j-POST: a Java Toolchain for Property-Oriented Software Testing Model-Based Testing / ETAPS'08

<u>Yliès Falcone</u> Laurent Mounier Jean-Claude Fernandez Jean-Luc Richier

Vérimag & LIG, Universities of Grenoble

March 30 2008, Budapest, Hungary

Testing a policy?

Notion of policy :

- Expected behavior of nowadays systems
- Specific domains, e.g. security

Issues when testing a policy :

- Availability and faithfulness of the model
- Discrepancy between the policy and the system

Testing a policy?

Notion of policy :

- Expected behavior of nowadays systems
- Specific domains, e.g. security

Issues when testing a policy :

- Availability and faithfulness of the model
- Discrepancy between the policy and the system

Proposed Approach

- No functional specification of the system
- policy = set of requirements expressed by logical formula
- **property-oriented** test generation based on the structure of each formula φ

Running example : a travel agency application

The Travel application

- Management of mission of employees in an organization
- Client/Server principle
- Users request to perform operations on the application

Running example : a travel agency application

The Travel application

- Management of mission of employees in an organization
- Client/Server principle
- Users request to perform operations on the application

A security policy for Travel

"It is impossible to create a mission in Travel before being connected" Formalization :

- elementary operation
- use of a logical formalism

 $(\neg mission_creation(user, mission)) \mathcal{U}$ connection(user)

Outline



- Test generation
- Test execution

2 The j-POST toolchain and its functionalities

3 Conclusion

Test generation

Test Generation : general principle

Aimed to produce a test case relatively to a logical formula

About the test case

A set of communicating test processes (typed process algebra)

- test modules : evaluation of elementary predicates *e.g.* : to connect, perform an operation
- test controllers : associated to operators of the formal requirement
- \rightarrow Defined compositionaly following the syntax of the formula

Underlying theory

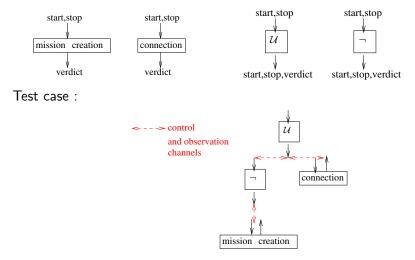
The j-POST toolchain and its functionalities 00000000

Conclusion

Test generation

$\varphi = (\neg mission_creation()) \mathcal{U} connection()$

Inputs : test modules and controllers :



Test generation

Compositional test generation

Test generation function

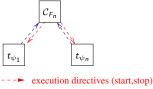
Produce $t_{\varphi} = GenTest(\varphi)$

• test module t_{p_i} associated to each predicate p_i

• each
$$\psi = F^n(\psi_1, \dots, \psi_n) \longrightarrow$$
 test process t_{ψ}
 $\hookrightarrow t_{\psi}$: parallel composition of $t_{\psi_1}, \dots, t_{\psi_n}$ and \mathcal{C}_{F^n} called a test controller for the operator F^n .

Roles of the controllers :

- execution management for sub-tests
- combination of verdicts



· -► verdict transmission

Principle of test execution

Produce a verdict for the initial requirement

 \hookrightarrow Parallel execution of the test case and the SUT with synchronisation on common action

Principle of test execution

Produce a verdict for the initial requirement

 \hookrightarrow Parallel execution of the test case and the SUT with synchronisation on common action

Issues for execution :

- Generated test case contains lots of possible executions
 - Non-determinism inside test modules
 - Parallel execution of test modules
- Information in the test case comes (only) from the formula (no additional behavior)

e.g. : insert a disconnection after being connected

Principle of test execution

Produce a verdict for the initial requirement

 \hookrightarrow Parallel execution of the test case and the SUT with synchronisation on common action

Issues for execution :

- Generated test case contains lots of possible executions
 - Non-determinism inside test modules
 - Parallel execution of test modules
- Information in the test case comes (only) from the formula (no additional behavior)

e.g. : insert a disconnection after being connected

 $\hookrightarrow \mathsf{Problem} \text{ of test selection}$

 \hookrightarrow A classical solution : use a test objective during test generation (e.g. TGV)

