ENTERPRISE ARCHITECT

Série de Guides d'Utilisateur

Testpoints

Author: Sparx Systems
Date: 7/11/2024
Version: 17.0

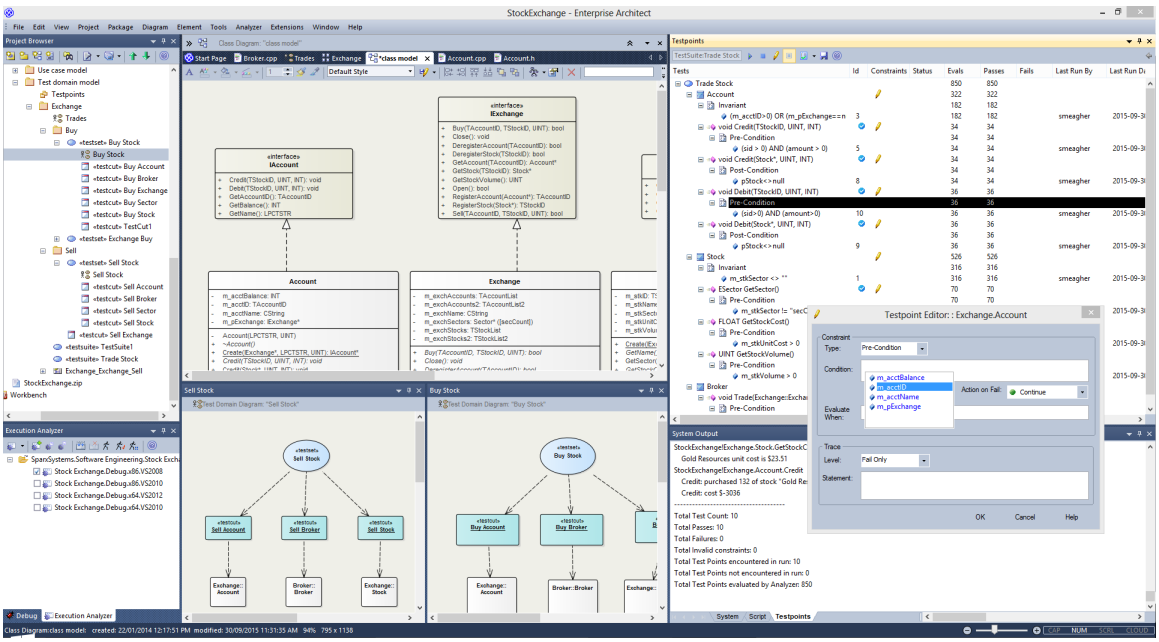
CRÉÉ AVEC ENTERPRISE ARCHITECT

Table des Matières

Testpoints

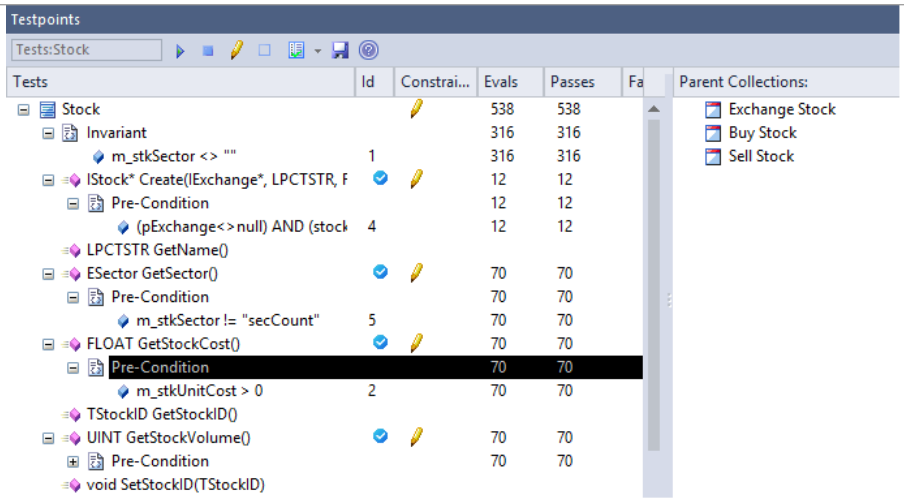
Testpoints présentent un schéma par lequel les contraintes et les règles régissant le comportement des objets peuvent être extraites du modèle et appliquées à une ou plusieurs applications. Les avantages offerts par des schémas tels que celui-ci sont la tolérance aux changements de code : l'ajout et la soustraction de lignes d'une fonction n'ont aucun effet sur les contraintes qui la régissent. Un autre avantage est que les modifications apportées aux règles de comportement ne nécessitent pas de modification correspondante du code source ; *ce qui signifie que rien ne doit être recompilé !*

De plus, la possibilité de vérifier plusieurs applications à l'aide d'un seul domaine de test est une question simple plutôt que contraignante. Le domaine Test est à la fois un modèle logique et relationnel ; les contraintes du modèle de classe peuvent être partitionnées avec des coupes Test . Celles-ci peuvent être agrégées simplement dans Ensembles Test et des suites Test à l'aide de connecteurs. En raison du découplage du domaine Test de la base de code, il suffit de choisir des boutons pour exécuter un programme normalement ou l' exécuter pour un domaine Test spécifique. Ce système offre également des avantages pratiques dans la mesure où aucune instrumentation n'est requise. Les résultats Test sont affichés dans la fenêtre de rapport pendant l' exécuter , en temps réel, pendant l'exécution du programme. Ces résultats peuvent être conservés et examinés à tout moment dans la dialogue « Détails Test » ou à l'aide des fonctionnalités de documentation d' Enterprise Architect .



