

All severity 2 bugs in *Testing Object-Oriented Systems: Models, Patterns and Tools* (as of December 20, 2002) are listed in this document.

The bug reports on this list include items found by the author and those submitted by Rana Ahmed, Antti Auer, Bill Bently, Prof. Lionel Briand, Dr. Arthur Carlson, Jeff Dever, Dion Dock, Ron Lusk, Patrick T. McGuigan, Norbert Mueller, Thomas Mueller, Prof. Arthur Reyes, Patrick Schoenbach, Peter Seibel, Serge Tanguay, Simon Watts, Michael Wells, and Russ Williams. Prof. Zdzislaw Ploski sent in an extensive list produced by painstaking effort. Thanks to all for your contributions.

The “sev” (severity) column designates the kind of bug:

- 1: Incorrect statement, concept, code syntax error, or other blunder.
- 2: Misspelling, misconstruction, or inconsistent usage.
- 3: Minor typo not likely to cause confusion or misunderstanding.

The “rev” (revision) column indicates the press run in which this bug is corrected. If blank, the bug correction hasn’t been published yet. The press run of your copy is on the bottom page iv (not numbered by convention.) If your copy is the 3rd or higher press run, items with rev=3 are corrected in your copy. Corrections for all other items will be published as soon as possible.

Pg.	Sev	Version 1.0	Correction	Rev
xlviii	2	Regan	Reagan	
6	2	is_isocelles	is_isosceles	
19	2	const true = 1;	const bool true = 1;	
23	2	[D'Souza+98]	[D'Souza+99]	
25	2	Parameterised Class Test	Generic Class Test	
28	2	how to implement a built-in test for the server	how to implement built-in test for a server	3
48	2	See Section 9.4	See Section 9.3	
49	2	A surprise is code that does not support a required capability.	A surprise is code that does not support any required capability.	3
57	2	because quartersSinceLastTx is ...	because quartersSinceLastTx() is ...	
57	2	lastTxDay.txDate	lastTxDate.txDay	

Pg.	S e v	Version 1.0	Correction	R e v
76	2	[Overbeck 94]	[Overbeck 94b]	
79	2	The actual behavior of polymorphic messages is determined by many variables not visible in the source code.	The actual behavior of polymorphic messages is determined by many variables not visible in the client's source code.	3
88	2	[Firesmith 92]	[Firesmith 93b]	
97	2	McGregor 92	delete	3
100	2	[Firesmith 92]	[Firesmith 93b]	
102	2	[McCabe+94b]	[McCabe+94]	
116	2	[Rumbaigh+91]	[Rumbaigh+91]	3
123	2	Derive the logic function for the model to validate its completeness and consistency.	Verify the model's completeness and consistency.	3
125	2	In table 6.1, identical values for Number of Claims are suppressed.	Add explicit row leader: Number of Claims in Variant 2 = 0, variant 4 = 1, variant 6 = 2 or 4.	
127	2	In figure 6.2, the six age conditions are expressed with “and”, e.g., “25 and younger.” This is inconsistent with the Boolean sense of “and.”	Change all six age conditions to “or”, e.g., “25 or younger”.	
134	2	Figure 6.4, key not given	Key: dash indicates undefined term or symbol	3
141	2	Deitmeyer	Dietmeyer	3
143	2	Steps 2 and 3	Steps 3 and 4	
143	2	... the truth values for each cell's row and column.	... its truth values for each cell of the group.	
143	2	Transcribe the product terms for this group.	Transcribe the product term for this group.	
144	2	adjecent	adjacent	
145	2	In figure 6.8, the two age conditions are expressed with “and”, e.g., “25 and younger.” This is inconsistent with the Boolean sense of “and.”	Change both age conditions to “or”, e.g., “25 or younger”.	

