ENTERPRISE ARCHITECT

Série de Guides d'Utilisateur

Ingénierie de Logiciel

Author: Sparx Systems
Date: 23/11/2023
Version: 16.1

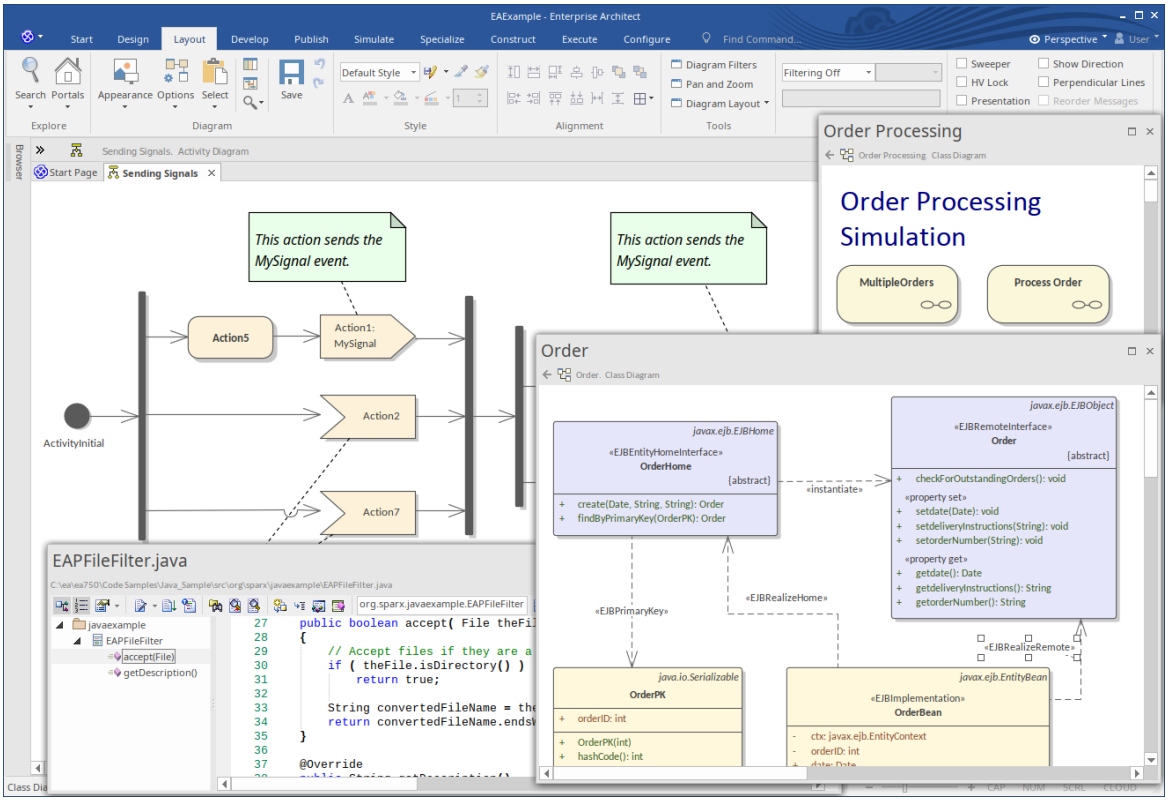
CRÉÉ AVEC ENTERPRISE ARCHITECT

Table des Matières

Ingénierie de Logiciel

Créer et gérer des modèles structurels et Comportementale efficaces et productifs de logiciels

Le génie logiciel est la discipline de conception, de mise en œuvre et de maintenance de logiciels. Le processus de génie logiciel commence par des exigences et des contraintes en tant qu'entrées, et aboutit à un code de programmation et à des schémas déployés sur diverses plates-formes, créant ainsi des systèmes en cours d'exécution.



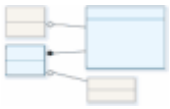
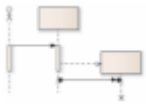
Enterprise Architect dispose d'un riche ensemble d'outils et fonctionnalités qui aident les ingénieurs logiciels à effectuer leur travail efficacement et à réduire le nombre d'erreurs dans les solutions mises en œuvre. Les fonctionnalités incluent des outils de conception pour créer des modèles de logiciels, la génération automatisée de code, l'ingénierie inverse du code source, des binaires et des schémas, ainsi que des outils pour synchroniser le code source avec les modèles de conception. Le code de programmation peut être visualisé et modifié directement dans les Éditeurs de Code intégrés dans Enterprise Architect , qui fournissent Intelli-sense et d'autres fonctionnalités pour faciliter le codage.

Un autre aspect intéressant de l'environnement est la capacité de retracer les cours de mise en œuvre jusqu'aux éléments de conception et à l'architecture, puis jusqu'aux exigences, contraintes et autres spécifications, et finalement jusqu'aux parties prenantes, à leurs objectifs et à leurs visions.

Enterprise Architect supporte un large éventail de langages et de plates-formes de programmation et offre une intégration légère et transparente avec les deux environnements de développement intégrés les plus répandus : Visual Studio et Eclipse. De plus, il existe un Analyseur d'Exécution complet qui permet à l'ingénieur logiciel de concevoir, de créer des modules logiciels de débogage et de test directement dans Enterprise Architect .

Facilités

Facilité	Description
Outils de développement	Découvrez l'environnement de développement étroitement intégré avec des outils et

	des fonctionnalités exceptionnels.
Coder, Build et Déboguer 	Modèle , développer, déboguer, profiler et gérer une application à partir de l'environnement modélisation .
Analyse visuelle de l'exécution du code 	Comprenez votre base de code en analysant visuellement le code en cours d'exécution. Utilisez les points Test , le profilage et la génération automatisée diagramme .
Générer Code Source 	Découvrez quelques-unes des façons de générer du code source pour une seule classe, une sélection de classes ou un Paquetage entier. Générer à partir de modèles structurels ou comportementaux.
Importation du code source 	Examinez les systèmes existants en important le code source dans Enterprise Architect . Vue et modifiez les définitions dialogue . Synchronisez le modèle avec les dernières mises à jour du code source.

Démarrage

Paramètres de configuration

Sélection de la perspective

Enterprise Architect divise les fonctionnalités étendues de l'outil en Perspectives , ce qui garantit que vous pouvez vous concentrer sur une tâche spécifique et travailler avec les outils dont vous avez besoin sans être distrait par d'autres fonctionnalités . Pour travailler avec fonctionnalités de Software Modèle , vous devez d'abord sélectionner l'une de ces Perspectives :

L'ensemble Ingénierie de Logiciel :

<nom de la perspective> > Ingénierie de Logiciel > Code Engineering

<nom de la perspective> > Ingénierie de Logiciel > GoF Motifs

<nom de la perspective> > Ingénierie de Logiciel > ICONIX

L'ensemble de conception UX :

<nom de la perspective> > Conception UX > Modèles UI Win 32

La définition de la perspective garantit que le Case Management Model and Notation diagrammes de notation, leurs boîtes à outils et autres fonctionnalités de la perspective seront disponibles par défaut.

Exemple Diagramme

Un exemple diagramme fournit une introduction visuelle au sujet et vous permet de voir certains des éléments et connecteurs importants que vous utilisez pour spécifier ou décrire des classes pour la visualisation de logiciels et l'ingénierie directe et inverse vers et depuis un large éventail de langages de programmation. .

Développement intégré

Dans cette rubrique, vous apprendrez à utiliser l'environnement de développement intégré complet. Vous apprendrez à créer des modèles structurels et comportementaux d'artefacts logiciels dans un éditeur de code riche, à générer et à effectuer de l'ingénierie inverse, à personnaliser la façon dont le code est généré, exécuter des scripts d'analyse pour optimiser le code, à utiliser le débogueur et à définir des tests unitaires et bien plus encore.

Modèles Comportementale

Modèles Comportementale

Dans cette rubrique, vous apprendrez à générer du code pour les langages de description de logiciels, de systèmes et de matériel directement à partir de diagrammes de comportement, notamment : Statemachine , Séquence et Activity Diagrammes . Cela ajoutera de nouvelles dimensions et précisions à la façon dont vous travaillez avec les logiciels et les systèmes d'ingénierie.

Motifs Gang des Quatre (GoF)

Ce sujet présente les vingt-trois motifs de conception renommés rassemblés sous le nom de motifs Gang of Four (GoF) qui font référence à leurs quatre auteurs. Vous aurez à portée de main les solutions aux problèmes courants auxquels sont confrontés les ingénieurs logiciels et serez en mesure d'injecter ces motifs dans vos propres modèles, ajoutant ainsi à la qualité et à la rigueur de vos systèmes logiciels.

Boîtes de dialogue Interface Utilisateur Win32

Dans cette rubrique, vous apprendrez à utiliser la fonctionnalité modélisation Interface Utilisateur d' Enterprise Architect qui vous permet de modéliser des écrans d'interface utilisateur à l'aide de contrôles Win32®. Les modèles peuvent faire l'objet d'une ingénierie directe ou inverse et peuvent également fournir une interface pour la simulation diagramme Statemachine et d'activité, leur permettant de recevoir et de traiter les entrées de l'utilisateur.

Cadre de code Gabarit

Dans cette rubrique, vous apprendrez à travailler avec le framework Code Gabarit qui régit la façon dont les modèles sont convertis en code. Il existe un ensemble standard de gabarits , mais vous pouvez les étendre pour créer vos propres gabarits et générer du code adapté à vos besoins. Il existe également gabarits qui contrôlent les transformations et la génération du langage de définition de base de données (DDL.

Cadre de grammaire

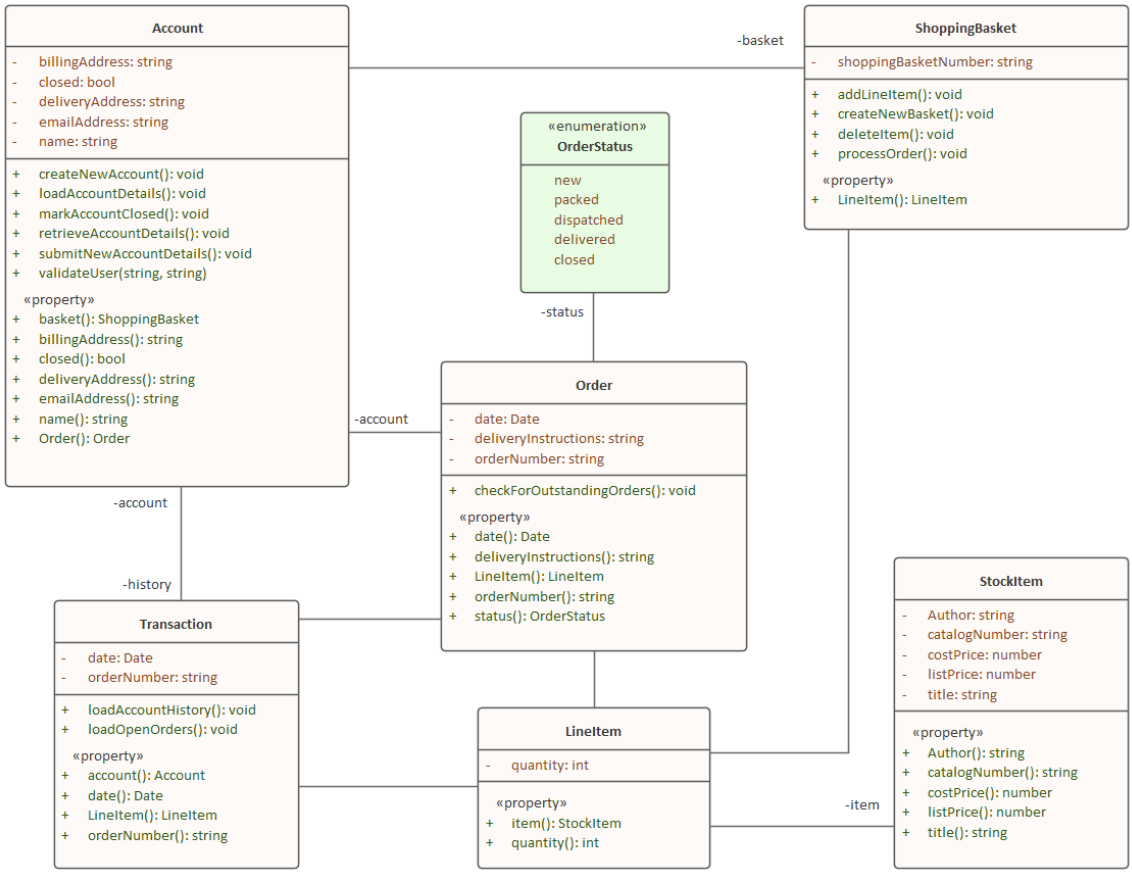
Dans cette rubrique, vous apprendrez comment créer une grammaire pour convertir un langage de programmation non pris en charge en modèle UML . Enterprise Architect prend en support un large éventail de langages de programmation, mais si vous devez travailler avec un langage non pris en charge, vous pouvez utiliser Grammar Framework pour écrire votre propre analyseur. La grammaire est utilisée pour procéder à l'ingénierie inverse du code de programmation sous forme de texte et est le complément direct du framework Code Gabarit qui vous permet de spécifier comment un modèle UML pour un langage non pris en charge est converti en code.

Cadre Code Miner

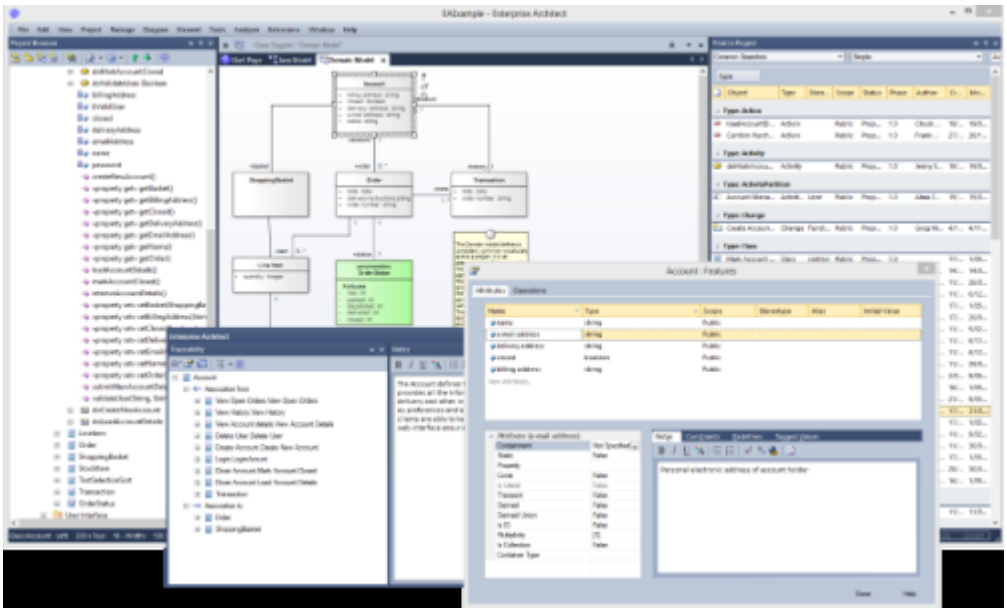
Dans cette rubrique, vous apprendrez à travailler avec une base de données de code source qui permet d'accéder aux données cachées dans le code source de manière rapide et efficace. Le code source est analysé pour créer une structure arborescente qui peut être utilisée pour analyser la structure du programme, calculer des métriques, tracer des relations et même effectuer une refactorisation.

Exemple Diagramme

diagrammes logiciels vous permettent de modéliser la structure et le comportement des logiciels, y compris les interfaces utilisateur. Enterprise Architect a pour base support fondamental pour les logiciels de modélisation et l'outil supporte un large éventail de langages et de paradigmes de programmation. Dans ce diagramme nous voyons les classes utilisées pour modéliser une boutique en ligne, y compris les classes qui contiennent des compartiments pour Attributes , les opérations et Propriétés . Une énumération a également été utilisée pour modéliser le statut de la commande.



Développement intégré



Enterprise Architect fournit un ensemble inégalé d'outils et fonctionnalités pour l'ingénieur logiciel, pour l'aider dans le processus de création de systèmes logiciels robustes et sans erreurs. L'ingénieur peut commencer par définir l'architecture et s'assurer qu'elle correspond aux exigences et aux spécifications. Les modèles technologiquement neutres peuvent être transformés pour cibler une gamme complète de langages de programmation. L'environnement de développement piloté Modèle convient à diverses technologies.

Fonctionnalités

Outils de développement

- Développement basé sur Modèle avec les meilleurs outils UML de leur catégorie
- Générer et rétro-ingénierie du code
- Personnalisez la génération de code avec gabarits
- Scripts d'Analyseur pour gérer vos applications
- Éditeurs de code pour créer la base de code
- Débogueurs pour étudier le comportement
- Profileurs pour visualiser le comportement
- Des analyseurs pour enregistrer le comportement
- Testpoints pour la validation des contrats de programmation
- Intégration avec jUnit et nUnit
- Intégration Eclipse ou Visual Studio si nécessaire

Traçabilité

En un coup d'œil, traçabilité des généralisations, réalisations, associations, dépendances et plus encore. Personnalisez les vues des relations. Parcourez facilement les éléments associés dans le modèle.

Usage

Parcourez rapidement l'utilisation des éléments dans tous diagrammes . Effectuez des recherches d'éléments efficaces à l'aide de requêtes sophistiquées.

Langues populaires

- C/C++

- Java
- Famille Microsoft .NET
- ADA
- Python
- Perl
- PHP

Boîtes à outils Des boîtes à outils sont fournies pour une vaste gamme de technologies modélisation et de langages de programmation.

Motifs d'application Enterprise Architect fournit des projets de démarrage complets, comprenant des informations sur le modèle, du code et des scripts de build, pour plusieurs types d'applications de base.

Aperçu Fonctionnalité

L'ingénierie de code avec Enterprise Architect englobe largement divers processus de conception, de génération et de transformation de code à partir de votre modèle UML .

Fonctionnalités

Ingénierie de code pilotée Modèle

- Génération de code source et ingénierie inverse pour de nombreux langages populaires, notamment C++, C# , Java, Delphi, VB.Net, Visual Basic, ActionScript, Python et PHP.
- Un éditeur de code source intégré de « coloration syntaxique »
- gabarits de génération de code, qui vous permettent de personnaliser le code source généré selon les spécifications de votre entreprise

Transformations pour un développement rapide

- Transformations avancées Model Driven Architecture (MDA) à l'aide gabarits de transformation
- Transformations intégrées pour DDL, C# , Java, EJB et XSD
- Un Modèle indépendant de la plate-forme peut être utilisé pour générer et synchroniser plusieurs modèles spécifiques à la plate-forme, offrant ainsi un gain de productivité significatif.
- diagramme de transformation XSL, boîte à outils, éditeur et débogueur.

Analyse d'Exécution Visuelle / Débogage, Vérification et Visualisation

- Exécuter des scripts de build, de test, de débogage, exécuter et de déploiement
- Intégrer le développement et modélisation UML au développement et à la compilation des sources
- Générer des classes de test NUnit et JUnit à partir des classes sources à l'aide des transformations MDA
- Intégrez le processus de test directement dans l'IDE Enterprise Architect
- Déboguer les applications .NET , Mono, Java et Microsoft Native (C, C++ et Visual Basic)
- Concevoir et exécuter des suites Test basées sur les principes de programmation par contrat
- Débogage de la feuille de style XSL

Modélisation de base de données

Enterprise Architect vous permet de :

- Ingénierie inverse à partir de nombreux SGBD populaires, notamment SQL Server, My SQL, Access, PostgreSQL et Oracle
- Modèle tableaux de base de données, colonnes, clés, foreign keys et relations complexes utilisant UML et un profil modélisation de données intégré
- Générer des scripts DDL pour créer des structures de base de données cibles

Ingénierie de la technologie XML

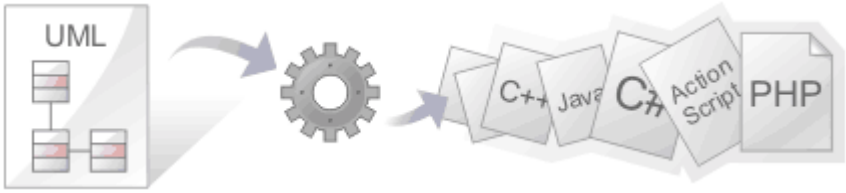
Enterprise Architect vous permet de modéliser, d'effectuer de l'ingénierie directe et de l'ingénierie inverse rapidement deux technologies XML clés du W3C :

- Schéma XML (XSD)
- Langage de définition de service Web (WSDL)

support de XSD et WSDL est essentielle au développement d'une architecture orientée services (SOA) complète, et le couplage d' UML 2.5 et de XML fournit le mécanisme naturel pour la mise en œuvre d'artefacts SOA basés sur XML au sein

d'une organisation.

Générer Code Source



La génération de code source est le processus de création de code de programmation à partir d'un modèle UML . Il y a de grands avantages à adopter cette approche car les Paquetages de code source, les classes et les interfaces sont automatiquement créés et élaborés avec des variables et des méthodes.

Enterprise Architect peut également générer du code à partir d'un certain nombre de modèles comportementaux, notamment diagrammes Statemachine , Séquence et Activity. Il existe un mécanisme gabarit très flexible qui permet à l'ingénieur de personnaliser complètement la façon dont le code source est généré, y compris les en-têtes de commentaires dans les méthodes et les classes de collection utilisées.

Du point de vue de l'ingénierie et de la qualité, l'avantage le plus convaincant de cette approche est que les modèles UML et donc l'architecture et la conception sont synchronisés avec le code de programmation. Un chemin traçable ininterrompu peut être créé depuis les objectifs, les moteurs commerciaux et les exigences des parties prenantes jusqu'aux méthodes du code de programmation.

Facilités

Facilité	Description
Langues	Enterprise Architect supporte la génération de code dans chacun de ces langages logiciels : • Script Action • Ada • ArcGIS • C • C# (pour .NET 1.1, .NET 2.0 et .NET 4.0) • C++ (standard, plus extensions C++ gérées .NET) • Delphes • Java (y compris Java 1.5, Aspects et Génériques) • JavaScript • mFQL • MySql • PHP • Python • TeradataSQL • Visual Basic • Visual Basic .NET • WorkFlowScript Vous pouvez également générer du code Hardware Definition Language dans ces langages : • VHDL • Verilog

	• SystèmeC
Éléments	Le code est généré à partir d'éléments de modèle de classe ou d'interface, vous devez donc créer les éléments de classe et d'interface requis à partir desquels générer. Tous les autres types d'éléments contribuant au code (tels que Statemachines ou les activités) doivent être des éléments enfants d'une classe. Ajoutez des attributs (qui deviennent des variables) et des opérations (qui deviennent des méthodes). Les contraintes et les réceptions sont également prises en charge dans le code.
Paramètres	Avant de générer du code, vous devez vous assurer que les paramètres par défaut pour la génération de code correspondent à vos besoins ; configurez les valeurs par défaut en fonction de la langue et des préférences requises. Les préférences que vous pouvez définir incluent les constructeurs et destructeurs par défaut, les méthodes pour les interfaces et les options Unicode pour les langages créés. Les langages tels que Java support les « espaces de noms » et peuvent être configurés pour spécifier une racine d'espace de noms. En plus des paramètres par défaut pour générer du code, Enterprise Architect facilite la définition d'options de génération spécifiques pour chacun des langages pris en charge.
Cadre de code Gabarit	Le Code Gabarit Framework (CTF) vous permet de personnaliser la manière dont Enterprise Architect génère le code source et permet également de générer des langages qui ne sont pas spécifiquement pris en charge par Enterprise Architect .
Chemins locaux	Les noms de chemin locaux vous permettent de remplacer les noms de répertoire par des balises.
Code Comportementale	Vous pouvez également générer du code logiciel à partir de trois paradigmes modélisation comportementale UML : • diagrammes d'interaction (Séquence) • diagrammes d'activité • diagrammes Statemachine (utilisant Legacy Statemachine Gabarits dans les opérations de génération de code sous « Tâches ») • diagrammes Statemachine (utilisant un artefact Statemachine Exécutable)
Génération de code en direct	Dans le menu déroulant « Développer > Code source > Options », vous avez la possibilité de mettre à jour votre code source instantanément lorsque vous apportez des modifications à votre modèle.
Tâches	Lorsque vous générez du code, vous effectuez une ou plusieurs de ces tâches : • Générer une seule classe • Générer un Groupe de Classes • Générer un Paquetage • Mettre à jour le contenu Paquetage

Notes

- La plupart des outils fournis par Enterprise Architect pour l'ingénierie de code et le débogage sont disponibles dans

les éditions Professional et supérieures d' Enterprise Architect ; La génération de code Comportementale est disponible dans les éditions Unified et Ultimate

- Lorsque la sécurité est activée, vous avez besoin des autorisations d'accès ' Générer Code Source et du DDL' et 'Reverse Engineer à partir du DDL et du code source'.