Fonctionnalités

Fonctionnalité	Détails
Composition Testpoint	La composition Testpoint est effectuée à l'aide de la fenêtre Testpoints . La fenêtre Testpoints est contextuelle et affiche le domaine Test de l'élément sélectionné dans la fenêtre Navigateur ou diagramme . La sélection d'une seule classe affiche la structure de la classe. Une icône en forme de crayon s'affiche en regard des classes et des méthodes qui ont des contraintes existantes. Lorsque vous sélectionnez un Test coupe, d'ensemble ou Test suite, la fenêtre Testpoints affiche la structure complète du domaine, y compris toutes les classes qui composent le domaine. Note : vous pouvez parcourir la hiérarchie du domaine à l'aide du volet « Navigation » sur la droite. Testpoints sont composés sous forme d'expressions, à l'aide des noms de variables des membres de la classe. Le raccourci Intelli-sense Ctrl+Espace est disponible dans l'éditeur pour vous aider à les trouver. Les expressions qui donnent la valeur True sont considérées comme une réussite. Le retour de False est considéré comme un échec.



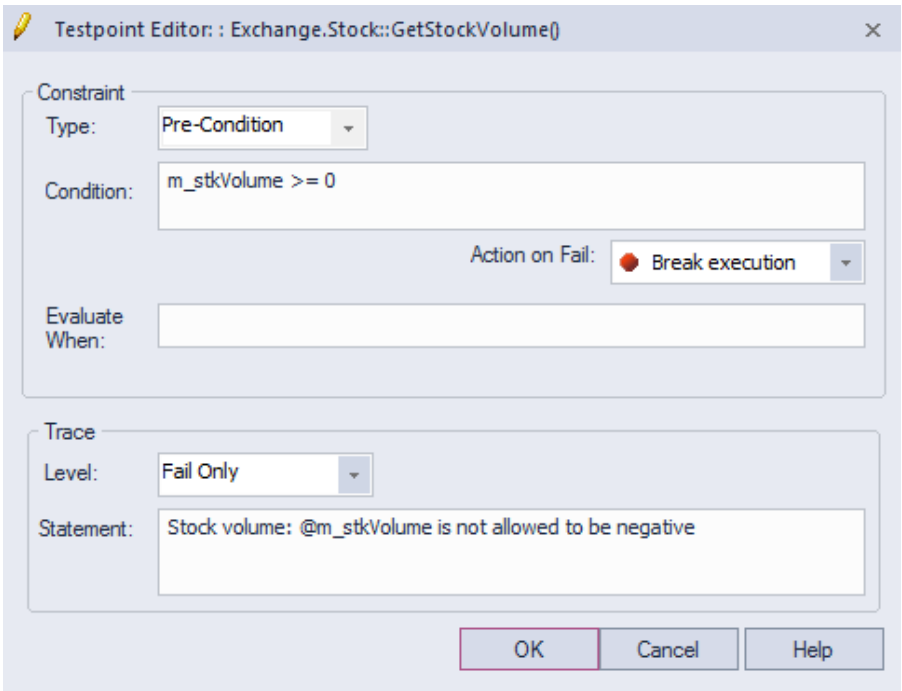
Vous pouvez ajouter ou modifier un invariant existant en double-cliquant sur la classe.

Vous pouvez ajouter ou modifier une pré-condition ou une post-condition existante de la même manière en double-cliquant sur la méthode.

