Property-based testing for Web Services

The PROWESS consortium

EU PROWESS project

Aims to improve testing, particularly for web services, through uptake and use of property-based testing (PBT).

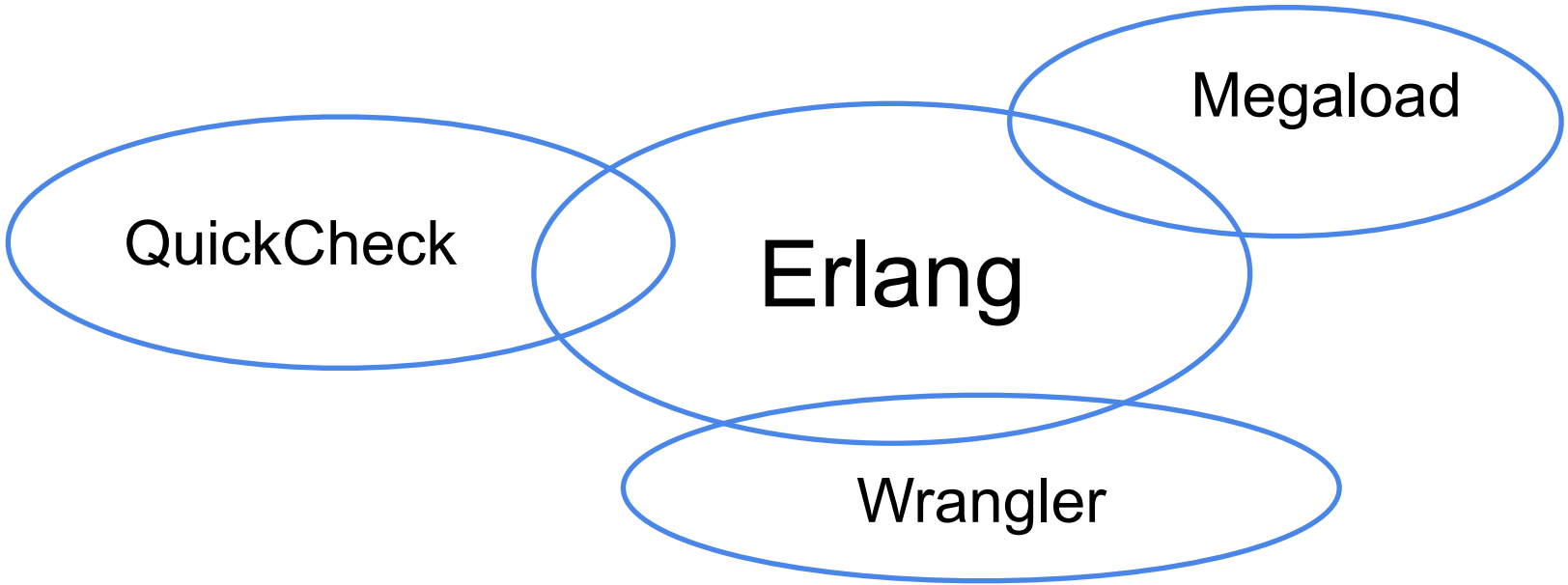
The *QuickCheck* tool for PBT can be used to test web services as well as systems built in Erlang, Java, C, …

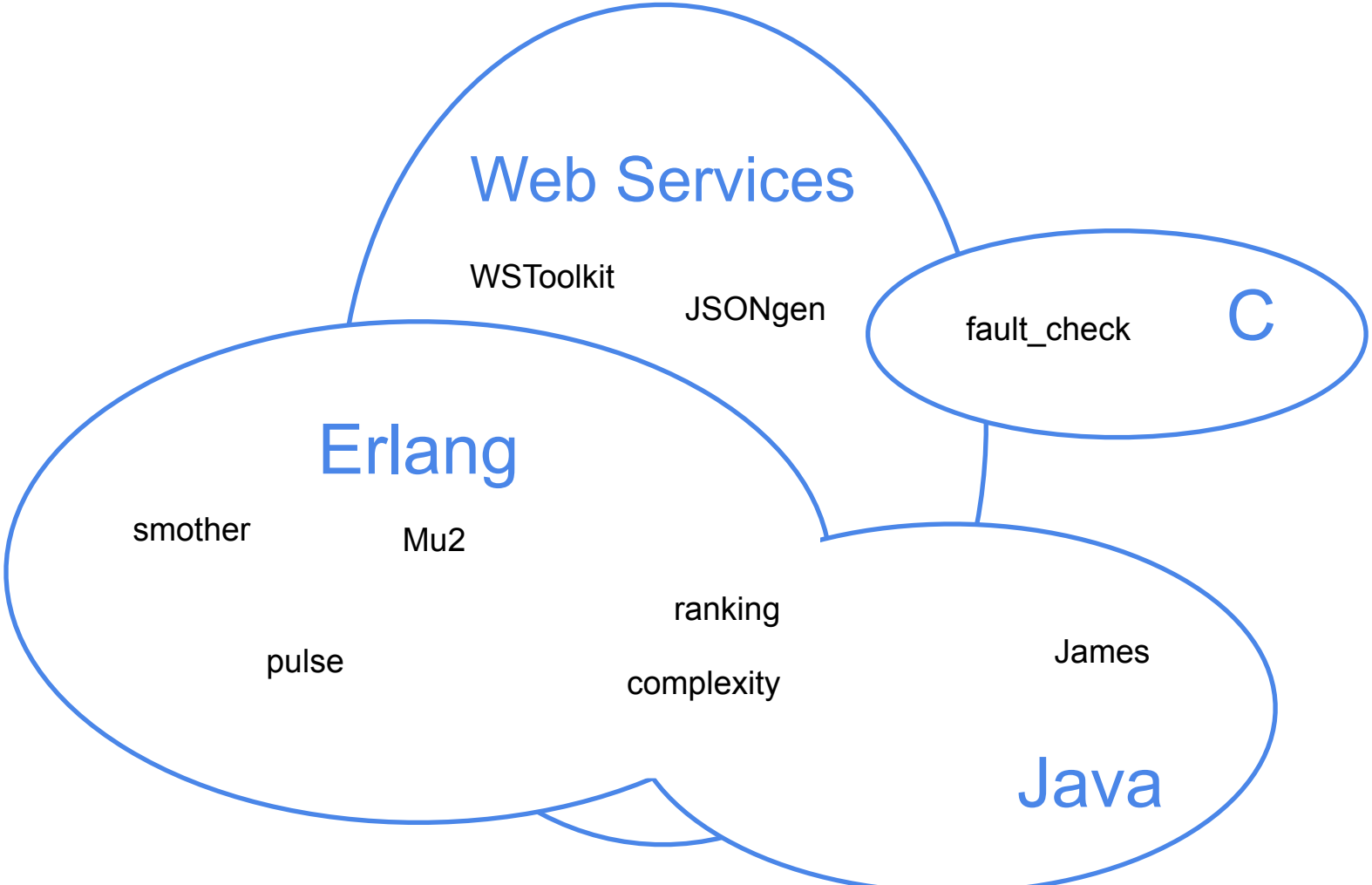
… but system models and properties are written in Erlang.

Consortium

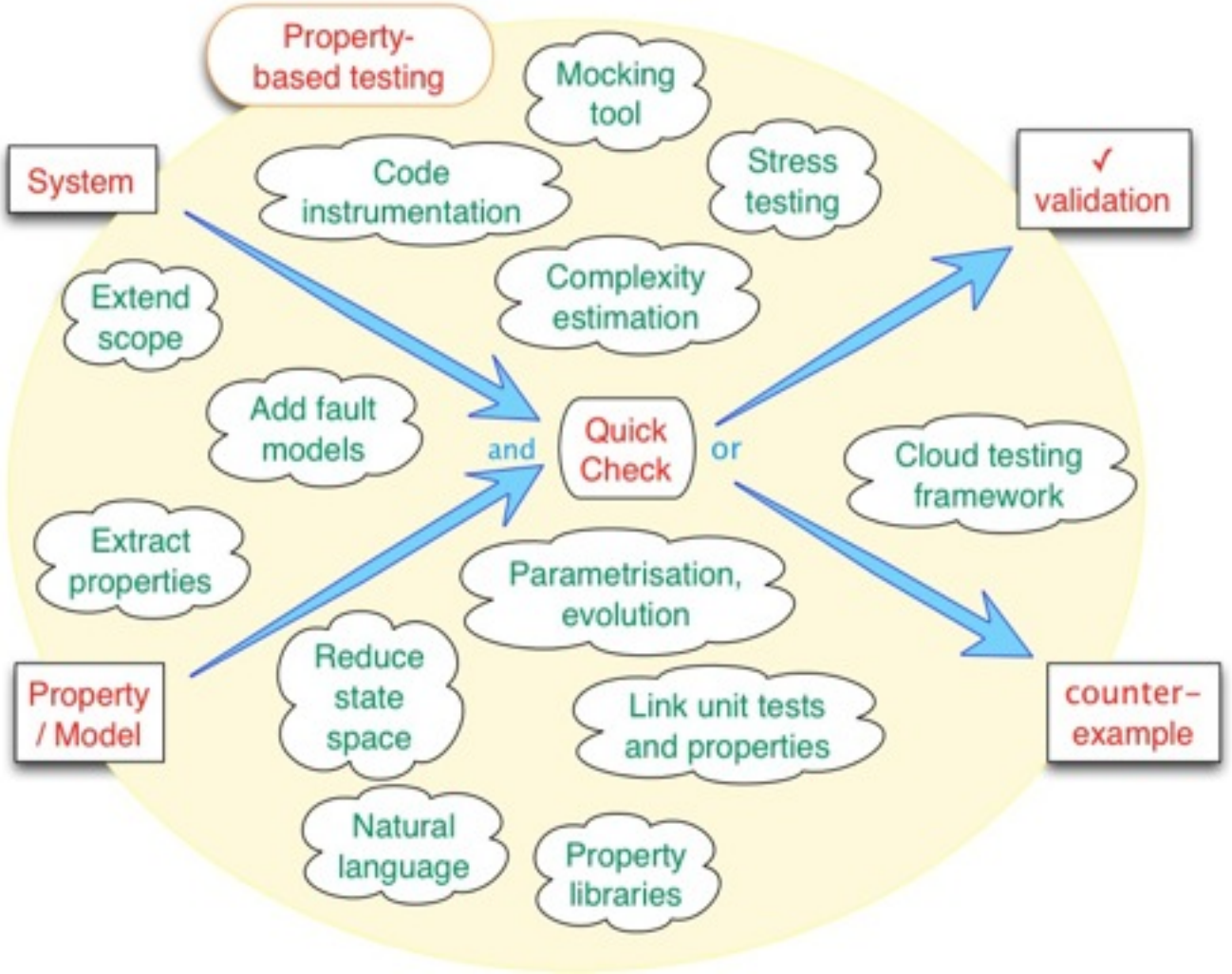
University of Sheffield	- UK
University of Kent	- UK
Chalmers University of Technology	- Sweden
Universidad Politécnica de Madrid	- Spain
University of Coruna	- Spain
Quviq AB	- Sweden
Erlang Solutions Ltd	- UK
Interoud Innovation S.L.	- Spain
SP Technical Research Institute of Sweden	- Sweden

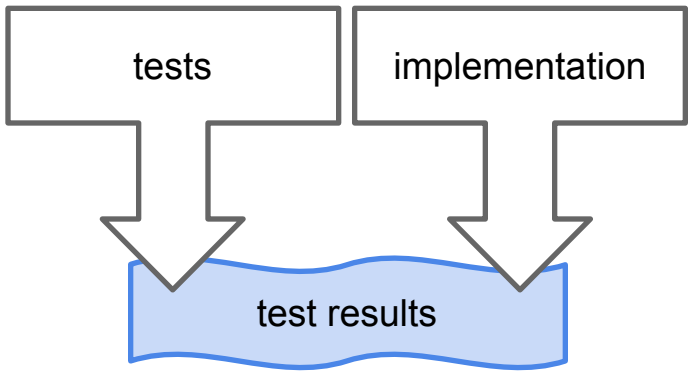
Erlang ecosystem



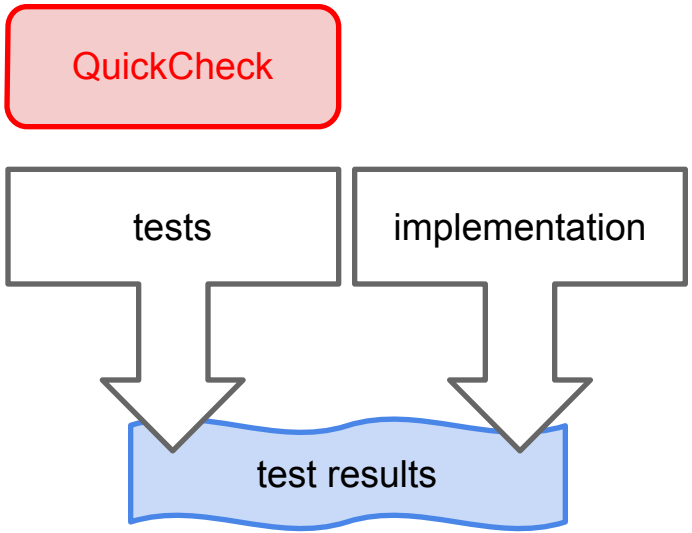

Web Services
WSToolkit
JSONgen
fault_check
C
Erlang
smother
Mu2
ranking
pulse
complexity
James
Java

