# Magenic Test Automation Framework

- Raja Gollamudi

Magenic®

# Agenda

- About Magenic
- Traditional automation approach
- Framework based automation approach
- Types of frameworks
- Magenic Automation Framework
- Demo
- Implementation phases

Magenic®

# Magenic Service Offerings

Since 1995, we've been building innovative custom solutions specifically designed to meet the unique needs of America's most recognized companies.

Application Development

User Design by Magenic Studios

Quality Assurance & Testing

Support & Maintenance

**Optimized Service Offerings**

Magenic®

# Magenic Contacts


Jeff White

Quality Assurance and Testing Director

Tony Mohl

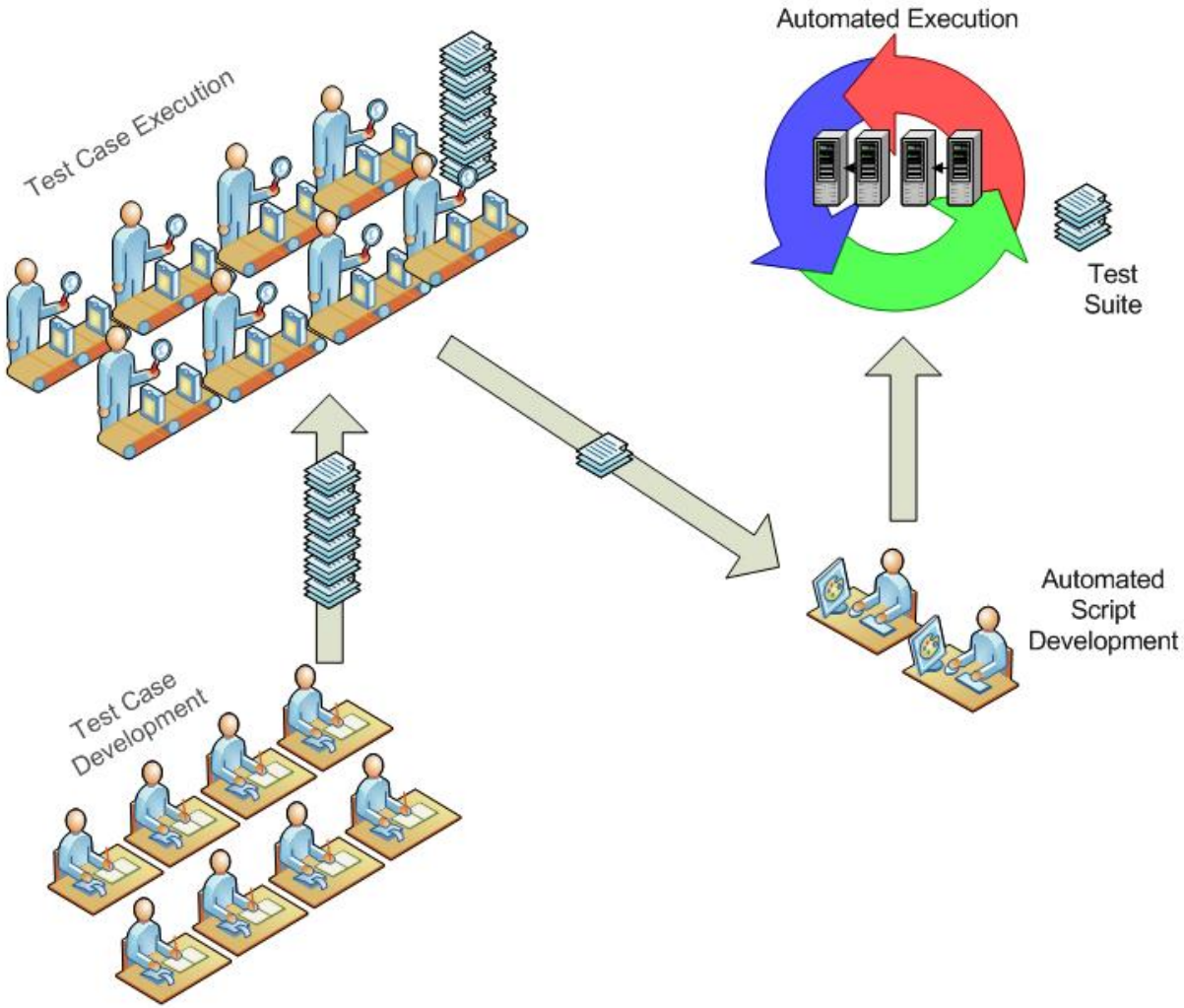
Delivery Center General Manager


Magenic®

# Typical Automation Formula

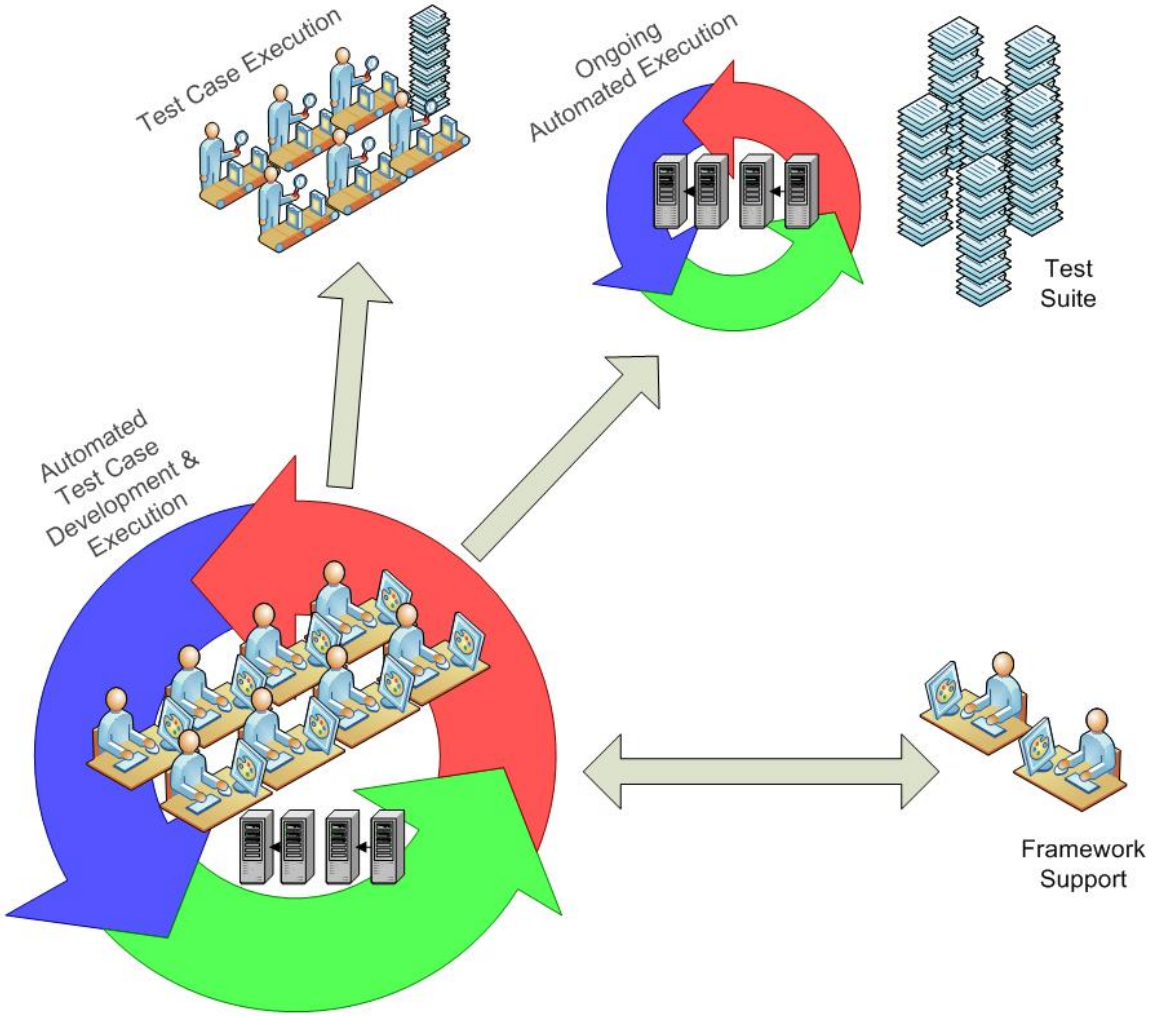
1. Purchase an expensive GUI test execution tool. (HP, Borland, Compuware etc.)
2. Define a lot of paper test procedures.
3. Hire an automation team to automate each one.
4. Build a comprehensive test library.
5. Keep fixing it.

# Traditional automation approach




Magenic®

# Framework based approach




Magenic®

# Types of Automation Frameworks

- Modular Test Scripts
- Test Libraries
- Data-Driven Testing
- Keyword-Driven/Table-Driven Testing
- Hybrid Test Automation

# Modular Test Scripts

- Begin with creating small, independent scripts that represent modules, sections, and functions of the AUT.
- combine the small scripts in a hierarchical fashion to construct larger tests.
- Modular framework yields a higher degree of modularization and improves overall maintainability.

