

Testing object Interactions Grüner, A.

Citation

Grüner, A. (2010, December 15). *Testing object Interactions*. Retrieved from https://hdl.handle.net/1887/16243

Version:	Corrected Publisher's Version
License:	<u>Licence agreement concerning inclusion of doctoral</u> <u>thesis in the Institutional Repository of the University</u> <u>of Leiden</u>
Downloaded from:	https://hdl.handle.net/1887/16243

Note: To cite this publication please use the final published version (if applicable).

INDEX

accessibility, 59 activation record CoJapl, 141 Japl, 45 specification language, 73 sequential programming language, 31 active statement, see control context, active anticipation, 87, 92, 159, 194 anticipation-based code structure, 205 anticipation-valid code. 194 configuration, 197 anticipation-validity, 197 auxiliary notations, 34 balance, 43 behavior-based testing, see interactionbased testing binary tree, 26 bisimulation, 106 testing, 109, 208 weak, 107, 189 $C^{\sharp}, 4$ $C^{++}, 3$ call stack CoJapl, 141 specification language, 157 Japl, 46 specification language, 73 sequential programming language, 31 code generation, 99

code generation, 83, 159, 187 Japl, 95, 102 correctness, 103 code-in, 101code-out, 100 correctness, 203, 217 CoJapl, 139 communication label, see transition label completeness, 58 component, 38 compositionality, 54, 177 concurrency, 139 configuration CoJapl, 143 Japl, 45 specification language, 73 initial, 36, 76, 149 active, 50 passive, 50 sequential programming language, 31 consistent control flow, 43 consistent information flow, 43 control context active, 67 passive, 67 control flow, 84 decomposition, 184 defect, 6 environment, 38 error, 6 executability, 57, 79

expectation body, 63 expectation statement. 59 expression evaluation, 34 external semantics Japl, 41 fail. see failure failure. 6 failure report, 111 faulty specification, 111 free variables, 46 incoming communication, 41 inheritance, 126 initial expectation identifier, adjustment, 92input enabledness, 79 integration testing, 5 interaction-based testing, 11, 57 interactions, 10 interface communication, 41 internal semantics, 41

Java, 3 Japl, 38 JMLUnit, 14 jMock, 10 JUnit, 7

label check, 45 labeled transition system, 41 labeling mechanism, 85

merge components, 54 configurations, 179 mock class specification, 65 mock objects, 11, 65 mock thread specification, 153 mocks, *see* mock objects mutual exclusion, 161

next, 87 nextlist, 160

object-oriented programming, 2

operational semantics, 36 CoJapl, 145-147, 150 specification language, 159 Japl, 48 specification language, 73 sequential programming language, 31 specification classes, 119 subtyping and inheritance, 131, 133 outgoing communication, 41 passive statement, see control context, passive preprocessing, 85, 89, 90, 95, 187 program execution CoJapl, 149 Japl, 50 sequential programming language, 36programming classes, 122 propagation of new names, 49 realizability

Japl. 42 renaming, 49 satisfiability, 58 Simula, 2 simulation, 104 testing, 108 weak, 105 Smalltalk, 3 software crisis, 1 software development process, see software development life-cycle spawn, 139, 152 specification language, see test specification language specification class, 115 specification execution, 75 state-based testing, 52 static semantics, 29, 141 CoJapl, 141 specification language, 157 Japl, 40

specification language, 68 programming classes, 124 sequential programming language, 26specification classes, 118 subtyping and inheritance, 128 subclassing, 126 subject reduction, 50, 173 subtyping, 126 syntax CoJapl, 140 specification language, 155 Japl, 39 specification language, 64 programming classes, 122 sequential programming language, 24specification classes, 116 subtyping and inheritance, 127 test specification language, 57, 115, 122, 151test thread specification, 152, 153 thread class, 139 thread configuration mapping, 143 thread-save, 156 trace Japl, 50 specification language, 75 trace semantics CoJapl, 149 Japl, 51 specification language, 76 traces. 50 transition label CoJapl, 146 Japl, 42 type system, *see* static semantics types sequential programming language, 29unit testing, 5, 7 variable binding, 84, 93

variable evaluation, 33 variable globalization, 94 V-model, 5

well-typedness, dynamic specification code, 207