance.

2.6 ITEM PASS/FAIL CRITERIA

The result of each test case have to be compared with previously defined and expected outcomes. Status to define the results of test cases can be described as follows:

- **Passed:** Expected outcome matches with the actual result that is obtained from the test execution. In the scope of this project, pass criteria is the exact same output determined before black box test execution for units and successful interaction with the Poker Academy. Also rational decisions made in small amount of time will be observed and in case of satisfaction, this analysis stage will be marked as passed.
- **Failed:** Expected outcome does not match with the obtained result. Failed state is described in the testing process of this project as unit and integration tests does not generate expected predetermined outputs, system integration and Poker Academy interaction failed due to some inconsistencies between components or PPA is not able to satisfy users in terms of predefined requirements.
- **Blocked:** Test results cannot be obtained due to a blockage. An example in PPA can be dependencies to Poker Academy's environment. Some system tests of this project requires Poker Academy to be run which is only possible in Windows OS. More information will be provided why the test case is blocked.
- **NoRun:** Test case is not executed since implementation of related module is not complete yet.

After tests are executed, failed and blocked ones will be analyzed and required changes will be done according to results.

2.7 TEST DELIVERABLES

Test deliverable of this project is this Software Test Documentation itself. It includes the combination of the content of the following document types.

1. System Test Plan(s)
2. System Test Design
3. System Test Case
4. System Test Procedure
5. System Test Report

3 TEST MANAGEMENT

3.1 PLANNED ACTIVITIES AND TASKS; TEST PROGRESSION

Test process of the PPA project will start by analyzing our design, class structure and client application Poker Academy. For understanding and executing test cases, a tester should be familiar with the PPA project. To achieve this, tester should read System Requirement Specification and System Design Description of the project. These documents are available on the website[1] of A4.

When tester feels comfortable about the details of PPA project, all test cases based on their goals can be executed. For a better understanding, test cases are divided into groups according to their objectives. Majors sections are unit, integration, validation and performance testing respectively. Also for each test case, expected outcomes will be decided and according to what we expect to test in the particular case, inputs/parameters will be specified.

System testing is a team effort and each team member is equally responsible for every action in the process.

The results of the each test case will be shown at section 6.2 Detailed Test Results. Based on test results, tester will be enable to see what should be done in the future phases of the project.

3.2 ENVIRONMENT/INFRASTRUCTURE

To execute tests of the PPA system, a computer system with the following software and hardware characteristics will be needed:

- Java Development Kit 1.5
- Windows Operating System
- 128 MB RAM, 200 MB disk space and 800MHz CPU for Poker Academy to be run as expected.

---

[1] senior.ceng.metu.edu.tr/2015/a4/

3.3 SCHEDULE, ESTIMATES AND COSTS

Testing is a continuous process during the lifetime of the project. Every work done in a timely manner should be tested deeply and any defect should be handled immediately to progress more steady. Hence in PPA project, testing is a big part of every iteration and correctness of each new functionality is checked by unit, integration and system tests constantly.

Also, there are some other test types that need to be considered for PPA project after all functionalities are implemented and observation and analysis steps are in progress. This tests are validation and performance tests. Extra time slot will be scheduled to maintain these tests and to make improvements if any anomaly is observed. This phase will be conducted at the last iterations of the project and will not be take longer than 3 sprint.

4 TEST CASE DETAILS (ONCE PER TEST CASE)

4.1 UNIT TESTS

4.1.1 UT1 - Hand Type Classification

Objectives
This test case aims to test whether best five cards among all revealed ones successfully mapped to 25 hand abstractions.
Inputs
Hands information including 5 to 7 cards taken from Poker Academy
Outcomes
25 hand type abstraction class name which are: BUSTED_LOW (busted hands up to 9-high), BUSTED_MEDIUM (busted hands includes 10-high and J-high), BUSTED_QUEEN, BUSTED_KING, BUSTED_ACE, PAIR_TWO, PAIR_THREE, PAIR_FOUR, PAIR_FIVE, PAIR_SIX, PAIR_SEVEN, PAIR_EIGHT, PAIR_NINE, PAIR_TEN, PAIR_JACK, PAIR_QUEEN, PAIR_KING, PAIR_ACE, TWO_PAIR, THREE_KIND, STRAIGHT, FLUSH, FULL_HOUSE, FOUR_KIND, STRAIGHT_FLUSH
Environmental Needs
Meerkat API 1.5 and JDK 1.5 are required.
Special Procedural Requirements
There is no special procedure to run this test. It can be run as a simple Java code.
Intercase Dependency
There is no dependency for this test case.