Principle of test execution

Produce a verdict for the initial requirement

 \hookrightarrow Parallel execution of the test case and the SUT with synchronisation on common action

Issues for execution :

- Generated test case contains lots of possible executions
 - Non-determinism inside test modules
 - Parallel execution of test modules
- Information in the test case comes (only) from the formula (no additional behavior)

e.g. : insert a disconnection after being connected

 $\hookrightarrow \mathsf{Problem} \text{ of test selection}$

 \hookrightarrow A classical solution : use a test objective during test generation (e.g. TGV)

Not applicable directly : no complete specification

Test Objectives (1) : definition

The test objective guides the test execution

- Features : local priorities and inserable behavior
- Reduce choices between actions

Test Objectives (1) : definition

The test objective guides the test execution

- Features : local priorities and inserable behavior
- Reduce choices between actions

Definition (Behavioural test objective)

A test objective O relatively to a test case t which semantics can be expressed by a LTS $(Q^{S_t}, Act^{S_t}, \rightarrow_{S_t}, q_0^{S_t})$ is :

- a deterministic LTS $(Q^O, Act^O, \rightarrow_O, q_0^O)$
- complete wrt. Act^{S_t} (*i.e.* $\forall q \in Q^{S_t}, \forall a \in Act^{S_t}, \exists q' \in Q^O \cdot q \xrightarrow{a}_O q')$
- Q^O contains two sink states Accept^O and Reject^O
- $Act^{O} \subseteq Act^{S_t} \cup Act^{com}$

Underlying theory

The j-POST toolchain and its functionalities ${\scriptstyle 0000000}$

Conclusion

Test execution

Test Objectives (2) : test selection

Priorities for actions :

- For $a \in Act_O$, prio(a)
- $\forall E \subseteq Act^*$.

 $prioritary(E) = \{a \in E \mid prio(a) = max\{prio(act) \mid act \in E\}\}$

Underlying theory

The j-POST toolchain and its functionalities 00000000

Conclusion

Test execution

Test Objectives (2) : test selection

Priorities for actions :

- For $a \in Act_O$, prio(a)
- $\forall E \subseteq Act^*$. prioritary(E) = { $a \in E \mid prio(a) = max{prio(act) \mid act \in E}$ }

Test selection via a test objective

For $a \in Act \cdot prioritary(O(o)) = \{a\}$,

$$\frac{a \in T(t) \qquad t \stackrel{a}{\rightarrow}_{S_t} t' \qquad o \stackrel{a}{\rightarrow}_{O} o'}{(t, o) \stackrel{a}{\rightarrow}_{S_t \times O} (t', o')} 1$$

$$\frac{a \notin T(t) \qquad \text{inserable}(a) \qquad o \stackrel{a}{\rightarrow}_{O} o'}{(t, o) \stackrel{a}{\rightarrow}_{S_t \times O} (t, o')} 2$$

$$\frac{o \stackrel{a}{\rightarrow}_{O} Reject_{O}}{(t, o) \stackrel{a}{\rightarrow}_{S_t \times O} Inc} 3$$

Outline

Underlying theory : Property-Oriented Software Testing

2 The j-POST toolchain and its functionalities

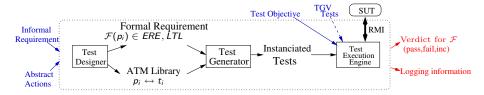
- Test designer
- Test generator
- Test execution engine

3 Conclusion

Overview

What is it?

A toolchain dedicated to Property-Oriented Software Testing.



Features

Several phases of testing are addressed :

- design (manual)
- generation (automatic)
- execution (automatic)

The j-POST toolchain and its functionalities

Features

Several phases of testing are addressed :

- design (manual)
- generation (automatic)
- execution (automatic)

Support for several logics :

- Extended Regular Expressions
- Linear Temporal Logic

Features

Several phases of testing are addressed :

- design (manual)
- generation (automatic)
- execution (automatic)

Support for several logics :

- Extended Regular Expressions
- Linear Temporal Logic

Using open formats for internal representations (tests modules, test controllers) :

- interaction with other tools (e.g. : TGV)
- logical formalism (plugin)