Générer une seule classe

Avant de générer du code pour une seule classe, vous :

- Compléter la conception de l'élément de modèle (Classe ou Interface)
- Créer des connecteurs d'héritage vers les parents et des associations avec d'autres classes utilisées
- Créez des connecteurs d'héritage aux interfaces que votre classe implémente ; le système fournit une option pour générer des stubs de fonction pour toutes les méthodes d'interface qu'une classe implémente

Générer du code pour une seule Classe

Étape	Action
1	Ouvrez le diagramme contenant la classe ou l'interface pour laquelle générer du code.
2	Cliquez sur la classe ou l'interface requise et sélectionnez l'option de ruban « Développer > Code source > Générer > Générer un seul élément », ou appuyez sur F11. La dialogue « Générer Code » s'affiche, à travers laquelle vous pouvez contrôler comment et où votre code source est généré.
3	Dans le champ « Chemin », cliquez sur le bouton ... et sélectionnez un nom de chemin pour lequel votre code source sera généré.
4	Dans le champ « Langue cible », cliquez sur la flèche déroulante et sélectionnez la langue à générer ; cela devient l'option permanente pour cette classe, alors modifiez-la si vous n'effectuez qu'un seul passage dans une autre langue.
5	Cliquez sur le bouton Avancé. La dialogue « Options Object » s'affiche, fournissant des sous-ensembles des pages « Ingénierie du code source » et des options de langage de code dans la dialogue « Préférences ».
6	Définissez les options personnalisées (pour cette classe uniquement), puis cliquez sur le bouton Fermer pour revenir à la dialogue ' Générer Code'.
7	Dans les champs « Import(s) / En-tête(s) », saisissez les instructions d'importation, #includes ou autres informations d'en-tête. Note que dans le cas de Visual Basic, ces informations sont ignorées ; dans le cas de Java, les deux zones de texte d'importation sont fusionnées ; et dans le cas du C++, la première zone de texte d'importation est placée dans le fichier d'en-tête et la seconde dans le fichier corps (.cpp).
8	Cliquez sur le bouton Générer pour créer le code source.
9	Une fois terminé, cliquez sur le bouton Vue pour voir ce qui a été généré. Note que vous devez d'abord configurer votre visualiseur/éditeur par défaut pour chaque type de langue ; vous pouvez également paramétrer l'éditeur par défaut sur la page ' Éditeurs de Code ' de la fenêtre Préférences (' Démarrer > Application > Préférences > Préférences > Ingénierie du Code Source > Éditeurs de Code ').

Générer un Groupe de Classes

En plus de pouvoir générer du code pour une classe individuelle, vous pouvez également sélectionner un groupe de classes pour la génération de code par lots. Lorsque vous effectuez cette opération, vous acceptez toutes les options de génération de code par défaut pour chaque classe de l'ensemble.

Générer un groupe de classe

Étape	Détail
1	Sélectionnez un groupe de classes et/ou d'interfaces dans un diagramme .
2	Cliquez sur un élément du groupe et sélectionnez l'option de ruban 'Développer > Code source > Générer > Générer les éléments sélectionnés' (ou appuyez sur Maj+F11). Si aucun code n'existe pour les éléments sélectionnés, la dialogue « Enregistrer sous » s'affiche dans laquelle vous spécifiez le chemin et le nom du fichier pour chaque fichier de code ; saisissez ces informations et cliquez sur le bouton Enregistrer.
3	La dialogue « Génération par lots » s'affiche, indiquant l'état du processus au fur et à mesure de son exécution (le processus est peut-être trop rapide pour voir cette dialogue). Si du code existe déjà pour les éléments de classe sélectionnés et que des modifications ont été apportées au nom ou à la structure de la classe, la dialogue « Synchroniser l'élément < nom paquetage >.<nom de l'élément> » peut également s'afficher ; ce dialogue permet de synchroniser le modèle et le code.

Notes

- Si l'un des éléments sélectionnés n'est pas des classes ou des interfaces, l'option de génération de code n'est pas disponible

Générer un Paquetage

En plus de générer du code source à partir de Classes uniques et de groupes de Classes, vous pouvez générer du code à partir d'un Paquetage . Cette fonctionnalité fournit des options pour générer de manière récursive du code à partir Paquetages enfants et générer automatiquement des structures de répertoires basées sur la hiérarchie Paquetage . Cela vous aide à générer du code pour une branche entière de votre modèle de projet en une seule étape.

Accéder

Ruban	Développer > Code Source > Générer > Générer Tout
Raccourcis Clavier	Ctrl+Alt+K

Générer du code à partir d'un Paquetage , sur la dialogue Générer le Code Source Paquetage

Étape	Action
1	Dans le champ « Synchroniser », cliquez sur la flèche déroulante et sélectionnez l'option de synchronisation appropriée : • « Synchroniser le modèle et le code » : le code des classes avec des fichiers existants est synchronisé avec ce fichier ; le code pour les classes sans fichier existant est généré dans le fichier cible affiché • « Code d'écrasement » : tous les fichiers cibles sélectionnés sont écrasés (générés en avant) • 'Ne pas générer' : Générer du code uniquement pour les Classes sélectionnées qui n'ont pas de fichier existant ; toutes les autres classes sont ignorées
2	Mettez en surbrillance les classes pour lesquelles générer du code ; laissez non sélectionné tout pour ne pas générer de code. Si vous souhaitez afficher davantage d'informations dans la disposition , vous pouvez redimensionner le dialogue et ses colonnes.
3	Pour Enterprise Architect génère automatiquement des répertoires et des noms de fichiers basés sur la hiérarchie Paquetage , cochez la case « Auto Générer des fichiers » ; cela active le champ « Répertoire racine », dans lequel vous sélectionnez un répertoire racine sous lequel les répertoires sources doivent être générés. Par défaut, la fonctionnalité « Générer automatiquement des fichiers » ignore tous les chemins de fichiers déjà associés à une classe ; vous pouvez modifier ce comportement en cochant également la case « Conserver les chemins de fichiers existants ».
4	Pour inclure le code de tous les sous-packages dans la sortie, cochez la case 'Inclure Pacquetages Enfants' .
5	Cliquez sur le bouton Générer pour commencer à générer du code. Au fur et à mesure de la génération du code, Enterprise Architect affiche des messages de progression. Si une classe nécessite un nom de fichier de sortie, le système vous prompts à en saisir un au moment approprié (en supposant que Générer automatiquement les fichiers n'est pas sélectionné). Par exemple, si

	les classes sélectionnées incluent des classes partielles, une prompt s'affiche pour saisir le nom de fichier dans lequel générer le code pour la deuxième classe partielle.

Plus d'informations sur le dialogue

Option	Action
Paquetage de racines	Vérifiez le nom du Paquetage pour lequel le code doit être généré.
Synchroniser	Sélectionnez les options qui spécifient la manière dont les fichiers existants doivent être régénérés.
Générer automatiquement des fichiers	Spécifiez si Enterprise Architect doit générer automatiquement des noms de fichiers et des répertoires, en fonction de la hiérarchie Paquetage .
Répertoire racine	Si Auto Générer des Fichiers est sélectionné, affichez le chemin sous lequel les structures de répertoires générées sont créées.
Conserver les chemins de fichiers existants	Si Générer automatiquement des fichiers est sélectionné, spécifiez s'il faut utiliser les chemins de fichiers existants associés aux classes. Si Générer automatiquement les fichiers n'est pas sélectionné, Enterprise Architect génère le code de classe vers des chemins déterminés automatiquement, que les fichiers source soient déjà associés ou non aux classes.
Inclure tous Paquetages enfants	Générez également du code pour toutes les classes de tous les sous-packages du Paquetage cible dans la liste. Cette option facilite la génération récursive de code pour un Paquetage donné et ses sous-Packages.
Sélectionner les objets à Générer	Répertoriez toutes les classes disponibles pour la génération de code sous les Paquetages cibles ; seul le code des classes sélectionnées (en surbrillance) est généré. Les classes sont répertoriées avec leur fichier source cible.
Tout sélectionner	Marquez toutes les classes de la liste comme sélectionnées.
Ne rien sélectionner	Marquez toutes les classes de la liste comme non sélectionnées.
Générer	Démarrer la génération de code pour toutes les Classes sélectionnées.
Annuler	Quittez la dialogue ' Générer Paquetage Source Code' ; aucun code de classe n'est généré.

Mettre à jour le contenu Paquetage

En plus de générer et d'importer du code, Enterprise Architect offre la possibilité de synchroniser le modèle et le code source, créant ainsi un modèle qui représente les dernières modifications apportées au code source et vice versa. Vous pouvez utiliser soit le modèle comme source, soit le code comme source.

Le comportement et les actions de synchronisation dépendent des paramètres que vous avez sélectionnés sur la page ' Attributes et opérations' de la dialogue 'Préférences'. En utilisant ces paramètres, vous pouvez protéger ou supprimer automatiquement les informations du modèle qui ne sont pas présentes dans le code, et prompt une décision sur fonctionnalités du code qui ne sont pas dans le modèle. Dans ces deux exemples, les cases appropriées ont été cochées pour une protection maximale des données :

- Vous avez généré du code source, mais avez apporté des modifications ultérieures au modèle ; lorsque vous générez à nouveau du code, Enterprise Architect ajoute de nouveaux attributs ou méthodes au code source existant, laissant intact ce qui existe déjà, ce qui signifie que les développeurs peuvent travailler sur le code source, puis générer des méthodes supplémentaires selon les besoins du modèle, sans avoir leur code. écrasé ou détruit
- Vous avez peut-être apporté des modifications à un fichier de code source, mais le modèle contient notes et des caractéristiques détaillées que vous ne souhaitez pas perdre ; en synchronisant le code source dans le modèle, vous importez des attributs et des méthodes supplémentaires mais ne modifiez pas les autres éléments du modèle

Grâce aux méthodes de synchronisation, il est simple de maintenir le code source et les éléments du modèle à jour et synchronisés.

Accéder

Ruban	Développer > Code source > Synchroniser > Synchroniser Paquetage

Synchronisez le contenu Paquetage avec le code source

Champ/Bouton	Action
Type de mise à jour	Sélectionnez le bouton radio pour effectuer l'ingénierie avancée ou l'ingénierie inverse des classes Paquetage , selon le cas.
Inclure paquetages enfants dans la génération	Cochez la case pour inclure Paquetages enfants dans la synchronisation.
OK	Cliquez sur le bouton pour démarrer la synchronisation. Enterprise Architect utilise les noms de répertoire spécifiés lors de la première importation/génération de la source du projet et met à jour le modèle ou le code source en fonction de l'option choisie. Si: • Effectuer une synchronisation directe AND • Il existe des différences entre le modèle et le code AND • La case « Lors de la synchronisation directe, prompt à supprimer fonctionnalités du code non présentes dans le modèle » est cochée dans la boîte de dialogue « Options - Attributes et opérations ». PUIS la dialogue « Synchroniser l'élément < nom paquetage >.<nom de l'élément> » s'affiche. Sinon, aucune autre action n’est requise.

Notes

- La synchronisation du code ne modifie pas les corps des méthodes ; le code comportemental ne peut pas être synchronisé et la génération de code ne fonctionne que lors de la génération de l'intégralité du fichier
- Dans les éditions Corporate , Unified et Ultimate d' Enterprise Architect , si la sécurité est activée, vous devez disposer de l'autorisation « Générer Code Source et DDL » pour synchroniser le code source avec les éléments du modèle.

Synchroniser Modèle et Code

Vous pourriez soit :

- Synchronisez le code d'un Paquetage de classes avec le modèle dans la fenêtre Navigateur , ou
- Régénérer le code à partir d'un lot de classes dans le modèle

Dans de tels processus, certains éléments du code peuvent ne pas être présents dans le modèle.

Si vous souhaitez intercepter ces éléments et les résoudre manuellement, cochez la case « Lors de la synchronisation avant, prompt à supprimer fonctionnalités de code non présentes dans le modèle » dans la boîte de dialogue « Options - Attributes et opérations », afin que « Synchroniser l'élément < nom paquetage > dialogue .<element name> s'affiche, offrant des options pour répondre à chaque élément.

Synchroniser Items

Bouton	Détail
Tout sélectionner	Mettez en surbrillance et sélectionnez tous les éléments de la colonne Fonctionnalité .
Effacer Tout	Désélectionnez et supprimez la surbrillance de tous les éléments de la colonne Fonctionnalité .
Supprimer	Marquez les fonctionnalités de code sélectionnées à supprimer du code (la valeur dans la colonne Action devient Supprimer).
Réaffecter	Marquez les fonctionnalités de code sélectionnées à réaffecter aux éléments du modèle. Cela n'est possible que lorsqu'un élément de modèle approprié est présent et n'est pas déjà défini dans le code. La boîte dialogue Sélectionner la Fonctionnalité de classe correspondante s'affiche, à partir de laquelle vous sélectionnez la classe à laquelle réaffecter la fonctionnalité . Cliquez sur le bouton OK pour marquer la fonctionnalité à réaffecter.
Ignorer	Marquez les éléments de code sélectionnés non présents dans le modèle à ignorer complètement (valeur par défaut ; la valeur dans la colonne Action reste la même ou devient <none>).
Réinitialiser aux valeurs par défaut	Réinitialisez les éléments sélectionnés sur Ignorer (la valeur dans la colonne Action devient <none>).
OK	Apportez les modifications assignées aux éléments et fermez le dialogue .

Namespaces

Les langages tels que Java support les structures Paquetage ou les espaces de noms. Dans Enterprise Architect vous pouvez spécifier un Paquetage comme racine d'espace de noms, qui indique l'endroit où commence la structure d'espace de noms de votre modèle de classe ; tous Paquetages subordonnés situés sous une racine d'espace de noms formeront la hiérarchie d'espace de noms pour les classes et les interfaces contenues.

Pour définir un Paquetage en tant que racine d'espace de noms, cliquez sur le Paquetage dans la fenêtre du Navigateur et sélectionnez l'option de ruban "Développer > Code source > Options > Définir comme racine d' Namespace ". L'icône Paquetage dans la fenêtre Navigateur change pour afficher un coin coloré indiquant que ce Paquetage est une racine d'espace de noms.

Le code source Java généré, par exemple, ajoutera automatiquement une déclaration Paquetage au début du fichier généré, indiquant l'emplacement de la classe dans la hiérarchie Paquetage sous la racine de l'espace de noms.

Pour effacer une racine d'espace de noms existante, cliquez sur le Paquetage racine de l'espace de noms dans la fenêtre Navigateur et désélectionnez l'option de ruban « Développer > Code source > Options > Définir comme racine d' Namespace ».

Pour afficher une liste d'espaces de noms, sélectionnez l'option de ruban « Paramètres > Données de référence > Paramètres > Racines Namespace » ; la dialogue « Namespaces » s'affiche. Si vous double-cliquez sur un espace de noms dans la liste, le Paquetage est mis en surbrillance dans la fenêtre Navigateur ; Alternativement, cliquez-droit sur l'espace de noms et sélectionnez l'option 'Localiser Paquetage dans Navigateur '.

Vous pouvez également effacer la racine de l'espace de noms sélectionnée en sélectionnant l'option « Attribut Namespace Effacer ».

Pour omettre un Paquetage subordonné d'une définition d'espace de noms, sélectionnez l'option de ruban « Développer > Code source > Options > Supprimer Namespace » ; pour inclure à nouveau le Paquetage dans l'espace de noms, désélectionnez l'option du ruban.

Notes

- Lors de la génération de code, tout nom Paquetage contenant des caractères d'espacement est automatiquement traité comme une racine d'espace de noms.

Importation du code source

La possibilité de visualiser simultanément le code de programmation et les modèles dont il est dérivé apporte de la clarté à la conception d'un système. L'une des fonctionnalités pratiques d'ingénierie de code Enterprise Architect est la possibilité de procéder à l'ingénierie inverse du code source dans un modèle UML . Un large éventail de langages de programmation est pris en charge et il existe des options qui régissent la manière dont les modèles sont générés. Une fois le code dans le modèle, il est possible de le maintenir synchronisé avec le modèle, que les modifications aient été apportées directement dans le code ou dans le modèle lui-même. Les structures de code sont mappées dans leurs représentations UML ; par exemple, une classe Java est mappée dans un élément Classe UML , les variables sont définies comme des attributs, les méthodes modélisées comme des opérations et les interactions entre les classes Java représentées par les connecteurs appropriés.

La représentation du code de programmation sous forme de constructions de modèle vous aide à mieux comprendre la structure du code et la manière dont il met en œuvre la conception, l'architecture et les exigences, et finalement comment il apporte la valeur métier.

Il est important de note que si un système n'est pas bien conçu, le simple fait d'importer le source dans Enterprise Architect ne le transforme pas en un modèle UML facilement compréhensible. Lorsque vous travaillez avec un système mal conçu, il est utile d'évaluer le code en unités gérables en examinant les Paquetages de modèles individuels ou les éléments générés à partir du code ; par exemple, en faisant glisser une classe d'intérêt spécifique sur un diagramme , puis en utilisant l'option « Insérer des éléments associés » à un niveau pour déterminer les relations immédiates entre cette classe et d'autres classes. À partir de ce point, il est possible de créer des cas d'utilisation qui identifient l'interaction entre les classes du code source, fournissant ainsi un aperçu du fonctionnement de l'application.

Plusieurs options guident la manière dont le code est procédé à l'ingénierie inverse, notamment si les commentaires sont importés dans notes et comment ils sont formatés, comment les méthodes de propriété sont reconnues et si des relations de dépendance sont créées pour le retour d'opération et les types de paramètres.

Propriété des droits d'auteur

Les situations qui se prêtent généralement à l'ingénierie inverse ont tendance à fonctionner sur un code source qui :

- Vous avez déjà développé
- Fait partie d'une bibliothèque tierce pour laquelle vous avez obtenu l'autorisation d'utiliser
- Fait partie d'un cadre que votre organisation utilise
- Est développé quotidiennement par vos développeurs

Si vous examinez un code que vous ou votre organisation ne possédez pas ou que vous n'avez pas l'autorisation spécifique de copier et de modifier, vous devez vous assurer que vous comprenez et respectez les restrictions de droits d'auteur sur ce code avant de commencer le processus d'ingénierie inverse.

Langues supportées pour Rétro-ingénierie

Langue
Script Action

Ada 2012 (éditions Unified et Ultimate)
C
C#
C++
CORBA IDL (MDG Technologie)
Delphes
Java
PHP
Python
SystemC (éditions Unified et Ultimate)
Verilog (éditions Unified et Ultimate)
VHDL (éditions Unified et Ultimate)
Visual Basic
Visual Basic .NET

Notes

- Rétro-ingénierie est supporté dans les éditions Professional , Corporate , Unified et Ultimate d' Enterprise Architect
- Si la sécurité est activée, vous devez disposer de l'autorisation « Ingénierie inverse à partir du DDL et du code source » pour procéder à l'ingénierie inverse du code source et synchroniser les éléments du modèle avec le code.
- À l'aide Enterprise Architect , vous pouvez également importer certains types de fichiers binaires, tels que les fichiers Java .jar et les fichiers .NET PE.
- Rétro-ingénierie dans d'autres langues est actuellement disponible grâce à l'utilisation de MDG Technologies répertoriées sur les pages MDG Technologie du site Sparx Systems

Importer des projets

Enterprise Architect prend en support l'importation de projets logiciels créés dans Visual Studio, Mono, Eclipse et NetBeans. Importer et travailler sur des projets dans Enterprise Architect présente de multiples avantages, notamment l'accès immédiat aux outils modélisation et fonctionnalités de gestion renommés d' Enterprise Architect , mais également l'accès aux outils de développement tels que la simulation, le débogage et le profilage.

Accéder

Ruban	Développer > Code source > Solutions > Importer un <type de projet>

Importer une solution Visual Studio

Cette option vous permet d'importer un ou plusieurs projets à partir d'un fichier de solution Visual Studio existant ou d'une instance en cours d'exécution de Visual Studio. L' assistant générera un modèle de classe pour chacun des projets et les Scripts d'Analyseur appropriés pour chaque configuration de Visual Studio.

Importer une solution mono

Cette option vous permet d'importer des projets Mono à partir d'un fichier de solution. La dialogue présentée est la même que la dialogue « Visual Studio Import », mais vous pouvez choisir de cibler Linux ou Windows . L' assistant générera un modèle de classe pour chacun des projets et les configurera pour le débogage. Les Scripts d'Analyseur générés référencent msbuild pour construire les projets.

Importer un projet Eclipse

L' Assistant Eclipse peut procéder à l'ingénierie inverse d'un projet Java décrit par son fichier Eclipse .project et sa version ANT. La fonctionnalité se traduira par un modèle de classe UML et Scripts d'Analyseur pour chacune des cibles ANT que vous sélectionnez. Le processus générera également un script pour chaque protocole de débogage que vous sélectionnez via l'« Assistant ». Vous aurez le choix entre JDWP (Java Déboguer Wire Protocol), idéal pour les serveurs, et JVMTI (Java Virtual Machine Tools Interface), adapté aux applications Java autonomes. Ces scripts doivent être utilisés pour déboguer le projet dans Enterprise Architect .

Importer un projet NetBeans

L' Assistant NetBeans peut procéder à l'ingénierie inverse d'un projet Java décrit par un fichier de projet XML NetBeans et une version ANT. L'« Assistant » créera un modèle de classe UML du projet et Scripts d'Analyseur pour chacune des cibles ANT que vous sélectionnez. Le processus générera également un script pour chaque protocole de débogage que vous sélectionnez via l'« Assistant ». Ces scripts doivent être utilisés pour déboguer le projet dans Enterprise Architect . Vous aurez le choix entre JDWP (Java Déboguer Wire Protocol), idéal pour les serveurs, et JVMTI (Java Virtual Machine Tools Interface), adapté aux applications Java autonomes.

Options d'importation

Lorsque vous choisissez d'importer une solution Visual Studio ou Mono, la dialogue « Importation de solution Visual Studio » s'affiche. Remplissez les champs comme indiqué dans ce tableau .

Lorsque vous choisissez d'importer une solution Eclipse ou Netbeans, l'écran de démarrage Assistant approprié s'affiche. Parcourez les écrans comme indiqué par les prompts sur chaque écran.

Option	Description
<liste des projets>	Après avoir sélectionné le fichier de solution, les projets de la solution sont répertoriés dans le panneau. Sélectionnez les projets à importer par l' Assistant . Vous pouvez utiliser le bouton Tous pour sélectionner tous les projets et le bouton Aucun pour effacer la sélection des projets.
Sélectionnez le fichier de solution	Recherchez et sélectionnez le fichier de solution à partir duquel importer. Les fichiers Mono Solution et Visual Studio Solution ont une extension de fichier .sln.
Effectuer un Dry Exécuter	Sélectionnez cette option pour effectuer l'importation en tant exécuter sèche, afin de vérifier toute erreur dans le processus ou la sortie avant de répéter l'importation pour modifier le contenu du modèle. Cliquez sur le bouton Vue Log pour vérifier le log de l'importation.
Créer Paquetage par fichier	Sélectionnez cette option pour effectuer l'importation avec une granularité plus fine, en créant un Paquetage distinct pour chaque fichier.
Importer	Cliquez sur ce bouton pour démarrer le processus d'importation.
Invite pour les définitions de macro manquantes	Non applicable aux importations de solutions Mono. Pour les projets C++ dans Visual Studio, l'analyseur peut rencontrer des macros non reconnues. Si vous sélectionnez cette option, vous serez averti lorsqu'un tel événement se produit et vous aurez la possibilité de définir la macro. Si vous ne sélectionnez pas cette option, le modèle de classe résultant peut manquer certains éléments.
Créer Diagramme pour chaque Paquetage	Une fois sélectionné, un diagramme de classes est créé décrivant le modèle de classe pour chaque Paquetage . Le résultat est un modèle plus grand mais plus coloré. La désélection de cette option entraînera l'omission de la création diagramme et l' exécuter plus rapide de l'importation.
Générer Scripts d'Analyseur	Pour les solutions Visual Studio, la sélection de cette option générera Scripts d'Analyseur pour chaque configuration de projet en plus des scripts pour chaque configuration de solution. Les scripts permettront de créer et de déboguer le(s) programme(s) décrit(s) par la solution immédiatement après la fin de l'importation. Cochez la case « Windows » ; si vous ne sélectionnez pas cette option, aucune fonctionnalités Analyseur d'Exécution ne sera configurée. Pour Mono Solutions, cette option vous permet de cibler soit Linux, soit Windows . Si vous sélectionnez Linux, on suppose que la machine sur laquelle Enterprise Architect s'exécute est Linux, que la plateforme (Java ou Mono) y est installée et que les programmes compilés exécuter sous Linux.
Projet de démarrage	Lorsque cette option est sélectionnée, le script de ce projet deviendra le modèle par défaut. Les outils de débogage, le ruban Exécuter et les boutons de la barre d'outils cibleront automatiquement ce programme.

Importer le code source

Vous pouvez importer du code source dans votre modèle Enterprise Architect pour procéder à la rétro-ingénierie d'un module. Au fur et à mesure de l'importation, Enterprise Architect fournit des informations sur la progression. Lorsque tous les fichiers sont importés, Enterprise Architect effectue une seconde passe pour résoudre les associations et les relations d'héritage entre les classes importées.

Procédure - Importer le code source

Étape	Action
1	Dans la fenêtre Navigateur , sélectionnez (ou ajoutez) un diagramme dans lequel importer les Classes.
2	Cliquez sur le fond diagramme et soit : • Sélectionnez l'option du ruban "Développer > Code source > Fichiers" et cliquez sur la langue appropriée, ou • Si la barre d'outils de génération de code s'affiche, cliquez sur la flèche déroulante « Importer » et sélectionnez la langue à importer. La liste des langues comprendra toutes les langues personnalisées pour lesquelles vous avez créé des structures de modèle.
3	Dans le navigateur de fichiers qui apparaît, recherchez et sélectionnez un ou plusieurs fichiers de code source à importer.
4	Cliquez sur le bouton Ouvrir pour démarrer le processus d'importation.

Notes sur l'importation du code source

Vous pouvez importer du code dans votre projet Enterprise Architect , dans une gamme de langages de programmation. Enterprise Architect supporte la plupart des constructions et mots-clés pour chaque langage de codage. Vous sélectionnez le type de fichier source approprié pour la langue, comme code source à importer.

S'il y a une fonctionnalité particulière pour laquelle vous avez besoin support et qui vous semble manquante, veuillez contacter Sparx Systems .