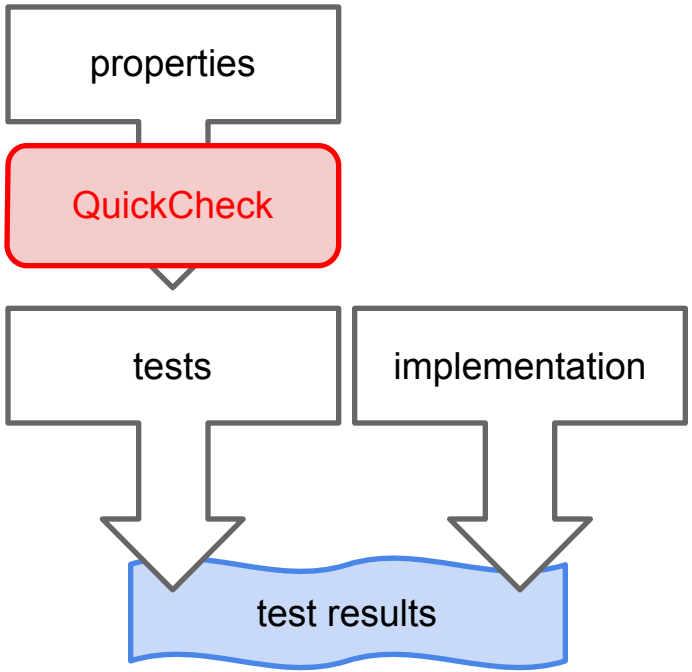
Overview - Big picture



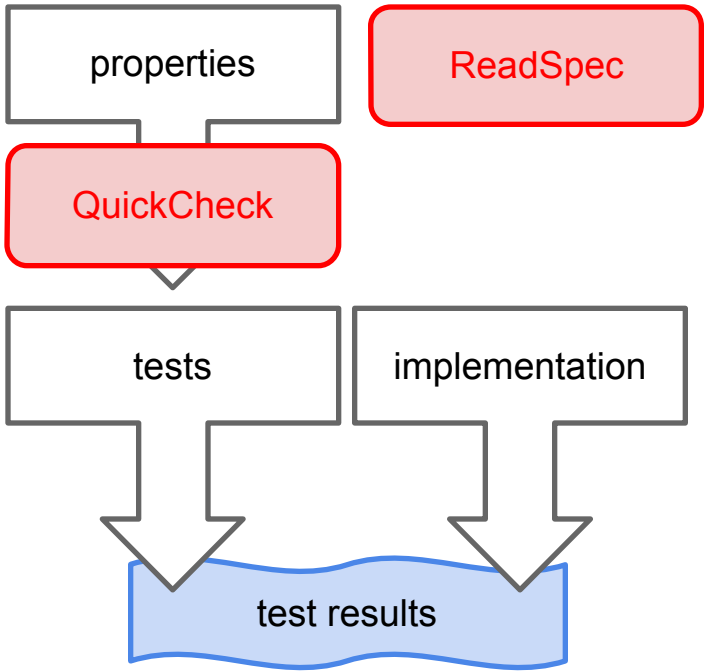

tests
implementation
test results

QuickCheck – random test generation

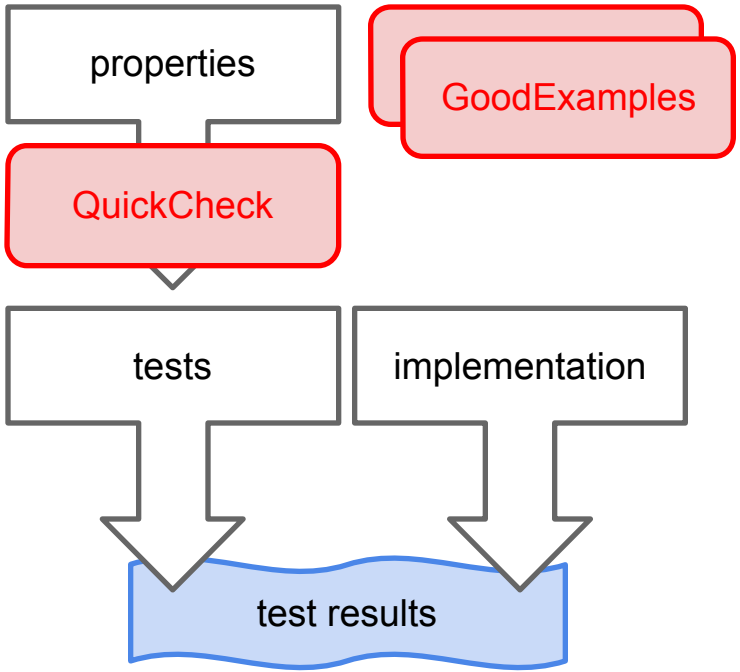


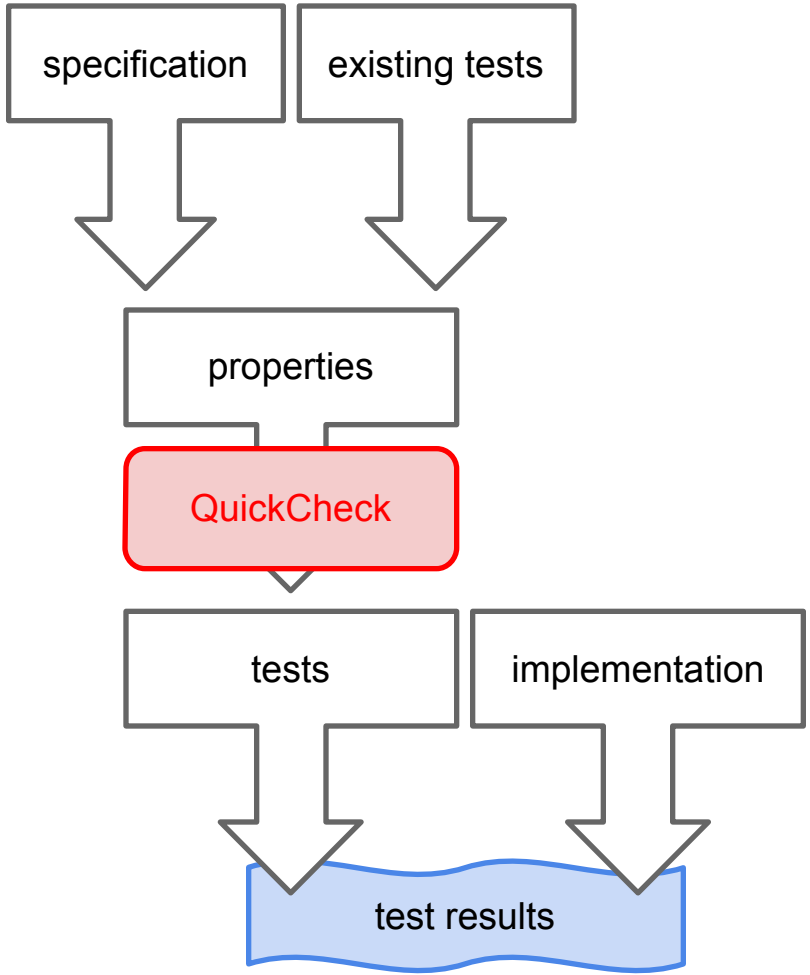

properties
QuickCheck
tests
implementation
test results

ReadSpec – helps you understand properties and models as natural language.

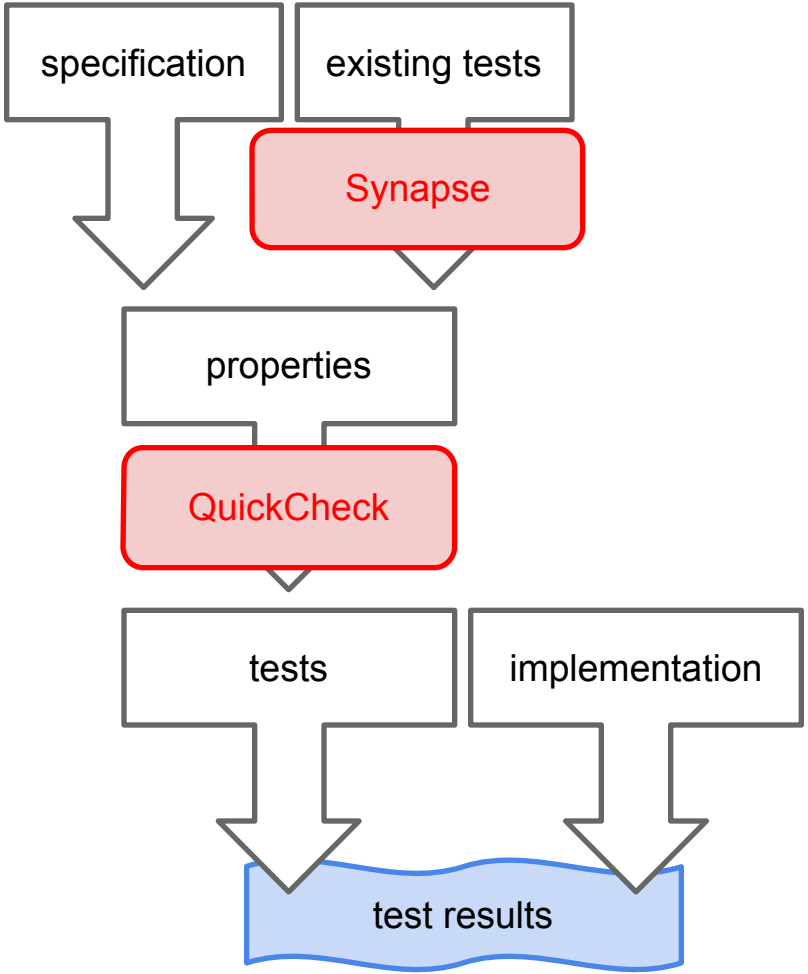


GoodExamples – helps you understand properties by example.

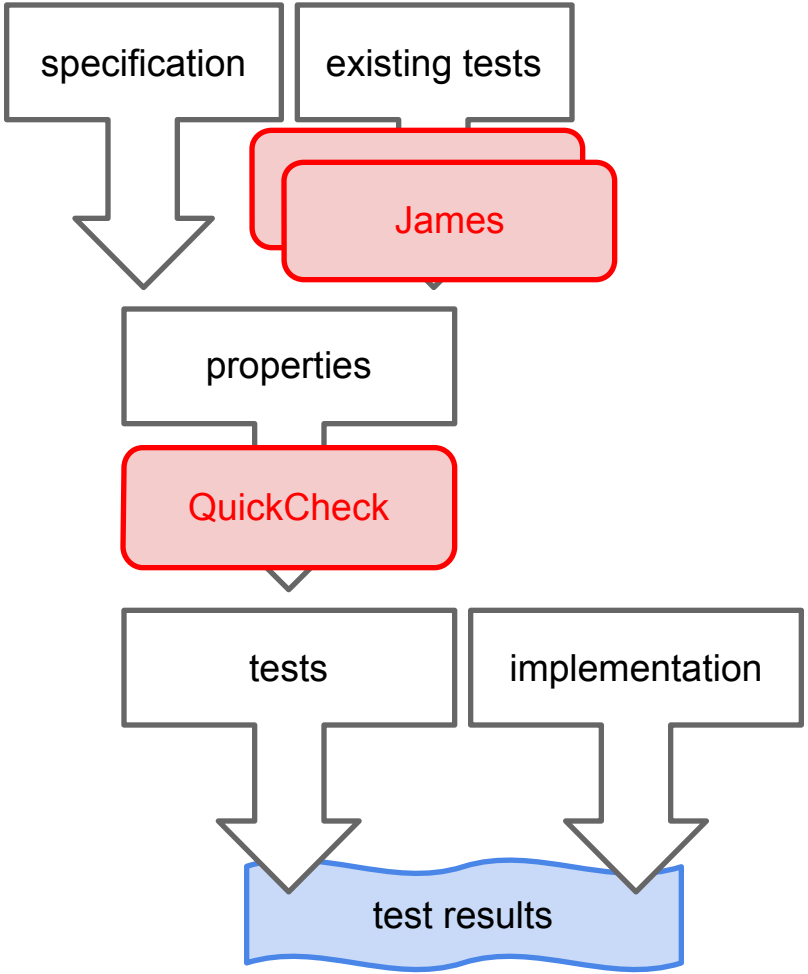



specification
existing tests
properties
QuickCheck
tests
implementation
test results

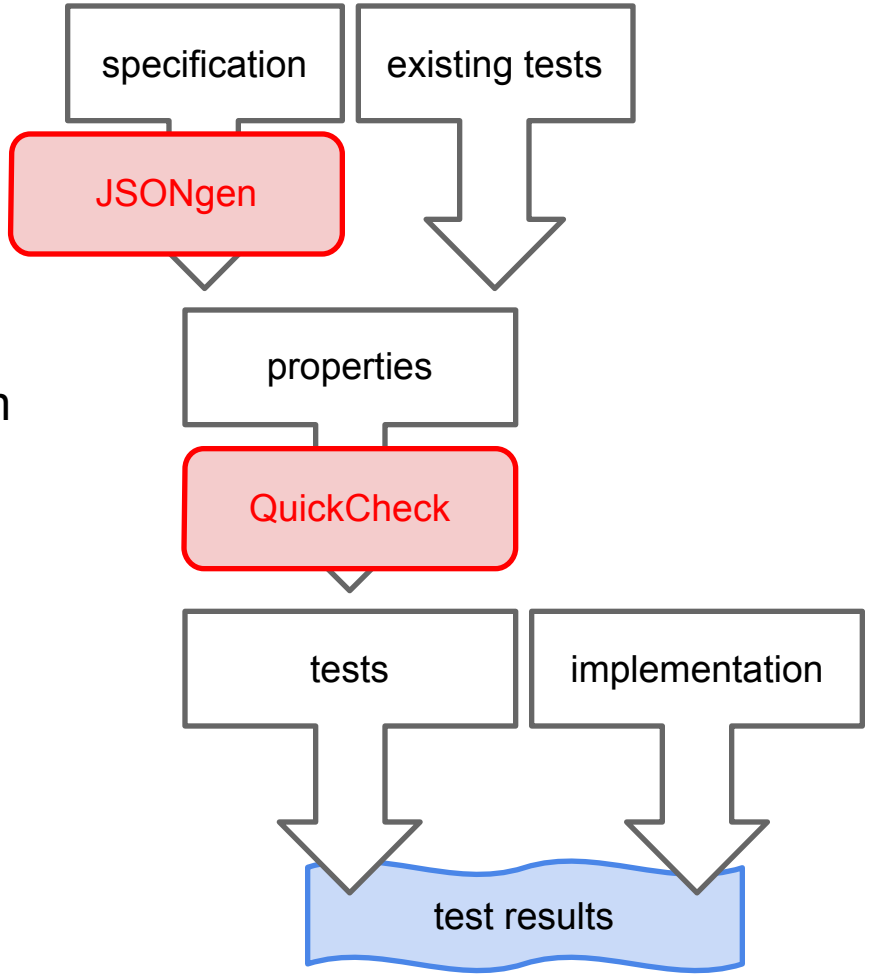
Synapse – visualise systems as FSMs.



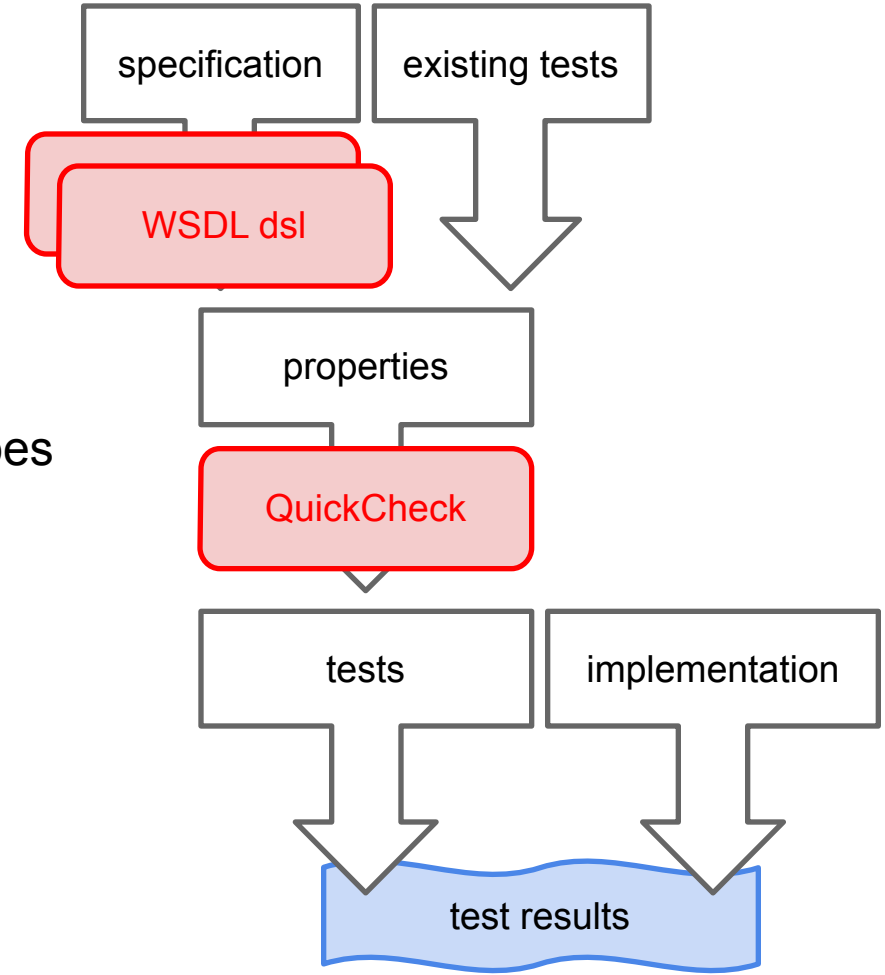
James – infer models for web services.



