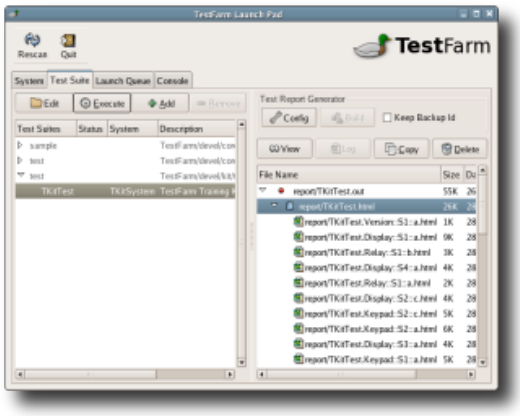
TestFarm Core
Automated Testing System Platform
for Real-Time, Embedded Software

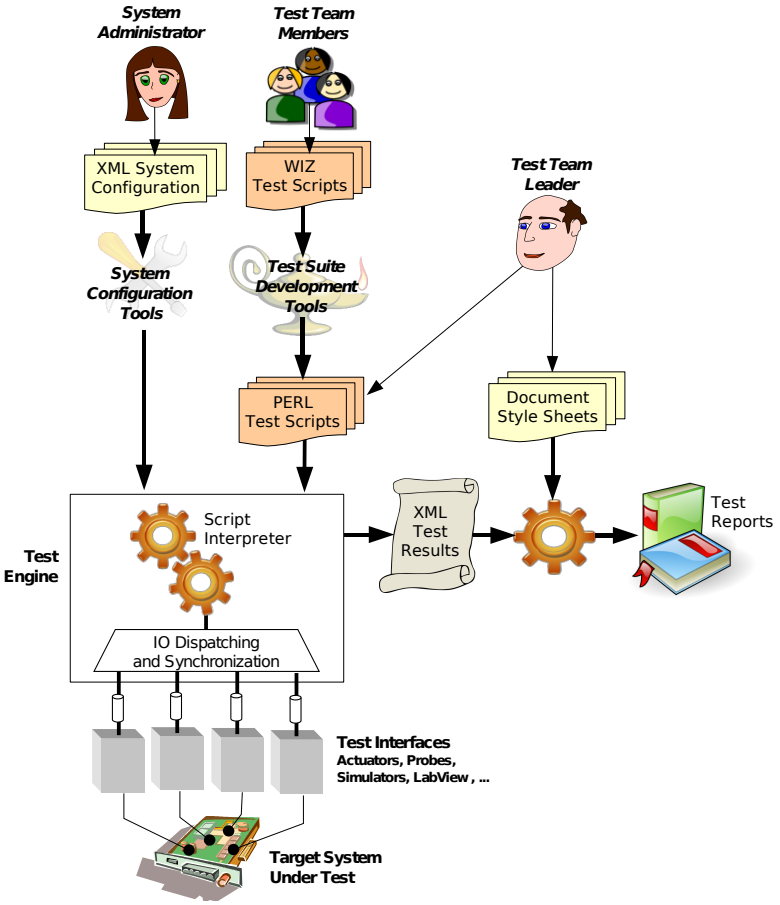
- TestFarm is a software environment targeted at performing non-intrusive (*black-box*) functionnal automated testing of **Real-Time Embedded Software**.
- **Open System**. The target SUT (System Under Test) is tested through an unlimited park of heterogeneous interfaces and instruments.
- **Open Standards**. Test Scripts are written in the widely spread PERL language. Test Results reporting is based on XML.
- **People Aware**. The TestFarm platform follows a strong Test Suite development methodology to efficiently use human skills and preserve teams motivation.

Overview

TestFarm is especially designed for testing **Embedded Systems** that use various standard or home made peripherals, in a heavily **Asynchronous** environment. Such systems are for instance payment terminals, which software has to interact with human actions (keypad, LCD, buttons, ...) while communicating to other system components (supervision servers, smart-cards,) etc.

The TestFarm architecture is sufficiently open and **flexible** to be targeted at other environments such as Graphical User Interfaces or any application running on a workstation, but its main strength resides in the management of real-time and asynchronous events.





The Testing Model

Software Focused, Software Agnostic

The TestFarm testing model conforms to a **Non-structural**, **Non-Intrusive** black-box approach. This means that the Testing System does not have to be aware of the internal structure of the target software being tested.

Whatever the OS, the programming language, the software architecture of the product under test, the Testing System is capable of stimulating and analysing the tested software through external interfacing. On an embedded system, this means to interface the target hardware. On a workstation, this means to have access to user interfaces and network interfaces. This gives the possibility to perform non-intrusive testing, without instrumenting the target software, thus allowing to really validate the software release that will be delivered to the final customer.

Convergence to Simplicity

The TestFarm model **Hides the Complexity** and **Preserves the Confidentiality** of the software internal design by focusing on functional aspects and use cases, rather than on structural aspects.

Target System Interfacing

Heterogeneous and Open Interfacing

The TestFarm platform provides flexibility and extensibility for connecting a heterogeneous collection of Test Interfaces. A Test Interface could be a measurement instrument, a software or hardware component, or even a complex LabView subsystem. Connected to the System Under Test, a Test Interface applies stimuli and gets reactions from it. This information are dispatched and synchronized by the Test Engine.

System Configuration

At the system configuration level, TestFarm administrators define the system interface and features in a XML file, from which system documentation and libraries are automatically generated. The configuration tool automatically generates an Interface Library, which plays the role of an hardware abstraction layer for the Test Script writers.

