الاتحـــــــــاد الدولـــــــــي للاتصـــــــــالات

ITU-T

قطـــاع تقييس الاتصـــالات
في الاتحاد الدولي للاتصالات

Z.143

(2006/03)

السلسلة Z: اللغات والجوانب العامة للبرمجيات في أنظمة الاتصالات

تقنيات الوصف الشكلي (FDT) – الاختبار وترميز ضبط الاختبار (TTCN)

الإصدار 3 من الاختبار وترميز ضبط الاختبار (TTCN-3): الدلالة التشغيلية

التوصيـــة ITU-T Z.143

الاتحاد الدولي للاتصالات

توصيات السلسلة Z الصادرة عن قطاع تقييس الاتصالات

اللغات والجوانب العامة للبرمجيات في أنظمة الاتصالات

تقنـــيات الوصـــف الشـــكلي (FDT)	
لغة المواصفة والوصف (SDL)	Z.109–Z.100
تطبيق تقنيات الوصف الشكلي	Z.119–Z.110
مخطط تعاقب الرسائل (MSC)	Z.129–Z.120
لغة تعريف الغرض الموسّعة (eODL)	Z.139–Z.130
الاختبار وترميز ضبط الاختبار (TTCN)	**Z.149–Z.140**
ترميز متطلبات المستعملين (URN)	Z.159–Z.150
لغـــات البرمجـــة	
CHILL: لغة المستوى الرفيع لدى قطاع تقييس الاتصالات	Z.209–Z.200
لغـــة الإنســـان–الآلـــة	
مبادئ عامة	Z.309–Z.300
قواعد النَظْم الأساسية وإجراءات التحاور	Z.319–Z.310
لغة الإنسان–الآلة (MML) الموسّعة من أجل مطاريف العرض المرئي	Z.329–Z.320
مواصفة السطح البيني الإنسان–الآلة	Z.349–Z.330
السطوح البينية الإنسان–الآلة الموجهة للمعطيات	Z.359–Z.350
السطوح البينية الإنسان–الآلة من أجل إدارة شبكات الاتصالات	Z.379–Z.360
الجـــودة	
جودة برمجيات الاتصالات	Z.409–Z.400
مظاهر الجودة للتوصيات المرتبطة بالبروتوكولات	Z.459–Z.450
الطـــرائق	
طرائق للتثبت من الصلاحية وللاختبار	Z.519–Z.500
البرمجيـــات الوســـيطة	
بيئة المعالجة الموزعة	Z.609–Z.600

لمزيد من التفاصيل، يرجى الرجوع إلى قائمة التوصيات الصادرة عن قطاع تقييس الاتصالات.

التوصيـــة ITU-T Z.143

الإصدار 3 من الاختبار وترميز ضبط الاختبار (TTCN-3): الدلالة التشغيلية

ملخص

تحدد هذه التوصية الدلالة التشغيلية *للإصدار 3 من الاختبار وترميز ضبط الاختبار* (TTCN-3). وهذه الدلالة ضرورية لتفسير الخصائص التي تتواءم مع الإصدار TTCN-3 بشكل لا لبس فيه. وتستند هذه التوصية إلى اللغة الأساسية للإصدار TTCN-3 المحددة في التوصية ITU-T Z.140.

المصدر

وافقت لجنة الدراسات 17 (2005-2008) التابعة لقطاع تقييس الاتصالات بتاريخ 16 مارس 2006 على التوصية ITU-T Z 143 بموجب الإجراء الوارد في التوصية ITU-T A.8.

تمهيـــد

الاتحاد الدولي للاتصالات وكالة متخصصة للأمم المتحدة في ميدان الاتصالات وتكنولوجيات المعلومات والاتصالات (ICT). وقطاع تقييس الاتصالات (ITU-T) هو هيئة دائمة في الاتحاد الدولي للاتصالات. وهو مسؤول عن دراسة المسائل التقنية والمسائل المتعلقة بالتشغيل والتعريفة، وإصدار التوصيات بشأنها بغرض تقييس الاتصالات على الصعيد العالمي.

وتحدد الجمعية العالمية لتقييس الاتصالات (WTSA) التي تجتمع مرة كل أربع سنوات المواضيع التي يجب أن تدرسها لجان الدراسات التابعة لقطاع تقييس الاتصالات وأن تُصدر توصيات بشأنها.

وتتم الموافقة على هذه التوصيات وفقاً للإجراء الموضح في القرار رقم 1 الصادر عن الجمعية العالمية لتقييس الاتصالات.

وفي بعض مجالات تكنولوجيا المعلومات التي تقع ضمن اختصاص قطاع تقييس الاتصالات، تعد المعايير اللازمة على أساس التعاون مع المنظمة الدولية للتوحيد القياسي (ISO) واللجنة الكهرتقنية الدولية (IEC).

ملاحظـــة

تستخدم كلمة "الإدارة" في هذه التوصية لتدل بصورة موجزة سواء على إدارة اتصالات أو على وكالة تشغيل معترف بها.

والتقيد بهذه التوصية اختياري. غير أنها قد تضم بعض الأحكام الإلزامية (بهدف تأمين قابلية التشغيل البيني والتطبيق مثلاً). ويعتبر التقيّد بهذه التوصية حاصلاً عندما يتم التقيّد بجميع هذه الأحكام الإلزامية. ويستخدم فعل "يجب" وصيغ ملزمة أخرى مثل فعل "ينبغي" وصيغها النافية للتعبير عن متطلبات معينة، ولا يعني استعمال هذه الصيغ أن التقيّد بهذه التوصية إلزامي.

حقوق الملكية الفكرية

يسترعي الاتحاد الانتباه إلى أن تطبيق هذه التوصية أو تنفيذها قد يستلزم استعمال حق من حقوق الملكية الفكرية. ولا يتخذ الاتحاد أي موقف من القرائن المتعلقة بحقوق الملكية الفكرية أو صلاحيتها أو نطاق تطبيقها سواء طالب بها عضو من أعضاء الاتحاد أو طرف آخر لا تشمله عملية إعداد التوصيات.

وعند الموافقة على هذه التوصية، لم يكن الاتحاد قد تلقى إخطاراً بملكية فكرية تحميها براءات الاختراع يمكن المطالبة بها لتنفيذ هذه التوصية. ومع ذلك، ونظراً إلى أن هذه المعلومات قد لا تكون هي الأحدث، يوصى المسؤولون عن تنفيذ هذه التوصية بالاطلاع على قاعدة المعطيات الخاصة ببراءات الاختراع في مكتب تقييس الاتصالات (TSB) في الموقع http://www.itu.int/ITU-T/ipr/.

© ITU 2010

جميع الحقوق محفوظة. لا يجوز استنساخ أي جزء من هذه المنشورة بأي وسيلة كانت إلا بإذن خطي مسبق من الاتحاد الدولي للاتصالات.

المحتـــويات

		الصفحة
1	مجال التطبيق	1
2	المراجع	1
3	التعاريف والمختصرات	1
1.3	التعاريف	1
2.3	المختصرات	1
4	مقدمة	1
5	هيكلية هذه التوصية	2
6	القيود	2
7	الاستعاضة عن الأشكال المقتضبة	3
1.7	ترتيب خطوات الاستعاضة	3
2.7	الاستعاضة عن الثوابت الإجمالية ومعلمات الوحدة	4
3.7	تضمين عمليات استقبال مفردة في بيانات alt	3
4.7	تضمن نداءات altstep القائمة في حد ذاتها في بيانات alt	3
5.7	الاستعاضة عن البيانات interleave	5
6.7	الاستعاضة عن عمليات trigger	17
8	دلالة الترميز TTCN-3 على أساس المخططات الانسيابية	18
1.8	المخططات الانسيابية	17
2.8	تمثيل سلوك TTCN-3 في شكل مخطط انسيابي	23
3.8	تعاريف الحالات في وحدات TTCN-3	29
4.8	الرسائل ونداءات الإجراء والإجابات والاستثناءات	39
5.8	سجلات النداء للوظائف والخطوات البديلة واختبارات الحالة	42
6.8	إجراء تقييم النموذج TTCN-3	38
9	مقاطع المخطط الانسيابي في بناء TTCN-3	45
1.9	البيان action	40
2.9	البيان activate	41
3.9	البيان alt	46
4.9	النداء altstep	52
5.9	البيان assignment	47
6.9	العملية Call	47
7.9	العملية catch	58
8.9	العملية check	54
9.9	العملية clear المطبقة على المنافذ	61
10.9	العملية connect	57
11.9	تحديد constant	62
12.9	العملية create	59
13.9	البيان deactivate	64
14.9	العملية disconnect	65
15.9	البيان do-while	66
16.9	العملية done المطبقة على المكونات	63
17.9	البيان execute	67
18.9	التعبير	70
18b.9	المقطع <dynamic-error> في مخطط الانسياب	69
19.9	المقطع <finalize-component-init> في مخطط الانسياب	70
20.9	المقطع <init-component-scope> في مخطط الانسياب	73
21.9	المقطع <parameter-handling> في مخطط الانسياب	71
22.9	المقطع <statement-block> في مخطط الانسياب	74
23.9	البيان for	75

الصفحة

24.9 النداء الوظيفي 76
25.9 العملية getcall 80
26.9 العملية getreply 78
27.9 العملية getverdict 81
28.9 البيان goto 79
29.9 البيان if-else 79
30.9 البيان label 82
31.9 البيان log 83
32.9 العملية map 81
33.9 العملية mtc 84
34.9 إعلان port 82
35.9 العملية raise 82
36.9 عملية read المنطبقة على المؤقتات 86
37.9 العملية receive 87
38.9 البيان repeat 90
39.9 العملية reply 88
40.9 البيان return 92
41.9 العملية running المنطبقة على المكونات 94
42.9 العملية running المنطبقة على المؤقتات 97
43.9 العملية self 98
44.9 العملية send 97
45.9 العملية setverdict 100
46.9 العملية start المنطبقة على المكونات 101
47.9 العملية start المنطبقة على المنافذ 102
48.9 العملية start المنطبقة على المؤقتات 103
49.9 العملية stop المنطبقة على المكونات 105
50.9 البيان stop المنطبق على التنفيذ 109
51.9 العملية stop المنطبقة على المنافذ 110
52.9 العملية stop المنطبقة على المؤقتات 111
53.9 العملية system 111
54.9 الإعلان timer 111
55.9 العملية timeout المنطبقة على المؤقتات 113
56.9 العملية unmap 113
57.9 إعلان المتغيرة 114
58.9 البيان while 115

10 قوائم مكوّنات الدلالة التشغيلية 116
1.10 الوظائف والحالات 116
2.10 الكلمات الرئيسية الخاصة 118
3.10 مخططات الانسياب لأوصاف السلوك TTCN-3 119
4.10 مقاطع مخطط الانسياب 119

الإصدار 3 من الاختبار وترميز ضبط الاختبار (TTCN-3): الدلالة التشغيلية

1 مجال التطبيق

تحدد هذه التوصية الدلالة التشغيلية للإصدار 3 من الاختبار وترميز ضبط الاختبار (TTCN-3). وتستند هذه التوصية إلى اللغة الأساسية للإصدار TTCN-3 المحددة في التوصية ITU-T Z.140 [1].

2 المراجع

تتضمن التوصيات التالية لقطاع تقييس الاتصالات وغيرها من المراجع أحكاماً تشكل من خلال الإشارة إليها في هذا النص جزءاً لا يتجزأ من هذه التوصية. وقد كانت جميع الطبعات المذكورة سارية الصلاحية في وقت النشر. ولما كانت جميع التوصيات والمراجع الأخرى تخضع إلى المراجعة، يرجى من جميع المستعملين لهذه التوصية السعي إلى تطبيق أحدث طبعة من التوصيات والمراجع الواردة أدناه. وتُنشر بانتظام قائمة توصيات قطاع تقييس الاتصالات السارية الصلاحية. والإشارة إلى وثيقة في هذه التوصية لا يضفي على الوثيقة في حد ذاتها صفة التوصية.

[1] التوصية ITU-T Z.140 (2006)، الإصدار 3 من الاختبار وترميز ضبط الاختبار: اللغة الأساسية.

3 التعاريف والمختصرات

1.3 التعاريف

لأغراض هذه التوصية، تنطبق المصطلحات والتعاريف الواردة في التوصية ITU-T Z.140 [1].

2.3 المختصرات

تستخدم هذه التوصية المختصرات التالية:

ASN.1 ترميز التركيب المجرد رقم 1

BNF الشكل باخوس–ناور

IDL لغة وصف السطح البيني

MTC مكونة الاختبار الرئيسية

SUT نظام قيد الاختبار

TTCN الاختبار وترميز ضبط الاختبار

4 مقدمة

تحدد هذه المقدمة معنى سلوك الإصدار TTCN-3 بطريقة حدسية لا لبس فيها. وليس الغرض من الدلالة التشغيلية أن تكون شكلية وبالتالي فإن القدرة على أداء براهين رياضية تقوم على هذه الدلالة محدودة جداً.

وتوفر هذه الدلالة التشغيلية نظرة على أساس الحالة بشأن تطبيق وحدة قياسية TTCN. ويتم إدخال أنواع مختلفة من الحالات ويرد وصف معنى مختلف تركيبات TTCN:

1) باستخدام معلومات الحالة لتحديد الظروف المسبقة لتنفيذ تركيبة ما؛

2) بتحديد كيفية تغيير الحالة بحكم تنفيذ التركيبة.

وتقتصر الدلالة التشغيلية على معنى السلوك في الإصدار TTCN-3، أي الوظائف والخطوات البديلة وحالات الاختبار والتحكم بالوحدات القياسية وهيكليات اللغة لتحديد سلوك الاختبار، أي عمليتا **send** و**receive** أو البيانان **if-else-** أو **while**. ويُشرح معنى بعض

تركيبات TTCN-3 بالاستعاضة عنها بتركيبات لغوية أخرى. مثال ذلك أن بيانات **interleave** أشكال مقتضبة لسلسلة من بيانات **alt** المدخلة ويرد شرح كل بيان **interleave** بالاستعاضة عنه بسلسلة مقابلة من بيانات **alt** المدخلة.

وفي معظم الحالات، يقوم تحديد الدلالة للغة ما على شجرة تركيب مجردة للشفرة التي يجب وصفها. ولا تنطبق هذه الدلالة على شجرة تركيب مجردة ولكنها تتطلّب تمثيلاً بيانياً لمواصفات سلوك TTCN-3 بشكل مخططات انسيابية. ويصف المخطط الانسيابي تدفق التحكم في وظيفة ما أو خطوة altstep أو حالة اختبار أو التحكم في وحدة قياسية. وتجري المقابلة مباشرة من مواصفات سلوك الإصدار TTCN-3 إلى مخططات الانسياب.

ملاحظة – إن مقابلة بيانات TTCN-3 في مخططات الانسياب خطوة غير رسمية لا يحددها استخدام قواعد الشكل باخوس–ناور (BNF) في التوصية ITU-T Z.140 [1]. ومرد ذلك أن قواعد الشكل BNF ليست مثلى لمقابلة حدسية، إذ إن عدداً من قواعد الدلالات السكونية مشفرة في قواعد BNF للسماح بعمليات سكونية للتحقق من الدلالة خلال التحقق من الترتيب.

5 هيكلية هذه التوصية

تتألف هذه التوصية من أربعة أجزاء:

1) يصف الجزء الأول (انظر الفقرة 6) القيود على الدلالة التشغيلية، أي المسائل المتعلقة بالدلالة التي لا تشملها هذه التوصية.

2) يحدد الجزء الثاني (انظر الفقرة 7) معنى الترميز المختصر والترميز الموسع TTCN-3 باستبدالها بهيكليات لغوية TTCN-3 أخرى. ويمكن اعتبار عمليات الاستبدال هذه في وحدة ما في TTCN-3 كخطوة معالجة مسبقة قبل تفسير الوحدة وفقاً لوصف الدلالة التشغيلية التالي.

3) يصف الجزء الثالث (انظر الفقرة 8) الدلالات التشغيلية للإصدار TTCN-3 بواسطة تفسير مخطط الانسياب وتغيير الحالات.

4) يحدد الجزء الرابع (انظر الفقرة 9) مقابلة مختلف بيانات TTCN-3 في مقاطع مخطط الانسياب الذي يوفر لبنات البناء لمخططات الانسياب التي تمثل الوظائف والخطوات البديلة altsteps وحالات الاختبار والتحكم في الوحدة.

6 القيود

لا تغطي الدلالة التشغيلية إلا الجوانب السلوكية للإصدار TTCN-3، أي أنها تصف معنى البيانات والعمليات. وهي لا توفر:

أ) دلالة لجوانب البيانات في الإصدار TTCN-3. ويشمل ذلك جوانب مثل التشفير وفك التشفير واستعمال البيانات المستقاة من الخصائص التي لا تتعلق بالإصدار TTCN-3.

ب) دلالة لآلية التجميع. يتعلق التجميع بالجزء الخاص بتعاريف وحدة TTCN-3 ولا يتضمن أي جوانب سلوكية.

ج) دلالة للبيان import. ويتعين استقاء التعاريف في الجزء الخاص بالتعاريف في وحدة TTCN-3. وتعالج الدلالة التشغيلية تعاريف مستقاة، كما لو كانت محددة في الوحدة التي تقوم بالاستقاء.

د) دلالة لتحديد معلمات المنافذ.

7 الاستعاضة عن الأشكال المقتضبة

ينبغي توسيع الأشكال المقتضبة بواسطة التعاريف الكاملة المقابلة على مستوى نصي قبل استخدام هذه الدلالة التشغيلية لشرح سلوك TTCN-3.

والأشكال المقتضبة للإصدار TTCN-3 هي:

- قوائم بمعلمات الوحدات، والإعلانات الثابتة والمتغيرة من النمط نفسه، وقوائم بإعلانات المؤقّت؛
- عمليات استقبال قائمة في حد ذاتها؛
- نداءات **altsteps** قائمة في حد ذاتها؛
- عمليات **trigger**؛
- بيانات **return** و**stop** مفقودة عند نهاية تعاريف الوظائف وحالات الاختبار؛
- بيانات تنفيذ **stop** مفقودة؛
- بيانات **interleave**.

بالإضافة إلى معالجة الأشكال المقتضبة، تتطلب الدلالة التشغيلية معالجة خاصة لمعلمات الوحدات والثوابت الإجمالية، أي الثوابت المحددة في الجزء الخاص بتعاريف الوحدات. ويستعاض عن كل الإحالات إلى معلمات الوحدات والثوابت الإجمالية بقيم ملموسة. وهذا يعني أنه من المفترض تحديد قيمة معلمات الوحدات والثوابت الإجمالية قبل أن تصبح الدلالة التشغيلية صالحة.

الملاحظة 1 – تختلف معالجة معلمات الوحدات والثوابت الإجمالية في الدلالة التشغيلية عن معالجتها في مؤلّف TTCN-3. وتصف الدلالة التشغيلية معنى سلوك TTCN-3 وهي ليست دليلاً لتطبيق مؤلّف TTCN-3.

الملاحظة 2 – تعالج الدلالة التشغيلية المعلمات والثوابت المحلية في مكونات الاختبار، وحالات الاختبار، والوظائف ومتغيرات على غرار التحكم في الوحدة. وينبغي التحقق سكونياً من الاستخدام الخاطئ للثوابت المحلية أو المعلمات **in** و**out** و**inout**.

1.7 ترتيب خطوات الاستعاضة

ينبغي أن تكون خطوات الاستعاضة النصية للأشكال المختصرة والمتغيرات الإجمالية ومعلمات الوحدات بالترتيب التالي:

1) الاستعاضة عن قوائم إعلانات معلمات الوحدات والثوابت والمتغيرات والمؤقّت بإعلانات فردية؛
2) الاستعاضة عن الثوابت الإجمالية ومعلمات الوحدات بقيم ملموسة؛
3) تضمين عمليات الاستقبال في حد ذاتها في بيانات **alt**؛
4) تضمين نداءات **altstep** القائمة في حد ذاتها في بيانات **alt**؛
5) توسيع بيانات **interleave**؛
6) الاستعاضة عن كل عمليات **trigger** بعمليات **receive** وبيانات **repeat** المكافئة؛
7) إضافة **return** في نهاية الوظائف دون بيان **return** وإضافة عمليات **self.stop** في نهاية تعاريف اختبار الحالة دون بيان **stop**؛
8) إضافة **stop** في نهاية قسم التحكم في الوحدة دون بيان **stop**.

ملاحظة – في حال لم يتم اتباع هذا الترتيب في خطوات الاستعاضة، فإن نتيجة عمليات الاستعاضة لن تمثل السلوك المحدد.

2.7 الاستعاضة عن الثوابت الإجمالية ومعلمات الوحدة

إن الثوابت المشار إليها في القسم المعني بتعاريف الوحدة إجمالية بالنسبة للتحكم في الوحدة وجميع مكونات الاختبار التي تستحدث خلال تطبيق الوحدة TTCN-3. ومن المفترض أن تكون معلمات الوحدة ثوابت إجمالية عند التطبيق.

ويستعاض عن جميع الإشارات إلى الثوابت الإجمالية وإلى معلمات الوحدة بالقيم الفعلية قبل أن تبدأ الدلالة التشغيلية بتفسير الوحدة. وإذا أعطيت قيمة ثابت ما أو معلمة وحدة في شكل عبارة عندئذ يجب تقييم هذه العبارة. وبعدئذٍ تحل نتيجة التقييم محل جميع الإشارات إلى الثابت أو إلى معلمة الوحدة.

3.7 تضمين عمليات استقبال مفردة في بيانات alt

عمليات الاستقبال TTCN-3 هي: receive وtrigger وgetcall وgetreply وcatch وcheck وtimeout وdone.

ملاحظة – تنطبق العمليات **receive** و**trigger** و**getcall** و**getreply** و**catch** و**check** عند المنافذ وهي تسمح بعملية التفريع بحكم استقبال الرسائل ونداءات الإجراء والردود والاستثناءات. إن العمليتين **timeout** و**done** ليستا عمليتي استقبال حقيقيتين ولكن يمكن استعمالهما بالطريقة نفسها التي تستخدم فيها عمليات الاستقبال، أي كبدائل في بيانات **alt**. وبالتالي، تعالج الدلالة التشغيلية العمليتين **timeout** و**done** على أنهما عمليتا استقبال.

من الممكن استخدام عملية الاستقبال على أنها بيان في حد ذاته في وظيفة أو خطوة بديلة أو اختبار حالة. ويمكن استخدام عملية **timeout** كذلك على أنها بيان في حد ذاته في التحكم في الوحدة. وفي هذه الحالة، تُعتبر عملية الاستقبال صيغة مختصرة للبيان **alt** مع بديل واحد تحدده عملية الاستقبال. أما بالنسبة إلى الدلالة التشغيلية، فإن أي عملية استقبال قائمة في حد ذاتها يجب الاستعاضة عنها ببيان **alt** ينطوي على بيان الاستقبال.

مثـــــال:

```
// The stand-alone occurrence of
    :
  MyCL.trigger(MyType:?);
    :
// shall be replaced by
    :
  alt {
      []   MyCL.trigger (MyType:?) { }
```

```
  }
   :
// or
   :
  MyPTC.done;
   :
// shall be replaced by
   :
  alt {
     []   MyPTC.done { }
  }
   :
```

4.7 تضمن نداءات altstep القائمة في حد ذاتها في بيانات alt

يسمح الترميز TTCN-3 بنداء الوظائف على غرار الخطوات البديلة في الوظائف، والخطوات البديلة وحالات الاختبار والتحكم في الوحدة. ويعطى **alt** معنى نداء قائم بذاته لخطوة بديلة ما ببيان **alt** مع فرع واحد فقط ينادي الخطوة البديلة. ويكون البيان **alt** مسؤولاً عن اللقطة الفورية التي يتم تقييمها ضمن الخطوة البديلة وعن استدعاء الآلية بالتغيب في حال عدم اختيار أي من الخطوات البديلة.

ملاحظة – لا يمكن لخطوة بديلة مستخدمة في التحكم في الوحدة أن تتضمن سوى بدائل مع عمليات **timeout** وفرع **else**.

مثـــــال:

```
// The stand-alone occurrence of
   :
  myAltstep(MyPar1Val);
   :
// shall be replaced by
   :
  alt {
     []   myAltstep(MyPar1Val) { }
  }
   :
```

5.7 الاستعاضة عن البيانات interleave

يحدد معنى البيان **interleave** بالاستعاضة عنه بسلسلة من بيانات **alt** مدمجة ضمناً ولها نفس المعنى. ويرد وصف خوارزمية تركيب الاستعاضة عن بيان **interleave** في هذه الفقرة. ويجب أن تكون عملية الاستعاضة على مستوى الترتيب.

ولا يمكن ضمن أي بيان **interleave**:

1) استخدام بيانات نقل التحكم **for** و**while** و**do-while** و**goto** و**activate** و**deactivate** و**stop** و**repeat** و**return**؛
2) استدعاء خطوات بديلة؛
3) استدعاء وظائف يحددها المستعمل وتتضمن عمليات اتصال؛
4) حماية فروع بيان **interleave** بعبارات منطق Boolean؛
5) تحديد فروع **else**.

وبسبب هذه القيود، فإن كل البيانات القائمة في حد ذاتها وغير المذكورة (مثلاً، assignment أو **log** أو **send** أو **reply**) وعمليات **call** *المانعة* والبيانات المركبة **interleave** و**if-else** و**alt** يمكن أن تستخدم في البيان **interleave**.

الملاحظة 1 – يمكن معاملة عمليات **call** المانعة وبيانات **if-else** كما لو كانت بيانات قائمة في حد ذاتها إذا لم تكن تنطوي على بيانات **alt**. أما في حال بيانات **alt** مدمجة، فإن البدائل تساهم في بيان **interleave** وتتطلب معاملة خاصة. وتوخياً للتبسيط فإن الخوارزمية الواردة أدناه لا تميّز بين هاتين الحالتين.

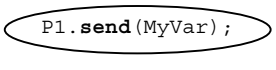
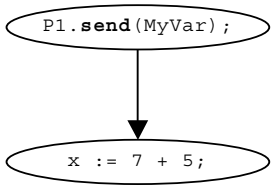
الملاحظة 2 – عمليات **call** غير المانعة ممكنة أيضاً في بيانات **interleave** وتعتبر بيانات قائمة في حد ذاتها. ولا تعمل الخوارزمية الموصوفة في هذه الفقرة إلا في حالة بيانات **interleave** دون بيانات **interleave** مدمجة فيها.

وفي حال بيان **interleave** ينطوي على بيانات **interleave** مدمجة فيه، ينبغي الاستعاضة عن بيانات **interleave** المدمجة قبل تطبيق الخوارزمية.

الملاحظة 3 – بسبب القيود من 1 إلى 5، من الممكن دائماً إيجاد استعاضات محددة فيما يتعلق ببيانات **interleave** المدمجة.

تطبق خوارزمية الاستبدال على تمثيل بياني لبيان **interleave** وتحوله إلى هيكلية متفرعة مكافئة من حيث الدلالة تصف سلسلة من بيانات **alt** المدمجة. لذا، هناك حاجة إلى تمثيل بياني للبيانات القائمة في حد ذاتها والبيانات المركبة **if-else** و**call** *المانعة* و**alt** و**interleave**.

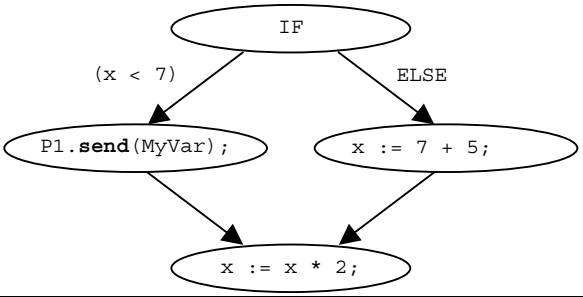
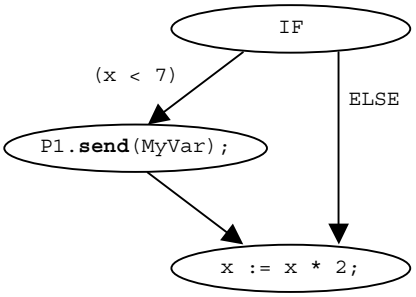
والبيان القائم في حد ذاته عبارة عن عقدة يكون فيها البيان بمثابة كتابة. والمتوالية من البيانات القائمة في حد ذاتها عبارة عن مجموعة من العقد موصولة بخط انسياب، انظر الشكل 1.

<table>
<tr><td><pre>P1.send(MyVar);</pre></td><td> 
</td></tr>
<tr><td>أ) بيان قائم في حد ذاته TTCN-3</td><td>ب) تمثيل بياني للبند أ)</td></tr>
<tr><td><pre>P1.send(MyVar);
x := 7 + 5;</pre></td><td> 
</td></tr>
<tr><td>ج) متوالية من البيانات القائمة في حد ذاتها TTCN-3</td><td>د) تمثيل بياني للبند ج)</td></tr>
</table>

الشكل Z.143/1 – تمثيل بياني للبيانات القائمة في حد ذاتها TTCN-3

يرد في الشكل 2 تمثيل بياني للبيان **if-else**. ويشار إلى بيان **if-else** بعقدة IF لها خطا انسياب موصولان بالبيان الأول في البديلين. ويشار إلى بيان **if-else** دون فرع ELSE بالطريقة نفسها، في حال وجود بيانات تلي البيان **if-else**. في هذه الحال، فإن خط الانسياب الذي يمثل الفرع *else* يكون موصولاً بالبيان الأول الذي يلي البيان **if-else**. ويشار إلى بيان **if-else** دون فرع ELSE ودون بيانات تالية بعقدة IF لها خط انسياب واحد.

الملاحظة 4 – إن الكتابات على خطوط التدفق الواردة في الشكل 1 متوفرة فقط توخياً للوضوح. فالخوارزمية تستخدم فقط العلاقة التي يعبر عنها خط التدفق وليس التسجيل.

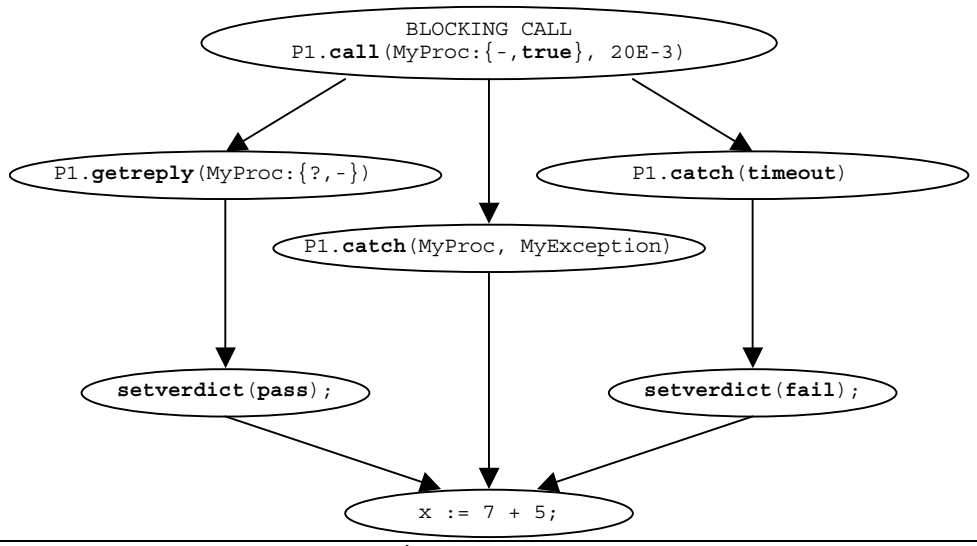
<table>
<tr><td><pre>if (x < 7) {
    P1.send(MyVar);
}
else {
    x := 7 + 5;
}
x := x * 2;</pre></td><td> 
</td></tr>
<tr><td>أ) البيان if-else TTCN-3</td><td>ب) تمثيل بياني للقسم أ)</td></tr>
<tr><td><pre>if (x < 7) {
    P1.send(MyVar);
}
x := x * 2;</pre></td><td> 
</td></tr>
<tr><td>ج) البيان if-else TTCN-3 دون الفرع else</td><td>د) تمثيل بياني للقسم ج)</td></tr>
</table>

الشكل Z.143/2 – تمثيل بياني لبيانات if-else TTCN-3

يرد في الشكل 3 التمثيل البياني للبيان **call** المانع. ويمثل البيان **call** المانع بعقدة BLOCKING-CALL لها خطوط انسياب موصولة ببيانات **getreply** و**catch** لمختلف البدائل.

```
P1.call (MyProc:{-, true}, 20E-3) {
    [] P1.getreply(MyProc:{?,-} {
         setverdict(pass);
       }
    [] P1.catch(MyProc, MyException) {}
    [] P1.catch(timeout) {
         setverdict(fail);
       }
}
x := 7 + 5;
```

أ) بيان Call مانع TTCN-3



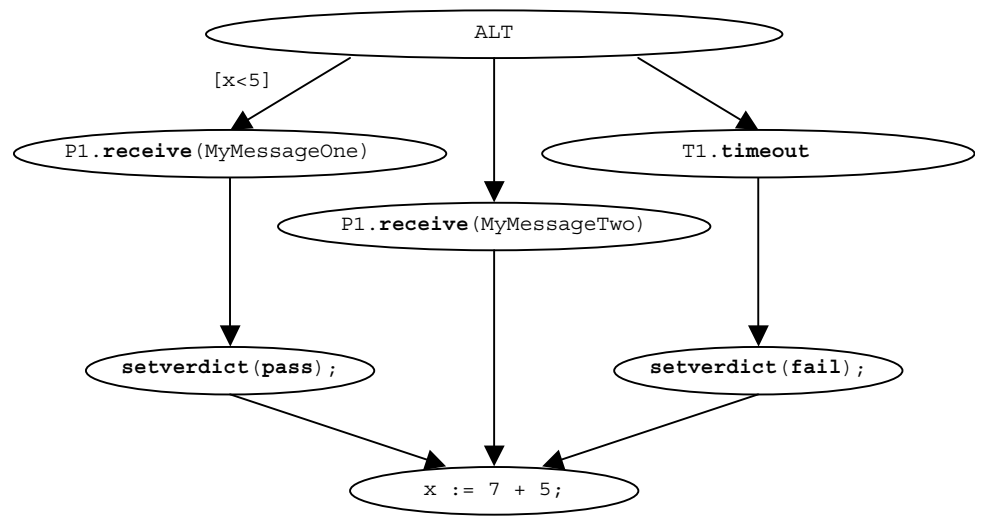
ب) تمثيل بياني للقسم أ)

الشكل Z.143/3 – تمثيل بياني لبيان call مانع TTCN-3

يرد في الشكل 4 التمثيل البياني للبيان **alt**. ويمثّل البيان **alt** بعقدة *alt* لها عدة خطوط انسياب موصولة بمختلف البدائل.

```
alt {
    [x<5] P1.receive(MyMessageOne} {
          setverdict(pass);
       }
    [] P1.receive(MyMessageTwo) {}
    [] T1.timeout {
          setverdict(fail);
       }
}
x := 7 + 5;
```

أ) بيان alt في TTCN-3



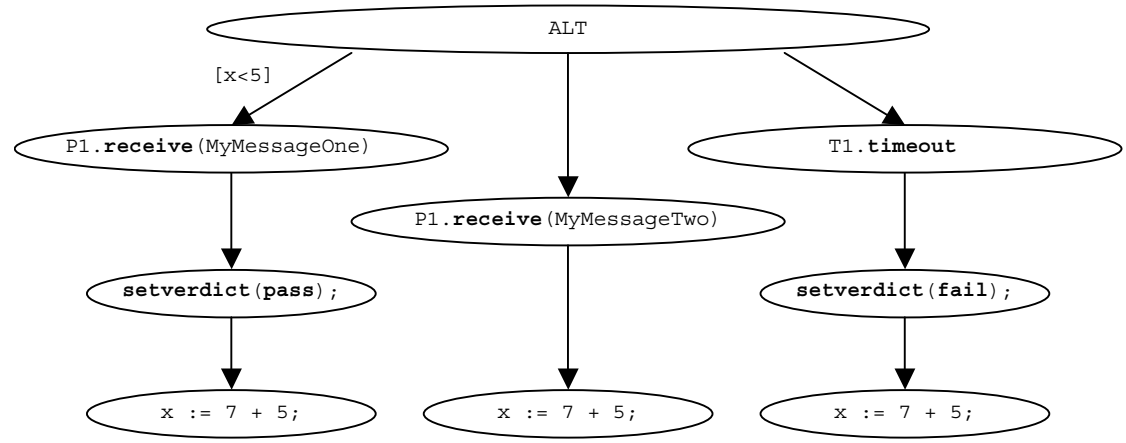
ب) تمثيل بياني للقسم أ

الشكل Z.143/4 – تمثيل بياني للبيان alt في TTCN-3

بشكل عام، تكون التمثيلات البيانية للبيانات **if-else** و**call** *المانع* و**alt** عبارة عن رسوم بيانية الموجهة دون عروة، حيث تجتمع خطوط انسياب مختلف البدائل عند الخروج من البيان. ويمكن بالاستنساخ تحويل هذه الرسوم البيانية الموجهة إلى تمثيلات متفرعة ومكافئة من حيث الدلالة. ويظهر ذلك في الشكل 5 للبيان **alt** الوارد في الشكل 4. وتسمح الخوارزمية الموصوفة أدناه ببناء هذه التمثيلات المتفرعة.

```
alt {
   [x<5] P1.receive(MyMessageOne} {
         setverdict(pass);
         x := 7 + 5;
      }
   [] P1.receive(MyMessageTwo) {
         x := 7 + 5;
      }
   [] T1.timeout {
         setverdict(fail);
         x := 7 + 5;
      }
}
```

أ) بيان alt في TTCN-3 مكافئ من حيث الدلالة للشكل 4–أ)



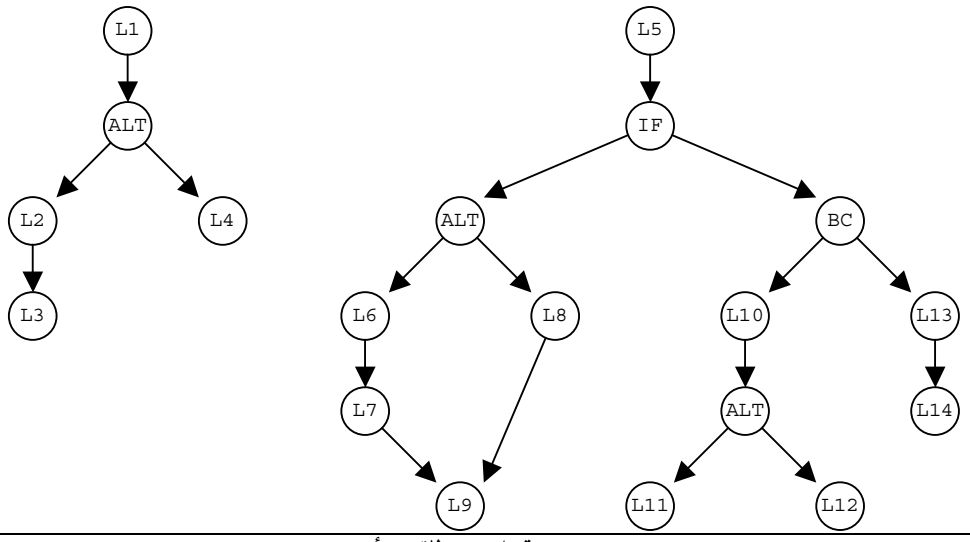
ب) تمثيل بياني للقسم أ (مكافئ من حيث الدلالة للشكل 4–ب)

الشكل Z.143/5 – تمثيل بياني لبيان alt في TTCN-3

يمكن وصف بيان **interleave** بواسطة رسم بياني يتكون من مجموعة من الرسوم البيانية الفرعية، يكون كل منها مبنياً على أساس بيانات مستقلة والبيانات المركبة **if-else** و**call** *المانعة* و**alt**. وتصف الرسوم البيانية الفرعية الموجهة انسيابات التحكم المشذرة. يرد مثال لذلك في الشكل 6. وتشير كتابات العقد في الشكل 6b إلى وسمات البيانات TTCN-3 الواردة في الشكل 6–أ).

```
interleave {
   [] P1.receive(M1) {                          // L1
        alt {                                       // ALT
           [] P1.receive(M3) {                  // L2
                setverdict(pass);               // L3
           }
           [] T1.timeout { }                        // L4
        }

     }
   [] P2.receive(M2) {                          // L5
        if (x < 5 ) {                               // IF
           alt {                                    // ALT
              [] P2.receive(M4) {               // L6
                   setverdict(pass);            // L7
              }
              [] Comp1.done { }                 // L8
           }
           x := 7 + 5;                          // L9
        }
        else {
           P3.call(MyProcTempl, 20E-3) {    // BC (= BLOCKING CALL)

              [] P3.getreply(ReplyTempl) {      // L10
                   alt {                            // ALT
                       [] P2.receive(M5) { }// L11
                       [] P2.receive(M6) { }// L12
                   }
              }
              [] P3.catch(timeout) {            // L13
                   setverdict(fail);            // L14
              }
           }
        }
     }
}
```

أ) بيان interleave في TTCN-3



ب) تمثيل بياني للقسم أ)

الشكل Z.143/6 – تمثيل بياني لبيان interleave في TTCN-3

ومن الناحية الشكلية، يمكن وصف البيان **interleave** بالرسم $GI = (St, F)$ حيث:

St هي مجموعة بيانات TTCN-3 المسموح بها

$F \subseteq (St \times St)$ تصف علاقة الانسياب

إن عبارة " بيانات TTCN-3 *المسموح بها*" تشير إلى القيود السكونية من 1 إلى 5 الواردة أعلاه.

وبالنسبة إلى خوارزمية البناء يتعين تحديد الوظائف التالية:

- الوظيفة REACHABLE تعيد كل البيانات التي يمكن بلوغها من بيان s في الرسم البياني GI = (St, F):

$$\underline{REACHABLE} \quad (s, GI) = \{ s \} \cup \{ stmt \mid stmt \in St \wedge \exists(s = x_1, x_2, \ldots, x_n = stmt) \text{ where } x_i \in St, i \in \{1...n\} \wedge (x_i, x_{i+1}) \in F\}$$

- الوظيفة *SUCCESSORS* تعيد كل العناصر اللاحقة لبيان s في الرسم البياني $GI = (St, F)$:

$$\underline{SUCCESSORS} \quad (s, GI) = \{ stmt \mid stmt \in St \wedge (s, stmt) \in F\}$$

- الوظيفة *ENABLED* تعيد كل البيانات في الرسم البياني $GI = (St, F)$ التي ليس لها عناصر سابقة:

$$\underline{ENABLED} \quad (GI) = \{ stmt \mid stmt \in St \wedge (F \cap (S \times \{s\}) = \varnothing)\}$$

- الوظيفة *KIND* تعيد نوع أو نمط بيان TTCN-3 في رسم بياني يمثل بيان **interleave**.

- الوظيفة *DISCARD* تحذف بيان s أو مجموعة بيانات S من رسم بياني $GI = (St, F)$ وتعيد الرسم البياني الناتج $GI' = (St', F')$:

بالنسبة إلى العقد المفردة:

$$\underline{DISCARD} \quad (s, GI) = GI' \text{ where: } GI' = (St', F'), \text{ with } St' = St \backslash \{s\} \text{ and } F' = F \cap (St \backslash \{s\} \times St \backslash \{s\}).$$

بالنسبة إلى مجموعات العقد:

$$\underline{DISCARD} \quad (S, GI) = GI' \text{ where: } GI' = (St', F'), \text{ with } St' = St \backslash S \text{ and } F' = F \cap (St \backslash S \times St \backslash S).$$

- الوظيفة *RECEIVING* تأخذ مجموعة من بيانات رسم بياني GI وتعيد كل بيانات الاستقبال:

$$\underline{RECEIVING} \quad (S) = \{ stmt \mid stmt \in St \wedge \underline{KIND}(stmt) \in \{\text{receive, trigger, getcall, getreply, catch, check, done, timeout}\}\}$$

- الوظيفة *RANDOM* تختار عنصراً s بشكل عشوائي من مجموعة معينة S وتعيد العنصر s.

$$\underline{RANDOM} \quad (S) = s \text{ where } s \in S$$

إن خوارزمية التفريع (انظر الشكل 7) عبارة عن إجراء تكراري حيث يتم عند كل نداء تكراري بناء العقد اللاحقة لعقدة معينة. ويتم توفير هذا الإجراء في شبه شفرة مشابهة للبرمجية C التي تستخدم الوظائف المحددة أعلاه وبعض الرموز الرياضية الإضافية.

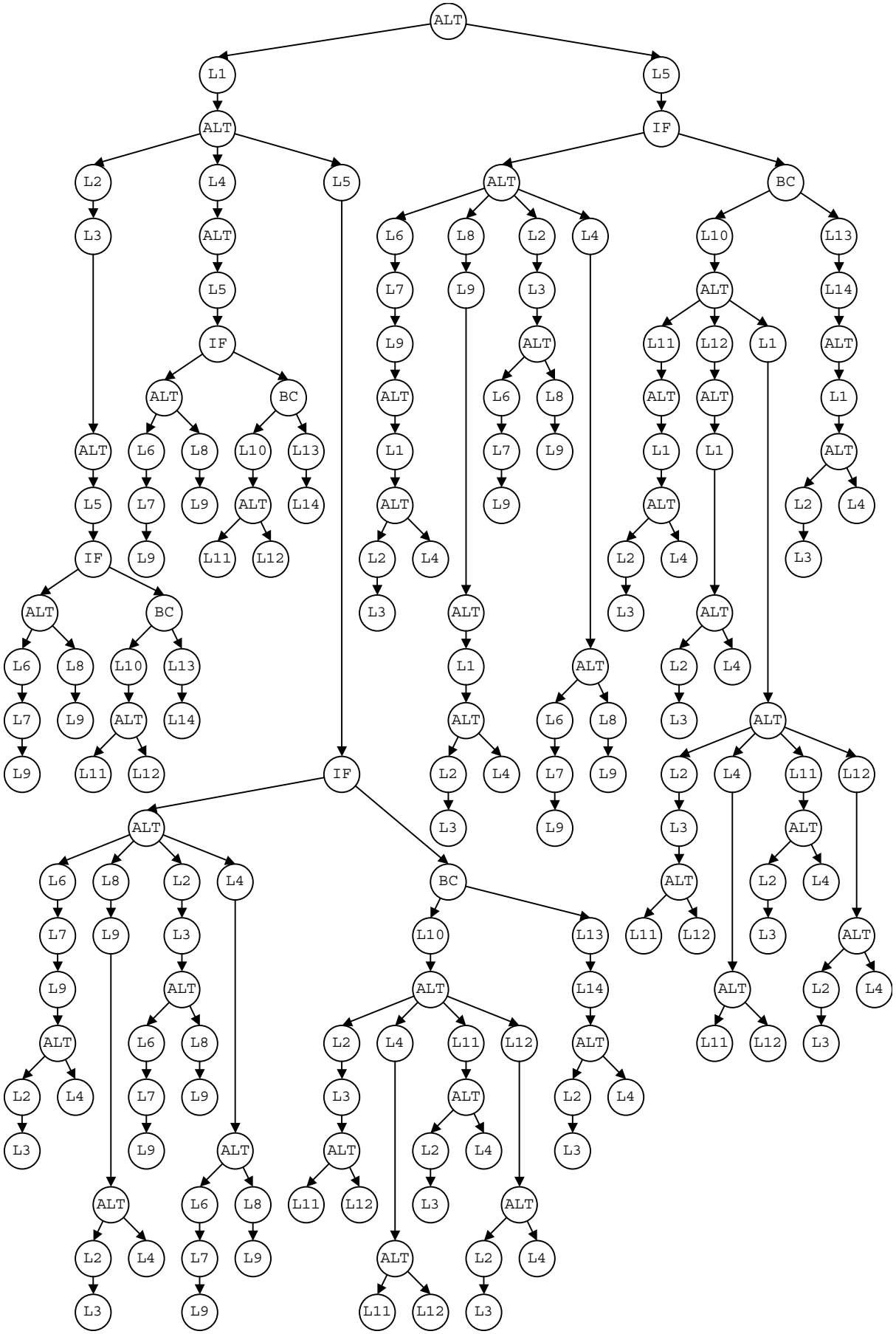
```
CONSTRUCT-SUCCESSORS (statementType *predecessor, graphType GI) {
    // - statementType refers to the type of a node of the tree that is constructed
    // - *predecessor refers to the last node that has been created
    // - graphType denotes type of the graph of TTCN-3 statements
    // - GI is called by value and refers to the subgraph consisting of all remaining TTCN-3
    // statements that have to be taken into consideration

    var graphType myGraph;
    var statementType i, myStmt;
    var statementType *newStmt, *firstInBranch; // pointers for new statement nodes in the
                                                // tree that is constructed recursively

    // Retrieving sets of TTCN-3 statements that have no predecessors in 'GI'
    var statementSet enabStmts := ENABLED(GI);          // all statements without predecessor
    var statementSet enabRecStmts := RECEIVING(enabStmts);  // receiving statements in 'enabStmts'
    var statementSet enabNonRecStmts := enabStmts\enabRecStmts;
                                                        // non receiving statements in 'enabStmts'

    if (GI.St == ∅) {   // We assume that GI.St refers to the set of statements in GI
        return;         // No statements are left, termination criterion of Recursion
    }
    elseif (enabNonRecStmts != ∅) { // Handling of non receiving statements in 'enabStmts'

        myStmt := RANDOM(enabNonRecStmts);
                // There can only be one statement in 'enabNonRec', because the Algorithm
                // continues the construction until there is a branch that contributes to
                // the interlave statement.
        newStmt := create(myStmt, predecessor);
                // Creation of a new tree node representing 'myStmt' in the tree
                // and update of pointers in 'newStmt' and 'predecessor'.

        if (KIND(myStmt) == IF || KIND(myStmt) == BLOCKING_CALL) {
            for each i in SUCCESSORS(myStmt, GI) {

                firstInBranch := create(i, newStmt);
                        // Creation of a second node for the first statement of in a branch due to
                        // an if-else statement.
                        // Note, this create statement will be used to create tree nodes
                        // representing the receiving statements in blocking call operations.

                myGraph := DISCARD({i, myStmt} ∪ REACHABLE(myStmt, GI)\REACHABLE(i, GI))
                        // Removal of i, myStmt and all statements that are reachable from
                        // myStmt but not reachable from i. The latter considers the branching of
                        // a flow of control in a subgraph of GI.

                CONSTRUCT-SUCCESSORS(firstInBranch, myGraph);       // NEXT RECURSION STEP
            }
        }
        elseif (KIND(myStmt) == ALT) {
            for each (i in SUCCESSORS(myStmt, GI) {

            CONSTRUCT-SUCCESSORS(mystmt, DISCARD(REACHABLE(myStmt, GI)\REACHABLE(i, GI)));
                    // NEXT RECURSION STEP, the DISCARD(REACHABLE(myStmt, GI)\REACHABLE(i, GI))
                        // argument considers the branching of a flow of control due to different
                        // receiving events.
            }
        }
        else {      // mystmt is a stand-alone statement
            CONSTRUCT-SUCCESSORS(newSonNode,                DISCARD(myStmt,                 GI));
                    // NEXT RECURSION STEP
        }
    }
    else {  // Handling of receiving events that interleave
        if (KIND(predecessor) != ALT) { // an alt node is missing and has to be created, if the
                                        // interleaving is not influenced by an embedded alt statement
            predecessor := create(ALT, predecessor);
        }

        for each i in enabRecStmts) {
            newStmt := create(i, predecessor);              // New tree node
            CONSTRUCT-SUCCESSORS(newStmt, DISCARD(i, GI));  // NEXT RECURSION STEP(S)
        }
    }
}
```

الشكل Z.143/7 – خوارزمية الاستعاضة عن بيانات interleave في TTCN-3

بدايةً، تستدعي الوظيفة CONSTRUCT-SUCCESSORS (انظر الشكل 7) مع *عقدة جذرية* لشجرة خالية ورسم بياني لبيانات TTCN-3 يصف بيان **interleave** الواجب الاستعاضة عنه. وعند الانتهاء من هذه الوظيفة، يمكن استخدام *العقدة الجذرية* من أجل النفاذ إلى الشجرة المبنية.

ويؤدي تطبيق الوظيفة CONSTRUCT-SUCCESSORS على بيان **interleave** الوارد في الشكل 6 إلى الشجرة المبينة في الشكل 8. وتقابل الوسمات بيانات الشكل 6-أ) والوسمات المتعددة هي نتيجة استنساخ الشفرة. والشفرة TTCN-3 التي تقابل الشجرة المبينة في الشكل 8 واردة في الشكل 9.

الملاحظة 5 – إن مثال تطبيق الخوارزمية في الشكل 7 (انظر الأشكال 6 و8 و9) شامل جداً. وهو يبين معظم الحالات الخاصة، بما في ذلك تفريع وانضمام خطوط الانسياب وبيان **alt** مندمج وبيان **call** *مانع* وبيان **if-else**.



الشكل Z.143/8 – نتيجة تطبيق خوارزمية الشكل 7 على بيان interleave في الشكل 6

```
alt {                                                                                   // ALT
    []  P1.receive(M1} {                                                                // L1
            alt {                                                                       // ALT
                []  P1.receive(M3) {                                                    // L2
                        setverdict(pass);                                               // L3
                        alt {                                                           // ALT
                            []  P2.receive(M2) {                                        // L5
                                    if (x < 5 ) {                                       // IF
                                        alt {                                           // ALT
                                                []  P2.receive(M4) {                    // L6
                                                    setverdict(pass);                   // L7
                                                    x := 7 + 5;                         // L9
                                                }
                                                []  Comp1.done {                        // L8
                                                    x := 7 + 5;                         // L9
                                    }   }           }
                                    else {
                                        P3.call(MyProcTempl, 20E-3) { // BC (= BLOCKING CALL)
                                            []  P3.getreply(ReplyTempl) {               // L10
                                                alt {                                   // ALT
                                                    []  P2.receive(M5) { }              // L11
                                                    []  P2.receive(M6) { }              // L12
                                                }   }
                                            []  P3.catch(timeout) {                     // L13
                                                setverdict(fail);                       // L14
                    }   }       }   }   }       }
                []  T1.timeout {                                                        // L4
                        alt {                                                           // ALT
                            []  P2.receive(M2) {                                        // L5
                                    if (x < 5 ) {                                       // IF
                                        alt {                                           // ALT
                                                []  P2.receive(M4) {                    // L6
                                                    setverdict(pass);                   // L7
                                                    x := 7 + 5;                         // L9
                                                }
                                                []  Comp1.done {                        // L8
                                                    x := 7 + 5;                         // L9
                                    }   }           }
                                    else {
                                        P3.call(MyProcTempl, 20E-3) { // BC (= BLOCKING CALL)
                                            []  P3.getreply(ReplyTempl) {               // L10
                                                alt {                                   // ALT
                                                    []  P2.receive(M5) { }              // L11
                                                    []  P2.receive(M6) { }              // L12
                                                }   }
                                            []  P3.catch(timeout) {                     // L13
                                                setverdict(fail);                       // L14
                    }   }       }   }   }       }
                []  P2.receive(M2) {                                                    // L5
                        if (x < 5 ) {                                                   // IF
                            alt {                                                       // ALT
                                []  P2.receive(M4) {                                    // L6
                                    setverdict(pass);                                   // L7
                                    x := 7 + 5;                                         // L9
                                    alt {                                               // ALT
                                        []  P1.receive(M3) {                            // L2
                                            setverdict(pass);                           // L3
                                        }
                                        []  T1.timeout { }                              // L4
                                }   }
                                []  Comp1.done {                                        // L8
                                    x := 7 + 5;                                         // L9
                                    alt {                                               // ALT
                                        []  P1.receive(M3) {                            // L2
                                            setverdict(pass);                           // L3
                                        }
                                        []  T1.timeout { }                              // L4
                                }   }
                                []  P1.receive(M3) {                                    // L2
                                    setverdict(pass);                                   // L3
                                    alt {                                               // ALT
                                        []  P2.receive(M4) {                            // L6
                                            setverdict(pass);                           // L7
                                            x := 7 + 5;                                 // L9
                                        }
                                        []  Comp1.done {                                // L8
                                            x := 7 + 5;                                 // L9
                                }   }       }
                                []  T1.timeout {                                        // L4
                                    alt {                                               // ALT
```

```
                                        []  P2.receive(M4) {                          // L6
                                            setverdict(pass);                         // L7
                                            x := 7 + 5;                               // L9
                                        }
                                        []  Comp1.done {                              // L8
                                            x := 7 + 5;                               // L9
                    }   }       }   }       }
                    else {
                        P3.call(MyProcTempl, 20E-3) {          // BC (= BLOCKING CALL)
                            []  P3.getreply(ReplyTempl) {                             // L10
                                alt {                                                 // ALT
                                    []  P1.receive(M3) {                              // L2
                                        setverdict(pass);                             // L3
                                        alt {                                         // ALT
                                            []  P2.receive(M5) { }                    // L11
                                            []  P2.receive(M6) { }                    // L12
                                        }   }
                                    []  T1.timeout {                                  // L4
                                        alt {                                         // ALT
                                            []  P2.receive(M5) { }                    // L11
                                            []  P2.receive(M6) { }                    // L12
                                        }   }
                                    []  P2.receive(M5) {                              // L11
                                        alt {                                         // ALT
                                            []  P1.receive(M3) {                      // L2
                                                setverdict(pass);                     // L3
                                            }
                                            []  T1.timeout { }                        // L4
                                        }   }
                                    []  P2.receive(M6) {                              // L12
                                        alt {                                         // ALT
                                            []  P1.receive(M3) {                      // L2
                                                setverdict(pass);                     // L3
                                            }
                                            []  T1.timeout { }                        // L4
                                }   }       }   }
                            []  P3.catch(timeout) {                                   // L13
                                setverdict(fail);                                     // L14
                                alt {                                                 // ALT
                                    []  P1.receive(M3) {                              // L2
                                        setverdict(pass);                             // L3
                                    }
                                    []  T1.timeout { }                                // L4
    }   }       }   }   }       }   }
[]  P2.receive(M2) {                                                                  // L5
        if (x < 5 ) {                                                                 // IF
            alt {                                                                     // ALT
                []  P2.receive(M4) {                                                  // L6
                    setverdict(pass);                                                 // L7
                    x := 7 + 5;                                                       // L9
                    alt {                                                             // ALT
                        []  P1.receive(M1} {                                          // L1
                            alt {                                                     // ALT
                                []  P1.receive(M3) {                                  // L2
                                    setverdict(pass);                                 // L3
                                }
                                []  T1.timeout { }                                    // L4
                    }   }       }   }
                []  Comp1.done {                                                      // L8
                    x := 7 + 5;                                                       // L9
                    alt {                                                             // ALT
                        []  P1.receive(M1} {                                          // L1
                            alt {                                                     // ALT
                                []  P1.receive(M3) {                                  // L2
                                    setverdict(pass);                                 // L3
                                }
                                []  T1.timeout { }                                    // L4
                    }   }       }   }
                []  P1.receive(M3) {                                                  // L2
                    setverdict(pass);                                                 // L3
                    alt {                                                             // ALT
                        []  P2.receive(M4) {                                          // L6
                            setverdict(pass);                                         // L7
                            x := 7 + 5;                                               // L9
                        }
                        []  Comp1.done {                                              // L8
                            x := 7 + 5;                                               // L9
                    }   }       }
                []  T1.timeout {                                                      // L4
                    alt {                                                             // ALT
```

```
                                [] P2.receive(M4) {                                     // L6
                                    setverdict(pass);                                   // L7
                                    x := 7 + 5;                                         // L9
                                }
                                [] Comp1.done {                                         // L8
                                    x := 7 + 5;                                         // L9
            }   }       }   }       }
            else {
                P3.call(MyProcTempl, 20E-3) {                       // BC (= BLOCKING CALL)
                    [] P3.getreply(ReplyTempl) {                                        // L10
                        alt {                                                           // ALT
                            [] P2.receive(M5) {                                         // L11
                                alt {                                                   // ALT
                                    [] P1.receive(M1} {                                 // L1
                                        alt {                                           // ALT
                                            [] P1.receive(M3) {                         // L2
                                                setverdict(pass);                       // L3
                                            }
                                            [] T1.timeout { }                           // L4
                                }   }       }   }
                            [] P2.receive(M6) {                                         // L12
                                alt {                                                   // ALT
                                    [] P1.receive(M1} {                                 // L1
                                        alt {                                           // ALT
                                            [] P1.receive(M3) {                         // L2
                                                setverdict(pass);                       // L3
                                            }
                                            [] T1.timeout { }                           // L4
                                }   }       }   }
                            [] P1.receive(M1} {                                         // L1
                                alt {                                                   // ALT
                                    [] P1.receive(M3) {                                 // L2
                                        setverdict(pass);                               // L3
                                        alt {                                           // ALT
                                            [] P2.receive(M5) { }                       // L11
                                            [] P2.receive(M6) { }                       // L12
                                    }   }
                                    [] T1.timeout {                                     // L4
                                        alt {                                           // ALT
                                            [] P2.receive(M5) { }                       // L11
                                            [] P2.receive(M6) { }                       // L12
                                    }   }
                                    [] P2.receive(M5) {                                 // L11
                                        alt {                                           // ALT
                                            [] P1.receive(M3) {                         // L2
                                                setverdict(pass);                       // L3
                                            }
                                            [] T1.timeout { }                           // L4
                                    }   }
                                    [] P2.receive(M6) {                                 // L12
                                        alt {                                           // ALT
                                            [] P1.receive(M3) {                         // L2
                                                setverdict(pass);                       // L3
                                            }
                                            [] T1.timeout { }                           // L4
                                }   }       }   }       }   }
                        [] P3.catch(timeout) {                                          // L13
                            setverdict(fail);                                           // L14
                            alt {                                                       // ALT
                                [] P1.receive(M1} {                                     // L1
                                    alt {                                               // ALT
                                        [] P1.receive(M3) {                             // L2
                                            setverdict(pass);                           // L3
                                        }
                                        [] T1.timeout { }                               // L4
}       }   }   }       }   }       }   }
```

الشكل Z.143/9 – شفرة TTCN-3 مكافئة من حيث الدلالة لبيان interleave في الشكل 6

6.7 الاستعاضة عن عمليات trigger

تقوم عملية **trigger** بفرز الرسائل على أساس معيار معين من المواءمة من تدفق من الرسائل عند منفذ معين. ويمكن وصف دلالة العملية **trigger** بالاستعاضة عنها بعمليتين **receive** وبيان **goto**. وفي الدلالة التشغيلية يُفترض أن هذه الاستعاضة تتم على مستوى التركيب.

المثـــــال 1:

```
// The following trigger operation ...
      alt {
       []  MyCL.trigger (MyType:?) { }
      }
// shall be replaced by ...
  alt {
      []   MyCL.receive (MyType:?) { }
      []   MyCL.receive {
           repeat
       }
  }

  }
```

إذا كان بيان **trigger** مستخدم في بيان **alt** أكثر تعقيداً، فتتم عملية الاستعاضة بالطريقة نفسها.

