FITTEST
FUTURE INTERNET TESTING
Valencia, 9th of December 2010
FITTEST (ICT-257574, 2010-2013)
## PARTICIPANTS

<table>
<thead>
<tr>
<th>Participant organisation</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universidad Politécnica de Valencia (coordinator)</td>
<td>Spain</td>
</tr>
<tr>
<td>University College London</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>Berner &amp; Mattner Systemtechnik</td>
<td>Germany</td>
</tr>
<tr>
<td>IBM</td>
<td>Israel</td>
</tr>
<tr>
<td>Fondazione Bruno Kessler</td>
<td>Italy</td>
</tr>
<tr>
<td>Utrecht University</td>
<td>The Netherlands</td>
</tr>
<tr>
<td>SOFTeam</td>
<td>France</td>
</tr>
<tr>
<td>Sulake</td>
<td>Finland</td>
</tr>
</tbody>
</table>
**Future Internet**

- The Future Internet (FI)
  - a complex interconnection of services, applications, content and

- Society increasingly dependent on FI
  - critical activities such as social services, learning, finance, business, as well as entertainment.

- Challenging problems for quality assurance/testing
FUTURE INTERNET: CHALLENGES (1)

- **CH1 Self modification**
  - Rich clients have increased capability to dynamically adapt the structure of the Web pages; server-side services are replaced and recomposed dynamically based on Service Level Agreements (SLA) and newly discovered services; components are dynamically loaded

- **CH2 Autonomic behaviour**
  - FI applications are highly autonomous; their correct behaviour cannot be specified precisely at design-time.

- **CH3 Low observability**
  - FI applications are composed of an increasing number of 3rd-party components and services, accessed as a black box, which are hard to test.
FUTURE INTERNET: CHALLENGES (2)

- **CH4 Asynchronous interactions**
  - FI applications are highly asynchronous and hence hard to test. Each client submits multiple requests asynchronously; multiple clients run in parallel; server-side computation is distributed over the network and concurrent.

- **CH5 Time and load dependent behaviour**
  - For FI applications, factors like timing and load conditions make it hard to reproduce errors during debugging.

- **CH6 Huge feature configuration space**
  - FI applications are highly customisable and self-configuring, and contain a huge number of configurable features, such as user-, context-, and environment-dependent parameters.

- **CH7 Ultra-large scale**
  - FI applications are often systems of systems; traditional testing adequacy criteria cannot be applied, since even in good testing situations low coverage will be achieved.
FITTEST PROJECT

Traditional

- Development
- Testing
- Fixed testware (models, testcases, oracles)

Release 1: Regression testing

Release 2: Regression testing

Release 3: Regression testing

Changed requirements

Adapt manually to changed requirements

FITTEST

- Self-modifying
- Changed requirements
- Autonomic

Evolving testware (models, testcases, oracles)

Automated Continuous Evolutionary Testing

FI applications, dynamically composed of 3rd party components and services
FITTEST PROJECT

- European project FITTEST (ICT-257574, 2010-2013)
- Aims at addressing FI testing challenges
- Continuous evolutionary automated testing
- Developing an integrated testing environment
  - Model inference, oracle learning, classification trees, concurrent testing, regression testing, etc.
- Assessment of developed techniques on FI applications provided by industrial partners
  - Modelio CASE Tool
  - Habbo (virtual world)
How (1)

- FITTEST Testing Environment
  - Integrates various FI testing techniques
  - Execution within IDE
  - Continuous execution
  - Visualisation

- Evolutionary Testing Package
  - Based on adaptive search based techniques
    - Can deal with high dynamism and variation of FI applications
  - Provides generic API

- Analysis Package
  - Provides language independent API
  - Dynamic: selective instrumentation, monitoring and logging
  - Static: code analysis, textual information processing
How(2)

- Future Internet Testing Techniques Package
  - Defined on top of the «Evolutionary Testing» package
    - Model inference tool
    - Test case generation tool
    - Oracle learning tool
    - Diagnosis and fault detection tool
    - Classification trees module for combinatorial testing
    - Concurrency testing tool
    - Co-evolving test case generator
    - Coverage and regression testing tool
How

Dynamic analysis package
- Language-specific code transformations
- Selective instrumentation
- Logging

FITTEST Integrated Testing Environment
- Execution within IDE
- Continuous execution
- Visualisation

FI testing techniques package
- Model inference tool
- Test case generation tool
- Oracle learning tool
- Diagnosis and fault detection tool
- Classification trees module for combinatorial testing
- Concurrency testing tool
- Co-evolving test case generator
- Coverage and regression testing tool

Static analysis package
- Language-specific code analysis
- Textual information processing

Evolutionary package
- Fitness functions library
- Search based algorithms
- Genetic programming module

Monitoring and logging information (e.g. states, coverage, etc.)

Static information (e.g. CFG, linguistics, etc.)

Generic API
WORKPACKAGE STRUCTURE

WP1: Project Management (UPVLC)

WP2: Experiments, Case Studies, Benchmarking and Assessment (UPVLC)

WP 3: Evolutionary Search Based Testing (UCL)

WP4: Model-Based Testing (FBK)

WP5: Combinatorial Testing (B&M)

WP6: Log-based Diagnosis and Oracle learning (UU)

WP7: Concurrency Testing (IBM)

WP8: Regression Testing and Coverage (IBM)

WP9: Integrated Environment for Continuous Testing (SOFT)

WP10: Case Studies (SUL)

WP11: Exploitation and Dissemination