Outline

Underlying theory : Property-Oriented Software Testing

2 The j-POST toolchain and its functionalities

- Test designer
- Test generator
- Test execution engine

3 Conclusion

Test designer

Functionalities of the test designer

About abstract test modules

- extended LTS
- Actions : internal, communication (internal to the tester), with the SUT
- A typed process algebra as an underlying formalism

Establishment of a test module library

- Edition of test modules
 - XML GUI (based on Eclipse RCP)
- Visualization of test modules
 - Generation of a graph
 - GraphViz

The j-POST toolchain and its functionalities $0 \bullet 0 \circ 0 \circ 0 \circ 0$

Conclusion

Test designer

		j-POST Test Design	er	
Eile				
Connection 🛛		- 0		
() Transitions State exists: s_init Transition added:s1				
Transition with an External Action				
- Transition with an Assignment Action				
Start State:	s_init			
End State:	sl		\sim	
Guard:	true		(s_init)	
Expression type:	verdict		[true]:=identify(Falcone,azerty)[true]verdict:=fail	
Expression value:	fail		sl	
Var name:	verdict			
∨ar type:	verdict			
Add Transition				
Transition with a Delaying Action				
Configuration Transitions	III	•		
Yliès Falcone (Vé	rimag)	j-POST	03/30/08, Budapest, Hungary	16 / 20

Test generator

Outline

Underlying theory : Property-Oriented Software Testing

The j-POST toolchain and its functionalities

- Test designer
- Test generator
- Test execution engine

3 Conclusion

Test generator

Functionalities of the test generator

Implements the GenTest function

- Construction of a set of communicating tests
- Instantiation of test modules and generic controllers with information from the formula (e.g. value of parameters)

Supported formalisms :