المثـــــال 2:

```
// The following alt statement includes a trigger statement ...
  alt {
      []   PCO2.receive {
           stop;
       }
      []   MyCL.trigger (MyType:?) { }
      []   PCO3.catch {
           setverdict(fail);
           stop;
       }
  }
// which will be replaced by
  alt {
      []   PCO2.receive {
           stop;
       }
      []   MyCL.receive (MyType:?) { }
      []   MyCL.receive {
           repeat;
       }
      []   PCO3.catch {
           setverdict(fail);
           stop;
       }
  }
```

8 دلالة الترميز TTCN-3 على أساس المخططات الانسيابية

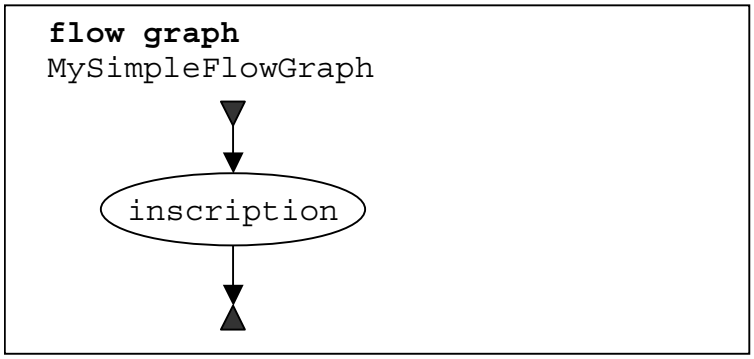
تستند الدلالة التشغيلية للترميز TTCN-3 إلى تفسير المخططات الانسيابية. وتتناول هذه الفقرة المخططات الانسيابية (انظر الفقرة 1.8) وتشرح بناء المخططات الانسيابية التي تمثل التحكم في وحدة TTCN-3 وحالات الاختبار والخطوات البديلة والوظائف وتعاريف نمط المكون (انظر الفقرة 2.8)، وتعّرف حالات الوحدة والمكون لوصف حالات تطبيق الوحدة TTCN-3 (انظر الفقرة 3.8) وتصف معالجة الرسائل ونداءات الإجراء عن بعد وإجابات نداء الإجراء عن بُعد والاستثناءات (انظر الفقرة 4.8)، وتشرح إجراء تقييم التحكم في الوحدة وحالات الاختبار (انظر الفقرة 6.8).

1.8 المخططات الانسيابية

مخطط الانسياب عبارة عن مخطط موجه مؤلف من عقد وأطراف موسومة. واجتياز مخطط الانسياب يصف انسياب التحكم الممكن خلال تطبيق وصف السلوك الممثّل.

1.1.8 إطار مخطط الانسياب

يجب وضع المخططات الانسيابية في إطار يحدد حدود المخططات الانسيابية. ويأتي اسم المخططات الانسيابية بعد عبارة **flow graph** (وهي ليست من الكلمات الرئيسية في اللغة النواة TTCN-3) ويكون في الزاوية العليا اليسرى من المخططات الانسيابية. ومن قبيل الاصطلاح نفترض أن اسم المخططات الانسيابية يشير إلى وصف السلوك TTCN الذي يمثله المخططات الانسيابية. ويرد في الشكل 10 مخططات انسياب بسيط.



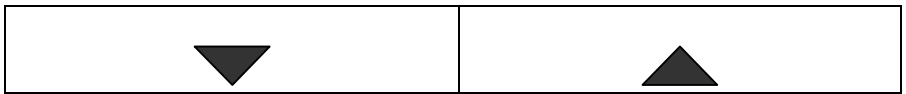
الشكل Z.143/10 – مخطط انسياب بسيط

2.1.8 عقد مخططات الانسياب

تتألف المخططات الانسيابية من عقد ابتدائية وعقد انتهائية وعقد أساسية وعقد مرجعية.

1.2.1.8 العقد الابتدائية

تصف العقد الابتدائية نقطة انطلاق مخطط الانسياب. ويكون لمخطط الانسياب عقدة ابتدائية واحدة فقط. ويّبين الشكل 11–أ عقدة ابتدائية.



ب) عقدة انتهائية في مخطط انسياب	أ) عقدة ابتدائية في مخطط انسياب

الشكل Z.143/11 – العقد الابتدائية والانتهائية

2.2.1.8 العقد الانتهائية

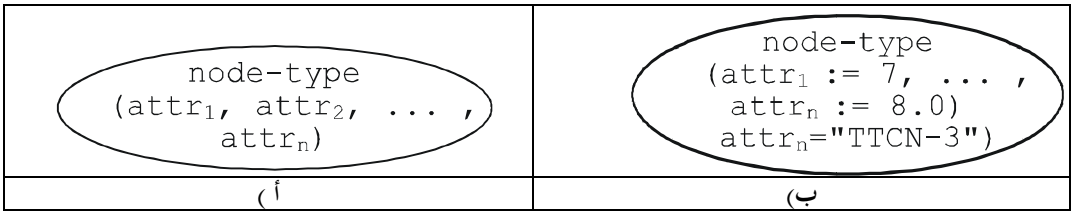
تصف العقد الانتهائية النقاط الانتهائية في مخطط انسياب. وقد يكون لمخطط انسياب عدة عقد انتهائية أو في حالات العرى، لا يكون له أي عقدة انتهائية. وينبغي توصيل العقد الأساسية (انظر الفقرة 3.2.1.8) والعقد المرجعية (انظر الفقرة 4.2.1.8) التي ليس لها عقد لاحقة لعقدة انتهائية للإشارة إلى أنها تصف الإجراء الأخير عبر مخطط الانسياب. ويبين الشكل 11–ب عقدة انتهائية.

3.2.1.8 العقد الأساسية

تصف العقدة الأساسية وحدة تنفيذ، أي يتم تنفيذها في خطوة واحدة. يكون للعقدة الأساسية نمط معين، ووفقاً لهذا النمط يمكن أن يكون لها قائمة بالنعوت المصاحبة. وفي الشكل 12 عقدتان أساسيتان.

وفي شرح عقدة أساسية، تأتي النعوت بعد نمط العقدة وتوضع بين قوسين. ويساعد النمط والنعوت على تحديد الإجراء الواجب تنفيذه عند بناء اللغة الممثلة. وتصف النعوت المعلومات التي ينبغي استخراجها من البناء TTCN-3 المقابل.

للنعوت قيم تستخرجها الدلالة التشغيلية بالإشارة إلى اسم النعت. وعند الضرورة يمكن تخصيص قيم واضحة في العقد الأساسية باستخدام علامة التخصيص ':='. يرد مثال لذلك في الشكل 12–ب).

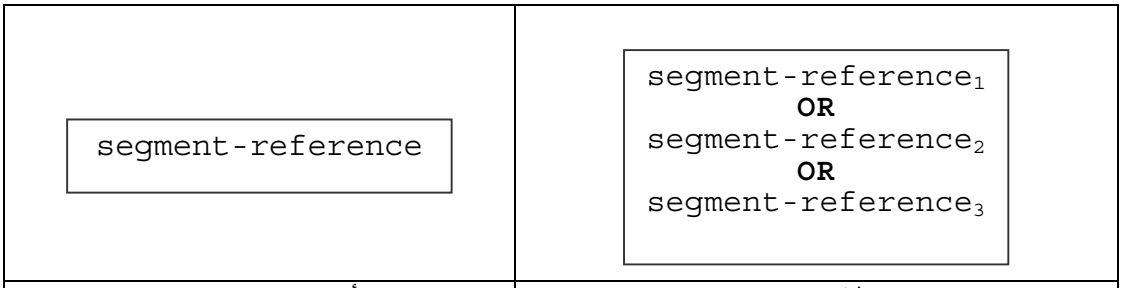


ب)	أ)

الشكل Z.143/12 – عقدة أساسية لها نعوت

4.2.1.8 العقد المرجعية

تحيل العقد المرجعية إلى أجزاء في مخطط الانسياب (انظر الفقرة 4.1.8) هي مخططات انسياب فرعية. ويتحدد معنى عقدة مرجعية ما بالاستعاضة عنها بالجزء المحال إليه في مخطط الانسياب. ويضم شرح العقدة المرجعية إحالة إلى جزء في مخطط الانسياب. ويبيّن الشكل 13-أ) عقدة مرجعية.



أ) عقدة مرجعية مفردة | ب) مجموعة لثلاث عقد مرجعية OR

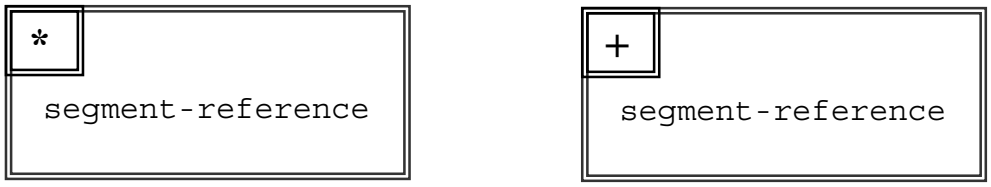
الشكل Z.143/13 – عقدة مرجعية

1.4.2.1.8 مجموعة OR من العقد المرجعية

في بعض الحالات، قد يحل عدد من أجزاء مخطط انسياب محل العقدة المرجعية. وعندئذٍ، يمكن استخدام الأداة OR للإحالة إلى عدة أجزاء في مخطط انسياب (انظر الشكل 13-ب). وفي مخطط الانسياب الفعلي الذي يمثل التحكم في الوحدة أو اختبار حالة أو وظيفة ما، لا يحدد البناء الممثل إلا بديلاً واحداً.

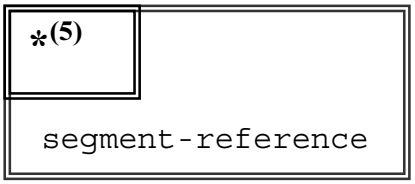
2.4.2.1.8 تعدد ورود العقد المرجعية

في بعض الحالات، يمكن أن ترد العقدة المرجعية نفسها في مخطط انسيابي مرة أو أكثر أو لا ترد مطلقاً. وفي التعبيرات الاعتيادية، يمكن وصف التكرار المحتمل لأجزاء من تعبير معين باستخدام العامل '+' (تكرار واحد أو أكثر) والعامل '*' (لا تكرار أو تكرار واحد أو أكثر). وكما يظهر في الشكل 14، اعتمد هذان العاملان في مخطط انسيابي من خلال إدراج عقد مرجعية في إطار مزدوج يضم رمز العامل المطلوب. وينبغي لخط انسياب وحيد (انظر الفقرة 3.1.8) أن يحل محل العقدة المرجعية عندما لا تكون هنالك عقدة مرجعية (باستخدام عقدة مرجعية في إطار مزدوج يضم رمز العامل '*').



الشكل Z.143/14 – تكرار العقد المرجعية

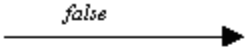
من الممكن وضع حد أقصى للتكرارات الممكنة للعقدة المرجعية في شكل عدد صحيح بين قوسين بعد الرمز '*' أو '+' في العقدة المرجعية مزدوجة الإطار. ويمكن أن تحدث الإحالة المرجعية في الشكل 15 من صفر إلى 5 مرات.

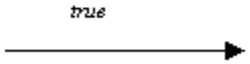


الشكل Z.143/15– التكرار المقيّد لعقدة مرجعية

3.1.8 خطوط الانسياب

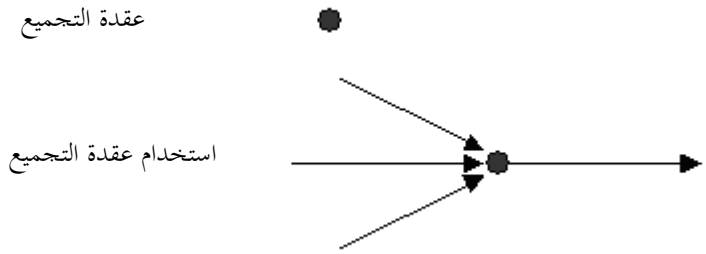
تمثل خطوط الانسياب في شكل أسهم. ويحمل كل خط انسياب كلمة "*صحيح*" أو "*خطأ*" تشير إلى الظرف الذي يتم فيه اختيار خط الانسياب خلال تفسير مخطط الانسياب. ومن قبيل التبسيط يمكن عدم ذكر كلمة "صحيح". وفيما يلي أمثلة عن خطوط الانسياب:





يماثل

وللتمكن من جمع عدة خطوط انسياب في خط واحد على المستوى البياني، تم إدخال عقدة تجميع خاصة. وفيما يلي شكل عقدة التجميع ومثال عن استخدامها:



ثمة صعوبة في رسم خطوط انسياب طويلة في مخططات كبيرة، كما هو ضروري مثلاً لنمذجة كل من **goto** و**label** في الترميز TTCN-3. في هذه الحالة، من الممكن استخدام وسمات لخطوط الانسياب الصادرة والواردة. وفيما يلي أمثلة عن ذلك.

:خط انسياب وارد يحمل وسمة

:خط انسياب صادر يحمل وسمة

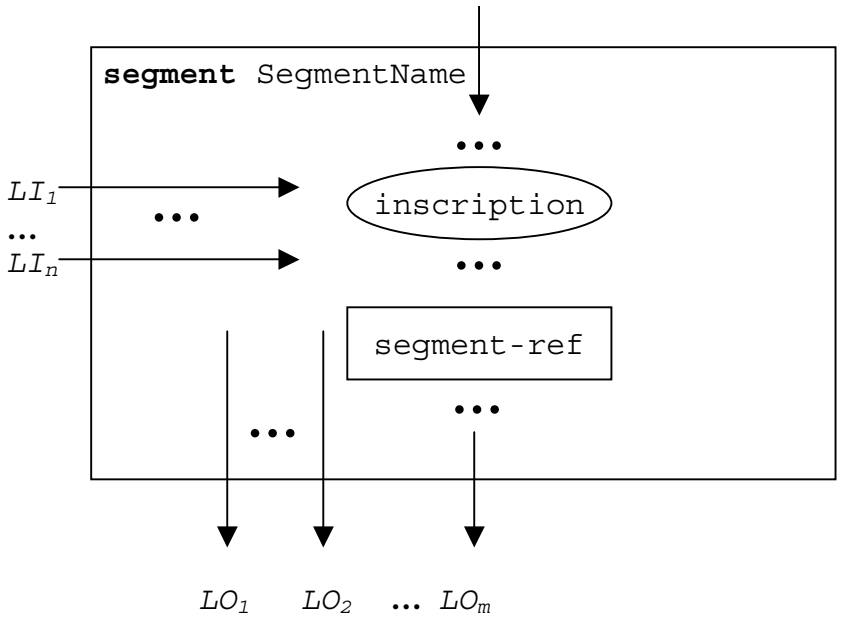
يوصل خط الانسياب الصادر حامل الوسمة بخط انسياب وارد حامل وسمة، إذا كانت الوسمتان متماثلتين. وينبغي أن تكون وسمات خط الانسياب الوارد فريدة. وإذا كانت عدة خطوط انسياب صادرة تحمل الوسمة ذاتها يعتبر ذلك تجميعاً للخطوط باتجاه خط الانسياب الوارد الذي يحمل الوسمة ذاتها.

4.1.8 أجزاء مخطط الانسياب

أجزاء مخطط الانسياب عبارة عن مخططات انسياب فرعية يشار إليها في العقد المرجعية وتحدد معنى هذه العقد المرجعية. ويمكن أن تتضمن عقداً مرجعية أخرى.

كما يبدو في الشكل 16، يكون لأجزاء مخطط الانسياب سطوح بينية محددة بدقة تتكون من خطوط انسياب واردة وصادرة. هناك خط انسياب وارد واحد فقط غير موسوم وخط انسياب صادر واحد غير موسوم أو لا شيء. بالإضافة إلى ذلك، قد يكون هنالك عدة خطوط انسياب واردة وصادرة موسومة. إذ يحتاج الأمر مثلاً إلى خطوط انسياب واردة وصادرة موسومة لوصف معنى بيانات **goto** و**alt** في TTCN-3.

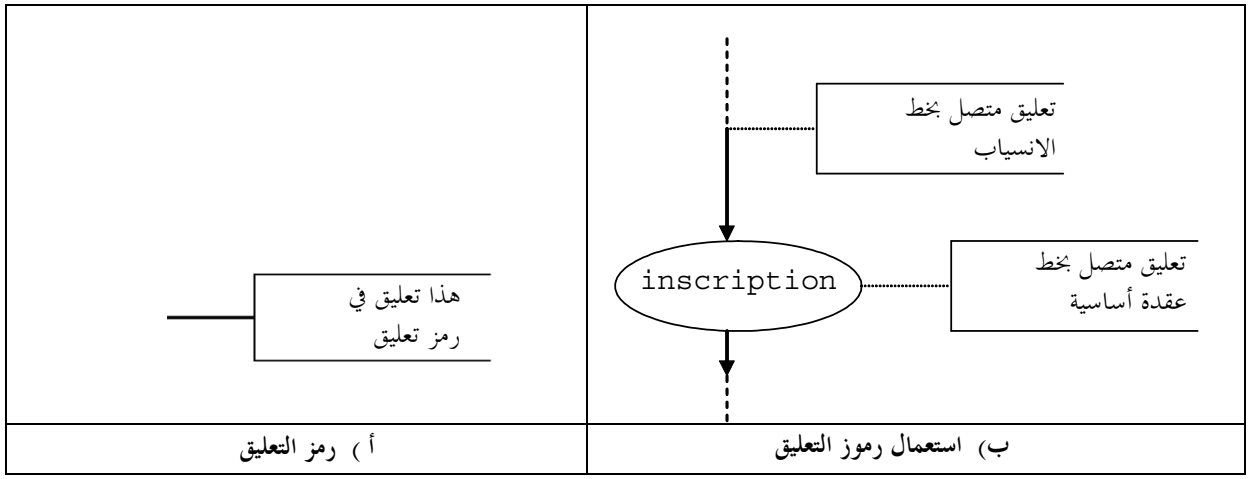
يوضع جزء مخطط الانسياب في إطار وينبغي أن يأتي اسم جزء مخطط الانسياب بعد الكلمة الرئيسية **segment** يتبعه اسم الجزء في الزاوية العليا اليسرى من الإطار. وينبغي لخطوط الانسياب التي تصف السطح البيني لجزء مخطط الانسياب أن تجتاز إطار ذلك الجزء.



الشكل Z.143/16 – بنية وصف جزء مخطط الانسياب

5.1.8 تعليقات

لتحسين قابلية القراءة والاتساق، يمكن استخدام رمز خاص للتعليق لربط التعليقات بعقد مخططات الانسياب وخطوط الانسياب. ويرد في الشكل 17 رمز التعليق وكيفية استخدامه.



الشكل Z.143/17 – تمثيل التعليقات في مخططات الانسياب

6.1.8 معالجة أوصاف مخططات الانسياب

يجتاز إجراء تقييم الدلالة التشغيلية مخططات انسياب تشمل فقط عقداً مرجعية، أي ينبغي تفصيل كل العقد المرجعية بما يقابلها من تعاريف أجزاء مخطط الانسياب. إن وظيفة *NEXT* ضرورية للقيام بهذا الاجتياز، وهي محددة كما يلي:

`successorNodeRef := actualNodeRef.NEXT(bool)` حيث:

- *actualNodeRef* مرجع عقدة أساسية في مخطط الانسياب؛
- *successorNodeRef* مرجع عقدة لاحقة للعقدة التي يحيل إليها *actualNodeRef*؛
- *bool* معلمة Boolean تحدد ما إذا أعيد التابع *true* أو *false* أم لا (انظر الفقرة 3.1.8).

2.8 تمثيل سلوك TTCN-3 في شكل مخطط انسيابي

في الدلالة التشغيلية نفترض أن أوصاف السلوك TTCN-3 معطاة في شكل مجموعة من المخططات الانسيابية، أي لكل وصف للسلوك TTCN-3 ينبغي رسم مخطط انسياب منفصل.

تفسّر الدلالة التشغيلية أنواع تعاريف TTCN-3 التالية على أنها أوصاف للسلوك:

أ) التحكم في الوحدة؛

ب) تعاريف اختبار الحالة؛

ج) تعاريف الوظيفة؛

د) تعاريف الخطوة البديلة؛

هـ) تعاريف نمط المكونة.

يحدد التحكم في الوحدة حملة الاختبارات، أي ترتيب تنفيذ (ربما تكرارياً) حالات الاختبار الفعلية. تحدد تعاريف اختبار الحالة سلوك مكونة الاختبار الرئيسية MTC. وتحدد الوظائف هيكل السلوك. ويتم تنفيذها من خلال التحكم في الوحدة أو من خلال مكونات الاختبار. وتستخدم الخطوات البديلة لتحديد السلوك بالتغيب أو بشكل وظائف لتحديد هيكل السلوك. ويفترض أن تكون تعاريف نمط المكونة أوصاف سلوك لأنها تحدد إنشاء وإعلان وتدميث المنافذ والثوابت والمتغيرات وعوامل التوقيت خلال إنشاء حالة من حالات نمط المكونة.

1.2.8 إجراء رسم مخطط الانسياب

إن مخططات الانسياب الواردة في الأشكال من 18 إلى 22 وأجزاء مخططات الانسياب الواردة في الفقرة 8 عبارة عن نماذج فقط. وتتضمن أقواساً '<' و'>' ينبغي إدراج معلومات فيها لإعداد مخطط انسياب فعلي أو جزء منه.

ويتم رسم مخطط انسياب وحدة TTCN-3 في ثلاث مراحل:

1) لكل بيان TTCN-3 في التحكم في الوحدة وفي تعريف اختبار الحالة والخطوة البديلة والوظيفة ونمط المكونة، يتم إعداد جزء فعلي من مخطط انسياب.

2) للتحكم في الوحدة ولكل تعريف اختبار الحالة والخطوة البديلة والوظيفة ونمط المكونة، يتم إعداد مخطط انسياب فعلي (يتضمن العقد المرجعية).

3) في إجراء متعدد الخطوات، يستعاض عن كل العقد المرجعية في مخططات الانسياب الفعلية بما يقابلها من تعاريف أجزاء مخططات الانسياب إلى أن تقتصر كل مخططات الانسياب على العقدة الابتدائية والعقد الانتهائية والعقد الأساسية.

الملاحظة 1 – تصف العقد الأساسية في مخططات الانسياب وحدات تنفيذ أساسية غير قابلة للقسمة. وتستند الدلالة التشغيلية للسلوك TTCN-3 إلى تفسير هذه العقد الأساسية. وتعرض الفقرة 6.8 طرائق التنفيذ التي تنطبق على هذه العقد فقط.

إن الاستعاضة عن عقدة مرجعية بتعريف جزء مخطط الانسياب المقابل يمكن أن يؤدي إلى عناصر غير موصولة في مخطط الانسياب، أي عناصر لا يمكن الوصول إليها انطلاقاً من العقدة الابتدائية باجتياز المخطط عبر خطوط الانسياب. إذ تتجاهل الدلالة التشغيلية العناصر غير الموصولة في مخطط الانسياب.

الملاحظة 2 – يكون العنصر غير الموصول في مخطط الانسياب نتيجة عملية الاستعاضة الآلية. ولإعداد مخطط انسيابي أمثل، يجب أن يؤخذ بعين الاعتبار أيضاً مختلف تجميعات بيانات TTCN-3. غير أن هذه التوصية ترمي إلى تحديد دلالة صحيحة وشاملة وليس إلى إعداد مخطط انسيابي أمثل.

2.2.8 تمثيل التحكم في الوحدة في شكل مخطط انسيابي

يكون ترتيب عناصر الوحدة TTCN-3 كما يلي:

```
module <identifier> <module-definitions-part> control <statement-block>
```

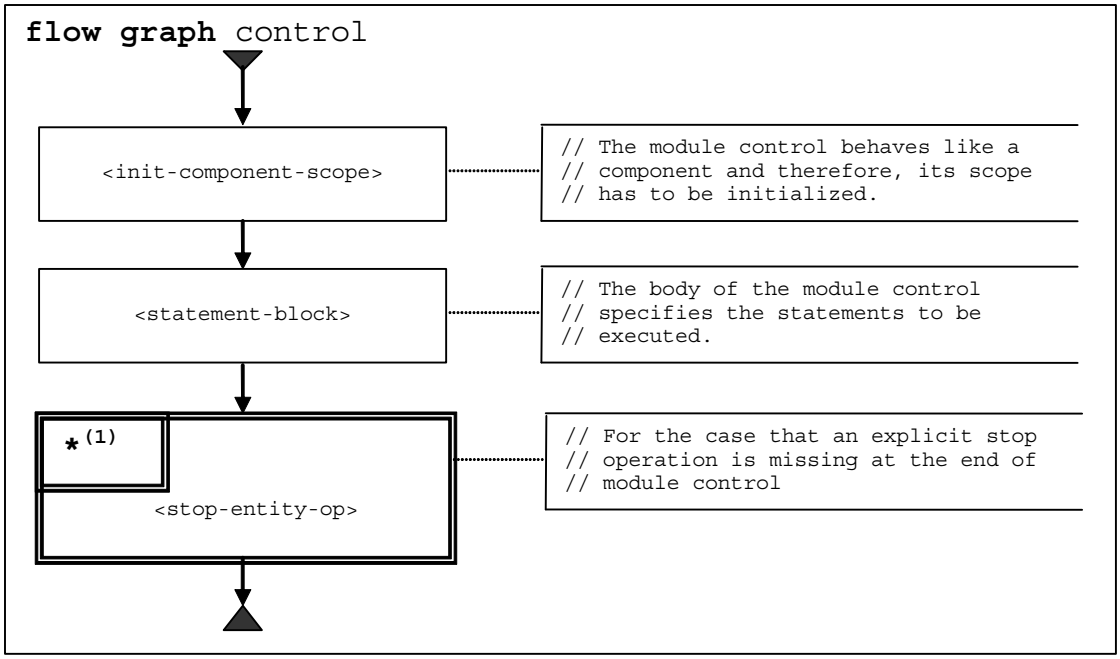
ولتمثيل السلوك في شكل مخطط انسيابي يكفي أن تتوفر المعلومات التالية:

```
module <identifier> <statement-block>
```

وهذا لا يختلف عن تحديد وظيفة ما ولذا فإن تمثيل التحكم في الوحدة في شكل مخطط انسيابي يماثل تمثيل وظيفة ما في شكل مخطط انسيابي (انظر الفقرة 4.2.8). ويكون نفاذ الدلالة إلى مخطط الانسياب الذي يمثل التحكم في الوحدة من خلال اسم الوحدة.

ملاحظة – إن معنى القسم المتعلق بتعاريف الوحدة يقع خارج إطار هذه الدلالة التشغيلية. ويتم تحديد معلمات الوحدة باعتبارها ثوابت إجمالية عند التنفيذ. وينبغي الاستعاضة عن الإحالات إلى معلمات الوحدة بقيمها الفعلية على مستوى الترتيب (انظر الفقرة 3.8).

ويرد في الشكل 18 تمثيل التحكم في الوحدة في شكل مخطط انسيابي. فيحدد اسم المخطط control المخطط الذي يمثل التحكم في الوحدة. وعقد المخطط مصحوبة بتعليقات تصف معنى مختلف العقد. وتغطي العقدة المرجعية <stop-entity-op> الحالة التي لا تحدد فيها أي عملية **stop** واضحة، أي أن الدلالة التشغيلية تفترض أن عملية **stop** مضافة ضمناً.



الشكل Z.143/18 – تمثيل التحكم في الوحدة في شكل مخطط انسيابي

3.2.8 تمثيل اختبار وحدة في شكل مخطط انسيابي

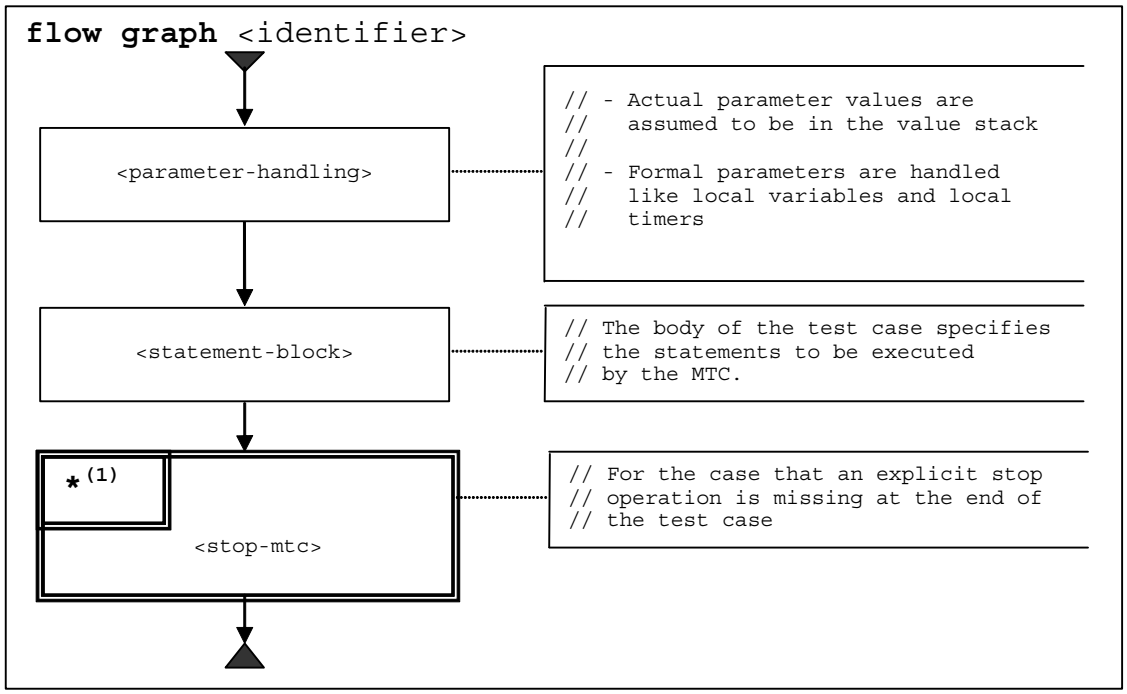
يكون ترتيب تحديد اختبار الحالة TTCN-3 كما يلي:

```
testcase <identifier> (<parameter>) <testcase-interface> <statement-block>
```

يحيل الجزء <testcase-interface> أعلاه إلى عبارتي **runs on** (الإلزامية) و**system** (الاختيارية) في تحديد اختبار الحالة. إن وصف اختبار الحالة في شكل مخطط انسيابي يوضح سلوك مكونة الاختبار الرئيسية MTC. والمعلومات التي تتوفر في <testcase-interface> ليست مفيدة للمكونة MTC. وسوف يستخدمها البيان **execute** ولكن لا حاجة إلى تمثيلها في اختبار حالة في شكل مخطط انسيابي. وبالتالي، يقتصر الأمر في تمثيل المخطط الانسيابي على المعلومات التالية:

```
testcase <identifier> (<parameter>) <statement-block>
```

ويرد في الشكل 19 تمثيل اختبار حالة في شكل مخطط انسيابي. ويحيل اسم المخطط <identifier> إلى اسم اختبار الحالة الممثلة. وعقد المخطط مصحوبة بتعليقات تصف معنى مختلف العقد. وتشمل العقدة المرجعية <stop-entity-op> الحالة التي لا تحدد فيها أي عملية **stop** واضحة للمكونة MTC، أي أن الدلالة التشغيلية تفترض أن عملية **stop** مضافة ضمناً.



الشكل Z.143/19 – تمثيل اختبار حالة في شكل مخطط انسيابي

4.2.8 تمثيل الوظائف في شكل مخطط انسيابي

يكون ترتيب الوظيفة TTCN-3 كما يلي:

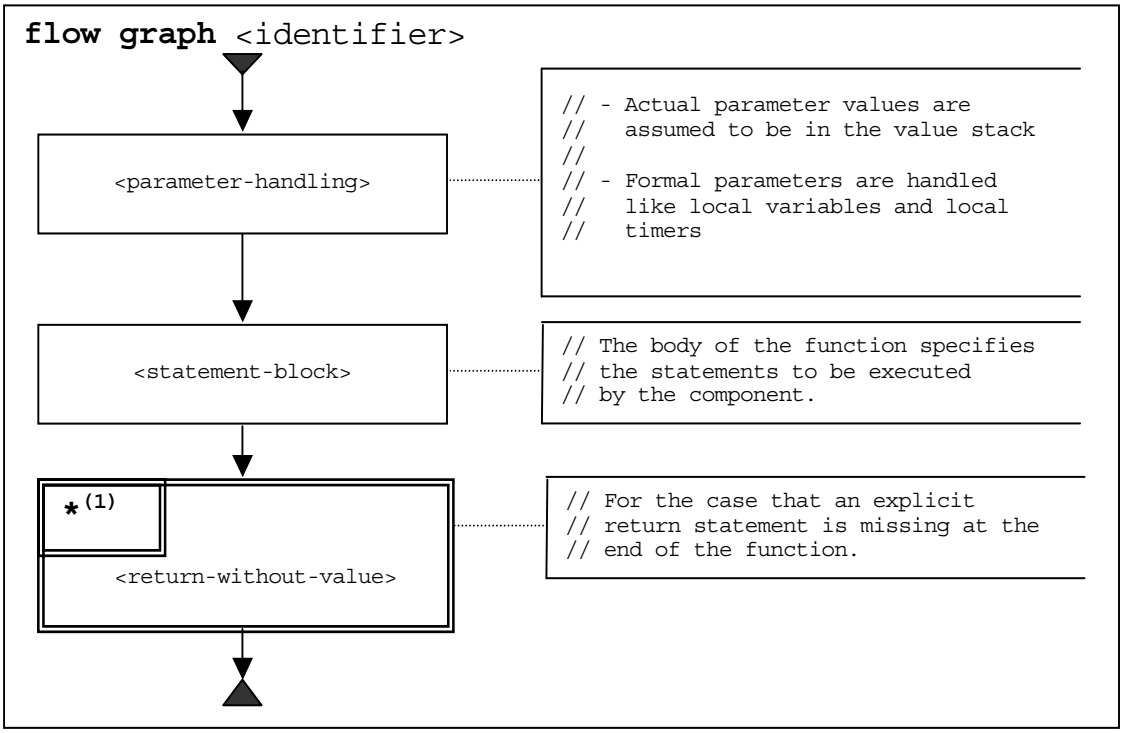
function <identifier> (<parameter>) [<function-interface>] <statement-block>

يحيل الجزء <function-interface> الاختياري أعلاه إلى عبارتي **runs on** و**return** في تعريف الوظيفة. والمعلومات التي تتوفر في <function-interface> ليست مفيدة لوصف السلوك. وسوف تستخدم لعمليات التحقق من الدلالة السكونية ولا حاجة إلى تمثيلها في المخطط الانسيابي. وبالتالي، يقتصر الأمر في تمثيل المخطط الانسيابي على المعلومات التالية:

function <identifier> (<parameter>) <statement-block>

ويكون نفاذ الدلالة إلى مخطط الانسياب الذي يمثل الوظائف من خلال أسماء الوظائف.

ويرد في الشكل 20 تمثيل وظيفة ما في شكل مخطط انسيابي. ويحيل اسم المخطط <identifier> إلى اسم الوظيفة الممثلة. وتشمل العقدة المرجعية <return-without-value> الحالة التي لا تحدد فيها أي عملية **return** واضحة، أي أن الدلالة التشغيلية تفترض أن عملية **return** مضافة ضمناً.



الشكل Z.143/20 – تمثيل الوظائف في شكل مخطط انسيابي

5.2.8 تمثيل الخطوة البديلة في شكل مخطط انسيابي

يكون ترتيب الخطوة البديلة TTCN-3 كما يلي:

```
altstep <identifier> (<parameter>) [<altstep-interface>]
            <constant-variable-timer-declarations>
                { <receiving-branch> | <else-branch> }*
```

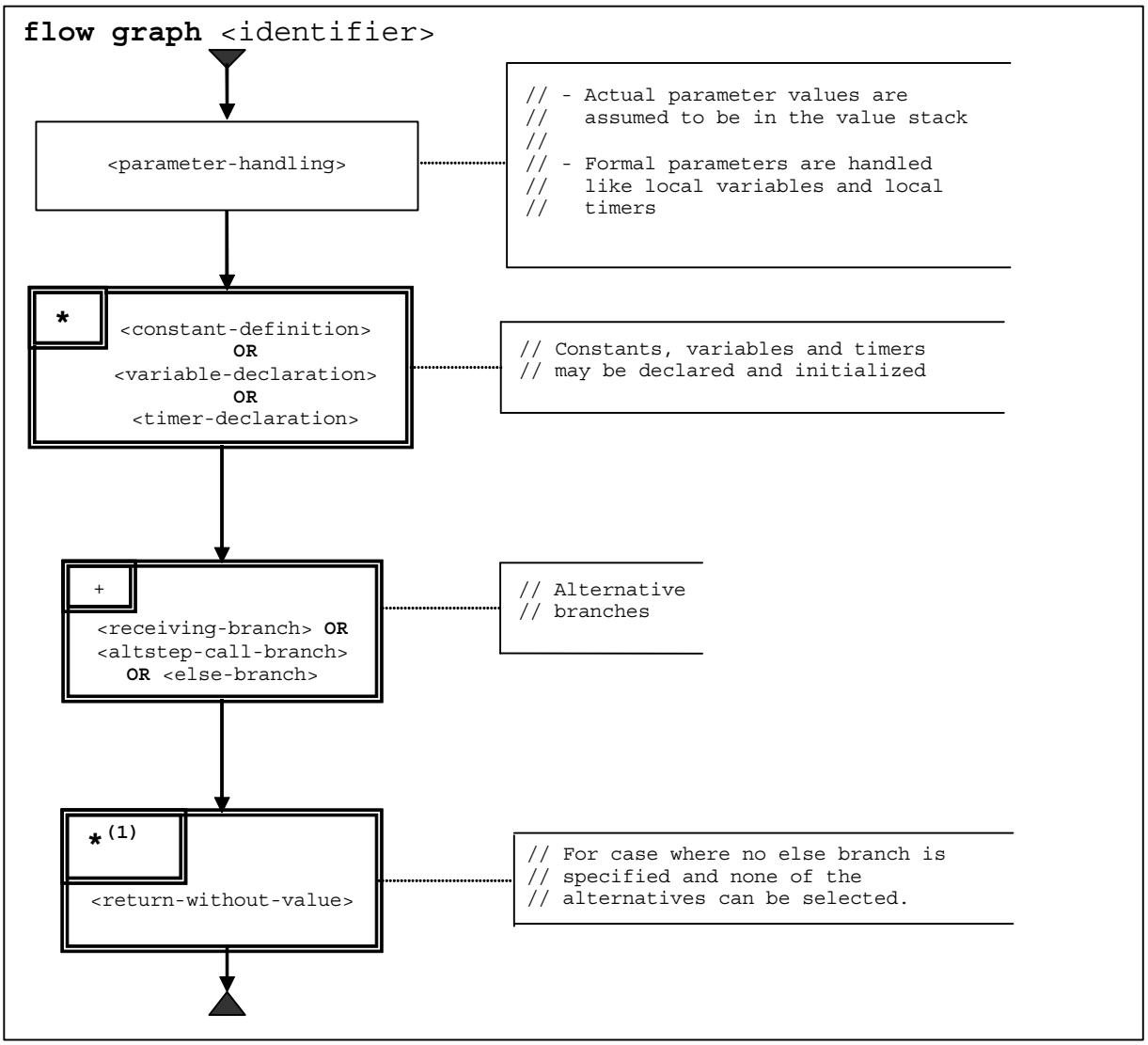
يحيل الجزء الاختياري <altstep-interface> أعلاه إلى عبارة **runs on** في تعريف الخطوة البديلة. والمعلومات التي تتوفر في <altstep-interface> ليست مفيدة لوصف السلوك. وسوف تُستخدم لعمليات التحقق من الدلالة السكونية ولا حاجة إلى تمثيلها في المخطط الانسيابي. وبالتالي، يقتصر الأمر في تمثيل المخطط الانسيابي على المعلومات التالية:

```
altstep <identifier> (<parameter>) [<altstep-interface>]
            <constant-variable-timer-declarations>
             { <receiving-branch> }*
             [ <else-branch> ]
```

ملاحظة – لا تؤخذ في الاعتبار إلا البدائل حتى الفرع else، بما في ذلك الفرع نفسه. ولا يمكن الوصول إلى الفروع التي تلي الفرع الأول else.

ويكون نفاذ الدلالة إلى مخططات الانسياب التي تمثل الخطوات البديلة من خلال أسماء الخطوات البديلة.

ويرد في الشكل 21 تمثيل الخطوة البديلة في شكل مخطط انسيابي. ويحيل اسم المخطط <identifier> إلى اسم الخطوة البديلة الممثلة. وتشمل العقدة المرجعية <unsuccessful-altstep-termination> الحالة حيث تنتهي الخطوة البديلة بعد انتقاء البديل وتنفيذه. وتحدد العقدة المرجعية <unsuccessful-altstep-termination> الحالة التي لا يكون قد نفّذ فيها أي بديل في الخطوة البديلة.



الشكل Z.143/21 – تمثيل الخطوات البديلة في شكل مخطط انسيابي

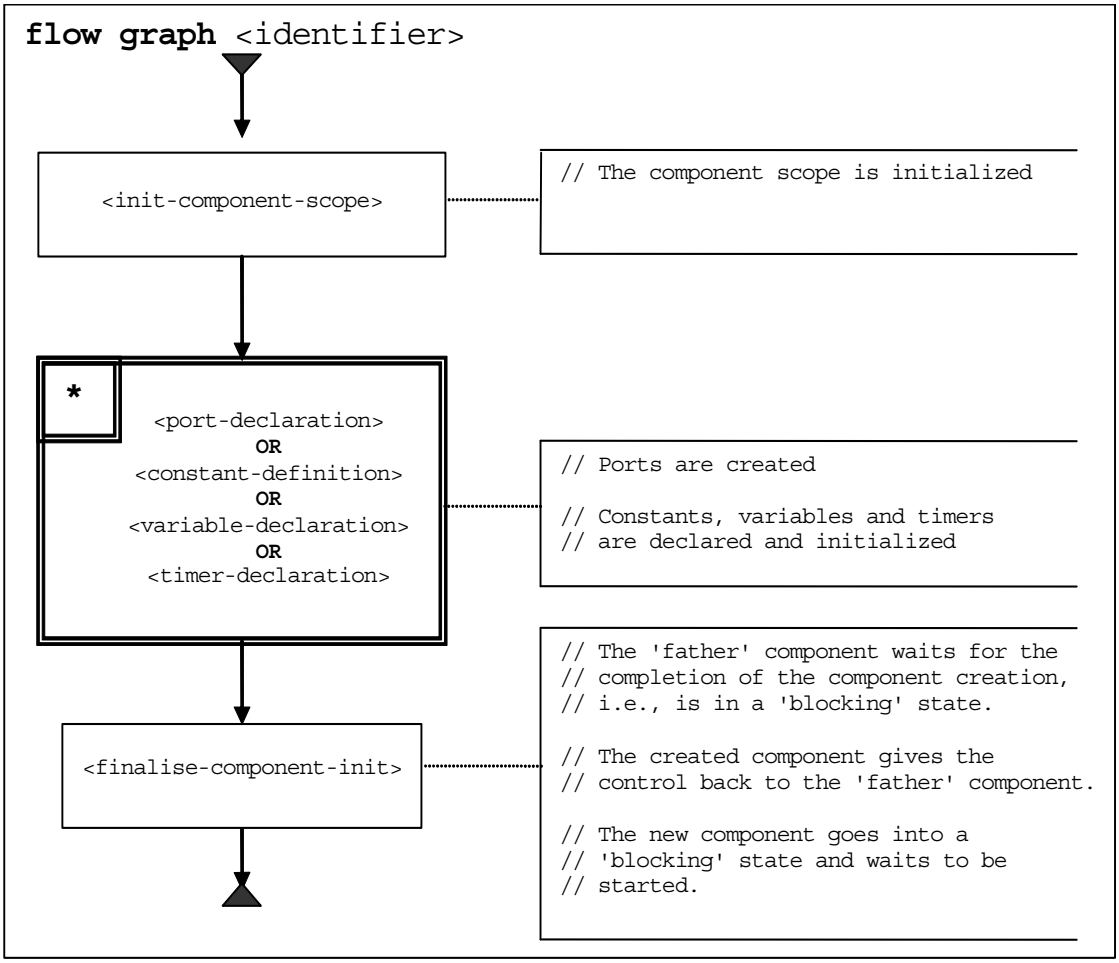
6.2.8 تمثيل تعريف نمط المكونة في شكل مخطط انسيابي

يكون ترتيب تعريف نمط المكونة TTCN-3 كما يلي:

```
type component <identifier> <port-constant-variable-timer-declarations>
```

ويكون نفاذ الدلالة إلى مخططات النفاذ التي تمثل أنماط المكونة من خلال أسماء هذه الأنماط.

ويرد في الشكل 22 تمثيل تعريف نمط المكونة في شكل مخطط انسيابي. ويحيل اسم المخطط <identifier> إلى اسم نمط المكونة الممثل.



الشكل Z.143/22 – تمثيل تعريف نمط المكونة في شكل مخطط انسيابي

7.2.8 استخراج عقدة ابتدائية من مخطط انسيابي

لاستخراج مرجع عقدة ابتدائية من مخطط انسيابي، يتطلب الأمر الوظيفة التالية:

The GET-FLOW-GRAPH function: GET-FLOW-GRAPH (flow-graph-identifier)

تعيد الوظيفة مرجعاً إلى العقدة الابتدائية في المخطط الانسيابي يحمل اسم *flow-graph-identifier*. ويشير هذا المعرف إلى اسم وحدة التحكم وإلى أسماء حالات الاختبار وإلى أسماء الوظائف وتعاريف نمط المكونة.

3.8 تعاريف الحالات في وحدات TTCN-3

لدى تفسير المخططات الانسيابية التي تمثل سلوك TTCN-3، تكون *حالات الوحدات* موضع مناولة. وحالة وحدة ما هي حالة مركبة تتكون من عدة حالات فرعية تصف حالات مكونات الاختبار والمنافذ. وتصف هذه الفقرة حالات الوحدات وحالات المكونات وحالات المنافذ. بالإضافة إلى ذلك، تحدد الوظائف التي تسمح باستخراج معلومات الحالات ومناولتها.

1.3.8 حالة الوحدة

كما يبدو في الشكل 23، تكون حالة الوحدة مركبة من *ALL-ENTITY-STATES* و*ALL-PORT-STATES* و*MTC* و*TC-VERDICT* و*DONE* و*SNAP-ACTIVE*. وتصف *ALL-ENTITY-STATES* حالة التحكم في الوحدة ولدى إجراء اختبار حالة تصف حالات مكونات الاختبار التمثيلية. ولا تنطبق *ALL-PORT-STATES* و*MTC* و*TC-VERDICT* إلا عند إجراء اختبار حالة. وتصف *ALL-PORT-STATES* حالات مختلف المنافذ. وتوفر المكونة *MTC* مرجعاً إلى مكونة الاختبار الرئيسية، ويتضمن *TC-VERDICT* نتيجة الاختبار الإجمالي الفعلي لاختبار حالة، و*DONE* هي عبارة عن قائمة بكل مكونات الاختبار المتوقفة خلال إجراء اختبار الحالة، وتستخدم *SNAP-ACTIVE* باعتبارها جزءاً من اللقطة الآنية للمكونة MTC. وتتضمن *SNAP-ACTIVE* عدد مكونات الاختبار النشيطة عندما تلتقط المكونة MTC لقطة آنية. وهي تستخدم لتقييم العمليات **all component.done** و**all component.running**.

الملاحظة 1 – إن عدد تحديثات الحالة TC-VERDICT مماثل لعدد مكونات الاختبار المنتهية.

يعامل سلوك التحكم في الوحدة (*M-CONTROL* في الشكل 23) بمثابة مكونة اختبار عادية وحالته هي العنصر الأول في *ALL-ENTITY-STATES* في حالة وحدة ما.

ALL-ENTITY-STATES					ALL-PORT-STATES				MTC	TC-VERDICT	DONE	SNAP-ACTIVE
	M-CONTROL	ES_1	...	ES_n		P_1	...	P_n				

الشكل Z.143/23 – تركيب حالة وحدة

الملاحظة 2 – يمكن اعتبار حالات المنافذ بأنها جزء من حالات الكيان. وبفضل **connect** و**map**، تصبح المنافذ مرئية من المكونات الأخرى وبالتالي تعالج هذه الدلالة التشغيلية المنافذ على المستوى الأعلى لحالة وحدة ما.

1.1.3.8 النفاذ إلى حالة الوحدة

تشكل الحالات *MTC* و*TC-VERDICT* و*SNAP-ACTIVE* أجزاء من حالة الوحدة ويتم معالجتها باعتبارها متغيرات إجمالية، أي يمكن استخدام الكلمتين الرئيسيتين *MTC* و*TC-VERDICT* لاستخراج وتغيير قيم حالة الوحدة المقابلة.

الملاحظة 1 – لا توجد إلا حالة وحدة واحدة لدى تفسير الوحدة TTCN-3. وبالتالي، تعتبر الكلمتان الرئيسيتان MTC وTC-VERDICT معرفين فريدين إجمالاً في عملية التقييم.

لدى معالجة القوائم *ALL-ENTITY-STATES* و*ALL-PORT-STATES* و*DONE*. يمكن استخدام عمليات *add* و*append* و*delete* و*member* و*first* و*length* و*next* و*random* و*change* المستخدمة في القوائم. ولهذه العمليات المعاني التالية:

- myList.add(item) تضيف *item* كأول عنصر في القائمة *myList*؛
- *myList.append(item)* تضيف *item* كآخر عنصر في القائمة *myList*؛
- *myList.delete(item)* تحذف *item* من القائمة *myList*؛
- *myList.member(item)* تعيد **true** إذا كان `item` عنصراً في القائمة *myList* و**false** خلاف ذلك؛
- *myList.first()* تعيد أول عنصر من *myList*؛
- *myList.length()* تعيد طول *myList*؛
- *myList.next(item)* تعيد العنصر الذي يلي *item* في *myList* أو **NULL** إذا كان *item* آخر عنصر في *myList*؛
- *MyList.random(<condition>)* يعيد عشوائياً من *myList* عنصراً يلبي الشرط *Boolean* <*condition*> أو **NULL** إذا لم يف أي عنصر في *myList* بالشرط <*condition*>؛
- *MyList.change(<operation>)* تسمح بتطبيق <*operation*> على جميع عناصر *myList*.

الملاحظة 2 – ليس من الشائع تطبيق العمليتين random وchange على القوائم. وهما واردتان لتفسير معنى الكلمتين الرئيسيتين **all** و**any** في العمليات TTCN-3.

2.3.8 حالات الكيانات

تُستخدم حالات الكيانات لوصف الحالات القائمة للتحكم في الوحدة ومكونات الاختبار. وفي حالة الوحدة، يتم معالجة حالات الكيان في القائمة ALL-ENTITY-STATES ويرد في الشكل 24 تركيب حالة الكيان.

<identifier>	STATUS	CONTROL-STACK	DEFAULT-LIST	DEFAULT-POINTER	VALUE-STACK	E-VERDICT	TIMER-GUARD	DATA-STATE	TIMER-STATE	SNAP-DONE

الشكل Z.143/24 – تركيب حالة الكيان

ويكون <*identifier*> المعرف الوحيد لكيان ما في نظام الاختبار، أي التحكم في الوحدة في مكونة الاختبار. يتم توليد هذه المعرفات الفريدة ضمنياً للتحكم في الوحدة والمكونة **mtc** ونظام الاختبار **system** عندما تبدأ التنفيذ وحدة ما أو تنفذ حالة اختبار بواسطة البيان **execute**. وهي تساعد في تعرّف ومعالجة كيانات نظام الاختبار، مثال ذلك في حالة العمليات **send** مع to أو العمليات **receive** مع **from**.

وتصف الحالة *STATUS* ما إذا كان التحكم في الوحدة أو مكون الاختبار **ACTIVE** أو **SNAPSHOT** أو **REPEAT** أو **BLOCKED**. ويكون التحكم في الوحدة معطلاً أثناء إجراء اختبار حالة. ومن الممكن تعطيل مكونات الاختبار أثناء توليد مكونات اختبار أخرى، أي أثناء تنفيذ العملية **create**. وتشير الحالة **SNAPSHOT** إلى أن المكونة نشيطة ولكنها في مرحلة تقييم لقطة آنية. وتشير الحالة **REPEAT** إلى أن المكونة نشيطة وأنها في البيان alt الذي ينبغي إعادة تقييمه بحكم البيان **repeat**.

CONTROL-STACK هي عبارة عن مكدّس مراجع لعقد مخططات الانسياب. والعنصر الأعلى في مكدّس *CONTROL-STACK* هو العقدة التالية في مخطط الانسياب التي يتعين تفسيرها. والمكدّس ضروري لنمذجة نداءات الوظيفة بشكل مناسب.

DEFAULT-LIST هي قائمة من حالات التغيب المنشطة، أي قائمة من المؤشرات التي تشير إلى العقد الابتدائية في حالات التغيب المنشطة. والقائمة مرتبة بعكس تعاقب التنشيط، فحالة التغيب التي نشطت أولاً تكون العنصر الأخير في القائمة.

وخلال تطبيق آلية التغيب، تشير الحالة *DEFAULT-POINTER* إلى حالة التغيب التالية الواجب تقييمها في حال فشل انتهاء حالة التغيب الفعلية.

و*VALUE-STACK* مكدّس من قيم كافة الأنماط الممكنة تسمح بتخزين وسيط للنتائج النهائية أو الوسيطة للعمليات والوظائف والبيانات. مثال ذلك أن نتيجة تقييم تعبير ما أو نتيجة عملية **mtc** سوف تدرج في *VALUE-STACK*. وإلى جانب قيم جميع أنماط البيانات المعروفة في وحدة ما، يتم تحديد القيمة الخاصة **MARK** باعتبارها جزءاً من أبجدية المكدّس. وعند الخروج من وحدة نطاق تستعمل القيمة **MARK** لتنظيف *VALUE-STACK*.

وتتضمن *E-VERDICT* النتيجة المحلية الفعلية لمكونة اختبار ما. وتهمل إذا كانت حالة الكيان تمثل التحكم في الوحدة.

ويمثل *TIME-GUARD* المؤقت الخاص المطلوب لحماية إجراء اختبارات الحالات ومدة عمليات النداء. وهو منمذج في شكل صلة توقيت (انظر الفقرة 4.2.3.8 والشكل 28).

وتُعتبر *DATA-STATE* بمثابة قائمة تضم قوائم وصلات متغيرات. وتعكس هيكلية قائمة القوائم وحدات النطاق المُدرجة بحكم نداءات الوظائف المدرجة. وتصف كل قائمة في قائمة قوائم الوصلات المتغيرات المعروفة وقيمها في وحدة نطاق معينة. ويكون دخول وحدة نطاق والخروج منها بمثابة إضافة أو حذف قائمة وصلات متغيرات في *DATA-STATE*. ويرد في الفقرة 2.2.3.8 وصف للجزء *DATA-STATE* في حالة كيان ما.

وتُعتبر *TIMER-STATE* قائمة قوائم وصلات التوقيت. وتعكس هيكلية قائمة القوائم وحدات النطاق المدرجة بحكم نداءات الوظائف المدرجة. وتصف كل قائمة في قائمة قوائم وصلات التوقيت المؤقتات المعروفة ووضعها في وحدة نطاق معينة. ويكون دخول وحدة نطاق والخروج منها بمثابة إضافة أو حذف قائمة حالات التوقيت في *TIMER-STATE*. ويرد في الفقرة 4.2.3.8 وصف للجزء *TIMER-STATE* في حالة كيان ما.

وتتناول *SNAP-DONE* الدلالة الآنية لمكونات الاختبار. وعند أخذ لقطة آنية، تخصص نسخة من قائمة *DONE* في حالة الوحدة إلى *SNAP-DONE*. أي تكون *SNAP-DONE* قائمة معرفات المكونات المتوقفة.

1.2.3.8 النفاذ إلى حالات الكيان

<*identifier*> هو المعرف الوحيد لحالة الكيان والذي يمكن أن يستعمل من أجل النفاذ إلى المكون الممثل بحالة الكيان وإلى مختلف أجزاء حالة الكيان.

وتعامل الأجزاء STATUS وDEFAULT-POINTER وE-VERDICT وTIMER-GUARD في حالة كيان على أنها متغيرات مرئية إجمالاً، أي يمكن استخراج أو تعديل قيم STATUS وDEFAULT-POINTER وE-VERDICT بواسطة عامل "نقطة"، مثل myEntity.STATUS أو myEntity.DEFAULT-POINTER أو myEntity.E-VERDICT حيث تشير myEntity إلى حالة كيان.

ملاحظة – فيما يلي، نفترض أن من الممكن استخدام عامل "النقطة" بواسطة إحالات ومعرفات فريدة. ففي *myEntity.STATUS*، مثلاً قد تكون *myEntityState* مؤشراً باتجاه حالة كيان أو قد تكون قيمة المجال <*identifier*>.

ويمكن معالجة الأجزاء CONTROL-STACK وDEFAULT-LIST وVALUE-STACK لحالة كيان myEntity باستخدام عامل "النقطة" myEntity.CONTROL-STACK أو myEntity.DEFAULT-LIST أو myEntity.VALUE-STACK.

ويمكن النفاذ إلى كل من CONTROL-STACK وVALUE-STACK والتصرف فيهما بواسطة العمليات المطبقة على المكدّسات push وpop وtop وclear وclear-until. وتحمل هذه العمليات المعاني التالية:

- myStack.push(item) يضع العنصر item في myStack؛
- myStack.pop() يحذف العنصر الأعلى في myStack؛
- myStack.top() يعيد العنصر الأعلى في myStack أو **NULL** إذا كان myStack فارغاً؛
- myStack.clear() ينظف myStack، أي يحذف جميع العناصر من myStack؛
- myStack.clear-until(item) يحذف العناصر من myStack حتى العنصر item أو حتى يصبح myStack فارغاً.

ومن الممكن النفاذ إلى *DEFAULT-LIST* والتصرف فيه بواسطة العمليات التي تطبق على القوائم *add* و*append* و*delete* و*member* و*first* و*length* و*next* و*random* و*change*. ويرد معنى هذه العمليات في الفقرة 1.1.3.8.

ولاستحداث حالة كيان جديدة، يفترض أن تكون الوظيفة *NEW-ENTITY* متاحة:

- إذ إن NEW-ENTITY (entityIdentifier, flow-graph-node-reference)؛

تستحدث حالة كيان جديدة وتعيد مرجعها. ويكون لمكونات حالة الكيان الجديدة القيم التالية:

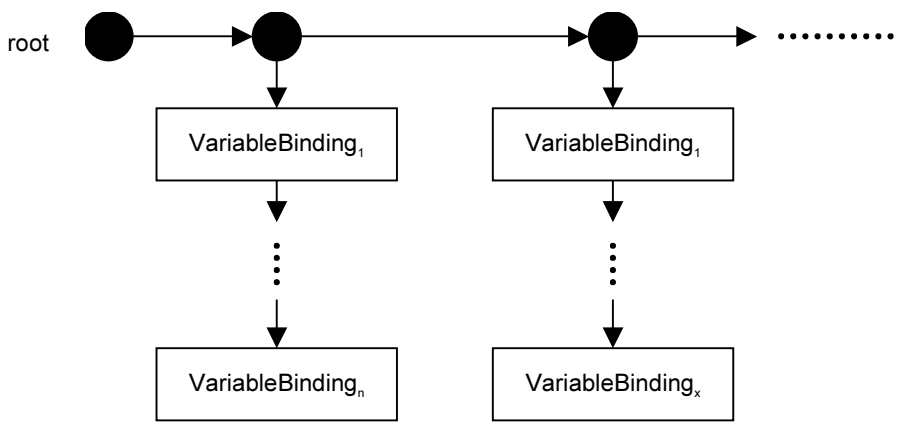
- <identifier> يتخذ القيمة entityIdentifier ويكون معرفاً وحيداً إجمالياً؛
- *STATUS* يكون **ACTIVE**؛
- flow-graph-node-reference العنصر (الأعلى) الوحيد في CONTROL-STACK؛
- *DEFAULT-LIST* قائمة فارغة؛
- *DEFAULT-POINTER* يتخذ القيمة **NULL**؛
- *VALUE-STACK* مكدس فارغ؛
- *E-VERDICT* يتخذ القيمة **none**؛
- *TIMER-GUARD* وصلة مؤقت جديدة (انظر الفقرة 4.2.3.8) تتخذ اسم GUARD ووضع **IDLE** دون تحديد أي مدة بالتغيب؛
- *DATA-STATE* قائمة فارغة؛
- *TIMER-STATE* قائمة فارغة؛
- *SNAP-DONE* قائمة فارغة.

لدى اجتياز مخطط الانسياب، غالباً ما تغير *CONTROL-STACK* قيمها بالطريقة نفسها: يحذف العنصر الأعلى من *CONTROL-STACK* وتضاف العقدة التالية للعقدة المحذوفة في *CONTROL-STACK*. وتتخلص هذه السلسلة من عمليات المكدسات في الوظيفة *NEXT-CONTROL* كما يلي:

```
myEntity.NEXT-CONTROL(myBool) {
      successorNode := myEntity.CONTROL-STACK.NEXT(myBool).top();
      myEntity.CONTROL-STACK.pop();
      myEntity.CONTROL-STACK.push(successorNode);
}
```

2.2.3.8 حالة البيانات ووصلات المتغيرات

إن حالة البيانات *DATA-STATE* لحالة كيان ما هي قائمة قوائم وصلات متغيرات، كما يظهر في الشكل 25 وتحدد كل قائمة من وصلات المتغيرات هذه الوصلات في وحدة نطاق محددة. وتعني إضافة قائمة جديدة من وصلات المتغيرات الدخول إلى وحدة نطاق جديدة، عند نداء وظيفة مثلاً. ويعني حذف قائمة من وصلات المتغيرات الخروج من وحدة نطاق، عندما تنفذ وظيفة ما البيان **return** مثلاً.



الشكل Z.143/25 – هيكلية الجزء *DATA-STATE* في حالة كيان ما

يصف الشكل 26 هيكلية وصلة متغيرات. ولكل متغير اسم وموضع <*location*> وقيمة *VALUE*. ويحدد *VAR NAME* متغير ما في وحدة نطاق. ويكون <*location*> معرف وحيد للموضع الذي تخزن فيه قيمة المتغير. أما الجزء *VALUE* لوصلة المتغير فهي تصف القيمة الفعلية للمتغير.

ملاحظة – عند الإعلان عن متغير ما ينبغي توفير معرف وحيد للموقع بشكل أوتوماتي.

VAR-NAME	<location>	VALUE

الشكل Z.143/26 – هيكلية وصلة متغيرة

الغرض من التمييز بين اسم المتغير وموقعه هو نمذجة نداءات الوظائف وإجراء اختبارات الحالات مع تحديد المعلمات إما بالقيمة أو بالإحالة بشكل مناسب:

أ) تعالَج المعلمة المرسلة بحسب القيمة على أنها إعلان عن متغير جديد، أي تضاف وصلة متغير جديد إلى قائمة وصلات المتغيرات في وحدة نطاق الوظيفة المطلوبة أو اختبار الحالة المنفذ. وتستعمل وصلة المتغير الجديد اسم المعلمة الشكلي *VAR-NAME* ويحصل على موضع جديد ويأخذ القيمة المرسلة إلى الوظيفة أو اختبار الحالة.

ب) تؤدي المعلمة المرسلة بالإحالة أيضاً إلى وصلة متغير جديد في وحدة نطاق الوظيفة المطلوبة أو اختبار الحالة المنفذ. وتستعمل وصلة المتغير الجديد أيضاً اسم المعلمة الشكلي VAR-NAME ولكنها لا تحصل على أي موضع جديد أو أي قيمة جديدة. وتحصل وصلة المتغير الجديد على نسخة من الموضع <location> والقيمة VALUE للمتغير المرسل بالإحالة.

