# FITNESS: A Framework for Automatic Testing of ASTERIX Based Software Systems

Vittorio Manetti
SESM s.c.a.r.l.
Via Circumvallazione esterna di Napoli,
Giugliano in Campania - 80014
Naples, Italy
vmanetti@sesm.it

Luigi Martin Petrella
SESM s.c.a.r.l.
Via Circumvallazione esterna di Napoli,
Giugliano in Campania - 80014
Naples, Italy
mpetrella@sesm.it

## ABSTRACT

As applications are developed, functional tests ensure they continue to function as expected. Nowadays, functional testing is mostly done manually, with human testers verifying a system's functionality themselves, following hand-written instructions: this make testing of software components one of the most expensive phases in the software development cycle, either in terms of time as well as human effort. Concerning in particular safety critical systems, such as the ones belonging to the Air Traffic Management field, for which it is always necessary to be taken complete and rigorous security test and evaluation among development team and/or by third-party security certification organization, performing automatic tests on such systems become a very tricky process considering that the goal is to verify not only the proper functioning of the SUT, but the system dependability too. However, such software testing is usually time consuming, cost consuming and boresome and thus technologies of software testing automation have alluring application foreground in that field: making the execution of test cases automatic allows to reduce costs and to improve software quality from a dependability point of view. In this paper we present FITNESS, a framework for the automation of testing procedures for complex software systems with strict safety and quality requirements, and in particular we have focused on Air Traffic Control (ATC) application who rely on ASTERIX standard as data exchange format with the intent to propose a flexible solution to automate testing procedure for a generic system that use such communication standard. We also present a quantitative study that analyze the effectiveness of the proposed approach using our framework to test a Secondary Surveillance Radar system and showing that most of manual test steps can be automatically converted to automated test steps with no human intervention.

## Categories and Subject Descriptors

D.2.4 [**Software/Program Verification**]: Validation; D.2.5 [**Testing and Debugging**]: Testing tools

## General Terms

Verification

## Keywords

Validation, Testing tools

## 1. 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 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.

This is particularly true for Air Traffic Control (ATC) applications, where the considerable redundancy of safety and security audits results in very complex and expensive testing campaign involving a big number of test cases.
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 Conic 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 centres 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.

*JAMAICA'13*, July 15, 2013, Lugano, Switzerland
Copyright 2013 ACM 978-1-4503-2161-7/13/07...$15.00
http://dx.doi.org/10.1145/2489280.2489292

In this work we introduce FITNESS, a framework for automatic testing of software components that rely on the ASTERIX protocol, a messaging format for the exchange of surveillance information between and within countries. The proposed 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 behaviour of the System Under Test, where the expected behaviour is a function of the system state at time 0 and the test environment (both coded in a XML file) on one side, and of the well- known application logic implemented by the SUT on the other.

We realized a prototype version of the proposed framework trough which we performed an experimental campaign aimed at verifying the effectiveness of our tool, considering as case study a Secondary Surveillance Radar system.

The rest of this paper is organized as follows. Section II gives an overview about ATC software assurance standards and section III introduces the ASTERIX Protocol, while section IV illustrates a detailed description of the architecture we propose. Section V describes the real world surveillance system representing the case study we used for the architecture validation, namely the Multilateration System, and Section VI proposes some consideration about the outcomes we get from the experimental campaign. Section VII offers conclusion remarks.