Table 3: UT1 - Hand Type Classification

4.1.2 UT2 - Hand Similarity Detection

Objectives
This test case aims to investigate the probabilities of possible similarities between current hand and all 25 hand types.
Inputs
Current hand information including 5 or 6 cards
Outcomes
Respective probabilities of 25 hand types
Environmental Needs
Meerkat API 1.5 and JDK 1.5 are required.
Special Procedural Requirements
Deck class of Meerkat API should be initialized.
Intercase Dependency
This test depends on the success of UT1.

Table 4: UT2 - Hand Similarity Detection

4.1.3 UT3 - Two Cards Evaluation

Objectives
This test case aims to test the evaluation of current hand strength with only two cards in the pre-flop stage.
Inputs
Current hand information including only two cards
Outcomes
Hand rank between 9 to 333 to differentiate hands correctly
Environmental Needs
Meerkat API 2.5 and JDK 1.5 are required.
Special Procedural Requirements
There is no special procedure to run this test. It can be run as a simple Java code.
Intercase Dependency
There is no dependency for this test case.

Table 5: UT3 - Two Cards Evaluation

4.1.4 UT4 - Bayesian Node Construction

Objectives
This test case aims to test whether nodes in the Bayesian Network are successfully created by using JavaBayes library.
Inputs
Node name and node states
Outcomes
An object instance from BayesianNode class
Environmental Needs
JavaBayes 0.346 and JDK 1.5 are required.
Special Procedural Requirements
Helper methods to decorate JavaBayes' capabilities are needed to be included in the test file.
Intercase Dependency
There is no dependency for this test case.

Table 6: UT4 - Bayesian Node Construction

4.1.5 UT5 - Bayesian Node Dependency

Objectives
This test case aims to test whether nodes in the Bayesian Network are successfully connected with each other due to their parent child relations by using JavaBayes library.
Inputs
Names of parent and child node
Outcomes
Probabilistic dependency between nodes
Environmental Needs
JavaBayes 0.346 and JDK 1.5 are required.
Special Procedural Requirements
Helper methods to decorate JavaBayes' capabilities are needed to be included in the test file.
Intercase Dependency
This test depends on the success of UT4.

Table 7: UT5 - Bayesian Node Dependency

4.1.6 UT6 - Bayesian CPT Construction

Objectives
This test case aims to test whether CPTs of nodes in the Bayesian Network are successfully created with all required dimensions by using JavaBayes library.
Inputs
Node name, Parent node names, Node states
Outcomes
Updated version of child node's CPT.
Environmental Needs
JavaBayes 0.346 and JDK 1.5 are required.
Special Procedural Requirements
Helper methods to decorate JavaBayes' capabilities are needed to be included in the test file.
Intercase Dependency
This test depends on the success of UT4 and UT5.

Table 8: UT6 - Bayesian CPT Construction

4.1.7 UT7 - Bayesian CPT Update

Objectives
This test case aims to test whether CPTs of nodes in the Bayesian network are successfully updated with new calculated values in each turn by using JavaBayes library.
Inputs
Node name, value array
Outcomes
Updated version of node's CPT.
Environmental Needs
JavaBayes 0.346 and JDK 1.5 are required.
Special Procedural Requirements
Helper methods to decorate JavaBayes' capabilities are needed to be included in the test file.
Intercase Dependency
This test depends on the success of UT4, UT5 and UT6.

Table 9: UT7 - Bayesian CPT Update

4.1.8 UT8 - Opponent's Hand Estimation

Objectives
This test case aims to test whether possible hand types of opponent are estimated correctly with using information taken from Bayesian Network and Poker Academy.
Inputs
Opponent action and game state
Outcomes
Respective probabilities of 25 hand types that opponent might have
Environmental Needs
JavaBayes 0.346 and JDK 1.5 are required.
Special Procedural Requirements
Helper methods to decorate JavaBayes' capabilities are needed to be included in the test file.
Intercase Dependency
This test depends on the success of UT4, UT5, UT6 and UT7.

Table 10: UT8 - Opponent's Hand Estimation

4.1.9 UT9 - Persistence of Game State and Opponent Behaviour

Objectives
This test case aims to test whether game state with decided action and opponent behaviour are correctly saved into database.
Inputs
Opponent action, game state, decided action
Outcomes
A new entity in related table in database
Environmental Needs
Meerkat API 2.5 and JDK 1.5 are required.
Special Procedural Requirements
A database with required tables should be created and connection with the database server must be established.
Intercase Dependency
There is no dependency for this test case.