عند تحديث قيمة متغير ما، عندما تنسب إلى متغير مثلاً، فإن اسم المتغير يستعمل لتحديد موضع ويتم في الوقت ذاته تحديث جميع وصلات المتغير في الموضع نفسه. وبالتالي، عند الخروج من وحدة النطاق، من الممكن حذف قائمة المتغيرات التي تنتمي إلى وحدة النطاق هذه من دون أي تحديث آخر. وبحكم إجراء التحديث تأخذ المتغيرات المرسلة بالإحالة قيمتها الصحيحة بشكل أوتوماتي.

3.2.3.8 النفاذ إلى حالات البيانات

من الممكن استخراج قيمة المتغير باستخدام عامل "النقطة" myEntity.myVar.VALUE حيث تحيل myEntity إلى حالة كيان ويكون myVar اسم المتغير.

بالنسبة إلى معالجة المتغيرات ونطاقات المتغير، يفترض أن تكون الوظائف التالية محددة:

أ) الوظيفة *VAR-SET*: *myEntity.VAR-SET (myVar, myValue)*

تضع الجزء *VALUE* من المتغير *myVar* في النطاق الفعلي لمتغير *myEntity* عند *myVal*. وكذلك يوضع الجزء *VALUE* لجميع المتغيرات في الموقع نفسه للمتغير *myVar* عند *myVal*.

ب) الوظيفة *INIT-VAR*: *myEntity.INIT-VAR (myVar, myVal)*

تستحدث وصلة متغير جديدة لمتغير *myVar* له القيمة الابتدائية *myVal* في وحدة النطاق الفعلية للكيان *myEntity*. واستخدام الكلمة الأساسية **NONE** للقيمة *myVal* يعني استحداث متغير له قيمة ابتدائية غير محددة. وتستحدث قيمة جديدة ووحيدة للموضع <*location*> أوتوماتياً.

ج) الوظيفة *GET-VAR-LOC*: *myEntity.GET-VAR-LOC (myVar)*

تستخرج موضع المتغير *myVar* الذي تملكه القيمة *myEntity*.

د) الوظيفة *INIT-VAR-LOC*: *myEntity.INIT-VAR-LOC (myVar,myLoc)*

تستحدث وصلة متغير جديدة للمتغير *myVar* مع موضع *myLoc* في وحدة النطاق الفعلية للقيمة *myEntity*. ويتم تدميث المتغير بقيمة متغير آخر في الموضع *myLoc*.

ملاحظة – إن المتغيرات في نفس الموضع تنتج عن تحديد المعلمات بالإحالة. وبالنظر إلى معالجة المعلمات بالإحالة كما جاء في الفقرة *2.2.3.8*، فإن جميع المتغيرات في نفس الموضع يكون لها قيم مماثلة خلال فترة حياتها.

هـ) الوظيفة *INIT-VAR-SCOPE*: *myEntity.INIT-VAR-SCOPE ()*

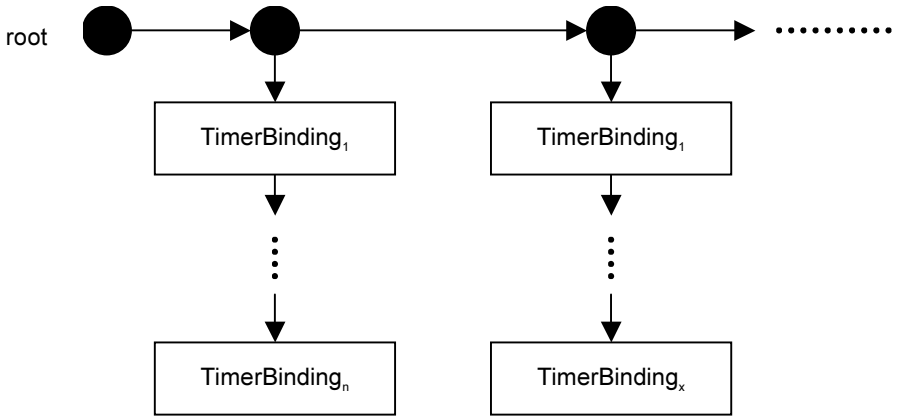
تدمث نطاق جديد من المتغير في حالة بيانات الكيان *myEntityK*، أي تضاف قائمة فارغة باعتبارها القائمة الأولى إلى قائمة قوائم وصلات المتغيرات.

و) الوظيفة *DEL-VAR-SCOPE*: *myEntity.DEL-VAR-SCOPE ()*

تحذف نطاق متغير من حالة بيانات *myEntity*، أي يتم حذف القائمة الأولى الواردة في قائمة قوائم وصلات المتغيرات.

4.2.3.8 حالة المؤقت ووصلة المؤقت

كما يبدو في الشكلين 27 و25، ثمة تشابه بين حالة المؤقت TIMER-STATE وحالة البيانات DATA-STATE في حالة كيان ما. فكل منهما قائمة قوائم وصلات وكل قائمة وصلات تحدد الوصلات الصالحة في نطاق معين. وإضافة قائمة جديدة تقابل الدخول في وحدة نطاق جديدة وحذف قائمة وصلات يقابل الخروج من وحدة نطاق.



الشكل Z.143/27 – هيكلية الجزء *TIMER-STATE* في حالة كيان ما

يبين الشكل 28 هيكلية وصلة المؤقت. ومعنى الاسم *TIMER-NAME* والموضع <*location*> مماثل لمعنى الاسم *VAR-NAME* والموضع <*location*> لوصلة متغير (الشكل 26).

TIMER-NAME	<location>	STATUS	DEF-DURATION	ACT-DURATION	TIME-LEFT	SNAP-VALUE	SNAP-STATUS

الشكل Z.143/28 – هيكلية وصلة مؤقت

تبين الحالة *STATUS* ما إذا كان المؤقت نشطاً أو خاملاً أو منتهياً. والقيم المقابلة في الحالة هي **IDLE** أو **RUNNING** أو **TIMEOUT**. وتصف *DEF-DURATION* المدة بالتغيب في المؤقت. وتتضمن *ACT-DURATION* المدة الفعلية التي تم إطلاق المؤقت من خلالها. وتصف *TIME-LEFT* المدة الفعلية المتبقية للمؤقت قبل توقفه.

ملاحظة – تكون *DEF-DURATION* غير محددة إذا كان المؤقت ليس له مدة بالتغيب. وتوضع كل من *ACT-DURATION* و*TIME-LEFT* بقيمة 0,0 إذا توقف المؤقت أو انتهت مدته. وإذا تم إطلاق المؤقت دون تحديد المدة، تنسخ قيمة *DEF-DURATION* إلى *ACT-DURATION*. ويحدث خطأ دينامي إذا أطلق مؤقت دون مدة محددة.

والحالتان *SNAP-VALUE* و*SNAP-STATUS* ضروريتان لدعم الدلالة الآنية للترميز TTCN-3. وعند التقاط اللقطة الآنية تتلقى *SNAP-VALUE* القيمة الفعلية للحالتين *ACT-DURATION* و*TIME-LEFT*. وتتلقى *SNAP-STATUS* القيمة نفسها للحالة *STATUS*. ولا يقوم تقييم اللقطة الآنية إلا على القيمتين *SNAP-VALUE* و*SNAP-STATUS*.

ولا يمكن إرسال المؤقت إلى الوظائف إلا بالإحالة، أي أن الآلية مماثلة لآلية المتغيرات الموصوفة في الفقرة 2.2.3.8. وهذا يعني أن وصلة مؤقت جديدة تنشأ (تحمل اسم المعلمة الشكلي) وأنها تتلقى نسخاً من <*location*> و*STATUS* و*DEF-DURATION* و*ACT-DURATION* و*TIME-LEFT* و*SNAP-VALUE* و*SNAP-STATUS* من المؤقت المرسل بالإحالة. وعند تحديث المؤقت، تحدث في الوقت نفسه جميع وصلات المؤقت التي لها نفس قيمة الموضع <*location*>.

5.2.3.8 النفاذ إلى حالات المؤقت

يمكن استخراج قيم STATUS وDEF-DURATION وACT-DURATION وTIME-LEFT وSNAP-VALUE وSNAP-STATUS التابعة للمؤقت myTimer باستخدام عامل "النقطة":

- myEntity.myTimer.STATUS؛
- myEntity.myTimer.DEF-DURATION؛
- myEntity.myTimer.ACT-DURATION؛
- myEntity.myTimer.TIME-LEFT؛

- myEntity.myTimer.SNAP-VALUE؛
- myEntity.myTimer.SNAP-STATUS.

إن العنصر *myEntity* الوارد في العلاقة المنقطة يشير إلى حالة الكيان التي تمثل حالة مكونة الاختبار أو التحكم في الوحدة الذي يحتوي المؤقت *myTimer*.

ولتغيير قيم *STATUS* و*DEF-DURATION* و*ACT-DURATION* و*TIME-LEFT* و*SNAP-VALUE* و*SNAP-STATUS* للمؤقت *timer-name*، ينبغي استخدام العملية النمطية *TIMER-SET*، مثال ذلك أن:

- *myEntity.TIMER-SET(myTimer, STATUS, myVal)*

تضع الحالة *STATUS* للمؤقت *myTimer* في النطاق الفعلي للحالة *myEntity* عند القيمة *myVal*. وعلاوة على ذلك، توضع الحالة *STATUS* لجميع المؤقتات في نفس موضع المؤقت *myTimer* عند القيمة *myVal*. ويمكن أيضاً استخدام الوظيفة *TIMER-SET* لتغيير قيم الحالات *DEF-DURATION* و*ACT-DURATION* و*TIME-LEFT* و*SNAP-VALUE* و*SNAP-STATUS*.

ولمعالجة المؤقتات ونطاق المؤقتات واللقطات الآنية يجب تحديد الوظائف التالية:

أ) الوظيفة *INIT-TIMER*: *myEntity.INIT-TIMER (myTimer, myDuration)*

تستحدث وصلة مؤقت جديدة للمؤقت *myTimer* تكون مدتها بالتغيب *myDuration* في النطاق الفعلي للكيان *myEntity*. واستخدام الكلمة الأساسية *NONE* للحالة *myDuration* يعني استحداث مؤقّت ليس له مدة بالتغيب.

ب) الوظيفة *GET-TIMER-LOC*: *myEntity.GET-TIMER-LOC (myTimer)*

تستخرج موضع المؤقت *myTimer* الذي تحتويه *myEntity*.

ج) الوظيفة *INIT-TIMER-LOC*: *myEntity.INIT-TIMER-LOC (myTimer, myLocation)*

تستحدث وصلة مؤقت جديدة للمؤقت *myTimer* يكون موضعه *myLocation* في وحدة النطاق الفعلية للحالة *myEntity*. ويتخذ المؤقت القيم *STATUS* و*DEF-DURATION* و*ACT-DURATION* و*TIME-LEFT* لدى مؤقت آخر في الموضع *<location>*.

ملاحظة – تكون المؤقتات في نفس الموضع نتيجة تحديد معلمات بالإحالة. ونظراً لمعالجة المعلمات المرجعية للمؤقت الموصوفة في الفقرة *3.2.3.8*، فإن جميع المؤقتات في نفس الموضع تكون لها نفس القيم *STATUS* و*DEF-DURATION* و*ACT-DURATION* و*TIME-LEFT* خلال مدة حياتها.

د) الوظيفة *INIT-TIMER-SCOPE*: *myEntity.INIT-TIMER-SCOPE ()*

تدمث نطاقاً جديداً للمؤقت في حالة المؤقت للكيان *myEntity*، أي تضاف قائمة فارغة باعتبارها القائمة الأولى في قائمة قوائم وصلات المؤقت.

هـ) الوظيفة *DEL-TIMER-SCOPE*: *myEntity.DEL-TIMER-SCOPE ()*

تحذف نطاق المؤقت لحالة مؤقت الكيان *myEntity*، أي تحذف القائمة الأولى في قائمة قوائم وصلات المؤقت.

و) الوظيفة *SNAP-TIMER*: *myEntity.SNAP-TIMER ()*

تقوم بتحديث كل من *SNAP-VALUE* و*SNAP-STATUS* في كافة المؤقتات التي تحتويها *myEntity* أي:

```
myEntity.SNAP-TIMERS () {
 for all myTimer in TIMER-STATE {
     myEntity.myTimer.SNAP-VALUE := myEntity.myTimer.ACT-DURATION -
     myEntity.myTimer.TIME-LEFT;
     myEntity.myTimer.SNAP-STATUS := myEntity.myTimer.STATUS;
 }
```

3.3.8 حالات المنافذ

تستخدم حالات المنافذ لوصف الحالات الفعلية للمنافذ. وتعالج حالات المنافذ، ضمن حالة وحدة، في القائمة *ALL-PORT-STATES* (انظر الشكل *23*). ويبين الشكل *29* هيكلية حالة منفذ. ويشير *PORT-NAME* إلى اسم المنفذ الذي يستخدم لتحديد المنفذ بواسطة مكون الاختبار *OWNER* الذي يحتويه. ويتضمن *STATUS* الحالة الفعلية للمنفذ. ويكون المنفذ إما **STARTED** أو **STOPPED**.

ملاحظة – يكون التعرف على منفذ في نظام اختبار فريداً من جانب مكون الاختبار الذي يحتويه <owner> واسم المنفذ <port-name> المحلي نسبة إلى <owner>.

وتحتفظ القائمة *CONNECTIONS-LIST* في حالة المنفذ بسجل الوصلات بين مختلف منافذ نظام الاختبار. وهذه الآلية موضحة في الفقرة 1.3.3.8.

ويختزن الجزء *VALUE-QUEUE* في حالة المنفَذ الرسائل والنداءات والإجابات والاستثناءات المتلقاة عند هذا المنفذ والتي لم تستنفد بعد.

ويتناول الجزء *SNAP-VALUE* آلية اللقطة الآنية *TTCN-3*. وعند التقاط اللقطة الآنية ينسخ العنصر الأول في الحالة *VALUE-QUEUE* إلى *SNAP-VALUE*. وتتخذ *SNAP-VALUE* القيمة **NULL** إذا كانت *VALUE-QUEUE* فارغة أو إذا كانت *STATUS* في حالة **STOPPED**.

PORT-NAME	OWNER	STATUS	CONNECTIONS-LIST	VALUE-QUEUE	SNAP-VALUE

الشكل Z.143/29 – هيكلية حالة منفذ

1.3.3.8 معالجة التوصيلات بين المنافذ

يتحقق الاتصال بين مكونتي اختبار بتوصيل منفذين من منافذهما بواسطة العملية **connect**. وبالتالي، يمكن لمكونة أن تستخدم بعد ذلك اسم المنفذ المحلي الخاص بها للتعامل مع الطابور البعيد. وكما يظهر في الشكل 30، فإن *التوصيل* ممثل في حالتي الطابورين الموصولين بزوج من *REMOTE-ENTITY* و*REMOTE-PORT-NAME*. ويحتوي *REMOTE-ENTITY* على المعرف الوحيد لمكون الاختبار الذي يملك المنفذ البعيد. ويشير *REMOTE-PORT-NAME* إلى الاسم المحلي الذي يستخدمه العنصر *REMOTE-ENTITY* لمعالجة الطابور. ويدعم الترميز TTCN-3 التوصيلات من منفذ إلى عدة منافذ وبالتالي تنظَم جميع توصيلات أي منفذ في قائمة.

الملاحظة 1 – يتم كذلك معالجة التوصيلات التي تنجزها العمليات **map** في قائمة التوصيلات. العملية **map**: تؤدي **map** (*PTC1*:*MyPort*, **system**.*PCO1*) إلى توصيلة جديدة (**system**, *PCO1*) في حالة المنفَذ للقيمة *MyPort* التي تحتويها *PTC1*. ويكون الطرف البعيد للتوصيلة مع *PC01* داخل النظام SUT. ويقع سلوكه خارج إطار هذه الدلالة.

الملاحظة 2 – تعالج الدلالة التشغيلية الكلمة الأساسية **SYSTEM** بمثابة عنوان رمزي. وتشير توصيلة (**system**, *myPort*) في قائمة توصيلات منفذ ما إلى أن المنفذ يقابله المنفذ *myPort* في السطح البيني لنظام الاختبار.

REMOTE-ENTITY	REMOTE-PORT-NAME

الشكل Z.143/30 – هيكلية التوصيل

2.3.3.8 معالجة حالات المنفذ

من الممكن النفاذ إلى طابور القيم في حالة منفذ والتصرف فيها باستخدام عمليات الطابور المعروفة *enqueue* و*dequeue* و*first* و*clear*. واستخدام وظيفة *GET-PORT* أو *GET-REMOTE* يشير إلى الطابور الذي ينبغي الوصول إليه.

الملاحظة 1 – تتخذ العمليات المطبقة على أي طابور *enqueue* و*dequeue* و*first* و*clear* المعاني التالية:

- *myQueue.enqueue(item)* تضع *item* كآخر عنصر في *myQueue*؛
- *myQueue.dequeue*() تحذف أول عنصر من *myQueue*؛
- *myQueue.first*() تعيد أول عنصر في *myQueue* أو **NULL** إذا كانت *myQueue* فارغة؛
- *myQueue.clear*() تحذف كل العناصر من *myQueue*.

وتكون معالجة حالات المنفذ بواسطة الوظائف التالية:

أ) الوظيفة *NEW-PORT*: *NEW-PORT* (*myEntity, myPort*)

تستحدث منفذاً جديداً وتعيد مرجعه. وتحتوي *myEntity* المنفذ الجديد وهو يحمل اسم *myPort* بالنسبة إلى المنفذ الذي يحدده مكون الاختبار *myEntity* واسم المنفذ myPort. ويكون وضع المنفذ الجديد **STARTED**. وتكون *CONNECTIONS-LIST* و*VALUE-QUEUE* فارغتين. وتكون *SNAP-VALUE* قيمتها **NULL** (أي أن طابور مدخل المنفذ الجديد فارغ).

ب) الوظيفة GET-PORT: GET-PORT (myEntity, myPort)

تعيد مرجعاً إلى المنفذ الذي يحدده مكون الاختبار *myEntity* الذي يحتوي المنفذ واسم المنفذ *myPort*.

ج) الوظيفة *GET-REMOTE-PORT*: *GET-REMOTE-PORT* (*myEntity*، *myPort*، *myRemoteEntity*)

تعيد المرجع إلى المنفذ الذي يحتويه مكون الاختبار *myRemoteEntity* والموصول بمنفذ يحدده *myEntity* و*myPort*. يعاد العنوان الرمزي **SYSTEM** إذا كان المنفذ البعيد يقابله منفذ في السطح البيني لنظام الاختبار.

الملاحظة 2 – تعيد الوظيفة *GET-REMOTE-PORT* حالة **NULL** إذا لم يكن هنالك منفذ بعيد أو إذا كان من الصعب التعرّف عليه صراحة. ويمكن استعمال القيمة الخاصة **NONE** كقيمة للمعلمة *myRemoteEntity* إذا كان الكيان البعيد غير معروف أو غير مطلوب، أي ليس هنالك سوى توصيلة وحيدة بين طرفين لهذا المنفذ.

د) تعالَج الحالة *STATUS* لمنفذ ما بمثابة متغير. ويمكن معالجتها بوصف الحالة *STATUS* بواسطة نداء *GET-PORT*:

GET-PORT(myEntity, myPort).STATUS

هـ) الوظيفة *ADD-CON*: *ADD-CON (myEntity, myPort, myRemoteEntity, myRemotePort)*

تضيف توصيلة *(myRemoteEntity, myRemotePort)* إلى قائمة توصيلات المنفذ *myPort* الذي يحتويه *myEntity*.

و) الوظيفة *DEL-CON*: *DEL-CON (myEntity, myPort, myRemoteEntity, myRemotePort)*

تزيل التوصيلة *(myRemoteEntity, myRemotePort)* من قائمة توصيلات المنفذ *myPort* الذي يحتويه *myEntity*.

ز) الوظيفة *SNAP-PORTS*: *SNAP-PORTS (myEntity)*

تحدّث *SNAP-VALUE* بالنسبة إلى جميع المنافذ التي تحتويها *myEntity*، أي:

```
SNAP-PORTS (myEntity) {
    for all ports p          /* in the module state */ {
        if (p.OWNER == myEntity) {
            if (p.STATUS == STOPPED) {
                p. SNAP-VALUE := NULL;
            }
            else {
                p. SNAP-VALUE := p.first()
            }
        }
    }
}
```

4.3.8 وظائف عامة لمعالجة حالات الوحدة

تفترض الدلالة التشغيلية وجود الوظائف التالية لمعالجة حالات الوحدة.

الملاحظة 1 – لدى تفسير وحدة TTCN-3 ما هنالك حالة واحدة من حالات الوحدة. ويفترض أن تكون مكونات حالة الوحدة مخزنة في متغيرات إجمالية وليس في شكل معقد للبيانات. وعليه من المفترض أن تنطبق الوظائف التالية على متغيرات إجمالية وأنها لا تتناول حالة وحدة محددة.

أ) الوظيفة *DEL-ENTITY*: *DEL-ENTITY*(*myEntity*)

تحذف كياناً له معرف وحيد *myEntity*. ويشمل الحذف:

- حذف حالة الكيان في *myEntity*؛
- حذف كل المنافذ التي تحتويها *myEntity*؛
- حذف كل التوصيلات التي تدخل فيها *myEntity*.

ب) الوظيفة *UPDATE-REMOTE-REFERENCES*:

UPDATE-REMOTE-REFERENCES (*source, target*)

تحدّث المتغيرات والمؤقتات التي لها نفس الموضع في كلا الكيانين. والقيم المستخدمة في عملية التحديث هي قيم المتغيرات والمؤقتات التي يحتويها *source*.

الملاحظة 2 – تُستخدم الوظيفة *UPDATE-REMOTE-REFERENCES* لدى إنهاء اختبارات الحالة. وهي تمكّن من تحديث متغيرات التحكم في الوحدة التي ترسل في شكل معلمات مرجعية إلى اختبارات الحالة.

4.8 الرسائل ونداءات الإجراء والإجابات والاستثناءات

إن تبادل المعلومات بين مكونات الاختبار من جهة وبين مكونات الاختبار والنظام قيد الاختبار SUT من جهة أخرى يقوم على *الرسائل ونداءات الإجراء والإجابات على نداءات الإجراء والاستثناءات* التي ينبغي إنشاؤها وتشفيرها وفك تشفيرها بهدف الاتصال. والتشفير الملموس، أي تحويل أنماط البيانات TTCN-3 إلى بتات وأثمونات، وفك التشفير، أي تحويل البتات والأثمونات إلى أنماط البيانات TTCN-3، يقعان خارج نطاق الدلالة التشغيلية. وفي هذه التوصية، تعالج *الرسائل ونداءات الإجراء والإجابات على نداء الإجراء والاستثناءات* على المستوى المفهومي.

1.4.8 الرسائل

ترتبط الرسائل بالاتصال القائم على الرسائل. ويمكن تبادل قيم جميع أنواع البيانات (المحددة مسبقاً أو المحددة من جانب المستعمل) بين الكيانات التي تقوم بالاتصال. وكما يبدو في الشكل 31، تعالج الدلالة التشغيلية الرسالة كغرض له بنية يتكون من ثلاثة أجزاء: *sender* و*type* و*value*. ويحدد الجزء *sender* الكيان مرسل الرسالة ويحدد الجزء *type* نمط الرسالة ويحدد الجزء *value* قيمة الرسالة.

sender	type	value

الشكل Z.143/31 – بنية الرسالة

ملاحظة – لا تقدم الدلالة التشغيلية إلا نموذجاً لمفاهيم الترميز TTCN-3. ومعرفة ما إذا كان يجب إرسال و/أو استقبال المعلومات *sender* وكيف يكون ذلك تتوقف على تنفيذ نظام الاختبار، ففي بعض الحالات مثلاً قد تكون المعلومات المتعلقة بالمرسل في القسم *value* من الرسالة، وبالتالي لا تشكل جزءاً منفصلاً من بنية الرسالة.

2.4.8 نداءات الإجراء والإجابات

ترتبط نداءات الإجراء والإجابات على الإجراءات بالاتصالات القائمة على الإجراء. وهي محددة كقيم في سجل تمثل مكوناته المعلمات. وتعالج الدلالة التشغيلية كذلك نداءات الإجراء والإجابات على الإجراءات باعتبارها قيماً في أنماط ذات بنية. وترد في الشكلين 32 و33 بنية نداء الإجراء وبنية الإجابة.

sender	procedure-reference	parameter-part			
			$\text{in-or-inout-parameter}_1$	...	$\text{in-or-inout-parameter}_n$	

الشكل Z.143/32 – بنية نداء إجراء

sender	procedure-reference	parameter-part			value
			$\text{inout-or-out-parameter}_1$	...	$\text{inout-or-out-parameter}_n$	

الشكل Z.143/33 – بنية الإجابة على نداء الإجراء

يكون لكل من الجزأين *sender* و*procedure-reference* المعنى نفسه في الشكلين. ويشير الجزء *sender* إلى مرسل النداء أو الإجابة على نداء الإجراء. ويشير الجزء *procedure-reference* إلى الإجراء المطبق على النداء وعلى الإجابة. ويشير الجزء *parameter-part* من نداء الإجراء في الشكل 32 إلى المعلمات **in** والمعلمات **inout**، ويشير الجزء *parameter-part* من الإجابة في الشكل 33 إلى المعلمات **inout** والمعلمات **out** في الإجراء المطبق على النداء والإجابة. وعلاوة على ذلك يكون، للإجابة جزء *value* لقيم العودة في الإجابة على نداء الإجراء.

الملاحظة 1 – كما ورد في الملاحظة السابقة (انظر 1.4.8) لا تقدم الدلالة التشغيلية إلا نموذجاً لمفاهيم الترميز TTCN-3. ومعرفة ما إذا كان يجب إرسال و/أو استقبال المعلومات الواردة في الشكلين 32 و33 وكيف يحدث ذلك تتوقف على تنفيذ نظام الاختبار.

الملاحظة 2 – بالنسبة لنداء الإجراء، لا تنطبق المعلمات **out** ولذلك لا ترد في الشكل 32. وبالنسبة للإجابة على نداء الإجراء، لا تنطبق المعلمات **in** ولذا لا ترد في الشكل 33.

الملاحظة 3 – يمكن دائماً الحصول على أنماط المعلمات ونمط قيمة العودة دون استثناء انطلاقاً من تعريف التوقيع ذي الصلة.

3.4.8 الاستثناءات

ترتبط الاستثناءات أيضاً بالاتصال القائم على الإجراء. ويبين الشكل 34 بنية الاستثناء، وهي تتكون من أربعة أجزاء. يحدد الجزء *sender* مرسل الاستثناء، ويشير الجزء *procedure-reference* إلى الإجراء الذي ينطبق على الاستثناء، ويحدد الجزء *type* نمط الاستثناء، ويعطي الجزء *value* قيمة الاستثناء. ويحدد توقيع الإجراء المشار إليه في الجزء *procedure-reference* قائمة أنماط الاستثناء المسموح بها. ويجب أن يتطابق الاستثناء المتلقي مع أحد الأنماط المذكورة. وبشكل عام، قد يكون ذلك أي نمط من البيانات TTCN-3 محدد مسبقاً أو محدد من جانب المستعمل.

sender	procedure-reference	type	value

الشكل Z.143/34 – بنية الاستثناء

4.4.8 بناء الرسائل ونداءات الإجراء والإجابات والاستثناءات

إن عمليات إرسال رسالة ما أو نداء إجرائي أو الإجابة على نداء إجرائي أو الاستثناء هي **send** و**call** و**reply** و**raise**. وكل عمليات الإرسال هذه مبنية بالطريقة نفسها:

```
<port-name>.<sending-operation>(<send-specification>) [to <receiver>]
```

ويحدد الجزءان <port-name> و<sending-operation> المنفذ والعملية المستخدمين لإرسال بند ما. وفي حالة التوصيلات من طرف إلى عدة أطراف، يجب تحديد الكيان <receiver>. ويتم بناء البند الذي ينبغي إرساله بواسطة التحديد <send-specification> الذي يمكن أن يستعمل قيماً ملموسة ومراجع نموذجية وقيماً متغيرة وثوابت وعبارات ووظائف وغير ذلك لبناء وتشفير البند الواجب إرساله.

تفترض الدلالة التشغيلية وجود وظيفة نمطية *CONSTRUCT-ITEM* وهي:

CONSTRUCT-ITEM (*myEntity,* <sending-operation>, <send-specification>)

التي تعيد رسالة ما أو نداء الإجراء أو إجابة على نداء الإجراء أو استثناء، وذلك تبعاً للجزأين <sending-operation> و<send-specification> (يشير الجزءان <sending-operation> و<send-specification>) إلى الأجزاء المقابلة في عملية الإرسال TTCN-3، ويكون مرجع الكيان *myEntity* هو مرسل البند الواجب إرساله. ويُفترض كذلك أن تكون معلومات *sender* هذه جزءاً من البند الواجب إرساله (انظر الأشكال 31 إلى 34).

5.4.8 مطابقة الرسائل ونداءات الإجراء والإجابات والاستثناءات

إن عمليات استقبال رسالة ما أو نداء إجرائي أو الإجابة على نداء إجرائي أو الاستثناء هي **receive** و**getcall** و**getreply** و**catch**. وكل عمليات الاستقبال هذه مبنية بالطريقة نفسها:

```
<port-name>.<receiving-operation>(<matching-part>) [from <sender>] [<assignment-part>]
```

ويحدد الجزءان <port-name> و<receiving-operation> المنفذ والعملية المستخدمين لاستقبال بند ما. وفي حالة التوصيلات من طرف إلى عدة أطراف، يمكن استعمال عبارة **from** لانتقاء كيان مرسل محدد <sender>. وينبغي للبند الواجب استقباله أن يفي بشروط التزاوج المحددة في الجزء <matching-part>. ويمكن أن يستعمل الجزء <matching-part> قيماً ملموسة ومراجع نموذجية وقيماً متغيرة وثوابت وعبارات ووظائف، وغير ذلك. لتحديد شروط المطابقة.

وتفترض الدلالة التشغيلية وجود وظيفة نمطية *MATCH-ITEM*:

MATCH-ITEM (*myItem,* <matching-part>, <sender>)

وهي تعيد **true** إذا كانت *myItem* تفي بالشروط <matching-part> وإذا تم إرسال *myItem* من جانب <sender>، وإلا فإنها تعيد **false**.

6.4.8 استخراج المعلومات من البنود المتلقاة

يمكن استخراج المعلومات الواردة في الرسائل ونداءات الإجراء والإجابات على نداءات الإجراء والاستثناءات في الجزء <assignment-part> (انظر الفقرة 5.4.8) من وظائف الاستقبال **receive** و**getcall** و**getreply** و**catch**. ويصف الجزء <assignment-part> كيف تنسب معلمات نداءات الإجراء والإجابات وقيم العودة المشفرة في الإجابات والرسائل والاستثناءات ومعرف الكيان <sender> إلى المتغيرات.

وتفترض الدلالة التشغيلية وجود وظيفة نمطية *RETRIEVE-INFO* كما يلي:

RETRIEVE-INFO (*myItem*, <assignment-part>)

حيث تُستخرج جميع القيم الواجب استخراجها وفقاً للجزء <assignment-part> وتنسب إلى المتغيرات المذكورة في جزء التخصيص. ويتم ذلك بواسطة العملية *VAR-SET*. أي أن المتغيرات في نفس الموقع تحدّث في آنٍ واحد.

5.8 سجلات النداء للوظائف والخطوات البديلة واختبارات الحالة

تستدعي (أو تنفذ) الوظائف والخطوات البديلة واختبارات الحالة بأسمائها وبقائمة من المعلمات الفعلية. وتتضمن المعلمات الفعلية مراجع إذا كانت المعلمات مرجعية وقيماً ملموسة إذا كانت المعلمات بحسب القيم، كما تحدده المعلمات الشكلية في تحديد الوظيفة أو اختبار الحالة. وتعالج الدلالة التشغيلية نداءات الوظيفة والخطوات البديلة واختبار الحالة بواسطة *سجلات النداء* كما هو وارد في الشكل 35. والقيمة *BEHAVIOUR-ID* هي اسم وظيفة أو اختبار حالة، وتشمل المعلمات حسب القيمة قيماً ملموسة <$parId_1$> ... <$parId_n$> للمعلمات

الشكلية <*parId*$_1$> ... <*parId*$_n$>. وتشمل المعلمات المرجعية مراجع إلى مواضع المتغيرات والمؤقتات القائمة. وقبل تنفيذ أي وظيفة أو اختبار حالة، ينبغي بناء سجل نداء مناسب.

behaviour-id	value-parameter				reference-parameter			
		$parId_1$	...	$parId_n$		$parId_1$	...	$parId_n$
		$value_1$	...	$value_n$		loc_1	...	loc_n

الشكل Z.143/35 – بنية سجل النداء

1.5.8 معالجة سجلات النداء

يمكن استخراج اسم الوظيفة أو اختبار الحالة وقيم المعلمات الفعلية بواسطة عامل النقاط، مثل *myCallRecord.parId*$_n$ أو *myCallRecord.behaviour-id*، حيث *myCallRecord* مؤشر يشير إلى سجل النداء.

ولأغراض بناء نداء ما من المفترض أن تكون الوظيفة *NEW-CALL-RECORD* متاحة:

NEW-CALL-RECORD(*myBehaviour*)

حيث تستحدث سجل نداء جديد للوظيفة أو اختبار الحالة *myBehaviour* وتعيد المؤشر باتجاه السجل الجديد. وتكون قيم مجالات معلمات سجل النداء الجديد غير محددة.

myEntity.*INIT-CALL-RECORD*(*myCallRecord*)

متغيرات ومؤقتات لمعالجة المعلمات حسب القيم والمعلمات المرجعية في النطاق الفعلي لمكونة الاختبار أو التحكم في الوحدة *myEntity*. وتدمث المتغيرات لمعالجة المعلمات بحسب القيم على أساس القيم المقابلة في سجل النداء. وتحصل المتغيرات والمؤقتات لمعالجة المعلمات المرجعية على الموضع المحدد. وعلاوة على ذلك تحصل على قيمة متغير أو مؤقت قائم في وحدة تطبيق أخرى للمكونة التي استحدث فيها سجل النداء.

6.8 إجراء تقييم النموذج TTCN-3

1.6.8 مراحل التقييم

يتكون إجراء تقييم الوحدة TTCN-3 من المراحل التالية:

1) *مرحلة التدميث*؛
2) *مرحلة التحديث*؛
3) *مرحلة الانتقاء*؛
4) *مرحلة التنفيذ*.

تكرر المراحل 2) و3) و4) حتى انتهاء التحكم في الوحدة. ويوصف إجراء التقييم في مزيج من نص غير شكلي وشبه شفرة والوظائف المذكورة في الفقرات السابقة.

1.1.6.8 المرحلة I: التدميث

تتضمن مرحلة التدميث الإجراءات التالية:

أ) **إعلان المتغيرات وتدميثها:**

– *INIT-FLOW-GRAPHS*(); // Initialization of flow graph handling. *INIT-FLOW-GRAPHS* is // explained in 8.6.2

– *Entity* := **NULL**; // *Entity* will be used to refer to an entity state. An entity state either // represents module control or a test component.

ملاحظة – تشكل المتغيرات الإجمالية التالية *ALL-ENTITY-STATES* و*ALL-PORT-STATES* و*MTC* و*TC-VERDICT* و*DONE* حالة الوحدة التي تم التصرف فيها خلال تفسير الوحدة TTCN-3 (انظر الفقرة 1.3.8).

– *ALL-ENTITY-STATES* := **NULL**;

– *ALL-PORT-STATES* := **NULL**;

- *MTC* := **NULL**;
- *TC-VERDICT* := **none**;
- *DONE* := **NULL**;
- *SNAP-DONE* := 0;

ب) **استحداث وتدميث التحكم في الوحدة:**

- *Entity:= NEW-ENTITY* (*GET-UNIQUE-ID*(),*GET-FLOW-GRAPH* (<*moduleId*>));
  // A new entity state is created and initialized with the start
  // node of the flow graph representing the behaviour of the
  // control of the module with the name <*moduleId*>.
  // *GET-UNIQUE-ID* is explained in 8.6.2.
- *Entity*.*INIT-VAR-SCOPE*(); // New variable scope
- *Entity*.*INIT-TIMER-SCOPE*(); // New timer scope
- *Entity*.*VALUE-STACK*.*push*(**MARK**); // A mark is pushed onto the value stack
- *ALL-ENTITY-STATES*.*append*(*Entity*); // The new entity is put into the module state.

2.1.6.8 المرحلة II: التحديث

تتناول مرحلة التحديث جميع الإجراءات التي لا تدخل في نطاق الدلالة التشغيلية ولكنها تؤثر في تفسير الوحدة TTCN-3. وتتكون مرحلة التحديث من الإجراءات التالية:

أ) **التقدم الزمني:** تحدّث جميع المؤقتات العاملة، أي (ربما) تتراجع القيم *TIME-LEFT* للمؤقتات الحالية، وإذا انتهت مدة أحد المؤقتات بسبب عملية التحديث، يتم تحديث وصلات المؤقت المقابل، أي يوضع *TIME-LEFT* إزاء 0,0 ويوضع *STATUS* إزاء **TIMEOUT**؛

الملاحظة 1 – يشمل تحديث المؤقتات جميع المؤقتات *TIMER-GUARD* الجارية في حالات الوحدة. وتستخدم المؤقتات *TIMER-GUARD* لحماية تنفيذ اختبارات الحالة وعمليات النداء.

ب) **سلوك النظام قيد الاختبار SUT:** توضع الرسائل ونداءات الإجراء البعيدة والإجابات على نداءات الإجراء البعيد والاستثناءات التي (ربما) ترد من النظام SUT في طوابير المنافذ التي تجري عندها عمليات الاستقبال المقابلة.

الملاحظة 2 – لا تضع هذه الدلالة التشغيلية أي افتراضات تتعلق بالتقدم الزمني وسلوك النظام SUT.

3.1.6.8 المرحلة III: الانتقاء

تتكون مرحلة الانتقاء من الإجراءين التاليين:

أ) **الانتقاء:** انتقاء كيان غير ممنوع، أي كيان تكون القيمة *STATUS* الخاصة به **ACTIVE** أو **SNAPSHOT**؛

ب) **التخزين:** تخزين معرف الكيان المنتقى في المتغير الإجمالي *Entity*.

4.1.6.8 المرحلة IV: التنفيذ

تتكون مرحلة التنفيذ من الإجراءين التاليين:

أ) **خطوة تنفيذ الكيان المنتقى:** تنفيذ عقدة المخطط الانسيابي في أعلى المكدّس *CONTROL-STACK* في *Entity*؛

ب) **التحقق من معيار الانتهاء:** وقف التنفيذ إذا كان التحكم في الوحدة منتهياً، أي التحقق مما إذا كانت قائمة حالات الكيان فارغة، وإلا تواصل إجراءات المرحلة II.

ملاحظة – يمكن اعتبار خطوة تنفيذ الكيان المنتقى بمثابة نداء إجرائي. ويتم التحقق من معيار الانتهاء عند الانتهاء من خطوة التنفيذ، أي عندما يعاد التحكم.

2.6.8 الوظائف العامة

يستخدم إجراء التقييم الوظيفتين العامتين *INIT-FLOW-GRAPHS* و*GET-UNIQUE-ID*:

أ) من المفترض أن تكون *INIT-FLOW-GRAPHS* الوظيفة التي تدمث معالجة المخطط الانسيابي. ويمكن أن تتضمن المعالجة استحداث المخططات الانسيابية ومعالجة المؤشرات نحو المخططات الانسيابية والعقد الخاصة بها.

ب) من المفترض أن تكون *GET-UNIQUE-ID* وظيفة تعيد معرفاً وحيداً كلما تم استدعاؤها. وقد يكون المعرف الوحيد في شكل متغير عداد يزداد ويعاد كل مرة يتم فيها استدعاء *GET-UNIQUE-ID*.

تستخدم شبه الشفرة الواردة في الفقرات التالية لوصف تنفيذ عُقد المخططات الانسيابية الوظائف *CONTINUE-COMPONENT* و*RETURN* و***DYNAMIC-ERROR***:

أ) *CONTINUE-COMPONENT*: تستمر مكونة الاختبار الفعلية للتنفيذ وتكون العقدة في أعلى مكدّس التحكم، أي لا يعاد التحكم إلى إجراء تقييم الوحدة الوارد في هذه الفقرة.

ب) *RETURN* يعيد التحكم مجدداً إلى إجراء تقييم الوحدة الوارد في هذه الفقرة. وتشكل الوظيفة *RETURN* الإجراء الأخير من *خطوة تنفيذ الكيان المنتقى في مرحلة* التنفيذ.

ج) تشير ***DYNAMIC-ERROR*** إلى حدوث خطأ دينامي. ولا يدخل إجراء معالجة الأخطاء في حد ذاته في إطار الدلالة التشغيلية. وفي حال حدوث خطأ دينامي فإن من المفترض عدم تحديد مجموع السلوك التالي في اختبار الحالة. وعندئذ، ينبغي إزالة الموارد المخصصة إلى اختبار الحالة وعزو القرار error إلى اختبار الحالة. ويعطى التحكم إلى البيان في الجزء المتعلق بالتحكم الذي يلي بيان التنفيذ حيث حدث الخطأ. وتبدو نمذجة ذلك في الجزء <dynamic-error> من مخطط الانسياب (انظر الفقرة 18b.9).

ملاحظة – إن حدوث خطأ دينامي ما يتعلق بسلوك الاختبار. ويشير الخطأ الدينامي، كما تحدده الدلالة التشغيلية، إلى مشكلة في استخدام الترميز TTCN-3، استخدام خاطئ أو حالة مزاحمة مثلاً.

د) تُستخدم *APPLY-OPERATOR* باعتبارها وظيفة نمطية لوصف تقييم المعامل (+ أو * أو / أو –) في العبارات (انظر الفقرة 4.18.9).

9 مقاطع المخطط الانسيابي في بناء TTCN-3

تمثل الدلالة التشغيلية سلوك TTCN-3 في شكل مخططات انسيابية. وترد خوارزمية بناء المخططات الانسيابية التي تمثل السلوك في الفقرة 2.8. وهي تستند إلى نماذج مخططات انسيابية ومقاطع مخططات الواجب استخدامها لبناء مخططات ملموسة للتحكم في الوحدة واختبارات الحالة والخطوات البديلة والوظائف وأنماط المكونات المحددة في الوحدة TTCN-3. وتتناول هذه الفقرة نماذج مقاطع المخطط الانسيابي. وهذه النماذج مقدمة بترتيب هجائي وليس بترتيب منطقي.

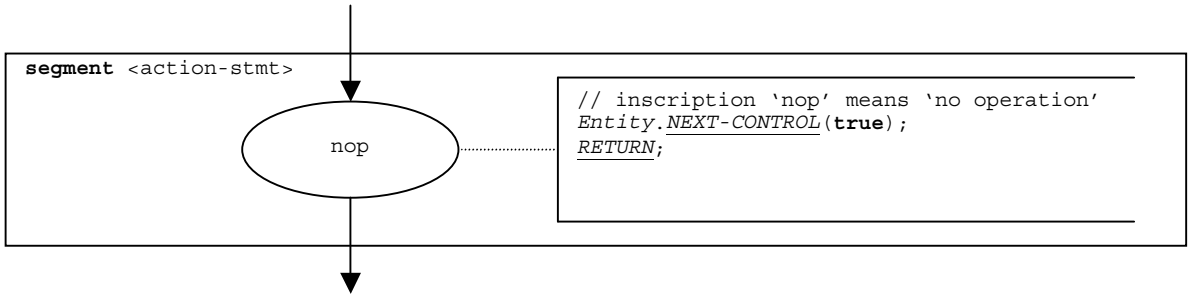
وتعرض تعاريف مقاطع المخططات ضمن الأشكال. وتكون عقد مخطط الانسياب في الجزء الأيسر من الأشكال والتعليقات المرتبطة بهذه العقد وخطوط الانسياب تظهر في الجانب الأيمن. وتحتوي الأشكال على تعليقات وصفية للعقد المرجعية وتعليقات في شكل شبه شفرة للعقد الأساسية. وتصف شبه الشفرة كيف يتم تفسير عقدة أساسية ما، أي كيف تعدل حالة الوحدة. وهي تستخدم الوظائف المحددة في الفقرة 8 والمتغيرات الإجمالية المعلن عنها والمدمثة في إجراء تقييم الوحدات TTCN-3 (انظر الفقرة 6.8). وتحتوي الفقرة 8 على نظرة عامة لإجمالي الوظائف والكلمات الأساسية المستخدمة في شبه الشفرة.

1.9 البيان action

تكون بنية بيان الإجراء **action** كما يلي:

```
action (<informal description>)
```

يحدد مقطع مخطط الانسياب <action-stmt> في الشكل 36 تنفيذ البيان **action**.



ملاحظة – ليس للمعلمة <informal description> في البيان **action** أي معنى للدلالة التشغيلية وبالتالي فهي ليست ممثلة في مقطع المخطط.

الشكل Z.143/36 – مقطع مخطط الانسياب <action-stmt>

2.9 البيان activate

تكون بنية البيان **activate** كما يلي:

activate(<altstep-name>([<act-par-desc$_1$>, ... , <act-par-desc$_n$>]))

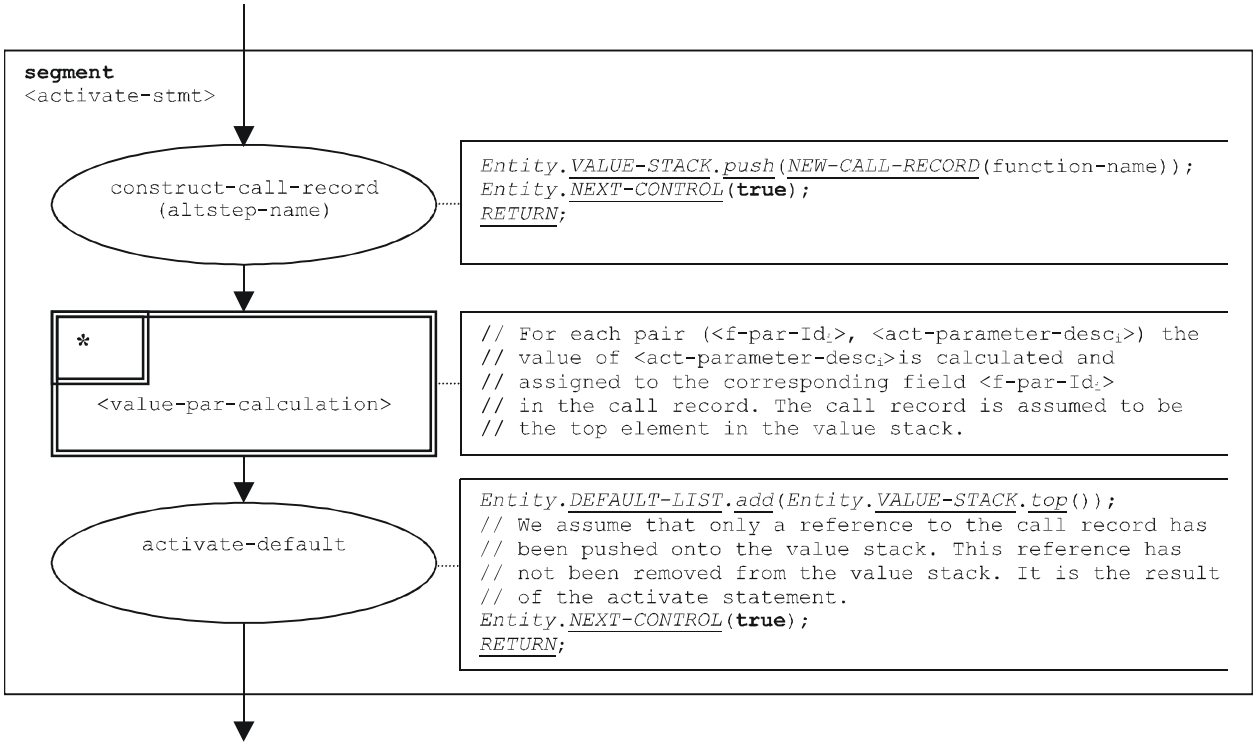
وتشير <altstep-name> إلى اسم خطوة بديلة فاعلة باعتبارها السلوك بالتغيب وتصف ... ,<act-par-descr1> <act-par- descrn> قيم المعلمات الفعلية للخطوة البديلة عند تفعليها.

ويفترض أنه لكل وصف <act-par-desc1>، يكون معرف المعلمة الشكلي المقابل <f-par-Id1> معروفاً، أي من الممكن تمديد البنية الواردة أعلاه على النحو التالي:

activate(<altstep-name>((<f-par-Id$_1$>,<act-par-desc$_1$>), ... , (<f-par-Id$_n$>,<act-par-desc$_n$>)))

ويحدد المقطع <activate-stmt> في الشكل 37 تنفيذ البيان activate. وينقسم التنفيذ إلى ثلاث مراحل. في المرحلة الأولى، يستحدث سجل نداء للخطوة البديلة <function-name>. وفي المرحلة الثانية، تحسب قيم المعلمة الفعلية وتنسب إلى المجال المقابل في سجل النداء. وفي المرحلة الثالثة، يوضع سجل النداء باعتباره العنصر الأول في القائمة *DEFAULT-LIST* للكيان الذي ينشط السلوك بالتغيب.

ملاحظة – بالنسبة إلى الخطوات البديلة المنشطة باعتبارها سلوكاً بالتغيب، لا يسمح إلا بالمعلمات بحسب القيم. وفي الشكل 37، توصف في المقطع <value-par-calculation> معالجة المعلمات بحسب القيم، وهي محددة في الفقرة 1.24.9.

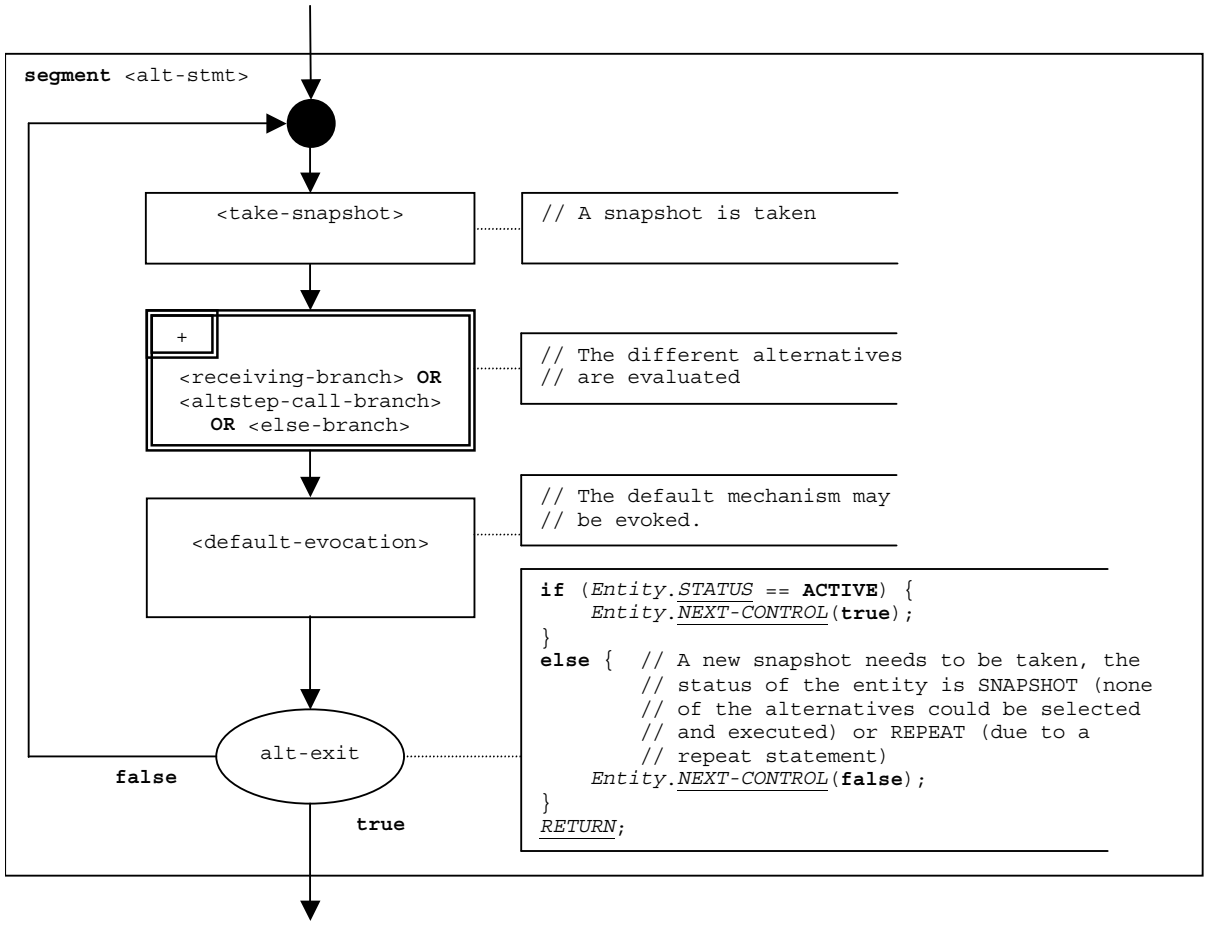


الشكل Z.143/37 – مقطع مخطط الانسياب <activate-stmt>

3.9 البيان alt

إن البيان **alt** من أكثر البيانات تعقيداً وأهمية للترميز TTCN-3. فهو يطبق دلالة اللقطة الآنية ويحدد التفريع الناجم عن استلام الرسائل والإجابات والنداءات والاستثناءات الناجمة عن انتهاء المؤقتات وعن انتهاء المكونات. بالإضافة إلى ذلك، يتصل استدعاء آلية التغيب TTCN-3 أيضاً بالبيان **alt**.

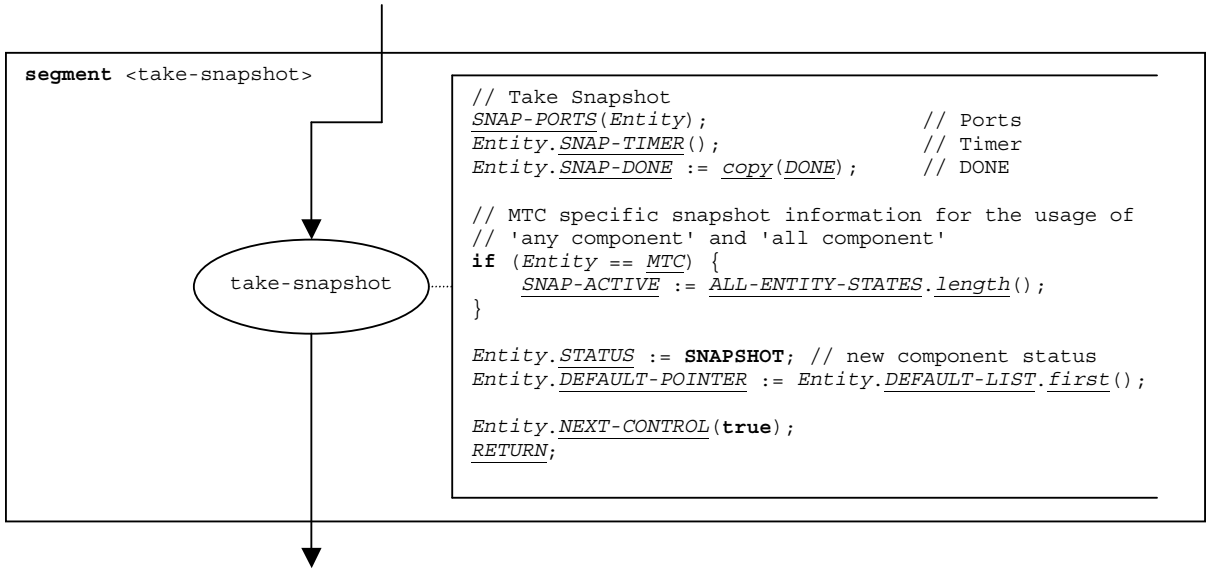
ويرد في الشكل 38 تمثيل بياني للبيان **alt** في شكل مخطط انسياب. وتكون البدائل المختلفة الناجمة عن استلام الرسائل والإجابات والنداءات والاستثناءات وعن انتهاء المؤقتات وعن انتهاء المكونات مستثمرة ضمن <receiving-branch> في مخطط الانسياب.



الشكل Z.143/38 – المقطع <alt-stmt> في مخطط الانسياب

1.3.9 المقطع <take-snapshot> في مخطط الانسياب

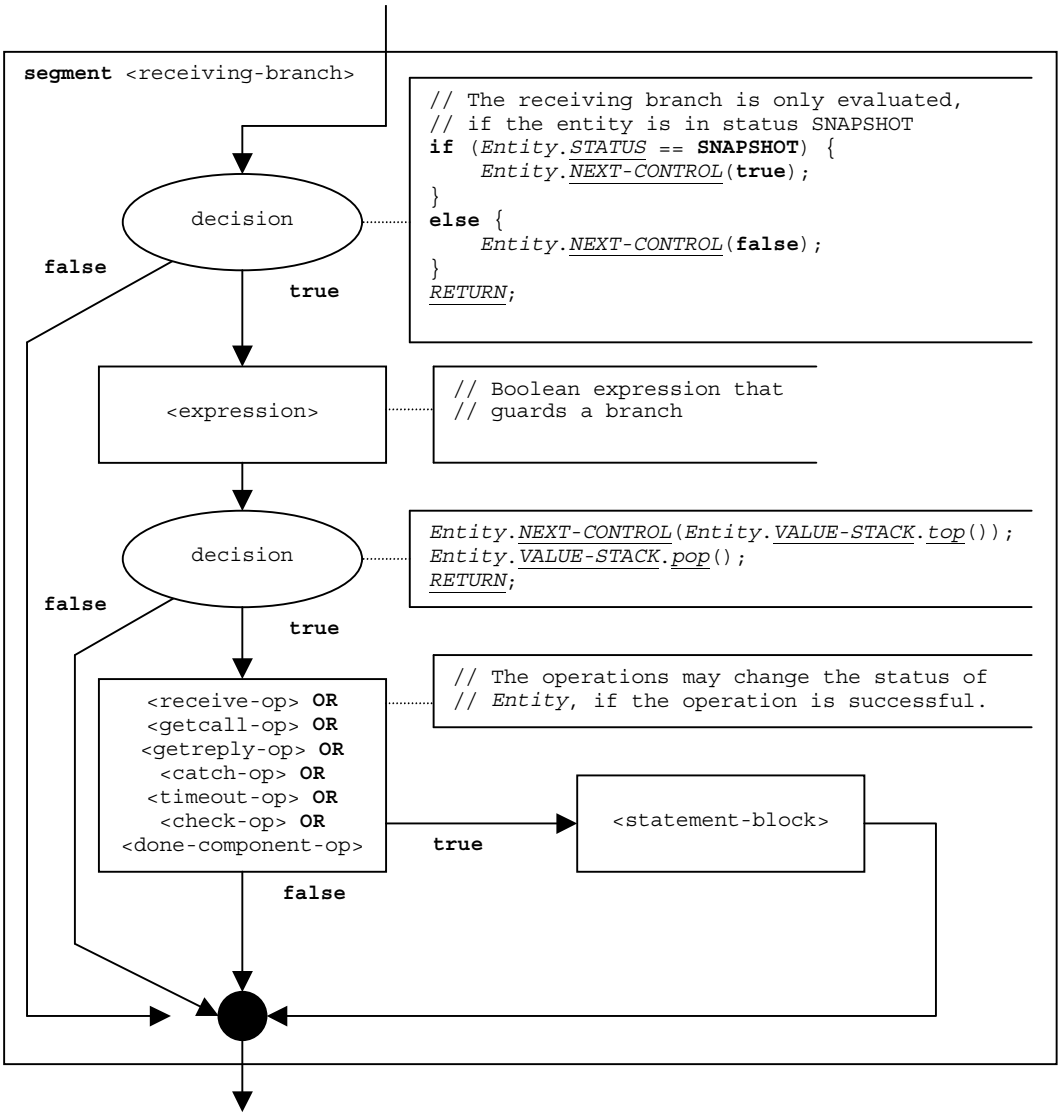
يصف المقطع <take-snapshot> في الشكل 39 إجراء لقطة آنية. وتسجل اللقطة الآنية قيم المنافذ والمؤقتات والمكونات المتوقفة.



الشكل Z.143/39– المقطع <take-snapshot> في مخطط الانسياب

2.3.9 المقطع <receiving-branch> في مخطط الانسياب

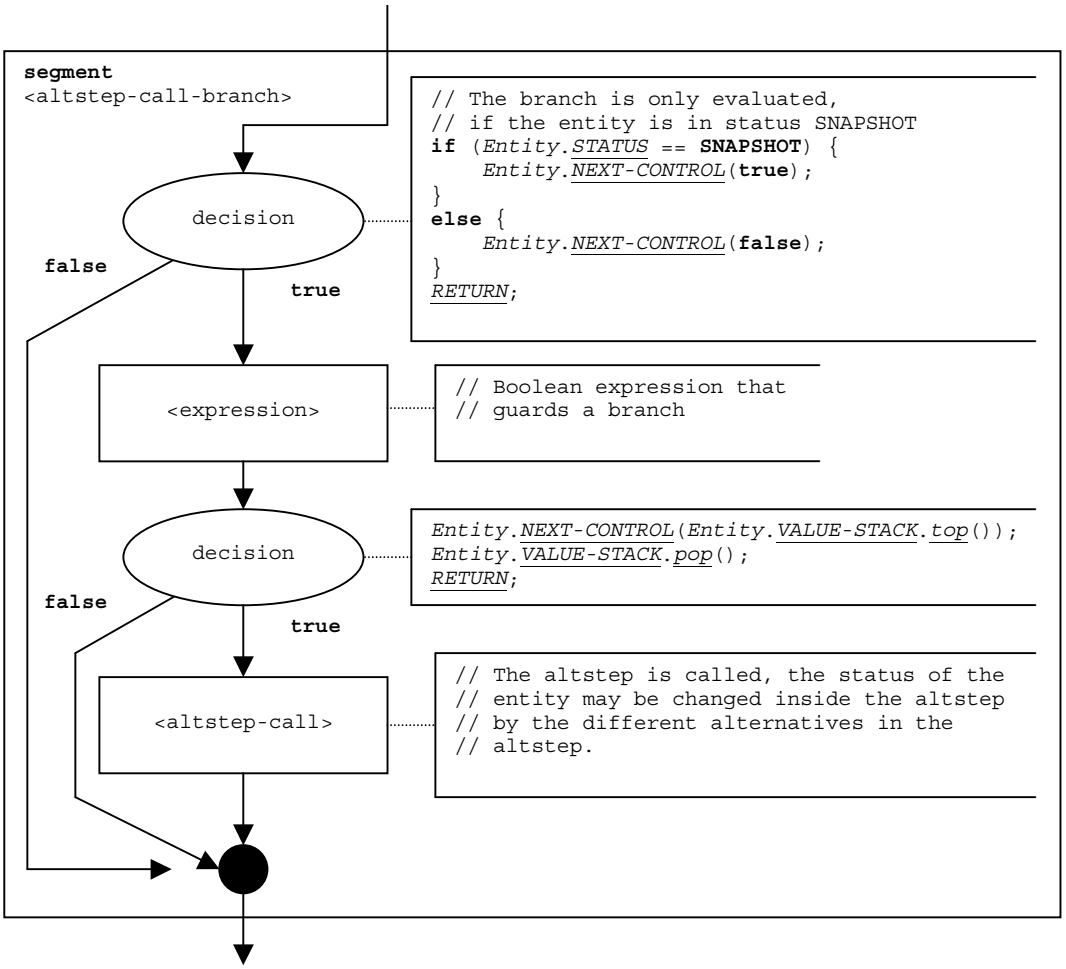
يرد في الشكل 40 تنفيذ المقطع <receiving-branch> في مخطط الانسياب.