- (Future) Linear Temporal Logic
- Extended Regular Expressions

Extendable architecture via logic plugins

The j-POST toolchain and its functionalities

Test generator

Bile I estace-DEMO 83 Generation of a test case Test Generation doe successfully. I specify a formula (not createMission()) until connection() The of controllers The of controllers Names of controllers Indicator controllers Inditeters Inditeters	X	j-POST Test Generator		
Cleaneration of a test case lest Generation done successfully. Click to load controllers: Add Controllers * 1. Specify a formula It is of controllers: Names of controllers (not createMission()) until connection() If is of controllers: until controllers: * 2. Specify an output directory containing the test case until controller.xml + (nomelylies/testcase-DEMO Browse. until controller.xml until controller.xml 3. Generate Test Case Click to load controllers: Add controllers: Add controllers: Produced files Browse. - always controller.xml until controller.xml Produced files Click to load Test Modules: Add Test Modules Add Test Modules Produced files Freduced files Is of Module Tests Add Test Modules Vertice Controllerinstanciation.xml createMission/1xml createMission/1xml createMission/1xml controllerinstanciation.xml createMission/1xml createMission/2xml createMission/2xml controllerinstanciation.xml controllerinstanciation.xml contection/1 contection/1 untilC1 Controllerinstanciation.xml contection/1 connection/1 contection/2	Eile			
I. Specify a formula (not createMission()) until connection()	testcase-DEM0 ⊠	- 8	Click to load controllers:	Add Controllers
(not createMission!) until connection() ippe of formula: and Controller.xml and (not createMission!) until connection() ippe of formula: ippe of cornula: ippe of cornula: • 2. Specify an output directory containing the test case intlim: intlim: intlim: intlim: /home/ylies/testcase-DEMO Browse. aways Controller.xml until intlim: 3. Generate Test Case Click to load Test Modules: Add Test Modules intlim: Produced files Click to load Test Modules: Add Test Modules idsconnect() Vertice files Click to load Test Modules: Add Test Modules idsconnect() Vertice files idsconnect() idsconnect() idsconnect() idsconnect() untilC/L Controllerinstanciation.xml createMission/1 and off/Profile() idsconnect() idsconnect() createMission/stanciation.xml createMission/2 and idsconnect() idsconnect() idsconnect() idsconnect() reateMission/2 and idston.xml createMission/2 and idston.xml createMission/2 idsconnect() idsconnect() reateMission/2 and idston.xml idsconnect() idsconnect() idsconnect() idsforofile 2.xml	(i) Generation of a test case Test Gener	ration done successfully . 🥌 📩	Files of controllers	Names of controllers
(not createMission()) until connection() • Produced files • produced files • Produced files • CreateMission() until connection() • CreateMission() until connection() • Produced files • CreateMission() until connection() • CreateMission() • Produced files • CreateMission() • CreateMission() • Connection() • CreateMission() • CreateMission() • Produced files • CreateMission() • CreateMission() • Until(Croorbiersmin • CreateMission() • CreateMission() • Intil • CreateMission() • CreateMission() • CreateMission() • Until(Croorbiersmin • CreateMission() • CreateMission() • CreateMission() • Until(Controllerinstanciation.xml • CreateMission() • CreateMission() • CreateMission() • CreateMission() • CreateMission() • CreateMission() • CreateMission()	▼ 1. Specify a formula		untilClController.xml	until
Click to load Test Modules: Add Test Modules Produced files Click to load Test Modules: Add Test Modules Click to load Tes	(act an stablic in ()) with some stime()	Time of formula:	andController.xml	and
A direct former of the set case A direct former of the set of the	(not createmission()) until connection()		plusController.xml	+
			starController.xml	*
			untilCrController.xml	until
Image: Produced files awaysController.xml awaysController.xml choice Y Produced files or choice Add Test Modules: Add Test Modules Instant Case Click to load Test Modules: Add Test Modules Add Test Modules Instant Case Click to load Test Modules: Add Test Modules Add Test Modules Instant Case Click to load Test Modules: Add Test Modules Add Test Modules Instant Case Click to load Test Modules: Add Test Modules Add Test Modules Instant Case Click to load Test Modules: Add Test Module Tests Market Modules Instant Case Click to load Test Modules: Add Test Module Tests Market Modules Instant Case Click to load Test Modules: Instant Module Tests Market Modules Instant Case Instant Module Tests Market Modules Instant Module Tests Market Modules Instant Case Instant Module Tests Market Modules Instant Module Tests Market Modules Instant Case Instant Module Tests Market Modules Instant Module Tests Instant Modules Instant Module Tests Instant CaseLauncher.xml Instant Mod	L		untilCmController.xml	until
Informelylies/testcase-DEMO Browse choiceController.xml not 3. Generate Test Case or choiceController.xml not • Produced files Click to load Test Module Tests Add Test Module Tests Freduced Files Click to load Test Module Tests Names of Module Tests untiCrit Controllerinstanciation.xml createMission.xml createMission.xml connectioninstanciation.xml createMission.xml createMission.xml createMissionInstanciation.xml connection() createMission() notLicontrollerinstanciation.xml connection() createMission() notLicontrollerinstanciation.xml connection() connection() notLicontrollerinstanciation.xml connection() connection() <td>▼ 2. Specify an output directory containing th</td> <td>e test case</td> <td>dotController.xml</td> <td></td>	▼ 2. Specify an output directory containing th	e test case	dotController.xml	
/home/yiles/testcase-DEMO Choice Controller,min Choice Controller,min Choice Controller,min or 3. Generate Test Case or Click to load Test Modules: Add Test Modules Produced files Click to load Test Modules: Add Test Modules Click to load Test Modules Produced files Click to load Test Modules: Add Test Modules Add Test Modules connectioninstanciation.xml createMission.xml createMission.xml createMission.xml createMissioninstanciation.xml createMission/1 modifyProfile/1 createMission2.3n connectioninstanciation.xml connection.xml createMission/2 connection.xml connectioninstanciation.xml connection.xml connection.xml connection.xml testCaseLauncher.xml agenceChosen.xml agenceChosen.xml agenceChosen.xml untiCl1 Controllerinstanciation.xml connection2.3ml connection2.1			alwaysController.xml	always