# Test Library

- Very similar to the Modular test script framework and offers the same advantages.
- Divides the AUT into procedures and functions (or objects and methods) instead of scripts.
- Requires creation of library files that represent modules, sections, and functions of the AUT.
- These library files are called directly from the test case script.
- This framework also yields a high degree of modularity and overall maintainability of the tests.

# Data Driven Testing

- Test input and output values are read from database/files (ODBC sources, CSV/Excel files, DAO/ADO objects, etc.) and stored as script variables.
- Variables are used for both input and output.
- Navigation, reading the data files, and logging test status are all coded in the test script.
- In data-driven testing, only test data is contained in data files.

# Advantages of Data-Driven Testing

- Scripts may be developed while application development is still in progress
- Utilizing a modular design, and using files for both input & data verification , reduces redundancy and duplication of effort in creating automated test scripts
- When functionality changes, only the specific "Business Function" script needs to be updated
- Data input/output and expected results are easily maintainable
- Functions return "TRUE" or "FALSE" values to the calling script, allowing effective error handling and robustness

# Disadvantages of Data-Driven testing

- Requires proficiency in the Scripting language used by the tool (technical personnel).
- Multiple data-files required for each Test Case. There may be a many number of data-inputs and verifications required, depending on the number of screens accessed. This usually requires data-files to be kept in separate directories/database by Test Case.
- Apart from maintaining the Detailed Test Plan with specific data, testers may also need to re-enter data in all the required data-files.
- Script-processing errors occur due to incorrect format and/or content being incorrect.

# Keyword/Table driven Testing

- Independent of the test automation tool used to execute them and the test script code that "drives" the AUT and the data.
- Keyword-driven tests look very similar to manual test cases.
- Functionality of the AUT is documented in a table along with step-by-step instructions for each test.
- Entire process is data-driven, including functionality.

# Keyword/Table driven testing - Example

- In order to open a window, the following table is devised, and it can be used for any other application, just by changing the window name.

**Test Table for Opening a Window**

| Window | Control | Action | Arguments |
|---|---|---|---|
| Window Name | Menu | Click | File, Open |
| Window Name | Menu | Click | Close |
| Window Name | Pushbutton | Click | Folder Name |
| Window Name | | Verify | Results |

- A driver script reads in each step and executes it based on the keyword contained the Action field, performs error checking, and logs any relevant information

# Advantages of keyword driven testing

- The Detailed Test Plan can be written in spreadsheet format containing all input and verification data.
- “Utility" scripts are created by someone proficient in the automated tool’s scripting language
- Tester use the tool via the "spreadsheet-input" method; no need to learn the scripting language.
- Testers need to learn the required "Key Words" and format to use within the Test Plan. This allows the tester to be productive with automation very quickly.

# Disadvantages of keyword driven testing

- Development of "customized" (Application-Specific) Functions and Utilities require proficiency in the tool's Scripting language.

  (This is true for any method)

- If application requires handful of "customized" utilities, testers need to learn a number of "Key Words" and special formats.

  This can be time-consuming, confusing.

# Hybrid Method

- Combination of all the methods discussed. Mostly Keyword driven and Data driven
- Architecture involves storing the steps, data and validations in database
- Allows tool agnostic framework