## 2. STANDARDS FOR ATC SOFTWARE ASSURANCE

Regarding the dependability of ATC software many efforts have been made from the point of view of standardization: in response to the increased use of software in airborne systems, the Radio Technical Commission for Aeronautics association, now known as RTCA Ltd., created the guidance document DO-178 âĂIJSoftware Considerations in Airborne Systems and Equipment CertificationâĂİ which has come to be accepted as the international avionics certification standard for airborne software. The standard provides detailed guidelines for the production of all software for airborne systems and equipment. The ground based complement to the DO-178 airborne standard is RTCA DO-278 / EUROCAE ED-109 âĂIJGuidelines for Communication, Navigation, Surveillance and Air Traffic Management (CNS/ATM) Systems software Integrity AssuranceâĂİ: it is intended as an interpretive guide for the application of DO-178 guidance and provides guidelines for the assurance of software contained in non-airborne CNS/ATM systems defining a set of objectives that are recommended to establish assurance that airborne software has been reviewed, and in some cases, modified for application to non-airborne CNS/ATM systems. The standard is basically a formalized testing protocol that requires a very careful collection of all the requirements in a formalized form which allows you have to develop comprehensive tests that test out every requirement; the effort of producing those tests means the code is very carefully looked at, even if it can't mathematical proof the absence of any fault. Given the high complexity required by the standard to acquire its certification, tools for automatic testing are a vital part of software design process: static analysis software tools analyze source code to derive properties that can help detect errors that might not be apparent to the programmer, while dynamic analysis tools help show what code is executed by a test suite. Currently, research and development initiatives of testing automation software in ATC field are carried out by private specialized companies:some cases in point are the GNAT Pro Safety-Critical framework provided by AdaCore and the LDRA tool suite by LDRA ltd. They both consist of an environment for high-reliability/safety-critical embedded application that has to meet safety standards such as RTCA DO-278 on a native platform. The tool we propose in this work while not contemplating the achievement of DO-278 certification stands as a possible mean to support the achievement of higher levels of quality for ATX-based systems that can be a step towards compliance to the DO-278 standard. Some key concepts that we will introduce further for Fitness framework architecture (e.g. oracle definition, SUT drivers, result processing) are in common with some other test automation tools, but we'd like to underline that what we have accomplished in our work was to adapt these concepts in a specific context, beforehand reducing the gap between a general purpose solution for automated testing with respect to enabling technologies (i.e. ASTERIX protocol) for the family of ATC systems taken into account.

## 3. THE ASTERIX PROTOCOL

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 Centres. 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, leading to the definition of the ASTERIX protocol.

ASTERIX (All purpose Structured Eurocontrol Surveillance Information eXchange) [6] is an ATM surveillance data binary messaging format which 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) Centres: today almost all ECAC States are using this data format in their ATC Centres. ASTERIX allows transmission of harmonized information among surveillance and automation systems; it 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

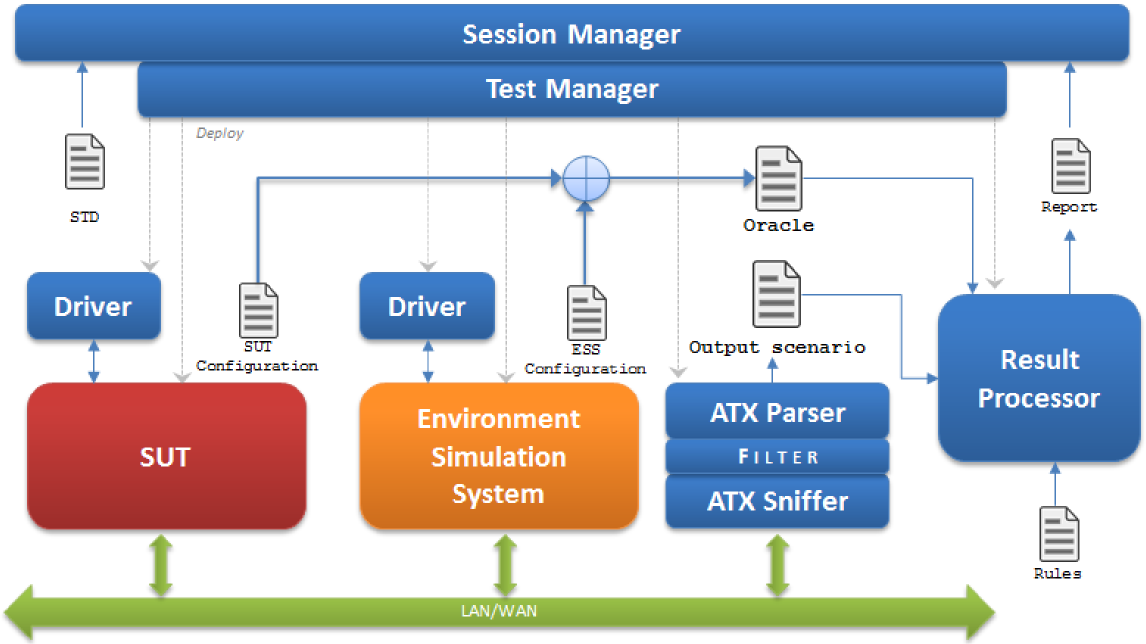


**Figure 1: View of the Porposed Architecture**

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.