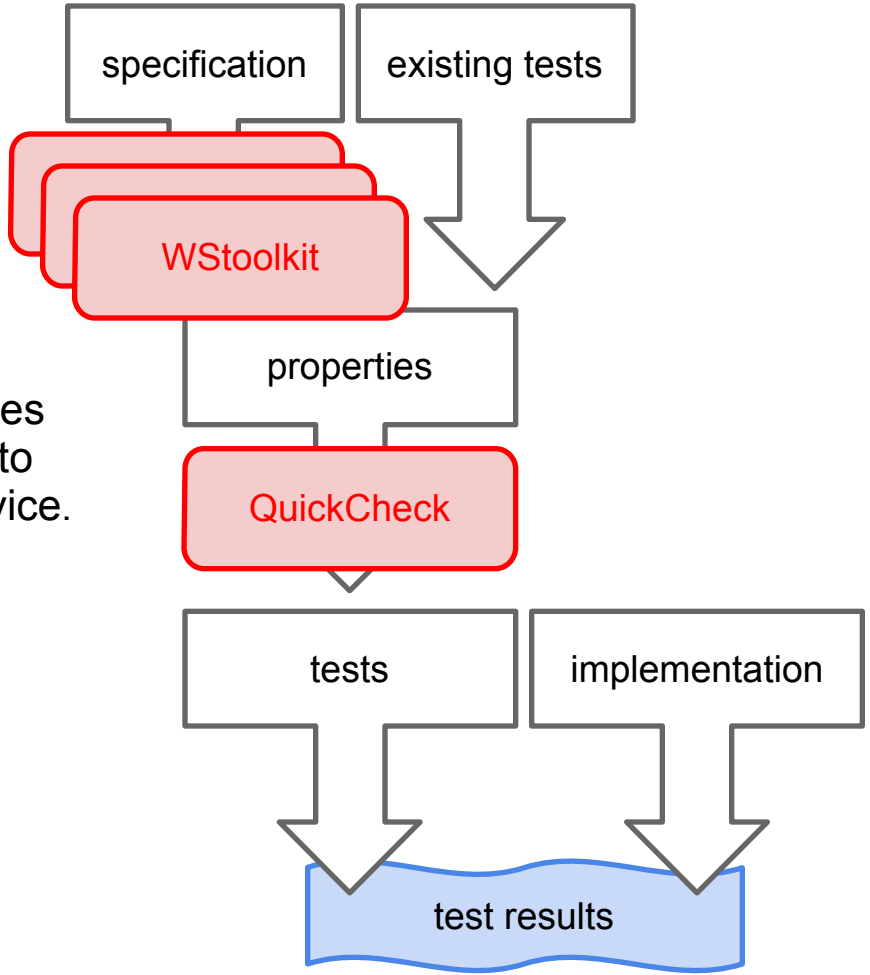
JSONgen – makes QC generators from JSON data.



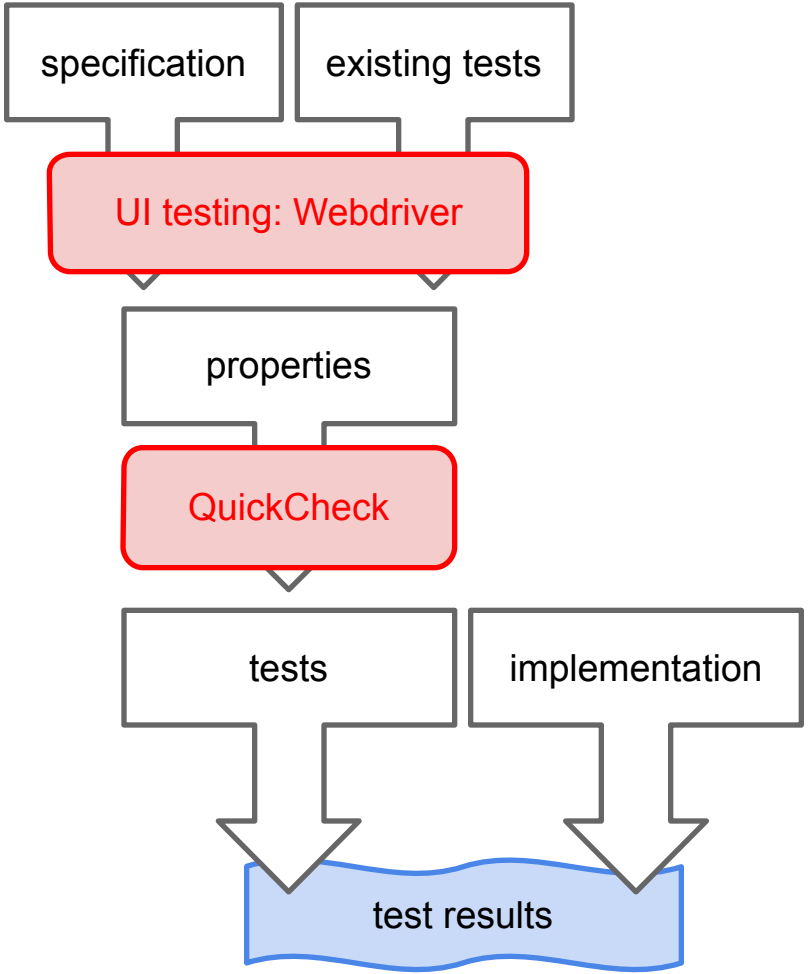
WSDL – how to express WSDL types as QC generators.

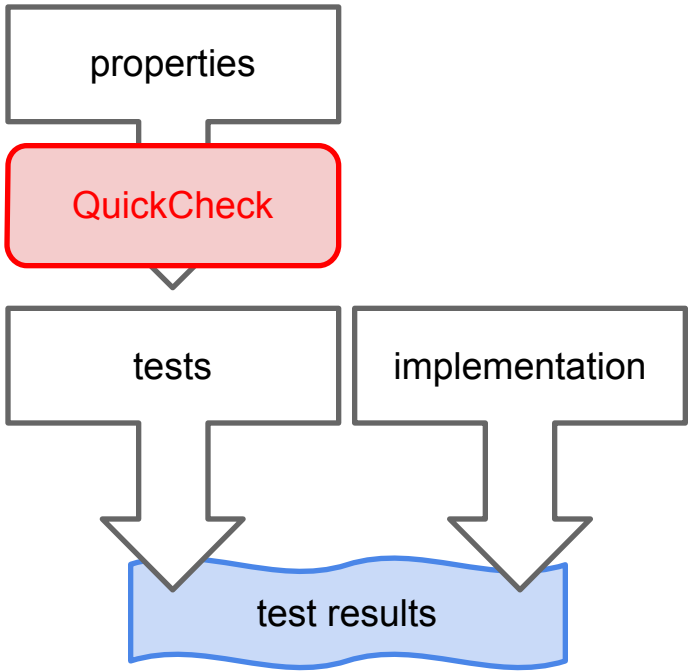


WStoolkit – generates an Erlang interface to underlying web service.

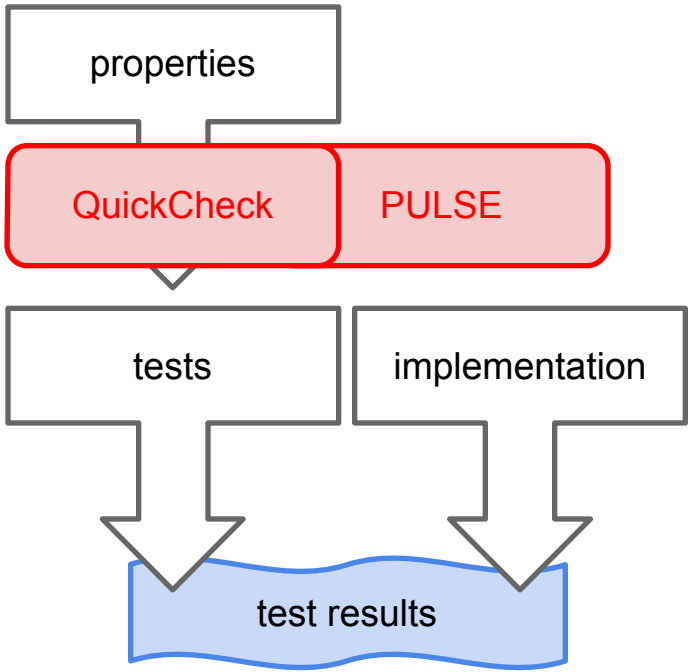


Erlang implementation of Webdriver.

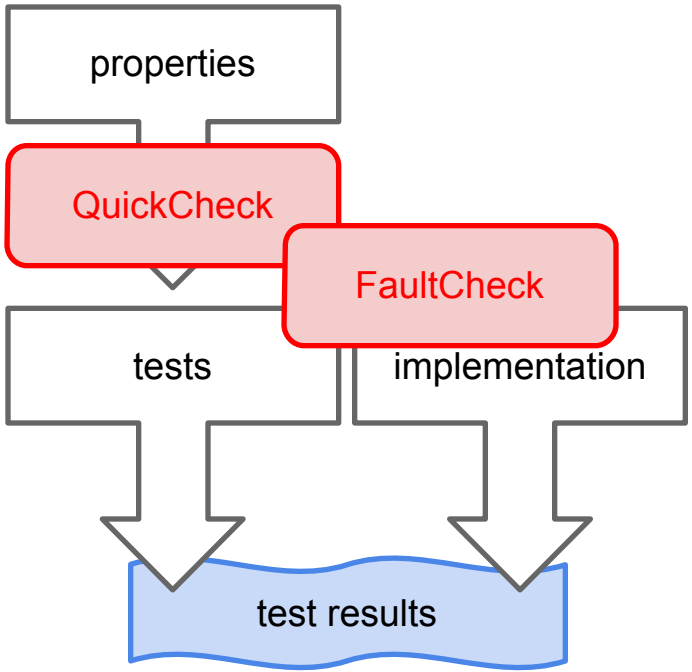



properties
QuickCheck
tests
implementation
test results

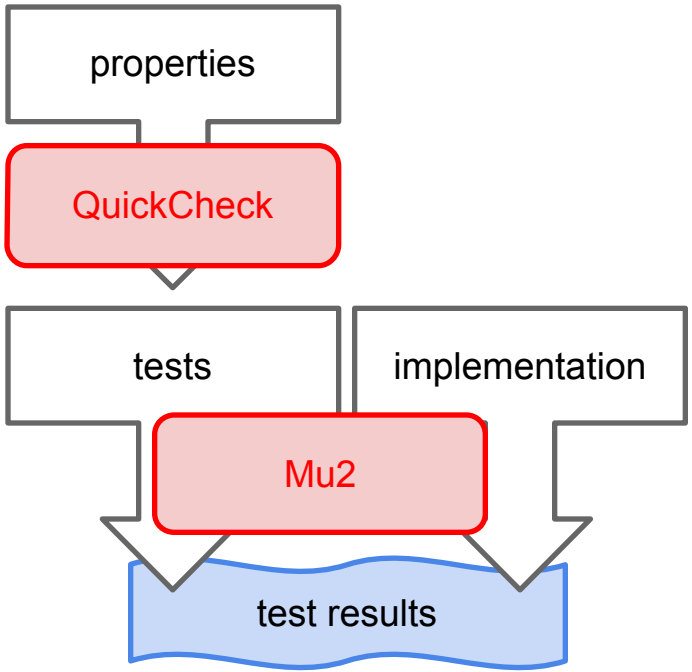
PULSE – additional support for concurrency testing.



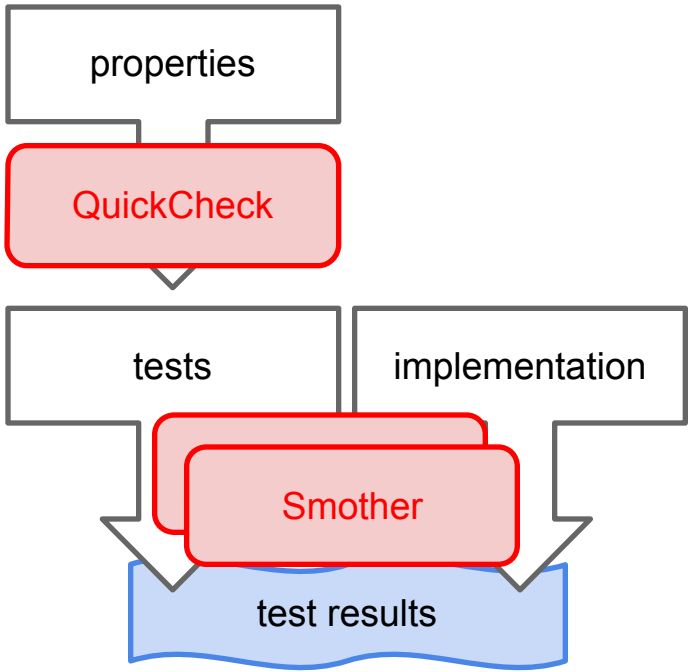
FaultCheck – combines fault-injection and PBT.



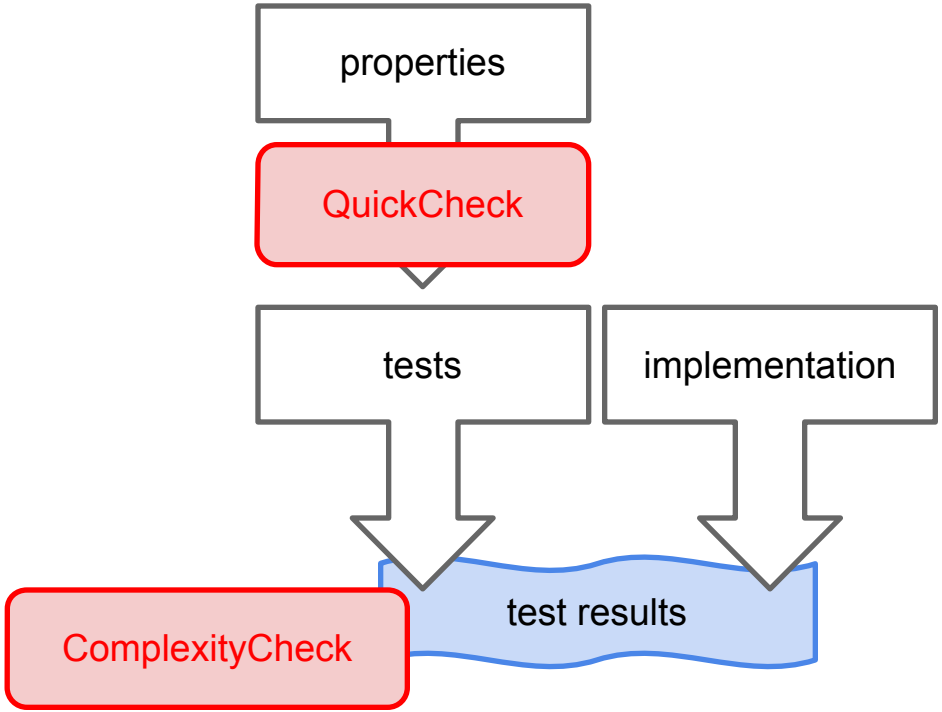
Mu2 – supports mutation Testing.

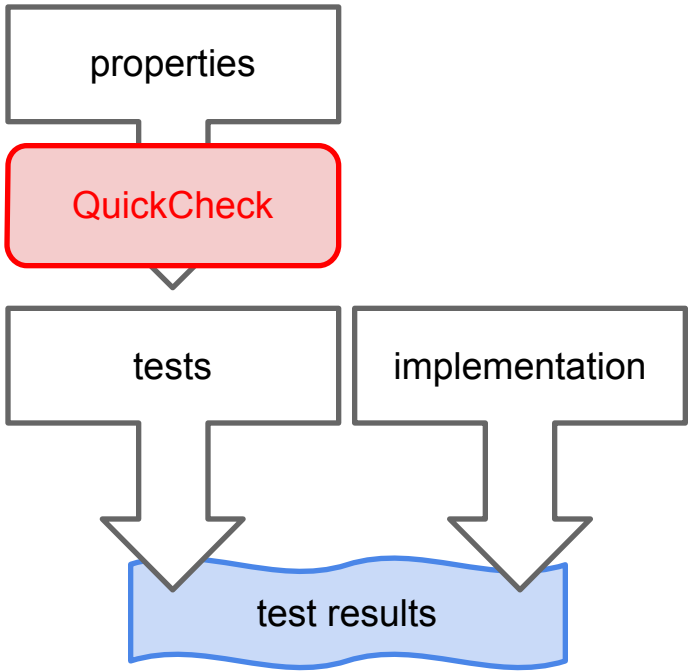


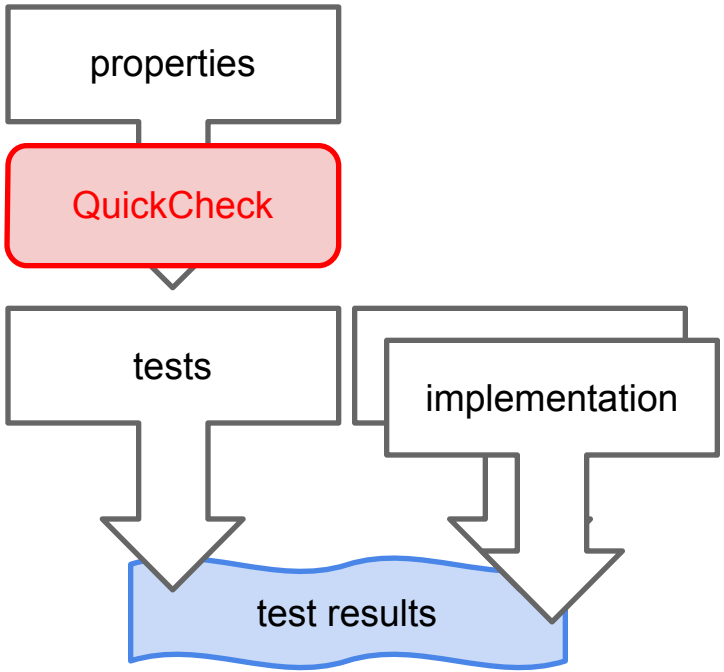
Smother – measures test coverage.