Notes

- Lors de l'ingénierie inverse d'attributs avec substitutions de paramètres (attributs modèles) :
  - Si une classe avec des définitions de paramètres gabarit appropriées est trouvée, un connecteur d'association est créé et ses substitutions de paramètres sont configurées
  - Un connecteur d'association est également créé si une entrée correspondante est définie comme classe de collection ou
  dans l'option 'Classes de collection supplémentaires' (pour C# , C++ et Java) ; pour un exemple, voir *Exemple Utilisation des classes de collection*

notes sur le langage de programmation

Langue	Notes
ActionScript	Type approprié de fichier source : fichier de code .as.
C	Type approprié de fichier source : fichiers d'en-tête .h et/ou fichiers .c. Lorsque vous sélectionnez un fichier d'en-tête, Enterprise Architect recherche automatiquement le fichier d'implémentation .c correspondant à importer, en fonction des options d'extension et de chemin de recherche spécifiées dans les options C. Enterprise Architect ne développe pas les macros qui ont été utilisées, celles-ci doivent être ajoutées à la liste interne des macros de langage.
C++	Type approprié de fichier source : fichier d'en-tête .h. Enterprise Architect recherche automatiquement le fichier d'implémentation .cpp en fonction de l'extension et du chemin de recherche définis dans les options C++ ; lorsqu'il trouve le fichier d'implémentation, il peut l'utiliser pour résoudre les noms de paramètres et notes de méthode si nécessaire. Lors de l'importation du code source C++, Enterprise Architect ignore les déclarations de pointeur de fonction. Pour les importer dans votre modèle, vous pouvez créer un typedef pour définir un type de pointeur de fonction, puis déclarer des pointeurs de fonction en utilisant ce type ; les pointeurs de fonction déclarés de cette manière sont importés en tant qu'attributs du type pointeur de fonction. Enterprise Architect ne développe pas les macros qui ont été utilisées ; celles-ci doivent être ajoutées dans la liste interne des macros de langue.
C#	Type approprié de fichier source : .cs.
Delphes	Type de fichier source approprié : .pas.

Java	Type approprié de fichier source : .java. Enterprise Architect supporte les extensions du langage AspectJ.  Les aspects sont modélisés à l'aide de Classes avec l'aspect stéréotype ; ces aspects peuvent alors contenir des attributs et des méthodes comme pour une Classe normale. Si un attribut ou une opération intertype est requis, vous pouvez ajouter une balise 'className' dont la valeur est le nom de la classe à laquelle elle appartient. Les pointcuts sont définis comme des opérations avec le stéréotype « pointcut » et peuvent se produire dans n'importe quelle classe, interface ou aspect Java ; les détails du pointcut sont inclus dans le champ « comportement » de la méthode. Le conseil est défini comme une opération portant le stéréotype « conseil » ; le point sur lequel opèrent ces conseils se situe dans le domaine du « comportement » et fait partie de la signature unique de la méthode. afterAdvice peut également faire revenir ou lancer l'une des Valeur Étiquetés .
PHP	Type approprié de fichier source : .php, .php4 ou .inc. Imbriqué si la syntaxe de condition est activée.
Python	Type approprié de fichier source : .py.
Visual Basic	Type approprié de fichier source : fichier de classe .cls.
Visual Basic .NET	Type approprié de fichier source : fichier de classe .vb.

Importer un script de ressource

Enterprise Architect supporte l'importation et l'exportation de Scripts de ressources Microsoft Windows (sous forme de fichiers .rc), qui contiennent les définitions dialogue Win32® (ceux avec le stéréotype «win32Dialog») pour l'interface utilisateur graphique d'une application. Les ressources Dialogue sont importées et exportées pour une langue spécifique, par défaut selon les paramètres régionaux du système informatique actuel.

Accéder

Ruban	Développer > Code source > Fichiers > Importer un script de ressource
Raccourcis Clavier	F7 (synchroniser l'élément avec le code)

Importer des ressources dialogue à partir d'un fichier .rc

Option	Action
Fichier de ressources	Cliquez sur le bouton ... et localisez le fichier .rc à partir duquel importer les éléments d'écran.
ID de ressource	Soit: • Laissez la valeur par défaut 'All' pour importer tous les éléments d'écran du fichier, ou • Cliquez sur la flèche déroulante et sélectionnez l' ID d'écran d'une dialogue spécifique à importer
Langue	Cliquez sur la flèche déroulante et sélectionnez la version linguistique (telle que Anglais - States -Unis) du ou le dialogue à importer.
Importer	Cliquez sur ce bouton pour importer les écrans du fichier ressource. La progression de l'importation est indiquée dans le champ situé sous le champ « Langue ».

Exporter une dialogue vers un fichier .rc

Option	Action
ID d'écran	Valeurs par défaut de l' ID Win32UI Valeur Étiquetée de l'élément Screen sélectionné. (Si le dialogue n'a pas cet ID , ouvrez la page 'Win32UI' du dialogue ' Propriétés ' de l'élément et fournissez une valeur pour la balise ID .)

Fichier de ressources	Cliquez sur le bouton ... et localisez le fichier .rc dans lequel exporter le(s) élément(s) d'écran. Si l'élément a été précédemment importé, ce champ correspond par défaut au fichier source.
Langue	Cliquez sur la flèche déroulante et sélectionnez la version linguistique (telle que Anglais - States -Unis) de la dialogue exportée.
Exporter	Cliquez sur ce bouton pour exporter les écrans du fichier ressource. La progression de l'export est signalée dans le champ situé sous le champ « Langue ».

Notes

- Les nouvelles boîtes de dialogue sont exportées vers un fichier .rc existant
- Lors d'une exportation vers un fichier .rc existant, aucune boîte de dialogue n'est jamais supprimée du fichier, même lorsqu'elle est supprimée du modèle
- Lors d'une importation, aucune boîte de dialogue n'est supprimée du modèle même lorsqu'elle est omise du fichier .rc d'origine

Importer une structure de répertoires

Vous pouvez importer à partir de tous les fichiers sources dans une structure de répertoires complète, ce qui vous permet d'importer ou de synchroniser plusieurs fichiers dans une arborescence de répertoires en un seul passage.

Enterprise Architect crée les Paquetages et diagrammes nécessaires pendant le processus d'importation.

Accéder

Ruban	Développer > Code source > Fichiers > Importer le répertoire source
Raccourcis Clavier	Ctrl+Maj+U

Importez une structure de répertoires à l'aide de la dialogue "Importer le répertoire source".

Champ	Action
Répertoire racine	Type ou recherchez le nom du répertoire à importer.
Type de Source	Type ou sélectionnez dans la liste déroulante la langue de codage des fichiers à importer dans le répertoire source.
Déposer	Type ou sélectionnez dans la liste déroulante les extensions de fichier à inclure dans l'importation. Utiliser un ';' pour séparer les valeurs.
Effectuer un Dry Exécuter	Si vous souhaitez effectuer l'import en exécuter sèche lorsque vous cliquez sur le bouton OK , cochez cette case. Une fois le traitement terminé, cliquez sur le bouton Vue Log pour vérifier le résultat prévu du processus.
Traiter les sous-répertoires de manière récursive	Si vous souhaitez inclure le contenu des sous-répertoires dans le processus d'importation, cochez cette case.
Importer des composants depuis	Si vous souhaitez importer des fichiers supplémentaires (comme décrit dans la dialogue « Importer des types de composants »), cochez cette case. Vous complétez ensuite l' prompt pour spécifier la provenance des composants.
Ne pas importer de membres privés	Si vous souhaitez exclure les membres privés du modèle lors de l'importation de bibliothèques, cochez cette case.
Invite pour les définitions de macro manquantes	Lors de l'importation, l'analyseur peut rencontrer des macros non reconnues. Si vous cochez cette case, vous serez averti lorsqu'un tel événement se produit et aurez la possibilité de définir la macro. Si vous ne sélectionnez pas cette option, la structure Paquetage résultante pourrait manquer certains éléments.
Structure Paquetage	Sélectionnez le bouton radio approprié pour créer un Paquetage pour chaque répertoire, chaque espace de noms ou chaque fichier ; cela peut être restreint en

	fonction du type de source sélectionné.
Créer Diagramme pour chaque Paquetage	Cochez cette case pour créer un diagramme dans chaque Paquetage créé lors de l'importation. Cliquez sur le bouton Options pour identifier les fonctionnalités des éléments à inclure dans les diagrammes .
Synchronisation	Sélectionnez le bouton radio approprié pour synchroniser les classes existantes ou écraser les classes existantes. Si une classe modèle correspondant à celle du code est trouvée : • « Synchroniser » met à jour la classe du modèle pour inclure les détails de celle du code, ce qui préserve les informations non représentées dans le code, telles que l'emplacement des classes dans diagrammes . • « Écraser » supprime la classe modèle et en génère une nouvelle à partir du code ; toute information supplémentaire n'est pas conservée. Si l'option « Utiliser les horodatages » est sélectionnée, alors la représentation avec le dernier horodatage (soit le modèle, soit le code) aura priorité.
Supprimer les classes introuvables dans le code	Sélectionnez le bouton radio approprié pour spécifier comment gérer les classes de modèles existantes qui ne sont pas présentes dans le code importé. • « Ne jamais supprimer » conserve toutes les classes existantes dans le modèle. • « Invite à l'action » vous permet de révision les cours individuellement • 'Toujours' supprimer 'supprime du modèle toute classe qui n'est pas présente dans le code importé.
OK	Cliquez sur ce bouton pour lancer l'importation.

Importer un module binaire

Enterprise Architect vous permet de procéder à la rétro-ingénierie de certains types de modules binaires.

Accéder

Ruban	Développer > Code source > Fichiers > Importer le module binaire

Utiliser

Actuellement, les types autorisés sont :

- Archives Java (.jar)
- Fichier .NET PE (.exe, .dll) - Les fichiers DLL et EXE Windows natifs ne sont pas pris en charge, seuls les fichiers PE contenant des données d'assemblage .NET
- Fichier de langue intermédiaire (.il)

Enterprise Architect crée les Paquetages et diagrammes nécessaires pendant le processus d'importation ; En cochant la case « Ne pas importer les membres privés », les membres privés des bibliothèques ne seront pas importés dans le modèle.

Lors de l'importation de fichiers .NET , vous pouvez importer par réflexion ou par désassemblage, ou laisser le système sélectionner la meilleure méthode - cela peut entraîner l'utilisation des deux types.

L'importateur basé sur la réflexion s'appuie sur un programme .NET et nécessite l'installation de l'environnement d'exécution .NET .

L'importateur basé sur le désassembleur s'appuie sur un programme Windows natif appelé Ildasm.exe, qui est un outil fourni avec le SDK MS .NET ; le SDK peut être téléchargé à partir du site Web de Microsoft.

Un choix de méthodes d'importation est disponible car certains fichiers ne sont pas compatibles avec la réflexion (comme mscorlib.dll) et ne peuvent être ouverts qu'à l'aide du désassembleur ; cependant, l'importateur basé sur la réflexion est généralement beaucoup plus rapide.

Vous pouvez également configurer :

- Qu'il s'agisse de synchroniser ou d'écraser les classes existantes lorsqu'elles sont trouvées ; si une classe modèle correspond à celle du fichier :
  - Synchroniser met à jour la classe de modèle pour inclure les détails de celle du fichier, ce qui préserve les informations non représentées dans le fichier, telles que l'emplacement des classes dans diagrammes
  - L'écrasement supprime la classe modèle et en génère une nouvelle à partir du fichier, qui supprime et ne remplace pas les informations complémentaires
- S'il faut créer un diagramme pour chaque Paquetage
- Ce qui est affiché sur diagrammes créés par l'importation

Classes introuvables lors de l'importation

Lors de l'ingénierie inverse à partir de votre code, il peut arriver que des classes soient délibérément supprimées de votre code source.

La fonctionnalité « Importer le répertoire source » assure le suivi des classes avec lesquelles elle s'attend à se synchroniser et, dans la dialogue « Importer la structure du répertoire », fournit des options sur la façon de gérer les classes qui n'ont pas été trouvées.

Vous pouvez sélectionner l'option appropriée pour Enterprise Architect , à la fin de l'importation, ignore les classes manquantes, les supprime automatiquement ou vous prompt à les gérer.

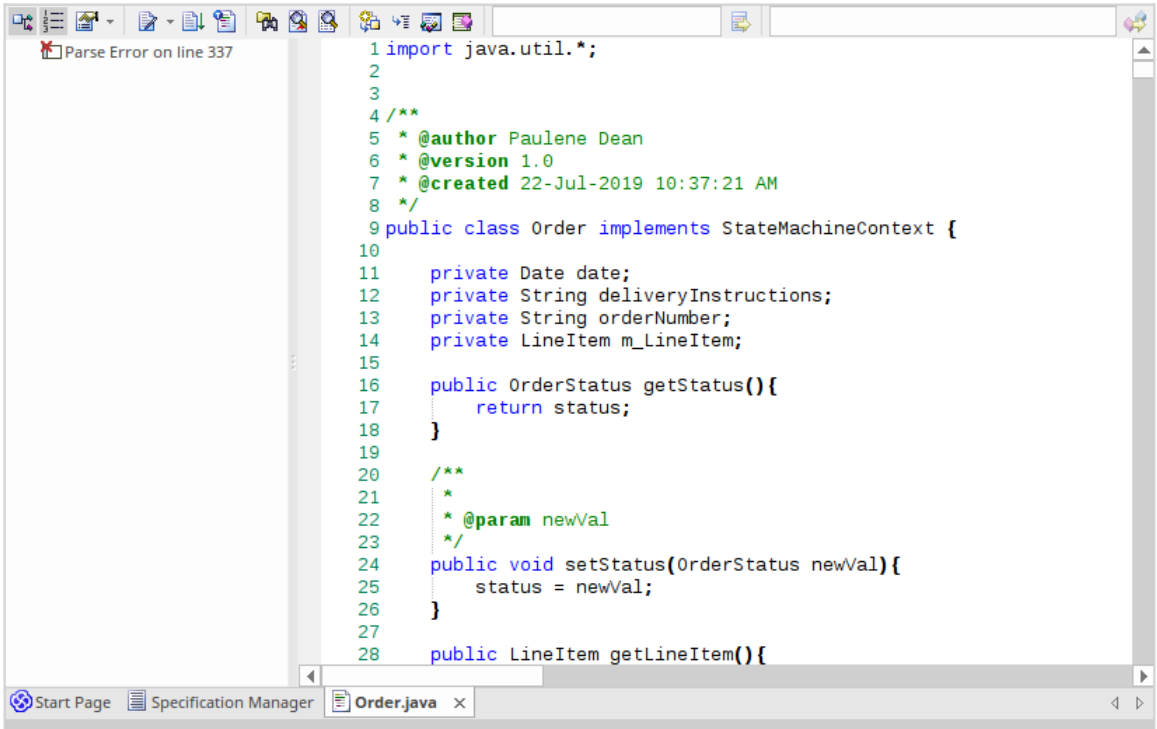
Dans la dialogue 'Importer la structure du répertoire', si vous sélectionnez le bouton radio 'Inviter à Action ' pour révision manuellement les classes manquantes, une dialogue s'affiche dans laquelle vous spécifiez la gestion de chaque classe manquante dans le code importé.

Par défaut, toutes les classes sont marquées pour suppression ; pour conserver une ou plusieurs Classes, sélectionnez-les et cliquez sur le bouton Ignorer.

Modification du code source

Enterprise Architect contient un éditeur de code source riche en fonctionnalités qui vous aide à afficher, modifier et maintenir votre code source directement dans l'outil. Une fois le code source généré pour une ou plusieurs classes, il peut être visualisé dans cet environnement d'édition flexible. Voir le code dans le contexte des modèles UML dont il est dérivé apporte de la clarté à la fois au code et aux modèles, et comble le fossé entre la conception et la mise en œuvre qui a historiquement introduit des erreurs dans les systèmes logiciels.

Le Source Éditeur de Code est complet, avec une arborescence de structure pour une navigation facile des attributs, des propriétés et des méthodes. Les numéros de ligne peuvent être affichés et les options de surbrillance de la syntaxe peuvent être configurées. De nombreuses fonctionnalités que les ingénieurs logiciels connaissent dans leur IDE préféré, telles qu'Intelli-sense et la complétion de code, sont incluses dans l'éditeur. Il existe de nombreuses fonctionnalités supplémentaires, telles que l'enregistrement de macros qui facilitent la gestion du code source dans Enterprise Architect . Il existe également de nombreuses options de gestion du code, disponibles via le menu contextuel de l'éditeur de code, la barre d'outils et les touches de fonction.



Pour la plupart des langages de programmation, un seul fichier est créé à partir d'une classe UML , mais dans le cas du C++, les classes d'en-tête et d'implémentation sont créées et l'éditeur de code source affiche ces fichiers dans des onglets séparés.

Un certain nombre d'options modifient le fonctionnement de l'éditeur de code source ; ils peuvent être modifiés à l'aide de la dialogue 'Préférences' disponible depuis le ruban Démarrer :

' Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Éditeurs de Code '

Il existe des variantes du Source Éditeur de Code , avec des modes d'accès différents. Les variantes sont abordées dans la rubrique *Comparer les éditeurs* .

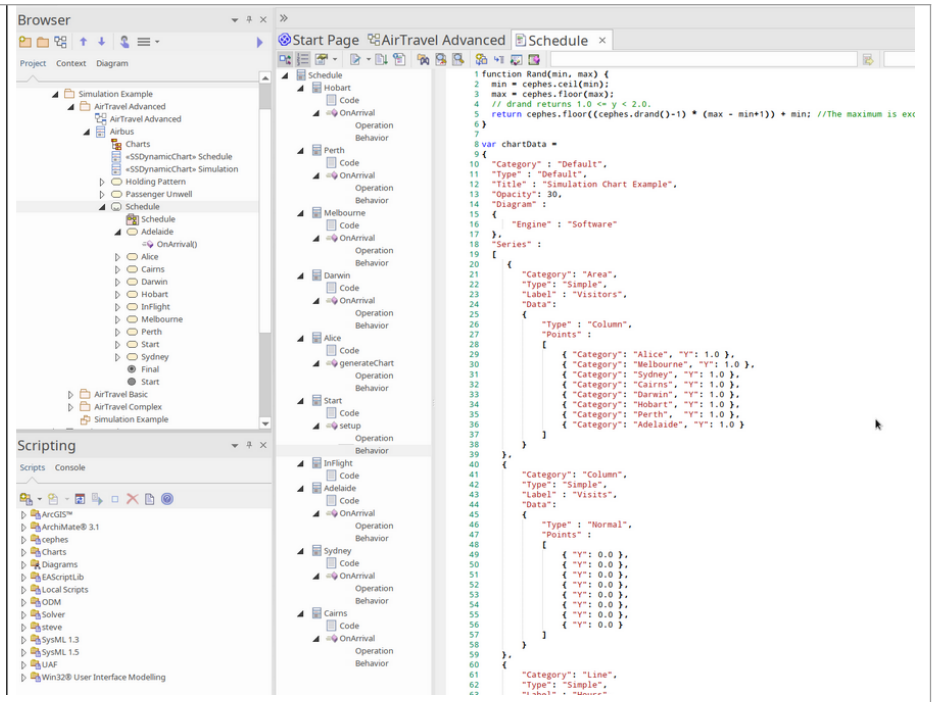
Accéder

Ruban	Exécuter > Source > Modifier > Fichier Open Source (fichier externe) ou Exécuter > Source > Modifier > Modifier la source de l'élément (pour un fichier source existant) ou

	Exécuter > Source > Modifier > Modifier le nouveau fichier source ou Conception > Élément > Comportement ou Développer > Code source > Comportement
Raccourcis Clavier	F12 ou Ctrl+E (pour le code existant pour les éléments du modèle) Ctrl+Alt+O (pour localiser les fichiers externes)

Facilités

Facilité	Description
Éditeur de code source	Par défaut, l'éditeur de code source est défini sur : • Analyser tous les fichiers ouverts et afficher une arborescence des résultats • Afficher les numéros de ligne CStation Location Name CStation(LPCTSTR, int) SetPosition(CPoint, size_t) ~CStation() 1 #pragma once 2 3 class CStation : public TObject 4 { 5 public: 6 CStation(LPCTSTR name,int ID); 7 ~CStation(void); Si vous modifiez un fichier XML, l'arborescence reflète l'ordre et la structure exacts du document. xs:schema xmlns:xs = http://www.w3.org/2001/XMLSchema xs:element "ContactInfo" type = ContactInfo xs:complexType "ContactInfo" xs:sequence xs:element "ContactInfo.homePhone" type = xs:string
Arborescence	L'arborescence de la structure des fichiers est disponible pour les fichiers de langage pris en charge, tels que C++, C# , Java et XML. L'arborescence peut être utile pour parcourir rapidement le contenu de la même manière qu'un tableau de contenu le ferait pour d'autres documents.
Comportements Simulation	Si vous modifiez les comportements des éléments dans un diagramme Statemachine ou Activity, l' Éditeur de Code vous permet de lister et de modifier ensemble les comportements de tous les éléments du diagramme , à l'aide d'une arborescence de structure.



Dans cette illustration, vous pouvez voir un certain nombre d' States au sein d'une Statemachine , dont chacun a des opérations et des comportements, et qui sont tous répertoriés ensemble et peuvent être sélectionnés sans quitter ni modifier la fenêtre de l'éditeur.

Notes

- Pour la plupart des éléments sélectionnés, vous pouvez utiliser les touches F12 ou Ctrl+E pour afficher le code source.
- Lorsque vous sélectionnez un élément pour afficher le code source, si l'élément n'a pas de fichier de génération (c'est-à-dire que le code n'a pas été ou ne peut pas être généré, comme pour un cas d'utilisation), Enterprise Architect vérifie si l'élément a un lien vers soit une opération, soit un attribut d'un autre élément - si un tel lien existe et que cet autre élément a un code source, le code de cet élément s'affiche
- Vous pouvez également localiser le répertoire contenant un fichier source qui a été créé ou importé dans Enterprise Architect et le modifier ou ses fichiers associés à l'aide d'un éditeur externe tel que le Bloc-notes ou Visual Studio ; cliquez sur l'élément dans la fenêtre Navigateur et appuyez sur Ctrl+Alt+Y

Langues prises en charge

Les Source Éditeurs de Code peuvent afficher le code dans un large éventail de langues, comme indiqué ici. Pour chaque langue, l'éditeur met en évidence - en texte coloré - la syntaxe du code standard.

- Ada (.ada, .ads, .adb)
- ActionScript (.as)
- Document BPEL (.bpel)
- C++ (.h, .hh, .hpp, .c, .cpp, .cxx)
- C# (.cs)
- Langage Query structuré DDL (.sql)
- Delphi/Pascal (.pas)
- Fichiers de différences/ Patch (.diff, .patch)
- Définition Type de document (.dtd)
- Fichiers batch DOS (.bat)
- Scripts de commande DOS (.cmd)
- HTML (.html)
- Langage de définition d'interface (.idl, .odl)
- Java (.java)
- JavaScript (.javascript)
- JScript (.js)
- Grammaire de la forme Backus-Naur modifiée (.mbnf)
- PHP (.php, .php4, .inc)
- Python (.py)
- Langage de balisage généralisé standard (.sgml)
- SystèmeC (.sc)
- Visual Basic 6 (.bas)
- VB.NET (.vb)
- VBScript (.vbs)
- Verilog (.v)
- Langage de description du matériel VHSIC (.vhdl)
- Configuration des ressources Visual Studio (.rc)
- XML (langage de balisage extensible) (.xml)
- XSD (définition de schéma XML)
- XSL (langage de feuille de style XML)

Configurer les associations de fichiers

Si vous êtes un utilisateur Windows®, vous pouvez configurer Enterprise Architect comme gestionnaire de documents par défaut pour vos fichiers source de langue.

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Éditeurs de Code : Configurer les associations de fichiers Enterprise Architect...

Actions

Pour chaque type de fichier que vous préférez ouvrir dans Enterprise Architect , cochez la case à gauche du nom du type de fichier. Après avoir sélectionné tous les types de documents dont vous avez besoin, cliquez sur le bouton Enregistrer.

Après cela, cliquer sur n'importe quel fichier correspondant dans l'Explorateur Windows® l'ouvrira dans Enterprise Architect .

Notes

- Vous pouvez modifier les programmes par défaut, ou les documents gérés par eux, directement via l'option « Programmes par défaut » du panneau de configuration Windows ®.

Comparer les éditeurs

Enterprise Architect propose quatre variantes principales d'éditeur de code, disponibles via un certain nombre de chemins d'accès. Les options d'accès les plus directes sont identifiées dans ces descriptions.

Les trois premières variantes d'éditeur de code répertoriées ont le même format d'affichage, la même barre d'outils d'options, les mêmes options de menu contextuel et les mêmes touches de fonction internes. Ils diffèrent par leur méthode d'accès et leur mécanisme d'affichage.

Variantes d'éditeur

Une variante	Détails
Vue du code source	F12 Ctrl+E Menu contextuel de la classe | 'Code source de Vue ' Description : Affiche le code sur un onglet du Diagramme Vue ; l'étiquette de l'onglet affiche le nom du fichier et son extension (telle que .java) ; encore une fois, pour C++, il existe deux onglets pour les fichiers d'en-tête et d'implémentation. Vous pouvez afficher le code source d'autres classes sur des onglets supplémentaires, en resélectionnant l'option/les touches de menu sur la classe suivante.
Fenêtre Code source (ancrable)	Alt+7 'Exécuter > Source > Modifier > Fichier Open Source' Description : Affiche le contenu du fichier source d'une Classe sélectionnée (sauf si le langage est C++, lorsque la fenêtre affiche un onglet pour le fichier d'en-tête et un onglet pour le fichier d'implémentation). Si vous sélectionnez une autre classe, la fenêtre change pour afficher le code de la nouvelle classe (sauf si la première classe appelle la seconde, auquel cas la fenêtre défile jusqu'au code de la deuxième classe).
Éditeur interne, code source externe	Ctrl+Alt+O Option du ruban « Exécuter > Source > Modifier > Fichier Open Source » Description : utilisez cette option si vous avez l'intention de modifier du code externe, des fichiers XML ou DDL (c'est-à-dire du code non importé ou généré dans Enterprise Architect). Affiche un navigateur externe, puis ouvre le fichier de code spécifique sélectionné sous forme d'onglet du Diagramme Vue (pour C++, pas deux fichiers de code) ; sinon, c'est identique à l'option F12.
Editeur Externe, Code Source Interne ou Externe	Ctrl+Alt+Y Menu contextuel de la classe | Annuaire Open Source Description : Affiche un navigateur de fichiers externe, ouvert sur le répertoire contenant les fichiers sources de la classe sélectionnée ; vous pouvez ouvrir les fichiers dans le Bloc-notes, Visual Studio ou d'autres outils que vous pourriez avoir sur votre système.