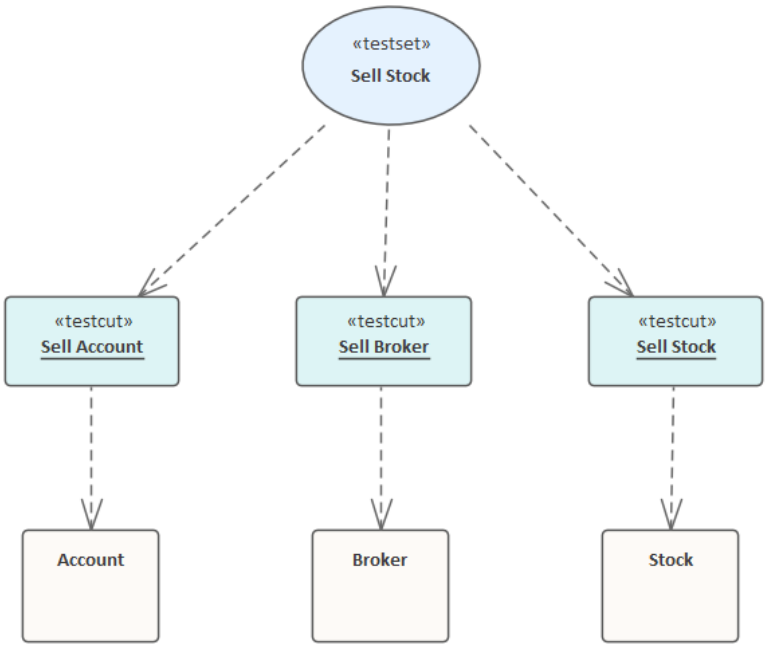
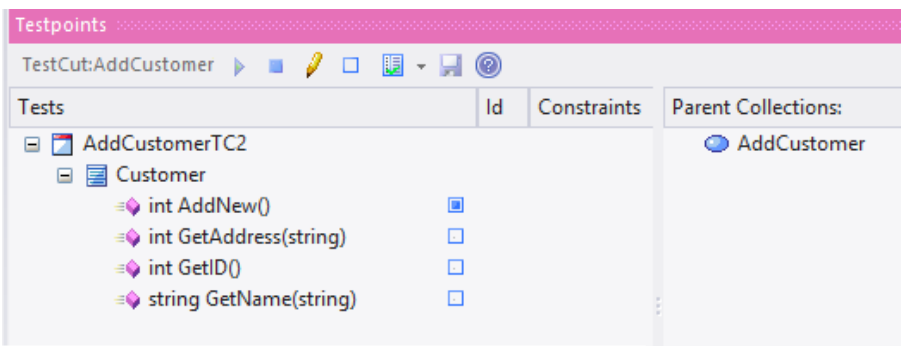
Un double-clic sur un Testpoint affichera automatiquement le code source s'il est disponible.

Il est préférable d'ajouter des conditions de ligne à partir de l'éditeur de code à l'aide de ses menus contextuels.

Cette image représente une condition préalable dans le domaine Test .



Déclarations de trace de Testpoint	Chaque Testpoint peut avoir sa propre instruction Trace. L'instruction Trace est un message dynamique qui peut référencer des variables dans son object ou sa portée locale. Elles sont générées lors de l'évaluation d'un test. Elles peuvent être configurées pour être générées à chaque fois qu'une contrainte est évaluée, ou plus généralement lorsqu'un test a échoué. Les instructions Trace peuvent être dirigées vers l'onglet « Testpoints » de la fenêtre Sortie système ou vers un fichier externe. Vous pouvez configurer cela dans n'importe quel script Analyzer.
Composition du domaine	Le diagramme de domaine Test est un support dynamique dans lequel Testpoints

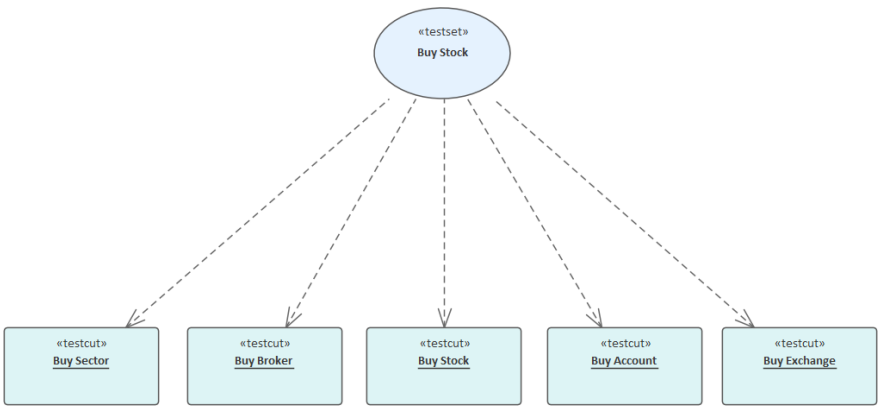
Test	sont assemblés pour tester les cas d'utilisation. Les cas d'utilisation dans un diagramme de domaine Test sont fournis dans trois stéréotypes différents : Test Cut, Ensemble de Test et Suite de Test . La gestion du domaine est aussi simple que modélisation sur n'importe quel diagramme . La boîte à outils et les menus contextuels donnent accès à tous les artefacts du domaine de Test . En bref, Testpoints de plusieurs classes sont agrégés dans Ensembles Test . Ensembles Test sont ensuite liés pour former des suites Test . Les coupes Test et Ensembles Test sont des ressources réutilisables. Lier le même Ensemble de Test à une ou plusieurs suites Test est une question de dessin de connecteurs. 
Domaine Test et Modèle de classe	Il est rare qu'un cas d'utilisation implique toutes les méthodes d'une seule classe. Il est très probablement réalisé à l'aide d'une variété de méthodes provenant de classes collaboratives. Nous appelons ce sous-ensemble de méthodes une « coupe », et l'artefact de coupe Test est l'outil que nous utilisons pour effectuer ces coupes. La fenêtre Testpoints s'adaptera en fonction du contexte, pour être celle d'un domaine Test ou d'un élément de classe. Cette image montre la fenêtre Testpoints lorsqu'une coupe Test a été sélectionnée. Note les cases à cocher, qui ne sont visibles que pour une coupe Test . Elles désignent les méthodes (coupures Test) qui contribuent à un Ensemble de Test . Dans cet exemple, le domaine Test a été généré par l' Analyseur d'Exécution , qui a effectué le travail d'identification des méthodes pour nous. 
Évaluation Testpoint	La fenêtre Testpoints est utilisée pour évaluer les domaines Test . La fenêtre dispose d'une barre d'outils pour démarrer ou se connecter à l'application cible. Le domaine à tester est toujours reflété par l'élément qui a un contexte, donc si vous sélectionnez une classe, la fenêtre affichera uniquement la structure de la classe et