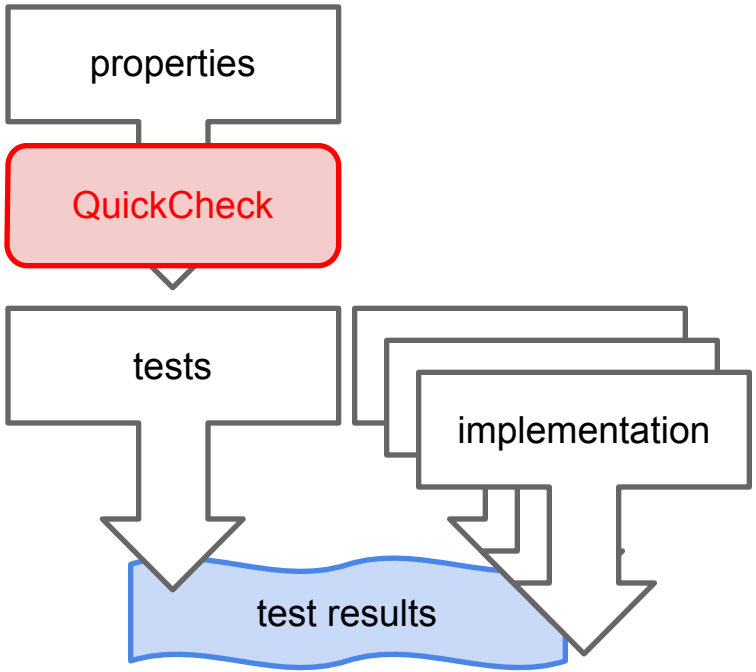


ComplexityCheck – identify scalability issues in code.

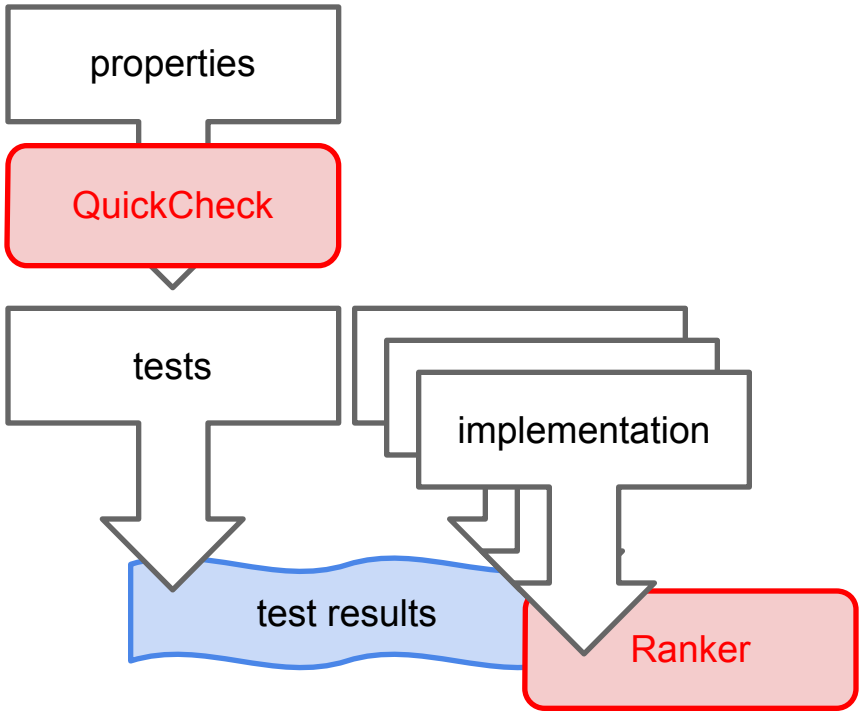



properties
QuickCheck
tests
implementation
test results

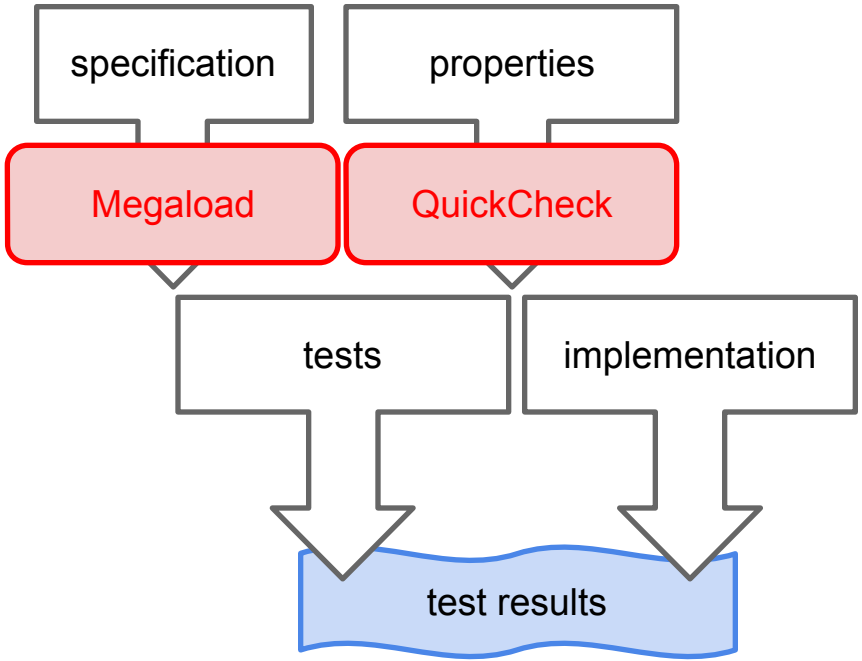

properties
QuickCheck
tests
implementation
test results


properties
QuickCheck
tests
implementation
test results

Ranker – comparing
different implementations.



Megaload – cloud-based testing.



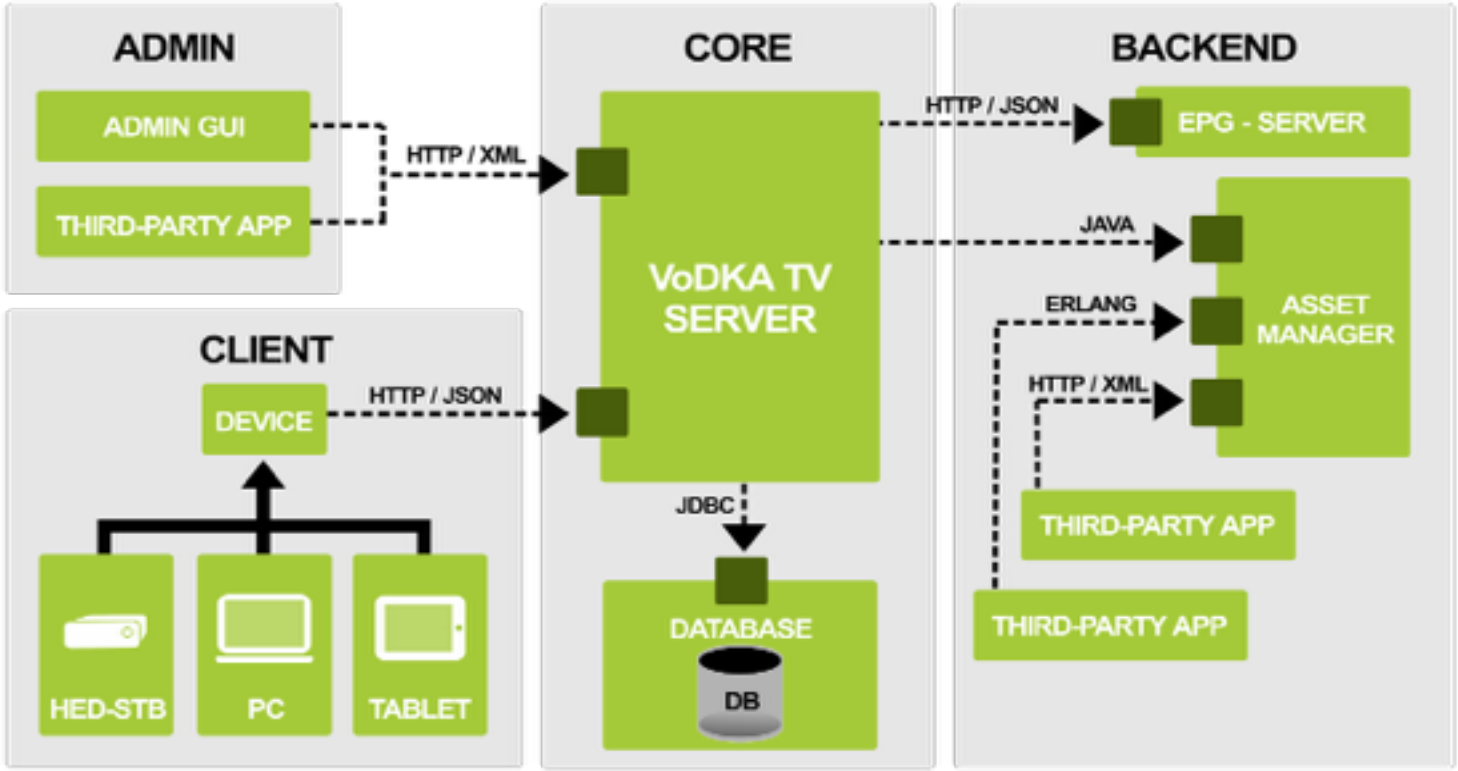
Case study - VoDKATV

Internet-Protocol TV (IPTV) / “Over the top” content (OTT) Cloud Middleware Architecture.

Interactive services for IPTV/OTT environments, eg, hotels.

Runs on a set-top-box (STB) , connected to a TV + remote.

Component-based; on client side: STB, tablet, PC, phone, …


ADMIN
ADMIN GUI
THIRD-PARTY APP
HTTP / XML
CORE
VoDKA TV SERVER
JDBC
DATABASE
DB
BACKEND
HTTP / JSON
EPG - SERVER
JAVA
ERLANG
HTTP / XML
ASSET MANAGER
THIRD-PARTY APP
THIRD-PARTY APP
CLIENT
DEVICE
HTTP / JSON
HED-STB
PC
TABLET

Set-top box

The STB includes

- a portable middleware layer implemented in Erlang,
- a UI layer developed in HTML, JavaScript and CSS (Webkit browser);
- communication between the UI layer and the middleware via a WebSocket-based protocol.

Some APIs respond in XML, others in JSON

Different kinds of authentication for access to the APIs:

- none required,
- authentication with cookies
- authentication with tokens, e.g. expiration time, max # logins per user, …

The toolset

Property-based testing

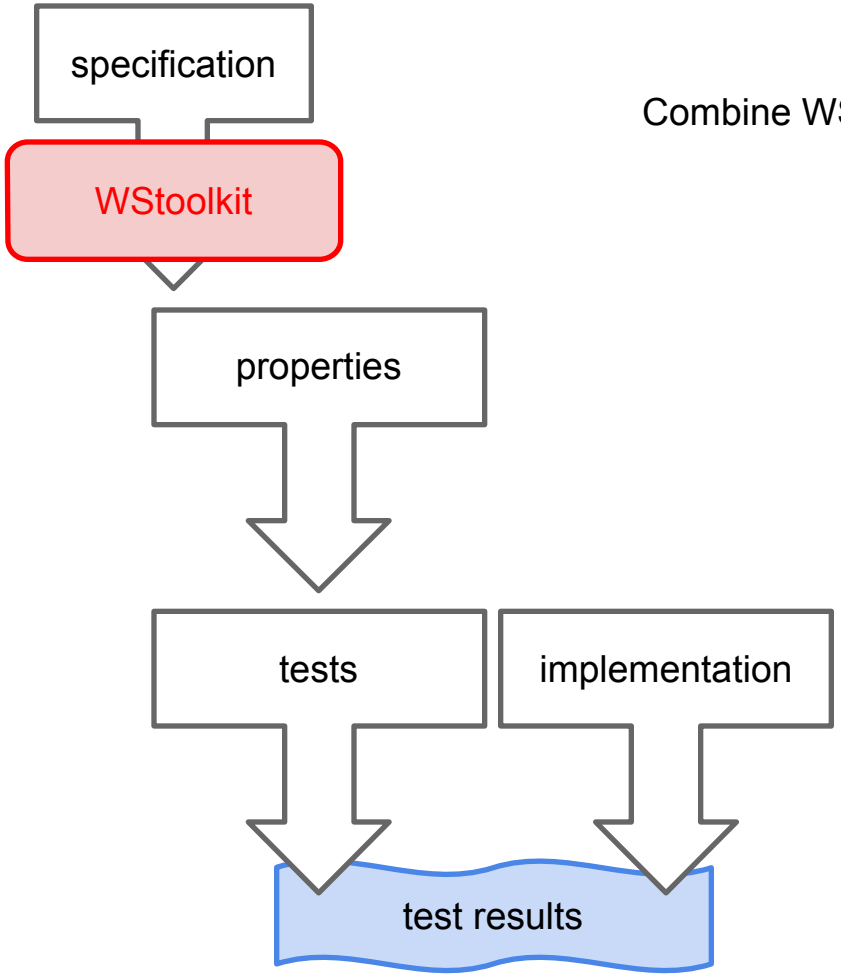
At centre of our ‘ego-system’ is property-based testing with QuickCheck.

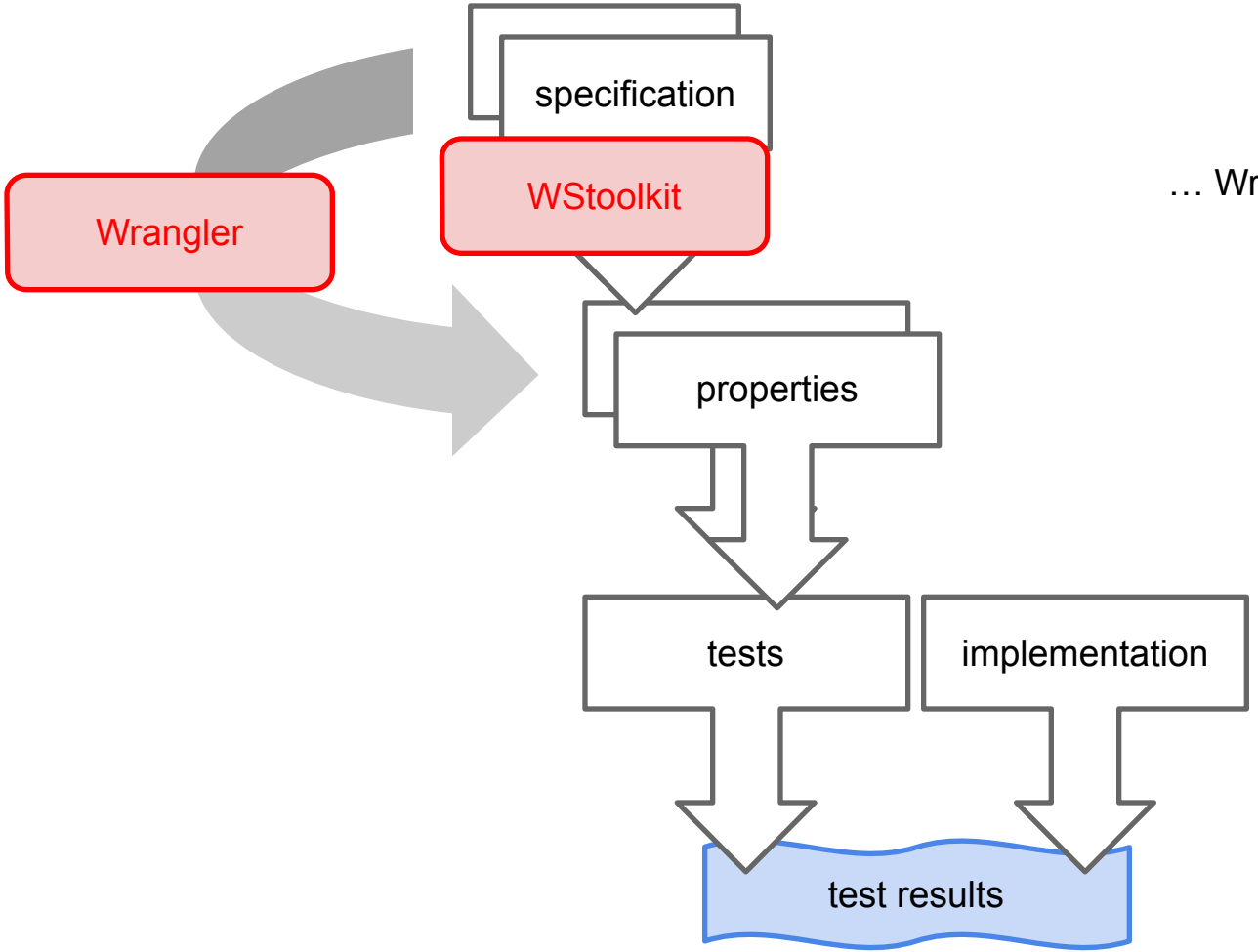
Controlled random test generation from a QuickCheck specification – which describes *properties* of interest.