Barre d'outils Éditeur de Code

Lorsque vous réviser le code d'une partie de votre modèle dans l'éditeur de code source, vous pouvez accéder à un large éventail de fonctions d'affichage et d'édition depuis la barre d'outils de l'éditeur.

Barre d'outils Éditeur de Code

Options barre d'outils

Arborescence

Cliquez sur cette icône pour afficher ou masquer le panneau de hiérarchie des éléments (le panneau de gauche de l'éditeur de code source).

Numéros de ligne

Cliquez sur cette icône pour afficher ou masquer les numéros de ligne par rapport aux lignes de code.

Propriétés d'ingénierie du code source

Cliquez sur la flèche déroulante pour afficher un menu d'options permettant de sélectionner des pages individuelles « Ingénierie du code source » de la dialogue « Préférences », à partir desquelles vous pouvez configurer les options d'affichage et de comportement pour l'ingénierie du code source :

- Langue
- Options de coloration syntaxique
- Éditeur de Code Options
- Options d'ingénierie de code
- Éditeur de Code Key Bindings

Fonctions de l'éditeur

Cliquez sur la flèche déroulante pour afficher un menu donnant accès à de nombreuses fonctions d'édition de code :

- Ouvrir le fichier correspondant (Ctrl+Shift+O) - ouvre l'en-tête ou le fichier d'implémentation associé au fichier actuellement ouvert
- Accédez à l'accolade correspondante (Ctrl+E) - pour une accolade d'ouverture ou de fermeture sélectionnée, met en surbrillance l'accolade de fermeture ou d'ouverture correspondante dans la paire.
- Aller à la ligne (Ctrl+G) - affiche une dialogue dans laquelle vous sélectionnez le numéro de la ligne à mettre en surbrillance ; cliquez sur le bouton OK pour déplacer le curseur sur cette ligne
- Historique du curseur précédent (Ctrl+-) - le visualiseur de code source conserve un historique des 50 positions précédentes du curseur, créant un enregistrement lorsque le curseur est déplacé de plus de 10 lignes par rapport à sa position précédente ou lors d'une opération de recherche et de remplacement. ; l'option de menu déplace le curseur vers la position dans l'enregistrement historique du curseur immédiatement précédent
- Historique du curseur suivant (Ctrl+Maj+-) : si vous êtes passé à une position antérieure du curseur, cette option déplace le curseur vers la position dans l'enregistrement de l'historique du curseur immédiatement suivant.
- Rechercher (Ctrl+F) : affiche une dialogue dans laquelle vous définissez une string de texte et des options de recherche pour localiser cette string de texte

dans le code.

- Remplacer (Ctrl+R) - affiche une dialogue dans laquelle vous définissez une string de texte et des options de recherche pour localiser cette string texte dans le code et la remplacer par une autre string de texte ; le dialogue a des options pour localiser et remplacer chaque occurrence comme vous le décidez, ou pour remplacer toutes les occurrences immédiatement
- Mettre en surbrillance les mots correspondants - (Ctrl+3) Active ou désactive la mise en surbrillance des mots correspondants lors d'une opération de recherche ; par défaut cette option est activée
- Enregistrer une macro : enregistre vos prochaines frappes au clavier pour les enregistrer sous forme de macro.
- Arrête d'Enregistrer et Enregistrer la macro - arrête l'enregistrement des frappes et affiche la dialogue 'Enregistrer la macro' dans laquelle vous spécifiez un nom pour la macro
- Play Macro - affiche la dialogue « Ouvrir la macro » à partir de laquelle vous sélectionnez et exécutez une macro enregistrée, pour répéter les frappes enregistrées.
- Basculer le commentaire de ligne (Ctrl+Shift+C) - commente (//) ou rétablit le code pour chaque ligne complète dans laquelle le texte est mis en surbrillance
- Basculer le commentaire de flux (Ctrl+Shift+X) - insère un commentaire de flux (/* */) à la position du curseur (commente uniquement les caractères et les lignes en surbrillance) ou rétablit le texte commenté sous forme de code
- Basculer les caractères d'espacement (Ctrl+Shift+W) - affiche ou masque les caractères d'espacement : --> (espace de tabulation) et . (espace de caractères)
- Basculer les caractères EOL (Ctrl+Shift+L) - affiche ou masque les caractères de fin de ligne : CR (retour chariot) et LF (saut de ligne)
- Toggle Tree Synchronization - sélectionne automatiquement l'élément de l'arborescence lorsque le contexte change dans l'éditeur de code
- Ouvrir le dossier contenant - ouvre le navigateur de fichiers dans le dossier contenant le fichier de code ; vous pouvez ouvrir d'autres fichiers dans votre éditeur externe par défaut à des fins de comparaison et de travail parallèle

Enregistrer la source et resynchroniser la classe

Cliquez sur cette icône pour enregistrer le code source et resynchroniser le code et la Classe dans le modèle.

Code Gabarits

Cliquez sur cette icône pour accéder à l'éditeur Code Gabarits , pour éditer ou créer gabarits de code pour la génération de code.

Rechercher dans Projet Navigateur

Pour une ligne de code sélectionnée, cliquez sur cette icône pour mettre en surbrillance la structure correspondante dans la fenêtre Navigateur . S'il existe plusieurs possibilités, la dialogue « Correspondances possibles » s'affiche, répertoriant les occurrences de la structure à partir desquelles vous pouvez sélectionner celle requise.

Rechercher dans les fichiers

Cliquez sur cette icône pour rechercher le nom object sélectionné dans les fichiers associés et afficher les résultats de la recherche dans la fenêtre Recherche de fichiers. Vous pouvez affiner et actualiser la recherche en spécifiant des critères dans la barre d'outils de la fenêtre Rechercher dans les fichiers.

Rechercher dans Modèle

Cliquez sur cette icône pour rechercher le texte sélectionné dans tout le modèle, et afficher les résultats de la recherche dans la vue Rechercher dans Projet .

Aller à la déclaration

Cliquez sur cette icône pour localiser la déclaration d'un symbole dans le code source.

Aller à la définition	Cliquez sur cette icône pour localiser la définition d'un symbole dans le code source (applicable aux langages tels que C++ et Delphi, où les symboles sont déclarés et définis dans des fichiers séparés).
Liste de saisie semi-automatique	Cliquez sur cette icône pour afficher la liste d'autocomplétion des valeurs possibles ; double-cliquez sur une valeur pour la sélectionner.
Informations sur les paramètres	Lorsque le curseur se trouve entre les parenthèses de la liste des paramètres d'une opération, cliquez sur cette icône pour afficher la signature de l'opération en mettant en évidence le paramètre en cours.
Rechercher la classe actuelle dans la fenêtre Navigateur	Cliquez sur cette icône pour afficher le nom de la Classe actuellement sélectionnée dans le code, et mettez ce nom en surbrillance dans la fenêtre Navigateur ; s'il y a plus d'une possibilité, la dialogue « Correspondances possibles » s'affiche, répertoriant les occurrences de la classe parmi lesquelles vous pouvez sélectionner celle requise.
Trouver un membre	Cliquez sur cette icône pour afficher le nom de l'attribut ou de la méthode actuellement sélectionné dans le code et mettez ce nom en surbrillance dans la fenêtre Navigateur ; s'il y a plus d'une possibilité, la dialogue « Correspondances possibles » s'affiche, répertoriant les occurrences de la fonctionnalité parmi lesquelles vous pouvez sélectionner celle requise.

Notes

- L'option « Enregistrer la macro » désactive Intelli-sense pendant l'enregistrement de la macro
- Vous pouvez attribuer des touches pour exécuter la macro, au lieu d'utiliser la liste déroulante de la barre d'outils et dialogue « Ouvrir la macro ».

Éditeur de Code Menu Contexte

Lorsque vous travaillez sur un fichier avec un éditeur de code, vous pouvez effectuer un certain nombre d'opérations de recherche et d'édition de code pour révision le contenu du fichier. Ces options sont disponibles via le menu contextuel de l'éditeur et peuvent varier en fonction de l'éditeur de code que vous utilisez.

Accéder

Menu Contexte	Cliquez-droit sur la string de texte du code sur laquelle vous travaillez

Possibilités

Aller à la déclaration

Recherchez et mettez en surbrillance la déclaration d'un symbole dans le code source.

Aller à la définition

Recherchez et mettez en surbrillance la définition d'un symbole dans le code source (applicable aux langages tels que C++ et Delphi, où les symboles sont déclarés et définis à des endroits séparés).

Ouvrir dans l'éditeur de grammaire

Ouvre une vue qui vous permet d'examiner ou de valider le code à l'aide de la grammaire appropriée.

Synchroniser l'arborescence avec l'éditeur

Recherche et affiche l'élément courant (méthode par exemple) dans l'arborescence.

Synchronisation automatique de l'arborescence et de l'éditeur

Lorsqu'elle est sélectionnée, l'arborescence de la structure affichera automatiquement l'élément sur lequel on travaille dans l'éditeur.

Validation du schéma XML

Permet de valider un schéma XML.

Recherchez '< string >'

Affichez un sous-menu proposant des options pour localiser la string de texte sélectionnée dans une plage d'emplacements.

- ' Rechercher dans Projet Navigateur ' - Mettre en surbrillance l' object contenant le texte sélectionné dans la fenêtre du Navigateur
- « Rechercher dans les fichiers ouverts » - Recherchez la string de texte sélectionnée dans les fichiers ouverts associés et affichez les résultats de la recherche dans la fenêtre Rechercher dans les fichiers ; vous pouvez affiner et actualiser la recherche en spécifiant des critères dans la barre d'outils de la fenêtre Rechercher dans les fichiers
- « Rechercher dans les fichiers » - Recherchez la string de texte sélectionnée dans tous les fichiers associés (fermés ou ouverts) et affichez les résultats de la recherche dans la fenêtre Rechercher dans les fichiers ; vous pouvez affiner et actualiser la recherche en spécifiant des critères dans la barre d'outils de la fenêtre Rechercher dans les fichiers (touche de raccourci : F12)

- 'Rechercher dans Modèle ' - Effectuer une recherche 'Nom d'élément' dans la Recherche Modèle facilité , et afficher les résultats dans l'onglet Recherche Modèle
- 'Rechercher dans Scripts ' - (Disponible lorsque vous travaillez dans l' Éditeur de Script) Ouvrez la fenêtre Rechercher dans les fichiers, définissez le champ 'Chemin de recherche' sur 'Rechercher dans Scripts ' et le champ 'Rechercher le texte' sur le texte sélectionné, puis recherchez tous les scripts pour la string de texte et afficher les résultats de la recherche ; vous pouvez affiner et actualiser la recherche en spécifiant des critères dans la barre d'outils de la fenêtre Rechercher dans les fichiers
- 'EA User Guide' - Afficher la description de l'élément de code dans le *Guide de l'Utilisateur d'Enterprise Architect*
- 'Google' - Afficher les résultats d'une recherche Google sur le texte
- 'MSDN' - Afficher les résultats d'une recherche sur le texte dans le Microsoft Developer Network (MSDN)
- 'Sun Java SE' - Afficher facilité les résultats d'une recherche sur le texte dans 'Sun Search' de Sun Microsystems
- 'Wikipedia' - Afficher n'importe quelle entrée sur l' object sur le site Web Wikipédia
- 'Koders' - Afficher les résultats d'une recherche de la string de texte sur Koders.com

Rechercher Intelli-sense

Effectuez une recherche sur la string spécifiée à l'aide du service ou de la bibliothèque Code Miner spécifié dans le script Analyzer actuel. Les résultats sont affichés dans l'onglet ' Code Miner ' de la fenêtre Rechercher dans les fichiers.

Touche de raccourci : Maj+F12

Régler Débogueur sur Ligne

(Si le débogueur est en cours d'exécution et a atteint un point d'arrêt.) Déplacez le point d'exécution vers la ligne actuelle. Vérifiez que vous n'ignorez aucun code ou déclaration affectant la section suivante du code en cours de débogage.

Variable d'affichage

(Si le débogueur est en cours d'exécution.) Ouvrez la fenêtre Locals et mettez en surbrillance la variable locale pour le point actuel du code.

Afficher dans la visionneuse String

Affichez le contenu complet d'une string variable dans le visualiseur String .

Créer un cas d'utilisation pour ' string '

Affichez la dialogue 'Créer un cas d'utilisation pour la méthode', à travers laquelle vous créez un cas d'utilisation pour la méthode contenant la string de texte.

Point d'Arrêt

Affichez un sous-menu d'options pour créer un marqueur d'enregistrement sur la ligne de code sélectionnée. Les marqueurs d'enregistrement que vous pouvez ajouter incluent :

- Point d'Arrêt
- Démarrer Enregistrement
- Marqueur de fin Enregistrement
- Marqueur de capture automatique de pile
- Marqueur d'enregistrement automatique de la méthode
- Point de Trace

Testpoints

Afficher les options pour ajouter un nouveau Testpoint , afficher le gestionnaire Testpoints (fenêtre Testpoints) ou modifier un Testpoint existant si un ou plusieurs sont déjà définis à l'emplacement sélectionné.

	(Les sous-options dépendent du type de fichier de code que vous réviser .)
Validation XML	Permet de vérifier la conformité d'un document XML avec ses propres références de schéma ou à l'aide d'un schéma spécifié par l'utilisateur ; soit un fichier de schéma local, soit une URL.
Ouvrir (Fermer) IME	Ouvrez (ou fermez) l'éditeur de méthode de saisie afin de pouvoir saisir du texte dans une langue étrangère sélectionnée, telle que le japonais. Vous définissez facilité la langue du clavier à l'aide du Panneau de configuration Windows - Options régionales et linguistiques.
Copier le lien hypertexte de position	Copie la position du curseur sous forme de lien hypertexte qui peut être collé dans les éditeurs Rich Notes , comme un message dans l'onglet « Chat » de la fenêtre Chat et courrier. Utilisez simplement l'option de menu contextuel « Coller » dans le message et spécifiez le texte du lien. Le lecteur peut cliquer sur le lien pour ouvrir le fichier source et déplacer le curseur vers la position sélectionnée dans le fichier.
Copier le lien hypertexte du texte	Copie la string de texte sélectionnée sous forme de lien hypertexte pouvant être collé dans les éditeurs Rich Notes , comme un message dans l'onglet « Chat » de la fenêtre Chat et courrier. Utilisez simplement l'option de menu contextuel « Coller » dans le message. Le lecteur peut cliquer sur le lien pour ouvrir le fichier source et déplacer le curseur vers la première occurrence de cette string de texte dans le fichier.
Numéros de ligne	(Éditeur de Script uniquement.) Afficher ou masquer les numéros de ligne de code sur le côté gauche de l'écran de l'éditeur.
Annuler **Couper** **Copie** **Pâte** **Supprimer** **Tout sélectionner**	Ces six options fournissent des fonctions simples pour éditer le code.

Notes

- Les options dans la moitié inférieure du sous-menu « Rechercher <chaîne> » (après « Rechercher dans Scripts ») sont configurables ; vous pouvez ajouter de nouveaux outils de recherche ou supprimer ceux existants en éditant le fichier searchProviders.xml dans le dossier Sparx Systems > EA > Config - ce fichier est au format de document de description OpenSearch

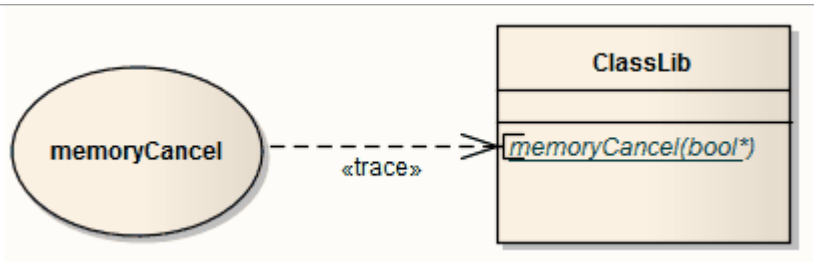
Créer un cas d'utilisation pour la méthode

À l'aide du menu contextuel de l'éditeur de code, vous pouvez créer un élément de cas d'utilisation pour une méthode que vous sélectionnez dans le code. Vous pouvez aussi:

- Lier le cas d'utilisation directement à la méthode
- Ajoutez la classe parent à un diagramme (si elle n'est pas déjà dans le diagramme sélectionné) et/ou ajoutez l'élément Use Case au diagramme
- Bloc l'affichage des attributs ou des méthodes qui ne sont pas également les cibles des liens fonctionnalité

Créer un cas d'utilisation pour une méthode, via l'éditeur de code

Étape	Action
1	(Si vous souhaitez représenter le cas d'utilisation et son lien vers la méthode dans un diagramme), cliquez sur le nom du diagramme dans la fenêtre Navigateur .
2	Dans l'éditeur de code, cliquez-droit sur le nom de la méthode ou sur n'importe quelle partie du corps de la méthode, et sélectionnez l'option 'Créer une méthode pour <nom de la méthode>'. La dialogue « Créer un cas d'utilisation pour la méthode » s'affiche.
3	La fonction de base de cette dialogue est de créer un cas d'utilisation pour la méthode sélectionnée : • Si c'est tout ce qui est requis, cliquez sur le bouton OK ; l'élément Use Case est créé dans la fenêtre Navigateur , dans le même Paquetage que la Classe parent de la méthode, et avec le même nom que la méthode • Si vous avez l'intention de rendre la relation tangible, poursuivez la procédure
4	Pour créer un connecteur Trace reliant le cas d'utilisation à la méthode, cochez la case « Lier le cas d'utilisation à la méthode ».
5	Pour ajouter la classe parent de la méthode au diagramme , si elle n'y est pas déjà, cochez la case 'Ajouter une classe au Diagramme '.
6	Pour ajouter le cas d'utilisation nouvellement créé au diagramme , cochez la case « Ajouter un cas d'utilisation au Diagramme » ; cela afficherait maintenant le connecteur Use Case, Class et Trace sur le diagramme .
7	Pour afficher uniquement les fonctionnalités (attributs et méthodes) de la classe parent qui sont les cibles des relations 'lien vers fonctionnalité ', cochez la case 'Afficher uniquement fonctionnalités liées dans la classe'. La classe peut contenir n'importe quel nombre d'attributs et de méthodes, mais ceux sans relation « lien vers fonctionnalité » sont masqués.
8	Cliquez sur le bouton OK pour créer et décrire le cas d'utilisation et la relation ; si vous avez sélectionné toutes les options, le diagramme contient maintenant des éléments liés ressemblant à cette illustration :


ClassLib
memoryCancel
«trace»
memoryCancel(bool*)

Fonctions Éditeur de Code

L' Éditeur de Code commun offre une variété de fonctions pour faciliter le processus d'édition de code, notamment :

- Mise en évidence de la syntaxe
- Signets
- Historique du curseur
- Correspondance des attelles
- Indentation automatique
- Commenter les sélections
- Guides de portée
- Zoom
- Sélection de ligne
- Intelli-sens
- Trouver et remplacer
- Rechercher dans les fichiers

Une gamme de ces fonctions est disponible via des combinaisons de touches du clavier et/ou des options de menu contextuel.

Vous pouvez personnaliser plusieurs fonctionnalités de l' Éditeur de Code en définissant des propriétés dans les fichiers de configuration de l' Éditeur de Code ; par exemple, par défaut, la ligne contenant le curseur est toujours mise en surbrillance, mais vous pouvez désactiver la mise en surbrillance.

Détails de la fonction

Fonctions Éditeur de Code

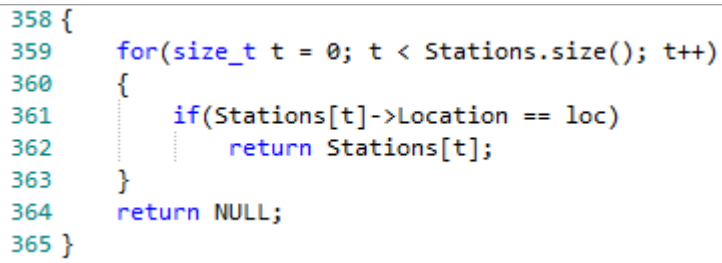
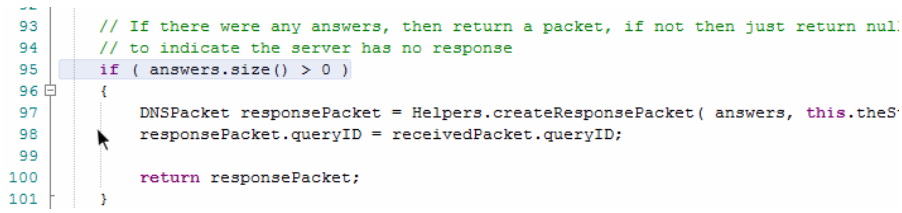
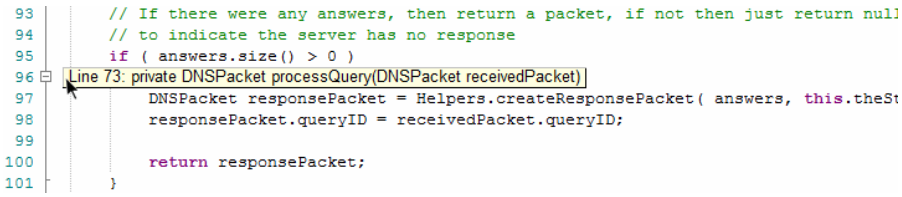
Fonction	Description

Mise en évidence de la syntaxe	L' Éditeur de Code met en évidence - en texte coloré - la syntaxe de code standard de tous les formats de fichiers de langage supportés par Enterprise Architect
	`1 #pragma once`
	`2 #include "afxwin.h"`
	`3 #include "afxcmn.h"`
	`4`
	`5`
	`6 // CToolBox dialog`
	`7`
	`8 class CToolBox : public CDialog`
	`9 {`
	`10 DECLARE_DYNAMIC(CToolBox)`
	`11 CRect m_rect;`
	`12 int m_offset;`
	Vous pouvez définir comment l' Éditeur de Code implémente la coloration syntaxique pour chaque langage, via la page ' Éditeurs de Code ' de la dialogue 'Préférences'.

Signets	Les signets désignent une ligne d'intérêt dans le document ; vous pouvez les activer et les désactiver pour une ligne particulière en appuyant sur Ctrl+F2. De plus, vous pouvez appuyer sur F2 et Shift+F2 pour accéder au signet suivant ou précédent dans le document. Pour effacer tous les signets du fichier de code, appuyez sur Ctrl+Maj+F2.
Historique du curseur	L' Éditeur de Code Control conserve un historique des 50 positions précédentes du curseur ; une entrée dans la liste historique est créée lorsque : • Le curseur est déplacé de plus de 10 lignes par rapport à sa position précédente • Le curseur est déplacé lors d'une opération de recherche/remplacement Vous pouvez accéder à un point antérieur dans l'historique du curseur en appuyant sur Ctrl+- et à un point ultérieur en appuyant sur Ctrl+Shift+-.

Correspondance des attelles	Lorsque vous placez le curseur sur une accolade ou une parenthèse, l' Éditeur de Code met en évidence son partenaire correspondant ; vous pouvez ensuite accéder à l'accolade correspondante en appuyant sur Ctrl+E.
	`28 function ProtectedFunctionTest: boolean;`
	`29 procedure ProtectedProcedureTest(a: WideString);`

Indentation automatique	Pour chaque langage supporté, l' Éditeur de Code ajuste l'indentation d'une nouvelle ligne en fonction de la présence d'instructions de contrôle ou de jetons de bloc de portée dans les lignes précédant la position du curseur.

	 Les niveaux de retrait sont indiqués par des lignes horizontales pâles. Vous pouvez également indenter manuellement les lignes et blocs de code sélectionnés en appuyant sur la touche Tab ; pour annuler l'indentation du code sélectionné, appuyez sur Maj+Tab.
Commenter les sélections	Pour les langages support les commentaires, l' Éditeur de Code peut commenter des sélections entières de code. L' Éditeur de Code reconnaît deux types de commentaires : • Commentaire de ligne : des lignes entières sont commentées depuis le début (par exemple : // Ceci est un commentaire) • Commentaires de flux : les sections d'une ligne sont commentées à partir d'un point de départ spécifié jusqu'à un point final spécifié (par exemple : /* Ceci est un commentaire */) Vous pouvez basculer les commentaires sur la ligne ou la sélection actuelle en appuyant sur : • Ctrl+Shift+C pour les commentaires de ligne, ou • Ctrl+Shift+X pour les commentaires du flux
Guides de portée	Si le curseur est placé sur un marqueur d'indentation, l' Éditeur de Code effectue un « retour en arrière » pour retrouver la ligne qui a commencé la portée à ce niveau d'indentation ; si la ligne est trouvée et est actuellement à l'écran, elle est surlignée en bleu clair.  Alternativement, si la ligne est hors écran, une info-bulle s'affiche pour indiquer le numéro de la ligne et son contenu : 
Zoom	Vous pouvez zoomer et dézoomer sur le contenu de l' Éditeur de Code en utilisant : • Ctrl+clavier + et • Ctrl+clavier - Le zoom peut être restauré à 100 % en utilisant Ctrl+clavier /.

