EvoTest

Evolutionary Testing for Complex Systems

Tanja E.J. Vos, ITI, Spain
Joachim Wegener, DaimlerChrysler AG, Germany
EvoTest Vision

- Improvement of test quality addressing
  - functional as well as
  - non-functional system characteristics
difficult to test and automate with traditional testing techniques

- Increased test efficiency through complete test automation

- Reliability figures for the systems under test on basis of the tests applied

- Open traditional formal software engineering to the benefits of using evolutionary computation to solve software engineering problems
Facts: Software System Testing

• Testing
  ▪ Process of executing a software system with test data in order to find errors
  ▪ Testing comprises up to 50% of the effort spent during development life cycle
  ▪ Testing is necessary for better quality systems and for providing insight into the risks

• General Trends
  ▪ Decreasing time to market
  ▪ Life depends more and more on software and, therefore, correct functioning of software becomes much more critical (e.g. in cars, phones, …)

• Automation of testing is the only way to
  ▪ perform tests efficiently
  ▪ develop well-functioning quality software systems within a competitive amount of time
Test Case Design - Determining Test Quality

- Exhaustive testing would result in the highest quality, but domain of possible inputs is usually too large to test exhaustively
  - Major challenge is finding effective test cases without requiring an excessive number of tests to be carried out
  - Effectiveness of test cases defines the quality of the whole testing process
- Fault-finding test case generation is very hard for complex systems. Partially, it is possible to generate test cases from behavioral models or formal specifications
  - coverage achieved is low, and
  - use of formal methods not common practice in industry
- Manually specifying test cases is difficult, time consuming, and expensive

There is a need for test automation
Facts: Testing complex software systems

- For complex systems, the automation of testing is even more important in order to master complexity
  - large amount of components
  - many interactions and dependencies
  - feature interactions
  - different types of requirements (functionality, dependability, real time, etc.)
  - no detailed system description possible by a single individual
  - constantly changing requirements
  - not likely to have a formal spec. for all components
  - even with formal specs., emergent non-functional characteristics of the whole system are hard to predict
  - several million lines of source code
EvoTest, Why?

• **Meta-heuristic search techniques:**
  - are designed to robustly find quasi-optimal solution in large complex search spaces that involve many variables
  - successfully applied in various engineering domains
  - make very few assumptions about the underlying problem they are attempting to solve

• **Testing could be transformed into a search problem**
  - search in the input domain of a system under test for important test data sets
  - Automation of test execution is a prerequisite because an evolutionary test generates many test cases in order to learn which are the important ones

• **Stimulate and enlarge the view of existing software development techniques and developers and vice versa**
EvoTest Goals

• Develop an Evolutionary Engine (EE) that applies meta-heuristic search techniques to generate test cases
  ▪ Transform testing into an optimization problem
  ▪ Develop new techniques to define the objective function for specific testing objectives (search for defects, testing of OO systems, model-based testing, temporal behavior testing,…)
  ▪ Develop new representations for test cases/data
  ▪ Develop new techniques to improve the search process (automatic adaptation of the search, seeding strategies, hybrid search techniques, optimization problem description)
  ▪ Develop new techniques to generate complex test data sets

• Develop Automated Testing Engines (ATEs) for at least 3 different types of complex pilot systems and properties that automate the execution of the generated test cases
  ▪ Develop a general architecture/framework that separates general and specific issues for different application areas and properties
EvoTest Goals

• **Combine Evolutionary Testing with other Static and Dynamic Analysis Techniques, e.g.**
  - search space reduction through data-flow analyses (evaluation, completion)
  - search space reduction through symbolic execution (possible value ranges for the different inputs)
  - testability transformation for easier test data generation
  - Combination of real-time testing and structural testing
  - Combination of functional testing and structural testing
  - Seeding of test data

• **Product Reliability**
  - Assessment of the abilities of evolutionary testing to contribute to product quality and reliability
  - Comparison and adaptation to traditional reliability growth models
EvoTest, How

- **Individuals**
- **Test Data**
- **Monitoring (SUT)**
- **Fitness Values**
- **Test Execution (SUT)**

**Selection**
- **Recombination**
- **Reinsertion**

**Evolutionary Engine (Search Process)**

**Test Results**

- **Initial Population**

**Automated Testing Engines (Testing Process)**

- **Test Case Generators (e.g. from specifications, flow diagrams, models, use-cases, proof objects, combinatorial, etc.)**

- **Evaluation**

- **Termination ?**

**Reporter tool for data collection and advanced analyses**

- **Add info on likelihood, probability, importance, risks, etc..**
EvoTest Preliminary Partners

• Potential Partners Meeting held on April 5th, 2005 in Berlin
• Partners:
  ▪ beXtec, Germany
  ▪ DaimlerChrysler, Germany
  ▪ Ericsson, Sweden
  ▪ Fraunhofer FIRST, Germany
  ▪ INRIA, France
  ▪ ITI, Spain
  ▪ King’s College, UK
  ▪ Motorola, France/UK
  ▪ RILA, Bulgaria
• All partners committed to contribute to the proposal
Evotest Preliminary Pilot Projects

- The techniques and tools from EvoTest will be generally applicable to many types of SW-based complex systems
- Pilot studies
  - Products widening access for disabled people
    - Many different interfaces involved,
    - Enabling technologies, e.g. voice-generating system
  - Large telecom applications
    - Telecommunications software are very complex artifacts with hundreds of different components and more than $10^7$ lines of code
  - Complex embedded automotive systems
    - Interaction of up to 60 embedded control systems communicating via different bus systems
    - Robust with respect to real system environments that do not only depend on the user but also on physics
1. Project management (ITI)
2. From test to optimization (DC)
3. Search techniques (INRIA)
4. Combination of traditional VVT methods and ET (King‘s College)
5. Automated Testing Engines (Fraunhofer First)
6. Automotive Case Study (DC)
7. Telecom Case Study (Ericsson)
   1. Motorola
   2. Ericsson
8. Disabled Situation Improvement Case Study (RILA)
9. Assessment (Motorola)
   1. Measures, Metrics
   2. Reliability Figures
   3. “Data mining” for test results from evolutionary testing
      1. Learning fitness function from the tests performed so far
      2. Predicting the number of local optima
   4. “Recycling” of test results
10. Dissemination (all partners)