الشكل Z.143/40 – المقطع <receiving-branch> في مخطط الانسياب

3.3.9 المقطع <altstep-call-branch> في مخطط الانسياب

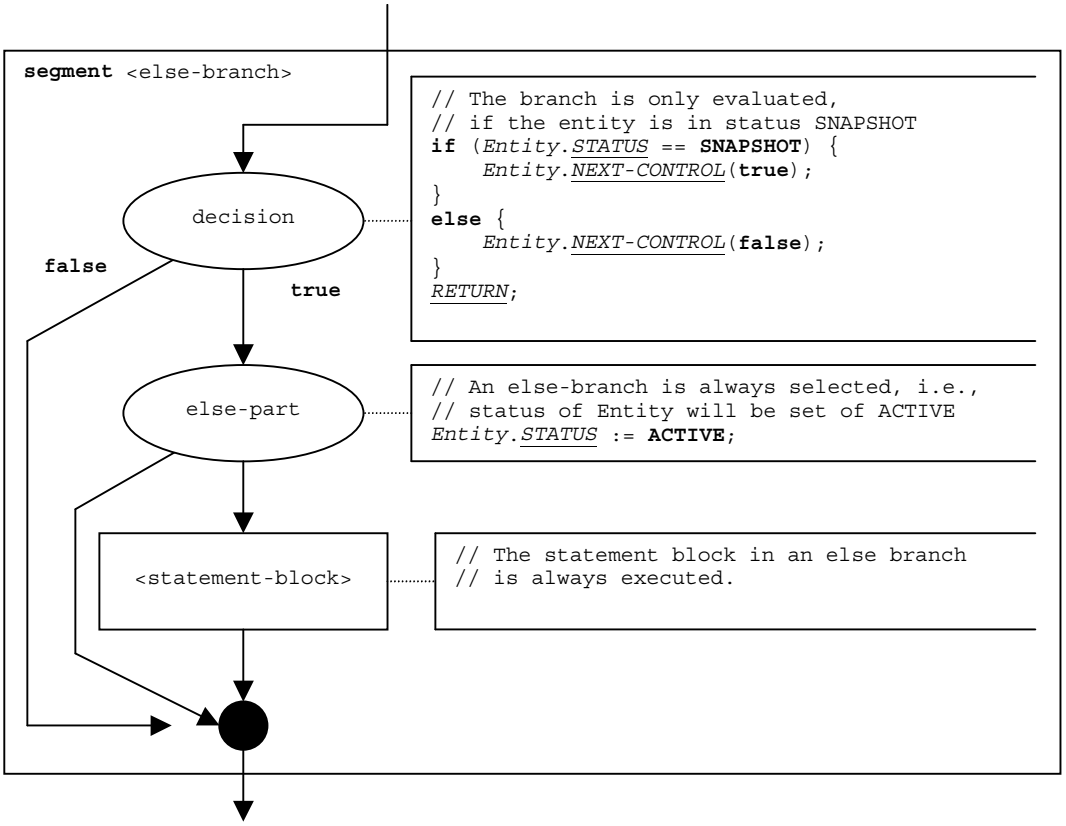
يرد في الشكل 41 استدعاء الخطوة البديلة في البيان **alt** في المقطع <altstep-call-branch> في مخطط الانسياب.



الشكل Z.143/41 – المقطع <altstep-call-branch> في مخطط الانسياب

4.3.9 المقطع <else-branch> في مخطط الانسياب

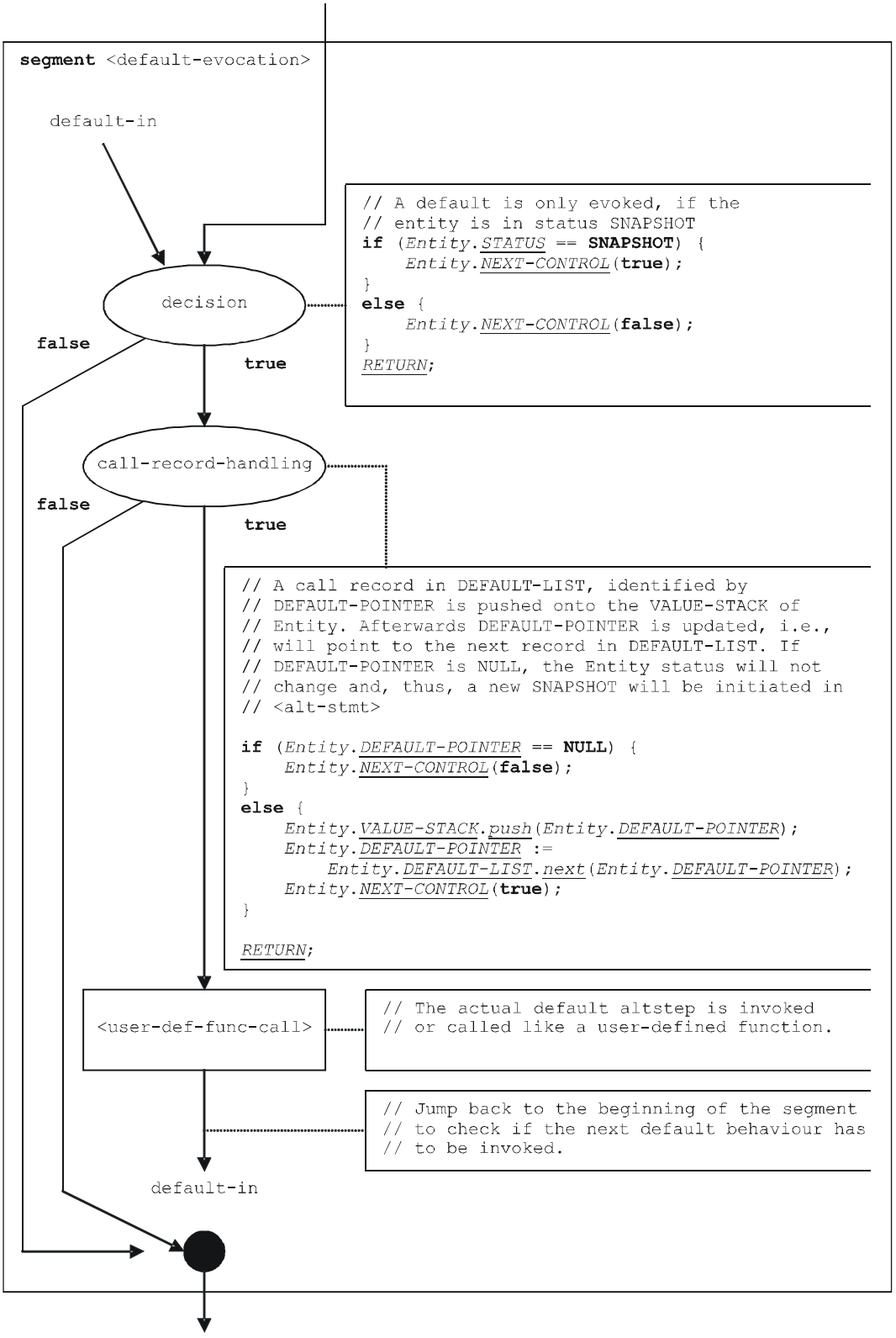
يرد في الشكل 42 تنفيذ الفرع **else** في البيان **alt** في المقطع <else-branch> من مخطط الانسياب.



الشكل Z.143/42 – المقطع <else-branch> في مخطط الانسياب

5.3.9 المقطع <default-evocation> في مخطط الانسياب

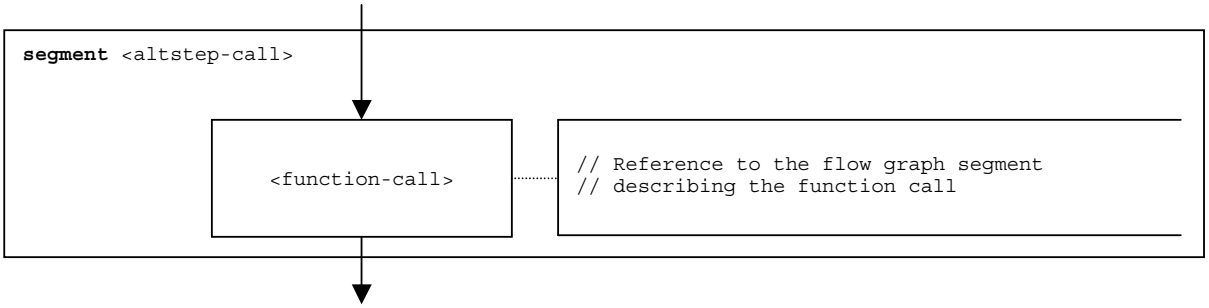
يرد في الشكل 43 استدعاء السلوك بالتغيب في نهاية بيانات **alt** في المقطع <default-evocation> من مخطط الانسياب.



الشكل Z.143/43 – المقطع <default-evocation> في مخطط الانسياب

4.9 النداء altstep

كما يظهر في الشكل 44، يعامَل نداء الخطوة البديلة بمثابة نداء للوظيفة.



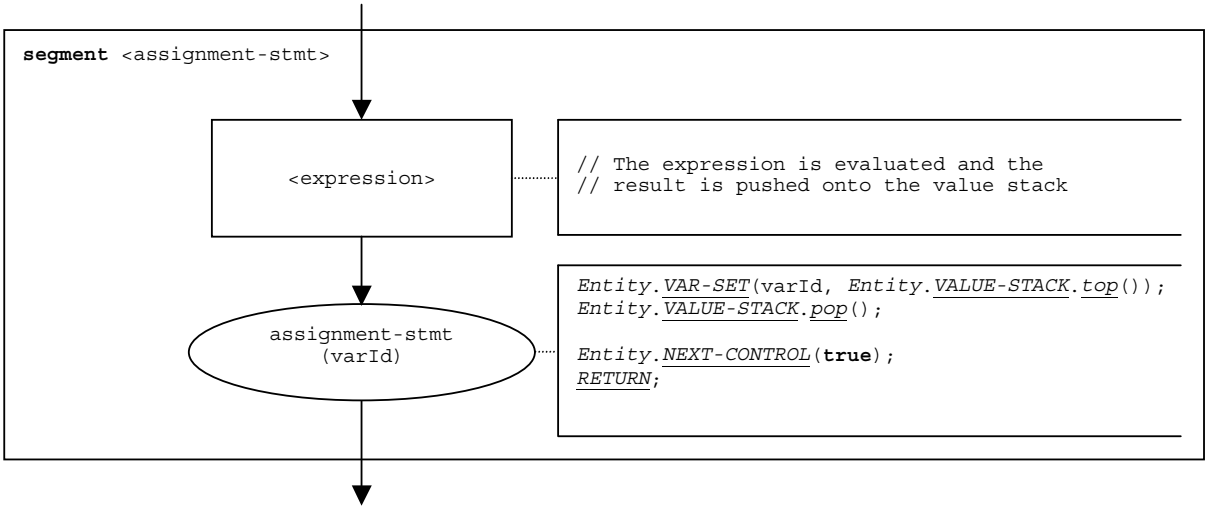
الشكل Z.143/44 – المقطع <altstep-call> في مخطط الانسياب

5.9 البيان assignment

تكون بنية البيان **assignment** كما يلي:

```
<varId> := <expression>
```

وتعطي قيمة التعبير <expression> للمتغير <varId>. ويعرّف تنفيذ البيان assignment بواسطة المقطع <assignment-stmt> من المخطط في الشكل 45.



الشكل Z.143/45 – المقطع <assignment-stmt> في مخطط الانسياب

6.9 العملية Call

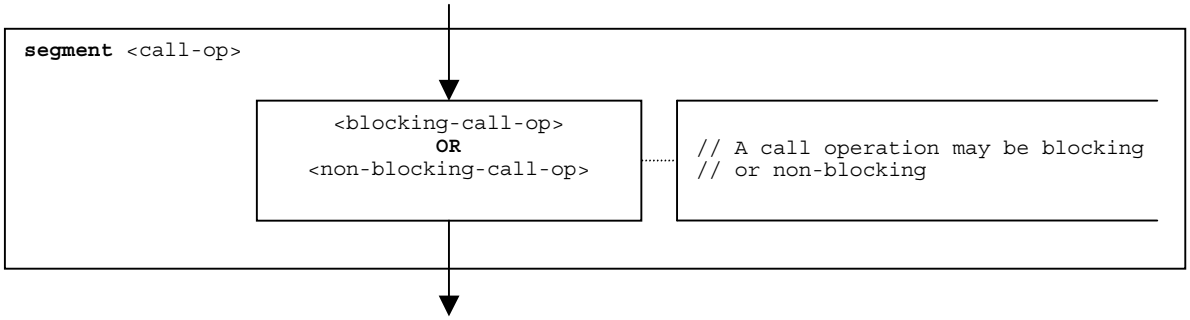
تكون بنية العملية **call** كما يلي:

```
<portId>.call (<callSpec> [<blocking-info>]) [to <component-expression>] [<call-reception-part>]
```

ويتضمن الجزء الاختياري <blocking-info> إما الكلمة الأساسية **nowait** أو مدة للاستثناء في انتهاء المؤقت. ويشير الجزء الاختياري <component-expression> في الحد **to** إلى الكيان المستقبل. وقد يكون إما في شكل قيمة متغيرة أو قيمة عودة في وظيفة، ويشير الجزء <call-reception-part> الاختياري إلى حالات الاَستقبال البديلة في حالة عملية **call** مانعة.

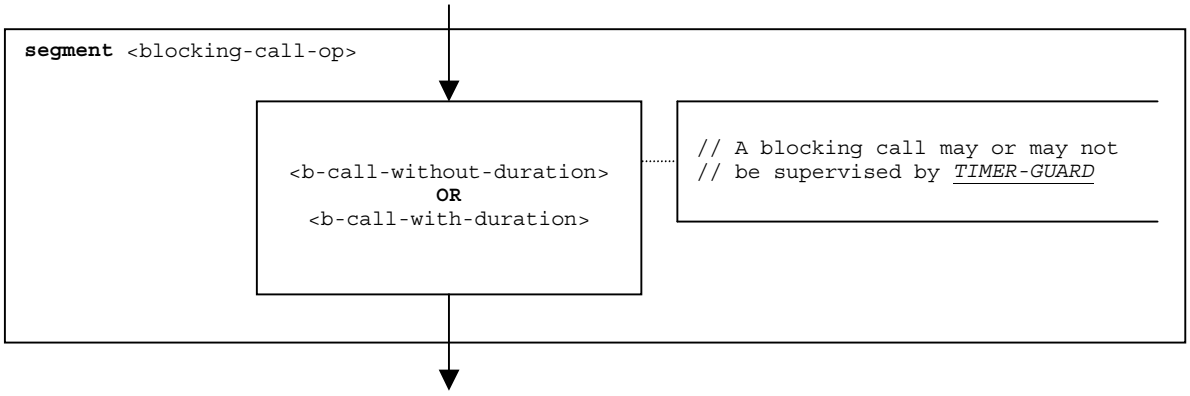
وتفرق الدلالات اللغوية التشغيلية بين عملية **call** المانعة وعملية **call** غير المانعة. وتكون **call** غير مانعة إذا استعملت الكلمة المفتاحية nowait في العملية **call** أو إذا كان الإجراء المستدعى غير مانع، بمعنى معرف باستعمال الكلمة المفتاحية **noblock** – ويكون للعملية call المانعة الجزاء <call.reception-part>

ويحدد المقطع <call-op> من المخطط في الشكل 46 تنفيذ عملية **call**. وهو يعكس التمييز بين النداءات المانعة والنداءات غير المانعة.

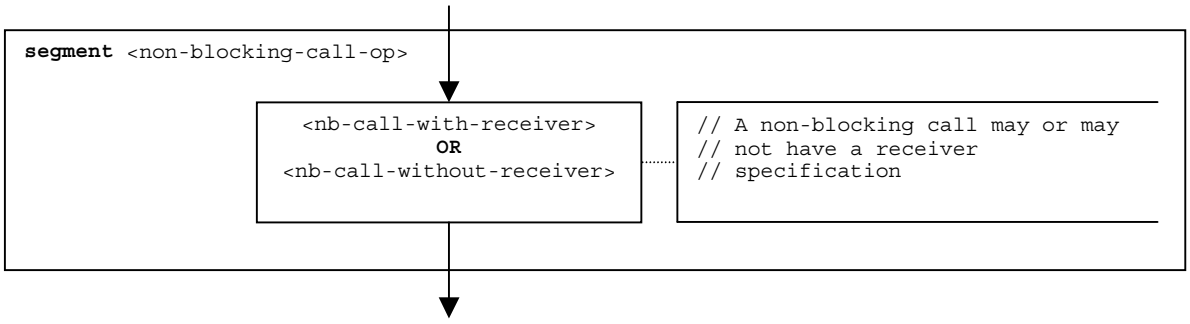


الشكل Z.143/46 – المقطع <call-op> في مخطط الانسياب

بالنسبة لعمليات النداء المانعة وغير المانعة، يمكن تحديد الكيان المستقبل في شكل تعبير. وترد الاحتمالات في الشكلين 47 و48.



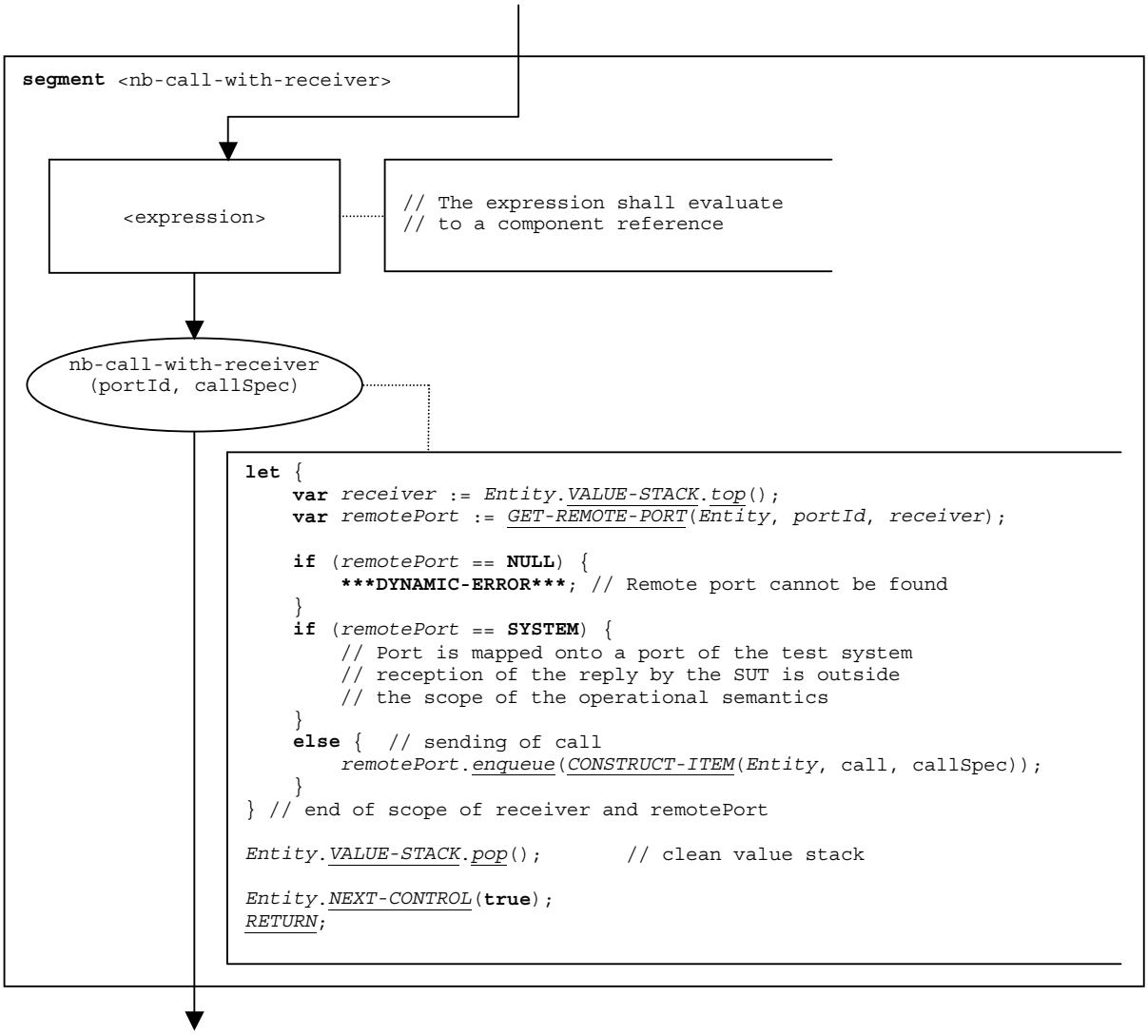
الشكل Z.143/47 – المقطع <blocking-call-op> في مخطط الانسياب



الشكل Z.143/48 – المقطع <non-blocking-call-op> في مخطط الانسياب

1.6.9 المقطع <nb-call-with-receiver> في مخطط الانسياب

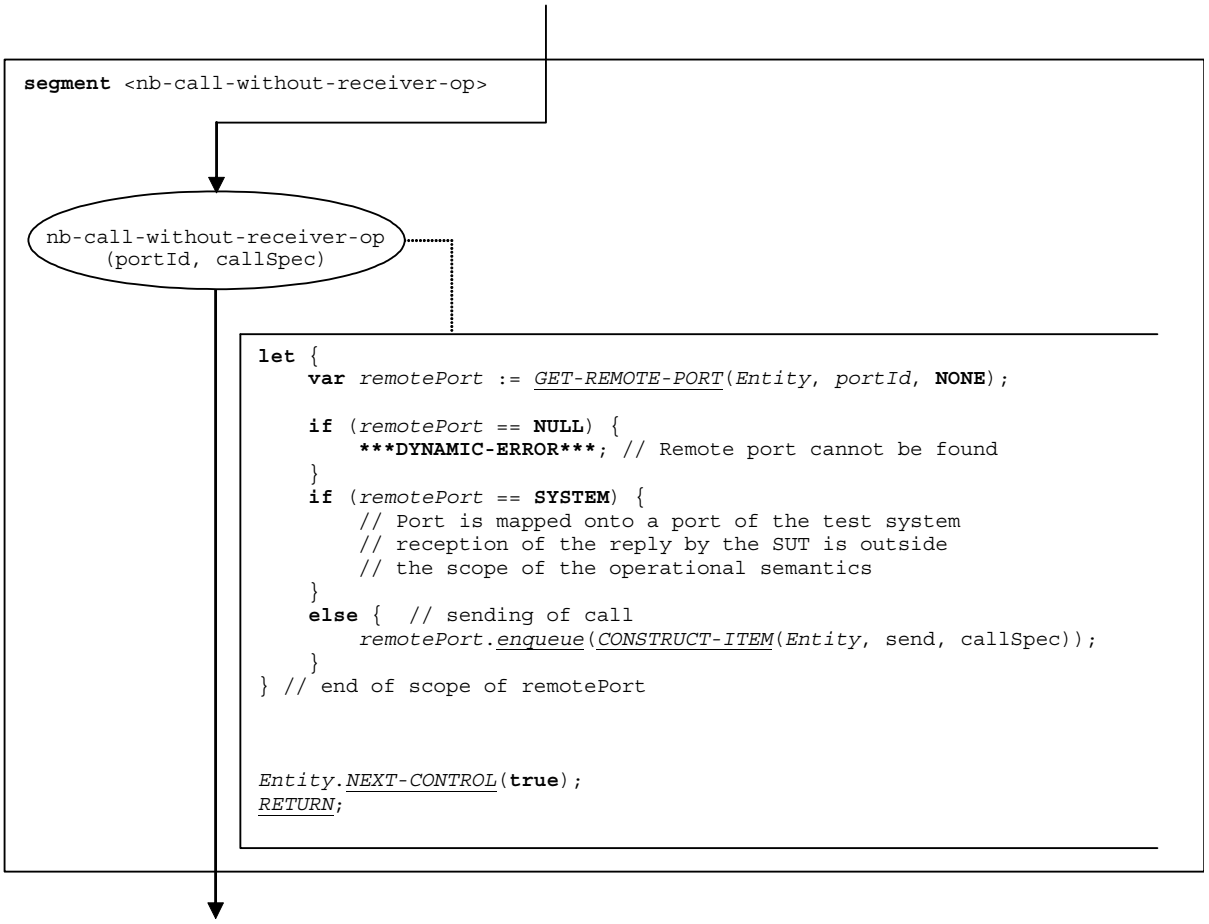
يحدد المقطع <nb-call-with-receiver> في الشكل 49 تنفيذ العملية **call** غير المانعة التي يحدد بشأنها المستقبِل في شكل تعبير.



الشكل Z.143/49 – المقطع <nb-call-with-receiver> في مخطط الانسياب

2.6.9 المقطع <nb-call-without-receiver> في مخطط الانسياب

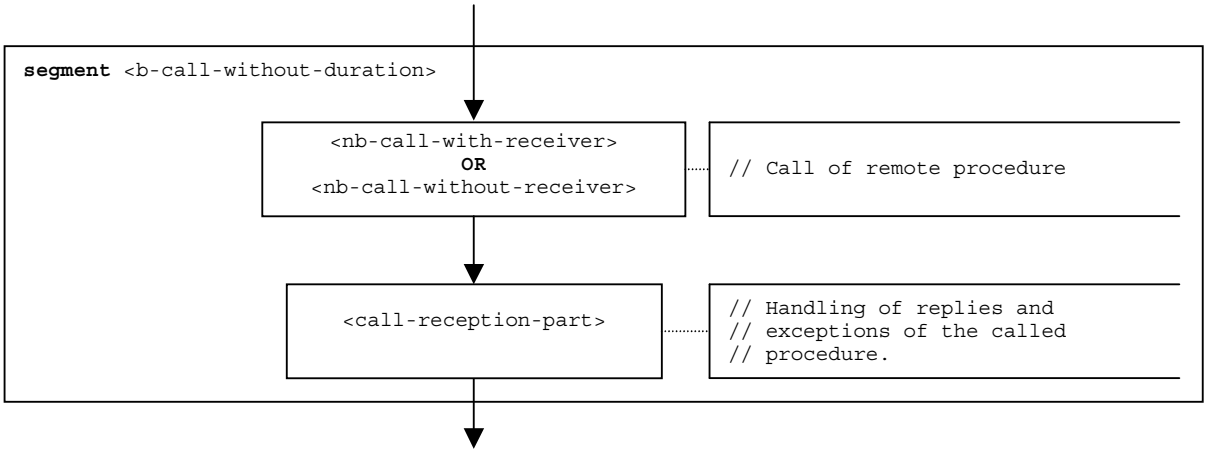
يحدد المقطع <nb-call-without-receiver> في الشكل 50 تنفيذ العملية **call** غير المانعة من دون البند **to**.



الشكل Z.143/50 – المقطع <nb-call-without-receiver> في مخطط الانسياب

3.6.9 المقطع <b-call-without-duration> في مخطط الانسياب

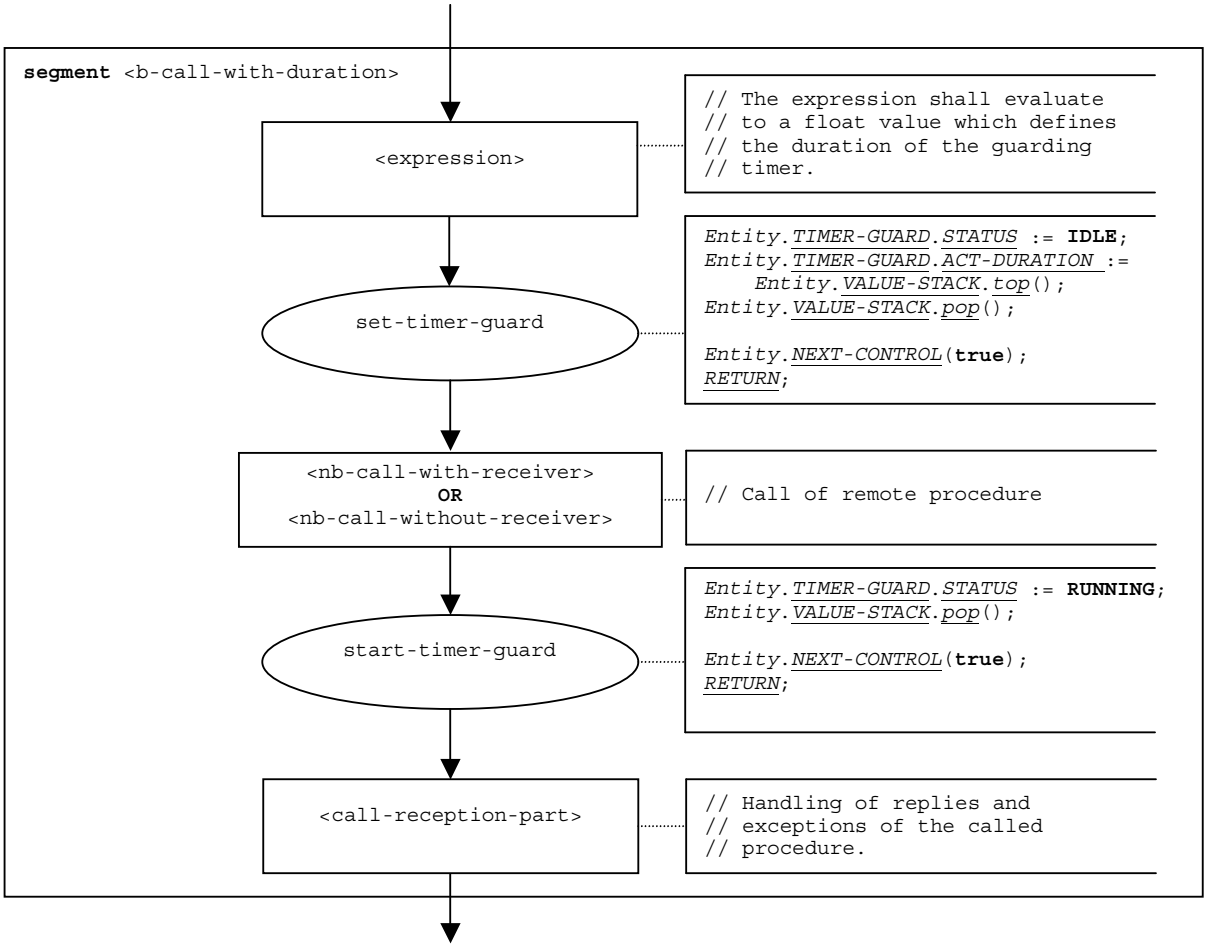
يتم نمذجة النداءات المانعة بنداء غير مانع يتبعه متن النداء الذي يعالج الإجابات والاستثناءات. ويصف المقطع <b-call-without-duration> الوارد في الشكل 51 تنفيذ نداء مانع دون أي مدة زمنية كحارس مؤقّت.



الشكل Z.143/51 – المقطع <b-call-without-duration> في مخطط الانسياب

4.6.9 المقطع <b-call-with-duration> في مخطط الانسياب

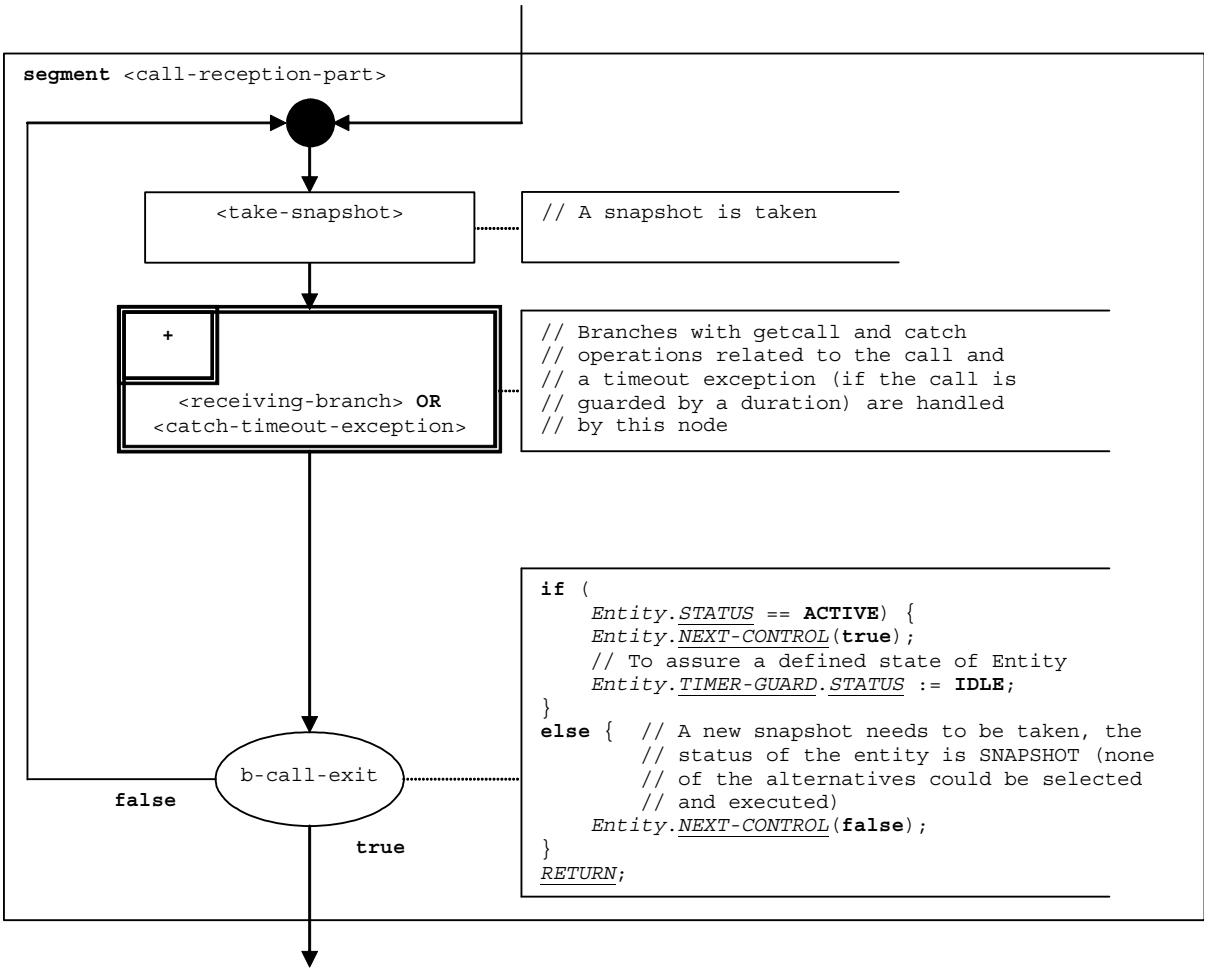
يصف المقطع <b-call-with-duration> (انظر الشكل 52) تنفيذ نداء مانع مع مدة زمنية كحارس مؤقّت.



الشكل Z.143/52 – المقطع <b-call-with-duration> في مخطط الانسياب

5.6.9 المقطع <call-reception-part> في مخطط الانسياب

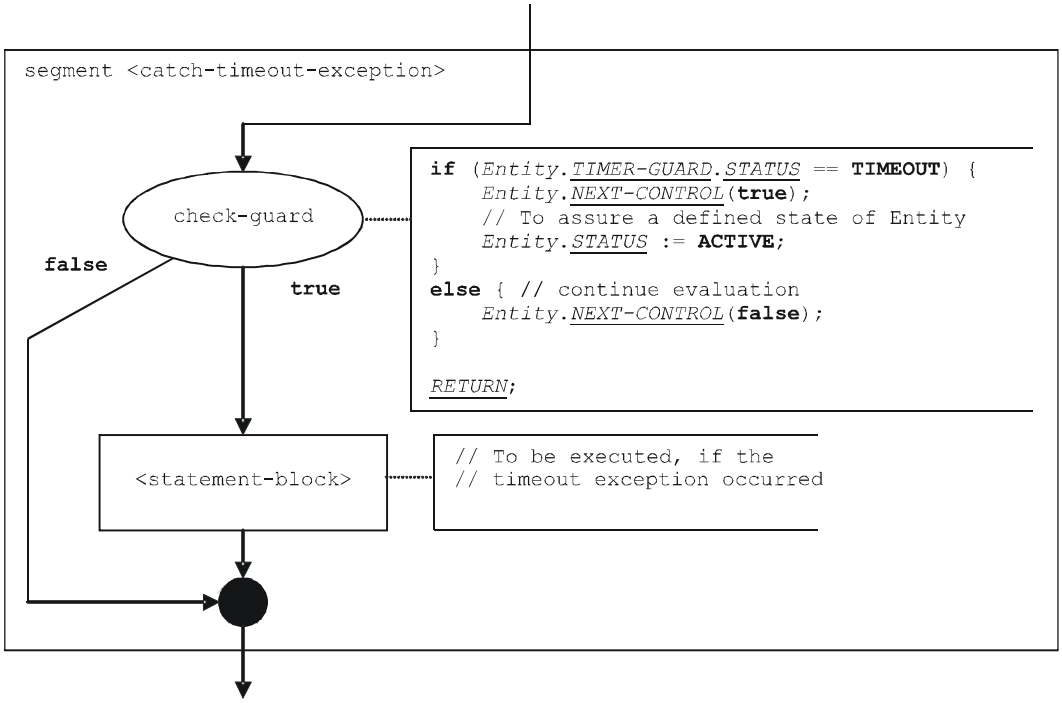
يصف المقطع <call-reception-part> (انظر الشكل 53) معالجة الإجابات والاستثناءات واستثناء انقضاء الوقت لعملية **call** مانعة.



الشكل Z.143/53 – المقطع <call-reception-part> في مخطط الانسياب

6.6.9 المقطع <catch-timeout-exception> في مخطط الانسياب

يتناول المقطع <catch-timeout-exception> (انظر الشكل 54) معالجة استثناء انتهاء الوقت لعملية نداء مانعة محمية لمدة محددة.



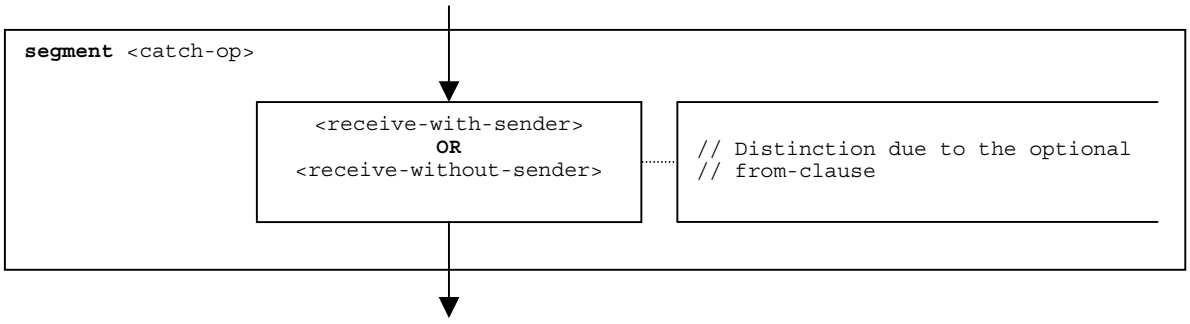
الشكل Z.143/54 – المقطع <catch-timeout-exception> في مخطط الانسياب

7.9 العملية catch

تكون بنية العملية **catch** كما يلي:

```
<portId>.catch (<matchingSpec>) [from <component_expression>] -> [<assignmentPart>]
```

بغض النظر عن الكلمة الأساسية **catch**، تكون هذه البنية مشابهة لبنية العملية **receive**. بالتالي، تعالج الدلالة التشغيلية العملية **catch** كما تعالج العملية **receive**، وهذا يرد في المقطع <catch-op> (الشكل 55) الذي يحدد تنفيذ العملية **catch**. ويشير الشكل إلى مقاطع مخطط الانسياب المرتبطة بالعملية **receive** (انظر الفقرة 37.9).



الشكل Z.143/55 – المقطع <catch-op> في مخطط الانسياب

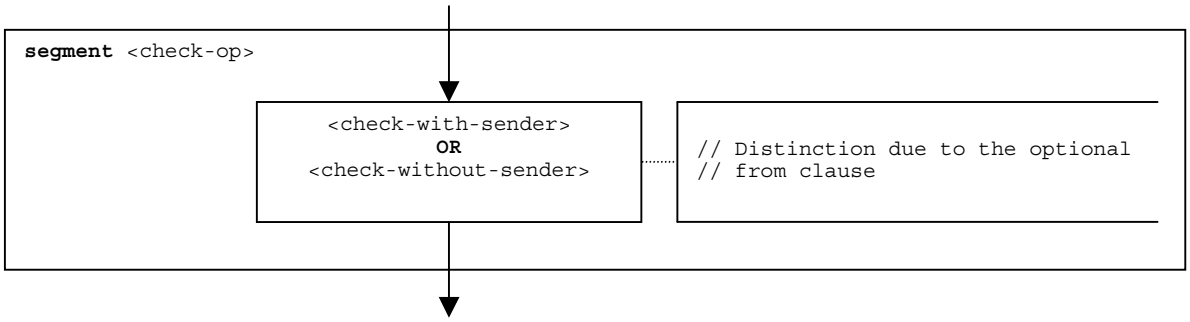
8.9 العملية check

تكون بنية العملية **check** كما يلي:

```
<portId>.check( receive|getcall|catch|getreply (<matchingSpec>)
        [from <component-expression>]) [-> <assignmentPart>]
```

يشير الجزء الاختياري <component-expression> في البند **from** إلى الكيان المرسل. ويمكن أن يكون قيمة متغيرة أو قيمة عودة وظيفة ما، أي أننا نفترض أنها تعبير. ويشير الجزء الاختياري <assignmentPart> إلى تخصيص المعلومات المتلقاة إذا كانت تراعي مواصفة المطابقة <matchingSpec> والبند (الاختياري) **from**.

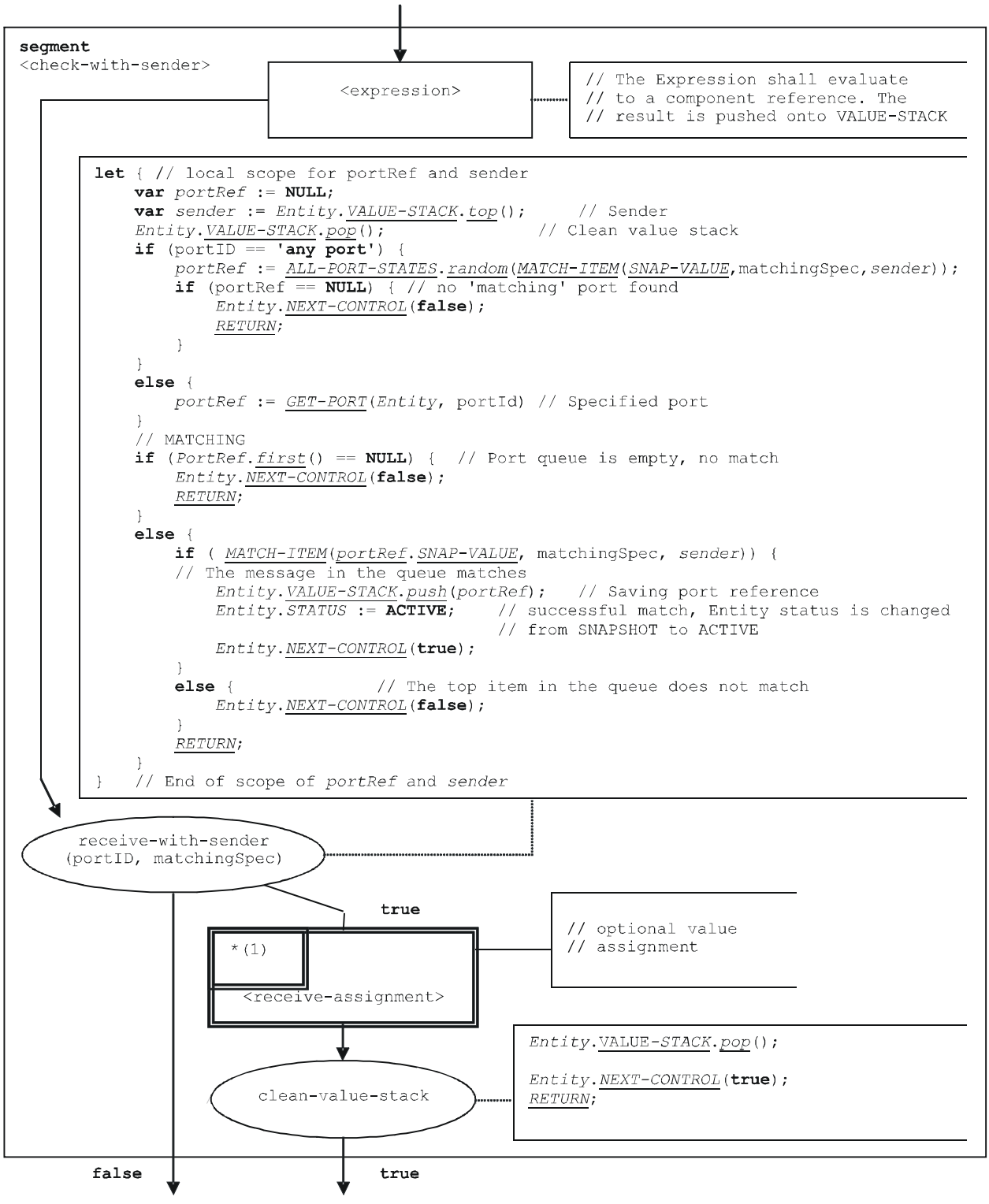
وتعالج الدلالة التشغيلية العمليات **receive** و**getcall** و**catch** و**getreply** بالطريقة نفسها، أي أنها توصف بالإحالة إلى نفسٍ مقاطع مخطط الانسياب <receive-with-sender> و<receive-without-sender>. وتعالج العملية **check** أيضاً مختلف العمليات بالطريقة نفسها. وبالتالي، فإن المقطع <check-op> في الشكل 56 والذي يحدد تنفيذ العملية **check** لا يحيل كذلك إلا إلى مقطعين في مخطط الانسياب. والاختلاف الوحيد فيما يتعلق بالمقطعين <receive-with-sender> و<receive-without-sender> هو أن البنود المتلقاة لا تحذف بعد المطابقة.



الشكل Z.143/56 – المقطع <check-op> في مخطط الانسياب

1.8.9 المقطع <check-with-sender> في مخطط الانسياب

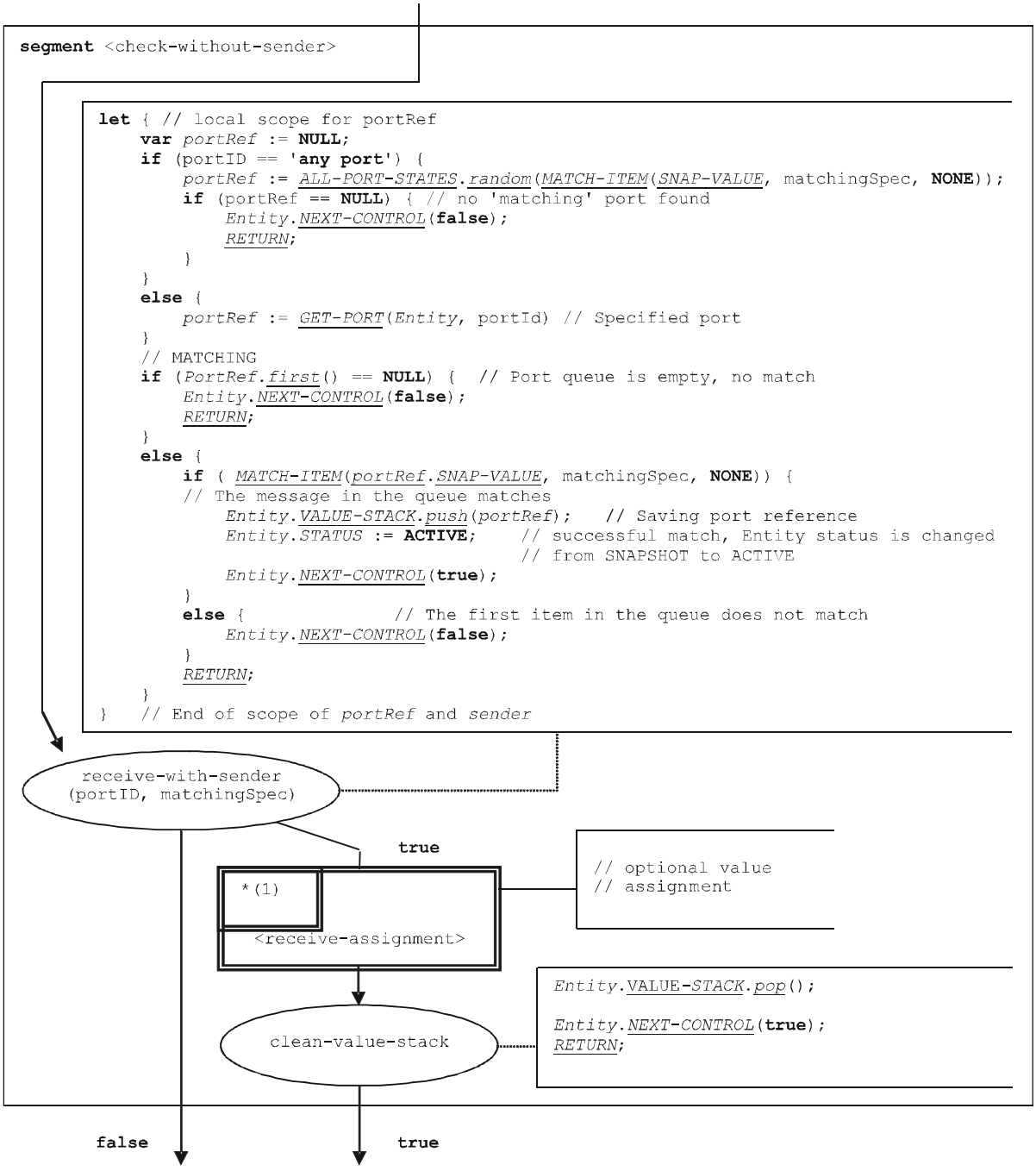
يحدد المقطع <check-with-sender> في الشكل 57 تنفيذ العملية **check** حيث يتم تحديد المرسل في شكل تعبير.



الشكل Z.143/57 – المقطع <check-with-sender> في مخطط الانسياب

2.8.9 المقطع <check-without-sender> في مخطط الانسياب

يحدد المقطع <check-without-sender> في الشكل 58 تنفيذ العملية **check** دون البند **from**.



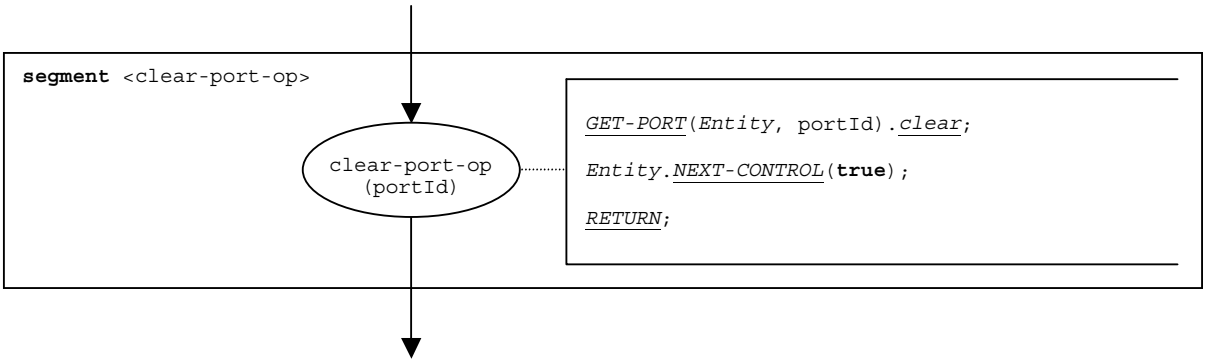
الشكل Z.143/58 – المقطع <check-without-sender> في مخطط الانسياب

9.9 العملية clear المطبقة على المنافذ

تكون بنية العملية **clear** المطبقة على المنافذ كما يلي:

```
<portId>.clear
```

يحدد المقطع <clear-port-op> في الشكل 59 تنفيذ العملية **clear** المطبقة على المنافذ.



الشكل Z.143/59 – المقطع <clear-port-op> في مخطط الانسياب

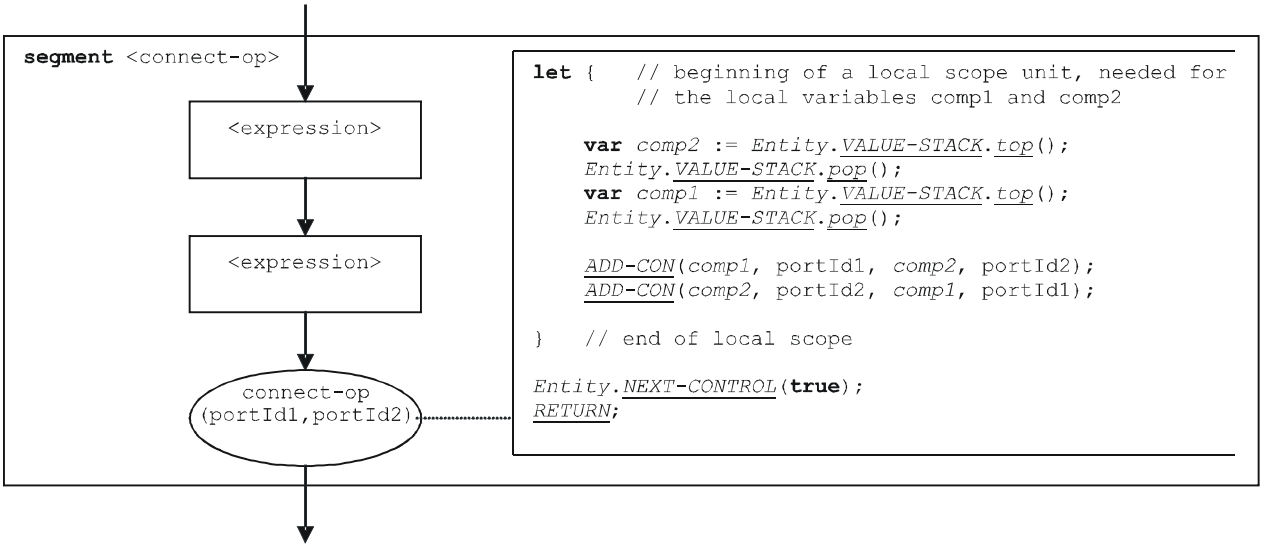
10.9 العملية connect

تكون بنية العملية **connect** كما يلي:

connect(<component-expression$_1$>:<portId1>, <component-expression$_2$>:<portId2>)

يعتبر كل من المعرفين <portId1> و<portId2> معرفاً لمنفذ مكونة الاختبار المقابلة. ويتم الإحالة إلى المكونة التي ينتمي إليها المنفذ بواسطة مرجع المكونة <component-expression$_1$> و<component-expression$_2$>. ويمكن تخزين المراجع في متغيرات أو إعادتها من خلال وظيفة ما، أي أنها تعابير يعطي تقييمها مراجع مكونات. ويستخدم مكدّس القيم لتخزين مراجع المكونات.

ويحدد المقطع <connect-op> تنفيذ العملية **connect** الموضحة في الشكل 60. وفي وصف مخطط الانسياب، يشير التعبير الأول الواجب تقييمه إلى <component-expression$_1$> والتعبير الثاني إلى <component-expression$_2$>، أي يكون التعبير <component-expression$_2$> على رأس مكدّس القيم عند تنفيذ العقدة connect-op.



الشكل Z.143/60 – المقطع <connect-op> في مخطط الانسياب

11.9 تحديد constant

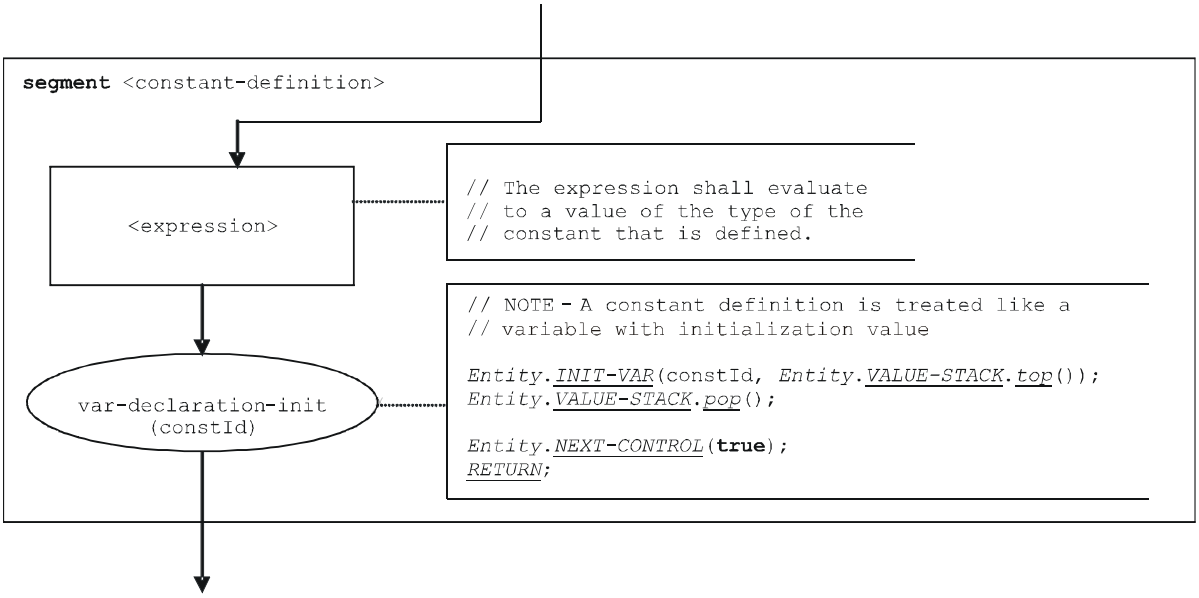
تكون بنية تحديد **constant** كما يلي:

```
const <constType> <constId> := <constType-expression>
```

تُعتبر قيمة ثابت ما تعبير يعطي تقييمه قيمة نمط الثابت.

ملاحظة – يستعاض عن الثوابت الإجمالية بقيمها في خطة معالجة مسبقة قبل تطبيق هذه الدلالة (انظر 2.9). وتعامل الثوابت المحلية بمثابة إعلانات متغيرة عند التدميث. وينبغي التحقق من صحة استخدام الثوابت لدى تحليل الدلالة السكونية لوحدة TTCN-3، علماً بأن الثوابت يجب ألا تكون في الجزء الأيسر من التخصيص.

ويحدد المقطع <constant-definition> في الشكل 61 تنفيذ إعلان ثابت ما حيث تعطى قيمة الثابتة في شكل تعبير.



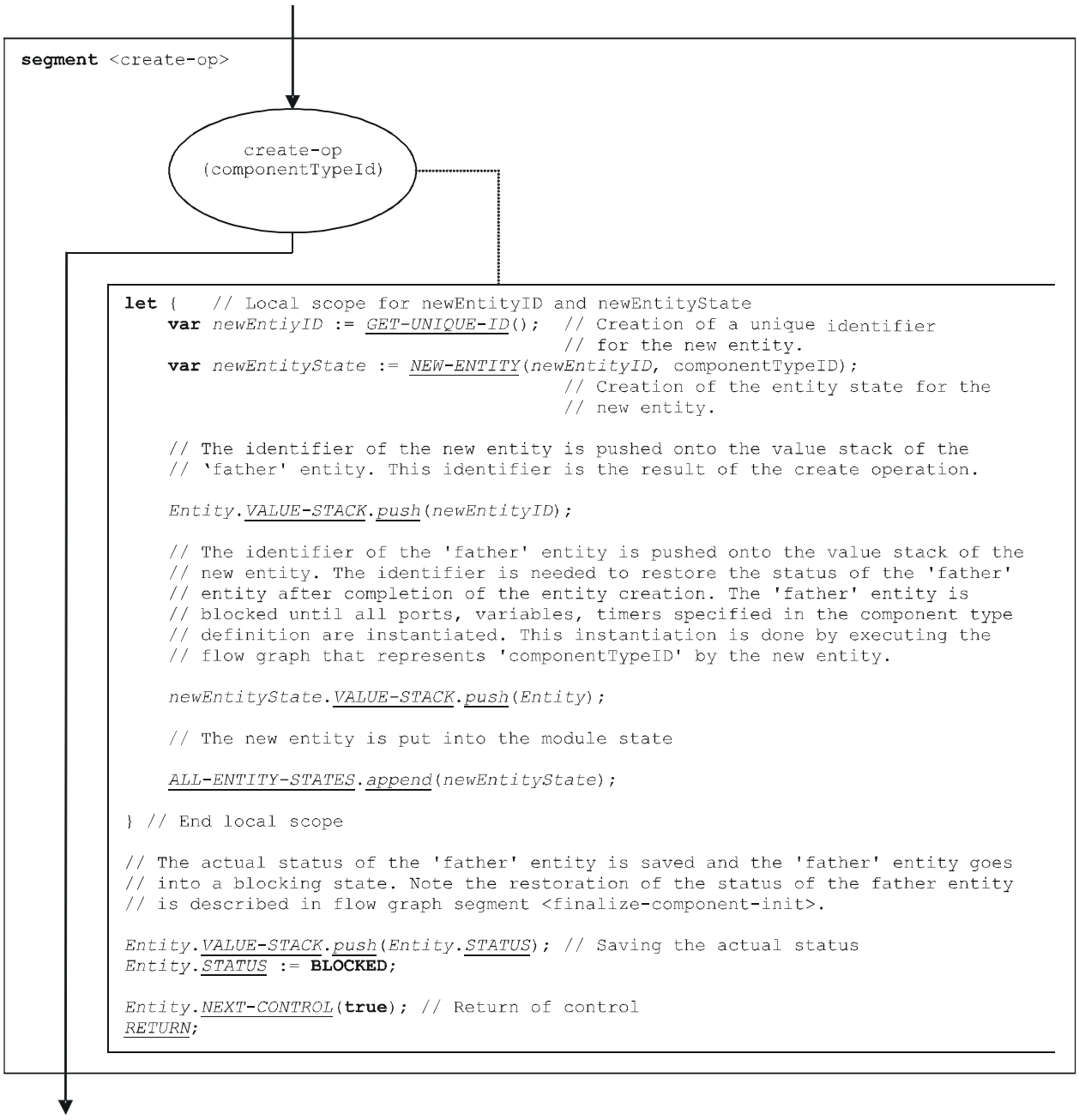
الشكل Z.143/61 – المقطع <CONSTANT-DEFINITION> في مخطط الانسياب

12.9 العملية create

تكون بنية العملية **create** كما يلي:

```
<componentTypeId>.create
```

ويحدد المقطع <create-op> في الشكل 62 تنفيذ العملية **create**.