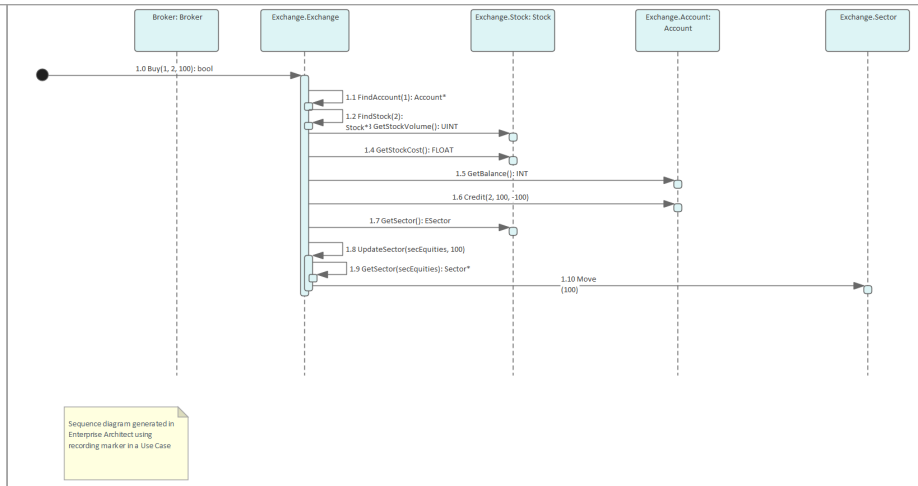
	Testpoints de cette classe. Si vous sélectionnez une Suite de Test , la fenêtre affichera toute la hiérarchie du domaine et tous les Testpoints qu'elle contient. Cliquer sur le bouton Exécuter chargera le domaine Testpoint dans l' Analyseur d'Exécution , qui évaluera, collectera et mettra à jour la fenêtre de rapport au fur et à mesure que les cas d'utilisation réussissent ou échouent à chaque test. Les détails exacts de chaque type de contrainte et le moment et la manière de la capture de cette contrainte sont les suivants : • Un invariant de classe est évalué par l'analyseur chaque fois qu'une méthode appelée sur un object de ce type de classe est terminée ; l'invariant sert à tester que l'état d'un object conforme est à la fois connu et autorisé • Les conditions préalables sont évaluées immédiatement avant l'appel d'une opération • Les post-conditions sont évaluées (en même temps qu'un invariant de classe) lorsque la méthode est terminée • Les conditions de ligne sont évaluées si et quand leur ligne de code spécifique entre dans le champ d'application pendant l'exécution du programme

Diagramme Domaine de Test

Le diagramme de domaine Test est le support sur lequel vous assemblez et regroupez les cas de test pour un domaine particulier. Un exemple de domaine Test peut être « Client ». L'étendue et la profondeur des domaines que vous assemblez dépendent de vous. Vous pouvez avoir des domaines distincts pour « Ajouter un client » et « Supprimer un client », en fonction de la manière dont vous considérez le mieux pour équilibrer la hiérarchie des domaines. La boîte à outils Diagramme et le menu contextuel fournissent un certain nombre d'artefacts pour vous aider à modéliser le domaine. Étant donné que le support est dynamique, ce qui vous permet de revisiter et de développer les relations entre les domaines Test , le système est un excellent modèle pour fournir des ressources réutilisables à une organisation qui ont une faible charge et s'intègrent à la fois à la vision UML du monde et aux rouages Ingénierie de Logiciel de la vie quotidienne.