QuickCheck provides combinators to define properties, observe the distribution of test data, and define test data generators.

Contact: Quviq.

Combine WStoolkit with





… Wrangler to support…

Evolution in PBT with WStoolkit

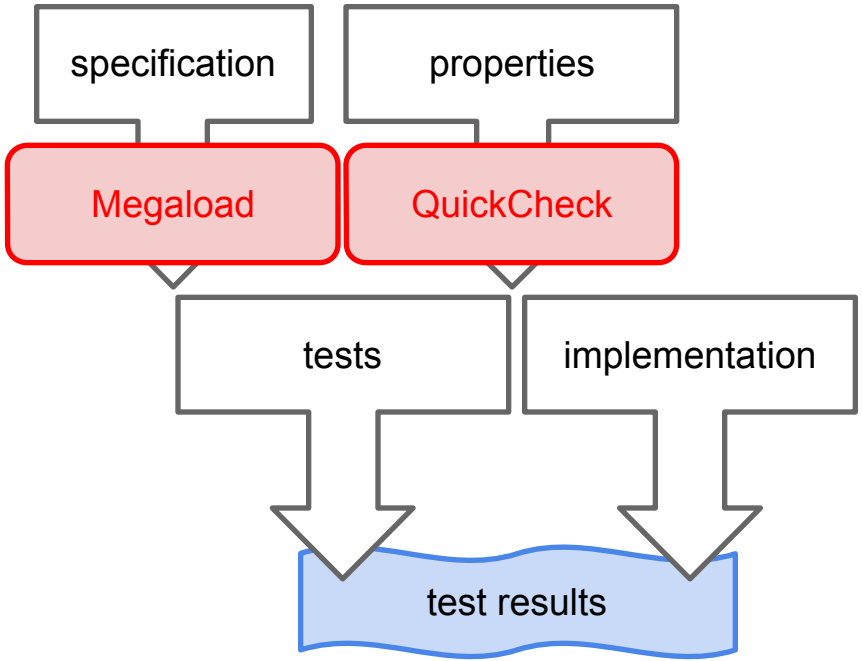
Using Wrangler, Kent's tool for refactoring Erlang systems.

Infer changes between WSDL descriptions …

… from these generate refactoring scripts …

… which automate model evolution as much as possible.

Contact: *Kent*


specification
properties
Megaload
QuickCheck
tests
implementation
test results

Megaload – Load testing VoDKA

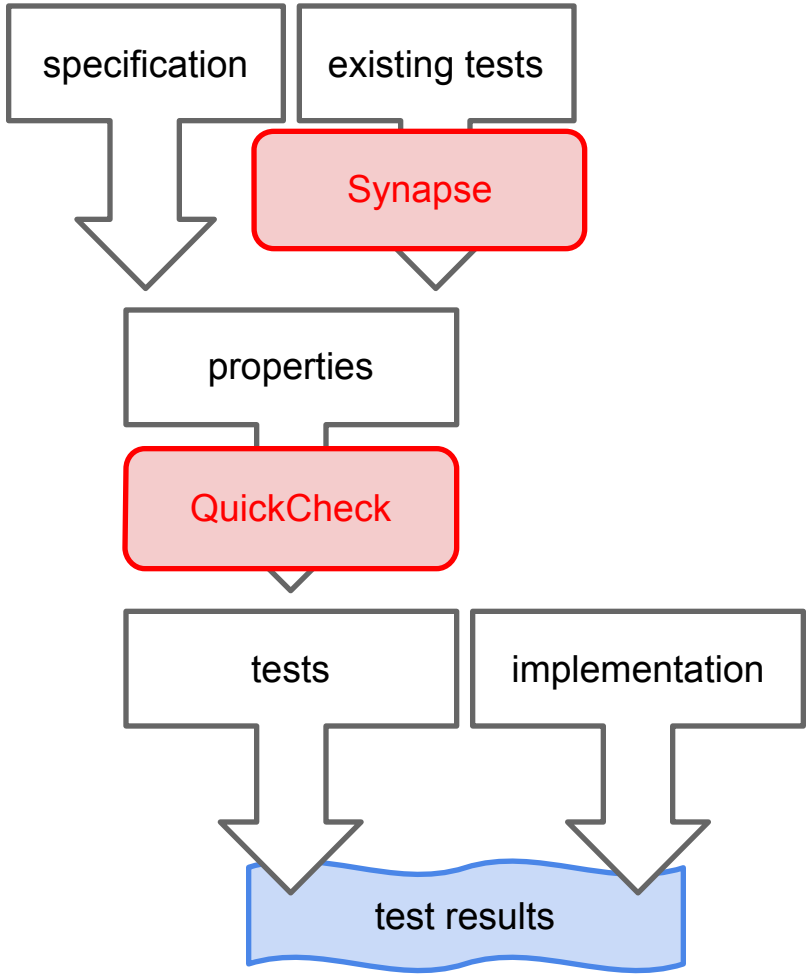
Cloud-based load testing of systems.

Megaload: loads, monitors and presents results.

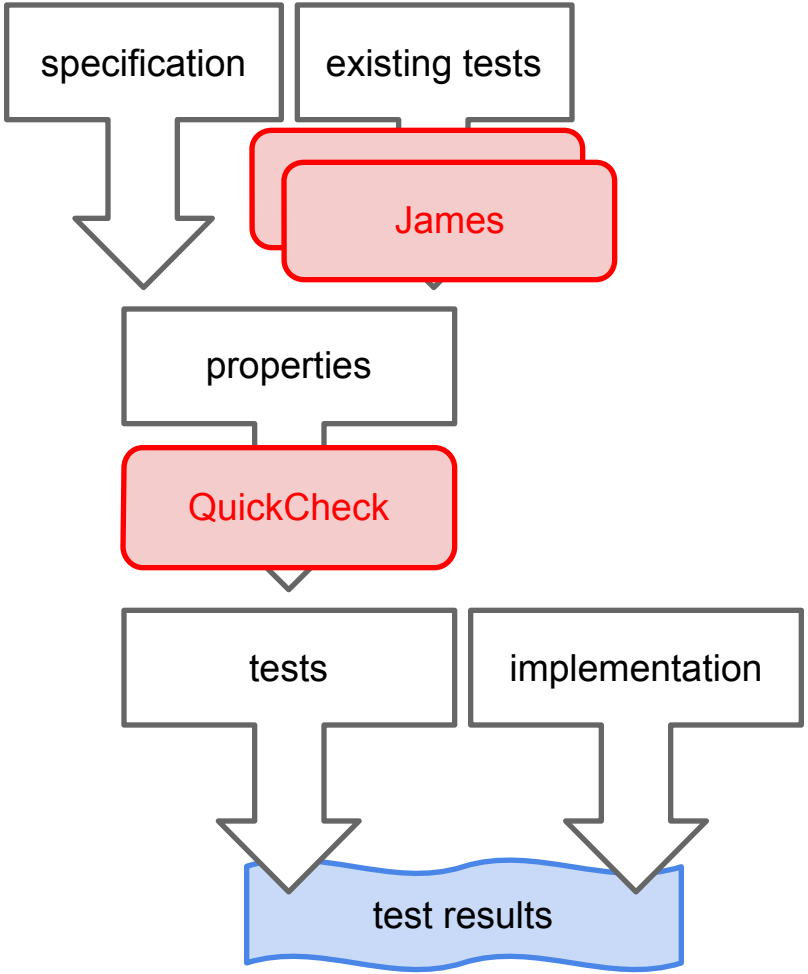
Generating load profiles …
… and shrinking to minimal (counter-) examples in the most load-effective way.

Contact: *ESL*

Synapse – visualise systems as FSMs.



James – infer models for web services.



How to develop properties for a system. Two tools:

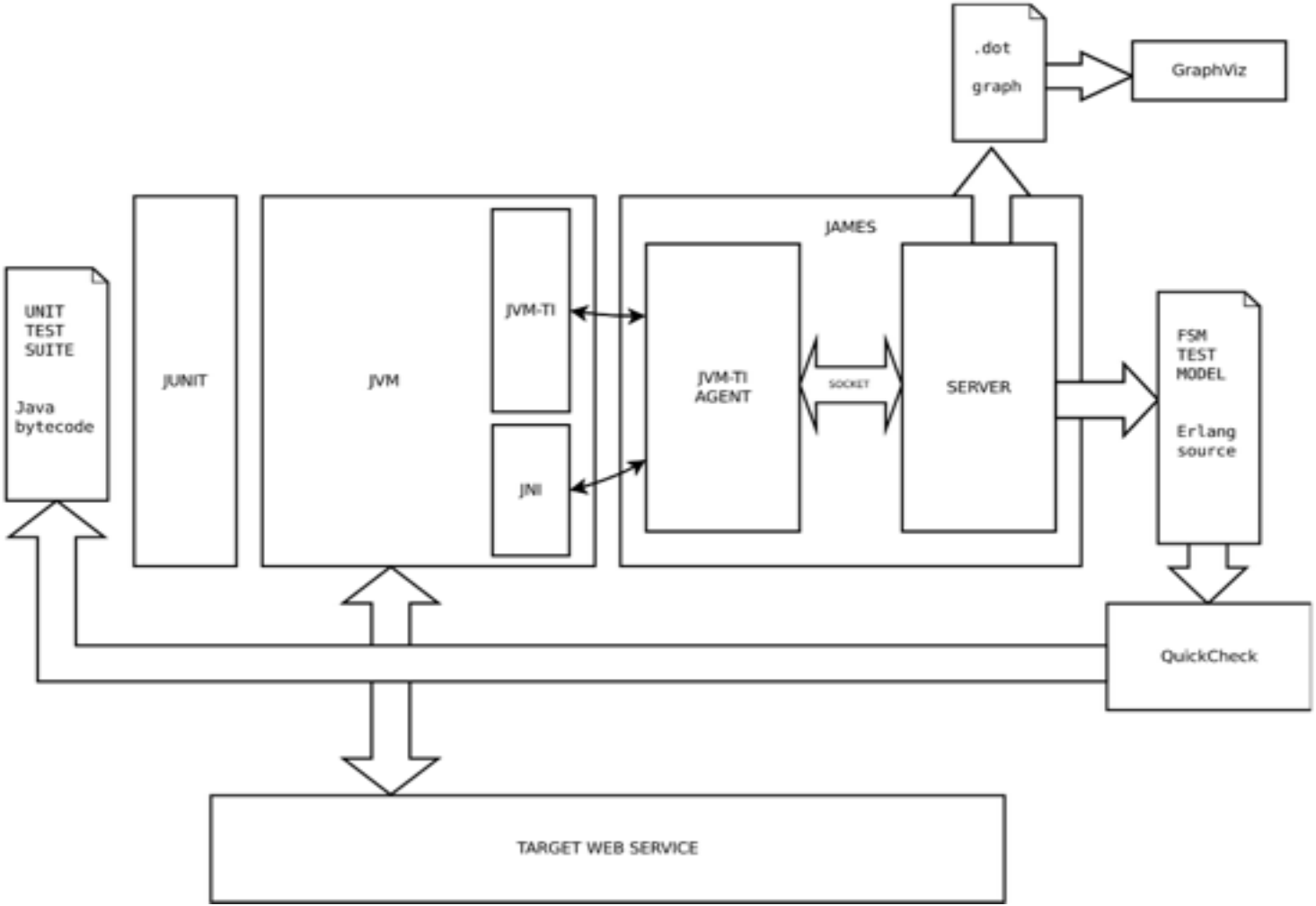
- *James* – infer models for web services from unit tests written in Java, using JUnit.
- *Synapse* - infer FSMs from systems, and visualise the difference between models / systems.

James

New JUnit tests from existing tests, by model inference.

Track a combination of data- / control-flow information …
… extracted from running the test suite on the SUT
… run the tests on the Java VM
… track information using C++ agent and JVM-TI API

Contact: *Kent*


.dot
graph
GraphViz
JAMES
UNIT
TEST
SUITE
Java
bytecode
JUNIT
JVM
JVM-TI
JNI
JVM-TI
AGENT
SOCKET
SERVER
FSM
TEST
MODEL
Erlang
source
QuickCheck
TARGET WEB SERVICE

James

Track and send to an Erlang server:

- the execution order of the calls in the JUnit tests, and
- how objects are reused.

Server generates a model … visualised through GraphViz.

Translate model into QuickCheck … then generate new tests, that can be added to the original test suite.

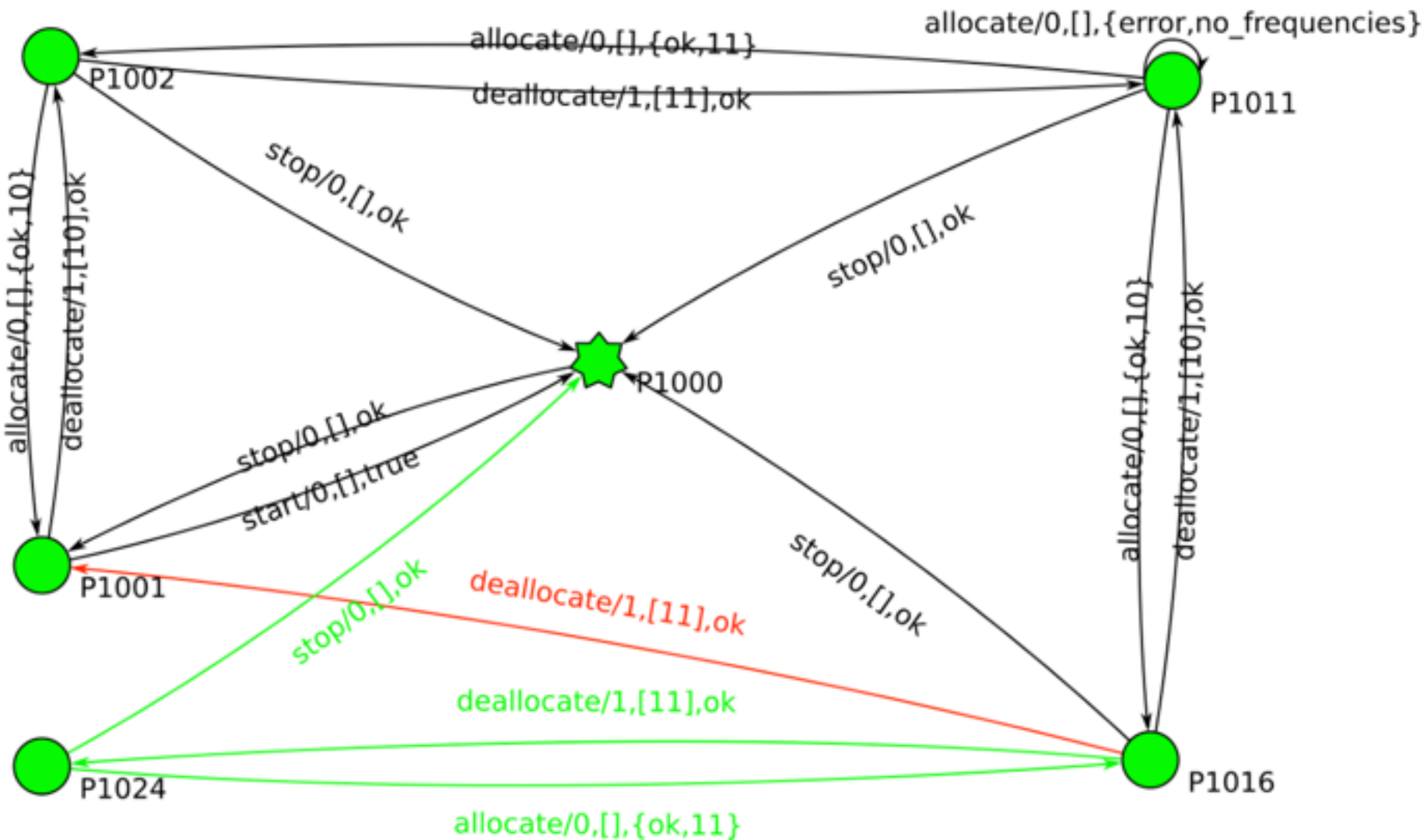
Synapse

An Erlang interface to grammar inference tools.