Sélection de ligne	Si vous souhaitez déplacer le curseur vers une ligne de code spécifique, appuyez sur Ctrl+G et, en réponse à l' prompt , saisissez le numéro de ligne. Appuyez sur le bouton OK ; l'éditeur affiche la ligne de code spécifiée avec le curseur à gauche.

Intelli-sens

Intelli-sense est une fonctionnalité qui propose des choix d'éléments de code et de valeurs au fur et à mesure que vous tapez. Tous les éditeurs de code n'utilisent pas Intelli-sense ; par exemple, Intelli-sense est désactivé lorsque vous enregistrez une macro dans la Visionneuse de code source .

Intelli-sense vous fournit une assistance contextuelle via des listes de saisie semi-automatique, des astuces d'appel et des informations de survol de la souris.

Facilités

Facilité	Description

Liste de saisie semi-automatique	Une liste de saisie semi-automatique fournit une liste de complétions possibles pour le texte actuel ; la liste est automatiquement invoquée lorsque vous entrez un jeton d'accesseur (tel qu'un point ou un pointeur d'accès) après un object ou un type contenant des membres.
	57 public void memoryRecall()
	58 {
	59 this.
	60 }
	61
	62 public
	63
	64 }
	65
	66 public mber1, int number2)
	67 {
	68 in r1 + number2;
	69 re
	70 }
	~ClassLib
	Dispose
	finalize
	m_delivery
	memory
	memoryCancel
	memoryMinus
	memoryPlus
	memoryRecall
	Vous pouvez également appeler la liste de saisie semi-automatique manuellement en appuyant sur Ctrl+Espace ; l' Éditeur de Code recherche ensuite les correspondances pour le mot menant au point d'invocation.
	Sélectionnez un élément dans la liste et appuyez sur la touche Entrée ou sur la touche Tab pour insérer l'élément dans le code ; pour fermer la liste de saisie semi-automatique, appuyez sur Échap.

Astuces d'appel	Les calltips affichent la signature de la méthode actuelle lorsque vous tapez le jeton de liste de paramètres (par exemple, une parenthèse ouvrante) ; si la méthode est surchargée, la calltip affiche des flèches que vous pouvez utiliser pour naviguer parmi les différentes signatures de méthode

	20 //PostDraw Adornments 21 //Stereotyped Static Adornments 22 //Add Stakeholder's STAKE 23 setpenwidth(; 24 // Add a the SetPenWidth(**int penwidth**) 25 startpath(); 26 moveto(25,37); 27 lineto(25,52); 28 endpath(); 29 strokepath(); 30 //Add tip
Informations sur le survol de la souris	Vous pouvez afficher la documentation de support pour les éléments de code (par exemple, les attributs et les méthodes) en passant le curseur sur l'élément en question. 11 dockable = "none"; 12 string 13 Dock elements together. Tagged V 14 Valid Values: none, standard 15 //PreDraw Derived Attribute I

Trouver et remplacer

Chacun des éditeurs de code d' Enterprise Architect facilite la recherche et le remplacement de termes dans l'éditeur, via la dialogue « Rechercher et remplacer ».

Accéder

Raccourcis Clavier	Mettez en surbrillance la string de texte requise et appuyez sur : • Ctrl+F pour les contrôles de recherche uniquement, ou • Ctrl+R pour les contrôles Rechercher et Remplacer Dans chaque cas, le champ « Rechercher » est renseigné avec le texte actuellement sélectionné dans l'éditeur. Si aucun texte n'est sélectionné dans l'éditeur, le champ « Rechercher » est renseigné avec le mot à la position actuelle du curseur. Si aucun mot n'existe à la position actuelle du curseur, le dernier terme recherché est utilisé.

Opérations de base - Commandes

Commande	Action
Rechercher suivant	Localisez et mettez en surbrillance l'instance suivante (par rapport à la position actuelle du curseur) du texte spécifié dans le champ « Rechercher ».
Remplacer	Remplacez l'instance actuelle du texte spécifié dans le champ "Rechercher quoi" par le texte spécifié dans le champ "Remplacer par", puis localisez et mettez en surbrillance l'instance suivante (par rapport à la position actuelle du curseur) du texte spécifié dans le champ "Rechercher". Trouver quoi'champ.
Remplace tout	Remplacez automatiquement toutes les instances du texte spécifié dans le champ « Rechercher » par le texte spécifié dans le champ « Remplacer par ».

Opérations de base - Options

Option	Action
Cas de correspondance	Spécifiez que la casse de chaque caractère de la string de texte dans le champ « Rechercher » est importante lors de la recherche de correspondances dans le code.
Correspond à un mot entier	Spécifiez que la string de texte dans le champ « Rechercher » est un mot complet et ne doit pas correspondre à des instances du texte qui font partie d'une string plus longue. Par exemple, les recherches sur ARE ne doivent pas correspondre à ces lettres dans les instances des mots AREA ou ARENA.

Rechercher	Effectuez la recherche à partir de la position actuelle du curseur jusqu'au début du fichier, plutôt que dans la direction par défaut de la position actuelle du curseur jusqu'à la fin du fichier.
Utiliser des expressions régulières	Évaluez des séquences de caractères spécifiques dans les champs « Rechercher quoi » et « Remplacer par » en tant qu'expressions régulières.

Concepts

Concept	Description
Expressions régulières	Une expression régulière est une définition formelle d'un Motif de recherche, qui peut être utilisée pour faire correspondre des caractères, des mots ou motifs de caractères spécifiques. Par souci de simplicité, le mécanisme « rechercher et remplacer » de l' Éditeur de Code ne supporte qu'un sous-ensemble de la grammaire standard des expressions régulières. Le texte des champs « Rechercher » et « Remplacer par » n'est interprété comme une expression régulière que si la case « Utiliser des expressions régulières » est cochée dans la dialogue « Rechercher et remplacer ».
Métaséquences	Si la case « Utiliser les expressions régulières » est cochée, la plupart des caractères du champ « Rechercher » sont traités comme des littéraux (c'est-à-dire qu'ils correspondent uniquement à eux-mêmes). Les exceptions sont appelées métaséquences ; chaque métaséquence reconnue dans la dialogue 'Rechercher et remplacer' de l' Éditeur de Code est décrite dans ce tableau : • \< - Indique que le texte est le début d'un mot ; par exemple : \<cat correspond à *catastrophe* et *cataclysm* , mais ne peut pas *être concaténé* • \> - Indique que le texte est la fin d'un mot ; par exemple : hat\> correspond à *cela* et *chat* , mais pas à *la haine* • (...) - Indique des caractères uniques alternatifs pouvant correspondre - les caractères peuvent être spécifiques (chr) ou dans une plage alphabétique ou numérique (am) ; par exemple : (hc) at correspond à *hat* et *cat* mais pas *bat* , et (am) Class correspond à n'importe quel nom dans la plage *aClass-mClass* • (^...) - Indique des caractères uniques alternatifs qui doivent être exclus d'une correspondance - les caractères peuvent être spécifiques (^chr) ou dans une plage alphabétique ou numérique (^am) ; par exemple : (^hc) at correspond à *rat* et *bat* , mais *hat* et *cat* sont exclus, et (^am) Class correspond à n'importe quel nom compris dans la plage *nClass* à *zClass* , mais *aClass* à *mClass* sont exclus • ^ - Correspond au début d'une ligne • $ - Correspond à la fin d'une ligne • * - Correspond au caractère (ou jeu de caractères) précédent 0 fois ou plus ; par exemple : ba*t correspond à *bt* , *bat* , *baat* , *baaat* et ainsi de suite, et b(ea) *t correspond à *bt* , *bet* , *bat* , *beat* , *beet* , *baat* et ainsi de suite • + - Correspond au caractère (ou au jeu de caractères) précédent 1 ou plusieurs fois ; par exemple : ba+t correspond à *bat* , *baat* et *baaat* mais pas à *bt* , et b(ea) +t correspond à *bet* , *bat* , *beat* , *beet* et *baat* mais pas *à bt* Si une métaséquence d'un seul caractère est précédée d'une barre oblique inverse

	(\), elle est traitée comme un caractère littéral : c\(at\) correspond à c(at) car les crochets sont traités littéralement. Lorsque la case « Utiliser les expressions régulières » est cochée, un menu d'aide à la métaséquence est disponible à droite des champs « Rechercher » et « Remplacer par » ; la sélection d'une métaséquence dans ce menu insère la métaséquence dans le champ, remplaçant ou enveloppant le texte actuellement sélectionné, selon le cas.
Régions marquées	Lors de la recherche et du remplacement avec des expressions régulières, jusqu'à neuf sections du terme d'origine peuvent être remplacées par le terme de remplacement. Les métaséquences '\(' et '\)' désignent le début et la fin d'une région étiquetée ; la section du texte correspondant qui se trouve dans la région balisée peut être incluse dans le texte de remplacement avec la métaséquence « \n » (où *n* est le numéro de région balisée entre 1 et 9). Par exemple: Rechercher : *les choses de* \((A-Za-z) +\) Remplacer par *des éléments appartenant à \1* Texte original : *Ce sont toutes les affaires de Michael* . Texte remplacé : *Ce sont tous des éléments qui appartiennent à Michael.*

Rechercher dans les fichiers

Les recherches de texte de fichier sont fournies par la fenêtre Rechercher dans les fichiers et depuis les Éditeurs de Code , pour rechercher dans les fichiers les noms et les structures des données. Ces fichiers peuvent être des fichiers de code externes, des fichiers de code que vous avez déjà ouverts dans Enterprise Architect , des scripts de modèle internes ou le sous-système d'aide.

L'onglet « Recherche de fichiers » conserve un historique des chemins de fichiers que vous avez explorés, vous aidant ainsi à revenir rapidement aux dossiers fréquemment utilisés dans votre système de fichiers. Vous pouvez de la même manière sélectionner une string de recherche précédemment utilisée si vous devez répéter une recherche plusieurs fois. Lorsque vous recherchez des fichiers de code, vous pouvez également limiter la recherche à des fichiers de types spécifiques, en sélectionnant les extensions de fichier, et inclure uniquement le dossier sélectionné ou tous ses sous-dossiers. Une autre facilité utile est de pouvoir choisir d'afficher les résultats de la recherche soit sous la forme d'une liste de chaque instance de la string , soit sous la forme d'une liste de fichiers contenant la string avec les instances regroupées sous le fichier dans lequel elles se trouvent.

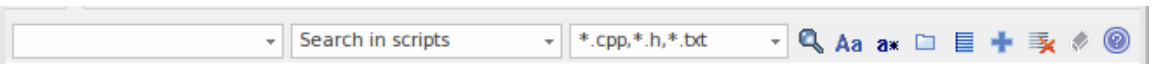
Pour toutes les recherches, vous pouvez qualifier la recherche pour qu'elle soit sensible à la casse et/ou pour faire correspondre la string de recherche aux mots complets.

Accéder

Ruban	Explorer > Rechercher > Fichiers Exécuter > Source > Rechercher Exécuter > Source > Modifier > Rechercher dans les fichiers
Menu Contexte	Cliquez-droit sur le texte sélectionné | Rechercher <texte sélectionné> | Rechercher dans les fichiers
Raccourcis Clavier	F12, Ctrl+Maj+Alt+F

Barre d'outils de recherche

Vous pouvez utiliser les options de la barre d'outils dans la fenêtre Rechercher dans les fichiers pour contrôler l'opération de recherche. L'état de chaque bouton persiste dans le temps pour toujours refléter vos critères de recherche précédents.



Possibilités

Option	Action
	Le champ « Texte de recherche ». Type la string de texte à rechercher. Tout texte que vous saisissez est automatiquement enregistré dans la liste déroulante, jusqu'à un maximum de dix chaînes ; le texte ajouté ensuite écrase la string de texte la plus ancienne de la liste. Vous pouvez cliquer sur la flèche déroulante et sélectionner l'une de ces chaînes de texte enregistrées, si vous préférez.

VAR more search text EALib Script SQL Verify Unit Test GUID Priority VAR	
Search in local help... more Browse for folder... Search in scripts Search in open files... Search in local help... c:\ C:\EA\VEA\Microsoft Native\CityLoop	Le champ « Chemin de recherche ». Spécifiez le dossier dans lequel rechercher ou le type de recherche. Vous pouvez saisir le chemin du dossier pour rechercher directement dans la zone de texte, ou cliquer sur la flèche déroulante et sélectionner « Rechercher un dossier » pour effectuer une recherche à l'aide de la dialogue « Rechercher un dossier ». Tous les chemins que vous saisissez sont automatiquement enregistrés dans la liste déroulante, jusqu'à un maximum de dix ; les chemins ajoutés par la suite écrasent le chemin le plus ancien de la liste. Vous pouvez sélectionner l'un de ces chemins enregistrés si vous préférez. Outre « Rechercher un dossier », il existe trois autres options fixes dans la liste déroulante : • « Rechercher dans les scripts », qui recherche les scripts locaux et définis par l'utilisateur dans la fenêtre Scriptant • « Rechercher dans les fichiers ouverts », qui limite la recherche aux fichiers que vous avez ouverts dans Enterprise Architect • « Rechercher dans l'aide locale », qui recherche les fichiers d'aide locaux installés à partir du site Web Sparx Systems ; les résultats répertorient les rubriques d'aide contenant le terme recherché, ainsi que le numéro de ligne et la ligne dans laquelle le texte apparaît Ces options désactivent la zone de liste « Rechercher les types de fichiers ».
.cs,.txt *.cpp,*.h,*.txt *.c,*.h *.java,*.txt *.cs,*.txt	Le champ « Rechercher les types de fichiers ». Cliquez sur la flèche déroulante et sélectionnez les types de fichiers (extensions de fichiers) à rechercher.
	Cliquez sur cette icône pour lancer la recherche. Au cours de la recherche, tous les autres boutons de la barre d'outils sont désactivés. Vous pouvez annuler la recherche à tout moment en cliquant à nouveau sur le bouton Rechercher. Si vous changez l'un de ces boutons à bascule, vous devez exécuter à nouveau la recherche pour modifier le résultat.
Aa	Cliquez sur cette icône pour basculer la sensibilité à la casse de la recherche. Le message de l'info-bulle identifie le paramètre actuel.
a*	Cliquez sur cette icône pour basculer entre la recherche de n'importe quelle correspondance et la recherche uniquement des correspondances qui forment un mot entier. Le message de l'info-bulle identifie le paramètre actuel.
	Cliquez sur cette icône pour basculer entre limiter la recherche à un seul chemin et inclure tous les sous-dossiers sous ce chemin. Le message de l'info-bulle identifie le paramètre actuel.

	Cliquez sur cette icône pour sélectionner le format de présentation des résultats de la recherche ; vous avez deux options : • List Vue - (comme indiqué) chaque ligne de résultat se compose du chemin du fichier et du numéro de ligne, suivis du texte de la ligne ; plusieurs lignes d'un fichier sont répertoriées sous forme d'entrées distinctes • Tree Vue - ( ) chaque ligne de résultat se compose du chemin du fichier qui correspond aux critères de recherche et du nombre de lignes correspondant au texte de recherche dans ce fichier ; vous pouvez développer l'entrée pour afficher le numéro de ligne et le texte de chaque ligne
	Cliquez sur cette icône pour ajouter un nouvel onglet de recherche. Vous pouvez créer jusqu'à quatre nouveaux onglets de recherche. Les recherches peuvent également exécuter simultanément.
	Cliquez sur cette icône pour effacer les résultats.
	Si nécessaire, cliquez sur cette icône pour supprimer toutes les entrées des listes déroulantes Chemin de recherche, Texte de recherche et Types de fichiers de recherche.

Trouver un fichier

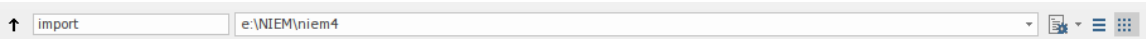
L'onglet « Rechercher un fichier » de la fenêtre Rechercher dans les fichiers fournit un outil qui peut vous aider à trouver des fichiers plus rapidement. L'onglet agit comme un explorateur de système de fichiers et offre une alternative rapide au dialogue d'ouverture de fichier commun. Les recherches de fichiers sont rapides et simples, vous permettant de rechercher des fichiers qui vous intéressent sans perdre votre flux de travail actuel. L'affichage peut être commuté entre la vue rapport et la vue liste.

Accéder

Ruban	Explorer > Rechercher > Fichiers > Rechercher un fichier
Raccourcis Clavier	Ctrl+Maj+Alt+F

Barre d'outils

La barre d'outils fournit une zone de liste déroulante de filtre de recherche et de navigation dans les dossiers. La barre d'outils fournit des options pour mémoriser les emplacements de recherche et alterner entre les vues de liste et de rapport.



Possibilités

	Cliquez pour accéder au dossier parent.
	Le contrôle de filtre vous permet d'exclure les fichiers qui ne correspondent pas aux critères que vous saisissez. Le symbole générique * est automatiquement ajouté au texte, il n'est donc pas nécessaire de l'ajouter vous-même. Pour rechercher tous les fichiers contenant le terme « jvm », tapez simplement « jvm ». Pour trouver des images .png contenant le terme « rouge », vous pouvez taper *red*.png. Appuyez sur la touche Entrée pour mettre à jour les résultats.
	Entrez le chemin d'un répertoire et appuyez sur la touche Entrée pour afficher les fichiers à cet emplacement Utilisez la liste déroulante pour sélectionner parmi les emplacements marqués pour le modèle actuel. Les emplacements peuvent être gérés à l'aide du menu de la barre d'outils.
	Permet de gérer les emplacements affichés dans le combo répertoire. • Mémoriser le chemin - stocke la valeur actuelle du champ « Répertoire » de sorte que, lorsque vous revenez ultérieurement à la fenêtre Rechercher dans les fichiers, le champ « Répertoire » soit par défaut à cette valeur (si c'est la seule valeur « mémorisée ») ou propose la valeur dans la liste déroulante • Oublier le chemin - efface la valeur actuelle de la mémoire afin qu'elle ne soit

	pas proposée comme valeur possible pour le champ « Répertoire » • Mémoriser le filtre - stocke la valeur actuelle dans le champ « Filtre » de sorte que lorsque vous revenez ultérieurement à la fenêtre Rechercher dans les fichiers, le champ « Filtre » prend par défaut cette valeur • Oublier le filtre : supprime la valeur du champ "Filtre" de la mémoire afin qu'elle ne soit pas placée dans le champ la prochaine fois que vous accéderez à la fenêtre
	Dans cette vue, la liste affiche les colonnes 'Nom', 'Date de modification', ' Type ' et 'Taille'. Les colonnes peuvent être triées par ordre croissant ou décroissant. Cliquez une troisième fois sur la colonne pour supprimer l'ordre de tri.
	La vue liste supprime les colonnes et est pratique lorsqu'un dossier contient de nombreux fichiers.

Raccourcis Clavier

	Ensembles se concentrent sur le contrôle du filtre.
	Permet d'accéder au dossier parent.
	Permet d'accéder au dossier parent.
	Si un dossier est sélectionné, ouvre le dossier, sinon ouvre les fichiers sélectionnés.

Rechercher Intelli-sense

Les capacités Intelli-sense d' Enterprise Architect sont construites à l'aide de l'outil Code Miner de Sparx Systems . Le Code Miner offre un accès rapide et complet aux informations contenues dans une base de code existante. Le système offre un accès complet à tous les aspects du code source d'origine, soit « à la volée » comme on pourrait le faire dans un éditeur de code, soit sous forme de résultats de recherche produits par des requêtes écrites dans le langage Code Miner mFQL.

Accéder

Dans la fenêtre Rechercher dans les fichiers, cliquez sur l'onglet « Code Miner ».

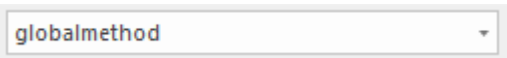
Ruban	Explorer > Rechercher > Fichiers
Raccourcis Clavier	Ctrl+Maj+Alt+F

Le contrôle Code Miner

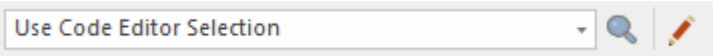
Ce contrôle présente une interface permettant d'effectuer des requêtes sur plusieurs bases de code à la fois. Les bases de code qu'il utilise sont des bases de données construites à l'aide de l'outil Code Miner d' Enterprise Architect . Ces bases de données forment une bibliothèque, qui peut également être partagée lorsqu'elle est déployée en tant que service. Les requêtes pouvant être exécuter sont répertoriées et sélectionnées à l'aide de la barre d'outils, qui permet d'accéder facilement au code source des requêtes, pour l'édition et la composition. Les requêtes n'ont pas besoin d'être compilées ; ils sont visualisés, édités et enregistrés comme n'importe quel fichier de code source. Les requêtes qui prennent un seul paramètre peuvent utiliser n'importe quelle sélection dans un éditeur de code ouvert. L'interface prend également supporte la saisie manuelle des paramètres pour les requêtes qui prennent plusieurs arguments.

Le premier contrôle de la barre d'outils répertorie les espaces de noms disponibles. La sélection d'un espace de noms limite les requêtes affichées à celles de cet espace de noms.

Le contrôle suivant fournit une liste déroulante de toutes les requêtes du fichier de requête pour l'espace de noms sélectionné.



Le troisième contrôle est une zone de liste déroulante d'édition. Par défaut, un seul paramètre de requête est extrait du texte sélectionné dans un éditeur de code ouvert, mais vous pouvez également saisir le(s) paramètre(s) directement dans ce champ. Plusieurs paramètres doivent être séparés par des virgules. Ceci est suivi du bouton Rechercher pour exécuter la requête. Les requêtes peuvent être modifiées à tout moment à l'aide du bouton Modifier à côté du bouton Rechercher.



Le panneau 'Résultat' est une arborescence qui répertorie les résultats de la requête regroupés par fichier.

Result

- e:\java\jdk-1.8.0_91\src\com\sun\org\apache\xerces\internal\util\domutil.java
- e:\java\jdk-1.8.0_91\src\java\util\stream\pipelinehelper.java
- e:\java\jdk-1.8.0_91\src\java\util\vector.java
- e:\java\jdk-1.8.0_91\src\javax\swing\defaultlistmodel.java

Bibliothèques Code Miner

Les bibliothèques Code Miner sont un ensemble de bases de données qui peuvent être utilisées par les fournisseurs Enterprise Architect Intelli-sense pour obtenir et interroger des informations sur plusieurs bases de code. Chaque base de données est créée à partir du répertoire racine du code source d'une base de code, en utilisant une grammaire spécialisée appropriée à son langage (C++, Java ou C#).

Les bibliothèques sont créées, mises à jour, supprimées ou ajoutées dans l' Éditeur de Script Analyseur . Un scénario typique d'utilisation de cette fonctionnalité serait de créer une base de données pour un projet de développement et des bases de données supplémentaires pour les frameworks référencés par le projet. Votre base de données de développement peut être mise à jour fréquemment à mesure que les modifications de code s'accumulent, tandis que les frameworks statiques seraient mis à jour moins souvent. Les bibliothèques peuvent être recherchées de la même manière que l'outil « Recherche de fichiers », mais Code Miner offre des capacités de recherche avancées grâce à son langage mFQL.

- Plusieurs domaines/frameworks peuvent être recherchés simultanément
- Une requête peut être exécuter en une fraction du temps requis pour une recherche de fichier
- Les requêtes peuvent être codées pour faciliter les critères de recherche complexes
- Les requêtes peuvent prendre plusieurs paramètres
- Tous les fichiers sont indexés sur la base de constructions UML équivalentes, permettant des recherches intelligentes produisant des résultats significatifs dans un cadre modélisation

Fichiers Query Code Miner

Les requêtes Code Miner sont conservées dans un seul fichier de code source qui doit avoir l'extension .mFQL. Un ensemble de requêtes de base est fourni avec chaque installation Enterprise Architect ; ceux-ci peuvent être situés dans le sous-répertoire config\codeminer. Ce fichier de requête doit être nommé par défaut dans tout script Analyzer que vous modifiez.

Avant de modifier une requête, il est conseillé de copier ce fichier vers un emplacement de travail et de nommer la copie dans n'importe quel script Analyzer que vous utilisez. De cette façon, vous aurez toujours un fichier de référence auquel revenir.