s. Generate Test Case or Controller.xml or y. Produced files Click to load Test Modules: Add Test Module Tests until(7) Controllerinstanciation.xml disconnect.xml disconnect.xml until(7) Controllerinstanciation.xml createMission.xml reateMission() until(7) Controllerinstanciation.xml waldMission.xml waldMission() connect.torillerinstanciation.xml createMission.xml createMission.xml channelList.xml createMission.xml createMission.Xml channelList.xml createMission.xml createMission.Xml testCaseLauncher.xml agenceChoosen.xml agenceChoosen.xml until(1) Controllerinstanciation.xml connection() connection() until(1) Controllerinstanciation.xml connection() connection()	/home/ylies/testcase-DEMO	wse	choiceController.xml	choice
3. Generate Test Case Add Test Modules: Add Test Module Tests reduced Files Files of Module Tests Names of Module Tests untif(T_C Controllerinstanciation.xml createMission.xml createMission.xml connectioninstanciation.xml createMission.xml createMission.xml connectioninstanciation.xml createMission.xml createMission.xml connectioninstanciation.xml createMission.xml createMission.xml connectioninstanciation.xml createMission.xml createMission.Xml notLControllerinstanciation.xml connection.xml connection.xml untit(CL Controllerinstanciation.xml connection.xml connection.xml itestSaseLauncher.xml createMission.Xml createMission.2xml itestCaseLauncher.xml agenceChosen.xml agenceChosen.xml itestL agenceChosen.xml agenceChosen.xml itestL connection.2,xml connection.2,xml			notController.xml	not
Produced files Cick to load Test Modules Add Test Modules Add Test Modules Add Test Modules Add Test Modules Cick to load Test			orController.xml	or
Produced files Cick to load Test Modules Add Test Modules Add Test Modules Add Test Modules Add Test Modules Cick to load Test		· · · · · · · · · · · · · · · · · · ·		
Produced files Names of Module Tests untilCr1 Controllerinstanciation.xml disconnect(.)ml untilCr1 Controllerinstanciation.xml createMission.xml connectioninstanciation.xml waldMission(.) createMissioninstanciation.xml createMission.xml createMissioninstanciation.xml createMission.xml createMissioninstanciation.xml createMission.xml createMissioninstanciation.xml createMission.xml createMissioninstanciation.xml createMission.xml createMissioninstanciation.xml createMission.Xml testCaseLauncher.xml createMission.Xml testCaseLauncher.xml createMission.Xml testCaseLauncher.xml agenceChoosen.xml untilC1 Controllerinstanciation.xml connection2.xml	3. Generate Test Case		Click to load Test Modules:	Add Test Modules
rreaceVisionInstanciation.xml creaceVisionInstanciation.xml creaceVisionInstanciation.xml untilCrDcontrollerinstanciation.xml validMission.xml validMission.xml creaceVisionInstanciation.xml modifyProfile() creaceVisionInstanciation.xml creaceVisionInstanciation.xml creaceVisionInstanciation.xml channelList.xml connectionInstanciation.xml connectionInstanciation.xml not1Controllerinstanciation.xml connectionInstanciation.xml connectionInstanciation.xml untilClCottrollerinstanciation.xml connectionInstanciation.xml connectionInstanciation.Xml not1Controllerinstanciation.xml connectionInstanciation.Xml connectionInstanciation.Xml testCaseLauncher.xml agenceChoosen.xml agenceChoosen.xml formula.tl connectionIC1.xml connectionIC1.Xml			Files of Module Tests	Names of Module Tests
untilmic toolinginstanciation.xml validMission.xml validMission.xml connection/instanciation.xml modifyProfile.xml modifyProfile.Xml createMission/instanciation.xml createMission2.xml createMission2.00 channellut.xml connection2.xml connection3.xml not1Controllerinstanciation.xml connection2.xml connection2.xml testCaseLauncher.xml connection2.xml modifyProfile2.01 until(III.Controllerinstanciation.xml connection2.xml agenceChoosen.xml formula.tl agenceChoosen.xml connection2.xml	Produced Files		disconnect.xml	disconnect()
unticulticitionization.xml modifyProfile.xml modifyProfile.xml connectionization.xml createMissionixaticiton.xml createMissionixaticiton.xml channellust.xml connectionixaticiton.xml connectionixml not1Controllerinstanciation.xml modifyProfile.xml modifyProfile.xml testCaseLauncher.xml modifyProfile.xml modifyProfile.xml untilCl1 Controllerinstanciation.xml agenechoosen.xml agenechoosen.xml formula.tl modifyProfile.xml connection2.xml	untilCr1 ControllerInstanciation.xml		createMission.xml	createMission()
createWissionZxml createWissionZxml createWissionZxml createWissionZxml channelList.xml connection.xml connection.xml notIControllerinstanciation.xml agenceChoosen.xml agenceChoosen.xml testSaeLauncher.xml connection.xml connection2xml testSaeLauncher.xml agenceChoosen.xml agenceChoosen.xml formula.lt connection2.xml connection2xml	untilCm1ControllerInstanciation.xml		validMission.xml	validMission()
create/sistom/stanciation.xml connection.xml connection.xml channellit.xml connection.xml modifyProfile2.xml not1 Controllerinstanciation.xml agenceChoosen.xml agenceChoosen.xml untiCliC controllerinstanciation.xml connection2.xml connection2.xml formula.tl connection2.xml connection2.xml	connectionInstanciation.xml		modifyProfile.xml	modifyProfile()
Institution.xml modifyProfile2.xml modifyProfile2.1 Institution.xml agenechoosen.xml agenechoosen.xml until12.controllerinstanciation.xml connection2.xml connection2.1	createMissionInstanciation.xml		createMission2.xml	createMission2()
testcaselauchonamia testcaselauchonamia untilli controllerinstanciation.xml formula.lti	channelList.xml		connection.xml	connection()
turticiti Controllernstanciation.xml connection2.xml connection2() formula.lt	not1ControllerInstanciation.xml		modifyProfile2.xml	modifyProfile2()
formula.ltl	testCaseLauncher.xml		agenceChoosen.xml	agenceChoosen()
	untilCl1ControllerInstanciation.xml		connection2.xml	connection2()
	formula.ltl			
actionList.xmi	actionList.xml			
Generation of a test case	Generation of a test case			

