Automatic Testing of software components for ATC

Gabriella Carrozza, Vittorio Manetti, Luigi Martin Petrella
SESM s.c.a.r.l.
Via Circumvallazione esterna di Napoli, Giugliano in Campania, 80014, Naples, Italy
Email: gcarrozza,vmanetti,mpetrella@sesm.it

***Abstract*—The testing of software components is one of the most expensive phases in the software development cycle, either in terms of time as well as human effort. This is particularly true for safety critical systems, for which making the execution of test cases automatic allows reducing costs and improving software quality from a dependability point of view. In this paper we present a framework for the automation of testing procedures for complex software systems with strict safety and quality requirement, actually used in Air Traffic Control applications.**

I. Introduction

Highly dependable software systems require intensive testing campaigns, aimed to verify the functional aspects of the produced code, as well as non functional requirements which impact on software reliability significantly. Performing software testing in such complex systems, not only in size and complexity but also in terms of frequent changes and daily releases, is not a trivial task especially when budget and time constraints have to be respected.
Notwithstanding the clear benefits that automation strategies and tools can bring, many companies find it difficult to integrate testing automation into their processes due to the:

- high costs of the start up phases;
- the need for highly skilled personnel in charge of preparing testing environment and developing testing procedures to be run automatically.

In this work we introduce a framework for automatic testing of software components belonging to ASTERIX-based surveillance systems.
The ASTERIX protocol has been developed and standardized by the European Organisation for the Safety of Air Navigation (EUROCONTROL) with the aim to ease the exchange of surveillance information between and within countries. The main users of such a Standard are the Air Traffic Control (ATC) Centers: today almost all ECAC States are using this data format in their ATC Centers.
Designed according to the black-box approach, the framework we propose is capable of implementing automatic testing of software components which are compliant with the ASTERIX Standard. The framework allows to perform the execution of a whole testing campaign requiring zero human work, obtaining in such a way the following significant goals:

- reduce costs in terms of human resources and time consuming;
- prevent not negligible errors that may occur when test procedures and checks are completely human made .

The success of each single test case is determined through the comparison between the expected and the observed behavior of the System Under Test, where the expected behavior is a function of the system state at time 0 and the test environment (both coded in a XML file) on one hand, and of the well-known application logic implemented by the SUT on the other.
The rest of this paper is organized as follows. Section II briefly presents the main features of an ATC system and introduces the ASTERIX Protocol, while Section III illustrates a detailed description of the architecture we propose. Section IV describes the real world surveillance system representing the case study we used for the architecture validation, namely the Multilateration System, and Section V proposes some consideration about the outcomes we get from the experimental campaign, as well as future works.

II. ATC Systems and the Asterix Protocol

Air Traffic Control (ATC) systems are composed of cooperating computers hosting applications that deal with surveillance and flight data (Flight Data Processing, Surveillance Data Processing), and auxiliary services (Medium Term Conict Detection, Recording and Playback, etc.) as well. Central units are also connected to other external systems for transmission/reception of ATC significant information, and to controller working positions (CWPs) that are used to provide a view of the environment scenario as well as of current and planned data. ATC centers belonging to the same system are often deployed over different cities in a given country, and pre-operational platforms can be spread over several company premises.

Since the Air Traffic volume is continuously increasing and a high level of safety must be maintained, surveillance mechanisms for ATC systems are always under constant evolution. New-generation surveillance technologies are developed respecting the need to cohabit with current systems, and considering that the information they generate must be transmitted in a harmonized and efficient way.
Up to thirty years ago, every National Administration developed its own format for delivering radar data to Air Traffic Control Centers. This implied a duplicate effort and made the exchange of radar data across borders an issue, so, the need for a common European data format became apparent.
ASTERIX (All purpose Structured Eurocontrol SuRveillance Information eXchange) [6] is an ATM surveillance data binary messaging format which allows transmission of harmonized information among surveillance and automation systems. It