Pg.	S e v	Version 1.0	Correction	R e v
154	2	If the implicit variant result from ...	If the implicit variant results from ...	
157	2	[Bryant 94]	[Bryant 92]	
158	2	The D branch above them ...	The D branches above them ...	
173	2	[Devedas+94]	[Devadas+94]	
183	2	the control, software	the control software	
186	2	Figure 7.6: <code>this.p2AddPoint()</code>	<code>this.p2_AddPoint()</code>	
194	2	[Holtzman 92]	[Holtzman 91]	
196	2	The event <i>FlashingRedOn</i> fires the transition <i>Red-FlashingRedOn</i> , <i>Yellow-FlashingRedOn</i> , or <i>Green-FlashingRedOn</i> ...	The event <i>FlashingRedOn</i> fires the transition <i>Red-FlashingRed</i> , <i>Yellow-FlashingRed</i> , or <i>Green-FlashingRed</i> ...	
200	2	Figure 7.15, motivation for elided states not clear.	The elided states represent safety requirements: (1) the cruise control action must be immediately suspended when the brake is applied, and (2) automatic brake modulation can only occur after sufficient pedal pressure to lock a wheel(s) has been applied. An IUT that can enter a Braking-Active or Modulating-Active state is dangerously defective; the engine would be generating increased power while the driver is trying to stop. Depending on the IUT's control laws, this could lead to an oscillation and complete loss of control.	
200	2	Guards missing right bracket	Added right bracket	3
201	2	[Firesmith 94]	[Firesmith 93a]	
202	2	[Beck 94]	[Beck+94b]	
203	2	[Sanet 95]	[Sane+95]	
205	2	In footnote: [Rational 97b]	[Rumbaugh+99]	
207	2	<code>Class Account</code>	<code>class Account</code>	

Pg.	S e v	Version 1.0	Correction	R e v
213	2	varibales	variables	3
222	2	[Brand 82]	[Brand+83]	
225	2	Figure 7.25, first guard on transition 2: [x!==0]	[x!=0]	
236	2	The subclass fails to retarget a superclass transition and instead switches to an incorrect or unrefined ...	The subclass fails to retarget a superclass transition and instead switches to an incorrect or undefined ...	
240	2	Table 7.7, item 4.10 duplicates item 4.9.	Delete item 4.10	
240	2	Table 7.7, item 3.1: on this state (for example	on this state. For example	
245	2	figure 7.36, transition separator missing at right margin.	Add separator: [this.p3_score()<20]/ and [this.p3_score()==20]/	3
248	2	pop [n < = 0]	pop [n <= 0]	
248	2	Figure 7.37: Response matrix (fig 7.37) is inconsistent with the statechart (fig 7.36).	(Add footnote, page 248.) Although the statechart (fig 7.36) shows the accessor methods p1_iswinner, ps_iswinner, and p3_iswinner are accepted only when the corresponding player has won, the Response Matrix (fig 7.37) represents them as being accepted in all states. These methods should return FALSE in the non-won states, so this is not a sneak path bug. A literal interpretation of the statechart would have required the opposite, i.e., treating these messages as sneak paths in all non-won states.	
259	2	Footnote 32: The <i>n</i> -switch is discussed in [Chow 78]. The set of round-trip paths is referred to as the <i>P</i> -set here.	The <i>n</i> -switch is discussed in [Chow 78]. The set of round-trip paths is referred to as the <i>P</i> -set here. Let <i>R</i> designate the round trip path set generated by the N+ algorithm, which is a minimal set of subpaths for all round trip paths. Path set <i>R</i> will have at most <i>m</i> switches, where <i>m</i> is number of transitions on the longest round trip path. In contrast, a tree which enumerates all switches to some level <i>k</i> must contain all paths in <i>R</i> . A switch-tree for a large enough <i>k</i> would subsume <i>R</i> .	
267	2	[Overbeck 92]	[Overbeck 94a]	

Pg.	S e v	Version 1.0	Correction	R e v
268	2	[Mealy 56]	[Mealy 55]	
268	2	[Moore 55]	[Moore 56]	
268	2	[Fujiwara 86]	[Fujiwara 85]	
275	2	[Warmer+99]	[Warmer+98]	
275	2	the elements of a graph that are connected by edges	an element of a graph connected by edges	
275	2	a line between a diagram element	an edge between diagram elements	
281	2	... to construct an operation relation	... to construct an operational relation	
290	2	Figure 8.6: getFreeAmount	getFeeAmount	
290	2	Figure 8.6: Eject	eject	
291	2	Table 8.6, “Represents” entry for Procedure Call Loop: The condition that selects ... in brackets	The scope of a loop. The loop termination condition may be noted next to the rectangle.	
293	2	An Activity Diagram represents sequences in which <i>activities</i> may occur.	An Activity Diagram represents sequences in which an <i>activity</i> may occur.	3
300	2	It represents only one slice of the IUT and can only support coverage analysis of this slice (see Chapter 9 for details of coverage analysis.)	It represents only one slice of the IUT and can support test design of only this slice.	3
309	2	... a pair of edges going in opposite directions, a bidirectional arrow, or an undirected line a pair of arrows going in opposite directions, a bidirectional arrow, or an undirected edge ...	
313	2	[Rumbaugh+99]	[Rumbaugh+98]	
320	2	[McConnell 94]	[McConnell+96]	
331	2	[Firesmith 95]	[Firesmith 96]	
348	2	These patterns show how method and scope level test patterns can be applied	These patterns show how method and class scope test patterns can be applied	3