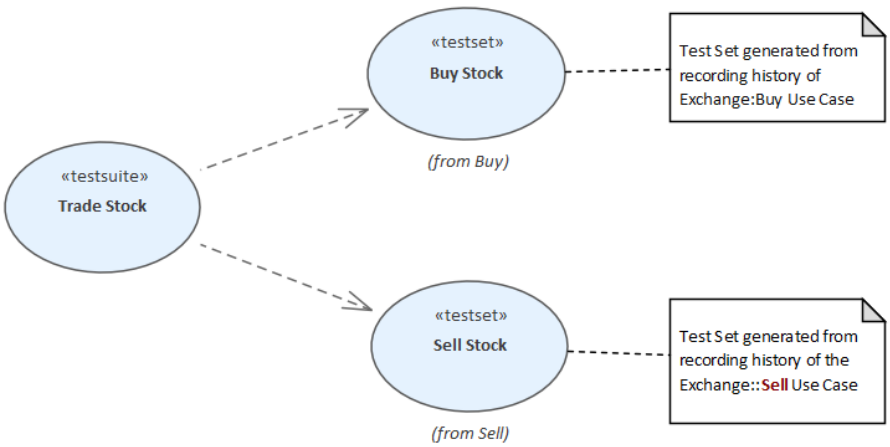
Facilités

Facilité	Détails
Génération de domaine Test	Si vous pensez que le processus de composition d'un domaine Test est complexe, c'est peut-être le cas, mais vous pouvez obtenir de l'aide ! L' Analyseur d'Exécution peut produire un diagramme de domaine Test pour vous. Il ne peut pas écrire les tests à votre place, mais il peut faire une partie du travail. Il peut identifier les classes et sélectionner uniquement les méthodes qui ont participé à un cas d'utilisation. Et ce n'est pas une conjecture. Le domaine Test de l'Analyseur est obtenu à partir d'un programme en cours d'exécution. Cette image montre le domaine de Test généré par l' Analyseur d'Exécution à partir de l'enregistrement d'un programme Modèle d'exemple.  Et c'est l'enregistrement lui-même (sous forme de diagramme Séquence) à partir duquel le domaine Test a été généré.

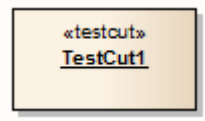


Composition du domaine Test

La première tâche sur un diagramme de domaine Test consiste à créer les cas d'utilisation (Ensembles Test). Ceux-ci définissent la responsabilité de ce domaine particulier. La boîte à outils Diagramme et le menu contextuel fournissent des artefacts pour vous aider à y parvenir. Le premier de ces éléments est le Test Cut, qui est utilisé dans l'étape suivante ; identifier les méthodes (à partir du modèle de classe) que vous considérez comme des participants au cas d'utilisation. L'artefact Test Cut est utile car il nous permet de partitionner une classe, en sélectionnant uniquement les méthodes pertinentes. Test Cut peuvent être exécuter individuellement ou liés à un ou plusieurs Test Ensembles . Test Ensembles peuvent à leur tour être liés à une ou plusieurs Test Suites. Dans tous les cas, tout élément de l'arborescence du domaine Test peut être exécuter individuellement ou dans son ensemble.



Coupe Test



Description

Un élément Test Cut est un élément Object stéréotypé, utilisé en interne dans Enterprise Architect pour définir Ensembles Test à l'aide des facilités de test de code Testpoint .

Une tâche, telle que « Imprimer », peut impliquer des opérations sur différentes classes. Pour créer un test « Imprimer », vous devez inclure uniquement les opérations « Imprimer » de ces classes et exclure toutes les autres opérations.

Un Test Cut vous permet de capturer uniquement les opérations qui représentent le comportement (dans ce cas, « Imprimer ») défini pour une seule classe. Vous pouvez ensuite placer le Test Cut de chacune des plusieurs classes dans une seule tâche en tant qu'Ensemble Ensemble de Test .

Lorsque vous faites glisser un élément Test Cut sur un diagramme Test Domain, vous créez une relation de dépendance avec l'élément Class requis. Par conséquent, lorsque vous sélectionnez l'élément Test Cut dans la fenêtre Testpoints , les opérations de la classe sont répertoriées dans la fenêtre, chacune avec une case à cocher. Vous sélectionnez ensuite la case à cocher en regard de chaque opération de classe à inclure dans le Test Cut.

Icône de la boîte à outils

Ensemble de Test

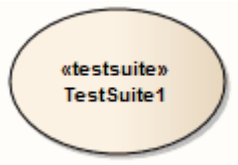
Description

Un élément Ensemble de Test est un élément de cas d'utilisation stéréotypé utilisé pour regrouper un ou plusieurs groupes de méthodes (coupures Test), qui peuvent s'étendre sur plusieurs classes, en une seule tâche. Ensembles Test peuvent également être regroupés en suites Test .

Vous liez les éléments Test Cut à l' Ensemble de Test à l'aide de connecteurs de dépendance.

Icône de la boîte à outils

Suite de Test



Description

Un élément Suite de Test est un élément Use Case stéréotypé, utilisé pour regrouper un ou plusieurs groupes de tâches (Ensembles Test).

Vous liez les éléments Ensemble de Test à la Suite de Test à l'aide de connecteurs de dépendances.