## 4. THE FITNESS FRAMEWORK ARCHITECTURE

Aiming at improving the testing process for software components both concerning the cost reduction and the quality increase, we propose FITNESS, a framework for automatic testing of software components belonging to the Air Traffic Control field that rely on to the ASTERIX Standard defined by Eurocontrol.

The proposed framework allows to execute an entire testing campaign composed by multiple tests organized in test suites, completely avoiding the attendance and/or the intervention of a human operator. Interaction with a human operator is only required before starting the test session in order to properly configure the framework components and the SUT (e.g. HW devices and SW agents): no matter how long does it takes to run the test suite and what happens during test case, operator is relieved on being present.

The suitability of a software component to be tested trough our framework has to be verified considering the following constraints:

- it essentially performs processing of data;
- communication with other components occurs only through network protocols;
- it does not offer advanced user interfaces both for collecting inputs and for showing output data.

The FITNESS Architecture consists in a number of software modules that handle all the stages of test cases execution, from the launch of the System Under Test (SUT) components and accessories, to the presentation of results.

Knowing the application context, we are quite sure to not stray from a real scenario making the following assumptions:

- the testing environment is composed from the SUT and from a mock up of the real environment interacting with it (in the following we refer to the latter as Environment Simulation System, ESS);
- the testing scenario for each test case is defined through configuration files containing information about the initial state and all the parameters needed to characterize the test, for both the SUT and the ESS.

Just a single software component, the Session Manager, serves as interface with the operator, where this latter provides as input to the framework a test session description (XML format) containing the following data:

- the list of test cases to be executed;
- for each test case: duration, checks to perform, runtime commands for both the SUT and the ESS.

For each test case, the Session Manager instantiates a Test Manager which deploys the SUT, the ESS, and all the other framework components depicted in Fig. 1.

Communications between the framework on one side, and SUT and ESS on the other, take place trough properly realized driver modules, allowing in such a way to avoid the interaction with the human operator. With the aim to suitably tune the status and then the behavior either for SUT and ESS, such modules send appropriate timed commands as specified in the test session description.

Since we are focusing on ASTERIX based systems, the main communication channel that we take into account is

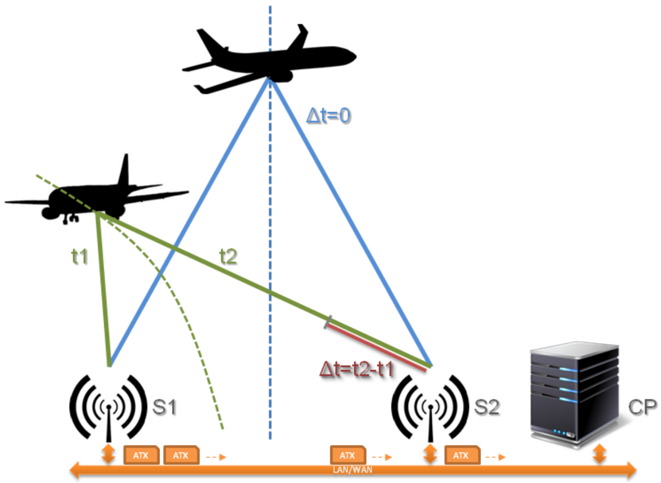


**Figure 2: Multilateration**

a LAN network: this peculiarity can be exploited to capture systems output by simply sniffing traffic flows passing through the network. For this purpose the ASTERIX Sniffer module is used: this software component listens over the network during all the test case execution and catches ASTERIX messages. The processing procedure of such messages consists in dropping all the ASTERIX categories which are not relevant for the current test case, and then in passing the remaining to the ATX Parser which provides a synthetic representation of the observed scenario.