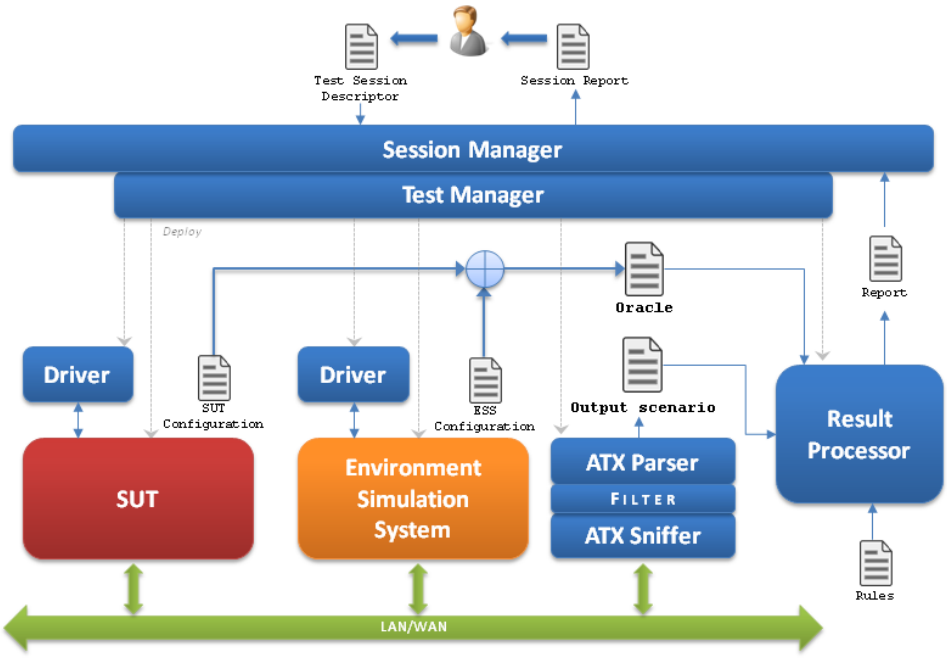

Fig. 1. View of the Porposed Architecture

defines the structure of the data to be exchanged over a communication medium, from the encoding of every bit of information up to the organization of the data within a block of data, without any loss of information during the whole process. Concerning the ISO/OSI Standard, ASTERIX refers to the Presentation and Application layers; transmission of ASTERIX coded surveillance information can make use of any available communication medium. Aimed at simplifying the data exchange among heterogeneous applications, ASTERIX specifies minimum requirements at the Application level, making it possible the communication between two different systems (even located in different countries) which is based on a core of commonly used surveillance related data transferred by the ASTERIX Presentation layer.

III. Design of the Proposed Architecture

The framework architecture we propose consists in a number of software modules that handle all the stages of test cases execution, namely:

I. Deployment and initialization of all software components that are needed to build the testing environment, i.e. the System Under Test (SUT) and the Environment Simulation System (ESS) which provides a mock up for the real environment that interact with the SUT (e.g. hw devices, sw agents);
II. Control and monitoring of the testing scenario during execution;
III. Capture and analysis of results, validation of the test case.

This allows the execution of multiple tests organized in test suites, with reduced or zero human work. In fact, interaction with an operator is only required before the start of the test session and subsequently when it is finished, providing a sort of bridge on execution: no matter how long does it takes to run the test suite and what happened during test case, operator is relieved on being present.
The entry point for the framework is the Session Manager component, which is the only component that interact with the operator.
The main framework input is a test session descriptor, which is an XML document containing:

I. Information about components to be deployed, including specific configuration options;
II. Required behaviour and logic state of SUT and ESS during test execution.