Table 11: UT9 - Persistence of Game State and Opponent Behaviour

4.2 INTEGRATION TESTS

4.2.1 IT1 - Bayesian Network Query Result

Objectives
This test case aims to test obtained results from the Bayesian Network with the given observed values of some nodes.
Inputs
The observed values of parent nodes and the node name of the required probability.
Outcomes
Probability of node state under interest
Environmental Needs
JavaBayes 0.346 and JDK 1.5 are required.
Special Procedural Requirements
Helper methods to decorate JavaBayes' capabilities are needed to be included in the test file.
Intercase Dependency
This test depends on the success of UT4, UT5, UT6 and UT7.

Table 12: IT1 - Bayesian Network Query Result

4.2.2 IT2 - Hand Evaluation

Objectives
This test case aims to verify that hand classification and hand similarity methods working compatibility and possible final hand types are estimated correctly.
Inputs
Current hand
Outcomes
Hand strength
Environmental Needs
Meerkat API 1.5 and JDK 1.5 are required.
Special Procedural Requirements
There is no special procedure to run this test.
Intercase Dependency
This test depends on the success of UT1, UT2 and UT3.

Table 13: IT2 - Hand Evaluation

4.2.3 IT3 - Probability Triple Construction

Objectives
This test case aims to verify that a rational probability triple to determine the next action of agent is created with information taken from Bayesian Network.
Inputs
Game state including information obtained from Poker Academy and constructed Bayesian Network
Outcomes
A probability triple of probabilities in the form of (fold, call, raise)
Environmental Needs
Meerkat API 1.5, JavaBayes 0.346 and JDK 1.5 are required.
Special Procedural Requirements
Bayesian network of the agent should be constructed.
Intercase Dependency
This test depends on the success of all the unit test cases.

Table 14: IT3 - Probability Triple Construction

4.3 SYSTEM TESTS

4.3.1 ST1 - Exporting Agent as Jar

Objectives
This test case is required to verify that source code written can be exported as jar without any compilation error since agents need be integrated into Poker Academy as jar files.
Inputs
Source code of agent
Outcomes
Jar file
Environmental Needs
JDK 1.5 are required.
Special Procedural Requirements
Source of agent should be compiled with Meerkat API 1.5 and JavaBayes 0.346 to jar resulting class files.
Intercase Dependency
There is no dependency for this test case.

Table 15: ST1 - Exporting Agent as Jar

4.3.2 ST2 - Integrating Agent into Poker Academy

Objectives
This test case aims to verify that exported jar file of the agent can be integrated into Poker Academy using Opponent Manager interface correctly.
Inputs
Jar file and pd file of agent
Outcomes
A new poker player in Poker Academy environment
Environmental Needs
Poker Academy
Special Procedural Requirements
Source code of agent should be compiled with Meerkat API 1.5 and JavaBayes 0.346 and should compose a jar file that includes class files. Also pd file for the agent should be created.
Intercase Dependency
This test depends on the success of ST1.

Table 16: ST2 - Integrating Agent into Poker Academy

4.3.3 ST3 - Adding Agent into Game Table

Objectives
This test case aims to verify that newly created agent can be added as player into a game table which means that it is recognized by Poker Academy without any pd file error.
Inputs
Newly added agent into bots folder of Poker Academy which is also added as an opponent
Outcomes
A player in the game table
Environmental Needs
Poker Academy
Special Procedural Requirements
Agent should be introduced Poker Academy via Opponent Manager interface.
Intercase Dependency
This test depends on the success of ST1 and ST2.

Table 17: ST3 - Adding Agent into Game Table

4.3.4 ST4 - Response Test

Objectives
This test case is required to test whether agent is able to get information from Poker Academy and send next decided action for itself.
Inputs
Game state
Outcomes
An observed action of agent
Environmental Needs
Poker Academy, Meerkat API 1.5 and JavaBayes 0.346 are required.
Special Procedural Requirements
Meerkat 1.5 and JavaBayes 0.346 should be added to Poker Academy as external libraries.
Intercase Dependency
This test depends on the success of all unit, integration and previous system test cases.

Table 18: ST4 - Response Test

4.4 VALIDATION TESTS

4.4.1 VT1 - Rationality Test

Objectives
This test case aims to verify and validate that poker agent is able to make rational decisions in the current game state.
Inputs
Game state
Outcomes
An observed rational action of agent
Environmental Needs
Poker Academy, Meerkat API 1.5 and JavaBayes 0.346 are required.
Special Procedural Requirements
Meerkat API 1.5 and JavaBayes 0.346 should be added to Poker Academy as external libraries. Agent should be integrated into Poker Academy and added as a player into game table.
Intercase Dependency
This test depends on the success of all unit, integration and system test cases.

Table 19: VT1 - Rationality Test

4.5 PERFORMANCE TEST

This test case aims to verify that agent makes rational decisions within an acceptable amount of time. Also the memory amount used by agent within a game should be checked against limits.