Results obtained from the ASTERIX messages processing will be compared with the test case oracle, that is composed by:

I. the ESS configuration: WHAT is the scenario in which the System Under Test operates;

II. the SUT configuration: HOW the system under test should behave.

Such task is made by the Result Processor that compares expected and observed data applying specific rules with the aim to establish if the distance between them is acceptable or not (e.g. the distance between expected and observed data is beneath specific thresholds), and to establish the overall result for the test case.

The Result Processor generates a report containing the current test case results, and push it to the Session Manager which will combine reports obtained from all the test cases to generate a human readable Session Report containing the following data:

- Results from each test case composing the test session.
- Errors and significant event logs.
- The overall test session outcome.

## 5. CASE STUDY: THE MULTILATERATION SYSTEM

An experimental session to validate the performance of the proposed framework has been implemented using a Multilateration system (MLAT) [3] as use case (Fig. 2), since this kind of ATC system relies upon the ASTERIX standard. Multilateration is a cooperative 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 MLAT main functions are Target Location and Target Identification: for the aircraft localization, the MLAT uses Short/Extended Squitter messages broadcasted by the Mode-S transponders (a technique that permits selective interrogation of aircraft by means of a unique 24-bit aircraft address) and reply messages transmitted by the Mode-S transponders after the reception of a selective interrogation. For the aircraft identification, the MLAT can emit selective interrogations, to request identification and altitude data, and related replies. To locate and identify aircrafts with conventional ATCRBS transponders (Air Traffic Control Radar Beacon System, equipments without Mode S capability), the MLAT system can interrogate aircrafts conventionally and process ATCRBS replies, types 3/A and C to request respectively the target address and the target altitude, received by the aircraft transponders.

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

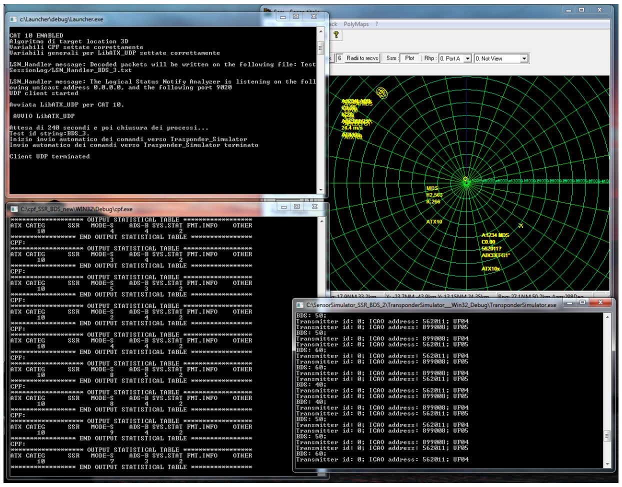
- A transmitting subsystem that includes interrogation message generation and transmission function;
- An optional Intelligent Interrogation process that determines whether an MLAT interrogation is required;
- A receiving antenna array subsystem that receives the transmissions from the target and timestamps 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.

In airport applications, the Multilateration system components are connected through not-redounded LAN Ethernet, using UDP (Layer 4) and IP (Layer 3) protocols.

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.

Furthermore, owing also to the increasing density of aircraft flights, the secondary surveillance civilian radar systems operating in S mode are obliged to selectively manage the routes of an increasing number of aircraft and must additionally allow the exchange of an increasing quantity of data

with the latter, which data are denoted ”commB Data Selector registers” (or ”BDS registers”) according to the terminology in current usage. The ”BDS” are numbered according to the registers of the transponder, which registers contain the flight data. Thus, for example, the following can be differentiated: BDS40 (aircraft intention), the BDS50 (track and turn report) or again the BDS60 (heading and speed report). BDS registers represent a key component of modern SSR systems and also the CP of the multilateration system has strict requirements about extraction and manipulation of BDS registers value.
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, as we describe in following section.