الشكل Z.143/62 – المقطع <create-op> في مخطط الانسياب

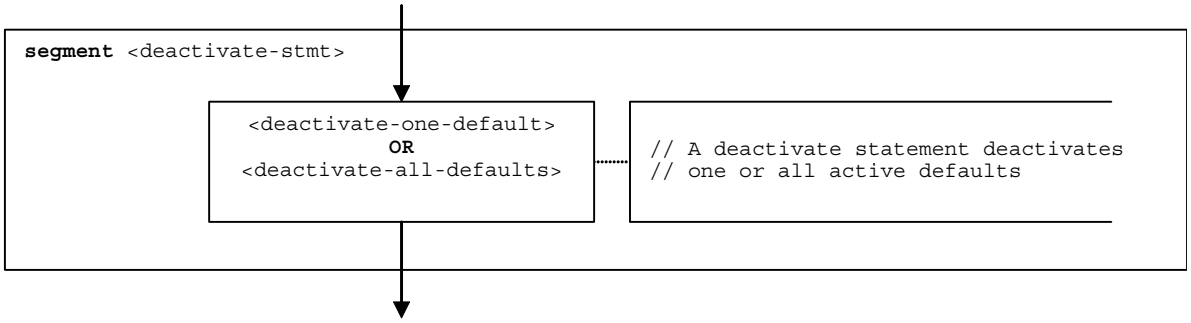
13.9 البيان deactivate

تكون بنية البيان **deactivate** كما يلي:

```
deactivate [(<default-expression>)]
```

ويحدد البيان **deactivate** وقف تنشيط إحدى أو كل حالات التغيب النشيطة للكيان الذي ينفذ البيان **deactivate**. فإذا توقف تنشيط سلوك واحد بالتغيب، عندئذ يجب أن يعطي تقييم التعبير الاختياري <default-expression> مرجعاً يحدد حالة التغيب الواجب إيقاف نشاطها. واستدعاء البيان **deactivate** دون تعبير <**default-expression**> يوقف نشاط كل حالات التغيب النشيطة.

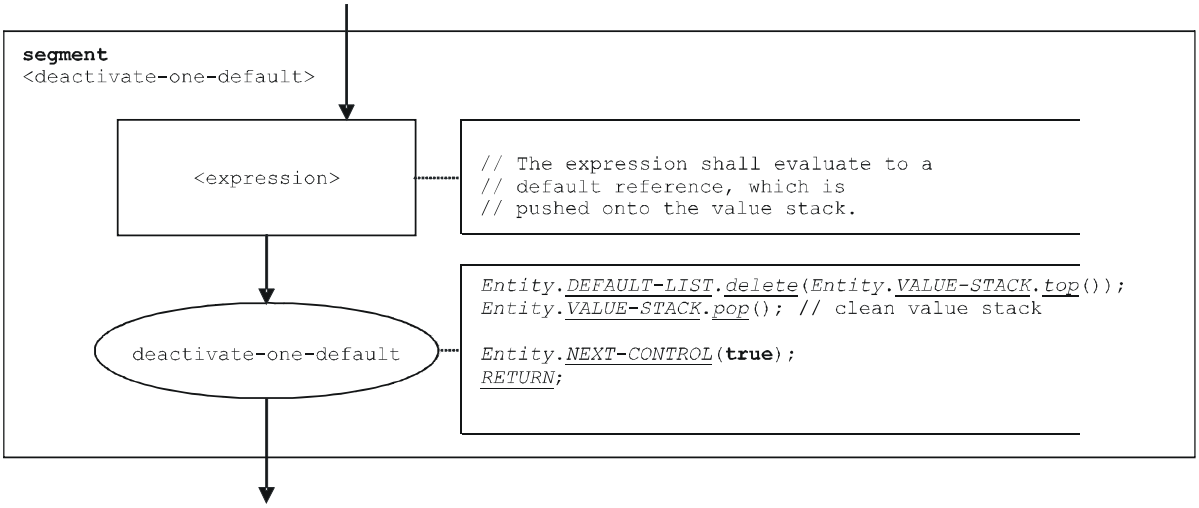
ويحدد المقطع <deactivate-stmt> في الشكل 63-a تنفيذ البيان **deactivate**.



الشكل Z.143/63-a – المقطع <deactivate-stmt> في مخطط الانسياب

1.13.9 المقطع <deactivate-one-default> في مخطط الانسياب

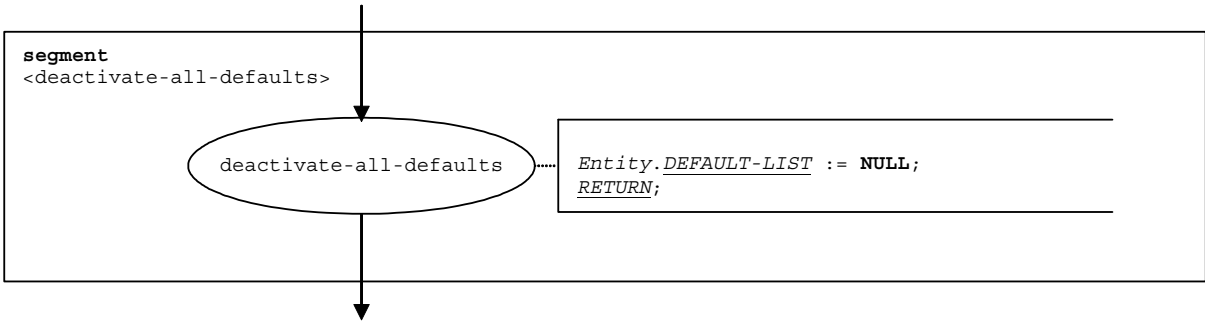
ويحدد المقطع <deactivate-one-default> في الشكل 63-b وقف تنشيط حالة تغيب نشيطة واحدة. ويجب أن يكون التعبير <default-expression> بحيث يعطي تقييمه سلوك مرجع بالتغيب. وقد يكون التعبير في شكل قيمة متغيرة أو وظيفة تعيد قيمة ما. ويزيل البيان **deactivate** حالة التغيب المحددة في القائمة *DEFAULT-LIST* للكيان الذي ينفذ البيان **deactivate**.



الشكل Z.143/63-b – المقطع <deactivate-one-default> في مخطط الانسياب

2.13.9 المقطع <deactivate-all-defaults>

يحدد المقطع <deactivate-all-defaults> في الشكل 63-c وقف تنشيط كل حالات التغيب النشيطة. ويزيل البيان **deactivate** محتوى القائمة *DEFAULT-LIST* للكيان الذي ينفذ البيان **deactivate**.



الشكل Z.143/63-c – المقطع <deactivate-all-defaults> في مخطط الانسياب

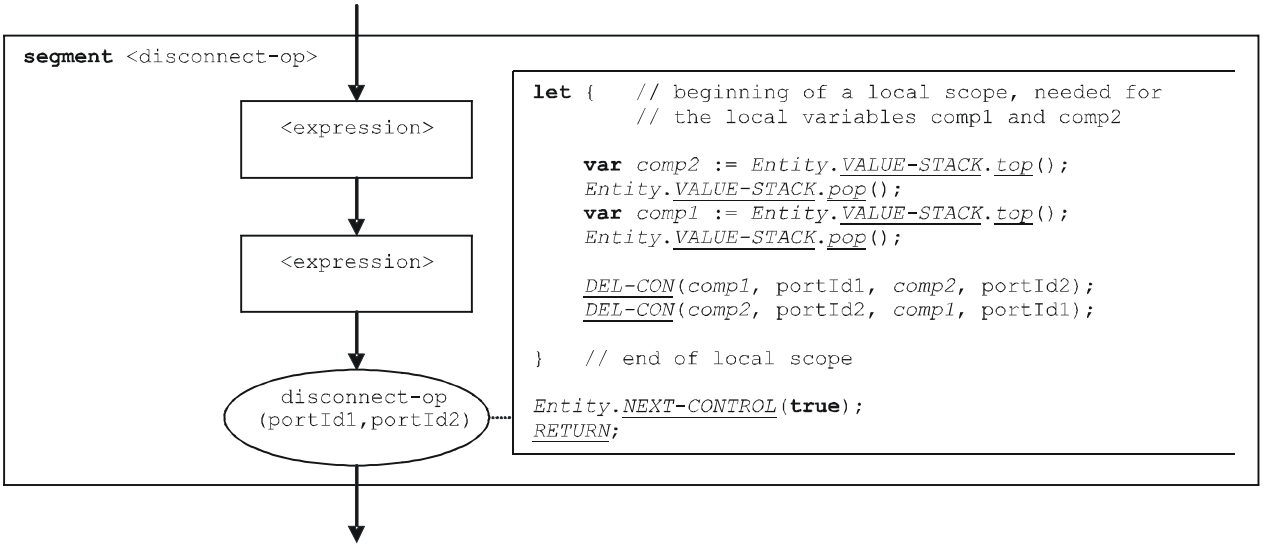
14.9 العملية disconnect

تكون بنية العملية **disconnect** كما يلي:

```
disconnect(<component-expression1>:<portId1>,
    <component-expression2>:<portId2>)
```

يعتبر كل من المعرفين <portId1> و<portId2> معرفاً لمنفذ مكونة الاختبار. ويتم الإحالة إلى المكونة التي ينتمي إليها المنفذ بواسطة مرجع المكونة <component-expression1> و<component-expression2>. ويمكن تخزين المراجع في متغيرات أو إعادتها من خلال وظيفة ما، أي أنها تعابير يعطي تقييمها مراجع مكونات. ويستخدم مكدّس القيم لتخزين مراجع المكونات.

ويحدد المقطع <disconnect-op> في الشكل 64 تنفيذ العملية **disconnect**. وفي مقطع مخطط الانسياب يشير التعبير الأول الواجب تقييمه إلى <component-expression1> والتعبير الثاني إلى <component-expression2>، أي أن التعبير <component-expression2> يكون على رأس مكدّس القيم عند تنفيذ العقدة disconnect-op.



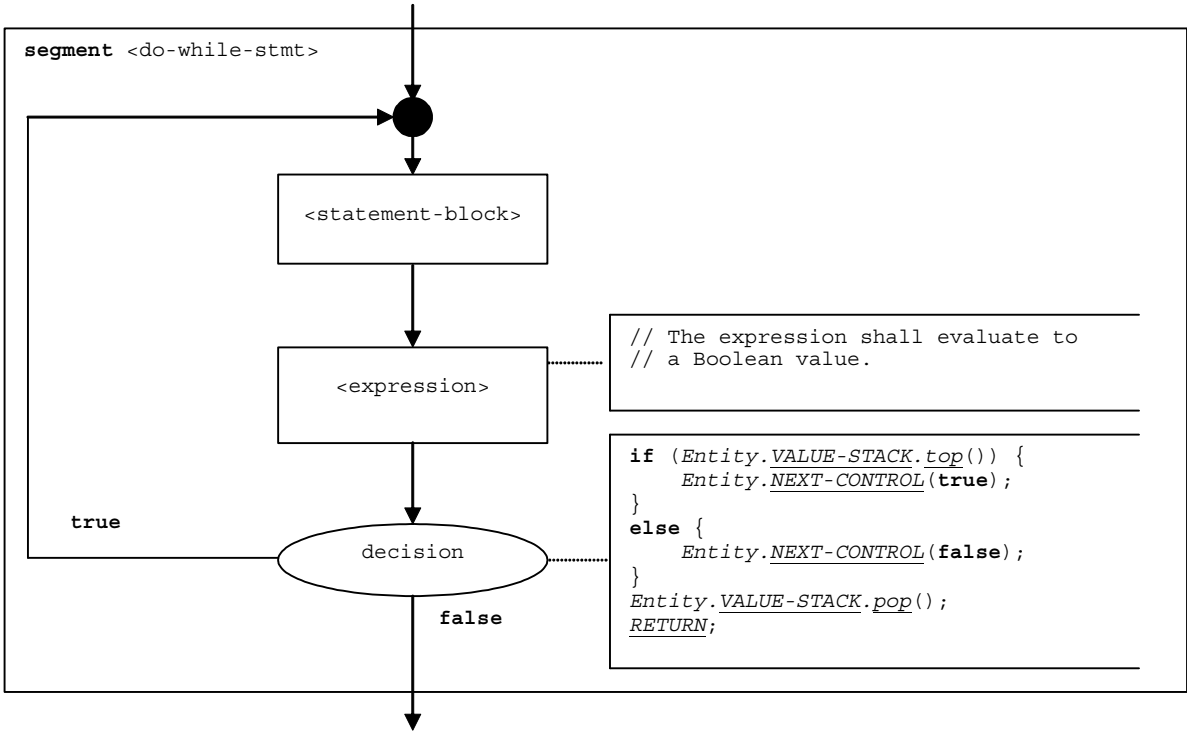
الشكل Z.143/64 – المقطع <disconnect-op> في مخطط الانسياب

15.9 البيان do-while

تكون بنية البيان **do-while** كما يلي:

```
do <statement-block>
while (<boolean-expression>)
```

ويحدد المقطع <do-while-stmt> في الشكل 65 تنفيذ البيان **do-while**.



الشكل Z.143/65 – المقطع <do-while-stmt> في مخطط الانسياب

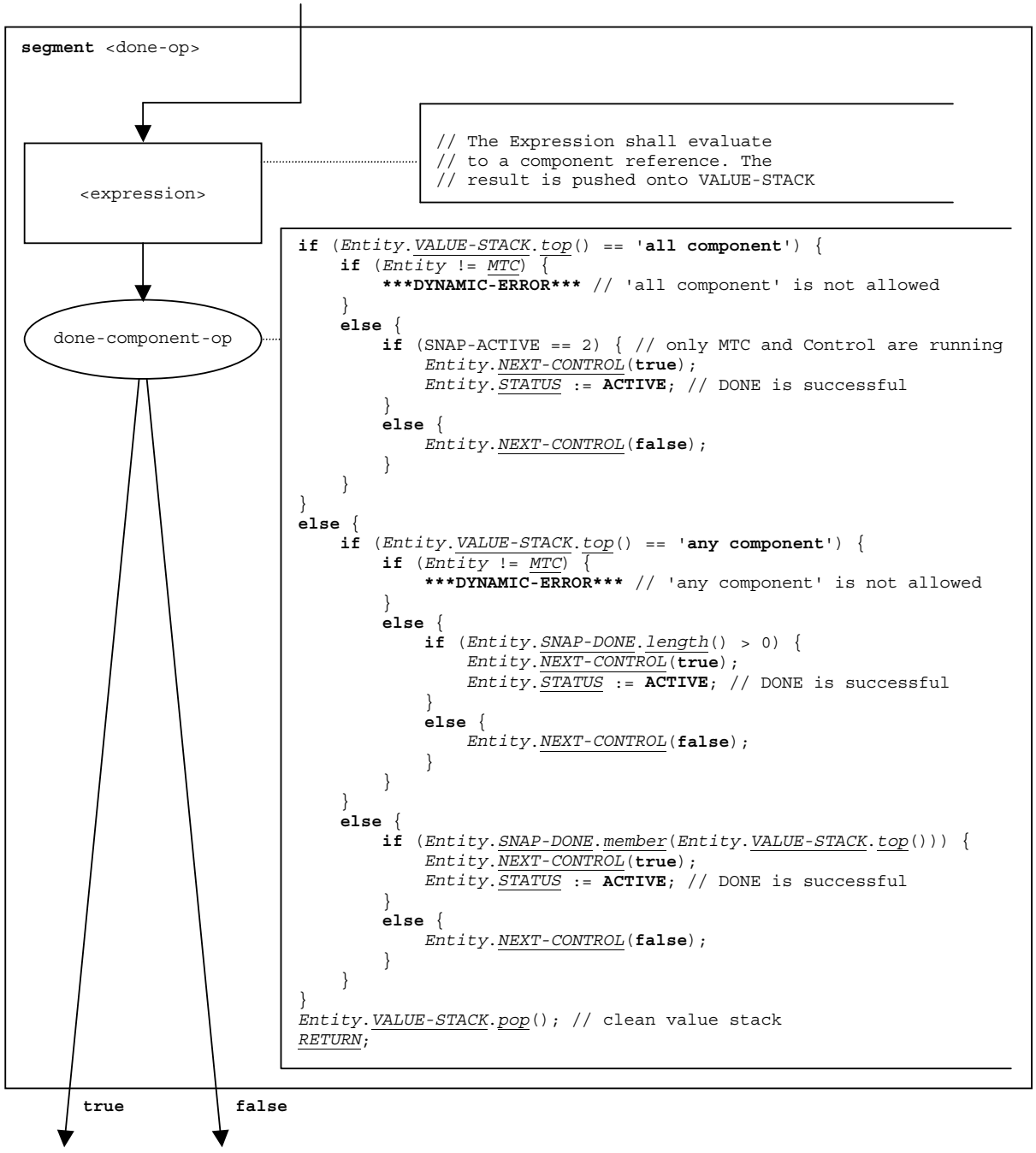
16.9 العملية done المطبقة على المكونات

تكون بنية العملية **done** المطبقة على المكونات كما يلي:

```
<component-expression>.done
```

تتحقق العملية **done** المطبقة على المكونات مما إذا كانت مكونة ما قيد التشغيل أو أنها توقفت. واستناداً إلى ما إذا كانت المكونة قيد التشغيل أم متوقفة عن العمل، تقرر العملية **done** كيف تستمر عمليات التحكم. ويحدد استخدام مرجع المكونة تلك المكونة الواجب التحقق منها. ويمكن تخزين المرجع في متغير أو يمكن إعادته بواسطة وظيفة ما، أي إنه في شكل تعبير. وتوخياً للتبسيط، يُعتبر التعبيران **all component** و**any component** بمثابة تعبيرين خاصين.

ويحدد المقطع <done-op> في الشكل 66 تنفيذ العملية **done** المطبقة على المكونات.



الشكل Z.143/66 – المقطع <done-component-op> في مخطط الانسياب

17.9 البيان execute

تكون بنية البيان **execute** كما يلي:

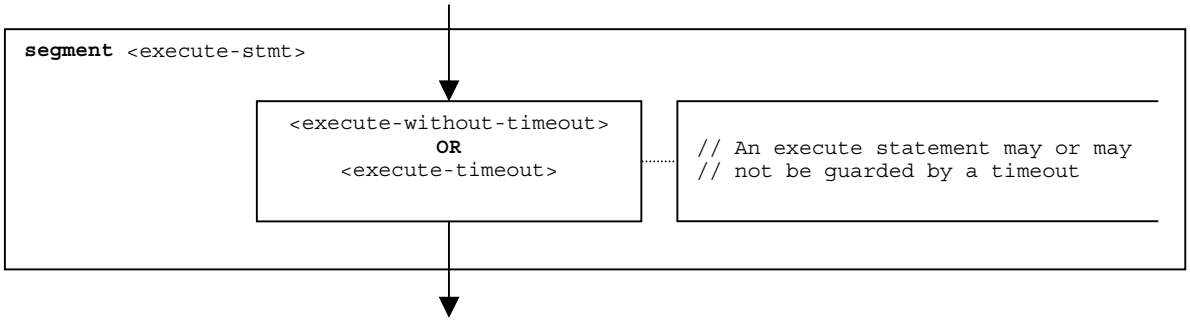
```
execute(<testCaseId>([<act-par1>, ... , <act-parn>)]) [, <float-expression>])
```

ويصف البيان **execute** تنفيذ اختبار الحالة <testCaseId> على أساس المعلمات الفعلية (الاختيارية) <act-par1>, ... , <act-parn>. ومن الممكن حماية هذا البيان اختيارياً بمدة معينة في شكل تعبير يفضي إلى قيمة من النمط **float**. فإذا لم يصدره خلال هذه المدة المحددة، عن اختبار الحالة أي قرار، يحدث استثناء لانتهاء الوقت ويتوقف اختبار الحالة ويعاد القرار بمثابة **error**.

ملاحظة – تقوم الدلالة التشغيلية بنمذجة توقف اختبار الحالة بوقف مكونة الاختبار الرئيسية MTC. في الواقع، قد تكون آليات أخرى أكثر ملاءمة.

في حال عدم حدوث أي استثناء لانتهاء المؤقت، تستحدث المكونة MTC وتحتبس حالة التحكم (التي تمثل الجزء المتعلق بالتحكم في الوحدة TTCN-3) إلى انتهاء اختبار الحالة، ولمتابعة تنفيذ اختبار الحالة يعهد بانسياب التحكم إلى المكونة MTC. ويعاد انسياب التحكم إلى حالة التحكم عندما ينتهي عمل المكونة MTC.

ويحدد المقطع <execute-stmt> في الشكل 67 تنفيذ البيان **execute**.



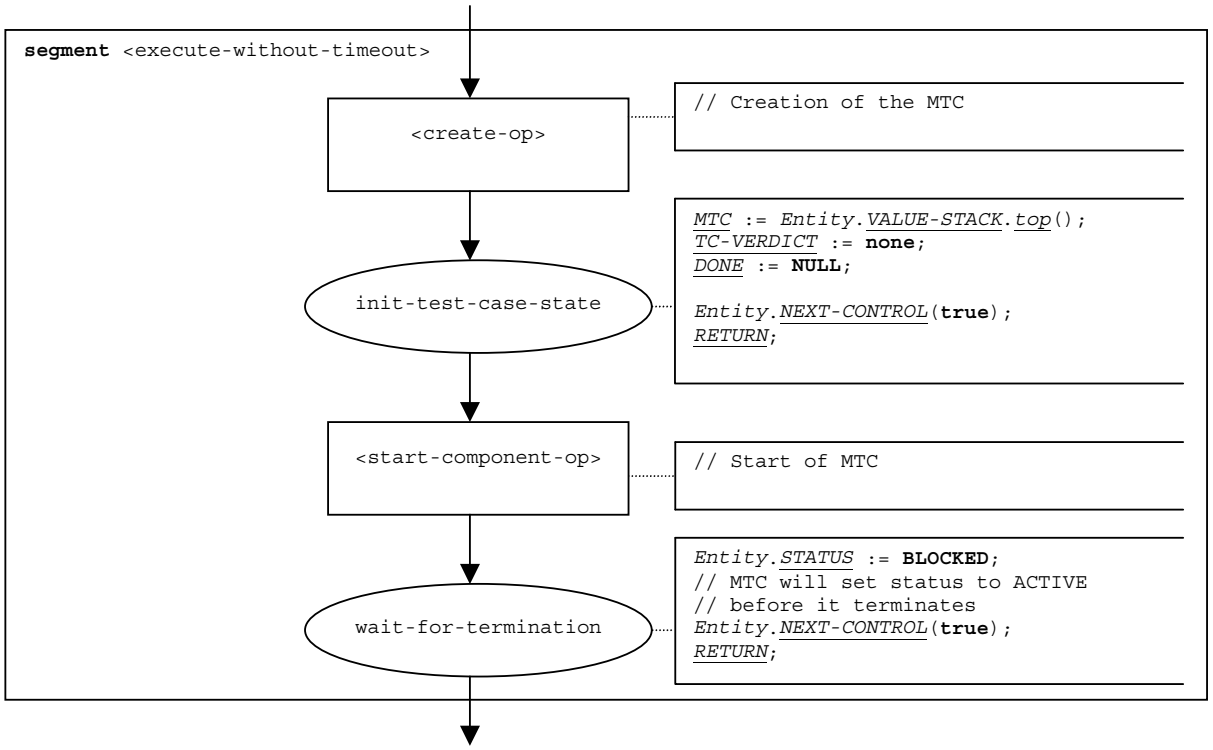
الشكل Z.143/67 – المقطع <execute-stmt> في مخطط الانسياب

1.17.9 المقطع <execute-without-timeout> في مخطط الانسياب

يبدأ تنفيذ اختبار حالة ما باستحداث المكونة MTC. ثم تطلق المكونة بالسلوك المحدد في تحديد اختبار الحالة. وبعدئذٍ ينتظر التحكم في الوحدة إلى أن ينتهي اختبار الحالة. ويمكن وصف استحداث وإطلاق المكونة MTC بواسطة البيانين **create** و**start**:

```
var mtcType MyMTC := mtcType.create;
MyMTC.start(TestCaseName(P1...Pn));
```

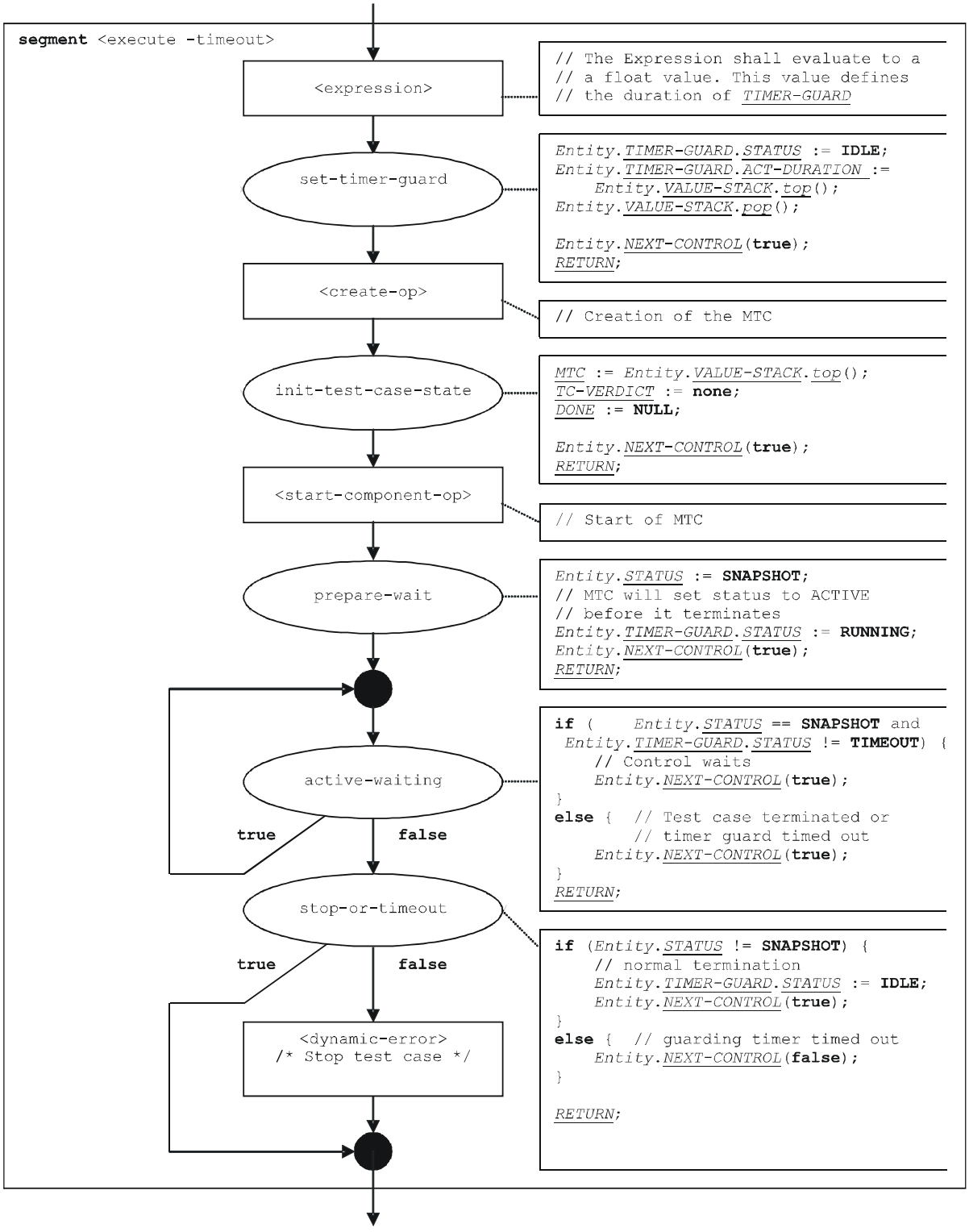
ويحدد المقطع <execute-without-timeout> في الشكل 68 تنفيذ البيان **execute** دون حدوث استثناء لانتهاء الوقت وذلك، بواسطة مقطع مخطط الانسياب للعمليتين **create** و**start**.



الشكل Z.143/68 – المقطع <execute-without-timeout> في مخطط الانسياب

2.17.9 المقطع <execute-timeout> في مخطط الانسياب

يحدد المقطع <execute-timeout> في الشكل 69 تنفيذ البيان **execute** المحمي بقيمة لانتهاء الوقت. ويقوم المقطع أيضاً بنمذجة استحداث وإطلاق المكونة MTC بواسطة العمليتين **create** و**start**. وعلاوة على ذلك تحمي *TIMER-GUARD* عملية الانتهاء.



الشكل Z.143/69 – المقطع <execute-timeout> في مخطط الانسياب

18.9 التعبير

لمعالجة التعابير، يجب التمييز بين الحالات الأربع التالية:

أ) التعبير قيمة حرفية (أو ثابت)؛

ب) التعبير متغير؛

ج) التعبير معامل ينطبق على معمول واحد أو أكثر؛

د) التعبير استدعاء وظيفة أو عملية.

وتكون بنية التعبير كما يلي:

```
<lit-val> | <var-val> | <func-op-call> | <operand-appl>
```

حيث:

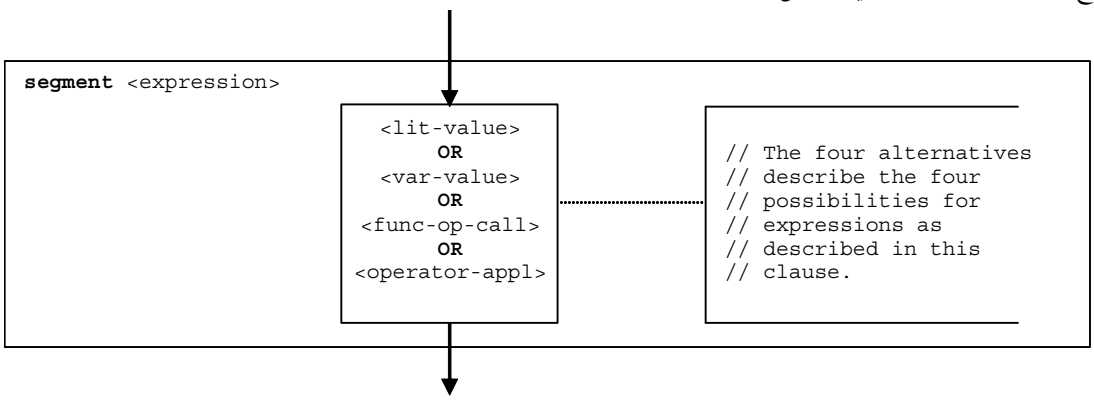
<lit-val> تشير إلى قيمة حرفية؛

<var-val> تشير إلى قيمة متغيرة؛

<func-op-call> تشير إلى استدعاء وظيفة أو عملية؛

<operator-appl> تشير إلى تطبيق معاملات رياضية مثل + و– و**not**، وغيرها.

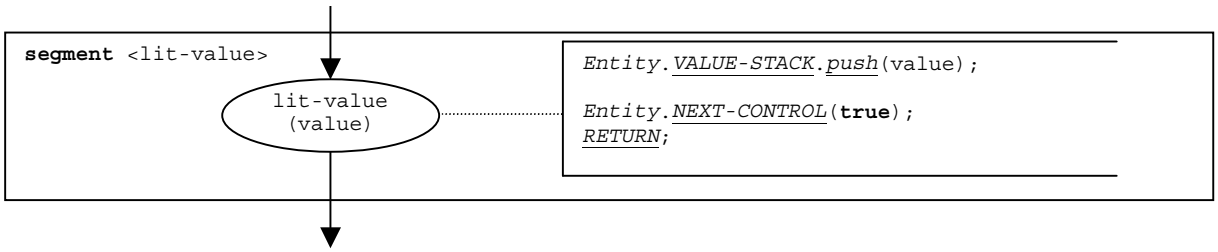
ويحدد المقطع <expression> في الشكل 70 تنفيذ التعبير.



الشكل Z.143/70 – المقطع <expression> في مخطط الانسياب

1.18.9 المقطع <lit-value> في مخطط الانسياب

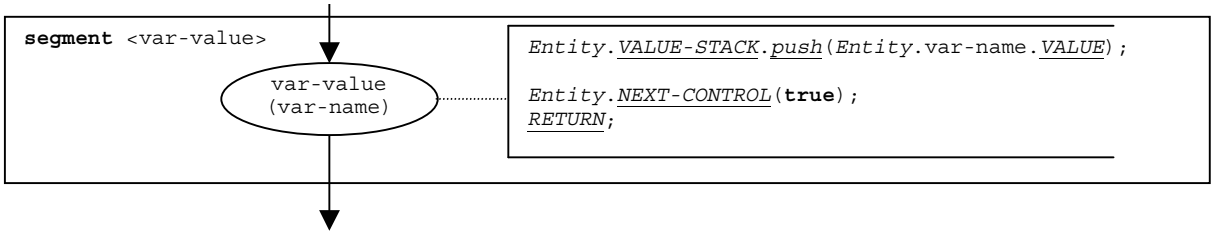
يضع المقطع <lit-value> في الشكل 71 قيمة حرفية في مكدّس قيم كيانٍ ما.



الشكل Z.143/71 – المقطع <lit-value> في مخطط الانسياب

2.18.9 المقطع <var-value> في مخطط الانسياب

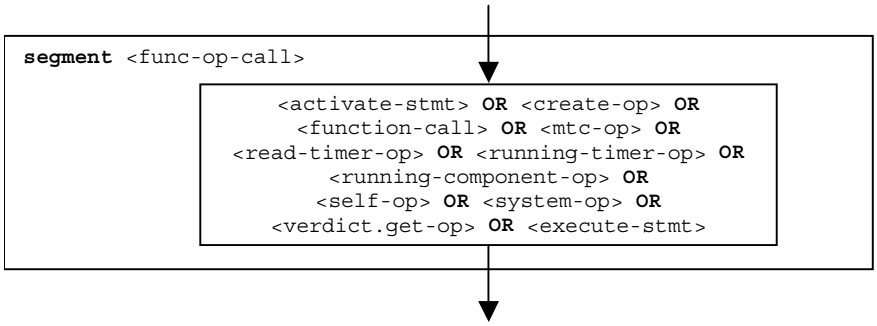
يضع المقطع <var-value> في الشكل 72 قيمة متغير في مكدّس قيم كيانٍ ما.



الشكل Z.143/72 – المقطع <var-value> في مخطط الانسياب

3.18.9 المقطع <func-op-call> في مخطط الانسياب

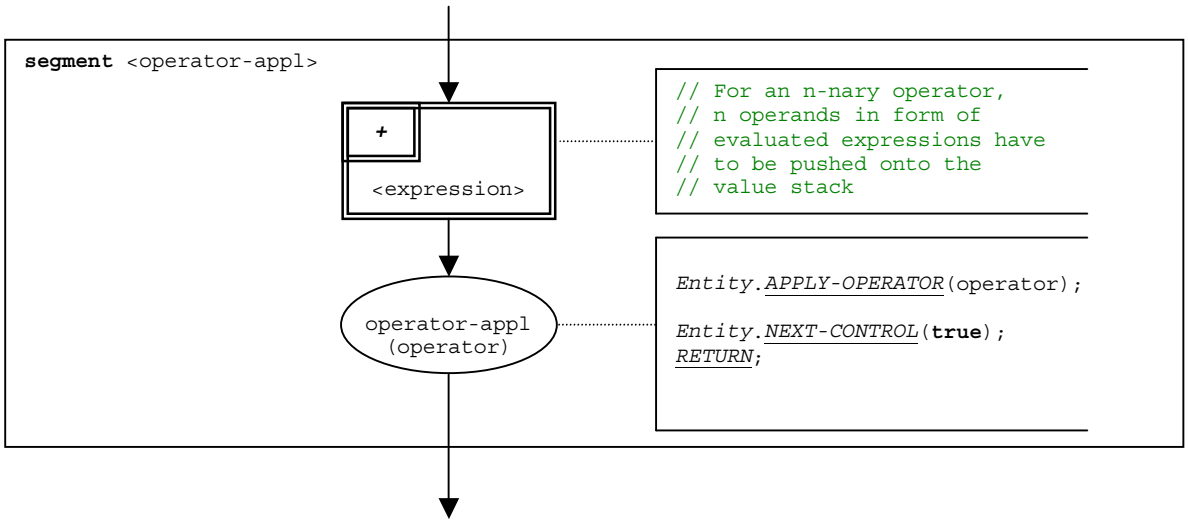
يشير المقطع <func-op-call> في الشكل 73 إلى استدعاءات وظائف أو عمليات تعيد قيمة موضوعة في مكدّس قيم كيانٍ ما. وتُعتبر كل هذه الاستدعاءات بمثابة تعابير.



الشكل Z.143/73 – المقطع <func-op-call> في مخطط الانسياب

4.18.9 المقطع <operator-appl> في مخطط الانسياب

يستند تمثيل مخطط الانسياب في الشكل 74 مباشرة إلى افتراض استخدام الترميز البولوني المعكوس لتقييم التعابير التي تنطوي على معامل. وتحسب نواتج المعامل وتوضع في مكدّس التقييم. وبالنسبة لتطبيق المعامل تزال النواتج من مكدّس التقييم ويطبق المعامل. ثم توضع نتيجة تطبيق المعامل في مكدّس التقييم. وتعتبر عمليتا إزالة النواتج ووضع النتيجة في المكدّس جزءاً من تطبيق المعامل (البيان Entity.APPLY-OPERATOR(operator) في الشكل 74)، أي أنهما ليستا منمذجتين بحسب الدلالة التشغيلية.

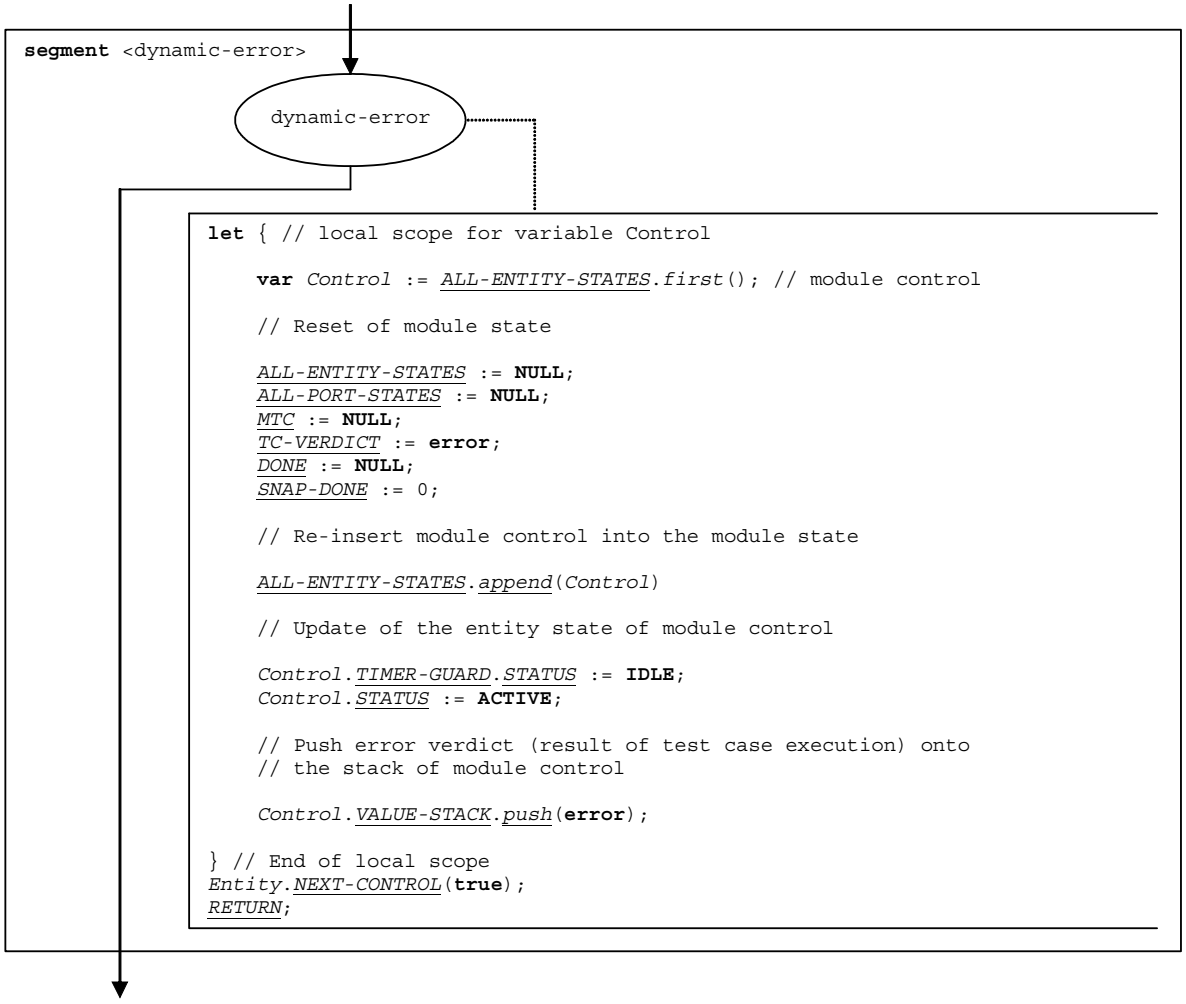


الشكل Z.143/74 – المقطع <operator-appl> في مخطط الانسياب

18b.9 المقطع <dynamic-error> في مخطط الانسياب

في حال حدوث خطأ دينامي، يستدعي نظام الاختبار المقطع <dynamic-error> (انظر الشكل b-74). وتزال جميع الموارد المخصصة لاختبار الحالة وينسب القرار **error** لاختبار الحالة. ويعطى التحكم إلى البيان في جزء التحكم الذي يلي البيان execute حيث حدث الخطأ.

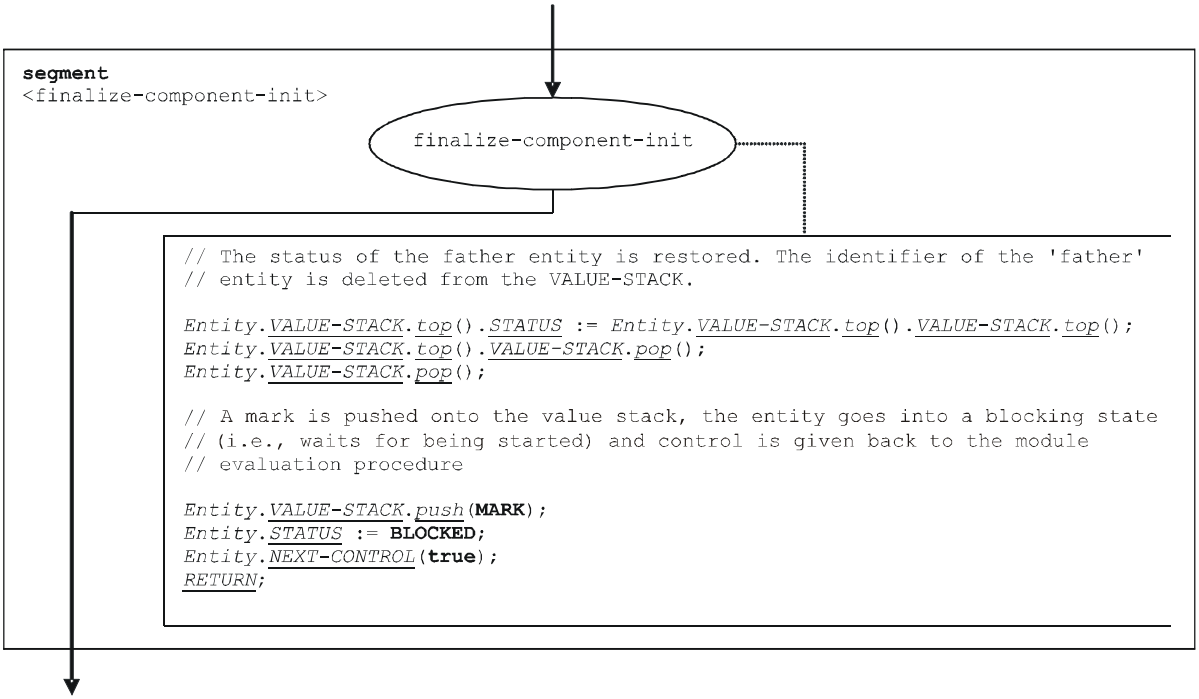
ويستدعي التحكم في الوحدة المقطع <dynamic-error> إذا لم ينته اختبار حالة ما في المهلة الزمنية المحددة (انظر الفقرة 2.17.9).



الشكل Z.143/74-b – المقطع <dynamic-error> في مخطط الانسياب

19.9 المقطع <finalize-component-init> في مخطط الانسياب

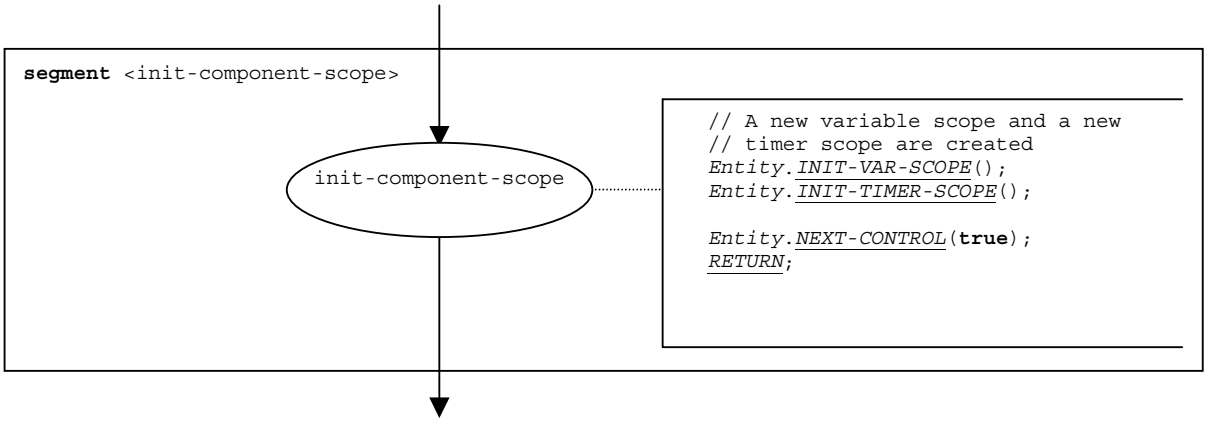
المقطع <finalize-component-init> جزء من مخطط الانسياب الذي يمثل سلوك تحديد نمط المكونة. ويرد وصف تطبيقه في الشكل 75.



الشكل Z.143/75 – المقطع <finalize-component-init> في مخطط الانسياب

20.9 المقطع <init-component-scope> في مخطط الانسياب

المقطع <init-component-scope> جزء من مخطط الانسياب الذي يمثل سلوك تحديد نمط المكونة. ويرد وصف تطبيقه في الشكل 76.

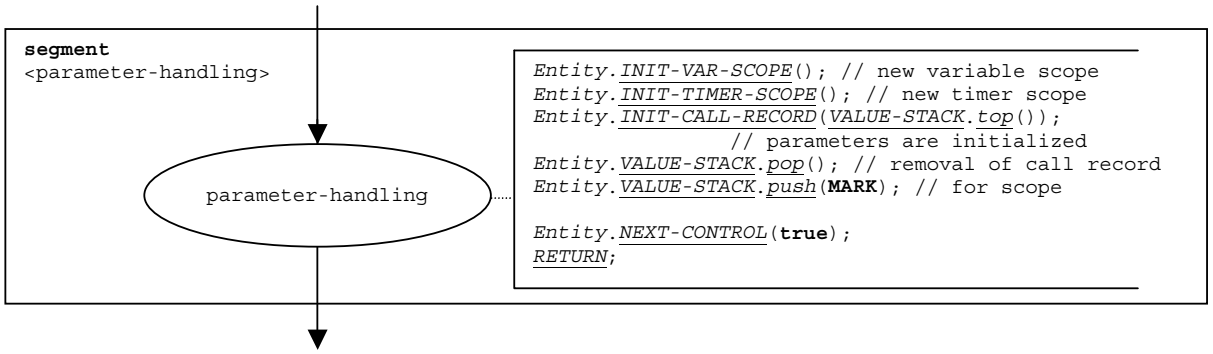


الشكل Z.143/76 – المقطع <init-component-scope> في مخطط الانسياب

21.9 المقطع <parameter-handling> في مخطط الانسياب

يستخدم المقطع <parameter-handling> في بداية مخططات الانسياب التي تمثل حالات الاختبار والخطوات البديلة والوظائف. وهي تدمث مجالاً جديداً وتولّد متغيرات ومؤقتات لمعالجة المعلمات. وفي المقطع <parameter-handling> نفترض أن سجل نداء حالة الاختبار أو الخطوة البديلة أو الوظيفة المطلوبة توجد في أعلى مكدّس القيم.

يرد في الشكل 77 تنفيذ المقطع <parameter-handling>.



الشكل Z.143/77 – المقطع <parameter-handling> في مخطط الانسياب

22.9 المقطع <statement-block> في مخطط الانسياب

تكون بنية تركيب فدرة البيان كما يلي:

{ <statement$_1$>; ... ; <statement$_n$> }

إن فدرة البيان هي وحدة نطاق. عند دخول وحدة نطاق، ينبغي تدميث نطاقات جديدة للمتغيرات والمؤقتات ومكدّس القيم. وعند الخروج من وحدة نطاق، ينبغي إبطال جميع المتغيرات والمؤقتات وقيم المكدس في هذا النطاق.

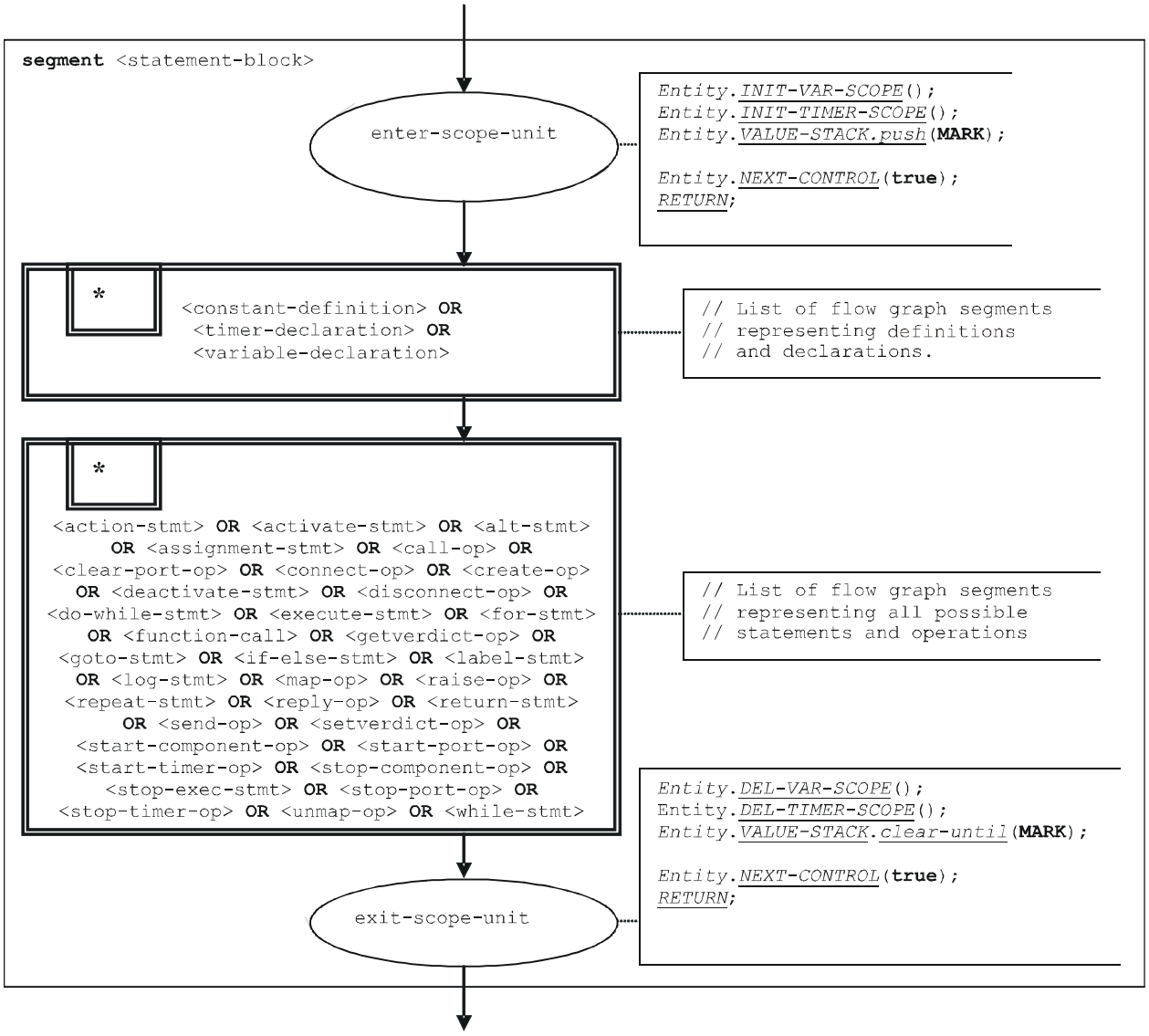
الملاحظة 1 – فدرة البيان ليست مفهوماً رسمياً للترميز TTCN-3. ولا تكون فدرات البيانات إلا في متن الوظائف والخطوات البديلة وحالات الاختبار والتحكم بالنموذج وضمن البيانات المركبة، مثل **alt** أو **if-else** أو **do-while**.

الملاحظة 2 – لا يمكن لعمليات الاستقبال ونداءات الخطوة البديلة أن تظهر في فدرات البيانات وهي مبيتة في بيانات **alt** أو في العمليات **call**.

الملاحظة 3 – تعالج الدلالة التشغيلية أيضاً العمليات والإعلانات في شكل بيانات، أي يمكن ورودها في فدرات البيانات.

الملاحظة 4 – بعض وظائف TTCN-3، مثلاً **system** أو **self**، تُعتبر بمثابة عبارات وهي ليست مفيدة كبيانات مستقلة في فدرات البيانات. وهي غير ممثلة بيانياً في مخطط الانسياب في الشكل 78.

يحدد المقطع <statement-block> في الشكل 78 عملية تنفيذ فدرة بيانات.



الشكل Z.143/78 – المقطع <statement-block> في مخطط الانسياب

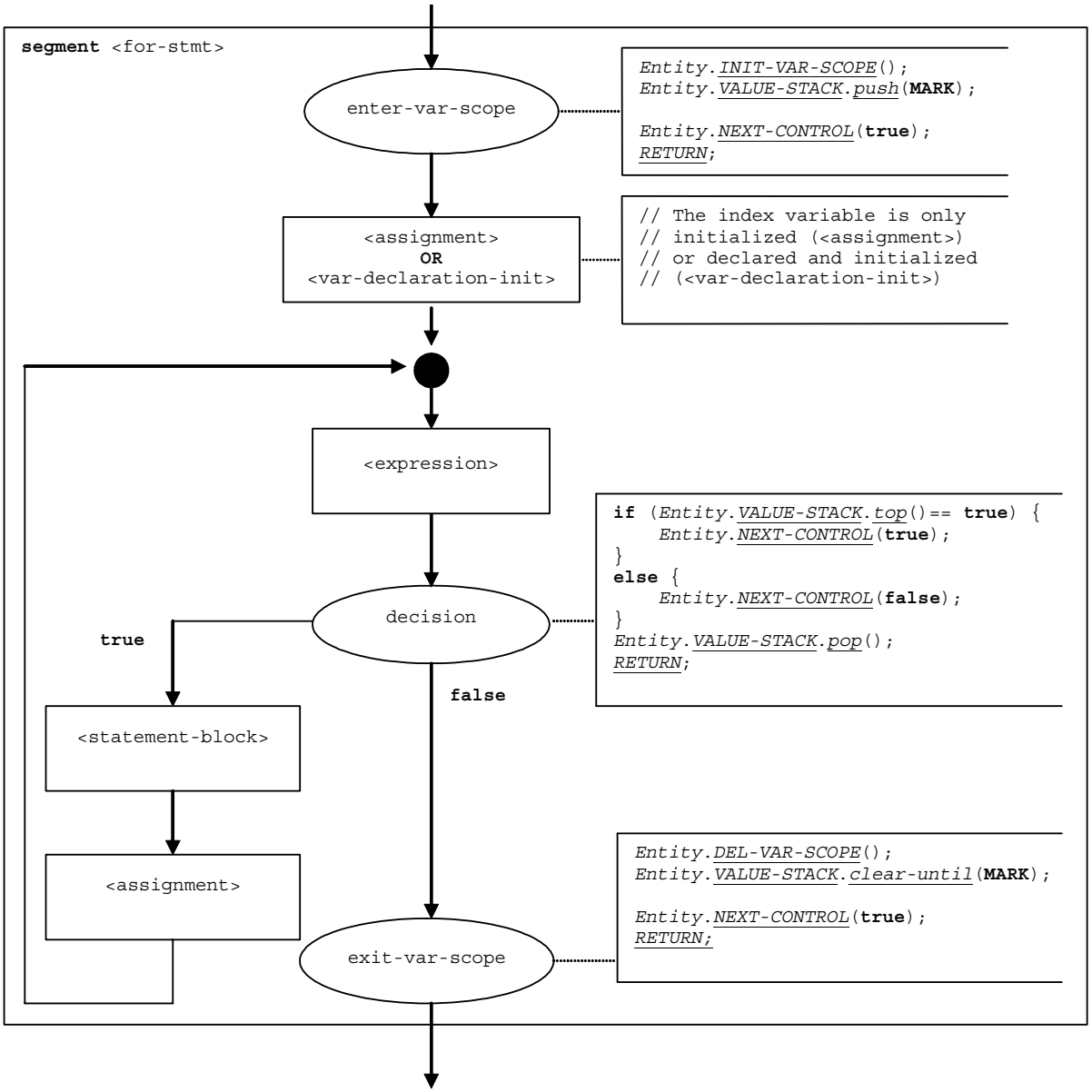
23.9 البيان for

تركيب البيان **for-statement** كما يلي:

```
for (<assignment>|<variable-declaration>, <boolean expression>, <assignment>) <statement-block>
```

إن تدميث متغيرة المؤشر والمعالجة المقابلة له يُعتبران بمثابة تخصيصات لمتغيرة المؤشر. ومن الممكن أيضاً إعلان وتدميث متغيرة المؤشر مباشرة في **for-statement**. وتصف العبارة <boolean-expression> معيار انتهائية العروة التي تحددها **for-statement** وتصف الفدرة <statement-block> مضمون العروة.

يحدد المقطع <for-stmt> في الشكل 79 تنفيذ **for-statement**. ويصف التخصيص الأولي <assignment> أو إعلان المتغيرة البديلة مع التخصيص <var-declaration-init> (انظر الفقرة 1.57.9) تدميث متغيرة المؤشر. ويصف التخصيص <assignment> في الفرع **true** من العقدة decision التعامل بمتغيرة المؤشر. و**for-statement** عبارة عن وحدة نطاق لمتغيرة مؤشر معلنة حديثاً، وذلك منمذج بواسطة العقدتين enter-var-scope وexit-var-scope.



الشكل Z.143/79 – المقطع <for-stmt> في مخطط الانسياب

24.9 النداء الوظيفي

تركيب النداء الوظيفي هو:

<function-name>([<act-par-desc$_1$>, ... , <act-par-desc$_n$>])

الاسم <function-name> هو اسم الوظيفة كما أن <act-par-descr$_1$>, ... , <act-par-descr$_n$> تصف قيم المعلمات الفعلية للنداء الوظيفي.

الملاحظة 1 – يعالَج النداء الوظيفي ونداء الخطوة البديلة بالطريقة نفسها. وبالتالي، يحيل نداء الخطوة البديلة (انظر الفقرة 4.9) إلى الفقرة الحالية.

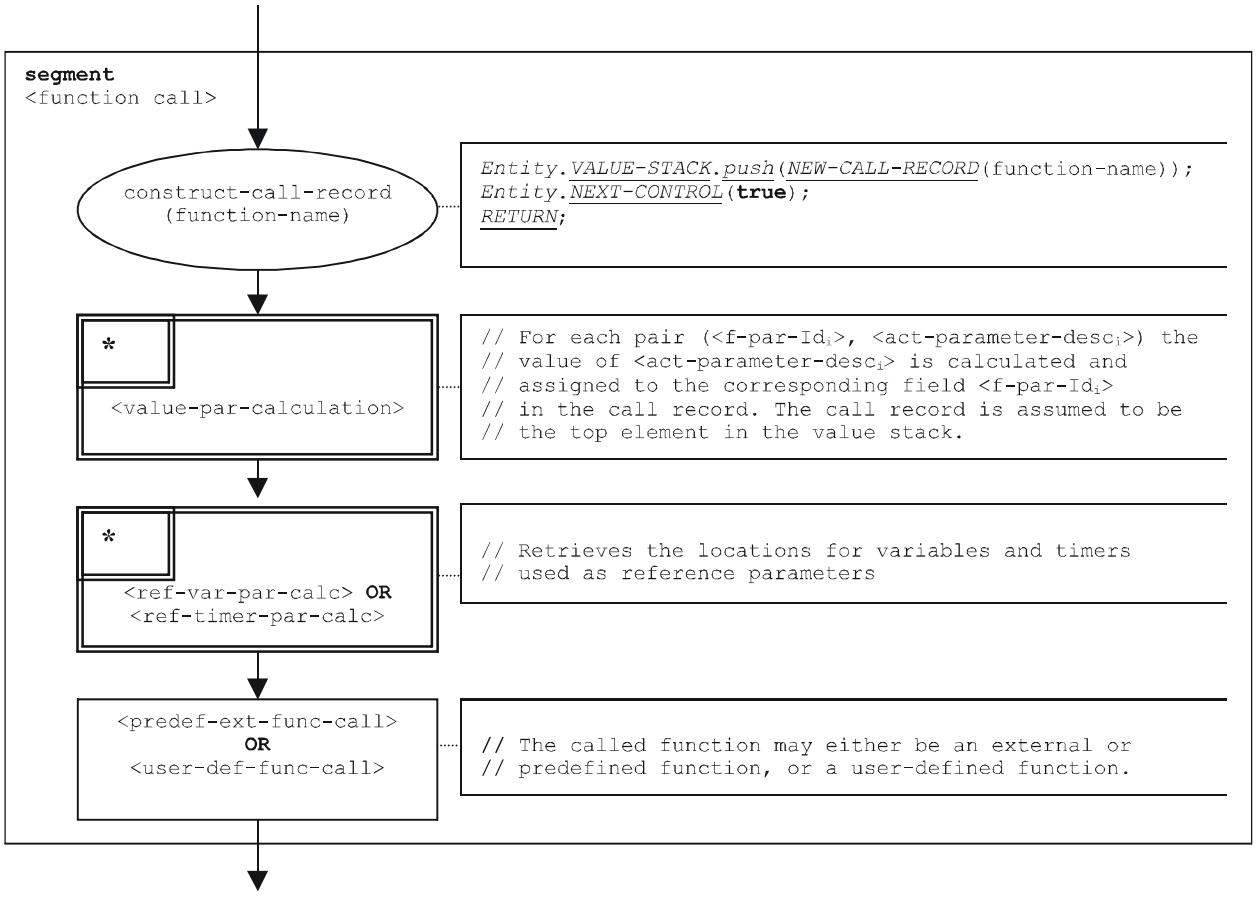
نفترض أنه لكل وصف <act-par-desc$_1$> يكون معرف المعلمة الرسمي المقابل <f-par-Id$_1$> معروفاً، أي من الممكن تمديد بنية التركيب أعلاه كما يلي:

<function-name>((<f-par-Id$_1$>,<act-par-desc$_1$>), ... , (<f-par-Id$_n$>,<act-par-desc$_n$>))

يحدد المقطع <function-call> في الشكل 80 تنفيذ نداء وظيفي. يقوم التنفيذ على ثلاث مراحل. في المرحلة الأولى يتم إنشاء سجل النداء الوظيفي <function-name>. في المرحلة الثانية يتم حساب قيم المعلمات الفعلية وتدرج في الحقل المقابل في سجل النداء. في المرحلة الثالثة، ينبغي التمييز بين حالتين: الوظيفة المطلوبة يحددها المستعمل (<user-def-func-call>)، أي أن هناك تمثيل بياني للوظيفة، أو الوظيفة المطلوبة وظيفة محددة مسبقاً أو خارجية (<predef-ext-func-call>). في حالة نداء وظيفي يحدده المستعمل، يعطى التحكم إلى الوظيفة المطلوبة. وفي حالة وظيفة محددة مسبقاً أو خارجية، نفترض أن سجل النداء يمكن استخدامه لتنفيذ الوظيفة في مرحلة واحدة. إن المعالجة الصحيحة للمعلمات المرجعية وقيمة العودة (التي يجب أن توضع في مكدس القيم) تقع على عاتق الوظيفة المطلوبة، أي أنها تخرج عن إطار الدلالة التشغيلية هذه.