Icône de la boîte à outils

La Fenêtre Testpoints

La Fenêtre Testpoints est le hub où sont composées les contraintes du Domaine Test . C'est aussi le contrôle qui vous permet de vérifier un Domaine Test particulier sur un programme. Le programme peut être déjà en cours d'exécution ou il peut être lancé à l'aide de la barre d'outils du contrôle. Ici vous pourrez également voir les résultats de vos tests, au fur et à mesure qu'ils se produisent. Ce contrôle est contextuel, réagissant à la sélection d'éléments dans la fenêtre Navigateur ou sur un diagramme . Selon la sélection, les tests peuvent être effectués sur une seule Classe, un Cas d'Utilisation (Ensemble de Test) ou une collection de Cas d'Utilisation (Une Suite de Test).

Accéder

Ruban	Exécuter > Outils > Testeur > Afficher la fenêtre Testpoint

Colonnes de la Fenêtre Testpoints

Colonne	Usage
Tests	Affiche le nom de l' object Testpoint sélectionné et la hiérarchie des objets situés en dessous. L' object sélectionné peut être un : • Classe • Opération • Coupe Test • Ensemble de Test ou • Suite de Test
Identifiant	Pour une opération, cette colonne affiche une icône de marqueur Testpoint ( ) lorsque l'analyseur a correctement lié cette opération dans l'application cible. Si aucune icône n'apparaît dans cette colonne pendant un exécuter , cela indique que le modèle et la base de code ne sont peut-être pas synchronisés ; peut-être que la signature de la fonction a changé, ou que l'opération est une nouvelle méthode sur laquelle vous travaillez qui existe dans le code source mais pas encore dans votre modèle. Pour un Testpoint , cette colonne affiche un numéro d'identification généré. Ce numéro d'identification est utilisé dans la sortie de trace pour indiquer la contrainte référencée.
Contraintes	Une icône en forme de crayon ( ) dans cette colonne indique qu'une ou plusieurs contraintes sont définies pour cette classe ou opération.
Statut	Lors d'un test exécuter , indique ces statuts possibles : • ( ) Échec - La contrainte a été évaluée comme fausse une ou plusieurs fois. • ( ) Déclaration non valide - La contrainte n'a pas pu être analysée en raison d'une syntaxe non valide. • ( ) Variable non trouvée - Aucun nom de variable référencé n'a été trouvé à

	l'emplacement où la contrainte a été évaluée. Aucune icône n'est affichée si une contrainte est réussie.
Évaluations	Lors d'un test exécuter , indique le nombre de fois que l' Analyseur d'Exécution a évalué cette contrainte.
Passes	Lors d'un test exécuter , indique le nombre de fois que le test a réussi.
Échecs	Lors d'un test exécuter , indique le nombre de fois où le test a échoué.
Dernier Exécuter Par	Affiche le nom d'utilisateur de la dernière personne à exécuter ce test. (Les valeurs sont dérivées des définitions de l'auteur du projet dans la dialogue « Personnes » - « Paramètres > Données de référence > Types Modèle > Personnes > Auteurs du projet ».)
Date du dernier Exécuter	Affiche la date et l'heure de la dernière évaluation de ce test.
Résultat du dernier Exécuter	Affiche le résultat du dernier test exécuter .
Volet des collections parentes	Répertorie toutes les collections parentes qui incluent l' object sélectionné dans le cadre de leur conception. Double-cliquez sur cette collection pour en faire l’ object sélectionné dans le volet de gauche. Le volet Collections parentes peut être masqué en cliquant sur le bouton Afficher/Masquer le volet Collections parentes dans la barre d'outils de la fenêtre Testpoints .

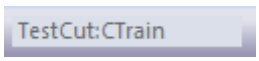
Barre d'Outils Testpoints

La Fenêtre Testpoints fournit des options permettant d'exécuter des tests configurés sur l' object Testpoint actuellement sélectionné, d' exécuter un test en cours, de filtrer les éléments affichés et d' exécuter les résultats d'un test terminé.