c:\Launcher\debug\Launcher.exe

```
CAT 10 ENABLED
Algoritmo di target location 3D
Variabili CPF settate correttamente
Variabili generali per LibATX_UDP settate correttamente

LSN_Handler message: Decoded packets will be written on the following file: Test
SessionLog/LSN_Handler_BDS_3.txt

LSN_Handler message: The Logical Status Notify Analyzer is listening on the foll
owing unicast address 0.0.0.0, and the following port 9020
UDP client started

Avviata LibATX_UDP per CAT 10.

 AVVIO LibATX_UDP

Attesa di 240 secondi e poi chiusura dei processi...
Test id string:BDS_3.
Inizio invio automatico dei comandi verso Trasponder_Simulator
Invio automatico dei comandi verso Trasponder_Simulator terminato

Client UDP terminated
```

C:\cpf_SSR_BDS_new\WIN32\Debug\cpf.exe

```
******************* OUTPUT STATISTICAL TABLE ******************
ATX CATEG      SSR   MODE-S    ADS-B SYS.STAT FMT.INFO    OTHER
       10                 4        4        2
***************** END OUTPUT STATISTICAL TABLE ****************
CPF:
******************* OUTPUT STATISTICAL TABLE ******************
ATX CATEG      SSR   MODE-S    ADS-B SYS.STAT FMT.INFO    OTHER
       10                 3        4        2
***************** END OUTPUT STATISTICAL TABLE ****************
CPF:
******************* OUTPUT STATISTICAL TABLE ******************
ATX CATEG      SSR   MODE-S    ADS-B SYS.STAT FMT.INFO    OTHER
       10                 5        3        2
***************** END OUTPUT STATISTICAL TABLE ****************
CPF:
******************* OUTPUT STATISTICAL TABLE ******************
ATX CATEG      SSR   MODE-S    ADS-B SYS.STAT FMT.INFO    OTHER
       10                 2        4        2
***************** END OUTPUT STATISTICAL TABLE ****************
CPF:
******************* OUTPUT STATISTICAL TABLE ******************
ATX CATEG      SSR   MODE-S    ADS-B SYS.STAT FMT.INFO    OTHER
       10                 4        4        2
***************** END OUTPUT STATISTICAL TABLE ****************
CPF:
******************* OUTPUT STATISTICAL TABLE ******************
ATX CATEG      SSR   MODE-S    ADS-B SYS.STAT FMT.INFO    OTHER
       10                 8        5        2
***************** END OUTPUT STATISTICAL TABLE ****************
CPF:
******************* OUTPUT STATISTICAL TABLE ******************
ATX CATEG      SSR   MODE-S    ADS-B SYS.STAT FMT.INFO    OTHER
       10                 8        4        2
***************** END OUTPUT STATISTICAL TABLE ****************
CPF:
******************* OUTPUT STATISTICAL TABLE ******************
ATX CATEG      SSR   MODE-S    ADS-B SYS.STAT FMT.INFO    OTHER
       10                 9        4        2
***************** END OUTPUT STATISTICAL TABLE ****************
CPF:
******************* OUTPUT STATISTICAL TABLE ******************
ATX CATEG      SSR   MODE-S    ADS-B SYS.STAT FMT.INFO    OTHER
       10                 7        3        2
***************** END OUTPUT STATISTICAL TABLE ****************
```

C:\SensorSimulator_SSR_BDS_2\TransponderSimulator__Win32_Debug\TransponderSimulator.exe

```
BDS: 50;
Transmitter id: 0; ICAO address: 562011; UF04
Transmitter id: 0; ICAO address: 899008; UF05
BDS: 50;
Transmitter id: 0; ICAO address: 899008; UF04
Transmitter id: 0; ICAO address: 562011; UF05
BDS: 60;
Transmitter id: 0; ICAO address: 562011; UF04
Transmitter id: 0; ICAO address: 899008; UF05
BDS: 60;
Transmitter id: 0; ICAO address: 899008; UF04
Transmitter id: 0; ICAO address: 562011; UF05
BDS: 40;
Transmitter id: 0; ICAO address: 562011; UF04
Transmitter id: 0; ICAO address: 899008; UF05
BDS: 40;
Transmitter id: 0; ICAO address: 899008; UF04
Transmitter id: 0; ICAO address: 562011; UF05
BDS: 50;
Transmitter id: 0; ICAO address: 562011; UF04
Transmitter id: 0; ICAO address: 899008; UF05
BDS: 50;
Transmitter id: 0; ICAO address: 899008; UF04
Transmitter id: 0; ICAO address: 562011; UF05
BDS: 60;
Transmitter id: 0; ICAO address: 562011; UF04
```