الملاحظة 2 – إذا نمذج النداء الوظيفي نداء خطوة بديلة، يتم اختيار الفرع <user-def-func-call> فقط، نظراً لوجود تمثيل بياني للخطوة البديلة المستدعاة.

الملاحظة 3 – يستخدم المقطع <function call> كذلك لوصف إطلاق المكونة MTC في البيان **execute**. وفي هذه الحالة، يتم إنشاء سجل نداء للاختبار الأولي ويتم اختيار الفرع <user-def-func-call> فقط.



الشكل Z.143/80 – المقطع <function-call> في مخطط الانسياب

1.24.9 المقطع <value-par-calculation>

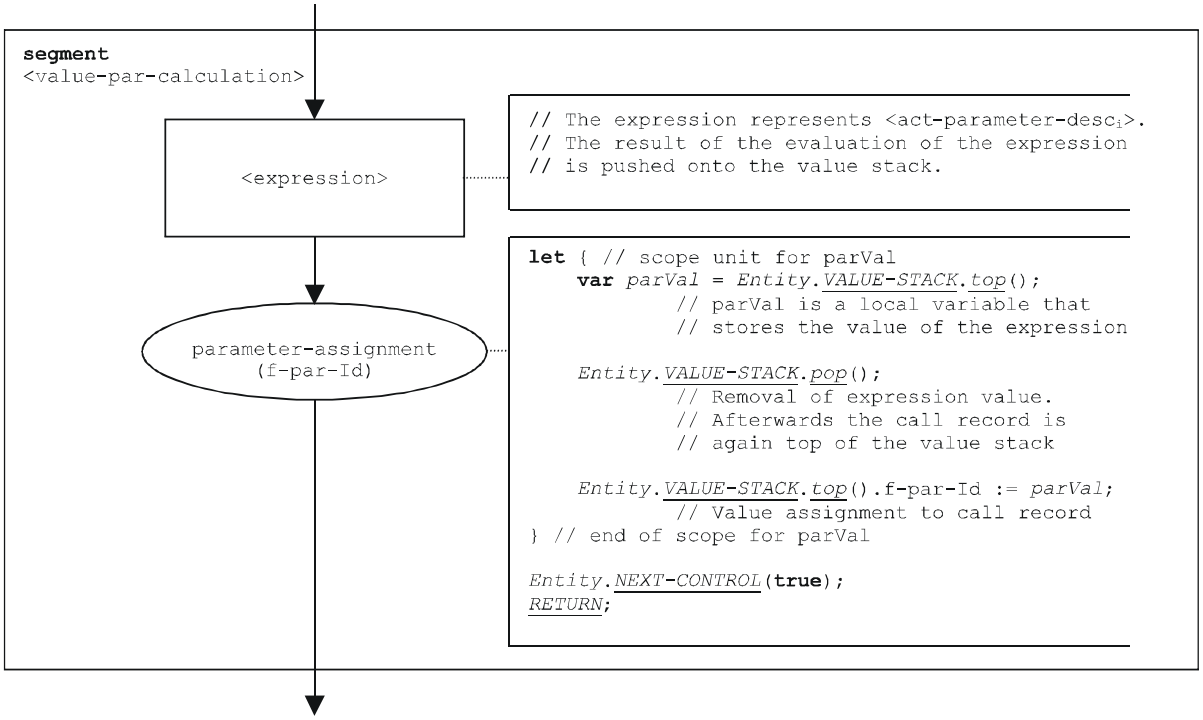
يساعد المقطع <value-par-calculation> على حساب قيم المعلمات الفعلية وإدراجها في الحقول المقابلة في سجلات النداء للوظائف والخطوات البديلة والاختبارات الأولية.

نفترض أن سجل النداء هو العنصر الأعلى في مكدّس القيم وأنه يجب معالجة الزوج:

(<f-par-Id$_i$>, <act-parameter-desc$_i$>)

وأن الوصف <act-parameter-desc$_i$> يجب تقييمه و<f-par-Id$_i$> هو معرف المعلمة الرسمية التي لها حقل مقابل في سجل النداء الوارد في مكدّس القيم.

يرد في الشكل 81 تنفيذ المقطع <value-par-calculation> في مخطط الانسياب.



الشكل Z.143/81 – المقطع <value-par-calculation> في مخطط الانسياب

2.24.9 المقطع <ref-par-var-calc>

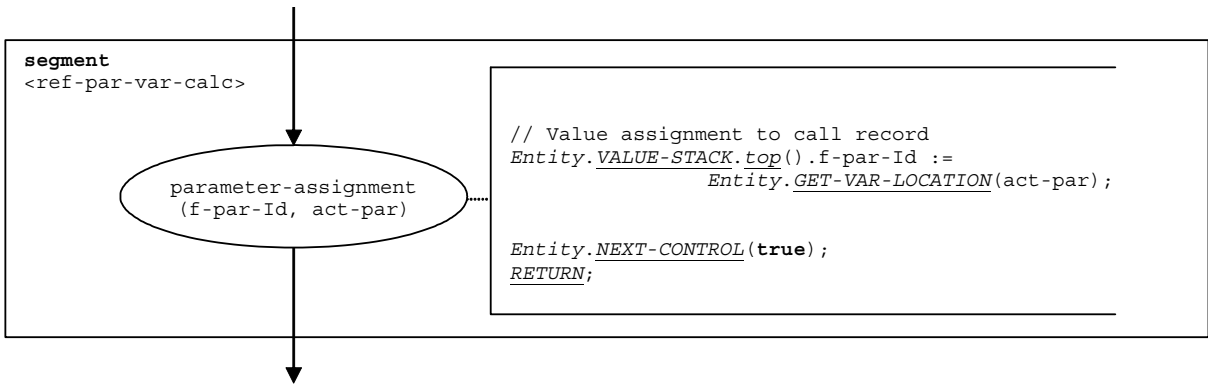
يساعد المقطع <ref-par-var-calc> على استخراج مواقع المتغيرات المستعملة كمعلمات مرجعية فعلية وعلى إدراجها في الحقول المقابلة في سجلات النداء للوظائف والخطوات البديلة والاختبارات الأولية.

ونفترض أن سجل النداء هو العنصر الأعلى في مكدّس القيم وأنه يجب معالجة الزوج:

(<f-par-Id$_i$>, <act-par$_i$>)

وتكون <act-par$_i$> المعلمة الفعلية التي يجب استخراج الموقع لها وتكون <f-par-Id$_i$> معرف معلمة رسمية لها حقل مقابل في سجل النداء الوارد في مكدّس القيم.

يرد في الشكل 82 تنفيذ المقطع <ref-par-var-calc> في مخطط الانسياب.



الشكل Z.143/82 – المقطع <ref-par-var-calc> في مخطط الانسياب

3.24.9 المقطع <ref-par-timer-calc>

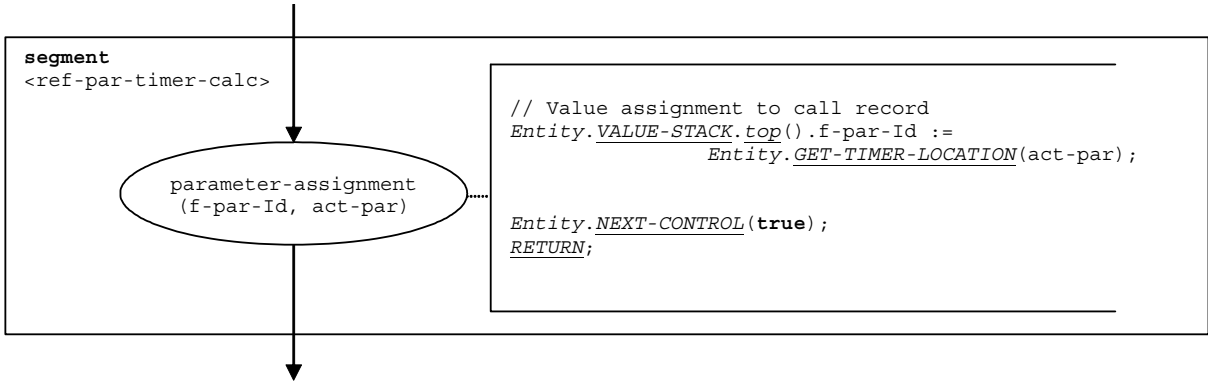
يساعد المقطع <ref-par-timer-calc> على استخراج مواقع المؤقتات المستعملة كمعلمات مرجعية فعلية وعلى إدراجها في الحقول المقابلة في سجلات النداء للوظائف والخطوات البديلة والاختبارات الأولية.

ونفترض أن سجل النداء هو العنصر الأعلى في مكدّس القيم وأنه يجب معالجة الزوج:

(<f-par-Id$_i$>, <act-par$_i$>)

وتكون <act-par$_i$> المعلمة الفعلية التي يجب استخراج الموقع لها وتكون <f-par-Id$_i$> معرف معلمة رسمية لها حقل مقابل في سجل النداء الوارد في مكدّس القيم.

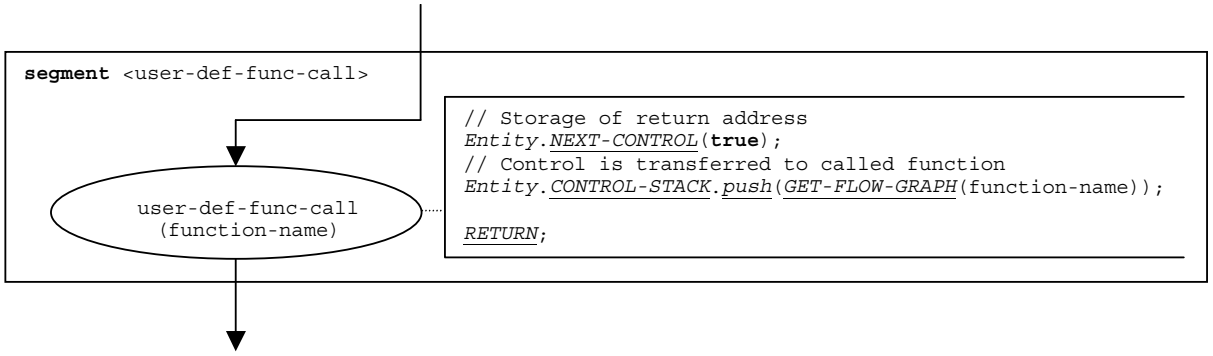
يرد في الشكل 83 تنفيذ المقطع <ref-par-timer-calc> في مخطط الانسياب.



الشكل Z.143/83 – المقطع <ref-par-timer-calc> في مخطط الانسياب

4.24.9 المقطع <user-def-func-call>

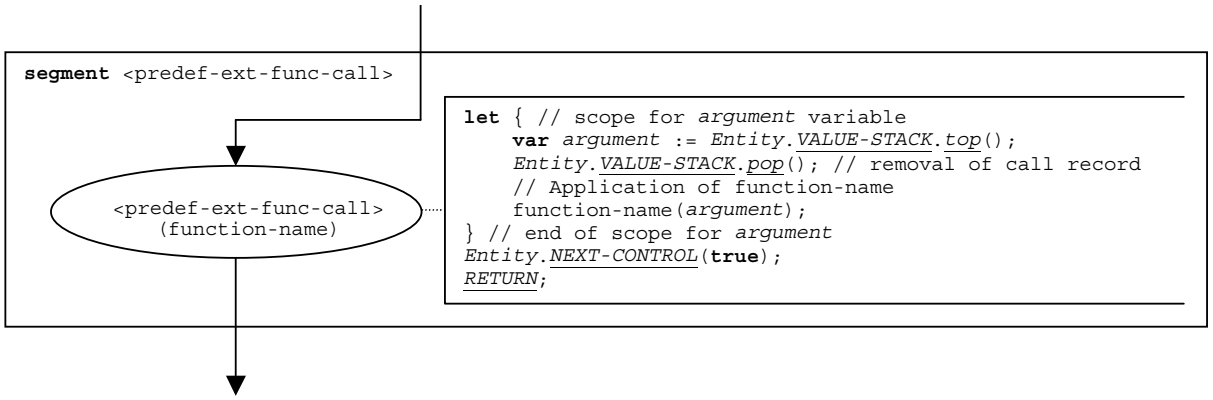
يصف المقطع <user-def-func-call> (الشكل 84) نقل التحكم إلى وظيفة مطلوبة يحددها المستعمل.



الشكل Z.143/84 – المقطع <user-def-func-call> في مخطط الانسياب

5.24.9 المقطع <predef-ext-func-call>

يصف المقطع <predef-ext-func-call> (الشكل 85) النداء الوظيفي المحدد سابقاً أو الخارجي.



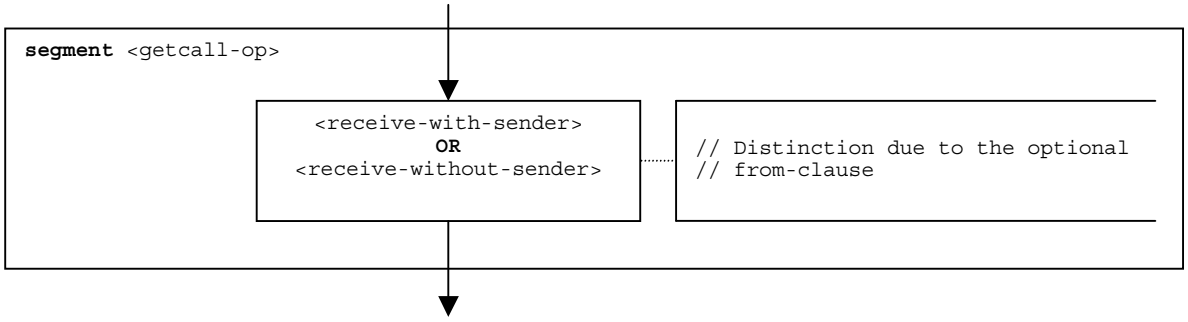
الشكل Z.143/85 – المقطع <predef-ext-func-call> في مخطط الانسياب

25.9 العملية getcall

يكون تركيب العملية **getcall** كما يلي:

```
<portId>.getcall (<matchingSpec>) [from <component_expression>] -> [<assignmentPart>]
```

فيما عدا الكلمة الرئيسية **getcall**، يكون هذا التركيب مشابهاً لتركيب العملية **receive**. بالتالي، تعالج الدلالة التشغيلية العملية **getcall** بالطريقة نفسها التي تستخدمها العملية **receive**، وهذا يرد في المقطع <getcall-op> (الشكل 86) الذي يحدد تنفيذ العملية **getcall**. ويحيل الشكل إلى مقاطع مخطط الانسياب المرتبطة بالعملية **receive** (انظر الفقرة 37.9).



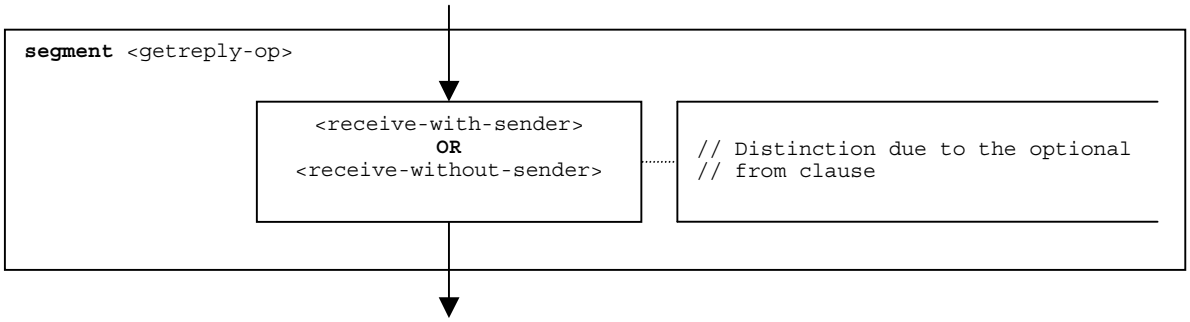
الشكل Z.143/86 – المقطع <getcall-op> في مخطط الانسياب

26.9 العملية getreply

يكون تركيب العملية **getreply** كما يلي:

```
<portId>.getreply (<matchingSpec>) [from <component-expression>] [-> <assignmentPart>]
```

فيما عدا الكلمة الرئيسية **getreply**، يكون هذا التركيب مشابهاً لتركيب العملية **receive**. بالتالي، تعالج الدلالة التشغيلية العملية **getreply** بالطريقة نفسها التي تستخدمها العملية **receive**، وهذا يرد في المقطع <getreply-op> (الشكل 87) الذي يحدد تنفيذ العملية **getreply**. ويحيل الشكل إلى مقاطع مخطط الانسياب المرتبطة بالعملية **receive** (انظر الفقرة 37.9).



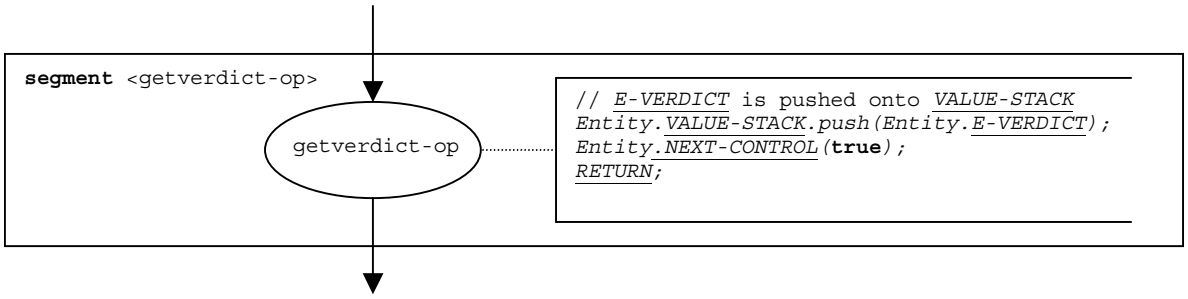
الشكل Z.143/87 – المقطع <getreply-op> في مخطط الانسياب

27.9 العملية getverdict

يكون تركيب العملية **getverdict** كما يلي:

```
getverdict
```

يحدد المقطع <getverdict-op> في الشكل 88 تنفيذ العملية **getverdict**.



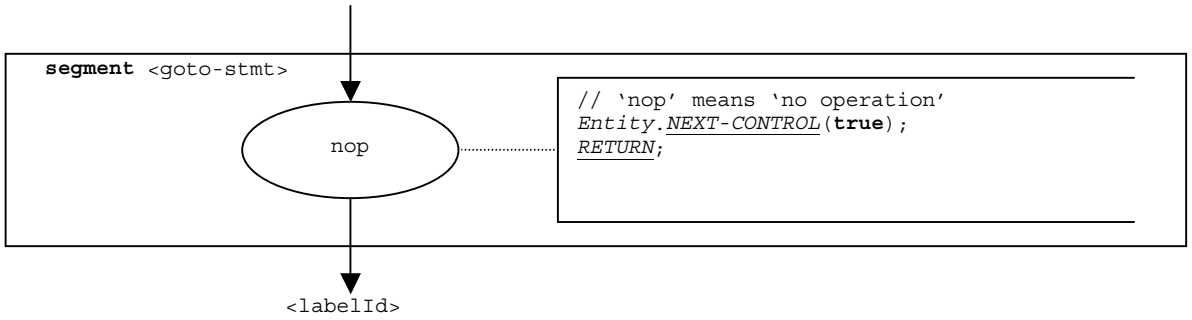
الشكل Z.143/88 – المقطع <getverdict-op> في مخطط الانسياب

28.9 البيان goto

يكون تركيب البيان **goto** كما يلي:

```
goto <labelId>
```

يحدد المقطع <goto-stmt> في الشكل 89 تنفيذ البيان **goto**.



ملاحظة – تشير المعلمة <labelId> للبيان **goto** إلى تحويل التحكم إلى المكان حيث تم تحديد الوسم *<labelId>* (انظر أيضاً الفقرة 30.9).

الشكل Z.143/89 – المقطع <goto-stmt> في مخطط الانسياب

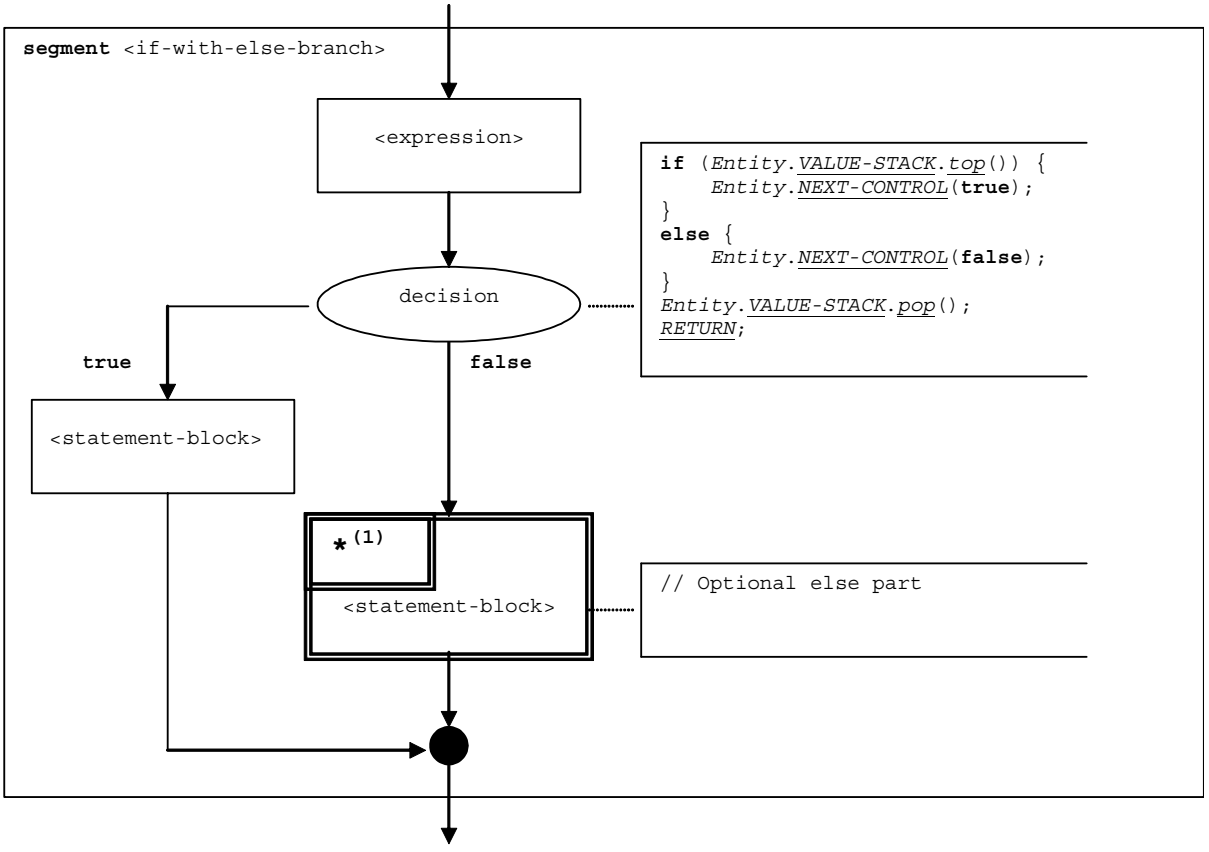
29.9 البيان if-else

يكون تركيب البيان **if-else** كما يلي:

```
if (<boolean-expression>) <statement-block1>
    [else <statement-block2>]
```

يكون الجزء else من البيان **if-else** اختيارياً.

يحدد المقطع <if-else-stmt> في الشكل 90 تنفيذ البيان **if-else**.



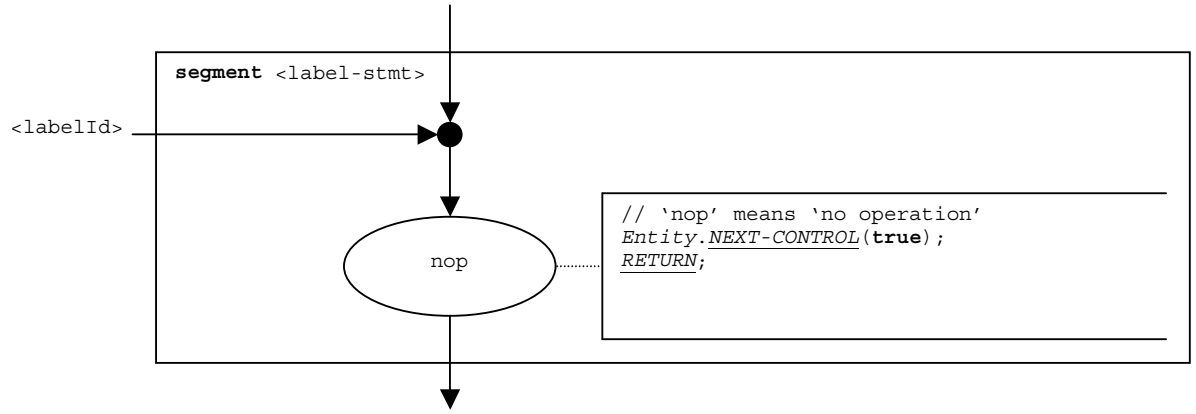
الشكل Z.143/90 – المقطع <if-else-stmt> في مخطط الانسياب

30.9 البيان label

يكون تركيب البيان **label** كما يلي:

```
label <labelId>
```

يحدد المقطع <label-stmt> في الشكل 91 تنفيذ البيان **label**.



ملاحظة – تشير المعلمة <labelId> للبيان **label** إلى احتمال أن يكون الوسم هدف قفزة يحددها البيان **goto** (انظر أيضاً الفقرة 28.9).

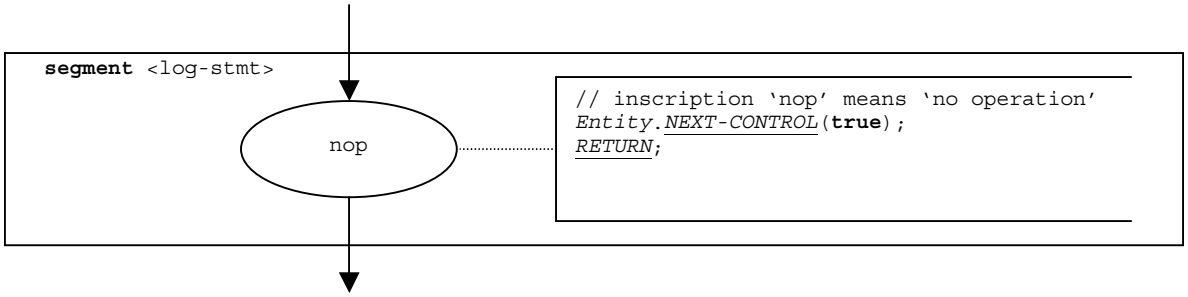
الشكل Z.143/91 – المقطع <label-stmt> في مخطط الانسياب

31.9 البيان log

يكون تركيب البيان **log** كما يلي:

```
log (<informal-description>)
```

يحدد المقطع <log-stmt> في الشكل 92 تنفيذ البيان **log**.



ملاحظة: لا تنطوي المعلمة <informal description> للبيان log على أي دلالة بالنسبة للدلالات التشغيلية وبالتالي لا تمثل في مقطع المخطط الانسيابي.

الشكل Z.143/92 – المقطع <log-stmt> في مخطط الانسياب

32.9 العملية map

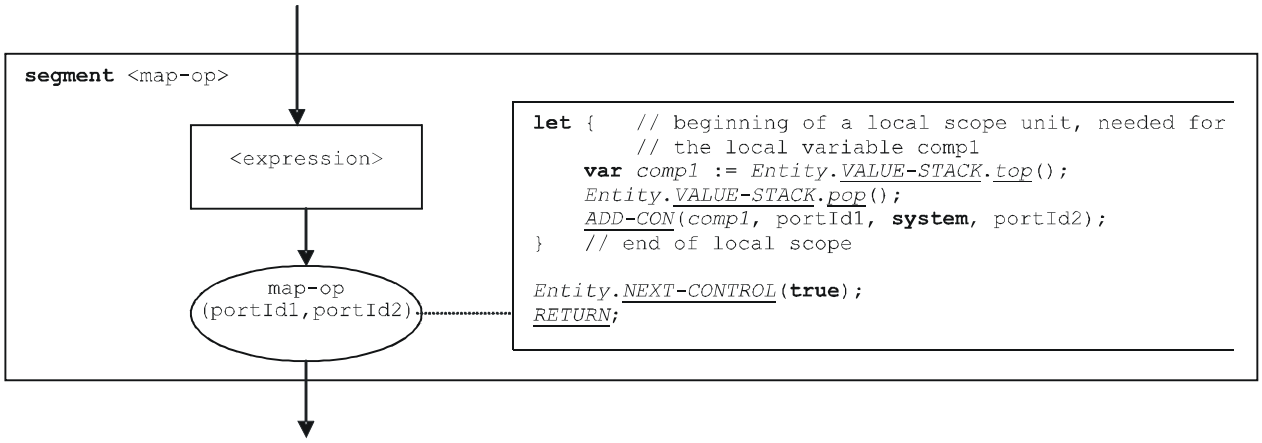
يكون تركيب العملية **map** كما يلي:

```
map(<component-expression>:<portId1>, system:<portId2>)
```

يعتبر <portId1> و<portId2> معرفين لمنفذ كل من مكونة الاختبار ومكونة السطح البيني لنظام الاختبار. ويتم الإحالة إلى المكونات التي تعود إليها المنافذ بواسطة مرجع المكونة <component-expression>. ويمكن تخزين المرجع في متغيرات أو إعادته من خلال وظيفة ما، أي إنه تعبير يعطي تقييمه مرجع المكونة. يستخدم مكدّس القيم لتخزين مرجع المكونة.

ملاحظة – بالنسبة للعملية **map** ليس من المهم إذا كان البيان <portId>**system:** معلمة أولى أو ثانية. وتوخياً للتبسيط، يفترض أنه دائماً المعلمة الثانية.

يحدد المقطع <map-op> في الشكل 93 تنفيذ العملية **map**.



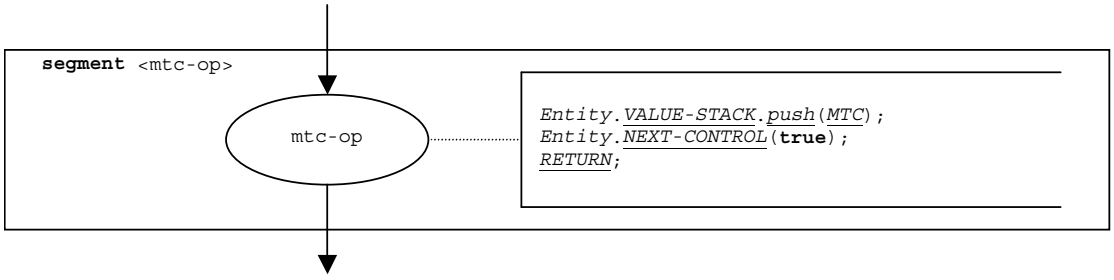
الشكل Z.143/93 – المقطع <map-op> في مخطط الانسياب

33.9 العملية mtc

يكون تركيب العملية **mtc** كما يلي:

```
mtc
```

يحدد المقطع <mtc-op> في الشكل 94 تنفيذ العملية **mtc**.



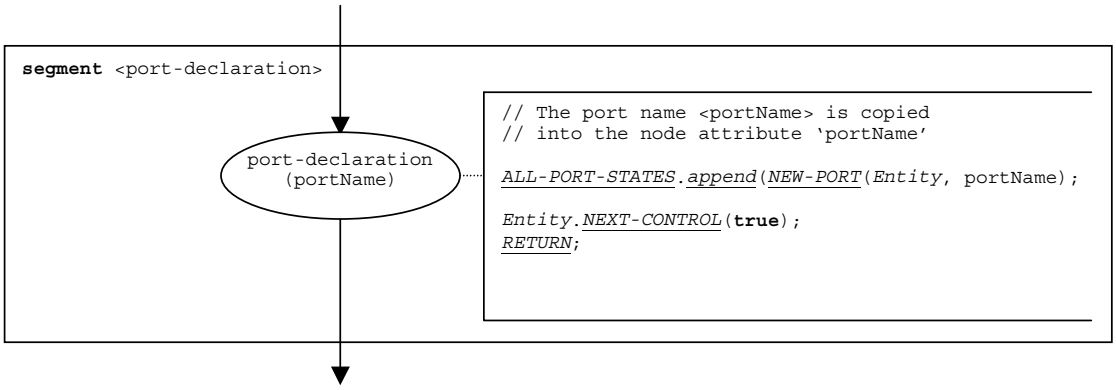
الشكل Z.143/94 – المقطع <mtc-op> في مخطط الانسياب

34.9 إعلان port

يكون تركيب إعلان **port** كما يلي:

```
<portType> <portName>
```

يمكن أن ترد إعلانات **port** في تحديدات نمط المكونة. ومن آثار إعلان **port** إنشاء منفذ جديد عند إنشاء مكونة جديدة من النمط المقابل. يحدد المقطع <port-declaration> في الشكل 95 تنفيذ إعلان port.



الشكل Z.143/95 – المقطع <port-declaration> في مخطط الانسياب

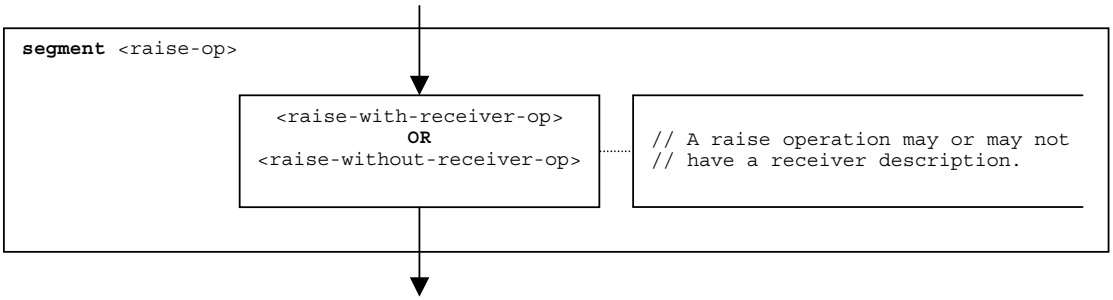
35.9 العملية raise

يكون تركيب العملية **raise** كما يلي:

```
<portId>.raise (<exceptSpec>) [to <component-expression>]
```

يشير الجزء الاختياري <component-expression> في الشطر **to** إلى الكيان المستقبل. ويمكن توفيره إما في شكل قيمة متغيرة أو قيمة العودة لوظيفة ما.

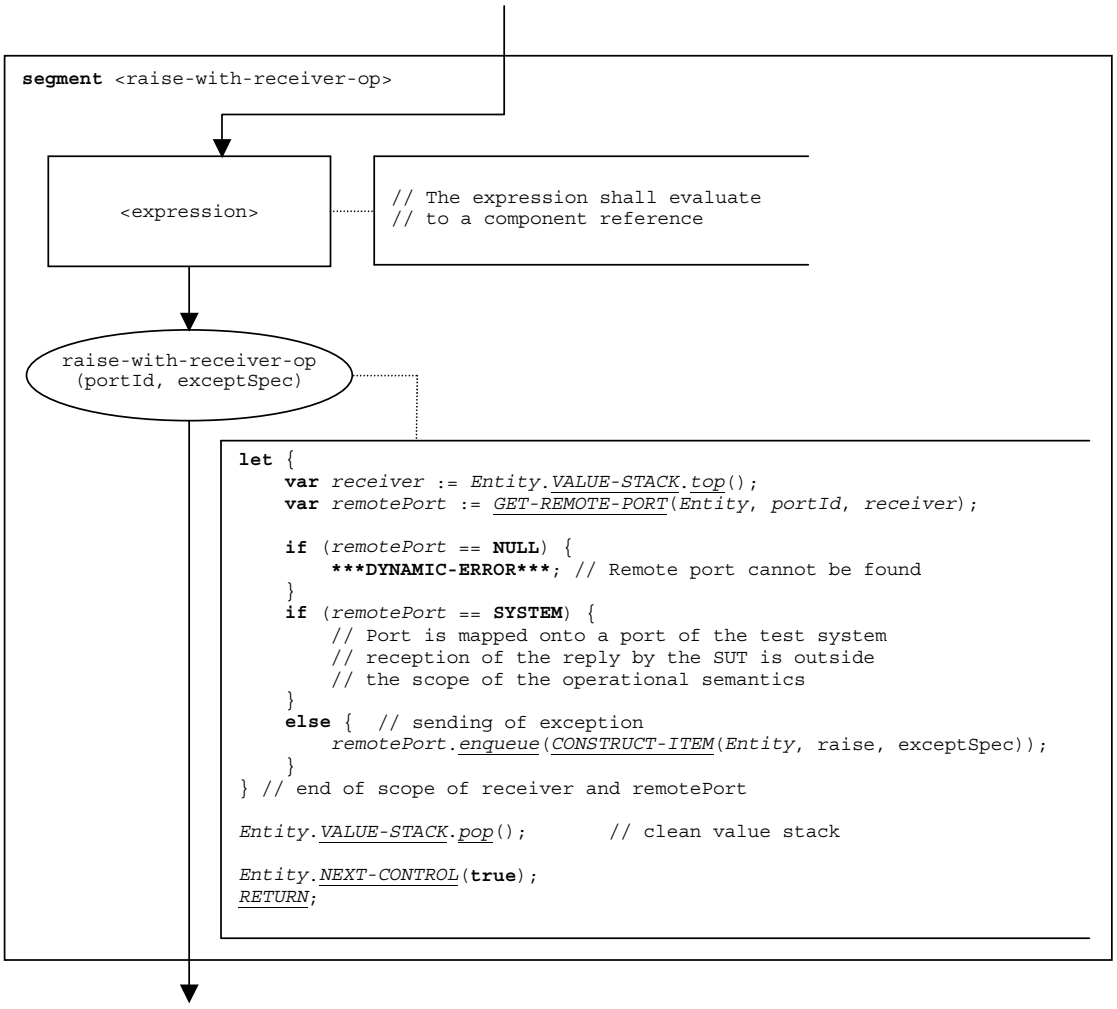
يحدد المقطع <raise-op> في الشكل 96 تنفيذ العملية **raise**.



الشكل Z.143/96 – المقطع <raise-op> في مخطط الانسياب

1.35.9 المقطع <raise-with-receiver-op>

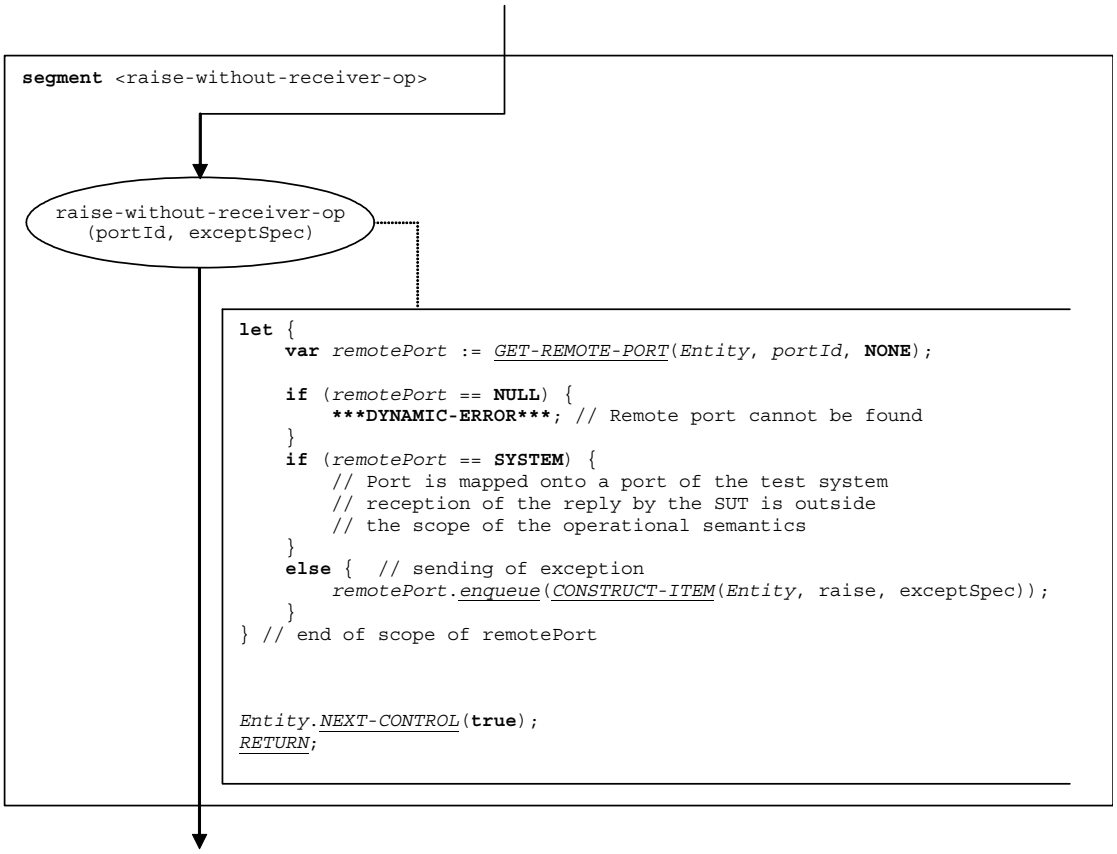
يحدد المقطع <raise-with-receiver-op> في الشكل 97 تنفيذ العملية **raise** التي يحدد من أجلها المستقبل في شكل عبارة.



الشكل Z.143/97 – المقطع <raise-with-receiver-op> في مخطط الانسياب

2.35.9 المقطع <raise-without-receiver-op>

يحدد المقطع <raise-without-receiver-op> في الشكل 98 تنفيذ العملية **raise** دون الشطر **to**.



الشكل Z.143/98 – المقطع <raise-without-receiver-op> في مخطط الانسياب

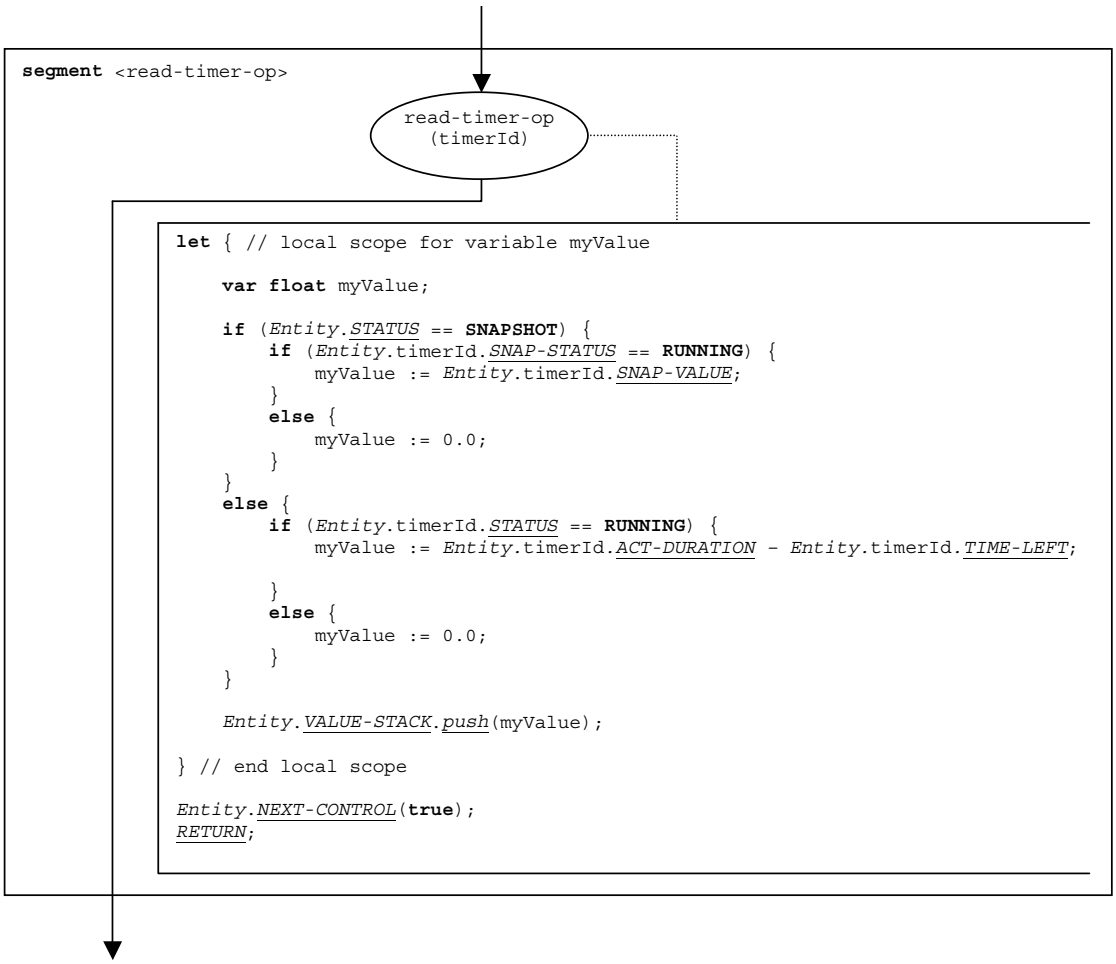
36.9 عملية read المنطبقة على المؤقتات

يكون تركيب العملية **read** المنطبقة على المؤقتات كما يلي:

```
<timerId>.read
```

يحدد المقطع <read-timer-op> في الشكل 99 تنفيذ العملية **read** المنطبقة على المؤقتات.

فيما يتعلق بالعملية **read** المنطبقة على المؤقتات، يمكن من جهة التمييز بين استخدامها في حماية منطق Boolean في بيان **alt** أو في عملية **call** مانعة، ومن جهة ثانية جميع الحالات الأخرى. فإذا استخدمت هذه العملية في حماية Boolean فإن نتيجة العملية تقوم على أساس اللحظة الفعلية، أي المدخلات *SNAP-STATUS* و*SNAP-VALUE* لوصلة المؤقت، وفي كل الحالات الأخرى، تحدد المدخلات *STATUS* و*ACT-DURATION* و*TIME-LEFT* لوصلة التوقيت نتيجة العملية.



الشكل Z.143/99 – المقطع <read-timer-op> في مخطط الانسياب

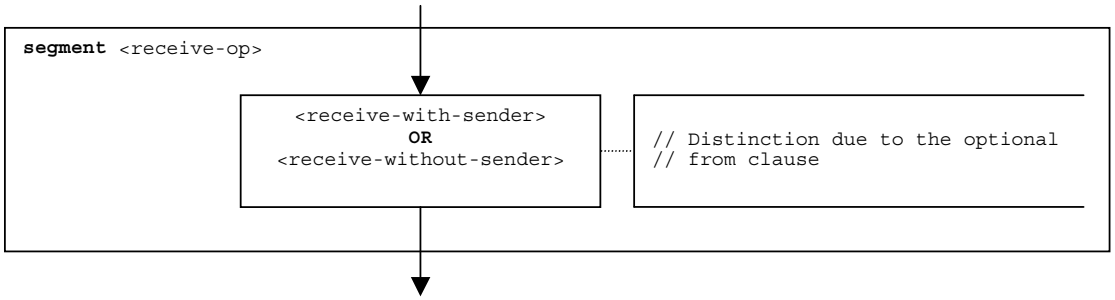
37.9 العملية receive

يكون تركيب العملية **receive** كما يلي:

```
<portId>.receive (<matchingSpec>) [from <component-expression>] [-> <assignmentPart>]
```

يشير الجزء الاختياري <component-expression> في الشطر **from** إلى الكيان المرسل. ويمكن توفيره إما في شكل قيمة متغيرة أو قيمة العودة لوظيفة ما أي يفترض أنها عبارة. ويشير الجزء الاختياري <assignmentPart> إلى تخصيص المعلومات المستقبلة إذا كانت الرسالة المستلمة تقابل مواصفة التزاوج <matchingSpec> والشطر (الاختياري) **from**.

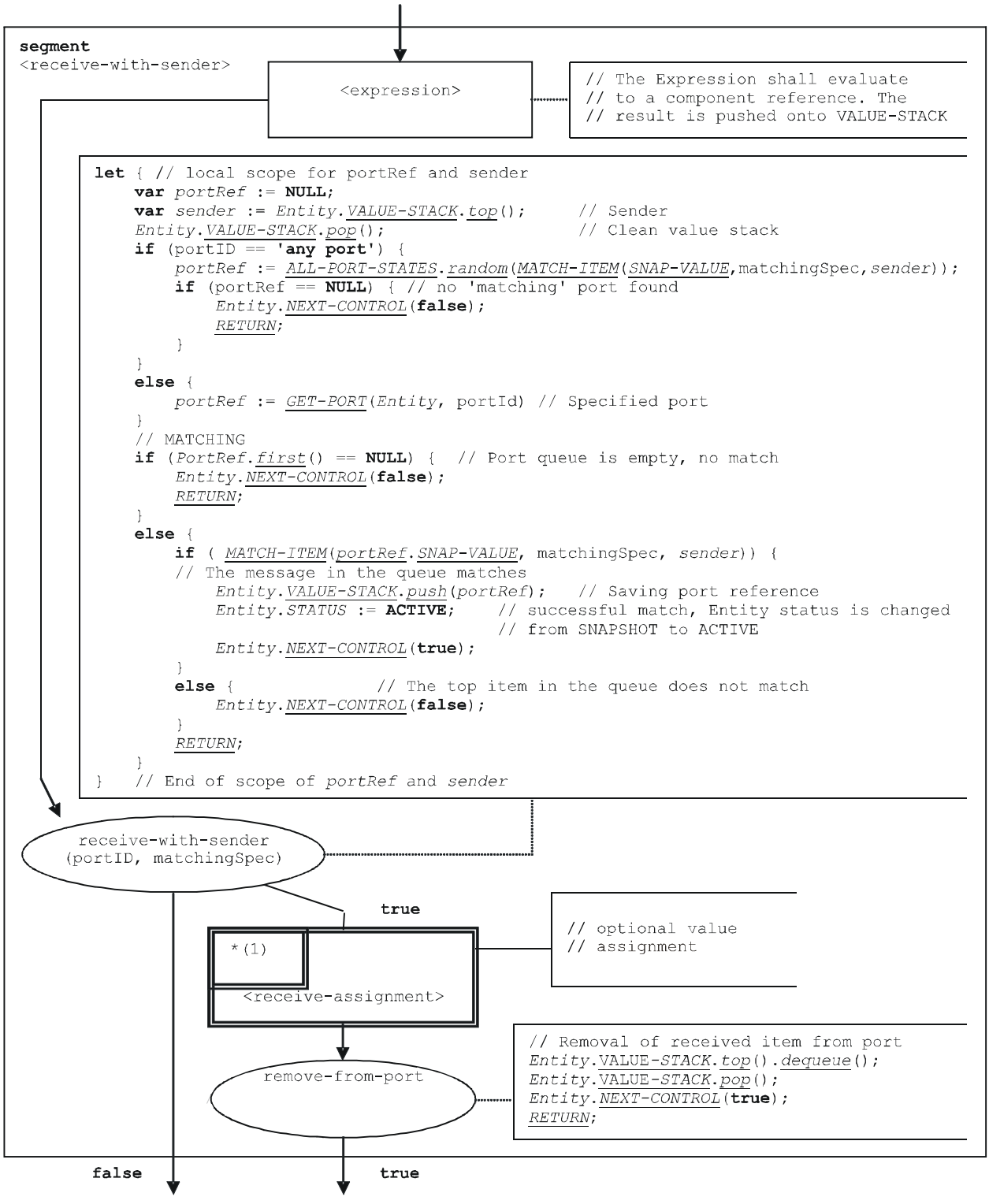
يحدد المقطع <receive-op> في الشكل 100 تنفيذ العملية **receive**.



الشكل Z.143/100 – المقطع <receive-op> في مخطط الانسياب

1.37.9 المقطع <receive-with-sender>

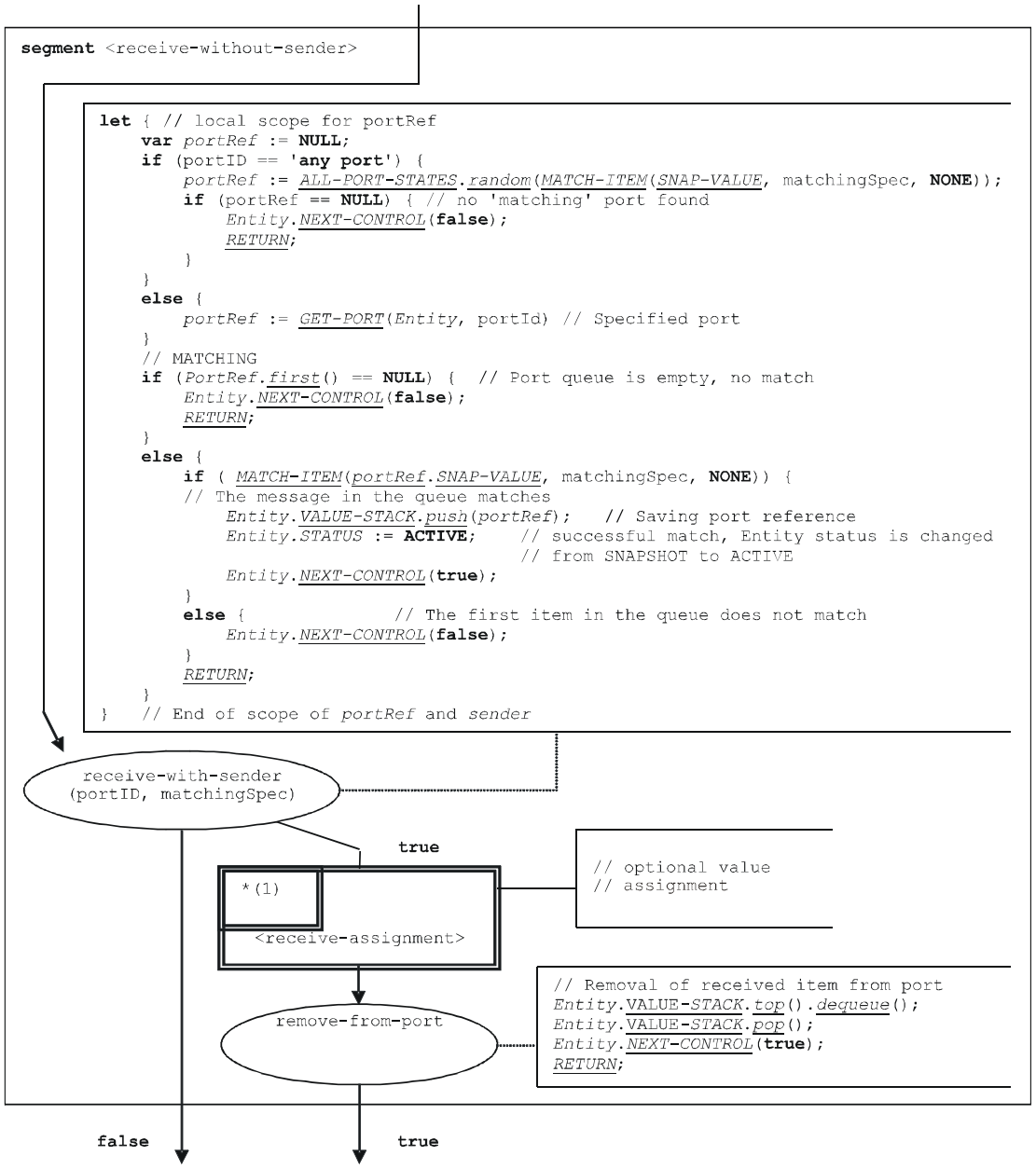
يحدد المقطع <receive-with-sender> في الشكل 101 تنفيذ العملية **receive** التي يحدد من أجلها المستقبل في شكل عبارة.



الشكل Z.143/101 – المقطع <receive-with-sender> في مخطط الانسياب

2.37.9 المقطع <receive-without-sender>

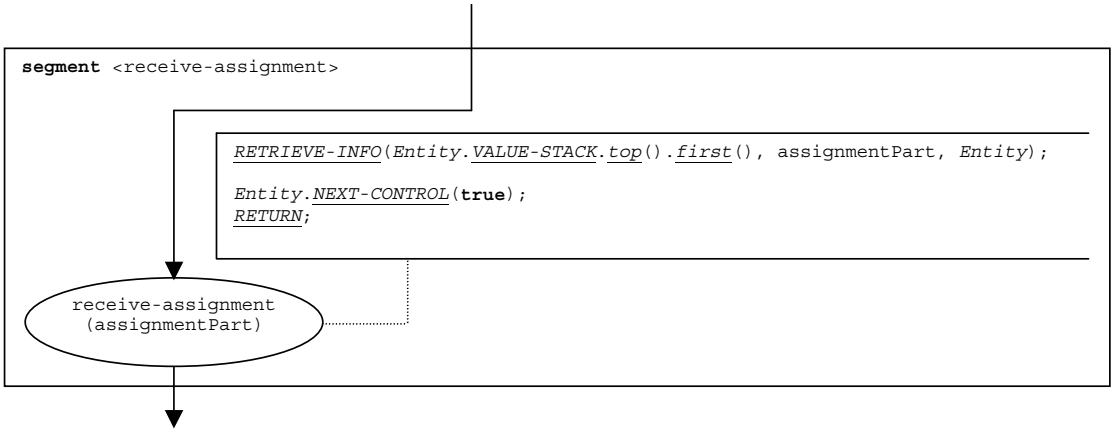
يحدد المقطع <receive-without-sender> في الشكل 102 تنفيذ العملية **receive** دون الشطر **from**.



الشكل Z.143/102 – المقطع <receive-without-sender> في مخطط الانسياب

3.37.9 المقطع <receive-assignment>

يحدد المقطع <receive-assignment> في الشكل 103 استخراج معلومات الرسائل المستلمة وربطها بمتغيرات.



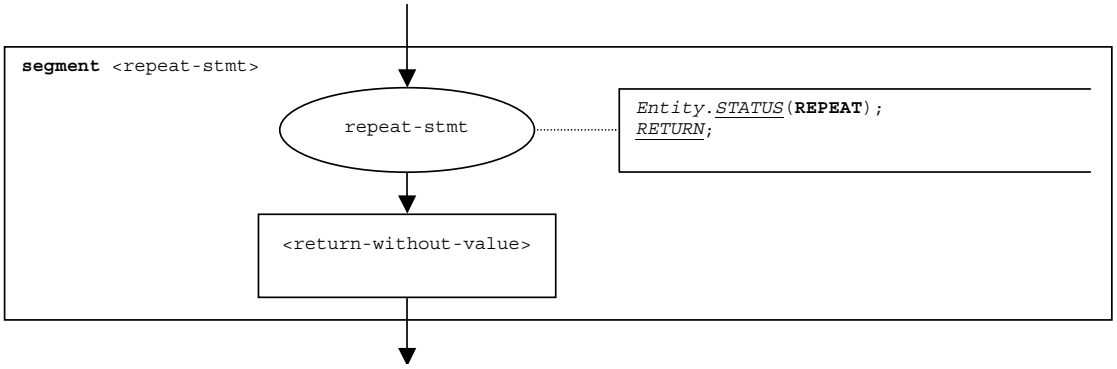
الشكل Z.143/103 – المقطع <receive-assignment> في مخطط الانسياب

38.9 البيان repeat

يكون تركيب البيان **repeat** كما يلي:

```
repeat
```

البيان repeat هو بيان return أساساً دون قيمة عودة مما يغير أيضاً حالة الكيان إلى REPEAT. وتفرض الحالة REPEAT إعادة تقييم البيان alt الذي تم فيه تنفيذ البيان repeat. ويحدد المقطع <repeat-stmt> في الشكل 104 تنفيذ البيان repeat.



الشكل Z.143/104 – المقطع <repeat-stmt> في مخطط الانسياب

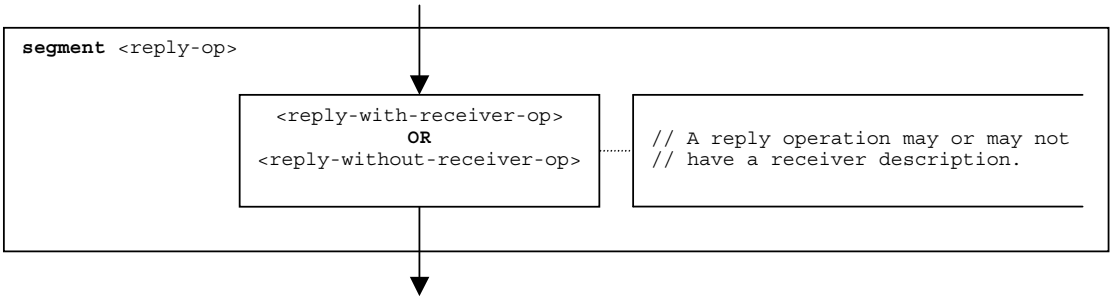
39.9 العملية reply

يكون تركيب العملية **reply** كما يلي:

```
<portId>.reply (<replySpec>) [to <component-expression>]
```

يشير الجزء الاختياري <component_expression> في الشطر **to** إلى الكيان المستقبل. ويمكن توفيره إما في شكل قيمة متغيرة أو قيمة العودة لوظيفة ما.