Synapse interfaces to the *StateChum* tool for passive and active inference of FSM models, as well as:

- active and passive learning,
- model differencing, and
- FSM and difference visualisation.

Contact: *Sheffield*


allocate/0,[],{error,no_frequencies}
allocate/0,[],{ok,11}
deallocate/1,[11],ok
P1002
P1011
stop/0,[],ok
stop/0,[],ok
allocate/0,[],{ok,10}
deallocate/1,[10],ok
P1000
allocate/0,[],{ok,10}
deallocate/1,[10],ok
stop/0,[],ok
start/0,[],true
P1001
stop/0,[],ok
deallocate/1,[11],ok
stop/0,[],ok
deallocate/1,[11],ok
P1024
P1016
allocate/0,[],{ok,11}

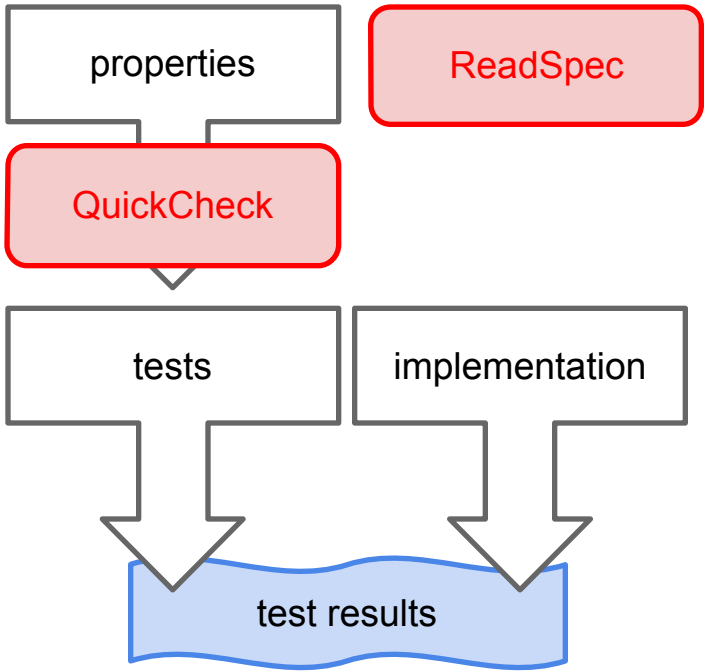
Understanding properties and models

Synapse tool allows users to visualise differences between variants of models / systems as FSMs.

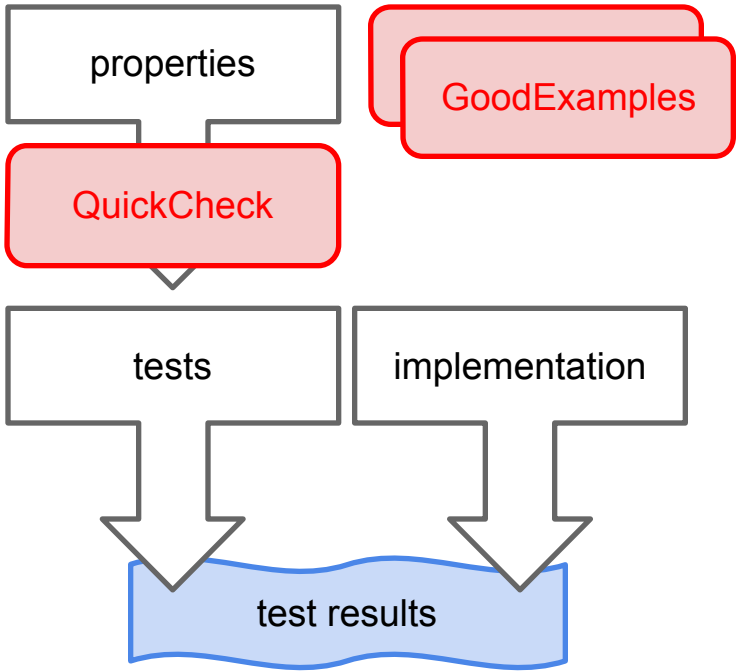
ReadSpec to render QuickCheck models in (semi-)natural language.

GoodExamples tool to make the meaning of a property more concrete by viewing it as a set of unit tests.

ReadSpec – helps you understand properties and models as natural language.



GoodExamples – helps you understand properties by example.



ReadSpec uses QuickCheck to automatically generate semi-natural language descriptions of QuickCheck properties and QuickCheck state machine models.

Example: `simple_eqc.erl` contains a property to test the delete operation of the lists module:

Contact: *UDC*

```
?FORALL({I,L}, {int(), list(int())},
                not lists:member(I,lists:delete(I,
              L)))
```

FEATURE: Simple QuickCheck properties
SCENARIO: Deleting an integer from a list should result in a list that does not contain that integer.
GIVEN I have the integer 19
AND I have the list [7, -24, -18, 17, -8, -9, -8]
THEN lists:member(19, lists:delete(19, [7,-24,-18,17,-8,-9,-8]))
IS FALSE.

GoodExamples tool

It can be hard to tell what a property tests…

properties - powerful and general;

unit tests - easy to understand but specific.

GoodExamples - makes the meaning of a property more concrete by viewing it as a set of unit tests.

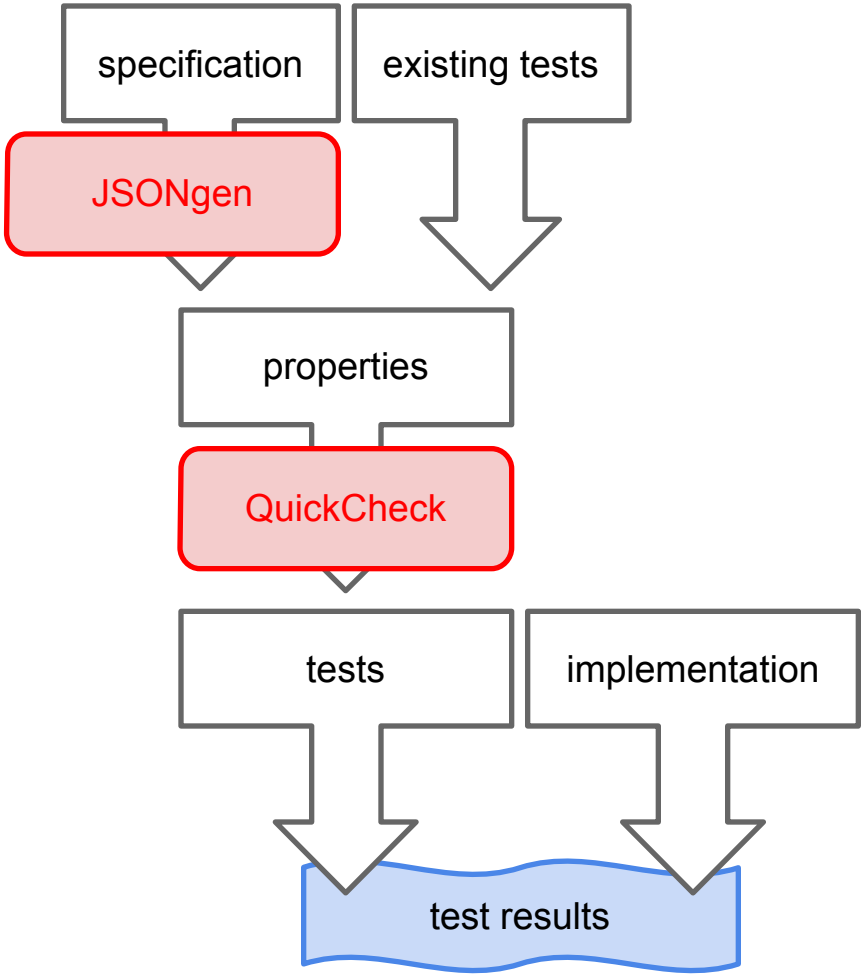
Contact: *Chalmers*

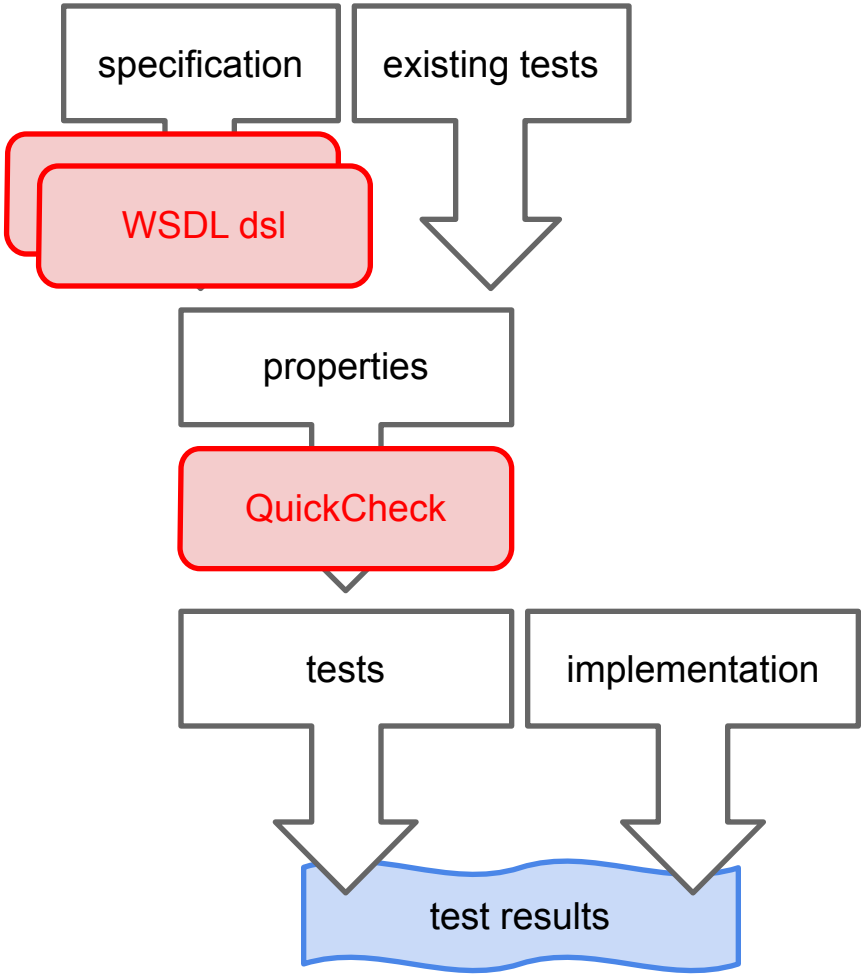
Support for Web Services

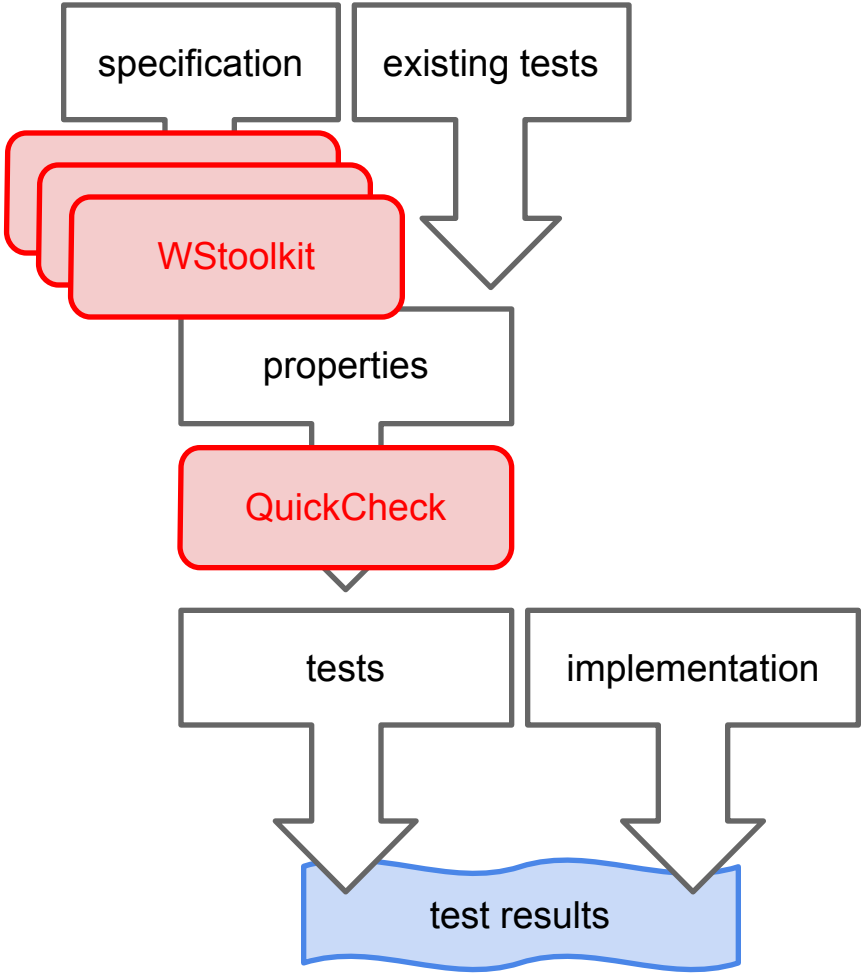
Tools to support data generation for web services models:

JSONgen is a library for generating QuickCheck generators from descriptions of JSON data using JSON schemas, and for automatically exploring and testing JSON web services.

wsdl_dsl is a QuickCheck library that implements a domain specific language which re-uses the WSDL syntax to allow users to express WSDL types as QuickCheck generators.


specification
existing tests
JSONgen
properties
QuickCheck
tests
implementation
test results


specification
existing tests
WSDL dsl
properties
QuickCheck
tests
implementation
test results


specification
existing tests
WStoolkit
properties
QuickCheck
tests
implementation
test results

JSONgen – convert and explore

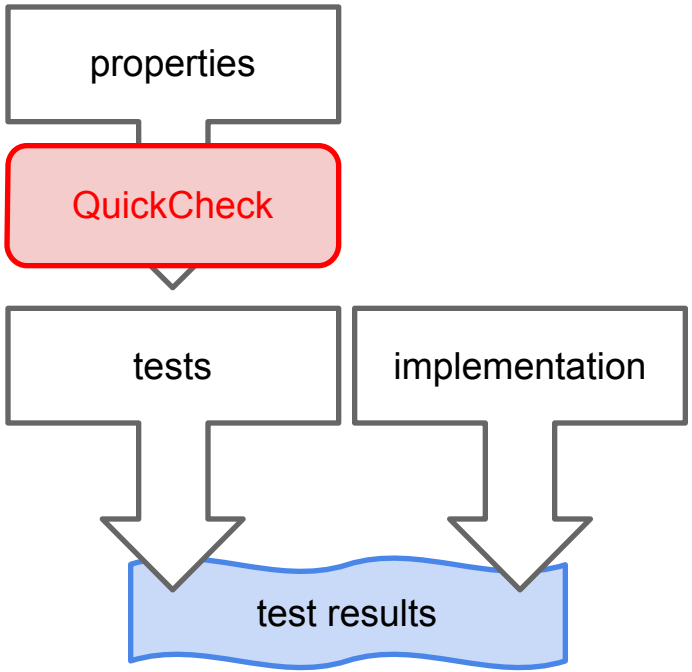
Convert JSON schema to mochijson2 Erlang term.

Convert JSON schema into a QuickCheck generator.

Convert JSON data value in mochijson2 format to text

Explore and test a JSON based web service using the web links / data types embedded in the JSON schema args.