Les requêtes sont mieux considérées comme des fonctions écrites dans le langage mFQL. En tant que tels, ils ont des noms uniques, peuvent être qualifiés par un seul espace de noms et peuvent spécifier des paramètres. Le fichier fournit les requêtes répertoriées dans la barre d'outils du contrôle Intelli-sense. Chaque fois que les modifications apportées à un fichier de requête sont enregistrées, les requêtes répertoriées dans la zone de liste déroulante de la barre d'outils de recherche seront mises à jour en conséquence. Cette image est un exemple de requête simple écrite en mFQL.

```
188
189 namespace java
190 {
191 //
192 // Find all references
193 //
194 query::findByName($param1)
195 {
196     distinct(GetByValue( $param1 +))
197 }
198
199 query::findMethodByName($name)
200 {
201     move( 1, "METHOD", intersect( GetByNode("NAME"), GetByValue( $name ) )  )
202 }
203
204 query::findMethodCall($name)
205 {
206     filter( "METHOD_ACCESS", intersect(GetByNode("NAME"), GetByValue( $name )) )
207 }
208
```

Éditeur de Code Key Bindings

Clés

Clé	Description
Ctrl+G	Déplacer le curseur vers une ligne spécifiée
↓	Déplacer le curseur d'une ligne vers le bas
Maj+↓	Étendre la sélection sur une ligne
Ctrl+↓	Faites défiler une ligne vers le bas
Alt+Maj+↓	Étendre la sélection rectangulaire sur une ligne
↑	Déplacer le curseur d'une ligne vers le haut
Maj+↑	Étendre la sélection d'une ligne
Ctrl+↑	Faire défiler une ligne vers le haut
Alt+Maj+↑	Étendre la sélection rectangulaire d'une ligne
Ctrl+(	Déplacer le curseur d'un paragraphe vers le haut
Ctrl+Maj+(	Étendre la sélection d'un paragraphe
Ctrl+)	Déplacer le curseur d'un paragraphe vers le bas
Ctrl+Maj+)	Étendre la sélection d'un paragraphe vers le bas
←	Déplacer le curseur d'un caractère vers la gauche
Maj+←	Étendre la sélection à un caractère gauche
Ctrl+←	Déplacer le curseur d'un mot vers la gauche
Ctrl+Maj+←	Étendre la sélection d'un mot à gauche
Alt+Maj+←	Étendre la sélection rectangulaire vers la gauche d'un caractère
→	Déplacez le curseur d'un caractère vers la droite.
Maj+→	Étendre la sélection d'un caractère vers la droite
Ctrl+→	Déplacer le curseur d'un mot vers la droite
Ctrl+Maj+→	Étendre la sélection d'un mot à droite

Alt+Maj+→	Étendre la sélection rectangulaire d'un caractère vers la droite
Ctrl+/	Déplacer le curseur d'une partie de mot vers la gauche
Ctrl+Maj+/	Étendre la sélection vers la gauche d'une partie de mot
Ctrl+\	Déplacer le curseur vers la droite d'une partie de mot
Ctrl+Maj+\	Étendre la sélection vers la droite d'une partie de mot
Maison	Déplacer le curseur au début de la ligne actuelle
Maj+Accueil	Étendre la sélection au début de la ligne actuelle
Ctrl+Accueil	Déplacer le curseur au début du document
Ctrl+Maj+Accueil	Étendre la sélection au début du document
Alt+Accueil	Déplacer le curseur au début absolu de la ligne
Alt+Maj+Accueil	Étendre la sélection rectangulaire jusqu'au début de la ligne
Fin	Déplacer le curseur à la fin de la ligne actuelle
Maj+Fin	Étendre la sélection jusqu'à la fin de la ligne actuelle
Ctrl+Fin	Déplacer le curseur à la fin du document
Ctrl+Maj+Fin	Étendre la sélection jusqu'à la fin du document
Alt+Fin	Déplacer le curseur à la fin absolue de la ligne
Alt+Maj+Fin	Étendre la sélection rectangulaire jusqu'à la fin de la ligne
Page précédente	Déplacer le curseur vers le haut d'une page
Maj+Page précédente	Étendre la sélection sur une page
Alt+Maj+Page précédente	Étendre la sélection rectangulaire vers le haut d'une page
Bas de page	Déplacer le curseur vers le bas d'une page
Maj+Page suivante	Étendre la sélection vers le bas d'une page
Alt+Maj+Page suivante	Étendre la sélection rectangulaire vers le bas d'une page
Supprimer	Supprimer le caractère à droite du curseur
Maj+Supprimer	Sélection de coupe

Ctrl+Supprimer	Supprimer le mot à droite du curseur
Ctrl+Maj+Supprimer	Supprimer jusqu'à la fin de la ligne
Insérer	Activer/désactiver la refrappe
Maj+Inser	Pâte
Ctrl+Inser	Copier la sélection
Retour arrière	Supprimer le caractère à gauche du curseur
Maj+Retour arrière	Supprimer le caractère à gauche du curseur
Ctrl+Retour arrière	Supprimer le mot à gauche du curseur
Ctrl+Maj+Retour arrière	Supprimer du début de la ligne jusqu'au curseur
Alt+Retour arrière	Supprimer Annuler
Languette	Indenter le curseur d'un onglet
Ctrl+Maj+I	Indenter le curseur d'un onglet
Maj+Tabulation	Supprimer le retrait du curseur d'un onglet
Ctrl+clavier(+)	Agrandir
Ctrl+clavier(-)	Dézoomer
Ctrl+clavier(/)	Restaurer le zoom
Ctrl+Z	Annuler
Ctrl+Y	Refaire
Ctrl+X	Sélection de coupe
Ctrl+C	Copier la sélection
Ctrl+V	Pâte
Ctrl+L	Ligne de coupe
Ctrl+T	Transposer la ligne
Ctrl+Maj+T	Copier la ligne
Ctrl+A	Sélectionner tout le document
Ctrl+D	Sélection en double

Ctrl+U	Convertir la sélection en minuscules
Ctrl+Maj+U	Convertir la sélection en majuscule
Ctrl+E	Déplacer le curseur sur l'accolade correspondante
Ctrl+Maj+E	Étendre la sélection à l'accolade correspondante
Ctrl+Maj+C	Basculer le commentaire de ligne sur la sélection
Ctrl+Maj+X	Activer/désactiver le commentaire du flux sur la sélection.
Ctrl+F2	Activer/désactiver le signet
F2	Aller au signet suivant
Maj+F2	Aller au favori précédent
Ctrl+Maj+F2	Effacer tous les signets du fichier actuel
Ctrl+Maj+W	Basculer les caractères d'espacement
Ctrl+Maj+L	Basculer les caractères EOL
Ctrl+Espace	Invoquez la saisie semi-automatique.
Ctrl+-	Revenir en arrière dans l'historique du curseur
Ctrl+Maj+-	Avancer dans l'historique du curseur
F12	Démarrer /Annuler la recherche de mot-clé dans le(s) fichier(s).
Ctrl+F	Rechercher du texte
Ctrl+R	Remplacer le texte

Notes

- En plus de ces touches, vous pouvez attribuer des combinaisons de touches (Ctrl+Alt+<n>) aux macros que vous définissez dans le Source Éditeur de Code

Motifs d'application (Modèle + Code)

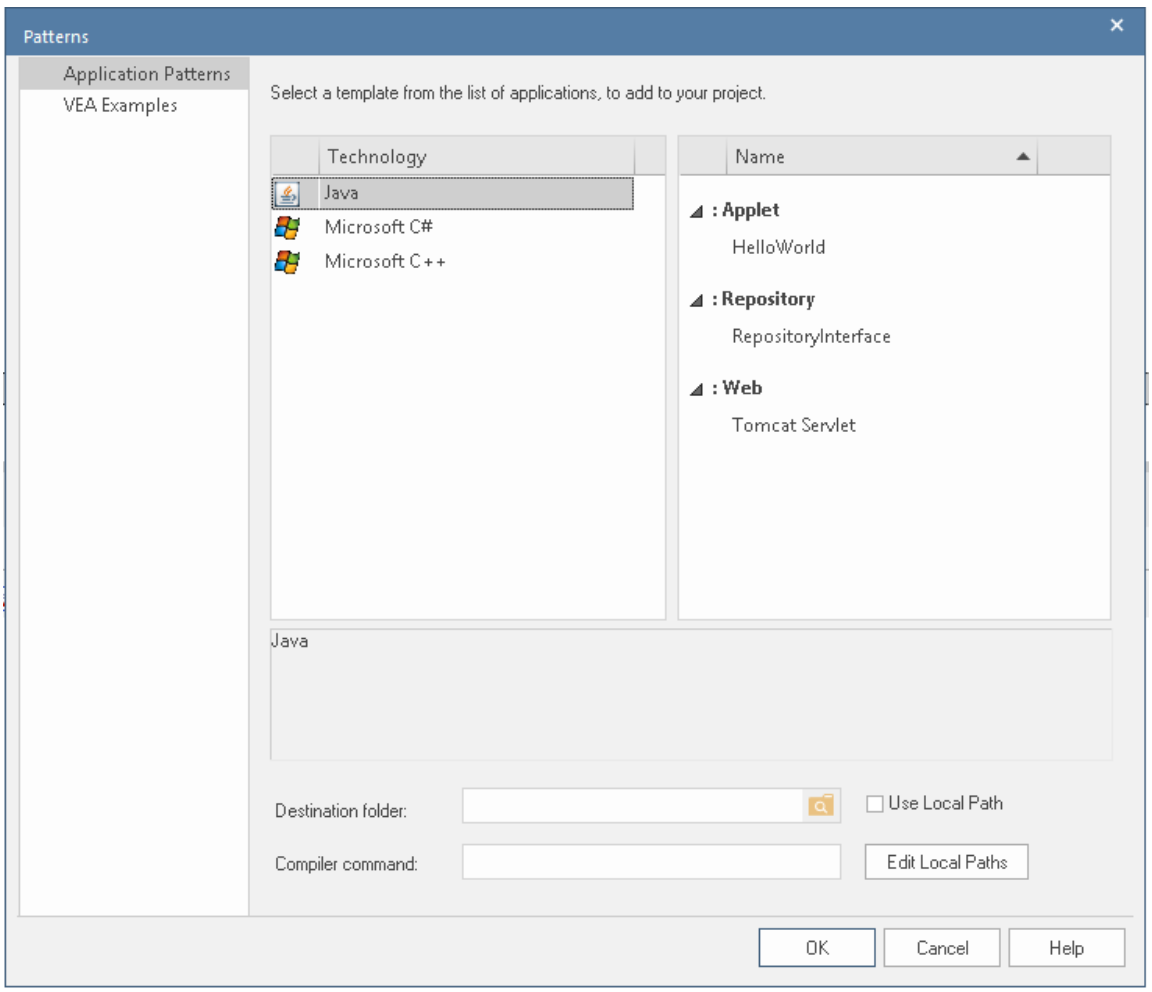
Pour démarrer le plus rapidement possible un projet basé sur du code, Enterprise Architect vous aide à générer des projets de démarrage comprenant des informations sur le modèle, du code et des scripts de construction pour l'un des nombreux types d'applications de base. Motifs incluent :

- Applications Windows MFC
- Programmes Java
- Services Web ASP.NET

Accéder

Ruban	Développer > Code source > Créer à partir de Motif > Motifs d'application

Générer des modèles



Option	Action

Technologie	Sélectionnez la technologie appropriée.
Nom	Affiche les Motifs d'application disponibles pour la technologie sélectionnée ; sélectionnez le Motif requis à importer.
<description>	Affiche une description du Motif sélectionné.
Dossier de destination	Recherchez et sélectionnez le répertoire dans lequel charger le code source de l'application.
Utiliser le chemin local	Activer la sélection d'un chemin local existant sous lequel placer le code source ; modifie le champ « Dossier de destination » en une sélection déroulante.
Commande Compilateur	Affiche le chemin de commande du compilateur par défaut pour la technologie sélectionnée ; vous devez soit : • Confirmez que le compilateur peut être trouvé à ce chemin, ou • Modifier le chemin d'accès à l'emplacement du compilateur
Modifier les chemins locaux	De nombreux Motifs d'application spécifient leur compilateur à l'aide d'un chemin local. La première fois que vous utilisez un Motif vous devez cliquer sur ce bouton pour vous assurer que le chemin local pointe vers le bon emplacement. La dialogue « Chemins locaux » s'affiche.

Notes

- Si nécessaire, vous pouvez publier Motifs d'application personnalisés en ajoutant des fichiers au répertoire *AppPatterns* où Enterprise Architect est installé ; les répertoires de niveau supérieur sont répertoriés comme Technologies et peuvent contenir un fichier d'icône pour personnaliser l'icône affichée pour la technologie
  Les répertoires situés en dessous sont définis comme des groupes dans la liste Motifs ; les Motifs sont identifiés par la présence de quatre fichiers portant un nom correspondant : un fichier zip (.zip), un fichier XMI (.xml), un fichier de configuration (.cfg) et une icône facultative (.ico)
- Le fichier de configuration supporte ces champs :
  - [fournisseur], [langue], [plateforme], [url], [description], [version] - tous affichés dans <description> champ
  - [xmirootpaths] - le chemin racine du code source dans le XMI exporté ; ceci est remplacé par le dossier de destination sélectionné lorsque l'utilisateur applique le Motif d'application

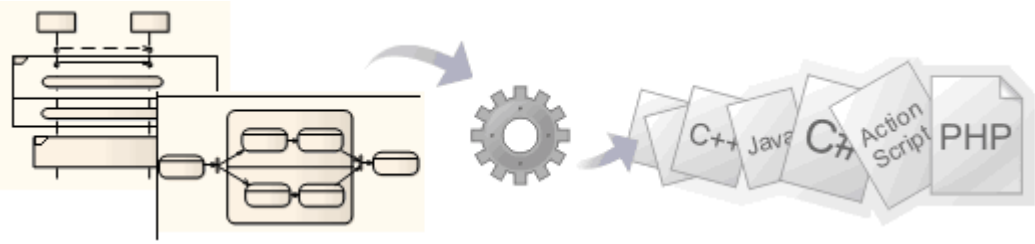
Intégration OMD et ingénierie de code

MDG Integration for Eclipse et MDG Integration for Visual Studio sont des produits qui vous aident à créer et à maintenir vos modèles UML directement dans ces deux environnements de développement intégrés populaires, à l'aide de la fenêtre Enterprise Architect Navigateur . Des modèles peuvent être générés vers le code source à l'aide du moteur gabarit riche et flexible qui donne à l'ingénieur un contrôle total sur la façon dont le code est généré. Le code source existant peut également faire l'objet d'une ingénierie inverse et être synchronisé avec les modèles UML . Une fois l'intégration installée, l'EDI deviendra une plate-forme modélisation riche en fonctionnalités, économisant du temps et des efforts et réduisant le risque d'erreur en reliant la gestion des exigences, l'architecture et la conception à l'ingénierie du code source.

Une documentation riche et expressive peut être générée automatiquement dans une large gamme de formats, notamment DOCX, PDF et HTML. La documentation peut inclure diagrammes des exigences, de la conception et de l'architecture ainsi que des descriptions du code source, mettant le code source en contexte.

Vous pouvez acheter MDG Integration for Eclipse ™ et MDG Integration for Visual Studio ™ ou télécharger les éditions d'essai sur le site Web Sparx Systems .

Génération de code Modèle Comportementale



Les capacités d'ingénierie système polyvalentes d'Enterprise Architect peuvent être utilisées pour générer du code pour les langages de description de logiciels, de systèmes et de matériel directement à partir de modèles comportementaux, tels que Statemachine , Séquence (Interaction) et diagrammes d'activité. Les langages pris en charge incluent C(OO), C++, C# , Java, VB.Net, VHDL, Verilog et SystemC.

Le code logiciel peut être généré à partir de diagrammes Statemachine , Séquence et Activité, et des langages de description de matériel à partir de diagrammes Statemachine (en utilisant les gabarits Legacy Statemachine).

Accéder

Ruban	Développer > Code Source > Générer

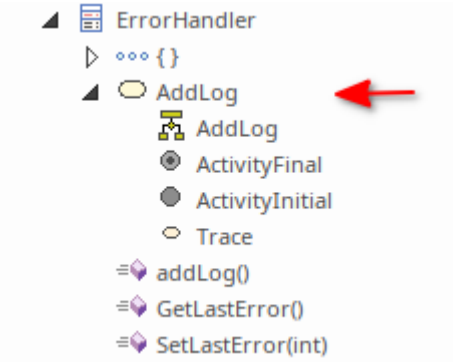
Spécificités Modèle Comportementale

La génération de code de modèle Comportementale est prise en charge pour les trois types clés de modèle comportemental ; cependant, chaque type de modèle comportemental a ses propres caractéristiques basées sur le type d'élément impliqué. Ces rubriques fournissent guidage et des références pour les types d'éléments de base utilisés.

Type	Description
Activité	Un aperçu des principaux types d'actions et des détails sur leur utilisation dans la génération de code.
Interaction	Détails couvrant l'utilisation de messages et de fragments pour la génération de code de diagrammes d'interaction (Séquence).
Statemachines	Détails couvrant les options de définition du code à générer à l'aide States , y compris les comportements - Entry/Exit/Do et Transitions in a Statemachine .

Structure

La génération de code de modèle Comportementale nécessite principalement que toutes les constructions comportementales soient contenues dans une classe (en tant qu'enfant de cette classe).



Si des constructions comportementales font référence à des éléments externes en dehors du Paquetage actuel, vous devez ajouter un connecteur Import du Paquetage actuel au Paquetage contenant les éléments externes. Pour plus de détails, consultez le type de connecteur *Importer* dans la rubrique d'aide *Paquetage Diagramme* .

Générer du code à partir de diagrammes comportementaux à l'aide du projet EAExample

Étape	Action
1	Ouvrez le fichier EAExample.eap en sélectionnant l'option de ruban ' Démarrer > Aide > Aide > Ouvrir l'exemple Modèle '.
2	Depuis la fenêtre Navigateur , sélectionnez l'un de ces Paquetages : **Exemples de langage logiciel :** • Exemple Modèle > Ingénierie de Logiciel > Modèle Java avec comportements Générer les classes Compte et Commande • Exemple Modèle > Ingénierie des Systèmes > Modèle d'implémentation > Logiciel > C# Générer la classe DataProcessor • Exemple Modèle > Ingénierie des Systèmes > Exemple SysML > Modèle d'implémentation > Logiciel > C++ Générer la classe IO • Exemple Modèle > Ingénierie des Systèmes > Exemple SysML > Modèle d'implémentation > Logiciel > Java Générer la classe IO • Exemple Modèle > Ingénierie des Systèmes > Exemple SysML > Modèle d'implémentation > Logiciel > VBNet Générer la classe IO **Exemples de langage matériel :** • Exemple Modèle > Ingénierie des Systèmes > Exemple SysML : Lecteur Audio Portable > Modèle d'implémentation > Matériel > SystemC Générer la classe PlayBack • Exemple Modèle > Ingénierie des Systèmes > Exemple SysML : Lecteur Audio Portable > Modèle d'implémentation > Matériel > VHDL Générer la classe PlayBack • Exemple Modèle > Ingénierie des Systèmes > Exemple SysML : Lecteur Audio Portable > Modèle d'implémentation > Matériel > Verilog Générer la classe PlayBack

3	Quand fini: • Sélectionnez la classe qui a été utilisée pour la génération • Appuyez sur Ctrl+E pour ouvrir le code source généré. Vous devriez voir les méthodes générées dans le code.

Notes

- La génération de code logiciel à partir de modèles comportementaux est disponible dans les éditions Unified et Ultimate d' Enterprise Architect
- La génération de code matériel à partir des modèles Statemachine est disponible dans les éditions Unified et Ultimate d' Enterprise Architect
- Pour C(OO), sur la page 'C Spécifications' de la dialogue 'Gérer les options Modèle ', définissez l'option ' Object Oriented Support ' sur True.
  Voir la rubrique d'aide *Options C - Modèle* .
- La synchronisation du code n'est pas prise en charge pour le code comportemental.

Génération de code - Diagrammes d'activités

La génération de code à partir diagrammes d'activité dans une classe nécessite une phase de validation, au cours de laquelle Enterprise Architect utilise l'optimiseur de graphique d'ingénierie système pour analyser le diagramme et le restituer en diverses constructions à partir desquelles le code peut être généré. Enterprise Architect transforme également les constructions en l'un des différents types d'action (le cas échéant), similaires aux constructions du diagramme d'interaction.

Actions

Action	Description
Actions d'appel (actions d'appel)	Utilisé pour invoquer des opérations ou des comportements dans un diagramme d'activité ; les deux principales variantes d'actions d'appel prises en charge dans la génération de code comportemental sont : • Action CallOperation - utilisée pour invoquer des opérations, qui peuvent se trouver dans la même classe ou dans d'autres classes du même Paquetage ; si vous faites référence à des opérations provenant d'autres classes au sein du même Paquetage , vous devez avoir une cible à laquelle la demande est transmise • CallBehavior Action - utilisé pour appeler une autre activité dans un flux d'activité ; l'activité référencée devrait être dans la même classe **Arguments** Les actions d'appel peuvent spécifier des valeurs d'argument correspondant aux paramètres du comportement ou fonctionnalité comportementale associée. Vous pouvez ajouter les arguments manuellement ou les créer automatiquement à l'aide du bouton Synchroniser de la dialogue 'Arguments'.
CréerObjectAction	Utilisé pour désigner une création object dans le flux d'activité ; vous pouvez définir le résultat Pin de CreateObjectAction comme object à créer, à l'aide de la fenêtre Propriétés de l'élément Action . Le classificateur de CreateObjectAction signifie le classificateur pour lequel une instance doit être créée.
DétruireObjectAction	Utilisé pour désigner une suppression object dans le flux d'activité ; vous pouvez définir le Pin cible de DestroyObjectAction comme object à détruire, à l'aide de la fenêtre Propriétés de l'élément Action .
Boucles	L'optimiseur de graphiques d'ingénierie système d' Enterprise Architect est également capable d'analyser et d'identifier les boucles ; une boucle identifiée est rendue en interne sous la forme d'une boucle Action , qui est traduite par les macros de génération de code EASL pour générer le code requis. Vous pouvez avoir une seule boucle, des boucles imbriquées et plusieurs niveaux de boucles imbriquées.
Expressions conditionnelles	Pour modéliser une instruction conditionnelle, vous utilisez les nœuds Décision /Merge. Alternativement, vous pouvez impliquer des décisions/fusions en interne ; l'optimiseur de graphique attend un nœud de fusion associé pour chaque nœud Décision , afin de faciliter le suivi efficace des différentes branches et l'analyse des constructions de code qu'elles contiennent.

Notes

- Pour pouvoir générer du code à partir de modèles comportementaux, toutes les constructions comportementales doivent être contenues dans une classe

Génération de code - Diagrammes d'interaction

Lors de la génération de code à partir de diagrammes d'interaction (Séquence) dans une classe, Enterprise Architect applique son optimiseur de graphes d'ingénierie système pour transformer les constructions de classe en paradigmes programmatiques. Les messages et les fragments sont identifiés comme deux des nombreux types d'action en fonction de leur fonctionnalité, et Enterprise Architect utilise les gabarits de génération de code pour afficher leur comportement en conséquence.

Actions

Action	Description
Appel Action	Un message qui appelle une opération.
Action Créer	Un message avec Lifecycle = Nouveau.
Action Détruire	Un message avec Lifecycle = Supprimer.
Boucle Action	Un fragment combiné avec Type = Alt.
Action si	Un fragment combiné avec Type = boucle.
Affecter à	Un message d'appel avec un attribut cible valide défini à l'aide du champ « Attribuer à » est rendu dans le code en tant qu'attribut cible d'une Action d'appel.

Notes

- Pour pouvoir générer du code à partir de modèles comportementaux, toutes les constructions comportementales doivent être contenues dans une classe
- Pour un diagramme d'interaction (Séquence), le moteur de génération de code comportemental s'attend à ce que le diagramme Séquence et tous ses messages et fragments d'interaction associés soient encapsulés dans un élément d'interaction.

Génération de code - Statemachines

Une Statemachine illustre comment un object (représenté par une Classe) peut changer d'état, chaque changement d'état étant une transition initiée par un déclencheur survenant lors d'un événement, souvent sous des conditions ou contraintes définies comme des gardes. Lorsque vous modélisez la façon dont l' object change d'état, vous pouvez générer et construire (compiler) du code à partir de celui-ci dans le langage logiciel approprié et exécuter le code, en visualisant l'exécution via le Modèle Simulator.

Il est également possible, dans Enterprise Architect , de combiner les Statemachines d'objets distincts mais liés pour voir comment ils interagissent (via Broadcast Événements), et de créer et générer rapidement du code à partir de variantes du modèle. Par exemple, vous pouvez modéliser le comportement de :

- La roue arrière hors côté d'un véhicule en modes propulsion arrière et traction avant (une Statemachine)
- Le volant et les quatre roues motrices d'un véhicule en mode 4 roues motrices (cinq Statemachines)
- Les roues d'un véhicule tout-terrain et d'une voiture de sport (deux artefacts, instances d'une combinaison de Statemachines)

L'élément Statemachine Exécutable Artifact est d'une importance cruciale dans la génération et le test du code pour toutes ces options. Cela fait office d'unité de génération de conteneur et de code pour vos modèles Statemachine .

Vous n'utilisez pas cette méthode pour générer du code pour les langages de définition matérielle, mais vous pouvez également générer à la fois du code HDL et du code logiciel à partir de Statemachines à l'aide des facilités génériques de génération de code dans Enterprise Architect (voir les procédures *Générer Code Source*).

Conditions préalables

- Sélectionnez 'Paramètres > Modèle > Options > Ingénierie du code source' et, pour le langage de codage logiciel approprié (Java, C, C# ou ANSI C++), définissez l'option 'Utiliser le nouveau Statemachine Gabarit ' sur 'True'.
- Si vous travaillez en C++, sélectionnez 'Paramètres > Modèle > Options > Ingénierie du code source > C++' et définissez l'option 'Version C++' sur 'ANSI'.

Cette méthode de génération de code ne s'applique pas aux gabarits de génération de code Legacy Statemachine développés avant Enterprise Architect Release 11.0, ni à la génération de code Hardware Definition Language.

Accéder

Faites glisser un artefact Statemachine Exécutable depuis la page ' Simulation ' de la boîte à outils Diagramme , sur votre diagramme . La page « Simulation » de la boîte à outils Diagramme est accessible en utilisant l'une des méthodes décrites dans ce tableau .

Ruban	Conception > Diagramme > Boîte à outils > Simulation
Raccourcis Clavier	Ctrl+Maj+3 > Simulation
Autre	Vous pouvez afficher ou masquer la boîte à outils Diagramme en cliquant sur les icônes » ou « à l'extrémité gauche de la barre de légende en haut de Diagramme Vue .

Préparez votre ou vos diagramme Statemachine

Étape	Action
1	Pour chaque Statemachine que vous souhaitez modéliser, créez un diagramme de classes.
2	Depuis la page « Classe » de la boîte à outils Diagramme , faites glisser l'icône « Classe » sur votre diagramme et donnez à l'élément un nom approprié.
3	Cliquez-droit sur l'élément Classe et sélectionnez le 'Nouveau Diagramme Enfant | Option de menu contextuel de Statemachine . Donnez au diagramme Statemachine un nom approprié.
4	Créez le modèle Statemachine pour refléter les transitions appropriées entre States .

