Defect discovery by simulation-based testing and model checking

Richard Lipka

Marek Paška

Tomáš Potužák

SERENE 2014, Budapest

EUROPEAN UNION
EUROPEAN REGIONAL DEVELOPMENT FUND
INVESTING IN YOUR FUTURE

OP Research and
Development for Innovation

Motivation and goals

- Software components as standard assembly units
  - Stored in a repository
  - Described in terms of
    - Provided services
    - Extra-functional properties
    - ? State ?
- Testing component-based application
  - Testing of components + testing of finished assembly
  - Replacement of human testers when it is possible
  - Speedup of testing after update

Basic concept

- Technologies
  - Java + OSGi (Blueprint extension)
  - Code generation + Java Pathfinder
- Simulation for
  - Testing of software functionality
  - Verifiing extra-functional properties
- Model checking for
  - Proving of properties

SimCo – simulation testing

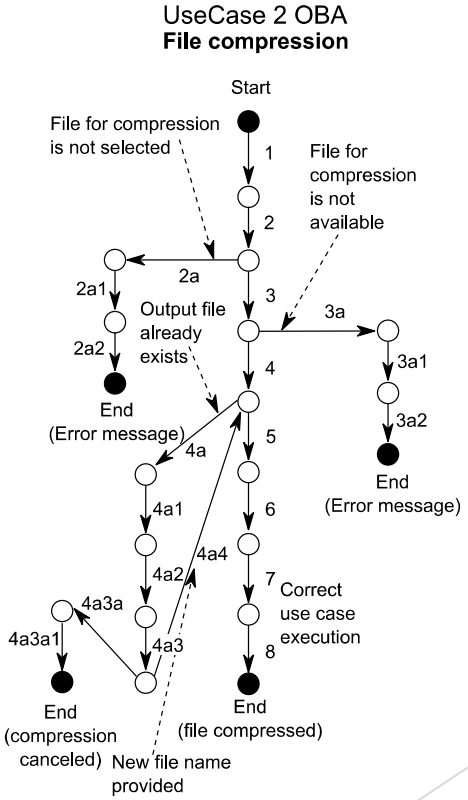
- Using of simulated components for
  - Replacing human testers
  - Oracles when real components are too slow or not available
  - Simulation of the environment
- Hybrid simulation
  - Simulated and real components run in the same environment
- Non-invasive testing
  - No changes in the real components → the same binaries as in production system

SimCo and use cases

- Use cases as a basic definition of application behaviours
  - Good base for test from the user point of view
  - Described in semi-structured way → graph representation of application behaviour
- Test capable of
  - Observing correct sequence of action
  - Observing data (correct values, data amount)
  - Checking properties (latency)