Pg.	S e v	Version 1.0	Correction	R e v
348	2	Five test patterns to design method scope, responsibility-based tests	Four test patterns to design method scope, responsibility-based tests	3
356	2	[Graham+94, Overbeck 94, Firesmith 95, Firesmith 96]	[Graham+93,Overbeck 94a, Firesmith 95a, Firesmith 95b]	
357	2	the alpha-omega test suite	an alpha-omega test suite	
366	2	I model multiple exit points with a node for each return, as the return may have a right-side expression to evaluate and add a common exit node for the method.	I model multiple exit points with a node for each return expression (including a <code>return;</code>) and add common exit node for the method.	
370	2	<pre>if (a=b) or else -- short ct evaluation (x=y and is_empty()); if (a=b) or -- full evaluation (x=y and is_empty());</pre>	<pre>if (a=b) or else -- short ct evaluation (x=y and is_empty()); or if (a=b) or -- full evaluation (x=y and is_empty());</pre>	
371	2	Because all the statements in a loop body can be reached with a single iteration, the loop is covered in one pass. It is unlikely that a single pass will reveal common loop control bugs.	Even when all statements in a loop body can be reached with a single iteration, it is unlikely that a single pass will reveal common loop control bugs.	3
390	2	Procedure 5: Look for a definition-clear subpath	Look for a subpath without any D or K actions.	
402	2	[Firesmith 92 ... Graham+94 ...]	[Firesmith 93b ... Graham 94 ...]	
407	2	Table 10.5, row 5: $y < = 14.0 - x$	$y < = 14.0 - x$	
408	2	This situation is modeled by the abstract state invariant function <code>!astack.isFull()</code> .	This is modeled with an expression using the abstract state invariant function <code>!aStack.isFull()</code> .	
408	2	an incorrect action is triggered by an incorrect evaluation of abstract state: For example the IUT could fail to observe this condition if a guard check was missing or branched the wrong way.	an incorrect action is triggered by an incorrect evaluation of abstract state. For example, the invariant is missing or the IUT branches the wrong way when an inconsistent state is detected.	
409	2	any state must have at least one other resultant state, there are always two or more ways to define on and off points for abstract states.	any state must have at least one other resultant state, so there are at least two or more on and off points for abstract states.	

Pg.	S e v	Version 1.0	Correction	R e v
409	2	An abstract state in-point is any value combination that places an object of the class in a given abstract state and is not an abstract state on point.	An abstract state in-point is any value combination that places an object of the class in a given abstract state and is neither an on or off point of that abstract state.	
424	2	Table 10.10, item and element both used to refer a member of a list.	Change all occurrences of “item” to “element”	
429	2	The conditions in the <code>triangle</code> classification ...	The conditions in the <code>Triangle</code> classification ...	
433	2	[Lewis+82]	[Lewis+81]	
434	2	= =	==	
442	2	Figure 10.22, missing semi-colon in Account member functions: <code>listTransactions()</code>	<code>listTransactions();</code>	
444	2	[Hoffman 93]	[Hoffman+95]	
450	2	If we change the balance of an account object	If we change the balance of an Account object	
451	2	<code>creditLimit ≥ (txCounter × 100) + 100</code>	<i>creditLimit</i> ≥ (<i>txCounter</i> × 100) + 100	
453	2	Figure 10.23: <code>!isclosed</code>	<code>!isclosed()</code>	
453	2	shading in row for <code>acount2,!closed</code> , on, test cases 11, 12 is reversed.	Switch shading.	3
455	2	class/scope	class scope	
465	2	A system that automatically generates Ada operation pairs is described in [Parrish 93a] and a similar approach for C++ is outlined in [Parrish 93b]. Some theoretical considerations for the application of data-flow models to class interfaces are developed in [Parrish 94].	A system that automatically generates Ada operation pairs is described in [Parrish+93a] and a similar approach for C++ is outlined in [Parrish+94]. Theoretical considerations for the application of data-flow models to class interfaces are developed in [Parrish+93b] and [Parrish+95].	
467	2	[Hoffman 92]	[Hoffman+95]	
467	2	duplicate ... duplicate ... duplicate	Duplicate ... Duplicate ... Duplicate	