For every test case, the Session Manager instantiates a Test Manager: according to required and disposable hardware resources, we can start one or more test cases at the same time. The Test Manager deploys SUT, ESS and all other framework components as depicted in Fig. 1.
Both SUT and ESS are controlled by DRIVER modules which replace interaction with human operator: they change SUT and ESS behaviour, and their operational states by sending appropriate timed commands as specified in the test session description. Drivers can interact with SUT/ESS in two ways: via their front-end (if one) e.g. command shell, or remotely by sending commands through network using backdoors.
Since we are focusing on ASTERIX based systems, the main communication infrastructure is a LAN/WAN: this peculiarity can be exploited to capture systems output simply by sniffing the network. For this purpose, an ASTERIX Sniffer is deployed: this component listens over LAN/WAN during all the

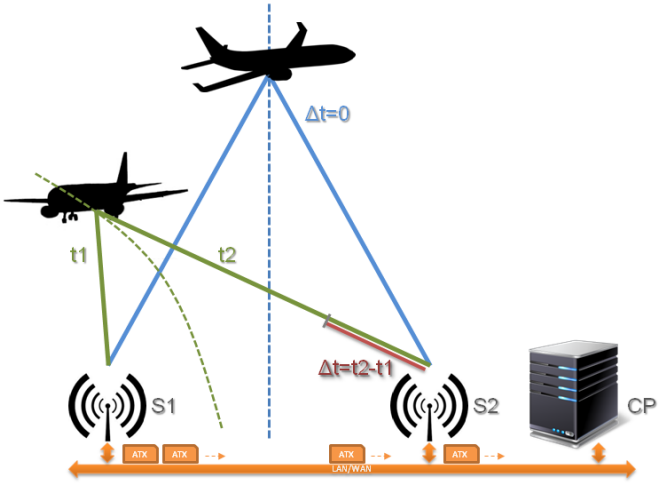

Fig. 2. Multilateration System

test case and catches all ASTERIX messages. These messages are filtered to drop all irrelevant ASTERIX categories for the test, and then are passed to the ATX Parser which provides a synthetic representation of the observed scenario. This information is combined with the test case oracle, which is obtained from two kind of information:

a) the ESS configuration, which set WHAT is the scenario that will be simulated, and so is expected to be perceived by SUT
b) the SUT configuration, which specifies HOW the perceived scenarios shall be interpreted

The Result Processor component compares expected and observed results applying specific rules to establish if the distance between them is acceptable or not, and to establish the overall result for the test case (e.g. the distance between expected and observed variable is beneath some threshold). The Result Processor generates a report containing test results and push it to the Session Manager which combines all report to generate a human readable Session Report that contains:

I. Single test case results;
II. Error and significant event logs;
III. Overall test session outcome.

IV. Case Study: Multilateration

An experimental session to validate the performance of the proposed framework has been implemented using a Multilateration system (MLAT) [5] as use case (Fig. 2), since this kind of ATC system relies upon the ASTERIX standard.
Multilateration is a co-operative independent surveillance technology: it makes use of signals transmitted by an aircraft to calculate the aircrafts position.
MLAT is an enabling technology that enhances the provision of ATM in a variety of applications, from radar-like air traffic control purposes to enhanced situational awareness of surface movements, and can be combined with other surveillance systems such as radar and ADS-B, to improve the total surveillance picture.

The processing of aircrafts signals on the ground requires a number of elements, so a complete MLAT system consists of the following components:

- A transmitting subsystem that includes interrogation message generation and transmission function;
- An optional Intelligent Interrogation process that determines whether an MLAT interrogation is required;
- A receiving antenna array subsystem that receives the transmissions from the target and timestamps receipt at each antenna;
- A Central Processor (CP) that calculates and outputs the MLAT tracks from the time difference of arrival (TDOA) of the signal at the different antennas.

The TDOA between two antennas corresponds, mathematically speaking, with a hyperboloid (in 3D) on which the aircraft is located. When four antennas detect the aircrafts signal, it is possible to estimate the 3D-position of the aircraft by calculating the intersection of the resulting hyperbolas. When only three antennas are available, a 3D-position cannot be estimated directly, but if the target altitude is known from another source, then the target position can be calculated. This is usually referred to as a 2D solution. With more than four antennas, the extra information can be used to either verify the correctness of the other measurements or to calculate an average position from all measurements which should have an overall smaller error.