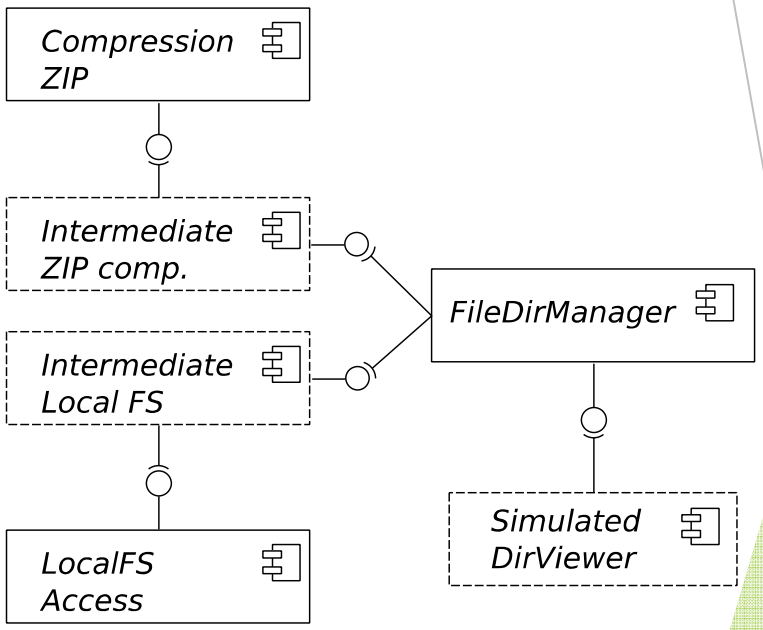
SimCo and use cases

1. The user starts the compression
<use: #selectedFileName>
<method:FileDirManager.compressFile; #selectedFileName; >
2. The system retrieves the file from FS access component
<use:selectedFileName> <create:selectedFile>
<method: FSAccess.getFile; #selectedFileName; #selectedFile>
<efp: min_time;500 ; ms>
3. The system prompts user to provide name of the compressed file
<method: DirViewer.getNameDialog; ;>
4. The user provides new filename and confirms it
<create:newFileName>
<method: DirViewer.getName; ; #newFileName>
5. The system creates an empty file
<create:newFile>
<method: FSAccess.createFile; #newFileName; #newFile>
6. The system asks compression component to perform compression to a new file
<use:newFile>
<method: Compression.compress; #selectedFile, #newFile; >
7. Compression component notifies system that the compression is finished
<create:compressionMessage>
<method: EventAdmin.sendMessage; "compression finished"; #compressionMessge>
8. The system informs the user
<use:compressionMessage>
<method: DirViewer.showDialog; #compressionMessage; >
Variation: 2a There is no file selected
2a1. System displays an error message
<method: dirViewer.showDialog; "No file selected"; >
2a2. Use-case aborts <abort>

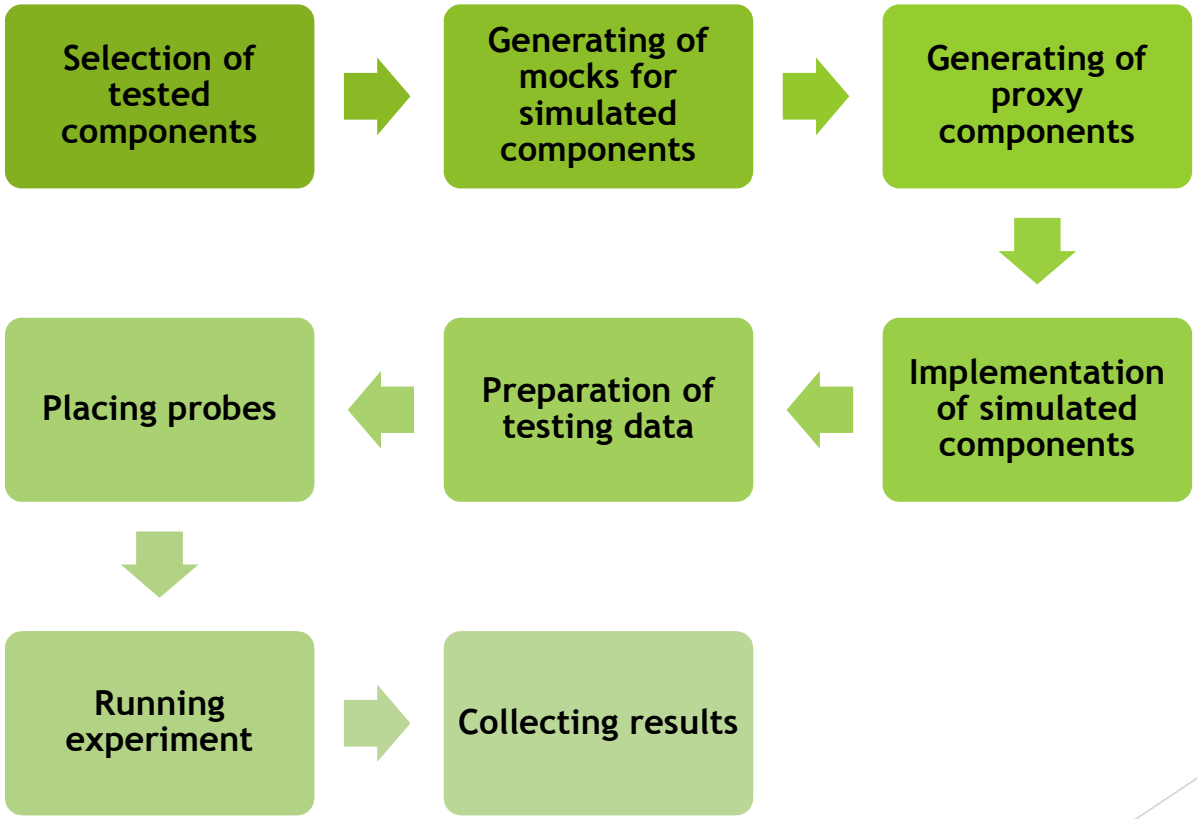


SimCo – Basic structure

- Core components
  - Calendar, results collecting
- Transparent proxy
  - Automatically generated
  - Runtime or source texts
  - Observation, network simulation
- Simulated components
  - Manually prepared
  - Mimic behaviour of the real ones