Pg.	S e v	Version 1.0	Correction	R e v
481	2	A class cluster that implements the State pattern [Gamma 95] is probably modal.	See section 7.2.9, State Machines and Object-oriented Development, for more examples.	3
486/ 487	2	Constants for Money objects given as 0	Should be 0.00	3
488	2	Table 10.30, all message names are capitalized.	All message names are lower case.	
489	2	It does not provide an external operation to check the time elapsed since the last transaction, although the class encapsulates some mechanism to monitor time.	It does not provide an external operation to check the time elapsed since the last transaction, so the class must implement some mechanism to monitor time.	3
491	2	Figure 10.35, all message names (events) are capitalized.	All message names are lower case.	
492	2	Constants for Money objects given as 0	Should be 0.00	3
492	2	Table 10.31, all message names are capitalized.	All message names are lower case.	
494	2	method/scope	method scope	
496	2	Chow's algorithm is mentioned	Chow's algorithm [Chow 78] is mentioned	
498	2	a stub implementation of these features.	a stub implementation of these components.	
503	2	[Stroustrup 92a]	[Stroustrup 92a]	
508	2	Figure 10.40: <code>Account.rollover</code>	<code>Account.rollover</code>	
516	2	[Firesmith 95]	[Firesmith 95b]	
516	2	[McGregor 94c]	[McGregor 94b]	
519	2	See section 7.4.6	See section 7.4.5	
521	2	[McGregor 94c]	[McGregor 94b]	
521	2	[Firesmith 95]	[Firesmith 95b]	
522	2	[Zhu 96]	[Zhu+97]	

Pg.	S e v	Version 1.0	Correction	R e v
522	2	Critiques of this concept appear in [Hamlet 90] and [Hamlet 97].	A critique of this concept appears in [Hamlet 90].	
532	2	The differences in strategy for producers and consumers is further discussed in the companion book ... [Binder 2000].	Delete.	
549	2	[Codine+97]	[Codenie+97]	
552	2	use <i>Risk-Based Allocation</i> [Binder 2000] , to plan the testing effort.	consider risks when planning the testing effort [Binder 97b].	
557	2	[Codine+97]	[Codenie+97]	
562	2	[Wolfe 97]	[Woolf 97]	
562	2	Wiede	Weide	
574	2	Fig 12.3: Variable/Condition/Type	Association/Condition/Type	3
575	2	Fig 12.4: Variable/Condition/Type	Association/Condition/Type	3
585	2	In Path 1, box ~aSession is not shaded	In Path 1, box ~aSession should be shaded	3
588	2	In Path 6, box ~aSession is not shaded	In Path 6, box ~aSession should be shaded	3
591	2	[Firesmith 92]	[Firesmith 93b]	
594	2	<i>Run</i> the system under test	<i>Mutate</i> . Run the system under test	3
598	2	[Firesmith 95]	[Firesmith 95b]	
599	2	subsystem/scope	subsystem scope	
599	2	[Jacobsen+92]	[Jacobson+92]	
605	2	Figure 12.13 inconsistent names: dfControl:Controller DeviceException dfItem.Select()	dfController:Controller DeviceException() dfItem.select()	

Pg.	S e v	Version 1.0	Correction	R e v
606 610	2	Figures 12.14 through 12.19: incorrect italic font for function names, missing () after function names, external events not in upper case, up arrow missing on some function call return actions.	Same font on all <code>function</code> names, add parens to function names, make external events consistent, add action up arrow where missing.	
611	2	Figure 12.20, title: controller class.	<code>controller class</code> .	
624	2	[Bosivert 97]	[Boisvert 97]	
632	2	Protocol Graph	deleted	3
648	2	Navigate all menu paths	Activate each user interface widget at least once.	3
667	2	Coverage analyzers	Some coverage analyzers	3
676	2	[Firesmith 92]	[Firesmith 93b]	
692	2	Fig 13.27 Boxes shaded	boxes should be unshaded.	3
714	2	[Lieberherr+92]	[Lieberherr+92]	
714	2	Firesmith's Test Stubs and Dependency-Based Testing patterns suggest how the subjects could be used to develop an integration test plan [Firesmith 96].	Firesmith's <i>Test Stubs</i> and <i>Dependency-Based Testing</i> patterns address integration test issues [Firesmith 96].	
717	2	has been not been	has not been	
726	2	Last sentence beginning on page 726 is a duplicate.	delete	
735	2	[Hien 85]	[Hein 85]	
739	2	A <i>Mode Machine</i> model, which models use cases	Use <i>Mode Machine</i> to model use cases	
758	2	[Jones 97] ... [Jones 97]	[Jones 97a] ... [Jones 97a]	
768	2	[Jones 97]	[Jones 97a]	
776	2	[Jones 97]	[Jones 97a]	
777	2	A delta component suffers a failure	A delta component induces a failure	3