Configurer l'artefact Statemachine Exécutable

Étape	Action
1	Créez un nouveau diagramme de classes pour contenir la ou les Statemachine modélisées à partir desquelles vous avez l'intention de générer du code.
2	Depuis la page ' Simulation ' de la Diagramme Toolbox, faites glisser l'icône ' Statemachine Exécutable ' sur le diagramme pour créer l'élément Artifact. Nommez l'élément et faites glisser ses bordures pour l'agrandir.
3	Depuis la fenêtre Navigateur , faites glisser le (premier) élément Class contenant un diagramme Statemachine sur l'élément Artifact du diagramme . La dialogue « Coller <nom de l'élément> » s'affiche. Dans le champ 'Déposer sous', cliquez sur la flèche déroulante et sélectionnez la valeur 'Propriété'. (Si le dialogue ne s'affiche pas, appuyez sur Ctrl tout en faisant glisser l'élément Classe depuis la fenêtre du Navigateur .)
4	Cliquez sur le bouton OK . L'élément Class est collé à l'intérieur de l'artefact en tant que pièce.
5	Répétez les étapes 3 et 4 pour toutes les autres classes avec Statemachines pour lesquelles vous souhaitez combiner et générer du code. Ceux-ci pourraient être : • Répéter les 'drops' de la même classe et de la même Statemachine , modélisation d'objets parallèles • Différentes classes et Statemachines , modélisation d'objets en interaction séparés
6	Cliquez-droit sur l'élément Artifact et sélectionnez l'option ' Propriétés > Propriétés ', développez la catégorie 'Avancé' et, dans le champ 'Langue', cliquez sur la flèche déroulante et définissez la langue du code sur la même langue que celle actuelle. défini pour les éléments Class. Vous pouvez maintenant faire glisser cet élément Statemachine Exécutable Artefact de la fenêtre Navigateur vers le diagramme autant de fois que vous le souhaitez, et modifier les parties pour modéliser les variations du système ou du processus, ou du même système ou processus avec différents langages de programmation.

Générer du code à partir d'un artefact

Étape	Action
1	Cliquez sur l'élément Statemachine Exécutable Artifact et sélectionnez l'option de ruban 'Simulate > États Exécutables > Statemachine > Générer '. La dialogue « Génération de code Statemachine exécutable » s'affiche.
2	Dans le champ « Répertoire de sortie du projet », saisissez ou recherchez le chemin du répertoire sous lequel créer les fichiers de sortie. Lors de la génération de code, tous les fichiers existants dans ce répertoire sont supprimés.
3	Sélectionnez le système cible. Si vous utilisez Windows sélectionnez l'option « Local ». Si vous travaillez sous Linux, choisissez l'option « À distance ». Le choix affecte les scripts générés pour support la Simulation .
4	Dans le champ « Emplacement du répertoire d'installation <compilateur> », saisissez ou recherchez le chemin du répertoire d'installation du compilateur, qui sera automatiquement mappé au chemin local (affiché à gauche du champ). Pour chaque langage de programmation, les chemins peuvent ressembler à ces exemples : • Java JAVA_HOME C:\Program Files (x86)\Java\jdk1.7.0_17 • C/C++ VC_HOME C:\Program Files (x86)\Microsoft Visual Studio 9.0 • C# CS_HOME C:\ Windows \Microsoft.NET\Framework\V3.5
5	Cliquez sur le bouton Générer . Les fichiers de code sont créés en fonction du langage de programmation. La fenêtre Sortie système s'affiche avec un onglet « Sortie Statemachine Exécutable », affichant la progression et l'état de la génération. Lors de la génération de code, une fonction de validation automatique est exécutée pour vérifier les erreurs diagramme ou de modèle par rapport aux contraintes UML . Toutes les erreurs sont identifiées par des messages d'erreur dans l'onglet ' Statemachine Exécutable Output'. Double-cliquez sur un message d'erreur pour afficher la structure modélisation dans laquelle l'erreur se produit, et corrigez l'erreur avant de regénérer le code.
6	Lorsque le code est généré sans erreur, cliquez sur l'élément Artifact et sélectionnez l'option du ruban 'Simulate > États Exécutables > Statemachine > Build' pour compiler le code. La fenêtre Sortie système s'affiche avec un onglet « Construire », affichant la progression et l'état de la compilation. Notez que la compilation inclut la configuration de l'opération de simulation.

Macros de génération de code

Vous pouvez également utiliser deux macros dans la génération de code pour Statemachines .

Nom de la macro	Description
SEND_EVENT	Envoyez un événement à un récepteur (la partie). Par exemple: %SEND_EVENT("événement1", "Partie1")%

BROADCAST_EVENT	Diffusez un événement à tous les récepteurs. Par exemple: %BROADCAST_EVENT("événement2")%

Exécuter/simuler le code à partir d'un artefact

Étape	Action
1	Sélectionnez l'option du ruban « Simuler > Simulation Dynamique > Simulateur > Appliquer l'espace de travail » pour afficher ensemble la fenêtre Simulation et la fenêtre Simulation Événements . Ancrez les deux fenêtres dans une zone pratique de l'écran.
2	Sur le diagramme ou la fenêtre Navigateur , cliquez sur l'élément Artefact et sélectionnez l'option de ruban 'Simulate > États Exécutables > Statemachine > Exécuter '. Le premier diagramme Statemachine de la série s'affiche avec la simulation du processus déjà commencé. Dans la fenêtre Simulation , les étapes du traitement sont indiquées sous ce format : [03516677] Part1[Class1].Initial_367_TO_State4_142 Effet [03516683] Partie 1[Class1].StateMachine_State4 ENTRÉE [03516684] Partie 1[Class1].StateMachine_State4 DO [03518375] Bloqué
3	Cliquez sur les boutons appropriés de la barre d'outils de la fenêtre Simulation pour parcourir la simulation comme vous le souhaitez. Lorsque la simulation se termine au niveau de l'élément Quitter ou Terminer, cliquez sur le bouton Arrêter dans la barre d'outils de la fenêtre Simulation .
4	Lorsque la trace indique Bloqué, la simulation a atteint un point où un événement Déclencheur doit se produire avant que le traitement puisse continuer. Sur la fenêtre Simulation Événements , dans la colonne ' Déclencheurs en Attente ', double-cliquez sur le Déclencheur approprié. Lorsque le Déclencheur est déclenché, la simulation continue jusqu'au point de pause suivant, Déclencheur ou sortie.

Notes

- Si vous apportez de petites modifications à un modèle Statemachine existant, vous pouvez combiner les opérations de génération, de build et exécuter de code en sélectionnant l'option du ruban 'Simulate > États Exécutables > Statemachine > Générer , build and exécuter '.
- Vous pouvez également générer du code en JavaScript

Gabarits Statemachine héritées

La génération de code fonctionne à l'aide d'un ensemble de gabarits de génération. À partir de la version 11.0 d' Enterprise Architect , un ensemble différent de gabarits est disponible par défaut pour la génération de code logiciel à partir d'un diagramme Statemachine vers du code Java, C, ANSI C++ ou C# . Vous pouvez toujours utiliser les gabarits d'origine, comme décrit ici, pour les modèles développés dans les versions antérieures d' Enterprise Architect , si vous ne souhaitez pas les mettre à niveau pour les nouveaux gabarit facilités .

Basculer entre gabarits Legacy et Release 11

Accéder

Affichez la dialogue 'Gérer les options Modèle ', puis affichez la page 'Spécifications de langue' pour la langue de votre choix, en utilisant l'une des méthodes décrites dans ce tableau . Si nécessaire, développez le groupe ' Statemachine Engineering (pour le modèle actuel)' et définissez l'option 'Utiliser le nouveau Statemachine Gabarit ' sur True (pour utiliser les gabarits ultérieurs) ou False (pour utiliser les gabarits hérités).

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > [nom de la langue]

Transformations héritées Gabarit

Une Statemachine dans une classe génère en interne un certain nombre de constructions dans des langages logiciels pour assurer une exécution efficace des comportements des States (action, entrée et sortie) et également pour coder l'effet de transition approprié si nécessaire.

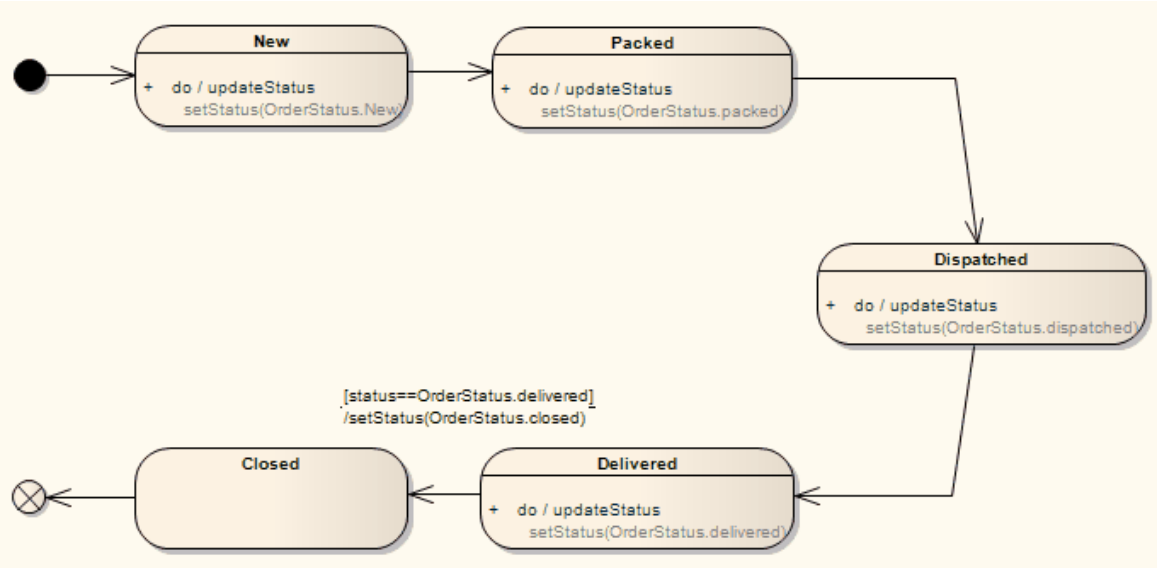
Objets Modèle	Objets de code
Énumérations	• StateType - consiste en une énumération pour chacun des States contenus dans la Statemachine • TransitionType – consiste en une énumération pour chaque transition à laquelle est associé un effet valide ; par exemple, ProcessOrder_Delivered_to_ProcessOrder_Closed • CommandType – consiste en une énumération pour chacun des types de comportement qu'un State peut contenir (Do, Entry, Exit)
Attributes	• currState:StateType - une variable pour contenir les informations de l' State actuel • nextState:StateType - une variable pour contenir les informations du prochain State , définie par les transitions de chaque State en conséquence • currTransition:TransitionType - une variable pour contenir les informations de transition actuelles ; ceci est défini si la transition est associée à un effet valide • transcend:Boolean - un indicateur utilisé pour indiquer si une transition est impliquée dans la transcendance entre différentes Statemachines (ou états de sous-machine) • xx_history:StateType - une variable d'historique pour chaque Statemachine /Submachine State , pour contenir des informations sur le dernier State à partir

	duquel la transition a eu lieu
Opérations	• StatesProc - une procédure States , contenant une carte entre le dénombrement d'un State et son fonctionnement ; il déréférence les informations de l' State actuel pour invoquer la fonction de State respectif • TransitionsProc - une procédure Transitions, contenant une carte entre l'énumération de la transition et son effet ; il invoque l'effet de Transition • <<State>> - une opération pour chacun des States contenus dans la Statemachine ; cela restitue les comportements d'un State en fonction de l'entrée CommandType et exécute également ses transitions • initializeStateMachine - une fonction qui initialise tous les attributs liés au framework • runStateMachine - une fonction qui parcourt chaque State et exécute leurs comportements et transitions en conséquence

Notes

- Pour pouvoir générer du code à partir de modèles comportementaux, toutes les constructions comportementales doivent être contenues dans une classe

Code Java généré à partir de l'ancienne Statemachine Gabarit



Énumération privée StateType : int

{

ProcessOrder_Delivered,

ProcessOrder_Packed,

ProcessOrder_Closed,

ProcessOrder_Dispatched,

ProcessOrder_Nouveau,

ST_NOSTATE

}

Énumération privée TransitionType : int

{

ProcessOrder_Delivered_to_ProcessOrder_Closed,

TT_NOTRANSITION

}

Type de commande d'énumération privée

{

Faire,

Entrée,

Sortie

}

privé StateType currState ;

StateType privé nextState ;

privé TransitionType currTransition ;

transcendance booléenne privée ;

privé StateType ProcessOrder_history ;

```
private void processOrder_Delivered (commande CommandType)
{
interrupteur (commande)
{
cas Faire :
{
// Faire des comportements..
setStatus (Livré);
// Transitions de State
if((statut==Livré))
{
nextState = StateType.ProcessOrder_Closed ;
currTransition = TransitionType.ProcessOrder_Delivered_to_ProcessOrder_Closed ;
}
casser;
}
défaut:
{
casser;
}
}
}
private void processOrder_Packed (commande CommandType)
{
interrupteur (commande)
{
cas Faire :
{
// Faire des comportements..
setStatus (Emballé);
// Transitions de State
nextState = StateType.ProcessOrder_Dispatched ;
casser;
}
défaut:
{
casser;
}
}
}
private void processOrder_Closed (commande CommandType)
{
```

```
interrupteur (commande)
{
cas Faire :
{
// Faire des comportements..
// Transitions de State
casser;
}
défaut:
{
casser;
}
}
}
private void processOrder_Dispatched (commande CommandType)
{
interrupteur (commande)
{
cas Faire :
{
// Faire des comportements..
setStatus(Expédié);
// Transitions de State
nextState = StateType.ProcessOrder_Delivered ;
casser;
}
défaut:
{
casser;
}
}
}
private void processOrder_New (commande CommandType)
{
interrupteur (commande)
{
cas Faire :
{
// Faire des comportements..
setStatus(nouveau);
// Transitions de State
nextState = StateType.ProcessOrder_Packed ;
```

```
casser;
}
défaut:
{
casser;
}
}
}
private void StatesProc (StateType currState, commande CommandType)
{
commutateur (état actuel)
{
cas ProcessOrder_Delivered :
{
processOrder_Delivered(commande);
casser;
}
cas ProcessOrder_Packed :
{
processOrder_Packed(commande);
casser;
}
cas ProcessOrder_Closed :
{
processOrder_Closed(commande);
casser;
}
cas ProcessOrder_Dispatched :
{
processOrder_Dispatched(commande);
casser;
}
cas ProcessOrder_New :
{
processOrder_New(commande);
casser;
}
défaut:
casser;
}
}
vide privé TransitionsProc (transition TransitionType)
```

```
{
interrupteur (transition)
{
cas ProcessOrder_Delivered_to_ProcessOrder_Closed :
{
setStatus (fermé);
casser;
}
défaut:
casser;
}
}
privé vide initalizeStateMachine()
{
currState = StateType.ProcessOrder_New ;
nextState = StateType.ST_NOSTATE ;
currTransition = TransitionType.TT_NOTRANSITION;
}
runStateMachine vide privé()
{
tandis que (vrai)
{
si (currState == StateType.ST_NOSTATE)
{
casser;
}
currTransition = TransitionType.TT_NOTRANSITION ;
StatesProc(currState, CommandType.Do);
// puis vérifie s'il y a une transition valide assignée après le comportement do
si (nextState == StateType.ST_NOSTATE)
{
casser;
}
si (currTransition != TransitionType.TT_NOTRANSITION)
{
TransitionsProc(currTransition);
}
si (étatcurr != état suivant)
{
StatesProc(currState, CommandType.Exit);
StatesProc(nextState, CommandType.Entry);
currState = état suivant ;
```

```
}
}
}
```

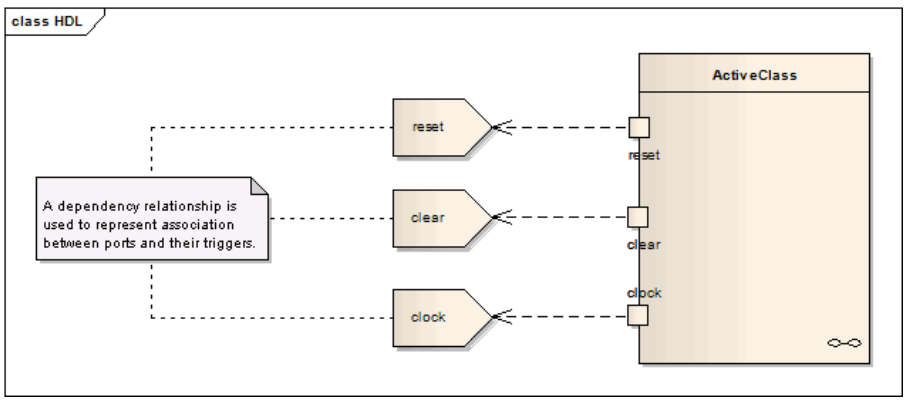
Modélisation Statemachine pour les HDL

Pour générer efficacement du code HDL (Hardware Description Language) à partir de modèles Statemachine , appliquez les pratiques de conception décrites dans cette rubrique. Les langages de description du matériel incluent VHDL, Verilog et SystemC.

Dans un modèle HDL Statemachine , vous pouvez vous attendre à :

- Désigner Déclencheurs de Conduite
- Établir une cartographie port-déclencheur
- Ajouter à la logique State Actif

Opérations

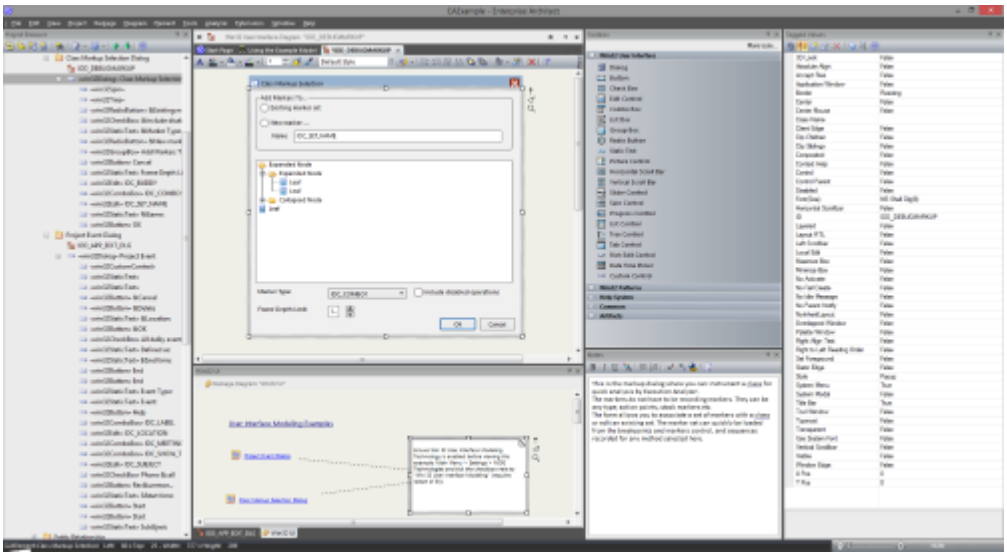
Opération	Description
Désigner Déclencheurs de Conduite	• Un Déclencheur 'Change' est considéré comme un Trigger asynchrone si : - Il y a une transition depuis l' State actuel de la sous-machine (qui encapsule la logique réelle) qu'il déclencheurs , et - L' State cible de cette transition a une auto-transition déclenchée par le même Déclencheur • Déclencheurs Asynchrones doivent être modélisés selon ce motif : - Le Déclencheur doit être de type Change (spécification : Vrai / Faux) - L' State actif (SubMachine State) doit avoir une transition déclenché par cela - L' State cible de la transition déclenchée doit disposer d'un transition avec le même Déclencheur • Un Déclencheur de type 'Time', qui déclencheurs les transitions vers l'état actif (SubMachine State), est réputé être l'Horloge ; la spécification de ce déclencheur doit être conforme à la langue cible : - VHDL - bord_montant / bord_descendant - Verilog - posedge / negedge - SystemC - positif / négatif
Établir une cartographie des ports-déclencheurs	Après modélisation avec succès les différents modes de fonctionnement du composant, et les Déclencheurs qui leur sont associés, vous devez associer les Déclencheurs aux Ports du composant. Une relation de dépendance du port au Déclencheur associé est utilisée pour signifier cette association. 
Logique State Actif	La désignation du Déclencheur moteur et l'établissement de la cartographie

	Port-Trigger mettent en place les préliminaires nécessaires à l'interprétation efficace des composants matériels. Nous modélisons maintenant la logique Statemachine réelle au sein de l' Actif (SubMachine) State .

Notes

- Pour pouvoir générer du code à partir de modèles comportementaux, toutes les constructions comportementales doivent être contenues dans une classe
- Le moteur de génération de code actuel supporte qu'une seule horloge Déclencheur pour un composant

Boîtes de dialogue Interface Utilisateur Win32



À l'aide de la technologie MDG Win32 UI , vous pouvez concevoir des écrans d'interface utilisateur qui s'affichent sous forme de contrôles Win32®. L'interface utilisateur produite peut être utilisée dans n'importe quel script de définition de ressource. Les scripts de définition de ressources, ou fichiers RC, sont une technologie Microsoft qui, comme pour les autres codes, peut être compilée et les ressources utilisées par les applications de bureau natives. Les écrans ou boîtes de dialogue de l'interface utilisateur peuvent être créés à partir de zéro ou par ingénierie inverse. Les modèles d'interface utilisateur peuvent également être évolués à l'aide de la fonction de code de synchronisation (F7). modélisation d'interface s'effectue sur diagrammes exactement de la même manière que vous travailleriez avec n'importe quelle technologie dans Enterprise Architect . Un aspect intéressant de la conception Interface Utilisateur dans Enterprise Architect est que les composants peuvent jouer un rôle actif dans la simulation des Statemachines et des activités, permettant à une simulation d'interagir avec les utilisateurs, un peu comme un vrai programme !

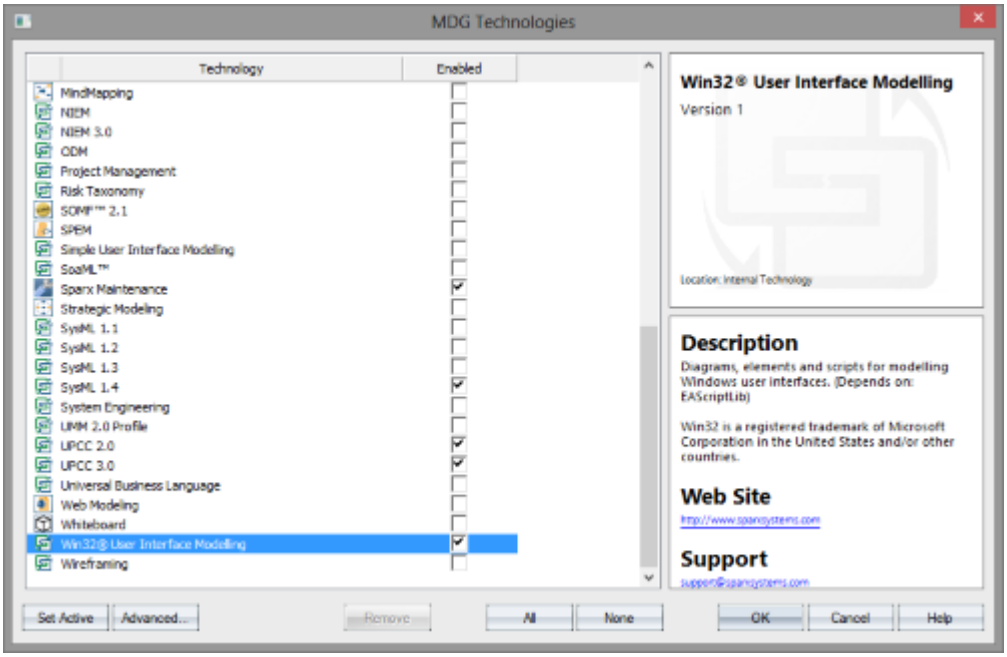
Accéder

Ruban	Conception > Diagramme > Ajouter Diagramme > Type > Interface Utilisateur Win32
Menu Contexte	Cliquez-droit sur Paquetage | Ajouter Diagramme > Type | Interface Utilisateur Win32
Autre	Menu de la barre de légende de la fenêtre Navigateur | Nouveau Diagramme | Interface Utilisateur Win32

Support

La technologie MDG Win32® Interface Utilisateur est disponible dans les éditions Enterprise Architect Professional , Corporate , Unified et Ultimate

Activation de la technologie Interface Utilisateur Win32



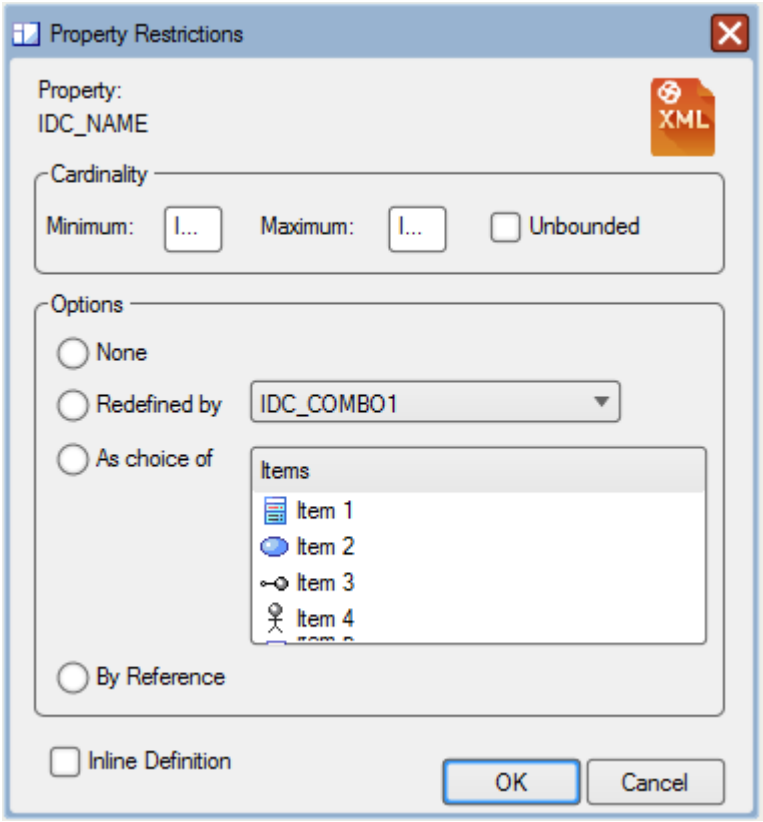
La technologie Win32® UI dans Enterprise Architect est activée ou désactivée à l'aide de la boîte dialogue « MDG Technologies » (sélectionnez l'option de ruban « Spécialiser > Technologies > Gérer la technologie »).