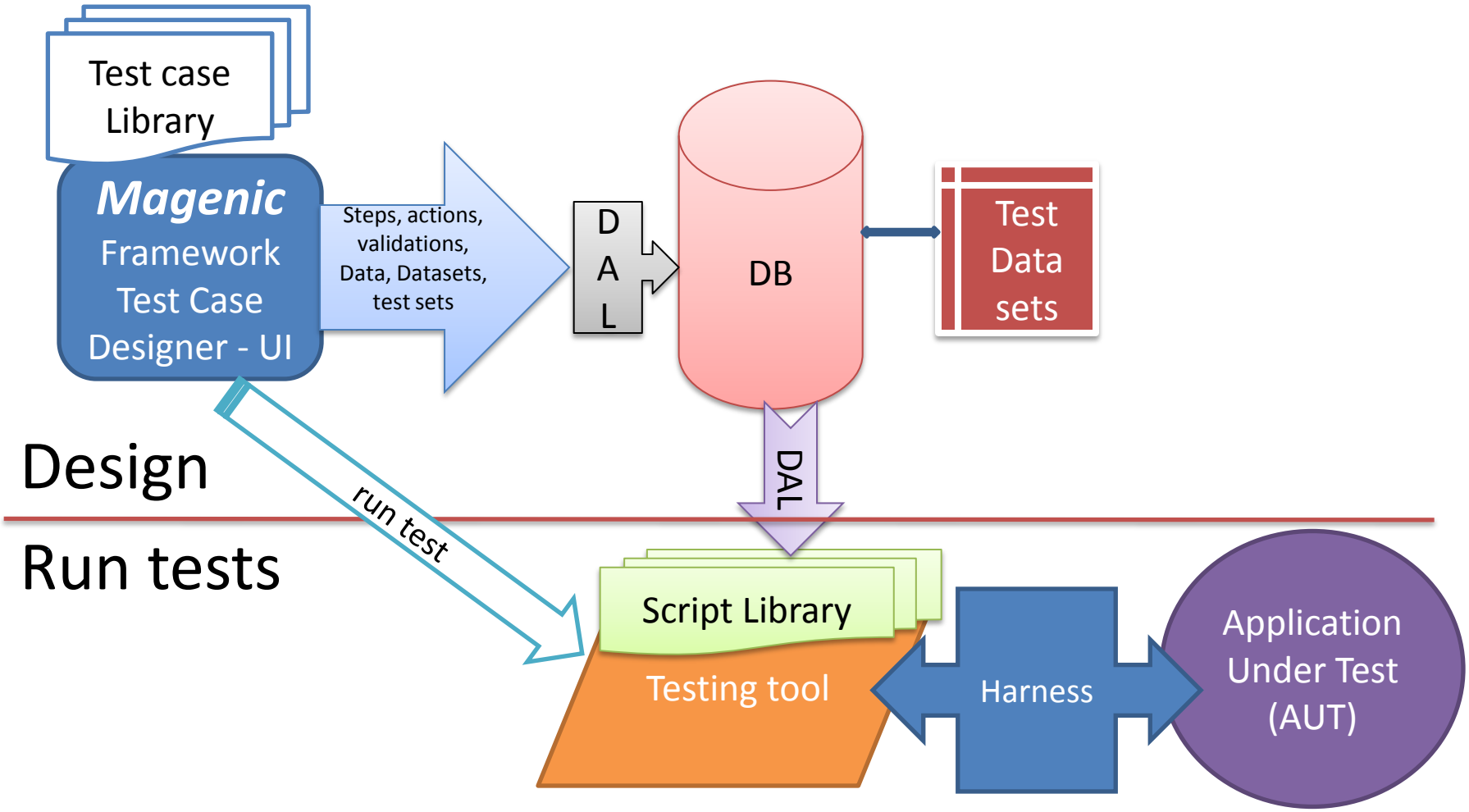
- Magenic Automation is Hybrid Framework