Pg.	S e v	Version 1.0	Correction	R e v
790	2	Table 15.7, column header: As A Bs Bs	A's A's B's B's	
796	2	In the worst case, the section	In the worst case, the selection	3
796	2	[Kung+95]	[Kung+95a]	
797	2	[Firesmith 95]	[Firesmith 95a]	
800	2	test case generations	test case generators	3
802	2	after two or three projects	after two or three development cycles	3
811	2	Figure 17.1: Type Inversion	Not LSP compliant	
811	2	Figure 17.1: (1) See discussion of type inversion in chapter 15.	(1) See <i>Polymorphic Server Test</i> .	
814	2	[Rumbaugh+99]	[Rumbaugh+98]	
819	2	Such code segements	Such code segments	
823	2	[Warmer+99]	[Warmer+98]	
823	2	[Rumbaugh+99]	[Rumbaugh+98]	
827	2	assert (min <= max)	assert(min <= max);	
830	2	assert (assert(
835	2	at least one acceptance condition	at least one accepting condition	
838	2	methodName	methodName	
838	2	App_Class::legalResult is a Boolean function that returns <i>true</i> when the current state, message, and entry state correspond to a legal transition. If the resultant state is allowed for the event, it returns <i>false</i> .	App_Class::legalResult is a Boolean function that returns <i>true</i> when the current (resultant) state, message, and entry state correspond to a legal transition. If the resultant state is not allowed for the entry state/event, it returns <i>false</i> .	3
845	2	[Parrish 93b]	[Parrish+93a]	

Pg.	S e v	Version 1.0	Correction	R e v
845	2	[Weide 91]	[Weide+91]	
857	2	suggests that the assertion action may be coded, although any appropriate	suggests how an assertion action could be coded, but any appropriate	3
860	2	[Stroustrup 88 Stroustrup92a]	[Stroustrup 88 Stroustrup92a]	
862	2	Stroustrup ... Stroustrup	Stroustrup ... Stroustrup	
872	2	system.err.println("Assertion violation " + ToGMTstring(when) ToGMTstring deprecated after JDK 1.1.	system.err.println("Assertion violation " + toGMTstring(when)	3
876	2	This combination is unusually powerful, with the preferred coding style of maximal dynamic binding producing maintainable, robust Objective-C applications.	This combination is unusually powerful. With the preferred coding style of maximal dynamic binding, there have been chronic difficulties in producing maintainable, robust Objective-C applications.	3
876	2	a full set of contact assertions	a full set of contract assertions	
879	2	Carillo ... Carillo	Carrillo ... Carrillo	
902	2	As assertions are limited in the extent to which they can detect all failures, this technique is not assured to find all failures resulting from random input generation.	As assertions are limited in the extent to which they can detect all failures, this technique should be viewed as an adjunct to adequate testing achieved with appropriate test design patterns.	3
909	2	[Meyer 92]	[Meyer 92a]	
909	2	universal qualifiers	universal quantifiers	
910	2	cannot be asserted in Eiffel.	cannot be directly asserted in Eiffel (such a relation can be wrapped in a Boolean function, however.)	3
914	2	[Shaw 81]	[Shaw 84]	
914	2	[Warmer+99]	[Warmer+98]	
915	2	[MacGregor 93a]	[McGregor 93a]	
915	2	[Bates 92]	delete	

