Trusting the machine

Prof. Gordon Pace
Mr Christian Colombo

As the 21st century unfolds, computers are becoming a natural part of our lives. Children are now growing up with a computer always within easy reach. This raises the need for secure and consistent computer systems. Runtime monitoring is a technique to continuously test systems to ensure they work correctly. This approach is being used in areas from astronomy to online casinos. Words by **Prof. Gordon Pace** and **Mr Christian Colombo**.

Computers are everywhere, in phones, fridges, airplanes, and cars, which leads us to take them for granted. Only when they stop working do we realise their importance. As users, we expect computers to work: reliably, correctly, consistently, and quickly. Failure to meet these expectations carries a high cost.

Disappointed customers quickly turn to alternative software packages and services. Computer software loopholes allow malicious users to gain access to confidential information, sometimes for a profit. Disgruntled users have even sued companies for damages. History is riddled with such stories; patients were over-radiated by the Therac-25 radiation therapy machine when software failed, while the Ariane-V rocket was lost because of malfunctioning software. Some lower profile incidents include phone software bugs allowing Wi-Fi password theft, and online store backend » programming errors permitting fake financial transactions. Computer science focuses on the development of techniques that support the construction of large, reliable, and efficient systems.

Software development needs rigorous testing during which a system's behavior is checked against questions such as: *Do the financial accounts controlled by the computer system always balance at the end of the day? Is the system response fast enough when processing up to 100 concurrent users?* Before the system is deployed and used, it is tested against various user inputs and situations. This allows developers to discover errors in their code and fix them before the system is used. However, no matter how thoroughly tested, it is impossible to check every single scenario and errors will always crop up.

A complementary technique to testing is runtime verification. Using this approach, the software continues checking for errors even when the system is being used. If anything goes wrong while the program is running, the code monitoring the system notices the error and raises an alarm. For example, if the system has stored client credit card information in a location which is accessible to outside users, the information is moved to a secure location or deleted. By using actual user input, the system bypasses the need to create hypothetical user inputs — the main difficulty in testing – while also guaranteeing that if something goes wrong it can be fixed immediately.

The approach may seem straightforward but is actually full of challenges. The authors and Dr Adrian Francalanza form a local team addressing these challenges aimed towards practical solutions. One major challenge is the monitoring costs inbuilt into this approach that eat computing power and storage. Reducing computer usage is crucial to allow its adoption to real-life applications. Another challenge is how to monitor systems that are physically in different locations. For example, a doctor might use a laptop to access and modify a patient's medical records that are kept in a hospital's database, while medical test results are kept in the laboratory's database. We are currently trying to address how all data is kept synchronised and correct, while ensuring that the system does not expose private information.

"…more than a million transactions have been verified…"

We have built and provide several software tools to other academics and developers. LARVA is the most widespread tool since it can monitor Java-based systems (Java is a programming language) used all over the Internet and on Android smartphones. It has been used in projects with various industrial partners, and applied to real-life software, from checking financial transactions to detecting intruders on file-transfer servers. We have also released other tools to handle other programming languages and special customised settings.

What follows are some examples of how runtime verification is applied in the real world.

Financial transactions

Financial transactions need high security. Flaws in financial transactions result in loss of user trust and direct losses through fraud. Financial systems support so many different services (different currencies, payment methods, user types, promotional offers and so on) that testing all possible real-life scenarios is virtually impossible. To make matters worse, a minor financial error or fraud like the loss of one cent on every transaction, might go undetected. To immediately detect errors, software can continuously monitor transaction validity as they are happening, which provides an added security layer.

Computing at ESO, controlling the very large telescopes. *Image: ESO/José Francisco*

In Malta, we have applied this approach at Ixaris Ltd, an international provider of virtual credit cards and payment services. To date, more than a million transactions have been verified through runtime monitoring.

Online Betting

Online betting companies are important for Malta's economy. They share similar problems to companies carrying out financial transactions, with two additions. Firstly, online betting needs to be fair to all players. Local authorities need to check this fairness quickly to uncover abuse. By using runtime monitoring, online betting companies can continuously monitor their own operations and prove that they play fair. Secondly, the same online betting company can host different casinos run by external parties. For every casino, the company forms a unique contract and partnership agreement. For example, for licensing reasons, the betting company may restrict a casino from making certain games available in particular countries. To ensure that contracts are being followed, the casino can use runtime monitoring to continuously check on the providers. We are trying to solve these problems by developing new contract languages and tools for betting companies.

Gaming

Games are serious business and errors have major consequences. Unexpected errors tarnish a user's experience, while allowing other players to cheat. In games with multiple players, cheating is a long running problem. Especially when money is involved, it threatens the trust of players in the fairness of the game.

Cheat detection is very hard because there are too many real-life situations to simulate, a laboratory environment would never catch them all. Constant monitoring of a gaming system through runtime verification is the best approach to try and detect cheats. Runtime verification lets companies easily observe users and cheats. By detecting abnormal behaviour, this approach can harvest rich game information without any changes to the game's structure.

Astronomy

Scientists scan the skies with radio telescopes which keep getting bigger to look deeper into space. The largest telescope in the production pipeline is the Square Kilometer Array (for a feature on SKA see pages 12–17), which uses areas covered by hundreds of radio telescopes to observe stars with phenomenal clarity. The large number of telescopes generate data at an enormous rate that needs super efficient processing to ensure quick results. For this application, runtime verification helps detect objects in space while minimising data processing costs .

Our team has detected astronomical phenomena called pulsars using runtime monitoring. Pulsars are highly magnetized, rotating neutron stars that emit a beam of radiation detected by the telescopes. The beam has a highly regular time interval making it easy to pick up. Its predictable properties make pulsar detection similar to software error pattern recognition. With this in mind, we could modify our techniques to detect pulsars from telescope data with great success.

Computers will continue integrating themselves within our lives. More research is needed to make them work better, simpler, and more reliably. The more trust we hand over to computers the more secure they need to become. Because of increased complexity, computer scientists need to design solutions that are more intelligent. Like never before, users will forget the computer behind their devices. ●

FURTHER READING

For more about this group's research see: http://www.cs.um.edu.mt/svrg/