5 DETAILS FOR SYSTEM PROCEDURE

5.1 INPUTS, OUTPUTS AND SPECIAL REQUIREMENTS

PPA will be tested in the Poker Academy as a third party application that works on Windows OS. This application will provide necessary information about current game through the Meerkat API 2.5 library. Therefore tests needing this information depends on this library. Since hand abstraction, hand similarity and hand evaluation in pre-flop stage in game are implemented using old version of this API, Meerkat API 1.5 is also needed as external library in test environment.

Bayesian Network that is used in both decision mechanism and opponent modelling is constructed via JavaBayes 0.346. Tests that related to creation and update of Bayesian Network and getting results from this network depend on JavaBayes 0.346 as an external library.

Since persistence of game state with decided action and opponent behaviour is necessary for the update of Bayesian Network MySQL 5.6 will be used as database server.

Some of the tests requires the knowledge of basic poker rules including determination of hand strength and interpretation of opponent behaviour. Furthermore, without this knowledge, it cannot be observed whether PPA makes rational decisions during the game or not.

Testing of the PPA project will be done manually, any test automation tools will not be used.

5.2 ORDERED DESCRIPTION OF THE STEPS TO BE TAKEN TO EXECUTE THE TEST CASE

Procedure steps taken during the execution of the test cases are listed below.

- **Setup:** Poker academy and necessary external libraries must be supplied. JDK 5 must be installed.
- **Start:** Test files should be organized together according to which tests wanted to be run. Poker Academy should be run if related test case does not include code execution but observation.

- **Proceed:** Compiled test files should be run in JDK 5 environment for the test cases including code execution and a game table must be created to observe PPA in Poker Academy.
- **Measurement:** For the observation of PPA statistical data provided by Poker Academy will be used such as win percentage of agent.
- **Shut down:** When game is finished while behaviour of PPA is observed, test is temporarily suspended.
- **Restart:** When game is finished during the observation of PPA, bankrolls should be reset to start a new game and continue testing.
- **Stop:** Tests which require code execution will not be automated therefore there is no need an extra action to stop testing. However if related test is done in Poker Academy abortion of current hand / game is needed.
- **Wrap-up:** Tests which does not require any action after execution however to analyze behaviour of PPA statistical data taken from Poker Academy must be examined.
- **Contingencies:** If PPA is not integrated into Poker Academy correctly, pd and jar file should be revised. If agent does not respond during the game code inspection should be conducted.

6 SYSTEM TEST REPORT DETAILS

6.1 OVERVIEW OF TEST RESULTS

In this section of the document, test cases that are explained in section 4. Test Case Details are concluded with their results according to pass/fail criteria that is also defined in the document at section 2.6 Item pass/fail criteria. Results of the all test cases are given in following subsection.

6.2 DETAILED TEST RESULTS

For the testing period of PPA, 18 different test cases are created. 13 of them passed successfully, while 3 of them are not executed since implementation of related features are not complete yet. 1 case failed to meet the expected outcomes since project needs further performance improvements and 1 case got blocked due to real evaluation of this test will only be possible after the failed test passed.

Test Case ID	Status
UT1	Pass
UT2	Pass
UT3	Pass
UT4	Pass
UT5	Pass
UT6	Pass
UT7	Pass
UT8	NoRun
UT9	NoRun
IT1	Pass
IT2	Pass
IT3	NoRun
ST1	Pass
ST2	Pass
ST3	Pass
ST4	Pass
VT1	Failed
PT	Blocked

Table 20: Test Cases with Their Current Status

7 GENERAL

7.1 QUALITY ASSURANCE PROCEDURES

All units in PPA are tested separately and with using black-box testing approach. Therefore only outcomes of unit tests are examined during testing rather than the contents of units and how the unit is implemented. Any anomaly that occurs is detected with using these outcomes directly.

All tests are prepared with the consideration of extreme cases which lead possible failure in units.

7.2 METRICS

Win percentage of PPA will be a metric for the evaluation during testing phase. However, while comparing PPA with other agents in literature or human players small betting unit per hands will be also used for getting more precise information about success level of our agent. Also, response time will be used as a measurement in the testing process.

7.3 TEST COVERAGE

Since the implementation of all components of PPA except opponent modelling and persistence layer is finalized, test cases that excludes these two components are covered. This corresponds to 75% of the test cases. Rest %25 of the test cases are in NoRun and blocked state.

7.4 GLOSSARY

All of the definitions, terms and abbreviations can be found in section 1.4 Definitions, Terms and Abbreviations.

7.5 DOCUMENT CHANGE PROCEDURES AND HISTORY

Record of Changes table can be found in page 2.