**Figure 3: Automatic Testing Session**

## 6. FRAMEWORK TUNING AND EXPERIMENTS

The tuning process started with the customization of the test session descriptor by adding the following 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 open 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 aircrafts position (both operating in mode S or intermodal A/C) and extraction frequency and values of BDS registers, 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, simulated position, BDS registers values; 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. Defined as 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.

III. All and only enabled BDS register values should be received. The extraction frequency must satisfy specifications.

In order to verify if some required mechanisms of error correction were correctly implemented in the CP, we provided the Result Processor with a lightweight logic replication of CP’s behavior in case of scenarios with ambiguous or noise affected data. For example, let’s assume that:

- The hamming distance between two target identifiers is less than a configurable threshold.

- One of the target is only seen by a number of receiving station beneath an acceptable minimum.

In case that these conditions are simultaneously true, the CP should merge the plots of both targets assuming that one of the identifier is only a corrupted version of the other one. Moreover, we deployed another component listening on the communication channel between the CP and the Maintenance Display Terminal (MDT) in order to capture and analyze messages carrying information about the logical status of the SUT: this allows to verify that the transactions between a logical status and another (operative/warning/failure) is coherent with the simulated scenarios, and otherwise to report some warning in the session log.

After implementing a proper customization as we just outlined, we used the framework to replicate the execution of a whole qualification test session for the CP component. 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 time consuming, hence very expensive in terms of human resources: test duration is limited by the availability of the operator, while the effectiveness of a test would grow proportionally with the increase of its duration. The automation process enabled through the use of the framework aims to reverse this proportionality between human effort and effectiveness, thus increasing efficiency. Major engagement by the operator is limited to start up and configuration phases: after that the execution time can have an arbitrary duration. Moreover, should be noted that manual testing procedure may be affected by human errors: on this side, 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. 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.

## 7. CONCLUSIONS AND FUTURE WORK

Performing testing campaigns aimed at verifying functional aspects as well as non functional requirements for software components, is not a trivial task especially when concerning highly dependable systems. In this paper we proposed FITNESS, a framework for automatic testing of ATC software components that rely on the ASTERIX protocol, a messaging format for the exchange of surveillance information. After introducing the framework architecture, we shown results obtained from an experimental campaign conducted with the aim to underline advantages provided by the FITNESS framework when compared to the execution of traditional testing procedures. As case study we referred to an ATC surveillance system named Multilateration system. Results from the experimental session led us to confirm that the execution of testing procedures through the proposed automatic tool, increases the efficiency of Verification and Validation processes thanks to a considerable reduction of the required human resources, and it also allows to prevent not negligible errors that may occur when testing procedures and checks are completely human made. The FITNESS framework was designed following a general purpose approach to allow the customization for any ASTERIX-based application. The framework is now a TRL 4 prototype, and it is continuously evolving. Current developments are aimed at simplifying the framework configuration process through a GUI that will be in charge of producing the needed input files. Another future work is related to the automatic generation of STR (Software Test Report) documents [7], that contain information produced by the FITNESS framework, organized according to predefined formats and requirements. We are also analyzing the chance to realize a different version of the framework that relies on the event driven approach.

## 8. 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 116âĂŞ119. Ieee, Sept. 2006.

[4] G. Galati, M. Leonardi, P. MagarÃš, 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).*

[7] J. Radatz, M. Olson, S. Campbell, *MIL-STD-498*, Crosstalk, the Journal of Defense Software Engineering 1995; 8(2):2âĂŞ5. http://www.stsc.hill.af.mil/crosstalk/1995/feb/milstd.asp [12 June 2002].