WHY

Software quality, a concern in IST.

One of the key issues in Information Society Technologies (IST) concerns the quality of software, or rather, the lack of it. Since the number of products and services that depend on software is growing, poor quality can significantly affect society. A lack of quality can reveal itself in a number of ways, like badly implemented or non-functioning features, operating system failures, erroneous outputs, unfulfilled time constraints, loss of service, etc. These faults can have various consequences that range from customer dissatisfaction to damage of physical property or the environment. Even considering the lesser of these effects, defective software gives rise to high maintenance costs for the company that develops it, thus making it lose its competitive edge or even leading it to bankruptcy.
Trends: increasing complexity and decreasing time to market.

On the one hand we detect trends such as globalisation, standardisation and shorter lifecycles that place great demands on the flexibility of the software industry. In order to compete and cooperate on an international scale, a constantly decreasing time to market and a constantly increasing level of quality of their products are essential.
On the other hand, however, complexity of software systems is increasing. Software systems get bigger, connect large amounts of components that interact in many different ways, and have constantly changing and different types of requirements (functionality, dependability, real-time, etc.).
Automation of tedious, difficult and time-consuming tasks seems to be the only way to master this complexity and develop well-functioning quality software systems within a competitive amount of time.
Where are the biggest problems and what should be automated.

There exists reasonable empirical evidence that two of the major sources of software and systems errors lie in: (1) poor, erroneous, missing and changing/unstable requirements and (2) deficient testing of both functional and non-functional properties. Requirements capture is a process that is difficult to automate because it requires human ingenuity and person to person communication. However, many activities of software testing can be automated. Testing comprises more that 50% of the activities in the whole development life-cycle and so automating some of them will definitely reduce the time to market and increase the quality of the resulting system. The testing life-cycle globally consists of preparation, specification, execution & evaluation and analysis activities. The most time consuming, tedious, difficult and error-prone activities are specification of the test cases and execution & evaluation of the results. Hence there is a need to automate these activities.
Why apply evolutionary algorithms and meta-heuristic search techniques.

The domain of possible inputs that can be fed to software, even for the most trivial program, is usually too large to test exhaustively. Consequently, one of the major challenges associated with automated testing is that of finding test cases that are effective at finding flaws without requiring an excessive number of tests to be carried out. Formal analytical and classical test generation methods often fail because of the combinatorial explosion of possible interleavings in the execution or functional specification of several properties. Stochastic optimisation and search techniques (like genetic and evolutionary algorithms) are designed to find good approximations to the optimal solution in large complex search spaces. Moreover, these general-purpose search techniques make very few assumptions about the underlying problem they are attempting to solve. As a consequence, they are useful during the automated generation of effective test cases because they avoid one of the most difficult obstacles with which the software tester is confronted: the need to know in advance what to do for every situation which may confront a program.