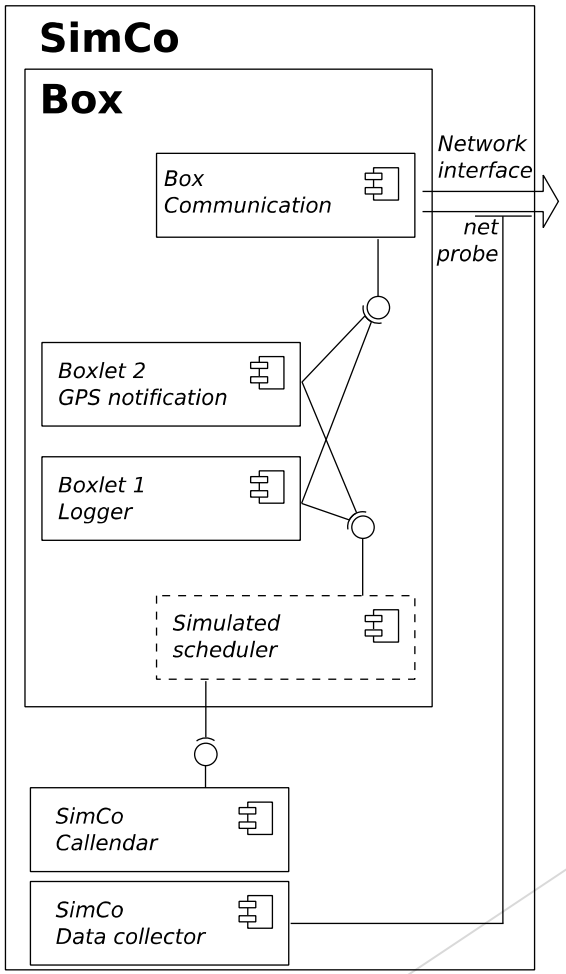


SimCo - workflow



SimCo – example

- Testing of car onboard software
  - Applications – „Boxlets“ as components, available from repository
  - Measurements of network communication
  - Speeding up of testing by replacement of original scheduler by its simulation

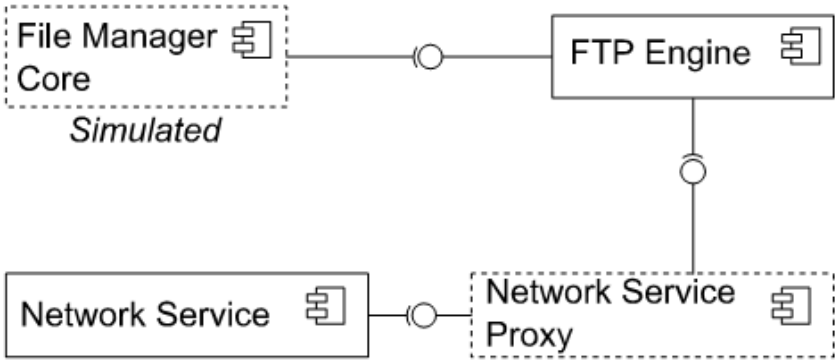


Model checking framework

- Source code generated from model
  - Executable models in Python-like language
  - Automated translation to
    - C language
    - Java language
- Based on Java Pathfinder
  - Special kind of JVM
  - All possible program paths are explored and executed

Case study – FTP protocol

- Interaction of FTP server and client
  - Component application



- Correct behaviour – requested data are recieved, no exception appears

Case study – Randomized server responses

- Server answers with random sequence of protocol messages
  → communication should fail gracefuly
- Error in client
  - When server fails at the end of data transfer, client states in state transfering
- SimCo uses random number generators
  - Cca 200 runs to discover exception
- Java Pathfinder explores all paths
  - 40 states visited before exception halts the execution

Case study – cutting responses

- Response of server randomly shorted
- SimCo discovers error every time – comparison of expected data with recieved data
- Java Pathfinder finds error too
  - When cutting of response hits also the header of message
  - Cca 1200 states traversed

Future work

- Multithreading
  - SimCo performs serialization of processes → cannot detect race conditions or deadlocks
  - Inserting of measurement changes the conditions when more threads are used
- Monitoring of library calls
  - All monitors are only on the components interfaces
- Automated integration testing after update

Thank you for your attention