All messages between targets, ground stations and CP, are exchanged using the ASTERIX data structure, and the CP implementation at our disposal was shipped with its own Sensor Simulator module, which provides a mock up of the complete ground environment (receivers, transmitters, communication infrastructure). Therefore, the CP component is suitable to be tested using the proposed framework after an appropriate tuning action.

First of all, we customized the test session descriptor adding these information for every test case:

I. Test duration;
II. Type of verifications to perform;
III. Command to be executed on the CP and the Sensor Simulator;
IV. Rules and thresholds to be applied by the Result Processor.

Given that natively both the CP and the Sensor Simulator were capable to receive runtime commands only by shell, we decided to opened backdoor exploiting some communication modules without compromising reliability and performance, but allowing the respective DRIVER to interface with them via LAN through an UDP channel and enabling remote control.
Test case were focused on the detection of the aircraft position, therefore we considered ASTERIX category 010 and 020, and filtering policy were configured to drop all other messages. Moreover, we provided custom parser to extract and store the ASTERIX items involved in the evaluation, i.e. 010/042-091-140-250 and 020/042-140-250.
Configuration files for the CP and the Sensor Simulator include

information respectively about multilateration type and options (e.g. 2D or 3D multilateration), and information about the simulated scenario, i.e. targets definition and simulated position; combining these information we get detailed indication about the expected outcome of the multilateration process performed by the CP. Concerning the results processing, the following two rules are applied:

I. All and only simulated targets should be observed;
II. Defining error the mean difference between the simulated and the computed position of targets, such an error must be lower than a predefined threshold, that may vary depending on the considered test cases.

V. Conclusion and Future Works

A whole qualification test session for the CP component can be completely automated by using the framework we propose, after implementing a proper customization as we just outlined. Traditional testing procedures for the CP component require an operator to be present during each test execution in order to:

- run commands on CP and Sensor Simulator shell;
- observe the plot of calculated targets on auxiliary display terminal to verify its stability and correctness.

Furthermore, at the end of every test case the operator should inspect the output logs generated by the CP to validate the results.

It is quite clear that the implementation of traditional testing procedures is very expensive in terms of human resources and time consuming, and mostly important, it may be affected by human errors.

Results from the experimental session lead us to confirm that executing test campaigns with automatic tools allows to prevent not negligible errors that may occur when test procedures and checks are completely human made, improving the quality of the released software products. We claim that the framework allows to perform testing campaigns in half the time if compared to manual execution, and with a very reduced human effort (limited to start up and configuration phases). Last, but very important, is the chance to perform testing starting from the very early development phases easily, thus improving product quality even in terms of requirements and design. The framework is still under development for improving its performances. Its main feature lies into the fact that it is not linked to the specific SUT: rather, it can be easily customized for any ASTERIX based system.

References

[1] A. Cervantes, *Exploring the Use of a Test Automation Framework*, IEEEAC paper #1477, version 2, updated January 9, 2009

[2] L. Nagowah, and P. Roopnah,*AsT - A Simple Automated System Testing Tool*, IEEE, 978-1-4244-5540-9/10, 2010

[3] G. Galati M. Gasbarra P. Magaro P. Marco L. Mene and M. Pici, *New Approaches to Multilateration processing: analysis and field evaluation*. In 2006 European Radar Conference, volume 9, pages 116119. Ieee, Sept. 2006.

[4] G. Galati, M. Leonardi, P. Magar, and V. Paciucci (2005, October). *Wide area surveillance using SSR mode S multilateration: advantages and limitations*. In Radar Conference, 2005. EURAD 2005. European (pp. 225-229). IEEE.

[5] N. McFarlane, *Generic Safety Assessment for ATC Surveillance using Wide Area Multilateration*, Helios Technology. WAM Safety Study & Surveillance Generic Safety. Eurocontrol. Bob Darby. November 9, 2007.

[6] Eurocontrol, All Purpose Structured. *RADAR DATA EXCHANGE Part 1 All Purpose Structured Eurocontrol Radar Information Exchange (ASTERIX).*