Can tailor the actions with a QuickCheck state machine.

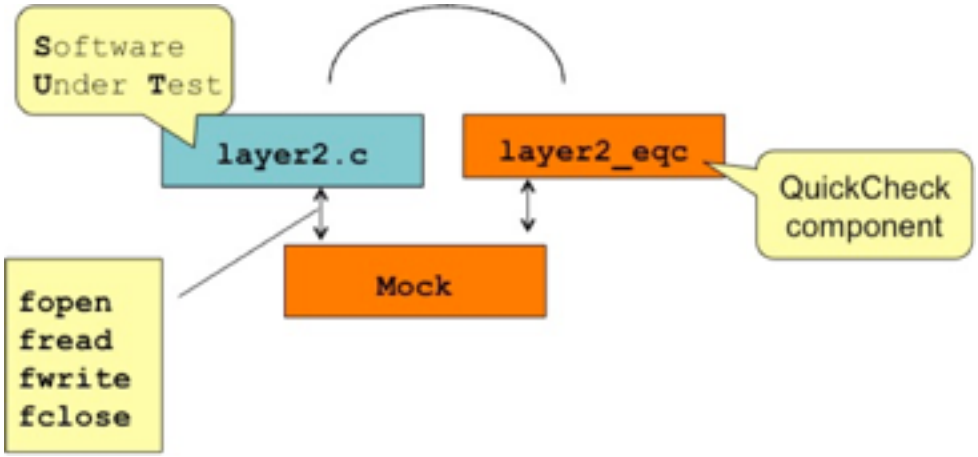

properties
QuickCheck
tests
implementation
test results

Scaling PBT

Model using *components* instead of a single model.

Library for *mocking* the behaviour of callout components.

Clustered system resulting from the component models.



QuickCheck “by hand”: run QC, fix bug, repeat …

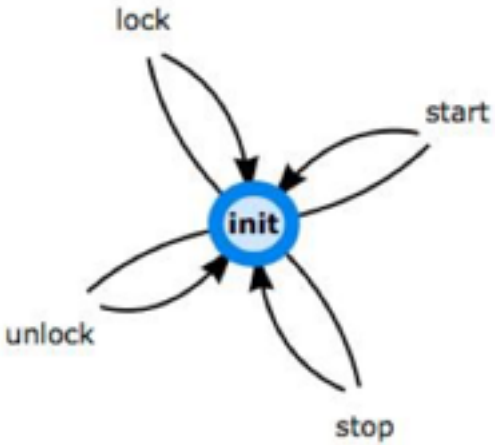
With MoreBugs, can find “all” bugs at once, through

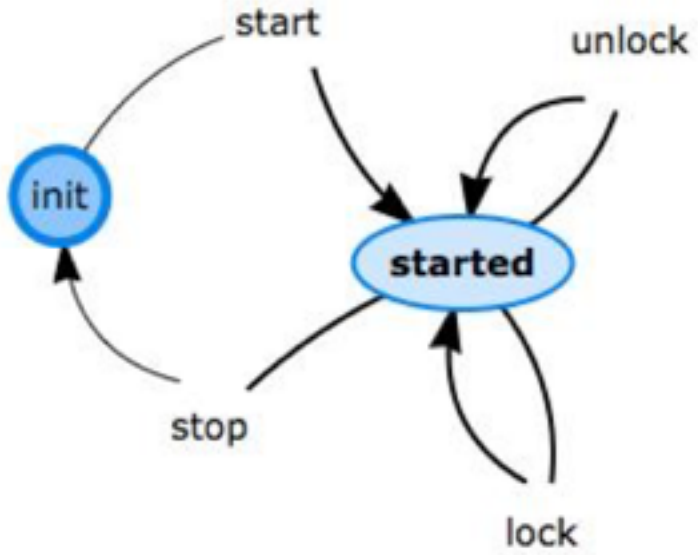
- find bug,
- generalise
- modify generator to avoid it

and repeat …

Contact: *Chalmers*

Support for graphical editing

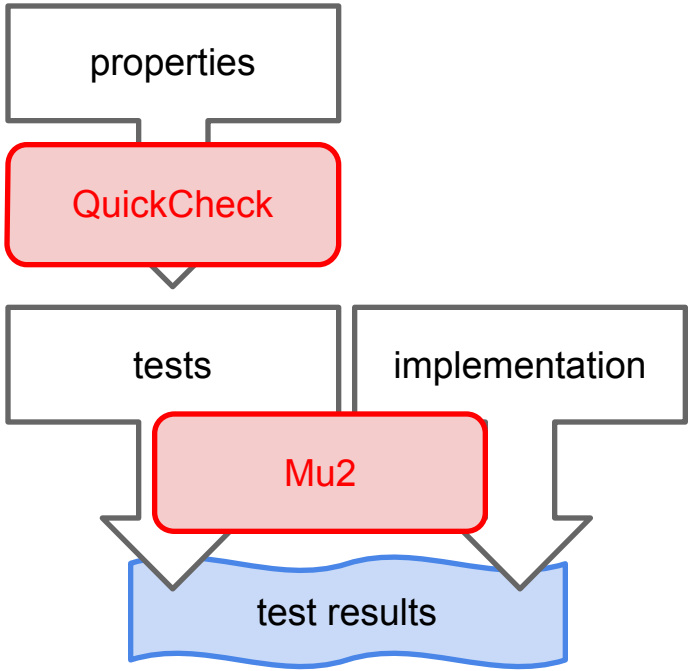




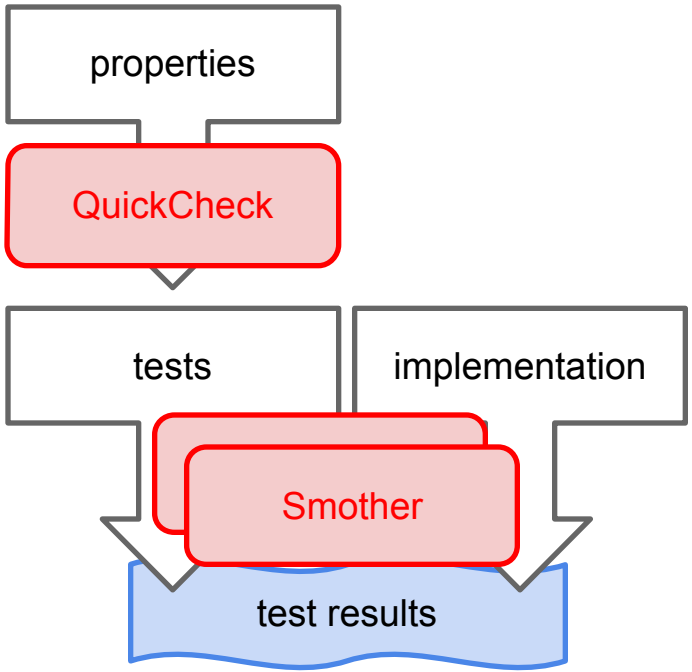
Contact: *Quviq*

How good is your test suite?

Mu2 – supports mutation Testing.



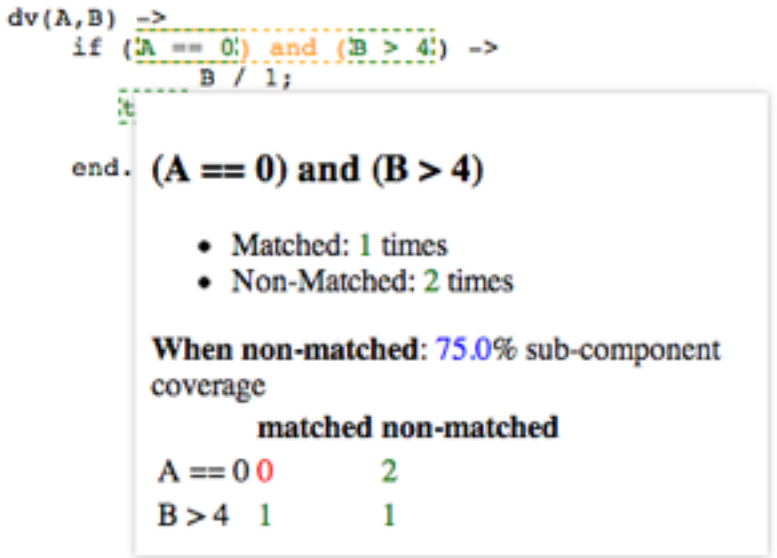
Smother – measures test coverage.



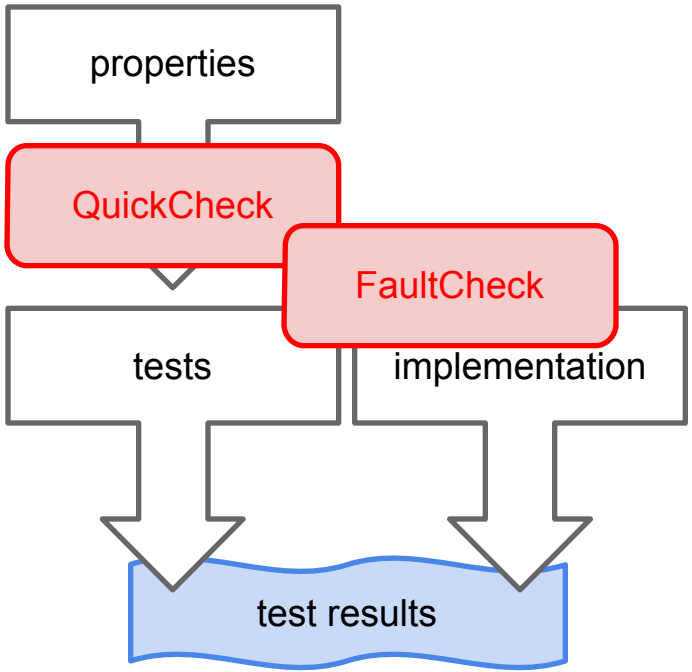
Validating quality of test suites

Smother used to assess the MC/DC coverage of a test suite.

Mu2 supports mutation testing – if I make small changes to the code, can I spot these ‘errors’ with my test suite?



Contact: *Sheffield*

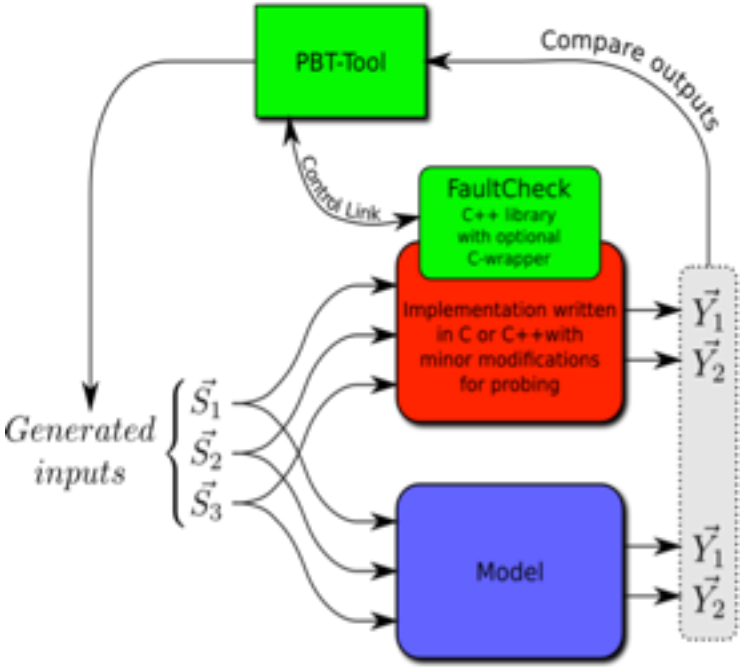

properties
QuickCheck
FaultCheck
tests
implementation
test results

Testing non-functional requirements

FaultCheck …

… a fault-injection tool for C code that combines fault-injection and property based testing using QuickCheck.

Contact: *SP*



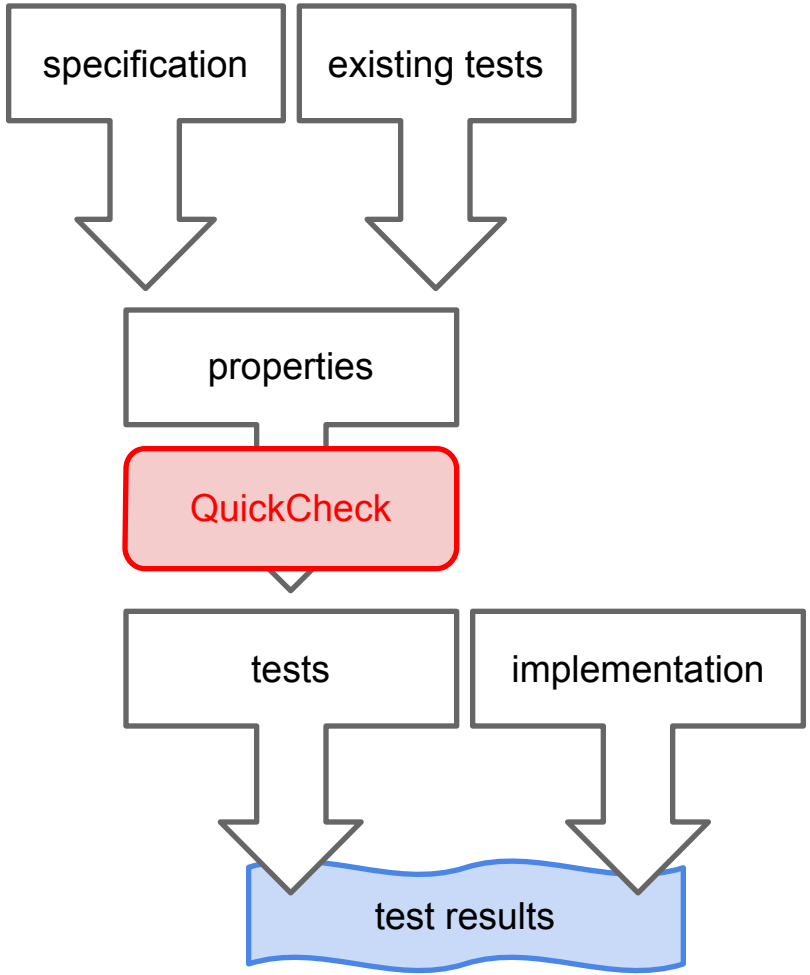
QuickCheck CI is a continuous integration server that runs QuviQ QuickCheck on a project.

Open Source Developers can use QuickCheck CI to get free access to QuviQ QuickCheck.

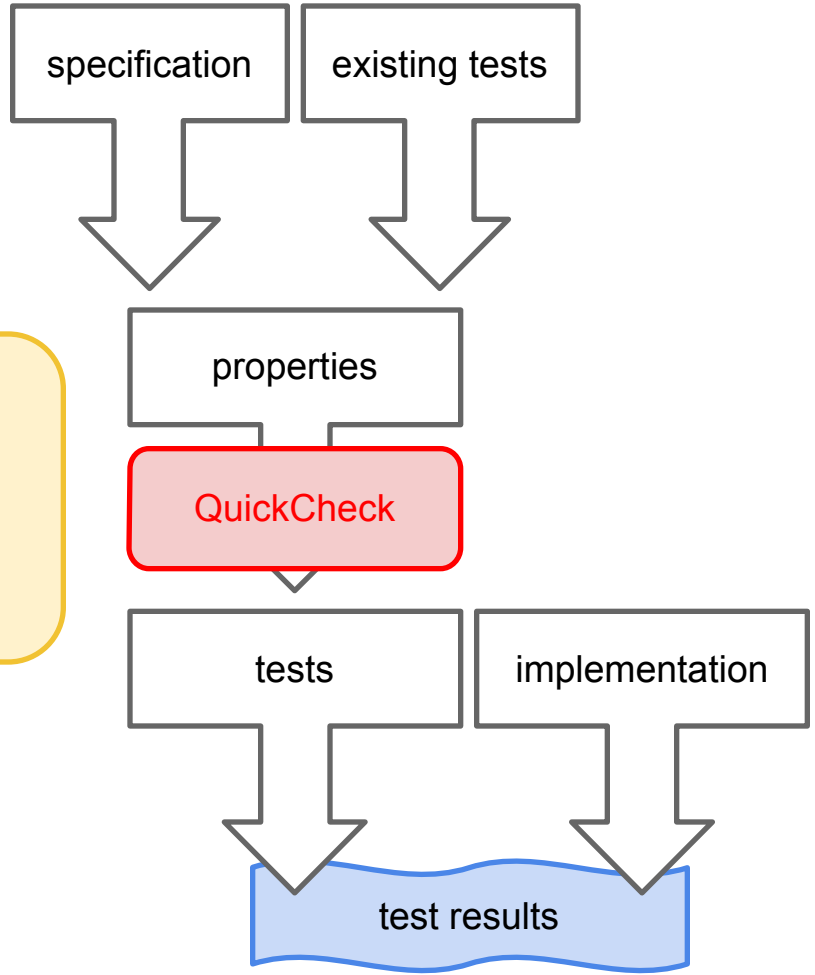
QuickCheck CI runs the full version of QuickCheck, including the connection to C.