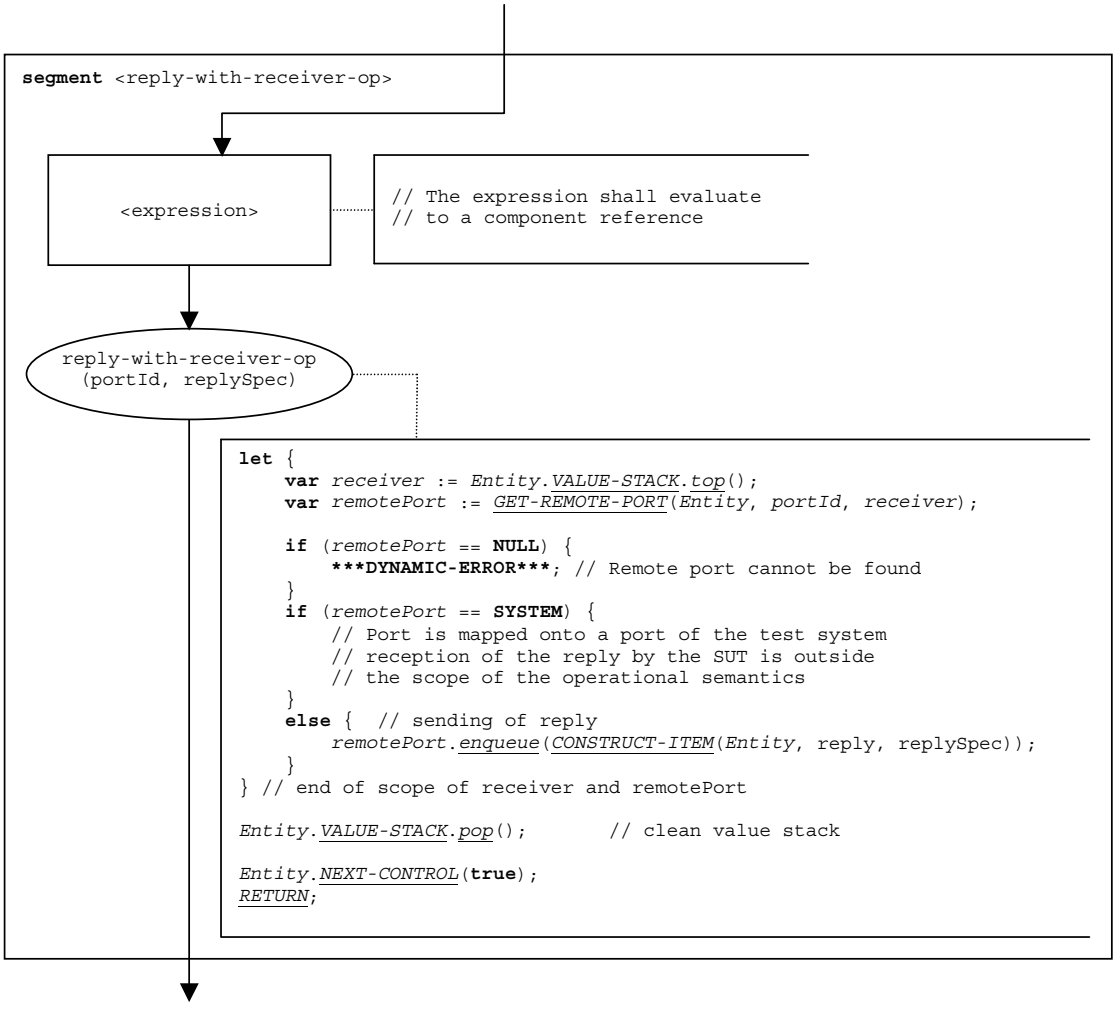
يحدد المقطع <reply-op> في الشكل 105 تنفيذ العملية **reply**.



الشكل Z.143/105 – المقطع <reply-op> في مخطط الانسياب

1.39.9 المقطع <reply-with-receiver-op>

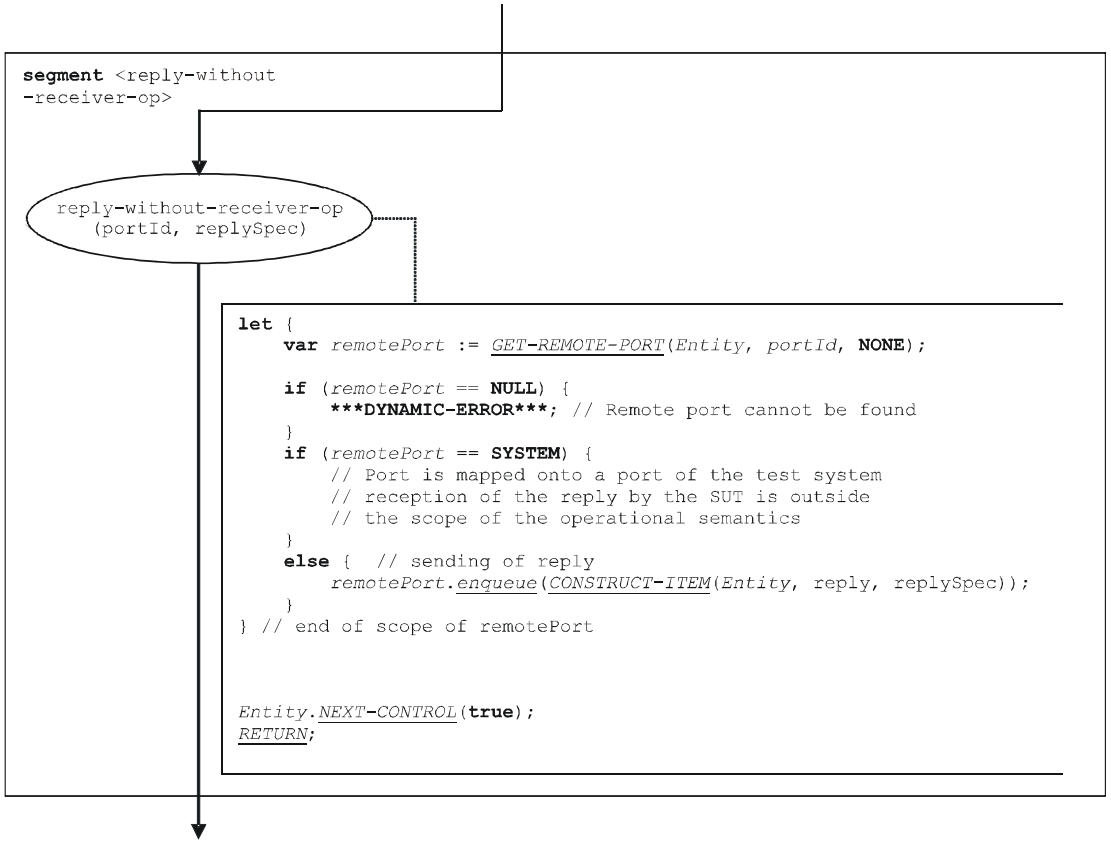
يحدد المقطع <reply-with-receiver-op> في الشكل 106 تنفيذ العملية **reply** التي يحدد من أجلها المستقبل في شكل عبارة.



الشكل Z.143/106 – المقطع <reply-with-receiver-op> في مخطط الانسياب

2.39.9 المقطع <reply-without-receiver-op>

يحدد المقطع <reply-without-receiver-op> في الشكل 107 تنفيذ العملية **reply** دون الشطر **to**.



الشكل Z.143/107 – المقطع <reply-without-receiver-op> في مخطط الانسياب

40.9 البيان return

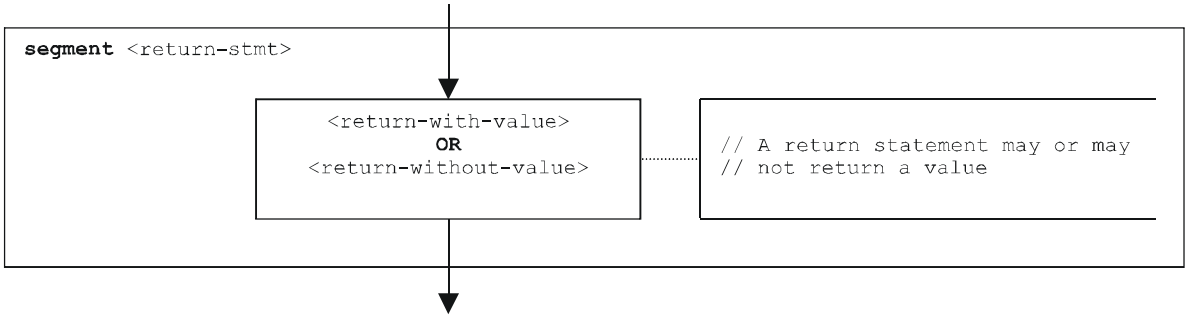
يكون تركيب البيان **return** كما يلي:

```
return [<expression>]
```

يصف الجزء الاختياري <expression> قيمة عودة ممكنة لوظيفة ما. ويعني تنفيذ البيان **return** أن التحكم يغادر وحدة التطبيق الفعلية، أي يجب حذف المتغيرات والمؤقتات المجهولة في هذه الوحدة ويجب تحديث مكدّس القيم. ويكون للبيان **return** أثر على عملية **stop** منطبقة على المكونات إذا كان البيان الأخير الوارد في وصف السلوك.

ملاحظة – تنتهي الاختبارات الأولية وعملية التحكم بالنموذج دائماً بالعملية **stop** المنطبقة على المكونات، ويعود ذلك إلى تمثيلها في مخطط الانسياب (انظر الفقرة 2.8). ويمكن للمكونات الأخرى فقط أن تنتهي بالبيان **return**.

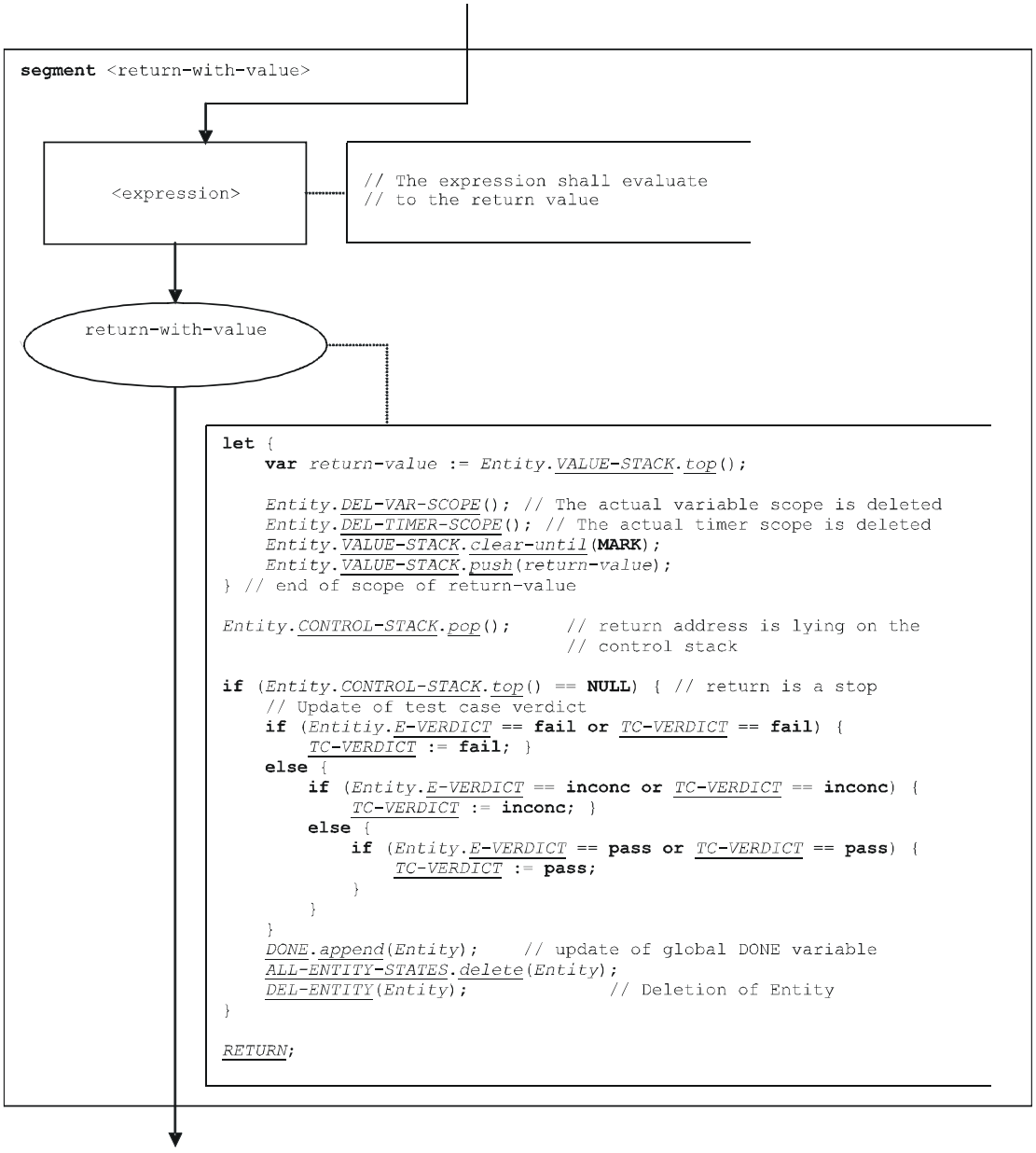
يحدد المقطع <return-stmt> في الشكل 108 تنفيذ البيان **return**.



الشكل Z.143/108 – المقطع <return-stmt> في مخطط الانسياب

1.40.9 المقطع <return-with-value>

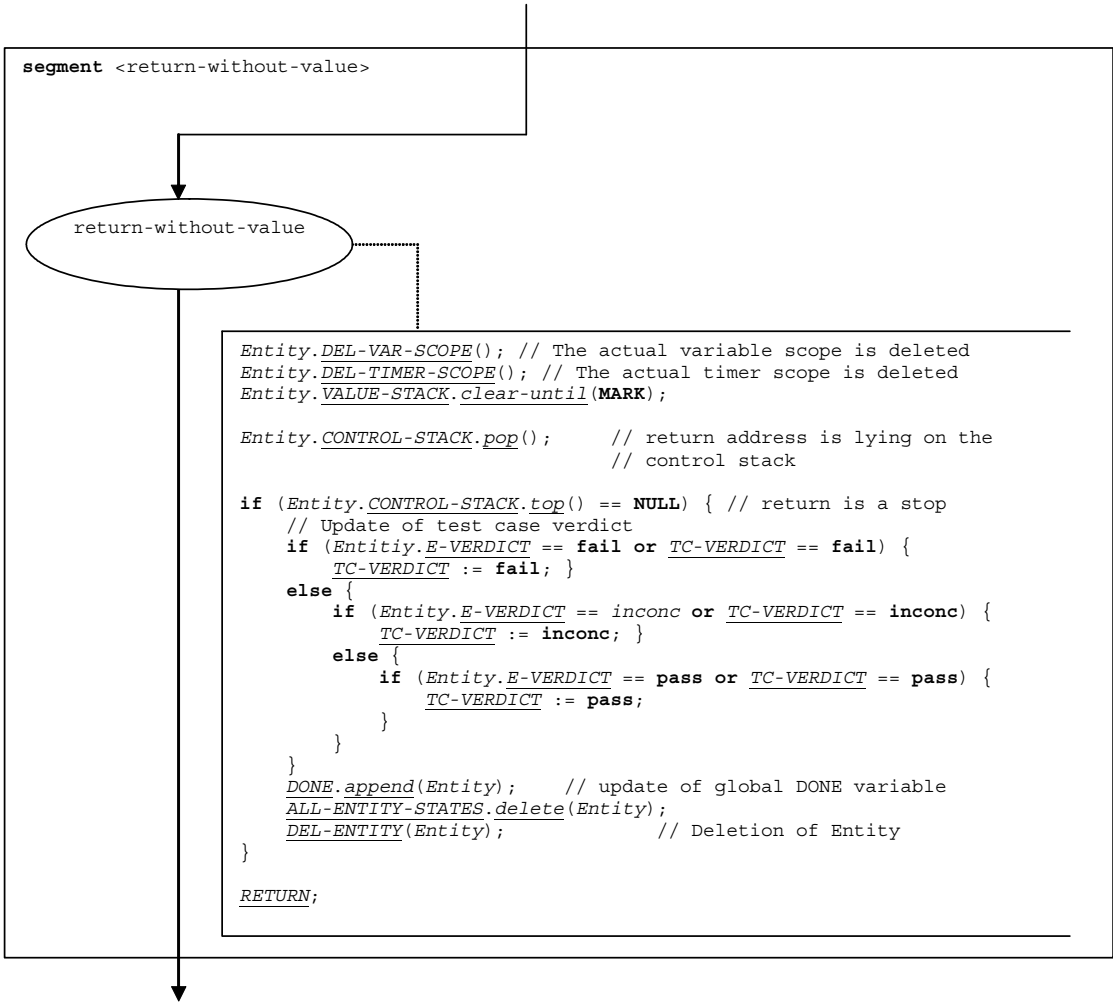
يحدد المقطع <return-with-value> في الشكل 109 تنفيذ البيان **return** الذي يعيد قيمة محددة في شكل عبارة.



الشكل Z.143/109 – المقطع <return-with-value> في مخطط الانسياب

2.40.9 المقطع <return-without-value>

يحدد المقطع <return-without-value> في الشكل 110 تنفيذ البيان **return** الذي لا يعيد أي قيمة.



الشكل Z.143/110 – المقطع <return-without-value> في مخطط الانسياب

41.9 العملية running المنطبقة على المكونات

يكون تركيب العملية **running** المنطبقة على المكونات كما يلي:

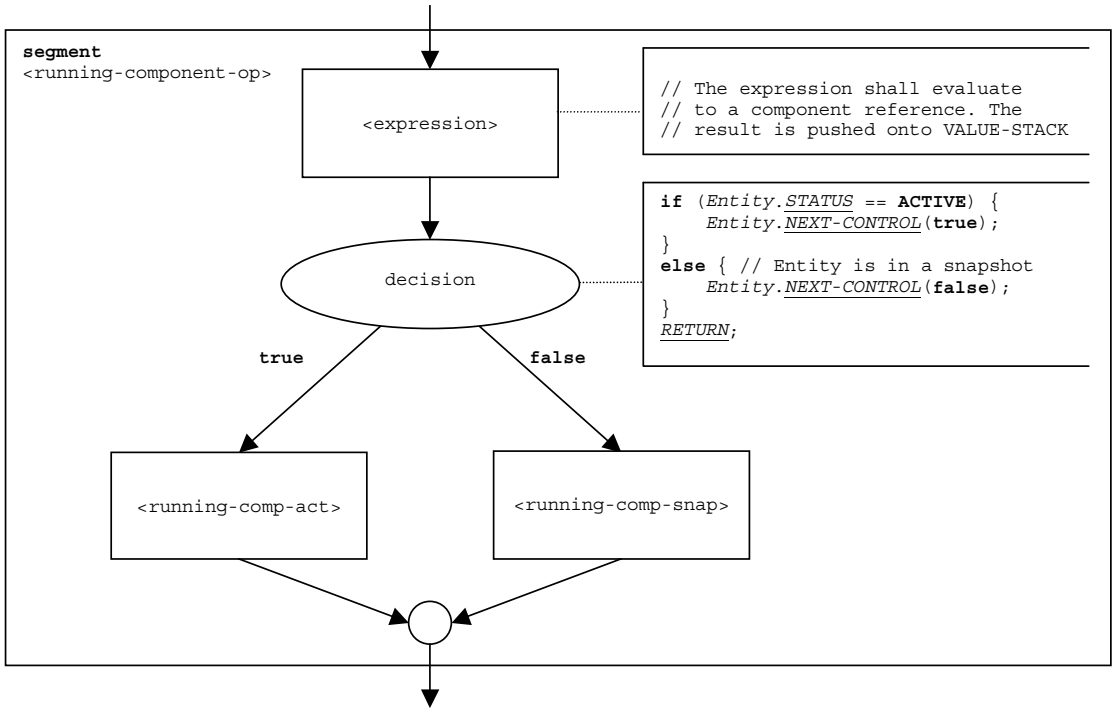
```
<component-expression>.running
```

تتحقق العملية **running** المطبقة على المكونات مما إذا كان مكون قيد التشغيل أو متوقفاً عن العمل. تُعرَّف المكونة الواجب التحقق منها بمرجع مكونة قد يكون متغيراً أو وظيفة تعيد قيمة ما، أي إنه عبارة. وتوخياً للتبسيط، يُعتبر التعبيران الرئيسيان **all component** و**any component** عبارتين خاصتين.

تميز العملية **running** المنطبقة على المكونات بين استخدامها في حماية Boolean لبيان **alt** أو لعملية وقف **call** وبين كل الحالات الأخرى. فإذا استخدمت في حماية Boolean، تكون نتيجتها على أساس اللحظة الفعلية، وفي جميع الحالات الأخرى تقابل نتيجتها مباشرة المعلومات المتعلقة بالحالة.

توضع نتيجة العملية **running** المنطبقة على المكونات في مكدّس قيم الكيان والتي تُدعى العملية.

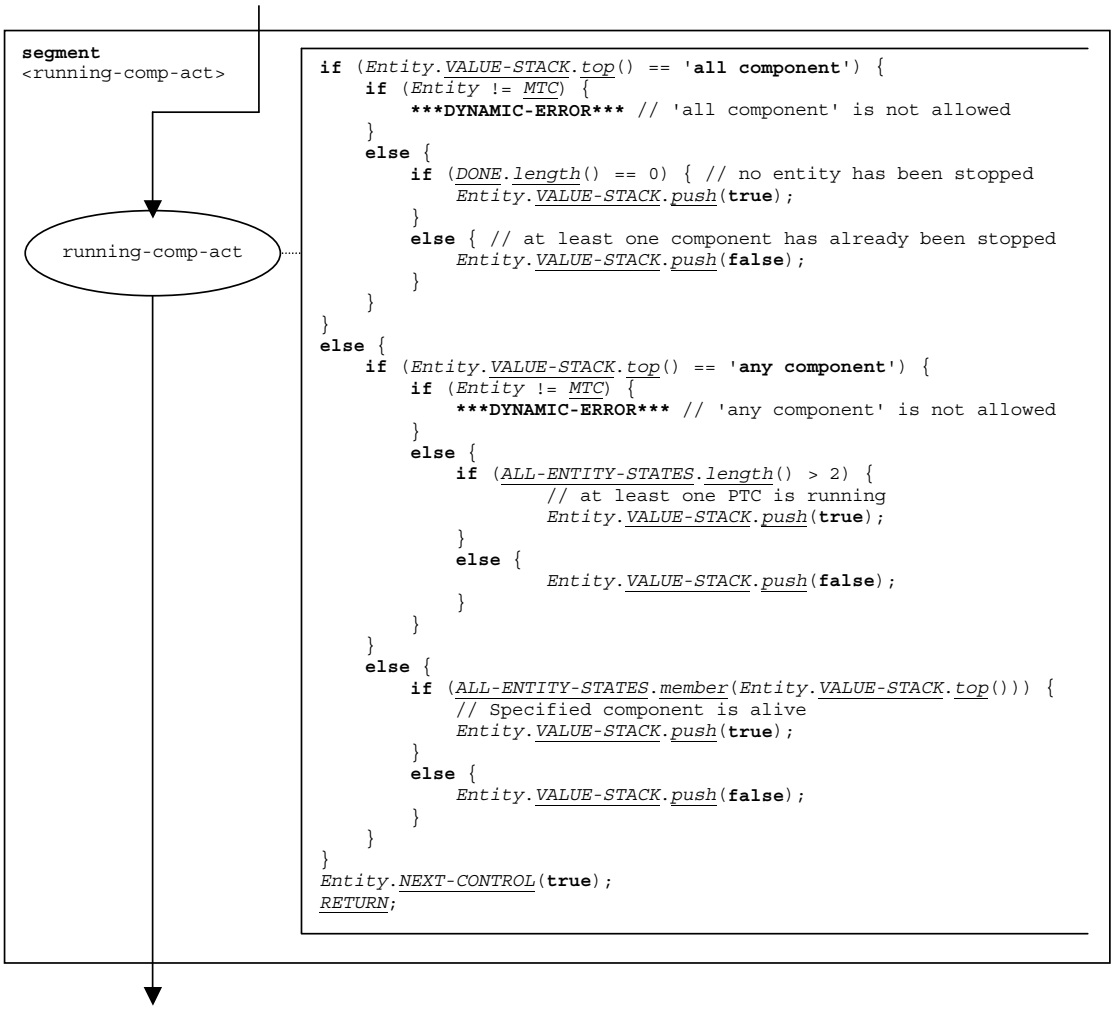
يحدد المقطع <running-component-op> في الشكل 111 تنفيذ العملية **running** المطبقة على المكونات.



الشكل Z.143/111 – المقطع <running-component-op> في مخطط الانسياب

1.41.9 المقطع <running-comp-act>

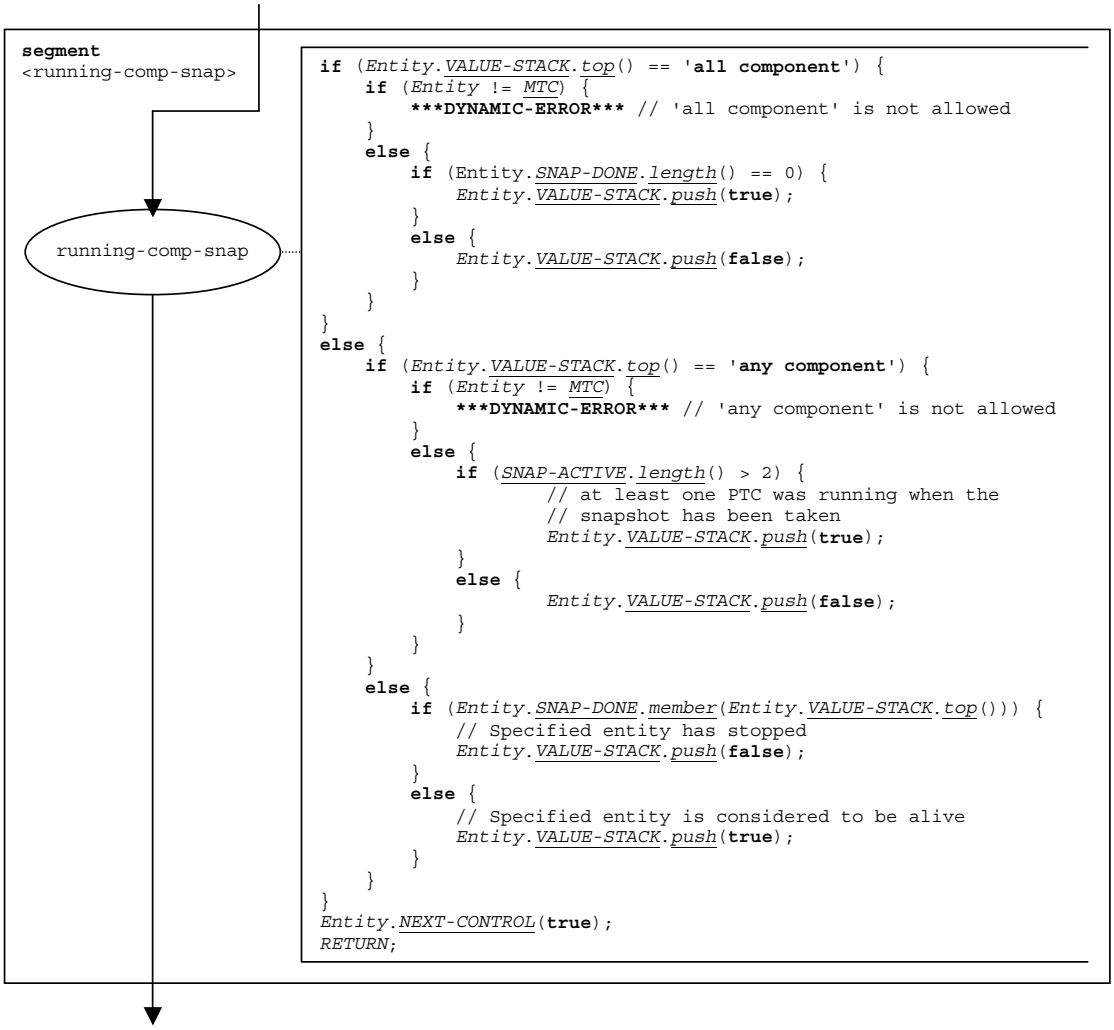
يحدد المقطع <running-comp-act> في الشكل 112 تنفيذ العملية **running** المنطبقة على المكونات خارج اللحظة الفعلية، أي عندما يكون الكيان في الحالة **ACTIVE**.



الشكل Z.143/112 – المقطع <running-compo-act> في مخطط الانسياب

2.41.9 المقطع <running-comp-snap>

يحدد المقطع <running-comp-snap> في الشكل 113 تنفيذ العملية **running** المنطبقة على المكونات خلال تقييم اللحظة الفعلية، أي عندما يكون الكيان في الحالة **SNAPSHOT**.



الشكل Z.143/113 – المقطع <running-compo-snap> في مخطط الانسياب

42.9 العملية running المنطبقة على المؤقتات

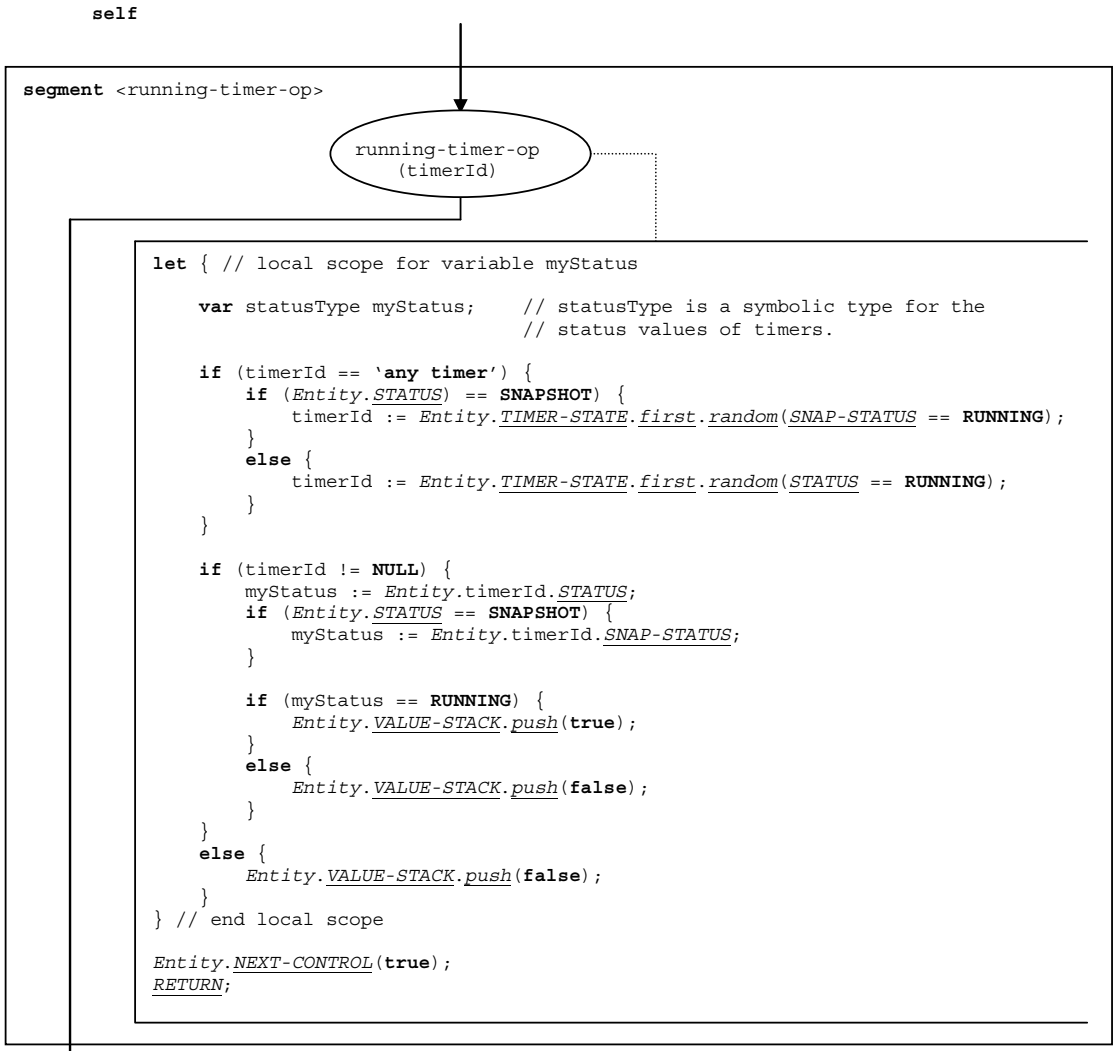
يكون تركيب العملية **running** المنطبقة على المؤقتات كما يلي:

```
<timerId>.running
```

يحدد المقطع <running-timer-op> في الشكل 114 تنفيذ العملية **running** المنطبقة على المؤقتات.

تميز العملية **running** المنطبقة على المؤقتات، بين استخدامها في حماية Boolean للبيان **alt** أو عملية وقف **call** وبين كل الحالات الأخرى. فإذا استخدمت في حماية Boolean تكون نتيجتها على أساس اللحظة الفعلية، أي مدخل *SNAP-STATUS* لوصلة المؤقت، وفي كل الحالات الأخرى، يحدد المدخل *STATUS* لوصلة المؤقت نتيجة العملية.

تعالج الكلمة الرئيسية **any** باعتبارها قيمة خاصة من قيم timerId.



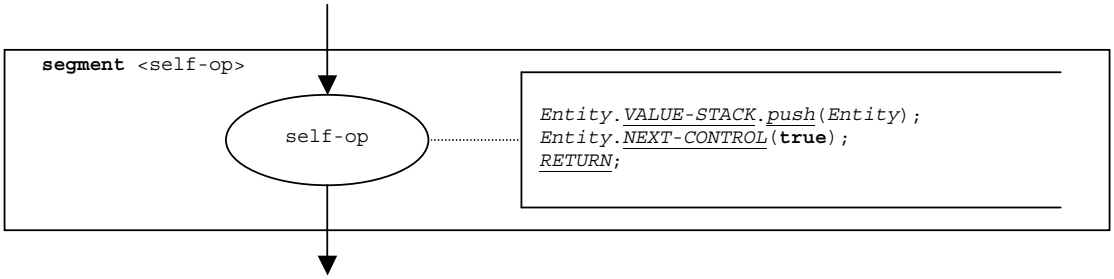
الشكل Z.143/114 – المقطع <running-timer-op> في مخطط الانسياب

43.9 العملية self

يكون تركيب العملية **self** كما يلي:

```
self
```

يحدد المقطع <self-op> في الشكل 115 تنفيذ العملية **self**.



الشكل Z.143/115 – المقطع <self-op> في مخطط الانسياب

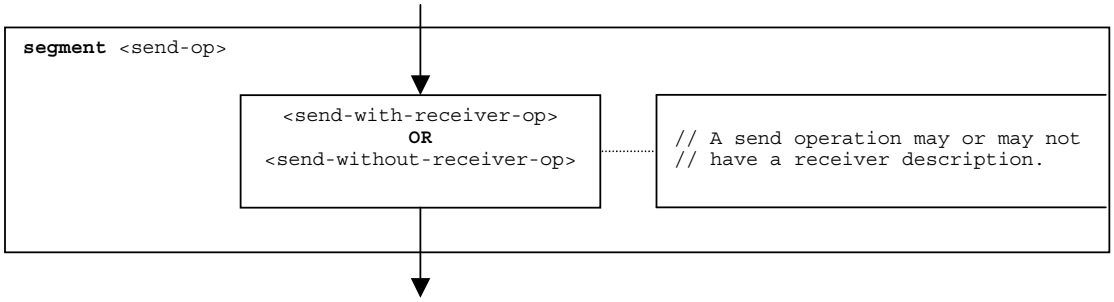
44.9 العملية send

يكون تركيب العملية **send** كما يلي:

```
<portId>.send (<send-spec>) [to <component-expression>]
```

يشير الجزء الاختياري <component-expression> في الشطر **to** إلى الكيان المستقبل. ويمكن توفيره إما في شكل قيمة متغيرة أو قيمة العودة لوظيفة ما.

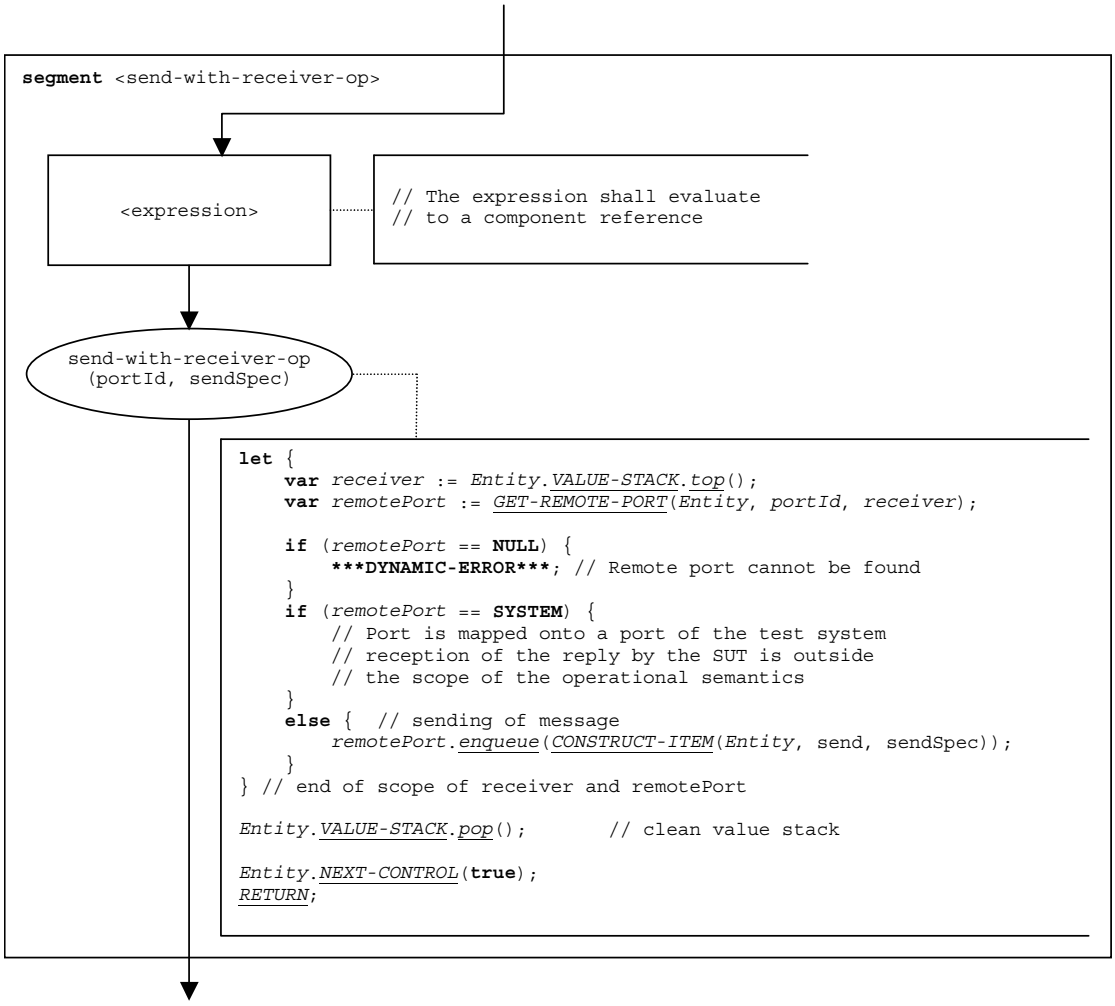
يحدد المقطع <raise-op> في الشكل 116 تنفيذ العملية **send**.



الشكل Z.143/116 – المقطع <send-op> في مخطط الانسياب

1.44.9 المقطع <send-with-receiver-op>

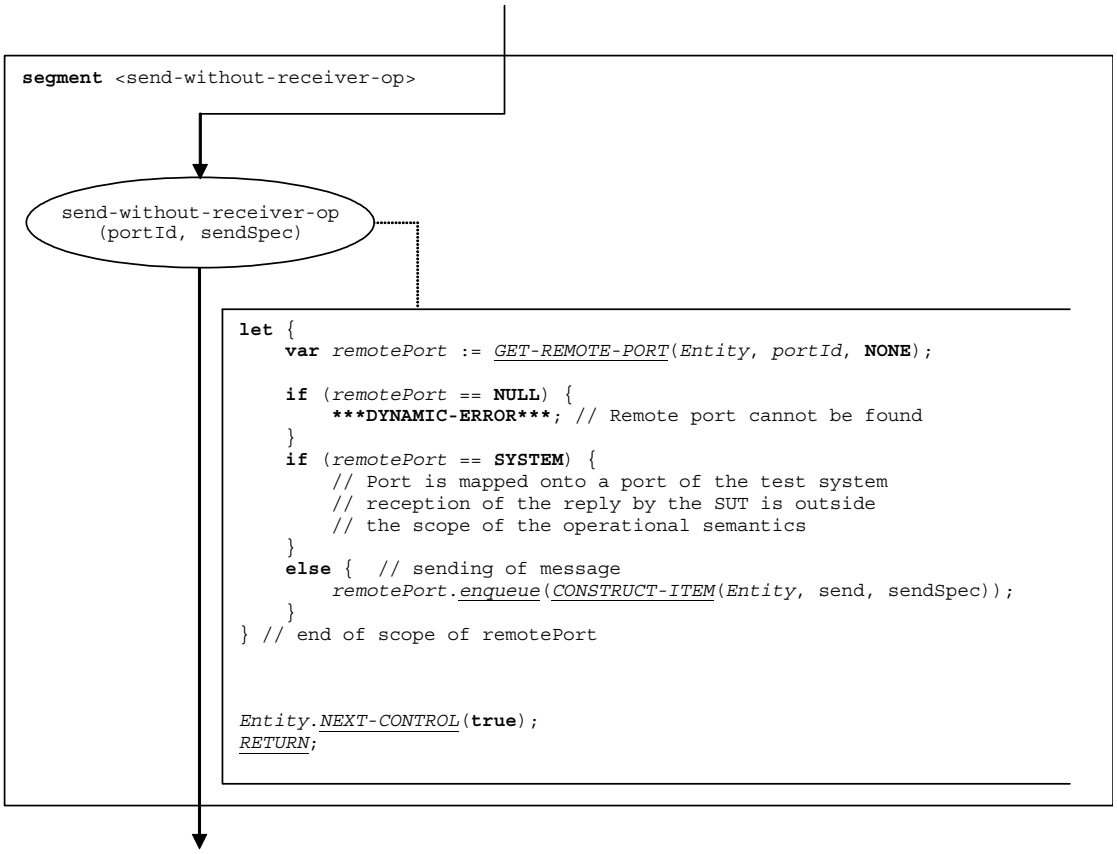
يحدد المقطع <send-with-receiver-op> في الشكل 117 تنفيذ العملية **send** حيث يحدد المستقبل في شكل عبارة.



الشكل Z.143/117 – المقطع <send-with-receiver-op> في مخطط الانسياب

2.44.9 المقطع <send-without-receiver-op>

يحدد المقطع <send-without-receiver-op> في الشكل 118 تنفيذ العملية **send** دون الشطر **to**.



الشكل Z.143/118 – المقطع <send-without-receiver-op> في مخطط الانسياب

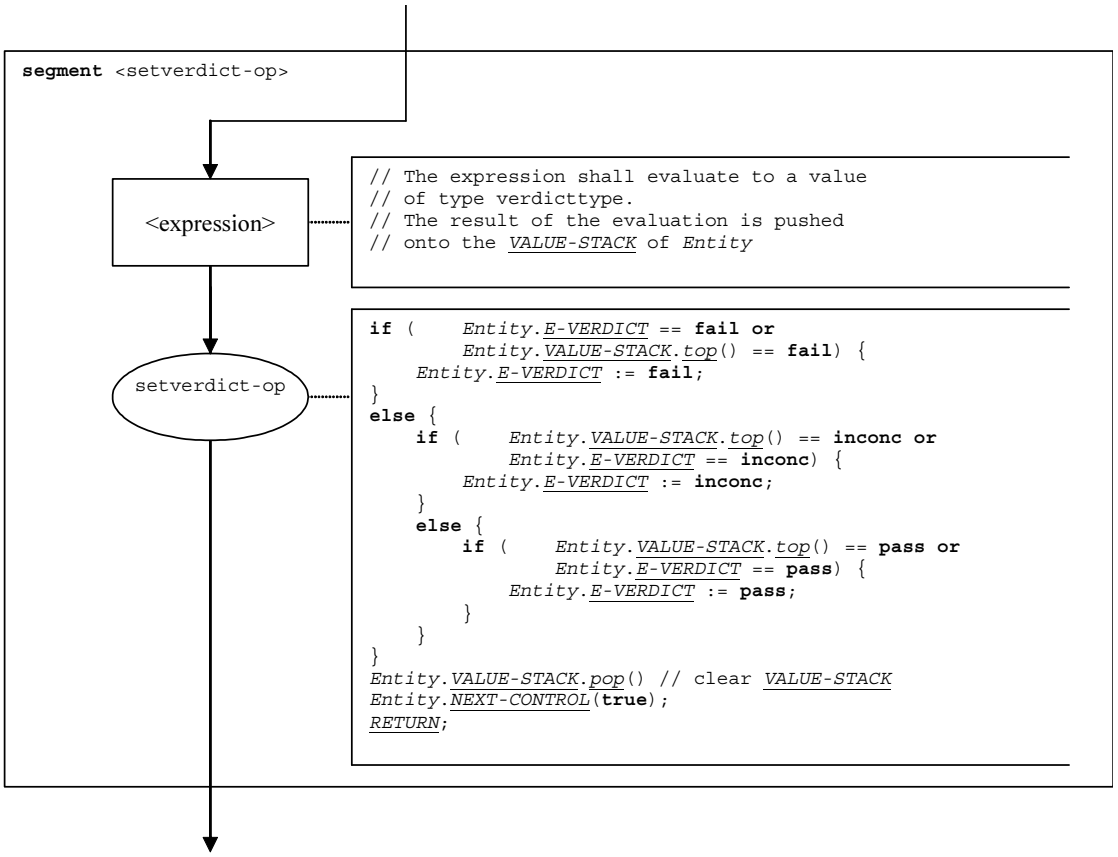
45.9 العملية setverdict

يكون تركيب العملية **setverdict** كما يلي:

```
setverdict(<verdicttype-expression>)
```

المعلمة <verdicttype-expression> للعملية **setverdict** عبارة يعطي تقييمها قيمة من النمط **verdicttype** أي **none** أو **pass** أو **inconc** أو **fail**. ويتم تقييم العبارة قبل تطبيق العملية **setverdict**.

يحدد المقطع <setverdict-op> في الشكل 119 تنفيذ العملية **setverdict**.



الشكل Z.143/119 – المقطع <setverdict-op> في مخطط الانسياب

46.9 العملية start المنطبقة على المكونات

يكون تركيب العملية **start** المنطبقة على المكونات كما يلي:

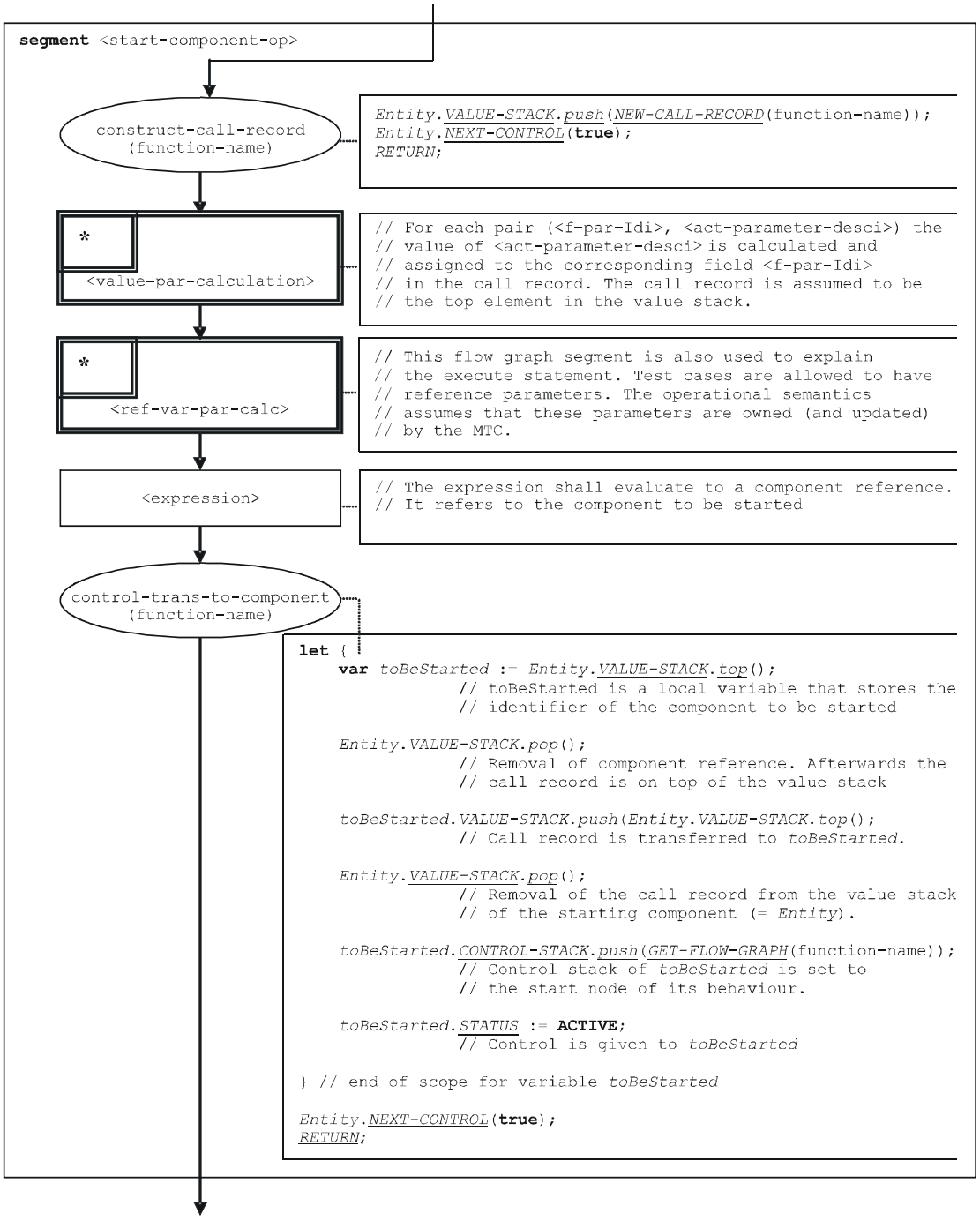
```
<component-expression>.start(<function-name>(<act-par-desc1>,..., <act-par-descn>))
```

تطلق العملية **start** المنطبقة على المكونات مكونة تم إنشاؤها حديثاً. واستخدام مرجع المكونة يحدد المكونة الواجب إطلاقها. ويمكن تخزين المرجع في متغيرة أو إعادتها بواسطة وظيفة ما، أي أنها عبارة يعطي تقييمها مرجع المكونة.

يشير الاسم <function-name> إلى اسم الوظيفة التي تحدد سلوك المكونة الجديدة وتصف <act-par-descr$_1$>, ..., <act-par-descr$_n$> قيم المعلمات الفعلية للوظيفة <function-name>. ولا تستعمل سوى معلمات القيم في الوظائف المشار إليها في العمليات **start** المنطبقة على المكونات. وتعطى أوصاف المعلمات الفعلية في شكل عبارات ينبغي تقييمها قبل التمكن من إجراء النداء. وتكون معالجة المعلمات بالقيم الرسمية والفعلية مماثلة لمعالجتها في النداءات الوظيفية (انظر الفقرة 24.9).

يحدد المقطع <start-component-op> في الشكل 120 تنفيذ العملية **start** المنطبقة على المكونات. وتنفذ هذه العملية في أربع مراحل. في المرحلة الأولى يتم إنشاء سجل النداء. وفي المرحلة الثانية تحسب قيم المعلمات الفعلية. وفي المرحلة الثالثة يُستخرج مرجع المكونة الواجب إطلاقها. وفي المرحلة الرابعة يعطى التحكم وسجل النداء إلى المكونة الجديدة.

ملاحظة – يتضمن المقطع في الشكل 120 معالجة المعلمات المرجعية (<ref-var-par-calc>). والمعلمات المرجعية ضرورية لشرح المعلمات المرجعية للاختبارات الأولية. وتفترض الدلالة التشغيلية، أن المكونة MTC تعالج هذه المعلمات.



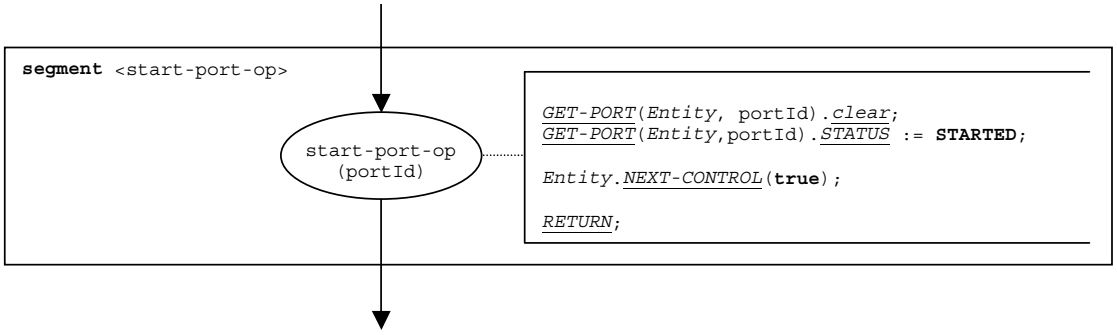
الشكل Z.143/120 – المقطع <start-component-op> في مخطط الانسياب

47.9 العملية start المنطبقة على المنافذ

يكون تركيب العملية **start** المنطبقة على المنافذ كما يلي:

```
<portId>.start
```

يحدد المقطع <start-port-op> في الشكل 121 تنفيذ العملية **start** المنطبقة على المنافذ.



الشكل Z.143/121 – المقطع <start-port-op> في مخطط الانسياب

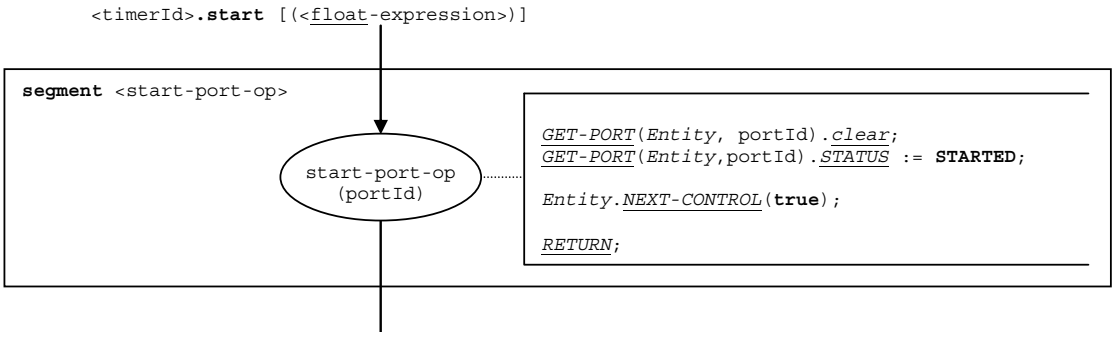
48.9 العملية start المنطبقة على المؤقتات

يكون تركيب العملية **start** المنطبقة على المؤقتات كما يلي:

```
<timerId>.start [(<float-expression>)]
```

تشير المعلمة الاختيارية <float-expression> للعملية **start** المنطبقة على المؤقتات إلى المدة الفعلية للمؤقت. وإذا لم توفر هذه المعلمة، تستخدم المدة بالتغيب في هذه العملية **start**. وينبغي أن يعطي تقييم العبارة قيمة نمطية **float**. وإذا توفرت المعلمة، ينبغي تقييم العبارة قبل تطبيق العملية **start**. وتوضع نتيجة التقييم في المكدس *VALUE-STACK* من *Entity*.

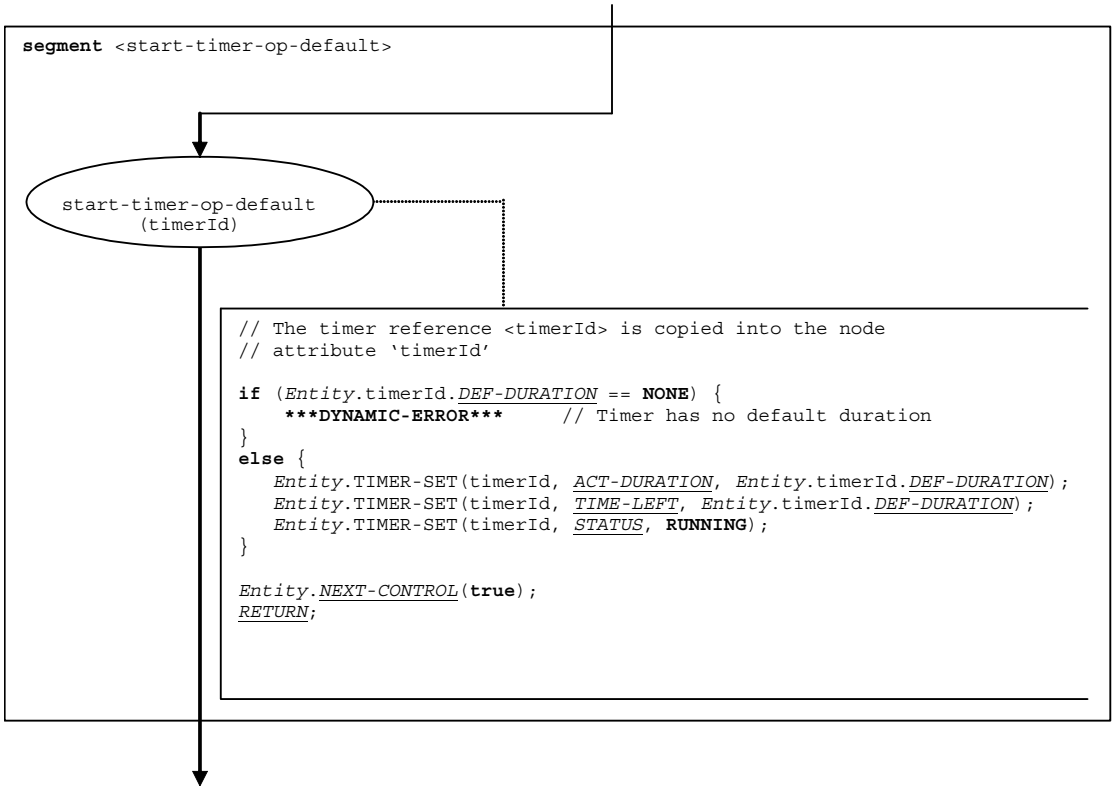
يحدد المقطع <start-timer-op> في الشكل 122 تنفيذ العملية **start** المنطبقة على المؤقتات.



الشكل Z.143/122 – المقطع <start-timer-op> في مخطط الانسياب

1.48.9 المقطع <start-timer-op-default>

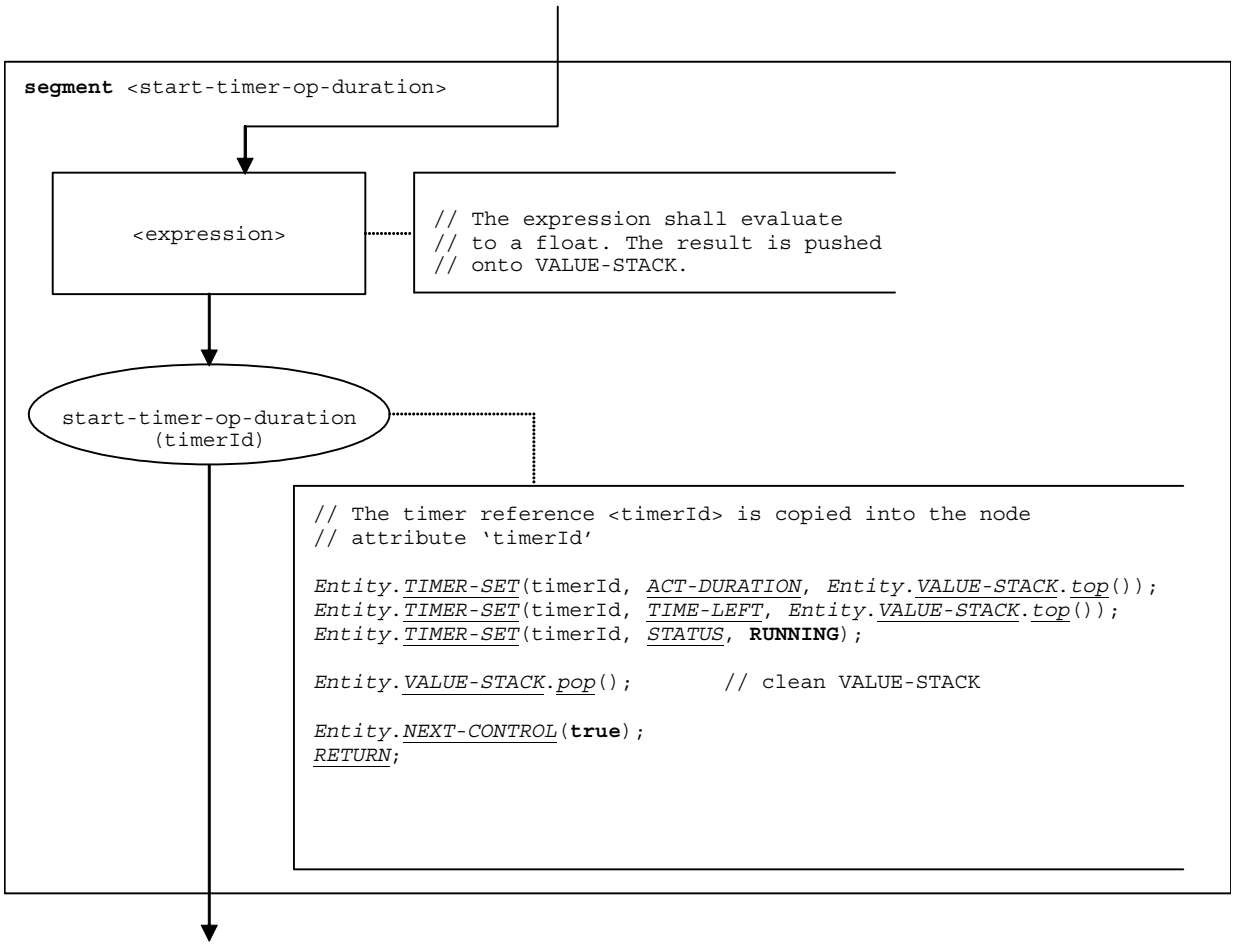
يحدد المقطع <start-timer-op-default> في الشكل 123 تنفيذ العملية **start** المنطبقة على المؤقتات مع القيمة بالتغيب.



الشكل Z.143/123 – المقطع <start-timer-op-default> في مخطط الانسياب

2.48.9 المقطع <start-timer-op-duration>

يحدد المقطع <start-timer-op-duration> في الشكل 124 تنفيذ العملية **start** المنطبقة على المؤقتات على أساس مدة معينة.