Accéder

Ruban	Exécuter > Outils > Testeur > Afficher la fenêtre Testpoint

Options de la barre d'outils Testpoints

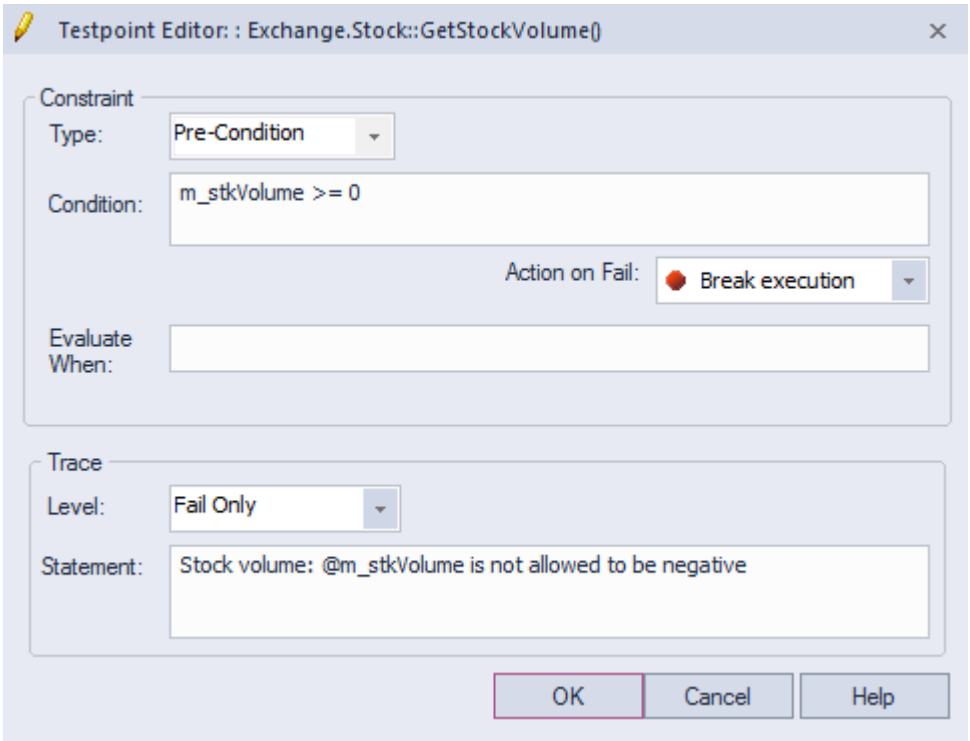
Bouton de la barre d'outils	Action
	Champ affichant le nom de l' object Testpoint actuellement sélectionné.
	Exécuter le test exécuter .
	Arrêter l' exécuter du test actuellement en cours.
	Basculez entre l'affichage de tous les éléments et l'affichage uniquement des éléments pour lesquels des contraintes sont définies.
	Basculez entre l'affichage de tous les éléments et l'affichage uniquement des opérations qui ont été marquées pour être incluses dans ce Test de coupe ; ce bouton n'est activé que lorsqu'un object Test Cut est sélectionné. Lorsqu'un Test coupe est sélectionné, chacune des opérations de sa classe associée est affichée avec une case à cocher ; vous utilisez cette case à cocher pour marquer les opérations qui s'appliquent à ce Test coupe.
	Cliquez sur la flèche déroulante à côté de cette icône pour afficher le menu « Options Exécuter Test », proposant ces options : • « Préfixer la sortie de trace avec un appel de fonction » : préfixer toutes les lignes de sortie de trace avec le nom de la fonction en cours d'exécution • « Activer Points d'Arrêt standard pendant Tester » : si cette option n'est pas cochée, l' exécuter de test ignore tous les points d'arrêt de l'ensemble de points d'arrêt actuel et toutes les tentatives de définition de points d'arrêt pendant l' exécuter sont ignorées. • « Sortie de la trace Vue » - Affiche l'onglet « Testpoints » de la fenêtre Sortie système
	Cliquez sur cette icône après avoir terminé un exécuter pour enregistrer les résultats dans l'élément Test sur l' object actuel. Les tests enregistrés peuvent être visualisés à l'aide de l'espace de travail Tester . Une prompt s'affiche pour sélectionner la classe Test : Unité, Intégration, Système, Inspection, Acceptation ou Scénario. Sélectionnez la classe Test appropriée et

	cliquez sur le bouton OK .
	Afficher la rubrique d’aide sur la gestion Testpoint .
	Afficher ou masquer le volet Collections parentes.

Éditeur Testpoints

L' Éditeur Testpoints permet de composer des contraintes pour les Classes et les Opérations. Les types de contraintes autorisés dépendent de l' object sélectionné. Pour les Classes, le type sera toujours Invariant. Pour les Opérations, le type peut être soit Pre-Condition, Post-Condition ou Line-Condition.

Les invariants sont évalués par l'analyseur lorsqu'une méthode appelée sur un object du type de classe sélectionné se termine. Les pré-conditions sont évaluées au début de chaque appel à l'opération spécifiée. Les post-conditions sont évaluées à la fin de chaque appel à l'opération spécifiée. Les conditions de ligne sont évaluées chaque fois que la ligne de code spécifiée est exécutée.



Accéder

Ruban	1. Exécuter > Outils > Testeur > Afficher la fenêtre Testpoint . 2. Dans la fenêtre Testpoints , double-cliquez sur une Classe ou une Opération pour afficher la dialogue « Éditeur Testpoints ».