Manual Interface

As an additional convenience tool, the Test Interfaces can be used manually from a Graphical User Interface generated using the standard *Glade Interface Designer*. The so-generated graphical manual interface is fully integrated into the Test Execution environment, so that it can be used either as a diagnostic tool or as a Test Script development assistant.

TestFarm provides a target-specific Manual Interface

Test Suites Development

Structured Test Suite

The Test Scripts are gathered into a structured test tree, following the organization of the System Under Test features. After the Test Suite is executed, this structure is respected in the generated Test Report.

PERL language: versatility

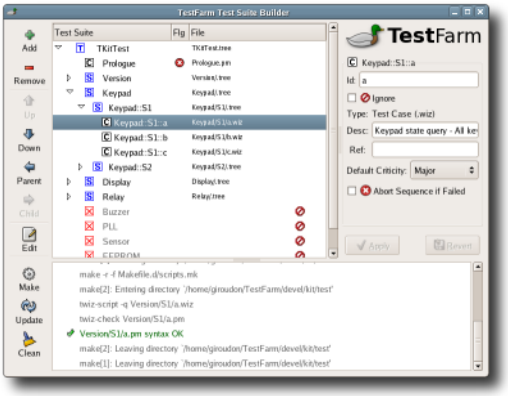
Test Suites are constructed with test scripts written in the well known and reputed **PERL** language. The TestFarm environment provides a rich collection of libraries dedicated to that purpose. No proprietary language is used for writing the test scripts. This brings all the power of PERL to the Test System, while avoiding to learn yet-another-proprietary language. For people who do not know this language, learning PERL is very easy and profitable, given the huge amount of resources available in the worldwide software and IT community.

WIZ macro-language: simplicity and reusability

The WIZ macro-language allows to automatically generate PERL test scripts while following a clear Test Suite development framework. Its structure is purely dedicated to writing Test Scripts. It provides a high rate of code reusability, and a high level of abstraction of test operations.

The WIZ macro-language requires the script writer to respect strong coding and syntax rules. It also performs automatic generation of macro documentations.

The WIZ macro-language allows people who have limited knowledge (and interest) in PERL programming to easily write Test Scripts. It also simplifies the task of Test Scripts code review, which is a key stage for validating the Test Suites.



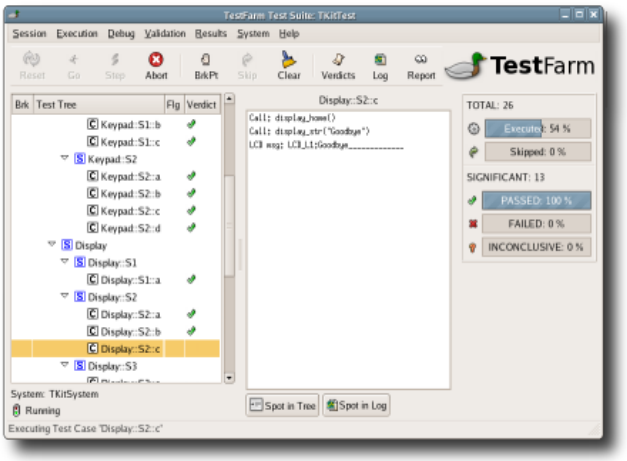
The TestFarm Builder allows to edit the Test Suite structure and to compile WIZ scripts.

Test Suites Execution

Execution Control

The TestFarm execution environment is based on the standard PERL interpreter, with full control over the Test Suite structure (progress indication, breakpoints management, execution resume and abort, ...). As an option, a PERL debugger interface may be launched to control execution at a fine-grain level.

As a diagnostic tool, the Manual Interface may still be used while the Test Suite execution is in progress. This helps quickly solving issues when developing test scripts or maintaining the TestFarm System.



The Test Suite execution environment

Quality Management

TestFarm provides some tools for easy integration into its owner's Quality System.

When a test Suite execution is started, some key information are collected in order to produce a precise Test Report: Operator name, identification of Test Report release, identification of System Under Test components, identification of Testing System components, etc.

A Test Suite validation tool allows to mark approved test scripts in order to detect if they have been altered. Moreover, it is highly recommended to use a Revision Control System to store and track the Test Suite developments.

Test Report Generation

During test suite execution, test results are stored in XML format. Test results include both data exchanged with the test interfaces and messages printed by the test scripts.

Test reports are automatically generated from the XML test results using customizable style sheets in standard XSL format. Default format for generated Test Reports is HTML. It contains several levels of details on the test results: from global statistics to detailed statistics and detailed information on what happened during the execution of the test suite.

Operating system

The TestFarm Core automated testing system runs on Fedora Core Linux Distributions. The software modules are fully integrated into the Gnome Desktop environment.

The open nature of the Linux operating system make it well suited for a highly technical application like TestFarm: modularity, security, robustness, efficient usage of computer resources, ease of maintenance and diagnosis, ease of interconnection with other systems.

TestFarm can be used remotely with a Graphical User Interface from any other platform (Unix, MS-Windows, ...), by using the native Linux implementation of the X Window system. It is even possible to access it through a broadband Internet connection (VPN secure tunnel recommended).

Ordering Information

TestFarm Core
Automated Testing System for Embedded Software

GOOBIE
23 avenue Louis Bréguet
78140 Vélizy-Villacoublay
FRANCE

www.testfarm.org
www.goobie.fr