الشكل Z.143/124 – المقطع <start-timer-op-duration> في مخطط الانسياب

49.9 العملية stop المنطبقة على المكونات

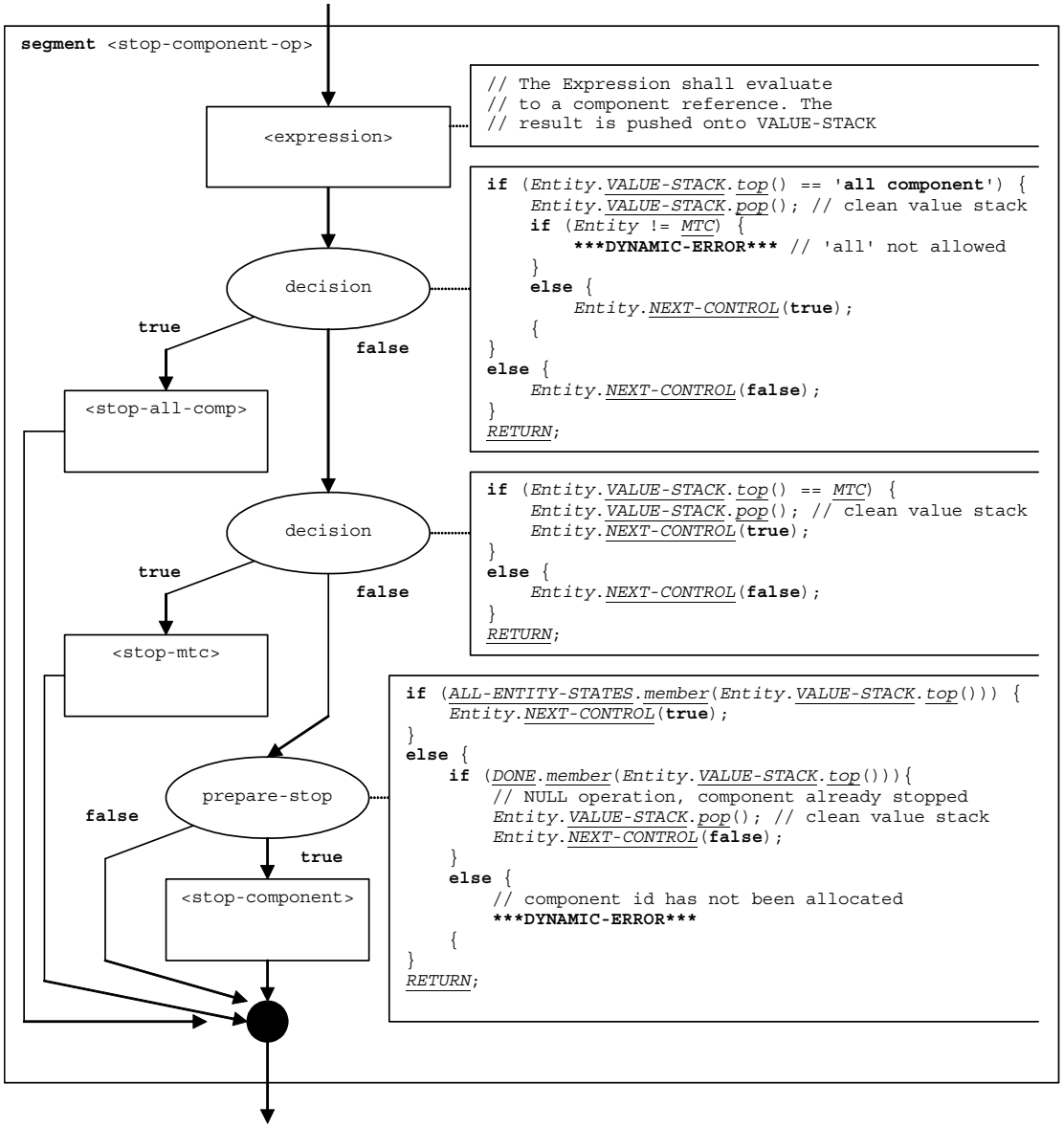
يكون تركيب العملية **stop** المنطبقة على المكونات كما يلي:

<component-expression>.**stop**

توقف العملية **stop** المنطبقة على المكونات المكونة المحددة. ويتم توقيف كل مكونات الاختبار، أي ينتهي الاختبار الأولي، إذا تم إيقاف المكون MTC (**mtc.stop**، مثلاً) أو توقف بنفسه (**self.stop**، مثلاً). ويمكن أن توقف المكونة MTC كل مكونات الاختيار الموازية باستخدام الكلمة الرئيسية **all**، أي **allcomponent.stop**.

يتم تحديد المكونة الواجب إيقافها بمرجع مكونة يقدم في شكل عبارة، قيمة أو وظيفة تعيد قيمة مثلاً. وتوخياً للتبسيط، تُعتبر الكلمة الرئيسية **'all component'** قيم خاصة للكيان <component-expression>. وتقيَّم العمليتان **mtc** و**self** وفقاً للفقرتين 33.9 و43.9.

ويحدد المقطع <stop-component-op> في الشكل 125 تنفيذ العملية **stop** المنطبقة على المكونات.



الشكل Z.143/125 – المقطع <stop-component-op> في مخطط الانسياب

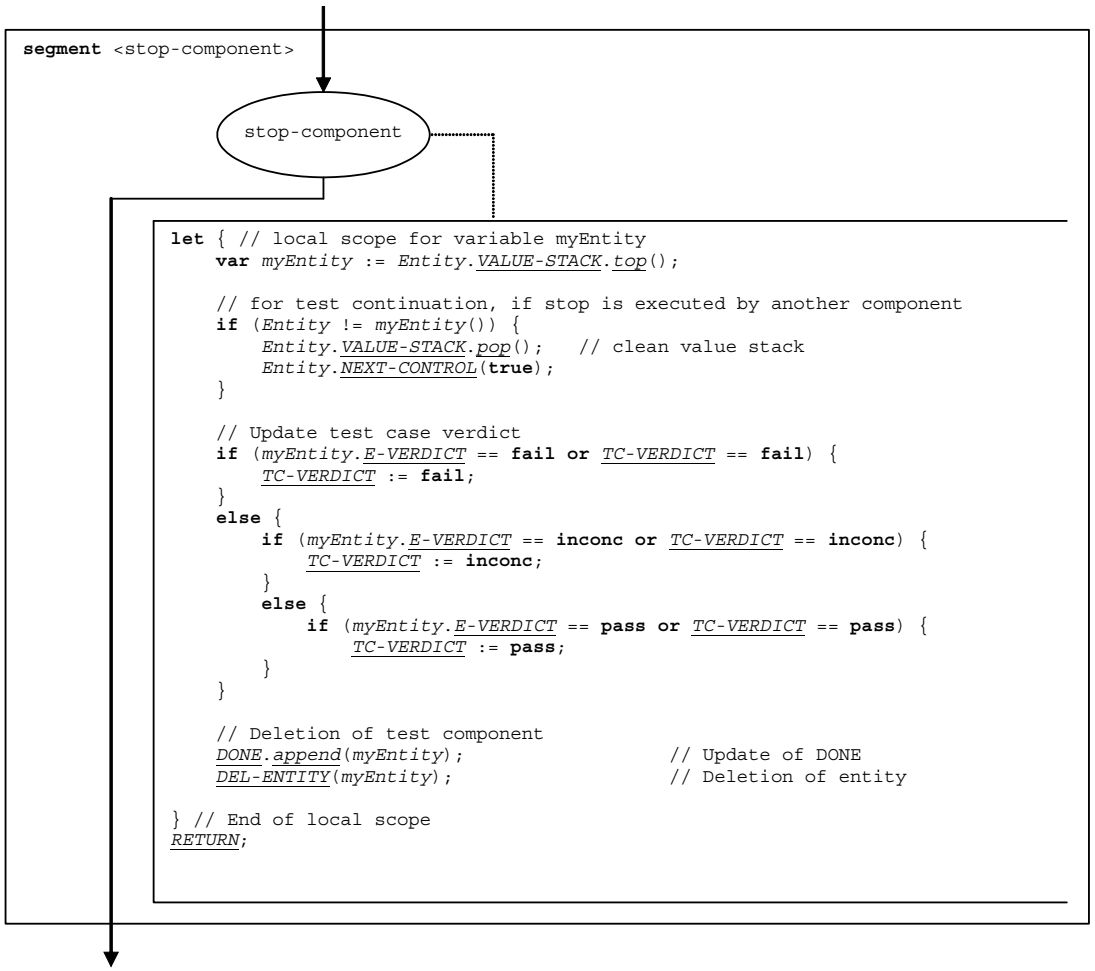
1.49.9 المقطع <stop-mtc>

يحدد المقطع <stop-mtc> في الشكل 126 توقيف المكونة MTC في اختبار أولي. ومن شأن ذلك أن يوقف الاختبار الأولي، أي يُحسب الحكم النهائي ثم يوضع في مكدّس قيم التحكم بالنموذج، ويتم تحرير كل الموارد وتفرّغ قائمة DONE لحالة النموذج وتنتهي جميع مكونات الاختبار بما فيها المكونة MTC.

الشكل Z.143/126 – المقطع <stop-mtc-op> في مخطط الانسياب

2.49.9 المقطع <stop-component>

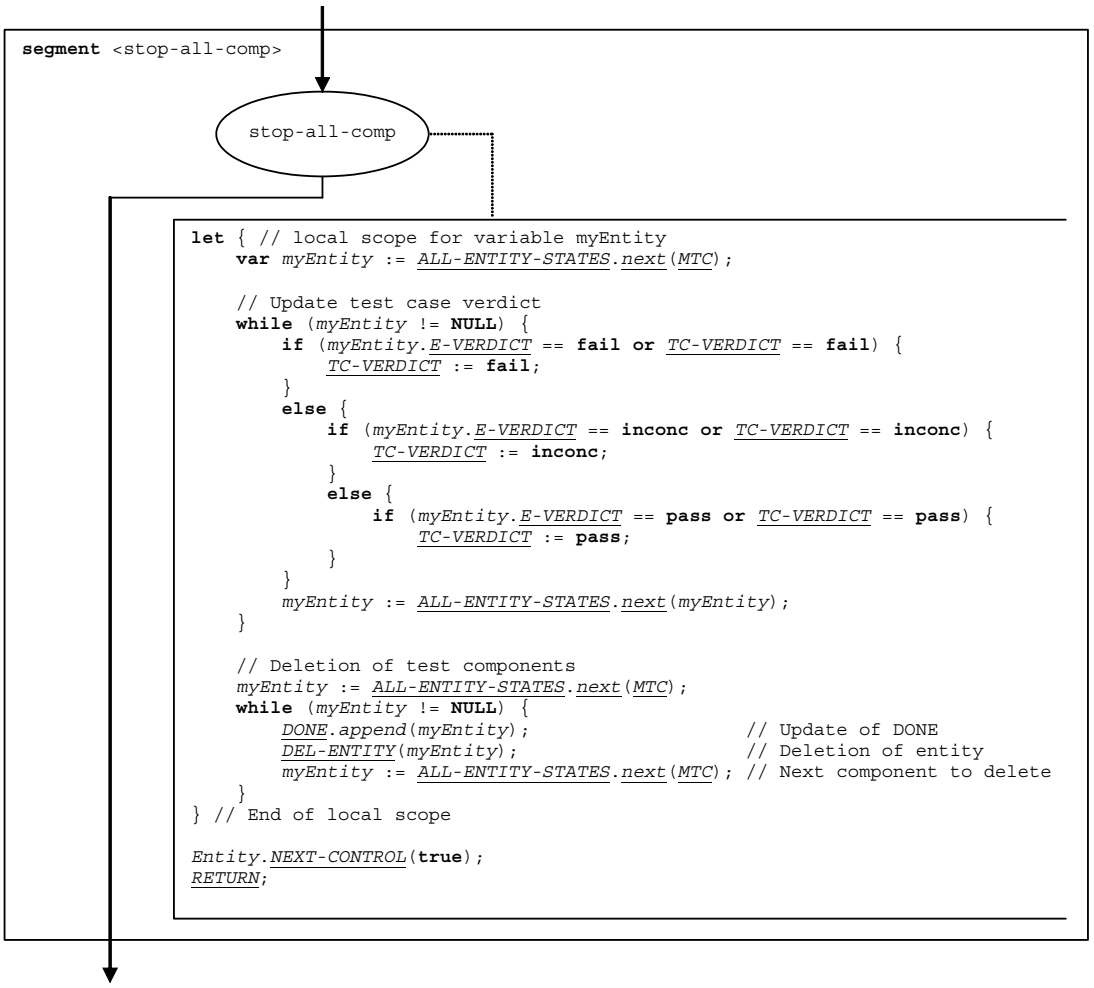
يصف المقطع <stop-component> في الشكل 127 توقيف مكونة اختبار موازية، أي لا المكونة MTC ولا التحكم بالنموذج. ومن شأن ذلك تحديث الاختبار الأولي *TC-VERDICT* وقائمة مكونات الاختبار المنتهية (*DONE*) وحذف المكونة من حالة النموذج. ويفترض المقطع <stop-component>، أن معرّف المكونة الواجب إيقافها موجود على رأس مكدّس قيم المكونة الذي ينفذ المقطع.



الشكل Z.143/127 – المقطع <stop-component> في مخطط الانسياب

3.49.9 المقطع <stop-all-comp>

يصف المقطع <stop-all-comp> في الشكل 128 توقيف كل مكونات الاختبار الموازية لاختبار أولي.



الشكل Z.143/128 – المقطع <stop-all-comp> في مخطط الانسياب

50.9 البيان stop المنطبق على التنفيذ

يكون تركيب البيان **stop** المنطبق على التنفيذ كما يلي:

```
stop
```

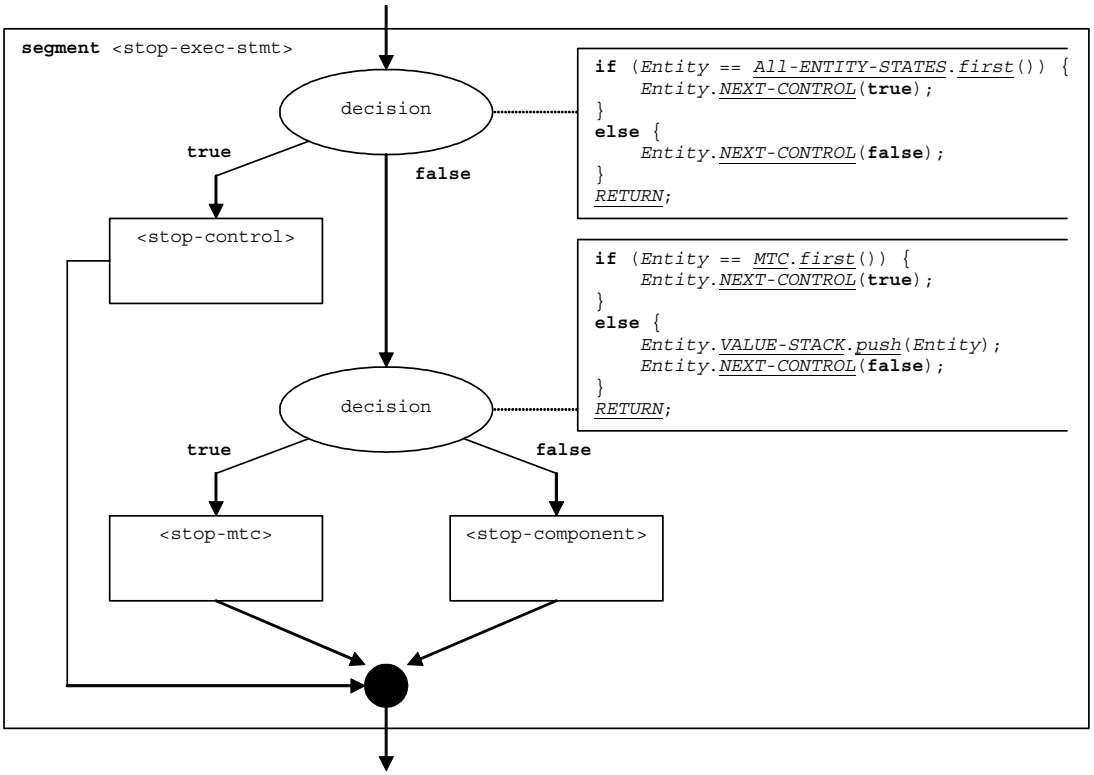
يتوقف أثر البيان **stop** المنطبق على التنفيذ على الكيان الذي ينفذه:

أ) إذا كان التحكم بالنموذج هو الذي ينفذه، فإن حملة الاختبارات تنتهي، أي أن كل مكونات الاختبار والتحكم بالنموذج تختفي من حالة النموذج.

ب) إذا كانت المكونة MTC هي التي تنفذه، يتم إيقاف تنفيذ كل مكونات الاختبار الموازية والمكونة MTC. ويتم تحديث حالة الاختبار الأولي العامة ثم توضع في مكدّس قيم التحكم بالنموذج. وأخيراً، يعطى التحكم إلى التحكم بالنموذج وتنتهي المكونة MTC.

ج) إذا كانت مكونة الاختبار هي التي تنفذه، يتم تحديث حالة الاختبار الأولي العامة *TC-VERDICT* والقائمة *DONE* العامة. ثم تختفي المكونة كلياً من النموذج.

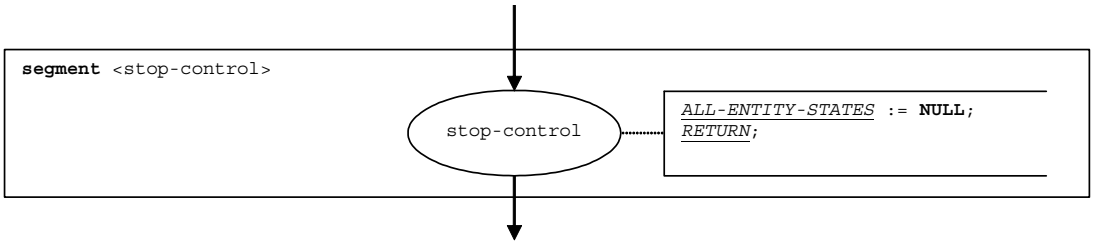
يصف المقطع <stop-exec-stmt> في الشكل 129 تنفيذ البيان **stop**.



الشكل Z.143/129 – المقطع <stop-exec-stmt> في مخطط الانسياب

1.50.9 المقطع <stop-control>

يصف المقطع <stop-control> في الشكل 130 توقيف التحكم بالنموذج. والأثر هو أن *ALL-ENTITY-STATES* يوضع على **NULL**، أي أنه تم استيفاء شرط انتهاء إجراء تقييم النموذج (انظر الفقرة 6.8).



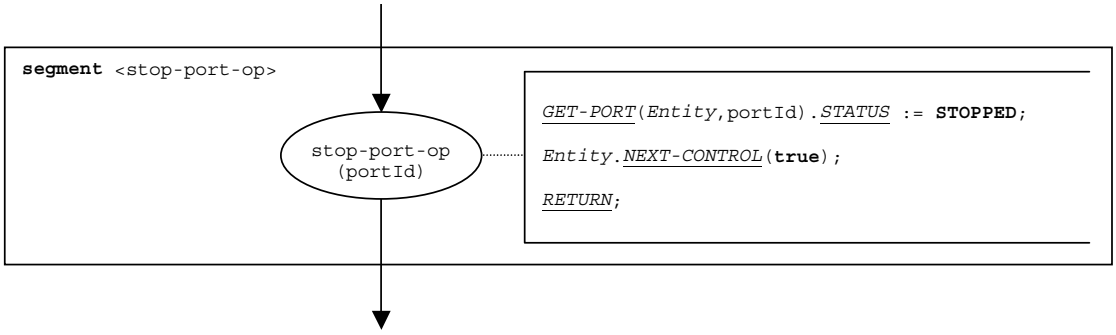
الشكل Z.143/130 – المقطع <stop-control-op> في مخطط الانسياب

51.9 العملية stop المنطبقة على المنافذ

يكون تركيب العملية **stop** المنطبقة على المنافذ كما يلي:

```
<portId>.stop
```

يحدد المقطع <stop-port-op> في الشكل 131 تنفيذ العملية **stop** المنطبقة على المنافذ.



الشكل Z.143/131 – المقطع <stop-port-op> في مخطط الانسياب

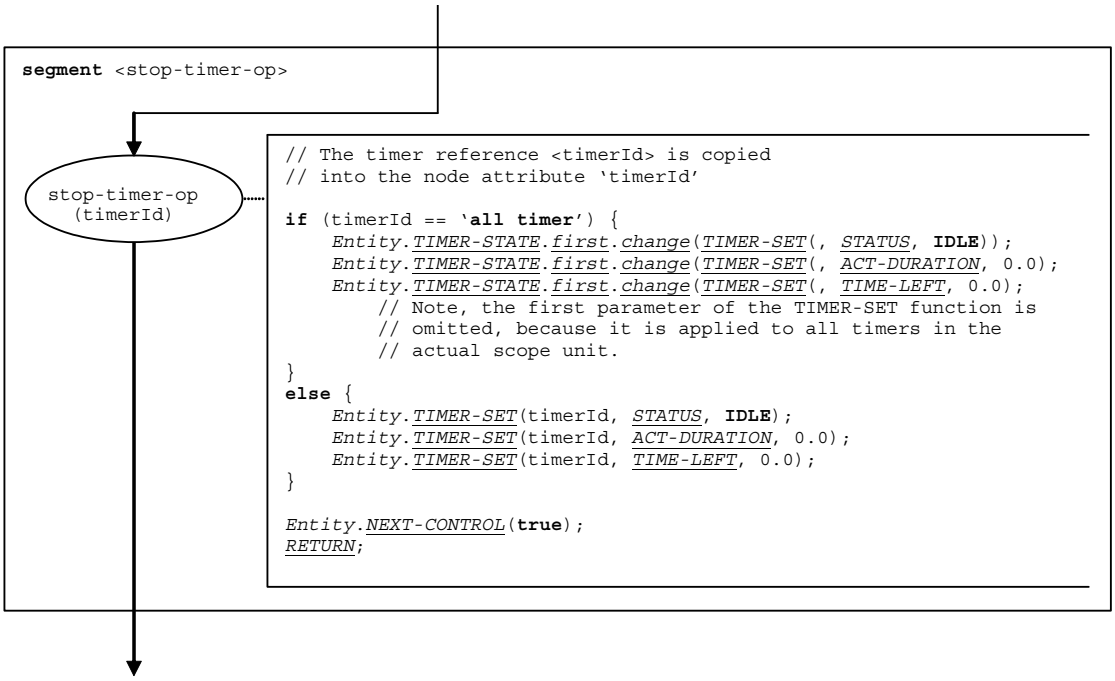
52.9 العملية stop المنطبقة على المؤقتات

يكون تركيب العملية **stop** المنطبقة على المؤقتات كما يلي:

```
<timerId>.stop
```

يحدد المقطع <stop-timer-op> في الشكل 132 تنفيذ العملية **stop** المنطبقة على المؤقتات.

تعالَج الكلمة الرئيسية **all** كقيمة خاصة من قيم timerId.



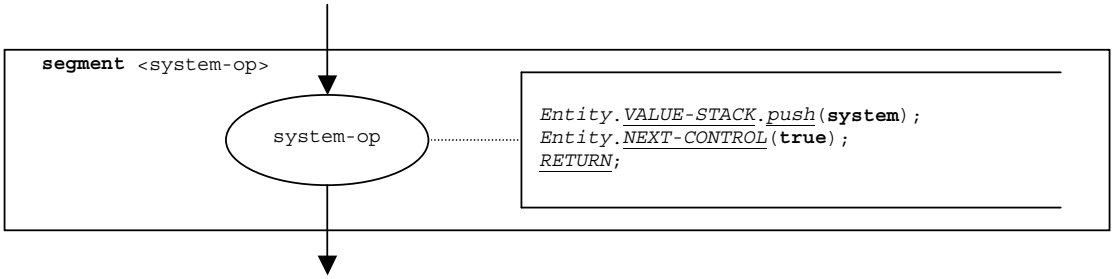
الشكل Z.143/132 – المقطع <stop-timer-op> في مخطط الانسياب

53.9 العملية system

يكون تركيب العملية **system** كما يلي:

```
system
```

يحدد المقطع <system-op> في الشكل 133 تنفيذ العملية **system**.



الشكل Z.143/133 – المقطع <system-op> في مخطط الانسياب

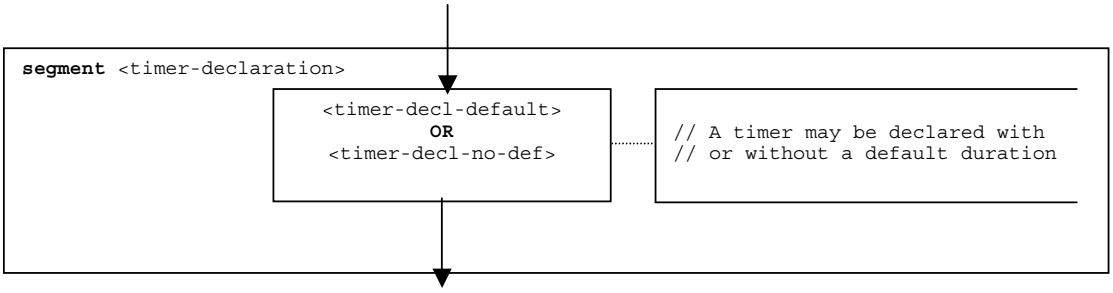
54.9 الإعلان timer

يكون تركيب الإعلان **timer** كما يلي:

timer <timerId> [:= <float-expression>]

إن أثر إعلان المؤقت هو إنشاء وصلة مؤقت جديدة. أما إعلان المدة بالتغيب فهو اختياري. وتُعتبر القيمة بالتغيب بأنها عبارة يعطي تقييمها قيمة من النمط **float**.

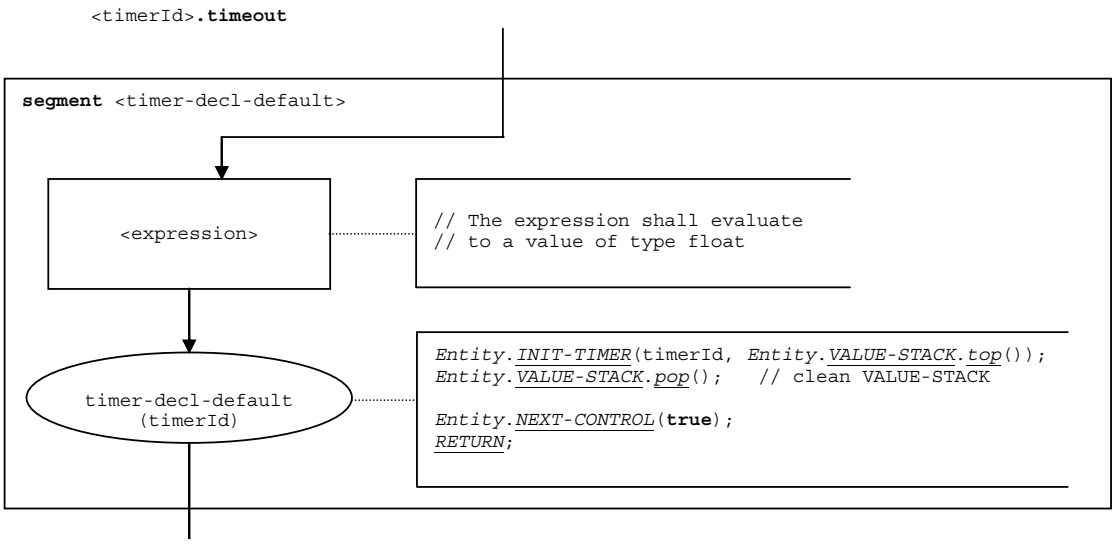
يحدد المقطع <timer-declaration> في الشكل 134 تنفيذ إعلان المؤقت.



الشكل Z.143/134 – المقطع <timer-declaration> في مخطط الانسياب

1.54.9 المقطع <timer-decl-default>

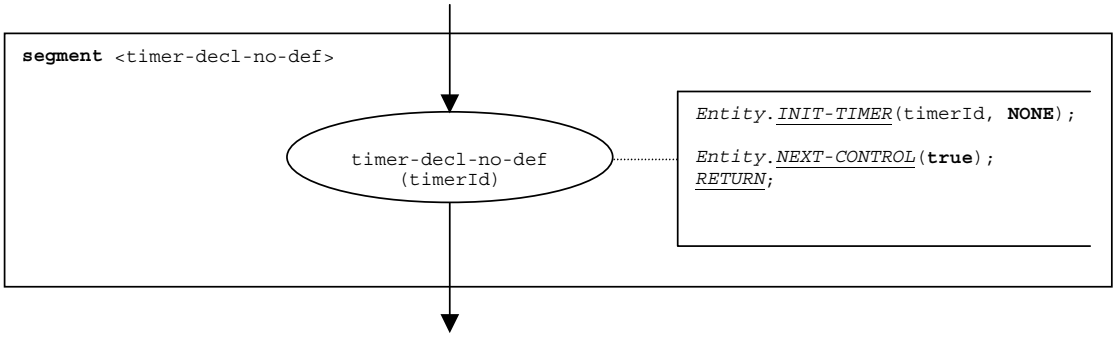
يحدد المقطع <timer-decl-default> في الشكل 135 تنفيذ إعلان المؤقت التي تحدد من أجله مدة بالتغيب في شكل عبارة.



الشكل Z.143/135 – المقطع <timer-decl-default> في مخطط الانسياب

2.54.9 المقطع <timer-decl-no-def>

يحدد المقطع <timer-decl-no-def> في الشكل 136 تنفيذ إعلان المؤقت التي لا تحدد من أجله أي مدة بالتغيب. أي إن مدة بالتغيب للمؤقت غير محددة.



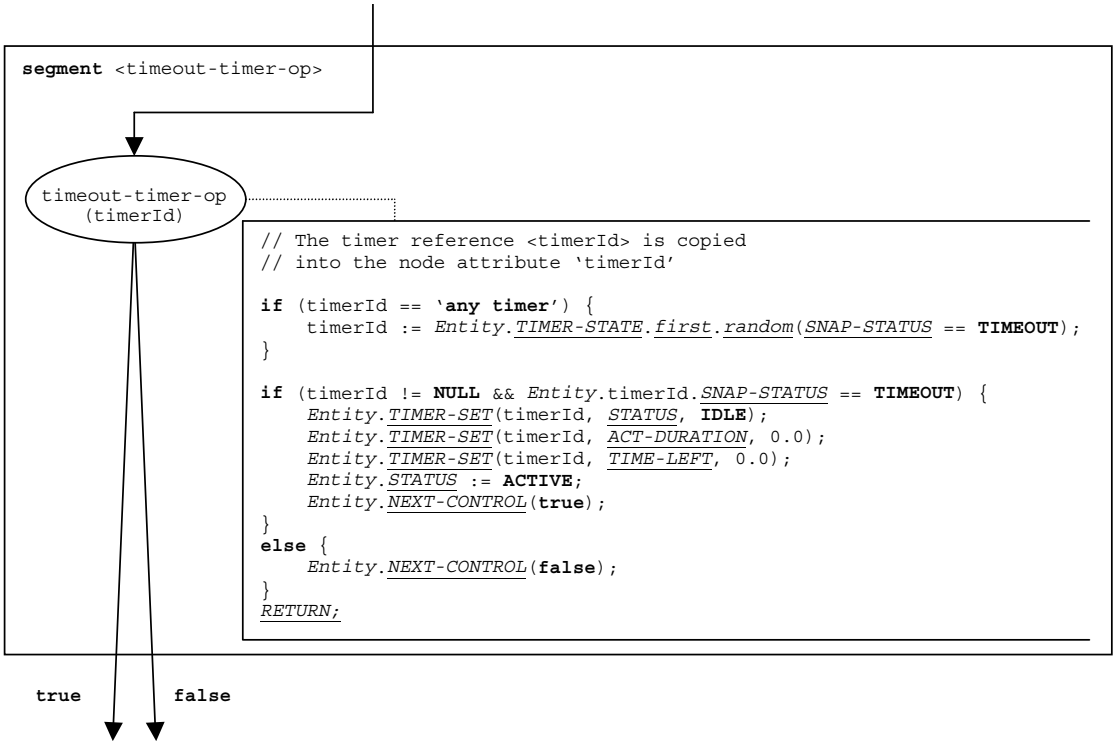
الشكل Z.143/136 – المقطع <timer-decl-no-def> في مخطط الانسياب

55.9 العملية timeout المنطبقة على المؤقتات

يكون تركيب العملية **timeout** المنطبقة على المؤقتات كما يلي:

```
<timerId>.timeout
```

يحدد المقطع <timeout-timer-op> في الشكل 137 تنفيذ العملية **timeout** المنطبقة على المؤقتات.



الملاحظة 1 – العملية **timeout** مبيتة في البيان **alt**. ويستند تقييمها إلى اللحظة الفعلية، أي يقوم القرار على أساس المدخل *SNAP-STATUS* لوصلة المؤقت. وإذا نجحت العملية **timeout**، أي *SNAP-STATUS* == TIMEOUT، يوضع المؤقت في الحالة IDLE وتتغير حالة المكونة من SNAPSHOT إلى ACTIVE.

الملاحظة 2 – عندما يعطي تقييم العملية **timeout** بيان **true** أو **false**، إما يستمر التنفيذ بموجب البيان الذي يتبع العملية timeout (الفرع **true**) أو يتعين التحقق من البديل التالي في البيان **alt** (الفرع **false**).

الملاحظة 3 – تعالَج الكلمة الرئيسية **any** كقيمة خاصة من قيم timerId.

الشكل Z.143/137 – المقطع <timeout-timer-op> في مخطط الانسياب

56.9 العملية unmap

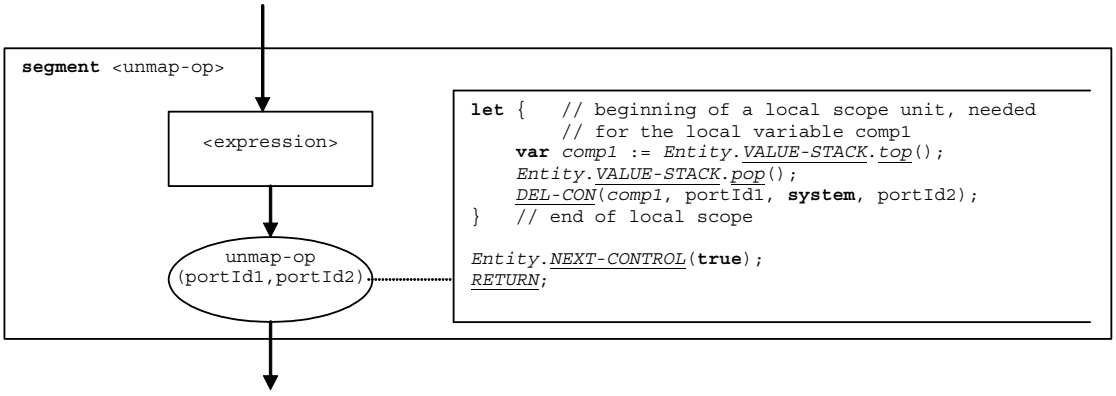
يكون تركيب العملية **unmap** كما يلي:

```
unmap(<component_expression>:<portId1>,system:<portId2>)
```

يعتبر المعرفان <portId1> و<portId2> معرفين لكل من منفذ مكونة الاختبار ومنفذ السطح البيني لنظام الاختبار. وتكون الإحالة إلى المكونة التي ينتمي إليها المعرف <portId1> بواسطة مرجع المكونة <component-expression>. ويمكن تخزين المرجع في متغيرات أو يمكن إعادتها من خلال وظيفة ما، أي إنها عبارة يعطي تقييمها مرجع المكونة. ويستخدم مكدّس القيم لتخزين مرجع المكونة.

ملاحظة – بالنسبة للعملية **unmap**، ليس من المهم إذا كان البيان **system**:<portId> معلمة أولى أم ثانية. وتوخياً للتبسيط، يفترض أنه دائماً المعلمة الثانية.

يحدد المقطع <unmap-op> في الشكل 138 تنفيذ العملية **unmap**.



الشكل Z.143/138 – المقطع <unmap-op> في مخطط الانسياب

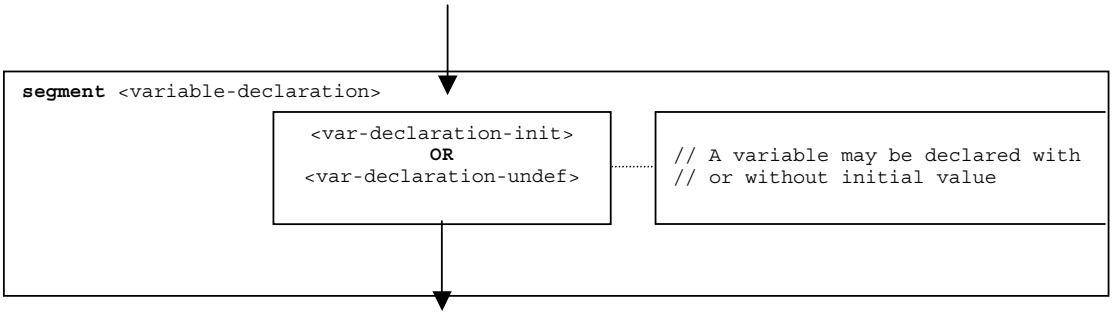
57.9 إعلان المتغيرة

يكون تركيب إعلان المتغيرة كما يلي:

var <varType> <varId> [:= <varType-expression>]

إن تدميث متغيرة ما من خلال توفير القيمة الأولية (في شكل عبارة) عملية اختيارية. وتُعتبر القيمة الأساسية عبارة يعطي تقييمها قيمة من نمط المتغيرة.

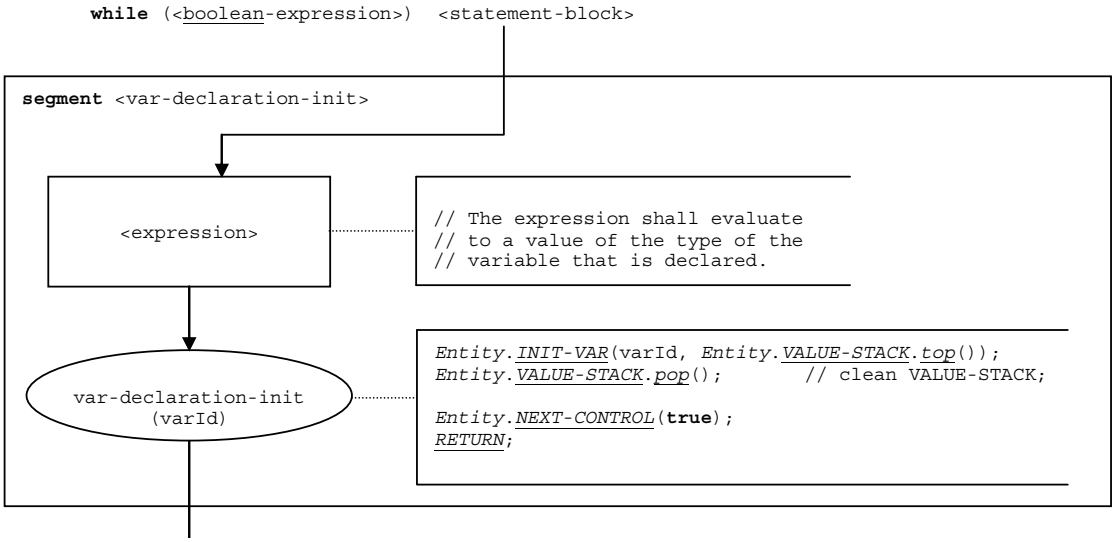
يحدد المقطع <variable-declaration> في الشكل 139 تنفيذ إعلان المتغيرة.



الشكل Z.143/139 – المقطع <variable-declaration> في مخطط الانسياب

1.57.9 المقطع <var-declaration-init>

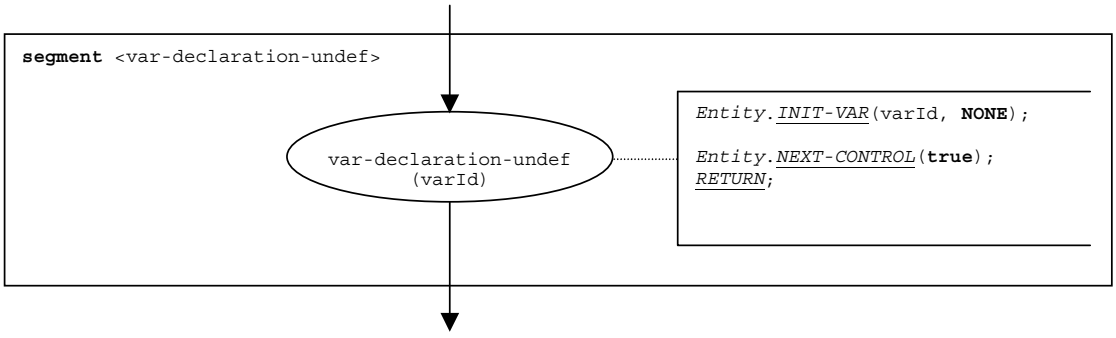
يحدد المقطع <var-declaration-init> في الشكل 140 تنفيذ إعلان المتغيرة التي تعطى لأجلها قيمة أولية في شكل عبارة.



الشكل Z.143/140 – المقطع <var-declaration-init> في مخطط الانسياب

2.57.9 المقطع <var-declaration-undef>

يحدد المقطع <var-declaration-undef> في الشكل 141 تنفيذ إعلان المتغيرة التي لم تعط لأجلها أي قيمة أولية، أي إن قيمة المتغيرة غير محددة.



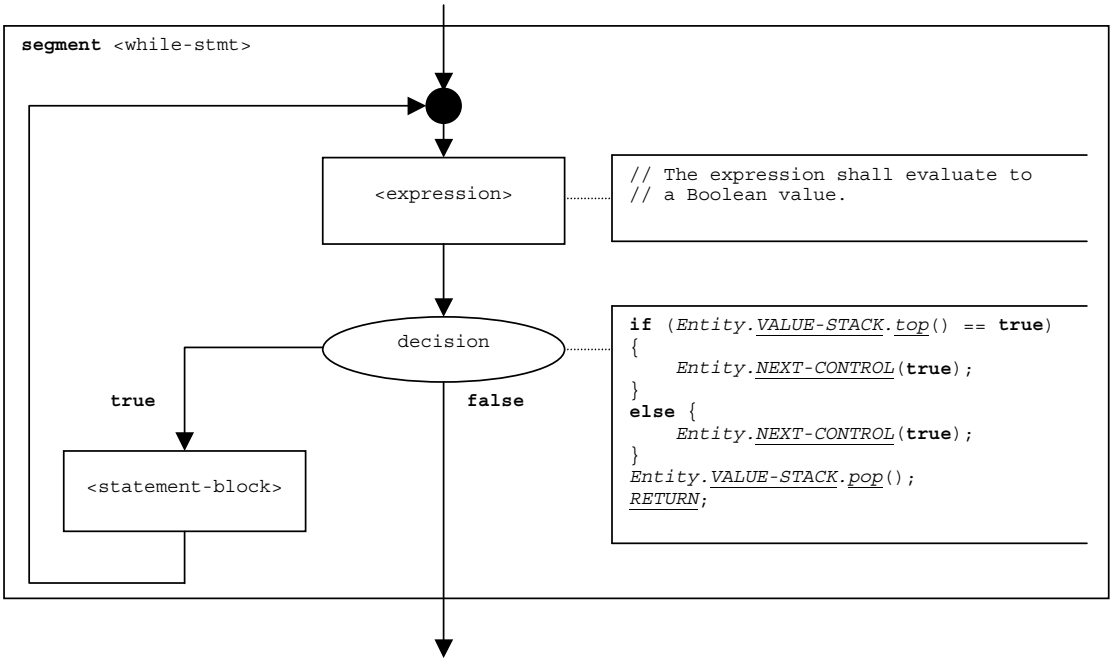
الشكل Z.143/141 – المقطع <var-declaration-undef> في مخطط الانسياب

58.9 البيان while

يكون تركيب البيان **while** كما يلي:

```
while (<boolean-expression>)  <statement-block>
```

يحدد المقطع <while-stmt> في الشكل 142 تنفيذ البيان **while**.



الشكل Z.143/142 – المقطع <while-stmt> في مخطط الانسياب

10 قوائم مكوّنات الدلالة التشغيلية

1.10 الوظائف والحالات

الاسم	الوصف	الفقرة
ACT-DURATION	المدة التي تم خلالها إطلاق مؤقت نشط	4.2.3.8
Add	عملية تنطبق على القوائم: إضافة بند كعنصر أول في القائمة	1.1.3.8
ADD-CON	إضافة وصلة إلى حالة منفذ	2.3.3.8
ALL-ENTITY-STATES	حالات المكونة في حالة النموذج	1.3.8
ALL-PORT-STATES	حالات المنفذ في حالة النموذج	1.3.8
append	عملية تنطبق على القوائم: إضافة بند كعنصر أخير في القائمة	1.1.3.8
APPLY-OPERATOR	تطبيق معاملات مثل +، – أو/	2.6.8
change	عملية تنطبق على القوائم: تغيير كل عناصر القائمة	1.1.3.8
clear	عملية تنطبق على المكدسات: تزال عناصر المكدس	1.2.3.8
clear	عملية تنطبق على صفوف الانتظار: تزال كل عناصر صف انتظار	2.3.3.8
clear-until	عملية تنطبق على المكدسات: تزال العناصر إلى أن يصبح بند محدد على رأس المكدس	1.2.3.8
CONNECTIONS-LIST	قائمة توصيلات المنفذ	3.3.8
CONSTRUCT-ITEM	إنشاء بند لإرساله	4.4.8
CONTINUE-COMPONENT	تتابع المكونة الحالية عملية التنفيذ	2.6.8
CONTROL-STACK	مكدِّس عقد مخطط الانسياب تشير إلى حالة التحكم الفعلي للكيان	2.3.8
DATA-STATE	حالة البيانات في حالة الكيان	2.3.8
DEF-DURATION	مدة المؤقت بالتغيب	4.2.3.8

الاسم	الوصف	الفقرة
DEFAULT-LIST	قائمة الأحوال بالتغيب النشيطة في حالة كيان	2.3.8
DEFAULT-POINTER	إشارة إلى الحالة بالتغيب الفعلية خلال تقييم الحالة بالتغيب	2.3.8
DEL-CON	إزالة توصيلة في حالة المنفذ	2.3.3.8
DEL-ENTITY	إزالة كيان من حالة النموذج	4.3.8
DEL-TIMER-SCOPE	إزالة مجال المؤقت	5.2.3.8
DEL-VAR-SCOPE	إزالة مجال المتغيرة	3.2.3.8
delete	عملية تنطبق على القوائم: حذف بند من القائمة	1.1.3.8
Dequeue	عملية تنطبق على صفوف الانتظار: حذف العنصر الأول من صف انتظار	2.3.3.8
DONE	معرفات مكونات الاختبار المنتهية (جزء من حالة النموذج)	1.3.8
E-VERDICT	نتيجة الاختبار المحلي لمكونة الاختبار	2.3.8
enqueue	عملية تنطبق على صفوف الانتظار: وضع بند كآخر عنصر في صف انتظار	2.3.3.8
first	عملية تنطبق على صفوف الانتظار: إعادة العنصر الأول في صف الانتظار	2.3.3.8
first	عملية تنطبق على القوائم: إعادة العنصر الأول في القائمة	1.1.3.8
GET-FLOW-GRAPH	استخراج عقدة البداية في مخطط الانسياب	7.2.8
GET-PORT	استخراج مرجع المنفذ	2.3.3.8
GET-REMOTE-PORT	استخراج مرجع المنفذ البعيد	2.3.3.8
GET-TIMER-LOC	استخراج موقع المؤقت	5.2.3.8
GET-UNIQUE-ID	إعادة معرف وحيد جديد عند استدعاء هذه العملية	2.6.8
GET-VAR-LOC	استخراج موقع المتغيرة	3.2.3.8
INIT-CALL-RECORD	تدميث متغيرات المعلمات للاتصال الإجرائي في وحدة المجال الفعلية لمكونة الاختبار	1.5.8
INIT-FLOW-GRAPHS	تدميث معالجة مخطط الانسياب	2.6.8
INIT-TIMER	توليد وصلة جديدة للمؤقت	5.2.3.8
INIT-TIMER-LOC	توليد وصلة جديدة للمؤقت مع موقع قائم	5.2.3.8
INIT-TIMER-SCOPE	تدميث وحدة مجال جديدة للمؤقت	5.2.3.8
INIT-VAR	توليد وصلة جديدة للمتغيرة	3.2.3.8
INIT-VAR-LOC	توليد وصلة جديدة للمتغيرة مع موقع قائم	3.2.3.8
INIT-VAR-SCOPE	تدمث وحدة مجال جديدة للمتغيرة	3.2.3.8
length	عملية تنطبق على القوائم: إعادة طول القائمة	1.1.3.8
M-CONTROL	معرف تحكم النموذج في حالة النموذج	1.3.8
MATCH-ITEM	التحقق من تطابق رسالة أو نداء أو إجابة أو استثناء مع عملية استقبال	5.4.8
member	عملية تنطبق على القوائم: التحقق من اندراج بند في قائمة	1.1.3.8
MTC	إشارة إلى المكون MTC في حالة النموذج	1.3.8
NEW-CALL-RECORD	استحداث سجل نداء لنداء وظيفي	1.5.8
NEW-ENTITY	استحداث حالة جديدة للكيان	1.2.3.8
NEW-PORT	استحداث منفذ جديد	2.3.3.8
NEXT	استخراج العقدة التالية لعقدة ما في رسم التدفق	6.1.8
next	عملية تنطبق على القوائم: إعادة العنصر التالي في القائمة	1.1.3.8
NEXT-CONTROL	إزالة العقدة العليا في مخطط الانسياب من مكدس التحكم ووضع العقدة التالية من مخطط الانسياب في مكدس التحكم	1.2.3.8
OWNER	صاحب المنفذ	3.3.8
pop	عملية تنطبق على المكدسات: إزالة بند ما من المكدّس	1.2.3.8
PORT-NAME	اسم المنفذ	3.3.8
push	عملية تنطبق المكدسات: وضع بند ما في المكدّس	1.2.3.8
Random	عملية تنطبق على القوائم: إعادة عنصر ما عشوائياً إلى القائمة	1.1.3.8
REMOTE-ENTITY	كيان بعيد في توصيلة في حالة المنفذ	1.3.3.8
REMOTE-PORT-NAME	اسم المنفذ في توصيلة في حالة المنفذ	1.3.3.8

الاسم	الوصف	الفقرة
RETRIEVE-INFO	استخراج المعلومات مما يرد من رسالة أو نداء أو إجابة أو استثناء	6.4.8
RETURN	إعادة التحكم إلى إجراء تقييم النموذج	2.6.8
SNAP-ACTIVE	عدد مكونات الاختبار النشطة عندما تأخذ المكونة MTC لحظة فعلية (جزء من حالة المكون)	1.3.8
SNAP-DONE	قائمة مكونات الاختبار المنـــتهية عند لحظة فعلية	2.3.8
SNAP-PORTS	توفير وظيفة اللحظة الفعلية، أي تحديث SNAP-VALUE	2.3.3.8
SNAP-STATUS	حالة المؤقت عند اللحظة الفعلية	4.2.3.8
SNAP-TIMER	توفير وظيفة اللحظة الفعلية، أي تحديث SNAP-VALUE وSNAP-STATUS	5.2.3.8
SNAP-VALUE	قيمة المؤقت عند اللحظة الفعلية	4.2.3.8
SNAP-VALUE	بالنسبة إلى دلالة اللحظة الفعلية، عملية التحديث عند إجراء اللحظة الفعلية	3.3.8
STATUS	حالة التحكم بالنموذج أو مكونة الاختبار (**ACTIVE** أو **SNAPSHOT** أو **REPEAT** أو **BLOCKED**)	2.3.8
STATUS	الحالة (**IDLE** أو **RUNNING** أو **TIMEOUT**) للمؤقت	4.2.3.8
STATUS	الحالة (**STARTED** أو **STOPPED**) للمنفذ	3.3.8
TC-VERDICT	نتيجة الاختبار الأولي في حالة النموذج	1.3.8
TIME-LEFT	المدة المتبقية قبل انتهاء المؤقت	4.2.3.8
TIMER-GUARD	مؤقت الحماية للبيانات execute والعمليات **call**	2.3.8
TIMER-NAME	اسم المؤقت	4.2.3.8
TIMER-SET	تحديد قيم المؤقت	5.2.3.8
TIMER-STATE	حالة المؤقت في حالة الكيان	2.3.8
top	عملية تنطبق على المكدسات: إعادة البند الأعلى إلى المكدس	1.2.3.8
UPDATE-REMOTE-REFERENCES	تحديث المؤقتات والمتغيرات التي لها نفس الموقع في كيانات مختلفة لإعطائها القيمة نفسها	4.3.8
VALUE	قيمة المتغيرة	2.2.3.8
VALUE-QUEUE	صف انتظار المنفذ	3.3.8
VALUE-STACK	مكدسات القيم لتخزين نتائج العبارات والمعاملات والعمليات والوظائف	2.3.8
VAR-NAME	اسم المتغيرة	2.2.3.8
VAR-SET	تحديد قيمة المتغيرة	3.2.3.8
DYNAMIC-ERROR	وصف حدوث الخطأ الدينامي	2.6.8
Identifier	معرف وحيد لمكونة الاختبار	2.3.8
location	استيعاب وحدات المجال والمعلمات المرجعية ومعلمات المؤقت. عبارة عن موقع لتخزين المؤقتات والمتغيرات.	4.2.3.8، 2.2.3.8

2.10 الكلمات الرئيسية الخاصة

الاسم	الوصف	الفقرة
ACTIVE	*STATUS* لحالة كيان	2.3.8
BLOCKED	*STATUS* لحالة كيان	2.3.8
IDLE	*STATUS* لحالة مؤقت	4.2.3.8
MARK	يستخدم كعلامة للكيان *VALUE-STACK*	2.3.8
NONE	يستخدم لوصف قيمة غير محددة	3.2.3.8، 5.2.3.8، 2.3.3.8
NULL	قيمة رمزية للمؤشرات والأنماط الشبيهة بها للإشارة إلى عدم معالجة أي أمر	1.1.3.8، 1.2.3.8، 3.3.8، 2.3.3.8، 1.1.6.8
REPEAT	*STATUS* لحالة كيان	2.3.8
RUNNING	*STATUS* لحالة مؤقت	4.2.3.8
SNAPSHOT	*STATUS* لحالة كيان	2.3.8
STARTED	*STATUS* لحالة منفذ	3.3.8
STOPPED	*STATUS* لحالة منفذ	3.3.8
TIMEOUT	*STATUS* لحالة مؤقت	4.2.3.8

3.10 مخططات الانسياب لأوصاف السلوك TTCN-3

	المرجع	
	الشكل	الفقرة
التحكم في النموذج	18	2.2.8
الاختبارات الأولية	19	3.2.8
الوظائف	20	4.2.8
الخطوات البديلة	21	5.2.8
تعاريف نمط المكونة	22	6.2.8

4.10 مقاطع مخطط الانسياب

المعرف	الهيكلية TTCN-3 ذات الصلة	المرجع	
		الشكل	الفقرة
<action-stmt>	البيان **action**	36	1.9
<activate-stmt>	البيان activate	37	2.9
<alt-stmt>	البيان **alt**	38	3.9
<altstep-call>	استدعاء خطوة بديلة	44	4.9
<altstep-call-branch>	بيان **alt**	41	3.3.9
<assignment-stmt>	بيان assignment	45	5.9
<b-call-with-duration>	العملية **call**	52	4.6.9
<b-call-without-duration>	العملية **call**	51	3.6.9
<blocking-call-op>	العملية **call**	47	6.9
<call-op>	العملية **call**	46	6.9
<call-reception-part>	العملية **call**	53	5.6.9
<catch-op>	العملية **catch**	55	7.9
<catch-timeout-exception>	العملية **call**	54	6.6.9
<check-op>	العملية **check**	56	8.9
<check-with-sender>	العملية **check**	57	1.8.9
<check-without-sender>	العملية **check**	58	2.8.9
<clear-port-op>	العملية **clear** المنطبقة على المنافذ	59	9.9
<connect-op>	العملية **connect**	60	10.9
<constant-definition>	تعريف constant	61	11.9
<create-op>	العملية **create**	62	12.9
<deactivate-stmt>	البيان deactivate	63	13.9
<default-evocation>	البيان **alt**	43	5.3.9
<disconnect-op>	العملية disconnect	64	14.9
<do-while-stmt>	البيان do-while	65	15.9
<done-component-op>	العملية **done** المنطبقة على المكونات	66	16.9
<else-branch>	البيان **alt**	42	4.3.9
<execute-stmt>	البيان **execute**	67	17.9
<execute-timeout>	البيان **execute**	69	2.17.9
<execute-without-timeout>	البيان **execute**	68	1.17.9
<expression>	عبارة	70	18.9
<finalize-component-init>	مستخدم في تعاريف نمط المكونات	75	19.9
<for-stmt>	البيان **for**	79	23.9
<func-op-call>	عبارة	73	3.18.9

المعرف	الهيكلية TTCN-3 ذات الصلة	المرجع	
		الشكل	الفقرة
<function-call>	استدعاء وظيفة	80	24.9
<getcall-op>	العملية **getcall**	86	25.9
<getreply-op>	العملية **getreply**	87	26.9
<getverdict-op>	العملية getverdict	88	27.9
<goto-stmt>	البيان **goto**	89	28.9
<if-else-stmt>	البيان **if-else**	90	29.9
<init-component-scope>	مستخدم في تعاريف نمط المكونات	76	20.9
<label-stmt>	البيان **label**	91	30.9
<lit-value>	عبارة	71	1.18.9
<log-stmt>	البيان **log**	92	31.9
<map-op>	العملية **map**	93	32.9
<mtc-op>	العملية **mtc**	94	33.9
<nb-call-without-receiver>	العملية **call**	50	2.6.9
<nb-call-with-receiver>	العملية **call**	49	1.6.9
<non-blocking-call-op>	العملية **call**	48	6.9
<operator-appl>	عبارة	74	4.18.9
<parameter-handling>	معالجة معلمات الوظائف والخطوات البديلة والاختبارات الأولية	77	21.9
<port-declaration>	إعلان المنفذ	95	34.9
<predef-ext-func-call>	استدعاء وظيفة (استدعاء وظيفة محددة مسبقاً أو خارجية)	85	5.24.9
<raise-op>	العملية **raise**	96	35.9
<raise-with-receiver-op>	العملية **raise**	97	1.35.9
<raise-without-receiver-op>	العملية **raise**	98	2.35.9
<read-timer-op>	العملية **read** المنطبقة على المؤقتات	99	36.9
<receive-assignment>	العملية **receive**	103	3.37.9
<receive-op>	العملية **receive**	100	37.9
<receive-with-sender>	العملية **receive**	101	1.37.9
<receive-without-sender>	العملية **receive**	102	2.37.9
<receiving-branch>	البيان **alt**	40	2.3.9
<ref-par-var-calc>	استدعاء وظيفة (معالجة المعلمات المرجعية)	82	2.24.9
<ref-par-timer-calc>	استدعاء وظيفة (معالجة معلمات المؤقت)	83	3.24.9
<repeat-stmt>	البيان **repeat**	104	38.9
<reply-op>	العملية **reply**	105	39.9
<reply-with-receiver-op>	العملية **reply**	106	1.39.9
<reply-without-receiver-op>	العملية **reply**	107	2.39.9
<return-stmt>	البيان **return**	108	40.9
<return-with-value>	البيان **return**	109	1.40.9
<return-without-value>	البيان **return**	110	2.40.9
<running-component-op>	العملية **running** المنطبقة على المكونات	111	41.9
<running-comp-act>	العملية **running** المنطبقة على المكونات	112	1.41.9
<running-comp-snap>	العملية **running** المنطبقة على المكونات	113	2.41.9
<running-timer-op>	العملية **running** المنطبقة على المؤقتات	114	42.9
<self-op>	العملية **self**	115	43.9
<send-op>	العملية **send**	116	44.9
<send-with-receiver-op>	العملية **send**	117	1.44.9

المعرف	الهيكلية TTCN-3 ذات الصلة	المرجع	
		الشكل	الفقرة
<send-without-receiver-op>	العملية **send**	118	2.44.9
<setverdict-op>	العملية setverdict	119	45.9
<start-component-op>	العملية **start** المنطبقة على المكونات	120	46.9
<start-port-op>	العملية **start** المنطبقة على المنافذ	121	47.9
<start-timer-op>	العملية **start** المنطبقة على المؤقتات	122	48.9
<start-timer-op-default>	العملية **start** المنطبقة على المؤقتات	123	1.48.9
<start-timer-op-duration>	العملية **start** المنطبقة على المؤقتات	124	2.48.9
<statement-block>	مجموعة البيانات في البيانات المركبة	78	22.9
<stop-component-op>	العملية **stop** المنطبقة على المكونات	125	49.9
<stop-mtc>	العملية **stop** المنطبقة على المكونات (إيقاف المكونة MTC)	126	1.49.9
<stop-component>	العملية **stop** المنطبقة على المكونات (إيقاف مكونة واحدة للاختبار)	127	2.49.9
<stop-all-comp>	العملية **stop** المنطبقة على المكونات (إيقاف جميع المكونات)	128	3.49.9
<stop-exec-stmt>	بيان التنفيذ **stop**	129	50.9
<stop-control>	بيان التنفيذ **stop** (توقيف التحكم في النموذج)	130	1.50.9
<stop-port-op>	العملية **stop** المنطبقة على المنافذ	131	51.9
<stop-timer-op>	العملية **stop** المنطبقة على المؤقتات	132	52.9
<system-op>	العملية **system**	133	53.9
<take-snapshot>	البيان **alt**	39	1.3.9
<timeout-timer-op>	العملية **timeout**	137	55.9
<timer-declaration>	إعلان المؤقت	134	54.9
<timer-decl-default>	إعلان المؤقت	135	1.54.9
<timer-decl-no-def>	إعلان المؤقت	136	2.54.9
<timeout-timer-op>	العملية **timeout**	137	55.9
<unmap-op>	العملية **unmap**	138	56.9
<user-def-func-call>	استدعاء وظيفة (استدعاء وظيفة يحددها المستعمل)	84	4.24.9
<value-par-calculation>	استدعاء وظيفة (معالجة المعلمات بحسب القيمة)	81	1.24.9
<var-declaration-init>	إعلان المتغيرة	140	1.57.9
<var-declaration-undef>	إعلان المتغيرة	141	2.57.9
<var-value>	عبارة	72	2.18.9
<variable-declaration>	إعلان المتغيرة	139	57.9
<while-stmt>	البيان **while**	140	58.9

سلاسل التوصيات الصادرة عن قطاع تقييس الاتصالات

السلسلة A	تنظيم العمل في قطاع تقييس الاتصالات
السلسلة D	المبادئ العامة للتعريفة
السلسلة E	التشغيل العام للشبكة والخدمة الهاتفية وتشغيل الخدمات والعوامل البشرية
السلسلة F	خدمات الاتصالات غير الهاتفية
السلسلة G	أنظمة الإرسال ووسائطه والأنظمة والشبكات الرقمية
السلسلة H	الأنظمة السمعية المرئية والأنظمة متعددة الوسائط
السلسلة I	الشبكة الرقمية متكاملة الخدمات
السلسلة J	الشبكات الكبلية وإرسال إشارات تلفزيونية وبرامج صوتية وإشارات أخرى متعددة الوسائط
السلسلة K	الحماية من التداخلات
السلسلة L	إنشاء الكبلات وغيرها من عناصر المنشآت الخارجية وتركيبها وحمايتها
السلسلة M	إدارة الاتصالات بما في ذلك شبكة إدارة الاتصالات (TMN) وصيانة الشبكات
السلسلة N	الصيانة: الدارات الدولية لإرسال البرامج الإذاعية الصوتية والتلفزيونية
السلسلة O	مواصفات تجهيزات القياس
السلسلة P	نوعية الإرسال الهاتفي والمنشآت الهاتفية وشبكات الخطوط المحلية
السلسلة Q	التبديل والتشوير
السلسلة R	الإرسال البرقي
السلسلة S	التجهيزات المطرافية للخدمات البرقية
السلسلة T	المطاريف الخاصة بالخدمات التلماتية
السلسلة U	التبديل البرقي
السلسلة V	اتصالات المعطيات على الشبكة الهاتفية
السلسلة X	شبكات المعطيات والاتصالات بين الأنظمة المفتوحة ومسائل الأمن
السلسلة Y	البنية التحتية العالمية للمعلومات وملامح بروتوكول الإنترنت وشبكات الجيل التالي
السلسلة Z	**اللغات والجوانب العامة للبرمجيات في أنظمة الاتصالات**

طُبعت في سويسرا
جنيف، 2010