Outline

Underlying theory : Property-Oriented Software Testing

The j-POST toolchain and its functionalities

- Test designer
- Test generator
- Test execution engine

3 Conclusion

The j-POST toolchain and its functionalities $\circ\circ\circ\circ\circ\circ\circ\circ$

Test execution engine

Functionalities of the test execution engine

Goal : produce a verdict for the initial requirement

Functionalities of the test execution engine

Goal : produce a verdict for the initial requirement

Multithreaded execution

- Java Thread / test process
- Scheduling policy (test objective)
- Priority order between actions

Functionalities of the test execution engine

Goal : produce a verdict for the initial requirement

Multithreaded execution

- Java Thread / test process
- Scheduling policy (test objective)
- Priority order between actions

"on the fly" concretisation

Mapping between external abstract actions and the actions on the SUT

- Remote Method Invocation
- Web service invocation

• . . .

X	j-POST Test Execution Engine		×
Ele			
Execution 🕱			- 0
Execution (3	Browse Test Case	Execution Informations Enable/Disable	
	38		ž.
Yliès Falcone (Vérimag)	j-POST	03/30/08, Budapest, Hungary	22 / 26

Outline



2 The j-POST toolchain and its functionalities

3 Conclusion

Conclusion

A Java toolchain dedicated to property testing

Extendable architecture

Proposed approach

- algebraic composition
- property-driven
- suitable for security policy testing

References

j-POST Web page : http ://www-verimag.imag.fr/~async/jpost.html

Theory

- "A Test Calculus Framework Applied to Network Security Policies". In FATES/RV'06 : Formal Approaches to TESting/Runtime Verification
- "A Compositional Testing Framework Driven by Partial Specifications". In TESTCOM/FATES'07 : TESTing of COMmunicating Systems/Formal Approaches to TESting
- "A Partial-specification Driven Compositional Testing Method". Vérimag Technical Report [TR-2007-4]

Implementation

- j-POST : a Java Toolchain for Property-Oriented Software Testing. Vérimag Technical Report [TR-2007-7]
- Experiments with j-POST on a Travel Agency Application

Perspectives

Assessment of quality for produced tests

- coverage criteria
- mutation

Perspectives

Assessment of quality for produced tests

- coverage criteria
- mutation

Using Abstraction/Concretisation in test modules

- Abstract types [Lestiennes,Gaudel'02]
 - Domain : finite set of values
 - Operations on abstract data types need corresponding concrete operations
- Issues :
 - Test execution can modify concrete domains
 - Coverage of concrete wrt. abstract domains