Using Static Code Analysis to Hunt Down Bugs

Jean-Jacques Lévy explains to us how these software verification tools have conquered the world of space science and avionics in just 10 years.

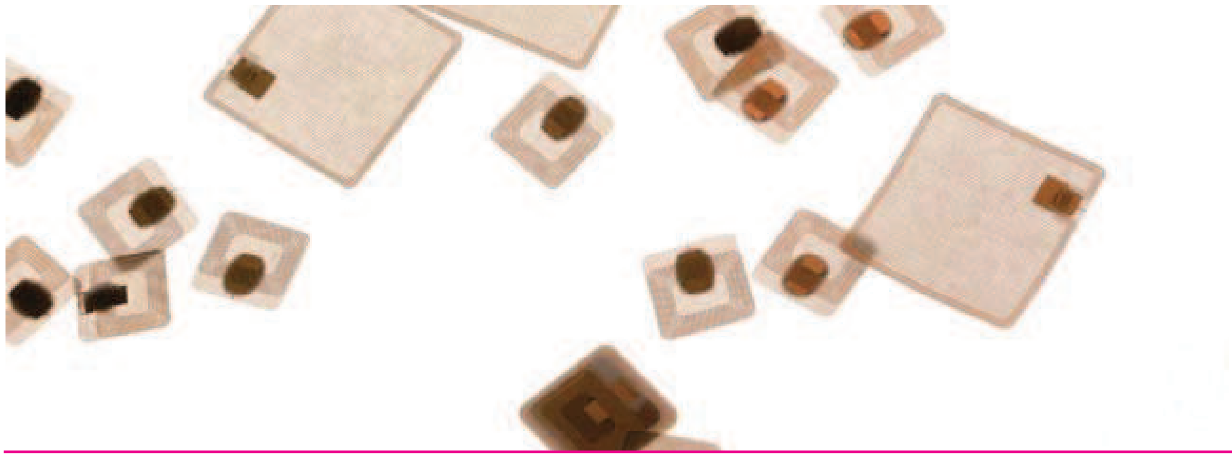
INédit : Everything began with the explosion of Ariane 5 during its maiden flight 501. Tell us what happened?
Jean-Jacques Lévy: It was back in July 1996. The new version of Ariane exploded 39 seconds after its launch. The CNES engineers quickly realized that it was a software problem, a type of error that had never been envisaged until then. The bug came from the navigation system, which had detected a software "exception" caused by an arithmetic overflow during the conversion of a floating point number into a whole number. The piece of the programme concerned, which had been inherited from Ariane 4, was in fact useless for Ariane 5 and had not been protected. The investigation committee led by Jacques-Louis Lions, President of the French Academy of Sciences, asked Gilles Kahn, the INRIA Scientific Director, to have the whole code investigated. This is how we brought together a small team of programming and programming language theory experts: Alain Deutsch, who sadly died last summer at the age of 41; Damien Doligez; Robert Ehrlich; Georges Gonthier; François Rouaix; Marcin Skubiszewski, and I.

INédit : You therefore went through all the software with a fine-tooth comb. Which tools did you use?
Jean-Jacques Lévy: We're talking about 140.000 lines of code in ADA, a high-level language! Firstly, we ran this code under Unix and we managed to compile it entirely. Then, thanks to the IABC code analyzer that Alain Deutsch had been developing for 10 years, which is the best in the world, we verified Bernstein's Conditions, namely the access to the variables shared by several concurrent tasks, an analysis that is based on the search for alias. In this way, Georges Gonthier discovered several logic errors in the flight program using graphic representations from the analyzer. This was the first time that static analysis of programmes was used on this scale. The value of this approach for verifying the safety of critical embedded software was clearly demonstrated.

INédit : Did you therefore become experts in aerospace programming as well?
Jean-Jacques Lévy: In a way, yes. Some of us took part in the qualification committee for flight 502, which was a success. Consequently, CNES subjected each new programme version to Alain's software. With Daniel Pilaud, they created a company, Polyspace Technologies, in 1999, to detect software execution errors automatically. In this way, they verified almost all the embedded codes of the EADS spacecraft, like the ATV (Automated Transfer Vehicle), the resupply vehicle for the International Space Station (ISS) and the Atmospheric Re-entry Demonstrator (ARD), and this success has spread to other sectors, like the automobile, defence and energy industries. Later, we were tasked with reviewing the embedded code of the European module (Columbus) of the ISS by the European Space Agency (ESA), and we contributed to the definition of new programming rules for space.