Champs du groupe de contraintes

Champ	Usage
Type	Le type de contrainte pour la classe ou l'opération sélectionnée : • Invariant - Évalué après la fin de toute méthode appelée sur la classe spécifiée • Pré-condition - Évaluée au début de chaque appel à une opération spécifique • Post-Condition - Évalué après la fin de chaque appel à une opération spécifique

	• Condition de ligne - Évaluée lors de l'exécution d'une ligne de code spécifique dans une opération
Compenser	Pour les conditions de ligne uniquement, le numéro de ligne dans l'opération spécifiée sur laquelle évaluer la contrainte. Une valeur de décalage est automatiquement définie si le Testpoint a été créé à l'aide du menu contextuel de Éditeur de Code .
Condition	La contrainte à évaluer lorsque ce Testpoint est déclenché. Un statut de réussite ou d'échec sera enregistré selon que cette condition de contrainte est évaluée comme vraie ou fausse.
Action en cas d'échec	Cliquez sur la flèche déroulante et sélectionnez parmi les trois options : • « Continuer » : ignorer l'échec de cette contrainte et continuer l'exécution • « Interrompre l'exécution » - interrompre l'exécution et afficher la trace de la pile • « Désactiver en cas d'échec » : ne pas exécuter à nouveau la contrainte après un échec
Évaluer quand	(Facultatif) Une contrainte supplémentaire qui doit être respectée avant que la condition Testpoint principal ne soit évaluée, offrant un meilleur contrôle sur la couverture du test.

Champs du groupe de trace

Option	Action
Niveau	Spécifie quand l'instruction de trace (si elle est définie) sera générée. Les options disponibles sont : • « Échec uniquement » : instruction de trace de sortie uniquement lorsque cette condition Testpoint échoue • « Toujours » : Instruction de trace de sortie à chaque fois que ce Testpoint est évalué
Déclaration	(Facultatif) Un message à afficher lorsque ce Testpoint est évalué. Les variables actuellement dans la portée peuvent être incluses dans la sortie d'une instruction de trace en préfixant le nom de la variable avec un jeton $ pour les variables string , ou un jeton @ pour les types primitifs tels que ' int ' ou 'long '. La sortie d'une instruction Trace peut être dirigée soit vers l'onglet « Testpoints » de la fenêtre de sortie système, soit vers un fichier externe, tel que configuré par le script Analyzer pour le Paquetage parent.

Contraintes Testpoint

Une contrainte est généralement composée de variables locales et membres dans des expressions, séparées par des opérateurs pour définir un ou plusieurs critères spécifiques qui doivent être respectés. Une contrainte doit être évaluée comme vraie pour être considérée comme réussie. Si une contrainte est évaluée comme fausse, elle est considérée comme échouée.

Toutes les variables référencées dans la contrainte doivent être dans la portée à la position où le Testpoint ou Point d'Arrêt est évalué.

Opérateurs généraux/ Arithmétique

Opérateur	Description
+	Ajouter Exemple : a + b > 0
-	Soustraire Exemple : a - b > 0
/	Diviser Exemple : a / b == 2
*	Multiplier Exemple : a * b == c
%	Module Exemple : a % 2 == 1
()	Parenthèses - Utilisées pour définir la priorité dans les expressions complexes. Exemple : ((a / b) * c) <= 100
[ ]	Crochets - Utilisés pour accéder aux tableaux. Exemple : Names[0].Surname == « Smith »
.	Opérateur point - Utilisé pour accéder aux variables membres d'une classe. Exemple : Station.Name == « Flinders »
->	Notation alternative pour l'opérateur Point. Exemple : Station->Nom == « Flinders »

Opérateurs de comparaison

Opérateur	Description

=	Égal à Exemple : a = b
==	Égal à Exemple : a == b
!=	Pas égal à Exemple : a != b
<>	Pas égal à Exemple : a <> b
>	Plus grand que Exemple : a > b
>=	Supérieur ou égal à Exemple : a >= b
<	Moins que Exemple : a < b
<=	Inférieur ou égal à Exemple : a <= b

Opérateurs logiques

Opérateur	Description
AND	AND logique Exemple : (a >= 1) AND (a <= 10)
OR	OR logique Exemple : (a == 1) OR (b == 1)

Opérateurs au niveau du bit

Opérateur	Description
&	AND au niveau du bit Exemple : (1 & 1) = 1 (1 et 0) = 0
|	OR au niveau du bit

	Exemple : (1 | 1) = 1 (1 | 0) = 1
^	XOR au niveau du bit (OR exclusif) Exemple : (1 ^ 1) = 0 (1 ^ 0) = 1

Exemples supplémentaires

Exemple	Description
((m_nValeur & 0xFFFF0000) == 0)	Utilisez un opérateur AND au niveau du bit (&) avec une valeur hexadécimale comme opérande de droite pour tester qu'aucun bit n'est défini dans les octets d'ordre supérieur de la variable.
((m_nValue & 0x0000FFFF) == 0)	Utilisez un opérateur AND au niveau du bit (&) avec une valeur hexadécimale comme opérande de droite pour tester qu'aucun bit n'est défini dans les octets de poids faible de la variable.
m_value[0][1] = 2	Accéder à un tableau multidimensionnel
a AND (b OR c)	Combinaison des opérateurs AND et OR , en utilisant des parenthèses pour garantir la priorité. Dans cet exemple, la variable « a » doit être vraie et « b » ou « c » doit être vraie.

Notes

- Les comparaisons String sont sensibles à la casse