Pg.	S e v	Version 1.0	Correction	R e v
915	2	universal qualifiers	universal quantifiers	
916	2	[Jézéquel 96a].	[Jézéquel 96].	
916	2	[Cox 92]	[Cox+91]	
924	2	Table 18.2 layout incorrect, doesn't indicate it is a contingency matrix.	First column in bold, don't extend header rules to leftmost column.	
949	2	[Stroustrup 92a]	[Stroustrup 92a]	
951	2	[Skublics 96]	[Skubics+96]	
955	2	[Dauchy+92]	[Dauchy+93]	
962	2	Design presents patterns for developing a test harness are grouped by the main components of a test harness.	Test harness design patterns are grouped by the main components of a test harness.	
966	2	The test case must be implemented in a language that can access the interface of IUT and that has sufficient to achieve this test design. The obvious solution is to use the same language as the IUT.	The test case must be implemented in a language that can access the interface of IUT and fully supports the test design. The IUT language or a powerful scripting language like Tcl will do the job.	
971	2	[Beck 94]	[Beck 94a]	
974	2	[Beck 94]	[Beck 94a]	
977	2	OUT	IUT	
985	2	[Thielen 92]	[Thielen 92]	
995	2	[Beck 94]	[Beck 94a]	
997	2	The primary advantage of passing the <code>out</code> (or <code>CUT</code>) to the	The primary advantage of passing the <code>OUT</code> (or <code>CUT</code>) to the	
1007	2	[McGregor+94b]	[McGregor+94c]	
1007	2	[Beck 94]	[Beck 94a]	

Pg.	S e v	Version 1.0	Correction	R e v
1018	2	Finally, a truly nasty preprocessor trick may be used to make everything visible at file scope.	Finally, a truly nasty preprocessor trick may be used to make all explicit private and public declarations visible at file scope.	
1019	2	Eiffel example inconsistent with standards	<pre> class INSPECTOR -- Adapted from [Barbey 97] 81 feature viewer: INTERNAL; make (object: CUT) is local i: Integer; do create viewer.make; Io.putstring (viewer.class_name (object)); Io.putint (viewer.field_count (object)); from i := 1; until i > viewer.field_count (object) loop Io.putstring(viewer.field_name (i, object)); i:= i+1; end; end; -- make end -- Inspector </pre>	3
1044	2	[Beck 94]	[Beck 94a]	
1046	2	The members of this collection may be	TestSuite objects may be	
1050	2	[Beck 94]	[Beck 94a]	
1050	2	pass/DNP	pass/no pass	
1052	2	[Seipmann+94]	[Siepmann+94]	
1059	2	- comment	-- comment	
1060	2	Post-Test IUT State	Expected Post-Test IUT State	
1061	2	Expected Output Action	Expected Output Actions	
1062	2	[...]	< ... >	
1063	2	[Jézéquel 96a]	[Jézéquel 96]	
1068	2	interactive/incremental	iterative/incremental	

Pg.	S e v	Version 1.0	Correction	R e v
1072	2	Figure A.1: Plan; CUT = Case under test	Plan	
1072	2	Figure A1 : Driver Stubs	Driver, Stubs	
1072	2	Figure A.1: Name of Cut	Name of CUT	
1075	2	baseline. A specification ... procedures [IEEE 90a].	baseline. A specification ... procedures [IEEE 610].	
1076	2	message parameters and instance.	message parameters and instance variables.	
1076	2	Compare top-down.	Compare top-down testing.	
1079	2	friend	friend function	
1079	2	cluster head. A single class that uses the others as instance variables or as message parameters. A small cluster ... constituents.	cluster head. A class that integrates supporting classes into a useful whole.	
1086	2	event. ... stimulus	event. ... stimulus	
1089	2	[Raymond 90]	[Raymond 91]	
1095	2	[Raymond 90]	[Raymond 91]	
1098	2	nondeterministic. ... finite state automation	nondeterministic. ... finite state automaton	
1113	2	testing. ... system testing	testing. ... system test	
1113	2	[Voas 92]	[Voas+91]	
1123	2	[Chillarege+96]	[Chillarege 96]	
1134	2	[McGregor+94]	[McGregor+94c]	
1136	2	[Parrish+93]	[Parrish+93b]	
1138	2	[Rumbaugh+99]	[Rumbaugh+98]	

Pg.	S e v	Version 1.0	Correction	R e v
1139	2	[Shaler+92] Sally J. Shaler	[Shlaer+92] Sally J. Shlaer	
1140	2	[Stroustroup 88] ... [Stroustroup 92a] ... [Stroustroup 92b]	[Stroustrup 88] ... [Stroustrup 92a] ... [Stroustrup 92b]	