Magenic®

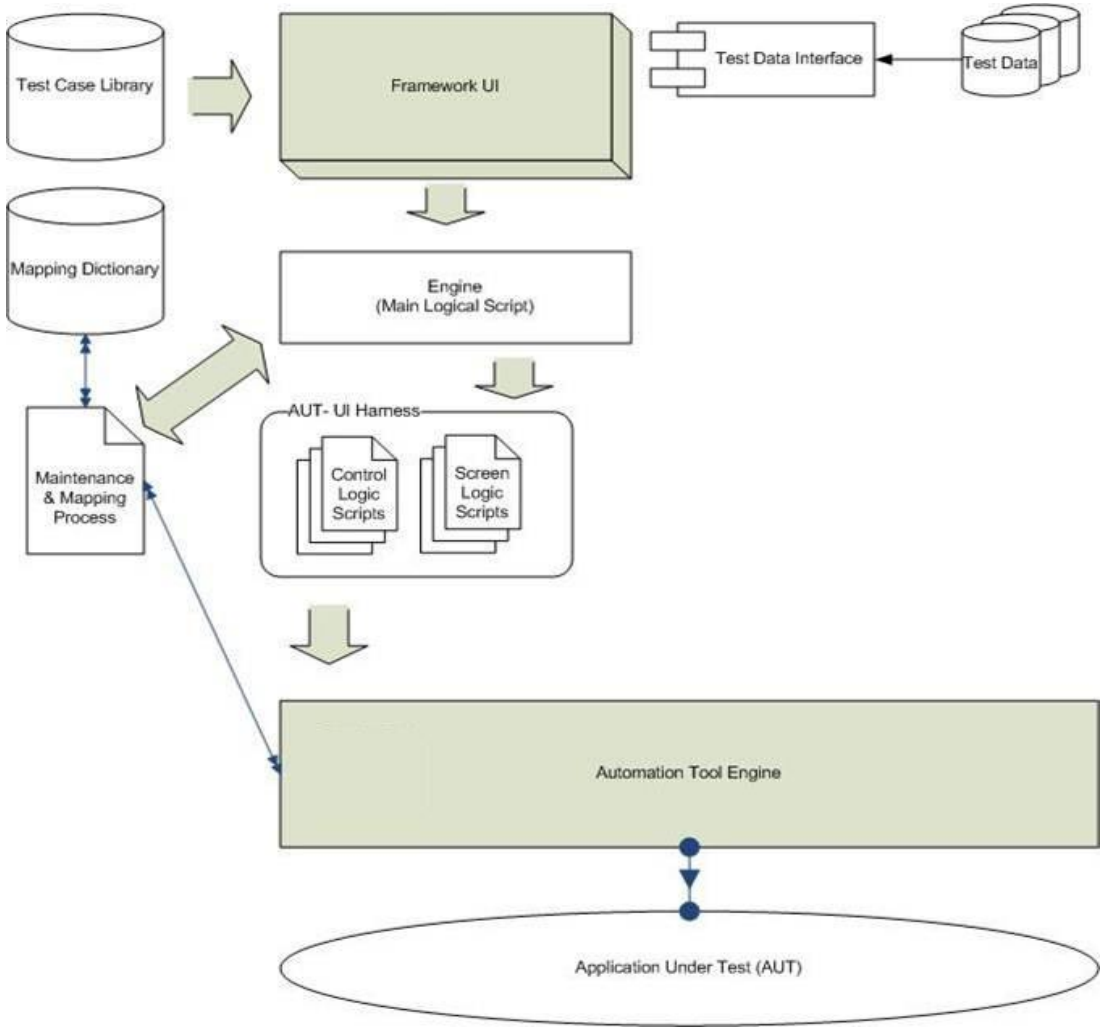
# Magenic Framework Objectives

- Provide a Tool Agnostic Framework
- Use Data Driven Scripts
- Define an Automation Independent Test Script Vocabulary
- Implement Reusable and Manageable Test Scripts
- Integrate Framework with Source Control
- Support for Execution across Code Branches and Multiple Environments
- Support Localization testing
- Provide Test Management and Productivity Utilities

# Magenic Framework Approach



# Automation Framework Design




Magenic®

# Test Case

- Pre-conditions
- Input
- Process or action
- Output verification pass/fail criteria
- Post conditions

  – Test case is a set of steps in a sequence

# Test steps

- Event
- Objects/Controls
- Action
- Verify
- Validate

Magenic®

# Step Types

- Event
- Verify
- Wait
  - A. For specified time
  - B. Specific event
  - C. until one of the above happens
- Run other test
  - Optionally store values and pass on to other tests.
- Conditional
  - If do this, else do other

Magenic®

# Scripts

- ‘Event’ and ‘Verify‘ step types involve selecting a control
- Each control has a set of Actions/Properties to choose from, based on its type.
- A function for each action that can be performed on that control type
- The scripts find the correct function for the step, and perform the action on the control

# Test tool Scripts

- Driver Script
- Control Mapping Script
- Database / dataset access scripts
- Library Scripts
- Action Scripts
- Validation Scripts
- Test management scripts
- Reporting scripts

Magenic®

# Demo

# Magenic Framework Advantages

- Automation Framework is re-usable
  - Implemented with base framework and database components
  - Customized components to support Clients' tools and process
  - Customized components for Application Under Test
  - Tool Agnostic Framework
  - Data Driven Scripts with Keyword driven approach
  - Intuitive test case designer and maintenance tool
- Multi browser compatibility for web based applications
  - Same test cases executed on multiple browsers
- Localization support
  - Framework supports multiple locales

Magenic®

# Implementation Phases

- Initiation (4-6 weeks)
  - Understand the application functionality
  - Analyze test cases
  - Derive a strategy
  - Implement base framework
  - Create sanity tests
- Customization (6-10 weeks)
  - Tailored functions
  - Interface with build management
  - Interface with test management
  - Customized reporting
  - Creation of tests
- Training, and test case entry (6-10 weeks)
  - Test team training
  - Maintenance handover
  - Creation of additional tests

Magenic®

# Questions

Magenic®

# Contacts


- Raja Gollamudi

  RajaG@magenic.com

- Jeff White

  JeffW@magenic.com

- Tony Mohl

  TonyM@magenic.com


Magenic®