Contact: *Quviq*

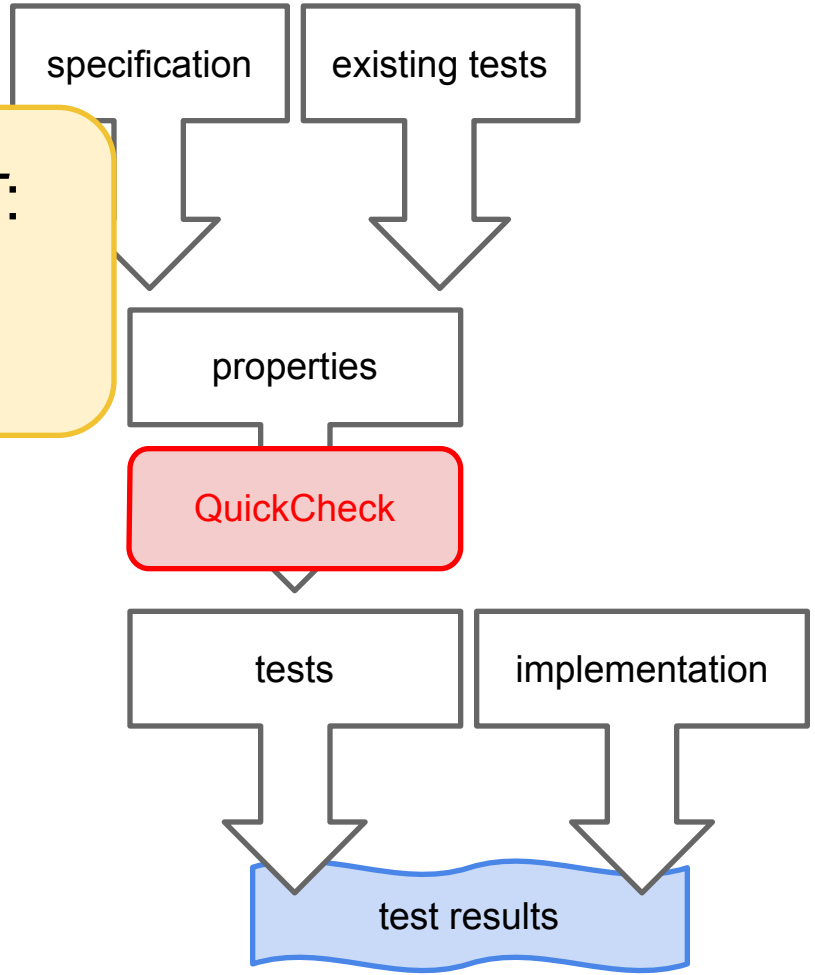
Results

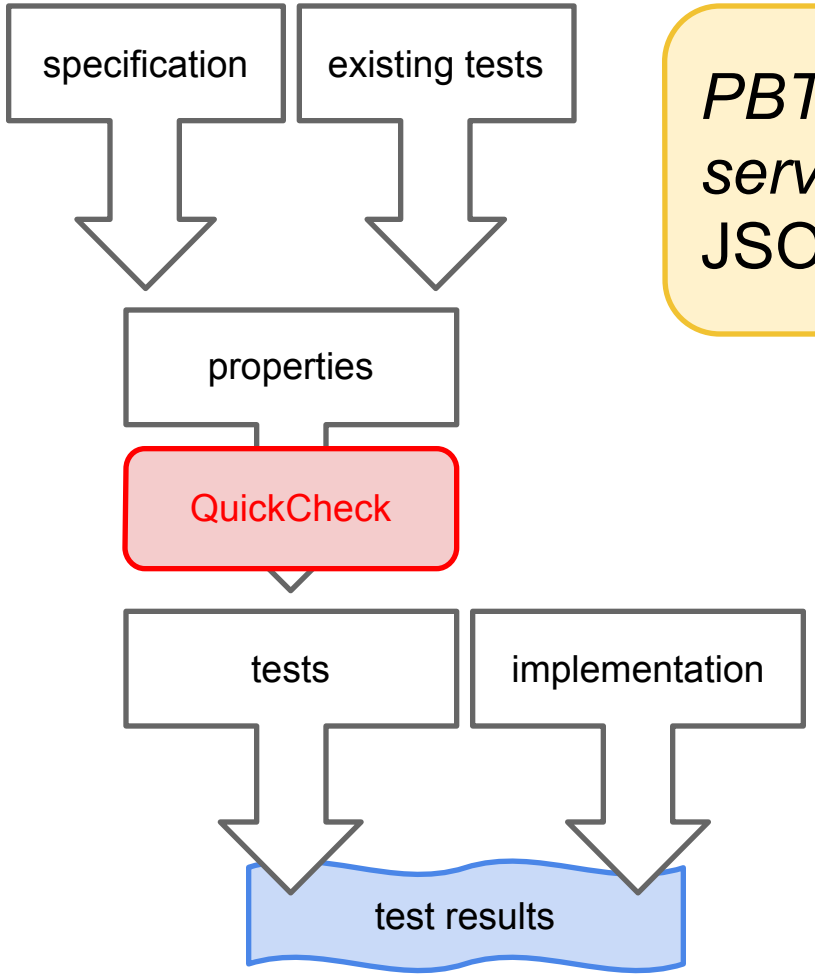

specification
existing tests
properties
QuickCheck
tests
implementation
test results

Scalable PBT: components and mocking

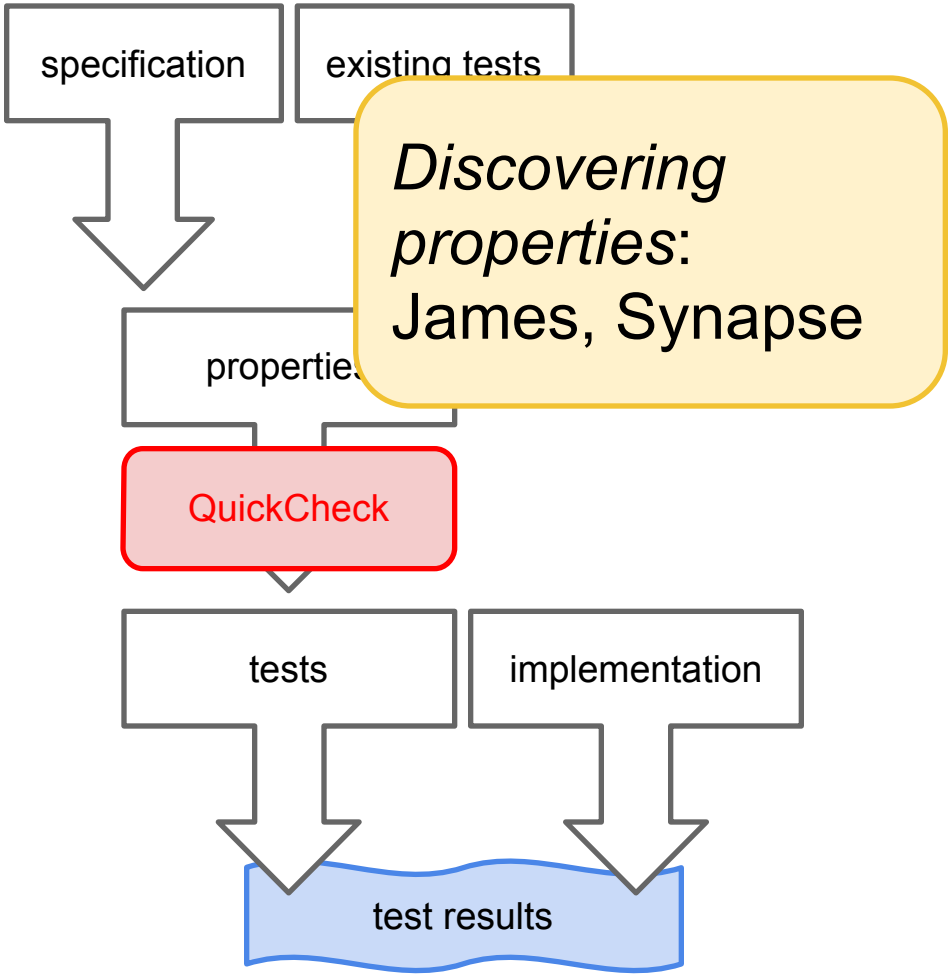


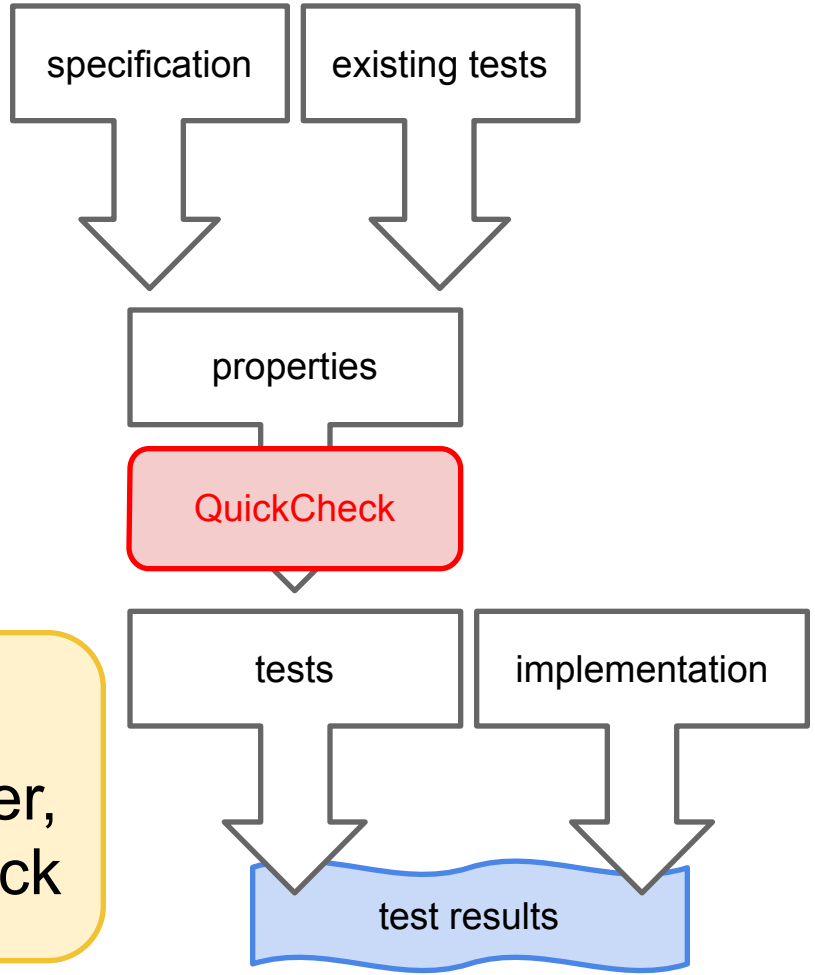
Accessible PBT:
ReadSpec,
GoodExamples



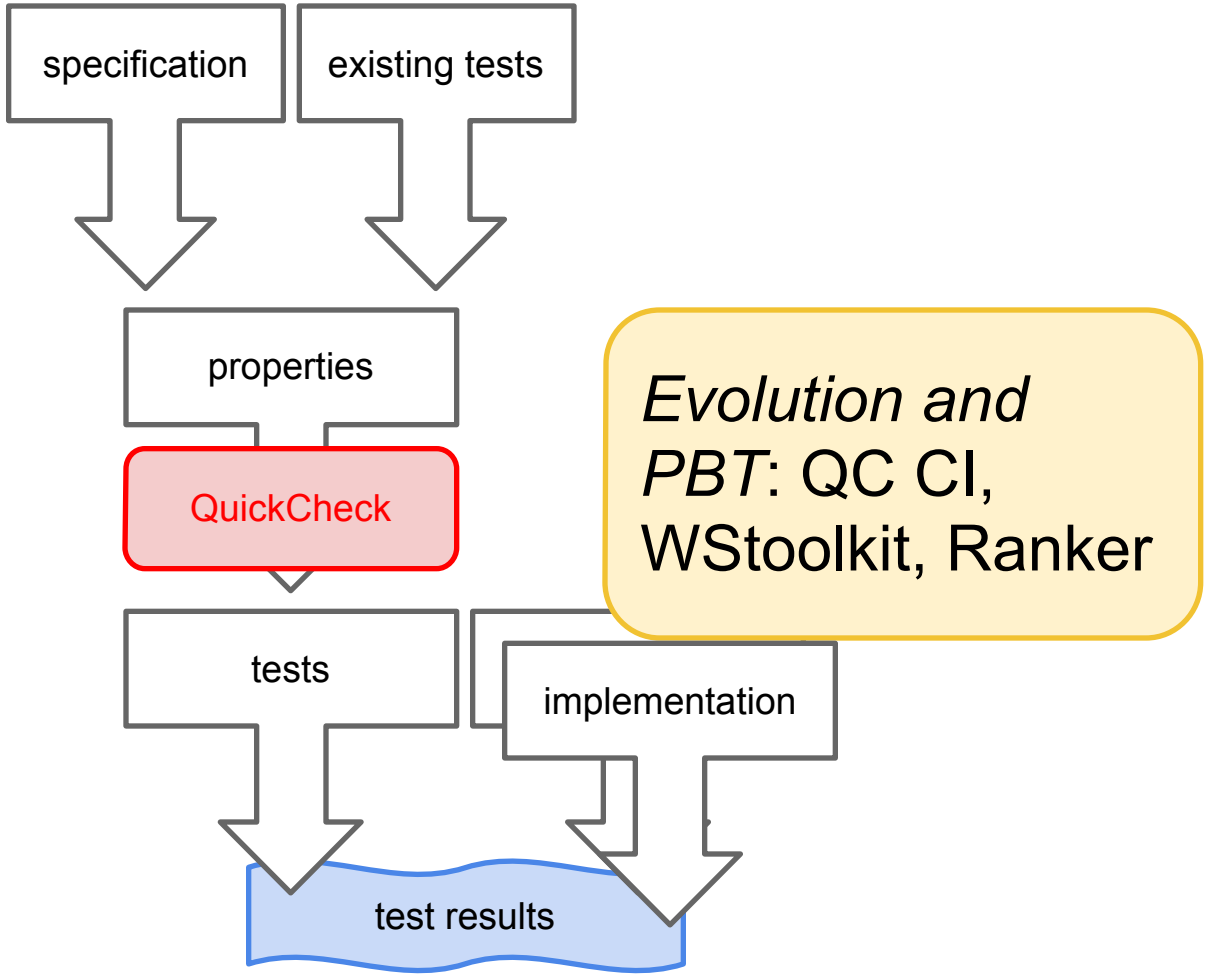


PBT for web services: WStoolkit, JSONgen


specification
existing tests
properties
QuickCheck
tests
implementation
test results
Discovering properties:
James, Synapse



Improved testing: Smother, Mu2, FaultCheck


specification
existing tests
properties
QuickCheck
tests
implementation
test results
Evolution and PBT: QC CI, WStoolkit, Ranker

Acknowledgement

The Universities of Sheffield, Kent, A Coruña, Chalmers Technical University and the Polytechnic University of Madrid; Quviq AB, Interoud, Erlang Solutions Ltd and SP gratefully acknowledge the support of the European Commission for the PROWESS project, funded under Framework Programme 7.

Results

Scalable PBT: components, mocking
Accessible PBT: ReadSpec, GoodExamples
PBT for web services: WStoolkit, JSONgen
Discovering properties: James, Synapse
Improved testing: Smother, Mu2, FaultCheck
Evolution and PBT: QC CI, WStoolkit, Ranker

www.prowess-project.eu