INédit : Autrement dit, l'analyse statique de code a gagné ses galons dans l'industrie ?
Jean-Jacques Lévy: Yes, indeed. This computer specialist outlook and these theoretical software verification tools have become an everyday part of life. They are now rightly seen as essential for detecting hidden design defects. And more recent progress has been made on the embedded code of the A380.■

CONTACT
Jean-Jacques Lévy, *Moscova Project-team, INRIA Research Centre - Paris-Rocquencourt, and director of the joint INRIA Microsoft Research research Centre*
Tel.: +33 1 39 63 56 89, Jean-Jacques.Levy@inria.fr



The Future is already Asynchronous

A page has turned: the industry is breaking with the synchronous model.

The first computers were of course asynchronous, leading to problems such as deadlock, famine and access conflicts to shared resources. To avoid these problems, architects have favoured a more simple approach in which all the parts of a circuit have to be synchronized to a single clock. For a long time, this compromise seemed acceptable: almost all circuits were synchronous, along with the CAD tools used to design them; and the continuous increase in clock frequencies compensated for the loss in performance caused by the single clock.
That situation is poised to change radically. At the macroscopic level, we are witnessing the emergence of distributed infrastructures (Internet, grids, and communicating embedded systems) which structure the world asynchronously. At the microscopic level, the race for performance demands the abandonment of the synchronous

model in circuits because of its disadvantages: high consumption, the silicon surface required for the clock, radio emissions which allow the operations of the circuit to be "spied on", and so forth. Not being able to increase the clock frequencies indefinitely, manufacturers are turning towards more asynchrony: multi-core processors, GALS (Globally Asynchronous, Locally Synchronous) architecture, asynchronous logic, etc. The different parts of the systems operate at different speeds and can even be stopped when their use is not required.
The same evolution is taking place in other domains that have traditionally applied the synchronous approach. In cars, the multiplicity of on-board computers renders it impossible or too costly to synchronize them in single-step mode. This is also the case in airplanes where distributed modular avionics (DMA) is tending to replace integrated modular avionics (IMA).
This move towards greater decentralization is however impeded by considerations of reliability. In fact, asynchronous systems are complex to design because of the presence of simultaneous actors and events, which shatter programming habits and require specific asynchronous design tools which are still not currently available.
The Vasy team is confronting this problem by using formal methods to model asynchronous systems and enumerative verification techniques. To limit the number of cases whose size would exceed the computer's memory or take up too much time, these researchers are combining "brute force" — exploring all the possible combinations — with "smart" strategies. Their results are implemented in the CADP Toolbox (see Inédit No. 58) used in the Multival Project of the Minalogic Competitiveness Centre and the Topcased Project of the AESE Centre, among others.■

CONTACT
Hubert Garavel, *Vasy Project-team, INRIA Research Centre - Grenoble Rhône-Alpes*
Tel.: + 33 4 76 61 52 24, Hubert.Garavel@inria.fr

Harnessing the Approaches to Generate Adapted Tests

The test methods based on specification models (model-based testing) and the methods based on code (structural testing) are coming together to create new test generation tools.

Making sure that a computer system does exactly what its designer intended it to do is a software engineering step that is even more decisive because the systems concerned are embedded or critical (see article on Ariane 5). In an embedded system, the difficulties relating to validation and testing are increased tenfold: the real-time criteria are much more constrictive than in telecommunication networks, for example, and the number and size of the variable fields to be taken into consideration are a lot greater. It is impossible, in these conditions, to test programmes exhaustively through the enumeration of the possible variable values. One solution, which has been developed over these last ten years, consists in processing the variables in a symbolic way by representing sets of vectors of values of variables by formulae. It is then possible to apply constraint resolution and propagation techniques and, more recently, abstract interpretation methods for models capable of guiding the test generation.

For the researchers of the Vertecs team, the future undoubtedly lies in the combination of these methods. They have therefore designed an original tool, the STG (Symbolic Test Generator), which selects tests from reactive system models and test objectives describing the abstract behaviour to be tested. The selection of tests relies on an approximate analysis through abstract interpretation to reinforce the constraints on the model inputs in order to meet the objective. These constraints are then resolved when the tests are executed on the actual system. The approach is supplementary to techniques based on code coverage (instructions, branches, etc.) stemming from structural testing. This approach can, in fact, be used as a basic module for the selection of tests based on the coverage, and completes this coverage by targeting risky behaviour. Forthcoming evolutions will focus on the complexity of the models (size, control structure, heterogeneity of data), and the adaptation of the coverage criteria to the testing based on models which, unlike structural testing, still do not have the same structure as the code.
For the researchers, STG is an example of the feasibility of this approach for reactive programme models manipulating Boolean and numeric variables. STG is currently being distributed and, if it gets the thumbs up, the model-based testing and structural testing communities will have everything to gain by working together despite the cultural and language differences.■

CONTACT
Thierry Jéron, *Vertecs Project-team, INRIA Research Centre - Rennes Bretagne Atlantique*
Tel.: +33 2 99 84 74 64, Thierry.Jeron@inria.fr