Technologie par défaut

Vous pouvez définir la technologie UI MDG Win32® comme technologie active par défaut pour accéder directement aux pages de la boîte à outils.

Boîtes de dialogue UI Modélisation

L' Interface Utilisateur MDG Technologie fournit les outils pour vous aider à concevoir une interface utilisateur qui émule fidèlement le style visuel et les options disponibles pour les boîtes de dialogue Windows .



Dialogue Win32

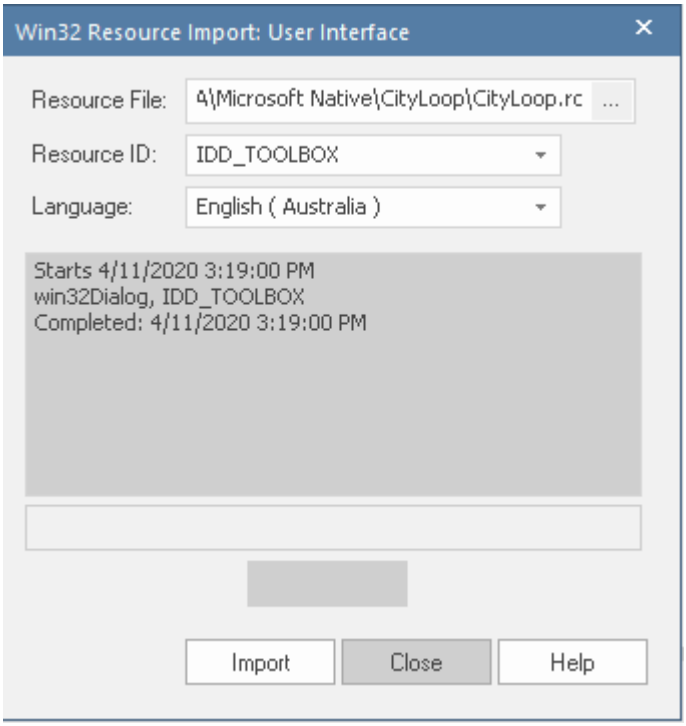
Ces composants d'interface utilisateur sont pris en charge, chacun correspondant à la ressource RC de nom équivalent.

Composant	Détails
win32Dialogue	L'équivalent des ressources DIALOG et DIALOGEX au format RC.
win32TexteStatique	L'équivalent des ressources LTEXT, RTEXT, CTEXT au format RC.
win32	L'équivalent de la ressource EDITTEXT au format RC.
bouton win32	L'équivalent du format RC BUTTON, DEFPUSHBUTTON et d'autres ressources.
win32CheckBox	L'équivalent de la ressource CHECKBOX au format RC.
win32ScrollBarH	L'équivalent de la ressource SCROLLBAR au format RC avec le style SBS_HORZ
win32ScrollBarV	L'équivalent de la ressource SCROLLBAR au format RC avec le style SBS_VERT.
win32GroupBox	L'équivalent de la ressource GROUPBOX au format RC.

win32ComboBox	L'équivalent de la ressource COMBOBOX au format RC. Note : Lorsque vous faites initialement glisser l'icône 'Combo Box' - de type 'Drop Down' ou 'Drop Down List' - sur un diagramme , la 'poignée de suivi' centrale de chaque côté de l'élément est blanche, indiquant que vous ne pouvez ajuster la largeur de l'élément. Pour ajuster la hauteur de l'élément ainsi que la largeur, cliquez sur la flèche déroulante faisant partie de l'image ; la « poignée de suivi » du milieu sur le bord inférieur est maintenant blanche, indiquant que vous pouvez faire glisser la base vers le bas pour définir la hauteur virtuelle (la hauteur de l'élément lorsqu'il est développé pour afficher toutes les valeurs possibles dans la liste déroulante).
win32ListBox	L'équivalent de la ressource LISTBOX au format RC.
win32RadioButton	L'équivalent de la ressource RADIOBUTTON au format RC.
win32TabPane	L'équivalent de la ressource TABPANE au format RC.
win32Image	L'équivalent de la ressource STATIC au format RC avec le style SS_BITMAP. Le contrôle peut restituer une image lorsqu'il est appliqué à partir de votre modèle. Une image peut être appliquée en la sélectionnant d'abord et en appuyant sur Ctrl+Shift+W pour afficher le gestionnaire d'images. Ensuite, vous devrez peut-être modifier la valeur de l' ID de ressource dans la Valeur Étiquetée appropriée.
win32CustomControl	L'équivalent de la ressource CONTROL au format RC.

Importer Dialogue unique à partir d'un fichier RC

Vous pouvez importer rapidement un seul dialogue par son nom.



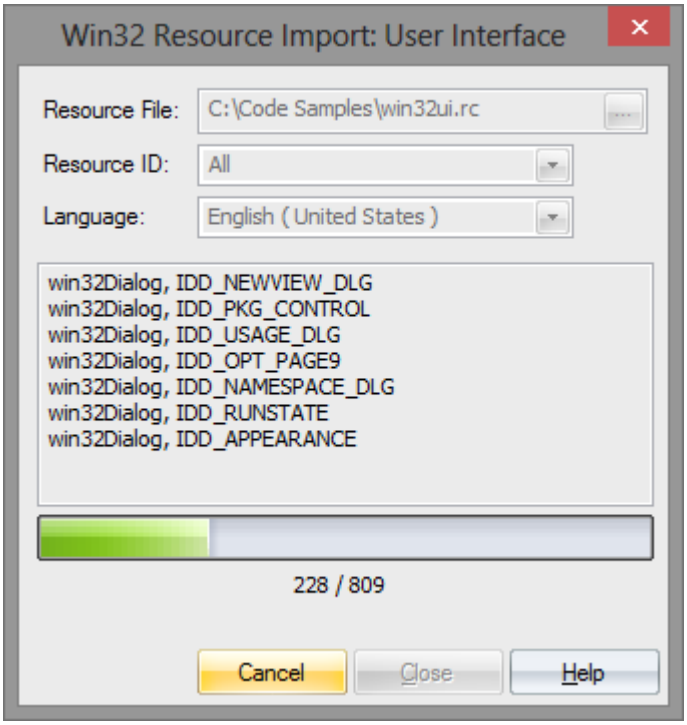
Accéder

Dans la fenêtre Navigateur , cliquez sur le Paquetage cible.

Ruban	Développer > Code source > Fichiers > Importer un script de ressource

Importer toutes les boîtes de dialogue à partir du fichier RC

Toutes les boîtes de dialogue d'un seul fichier RC peuvent être importées dans votre modèle. Cette image a été capturée une minute après le début de l'importation, moment auquel plus de 200 grandes définitions dialogue avaient été importées.

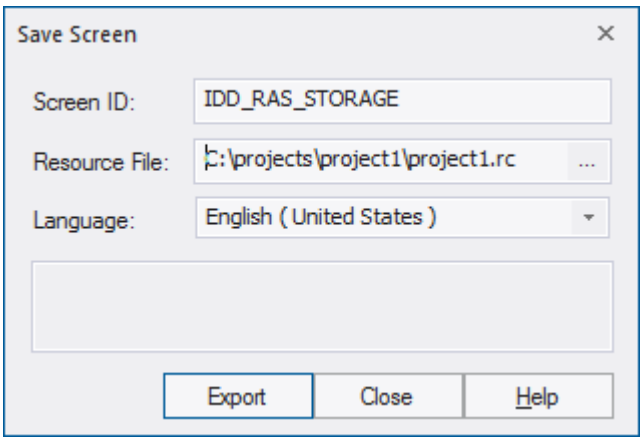


Accéder

Ruban	Développer > Code source > Fichiers > Importer un script de ressource

Exporter Dialogue vers un fichier RC

Une fois qu'une conception d'écran est modifiée ou qu'une nouvelle est créée, vous souhaiterez peut-être la récupérer dans le fichier RC que vous utilisez pour créer votre application, afin de pouvoir voir à quoi elle ressemble avec des données réelles. Commencez par sélectionner l'élément Win32Dialog dans la fenêtre Navigateur , puis utilisez le ruban pour effectuer la synchronisation.



Accéder

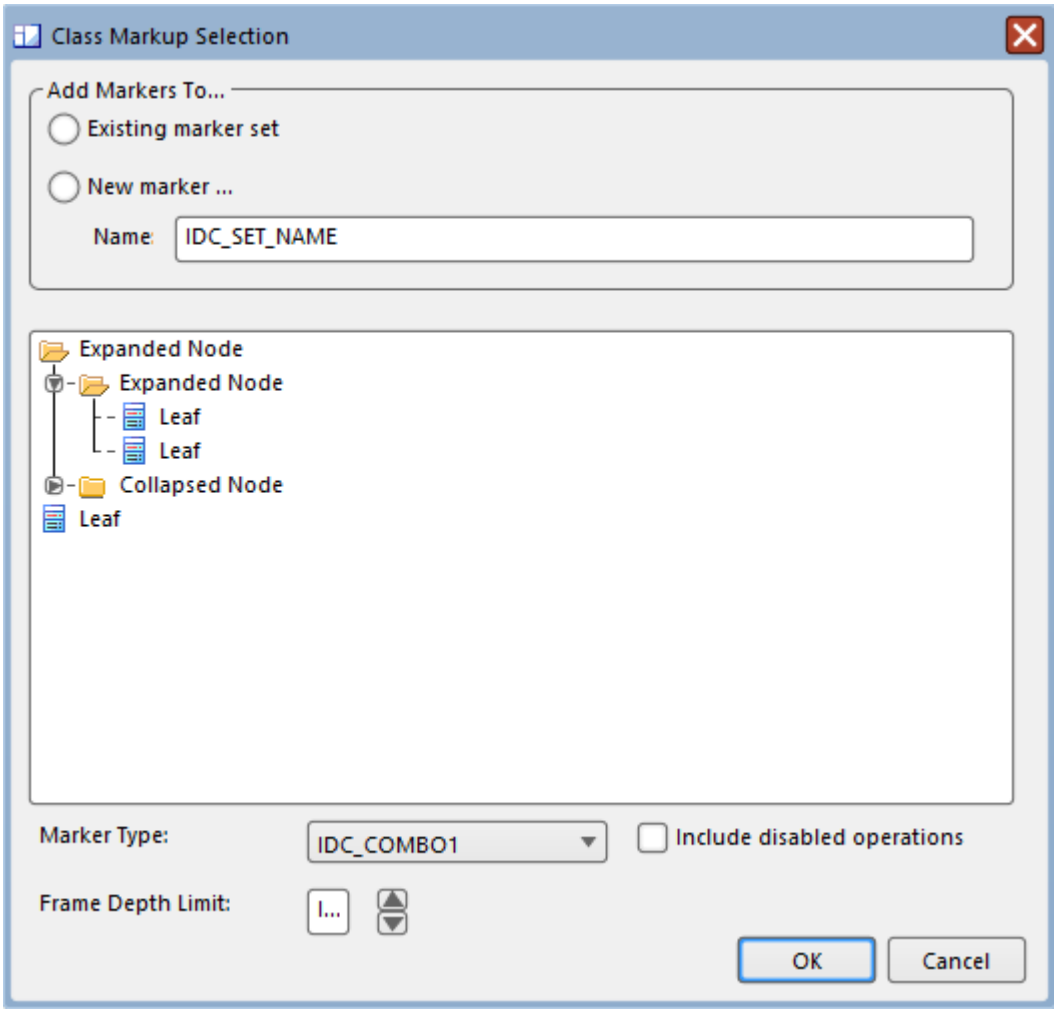
Cliquez sur l'élément win32Dialog.

Ruban	Développer > Code Source > Générer > Générer un seul élément
Raccourcis Clavier	F11

Concevoir un nouveau Dialogue

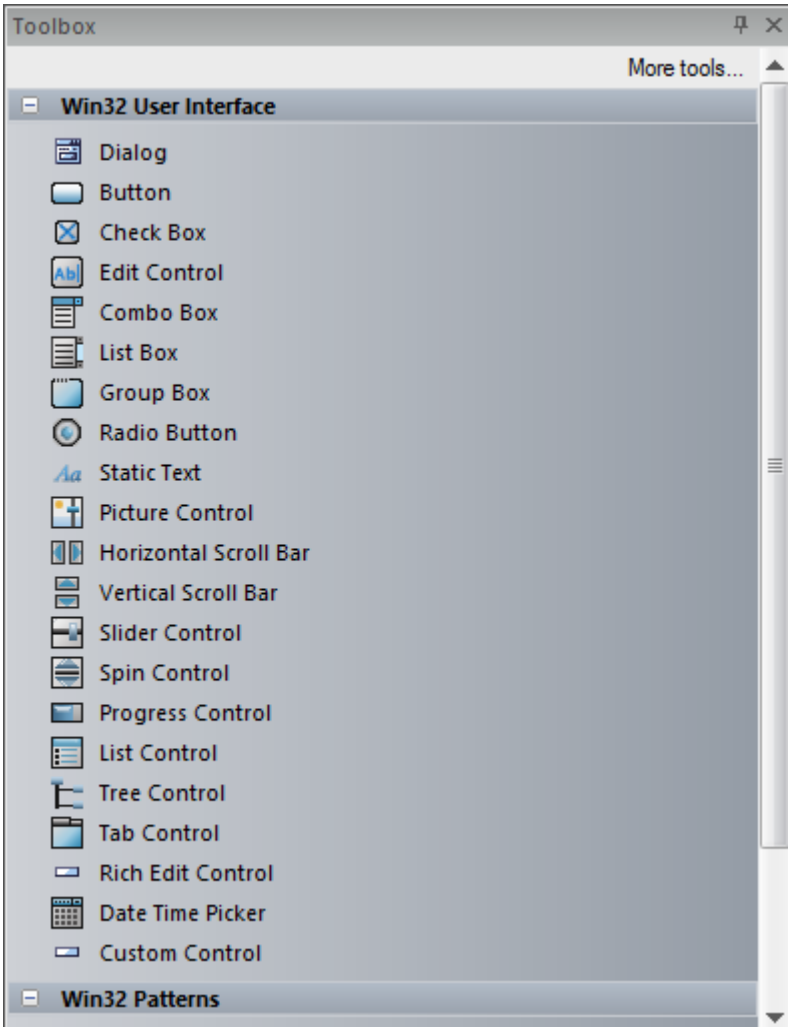
Créer un nouveau dialogue Win32 est simple et principalement visuel. Vous aurez probablement besoin d'un espace de travail qui affiche :

- Le nouveau diagramme (sélectionnez le chemin du ruban 'Conception > Diagramme > Ajouter Diagramme > Interface Utilisateur - Win32 > Interface Utilisateur - Win32')
- La boîte à outils Interface Utilisateur Win32 (sélectionnez l'option de ruban 'Conception > Diagramme > Boîte à outils') et
- L'onglet Valeur Étiquetés de la fenêtre Propriétés



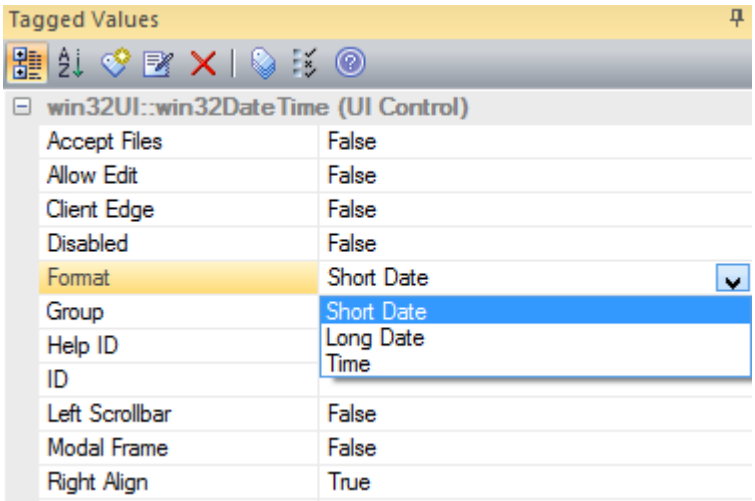
La boîte à outils UI

Tous les éléments RC courants peuvent être trouvés dans la boîte à outils UI



L'onglet Balises

Cet onglet est fourni dans la fenêtre Propriétés et dialogue ' Propriétés ' d'un object , et c'est là que toutes les propriétés d'un contrôle peuvent être affichées et modifiées.



Utilisation du Picture Control

Les images de votre modèle (voir *Image Manager*) peuvent être appliquées en sélectionnant le contrôle sur le dialogue et en appuyant sur Ctrl+Shift+W. Vous devrez peut-être saisir la valeur de l' ID de ressource dans le champ Valeur Étiquetée approprié.

Note

- Vous pouvez copier et coller Paquetages dialogue

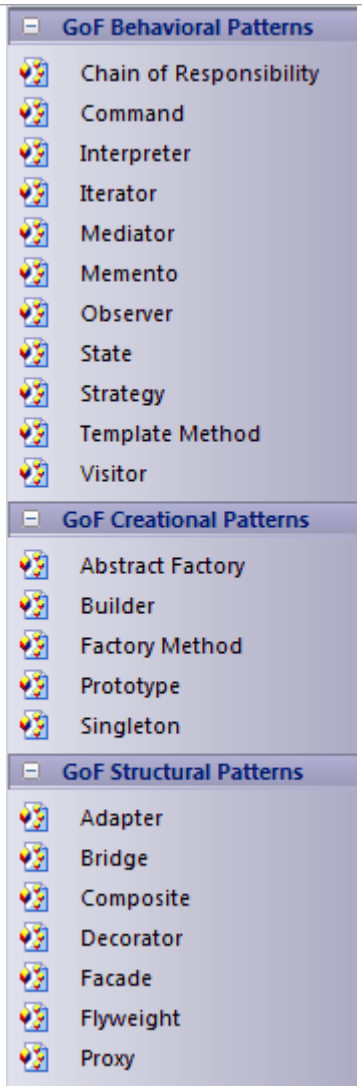
Motifs Gang des Quatre (GoF)

Un Motif de conception est un gabarit permettant de résoudre des problèmes de conception couramment récurrents ; il se compose d'une série d'éléments et de connecteurs pouvant être réutilisés dans un nouveau contexte. L'avantage de l'utilisation Motifs est qu'ils ont été testés et affinés dans un certain nombre de contextes et constituent donc généralement des solutions robustes à des problèmes courants. Enterprise Architect fournit le Gang of Four Motifs en tant que MDG Technologie qui peut être chargée dans le référentiel actuel.

Les Motifs Gang of Four (Gof) sont un groupe de vingt-trois Motifs conception initialement publiés dans un livre fondateur intitulé *Design Motifs : Elements of Realistic Object-Oriented Software* ; le terme « Gang of Four » fait référence aux quatre auteurs. Enterprise Architect affiche ces Motifs dans son moteur Motif , vous aidant à visualiser les éléments du Motif et à ajuster le Motif au contexte de votre problème de conception logicielle.

Motifs GoF dans Enterprise Architect

Fonctionnalités	Description
Facilités GoF Motif	Les Motifs GoF sont fournis sous la forme de : • Pages Motifs Comportementale du GoF, Motifs créationnels du GoF et Motifs structurels du GoF dans la boîte à outils • Entrées de groupe de quatre Motif dans le menu contextuel de la boîte à outils **Pages de la boîte à outils GoF Motif** Vous pouvez accéder aux pages 'GoF Motif ' de la Toolbox en cliquant sur pour afficher la dialogue ' Trouvez Item de Boîte à Outils ' et en spécifiant 'GoF Motifs ' ; ces icônes sont disponibles :



Lorsque vous faites glisser l'un des éléments Motif sur un nouveau diagramme , la dialogue « Ajouter Motif GoF < groupe motif >< type motif > » s'affiche ; si nécessaire, modifiez l'action et/ou le défaut des éléments du composant, puis cliquez sur le bouton OK pour créer un diagramme basé sur le Motif .

ICONIX

Le processus ICONIX est une méthodologie propriétaire de développement de logiciels basée sur UML . Le processus est basé sur des cas d'utilisation et utilise diagrammes basés sur UML pour définir quatre étapes. La principale fonctionnalité du processus est un concept appelé modélisation de robustesse, basé sur les premiers travaux d'Ivar Jacobson, qui permet de combler le fossé entre l'analyse et la conception.

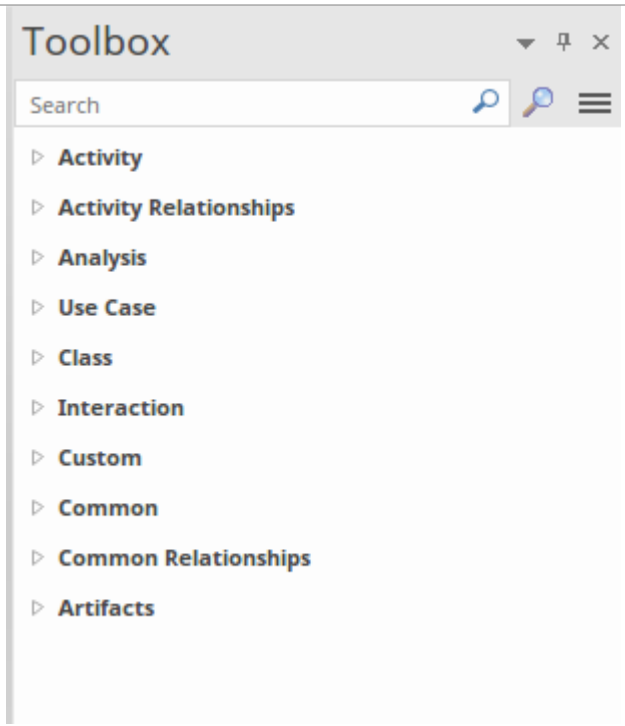
Ce texte est dérivé de l'entrée ICONIX dans Wikipédia en ligne :

« Le processus ICONIX est une approche minimaliste et rationalisée de modélisation UML basée sur des cas d'utilisation qui utilise un sous-ensemble principal de diagrammes et de techniques UML pour fournir une couverture complète de l'analyse et de la conception orientées objet. Son activité principale est l'analyse de robustesse, une méthode permettant de combler le fossé entre l'analyse et la conception. L'analyse de robustesse réduit l'ambiguïté dans les descriptions de cas d'utilisation, en garantissant qu'elles sont écrites dans le contexte d'un modèle de domaine qui les accompagne. Ce processus rend les cas d'utilisation beaucoup plus faciles à concevoir, tester et estimer.

Le processus ICONIX a été développé par Doug Rosenberg ; pour plus d'informations sur ICONIX, consulter ICONIX Ingénierie de Logiciel Inc.

Aspects

Aspect	Détail
ICONIX dans Enterprise Architect	Enterprise Architect vous permet de développer des modèles sous ICONIX rapidement et simplement, grâce à l'utilisation d'une MDG Technologie intégrée au programme d'installation Enterprise Architect . Les facilités ICONIX se présentent sous la forme de : • Un ensemble de pages ICONIX dans la boîte à outils • Éléments ICONIX et entrées de relation dans le menu « Raccourci de la boîte à outils » et Quick Linker Pour vous aider davantage à développer et gérer un projet sous ICONIX, Enterprise Architect propose également un livre blanc sur la Feuille de Route ICONIX.
Pages de la boîte à outils ICONIX	Dans la boîte à outils, Enterprise Architect fournit des versions ICONIX des pages pour les diagrammes d'analyse UML , de cas d'utilisation, de classe, d' diagrammes (Séquence), d'activité et personnalisés (qui constituent souvent la base des diagrammes de robustesse). Par rapport aux pages Toolbox standard, celles-ci comportent des ensembles d'éléments et de relations légèrement différents ; vous pouvez y accéder soit : • Spécifier 'ICONIX' dans la dialogue ' Trouvez Item de Boîte à Outils ' et sélectionner une page de boîte à outils spécifique • En sélectionnant l'option « ICONIX » dans le champ déroulant de la barre d'outils Outils par défaut, qui ajoute les six pages à la boîte à outils ; toutes les pages sont fermées


Toolbox
Search
Activity
Activity Relationships
Analysis
Use Case
Class
Interaction
Custom
Common
Common Relationships
Artifacts

Paramètres de configuration

Vous pouvez définir les options de code par défaut telles que les éditeurs pour chacun des langages de programmation disponibles pour Enterprise Architect et des options spéciales pour la manière dont le code source est généré ou rétro-conçu. Ces options sont définies selon qu'elles s'appliquent :

- Tous les utilisateurs du modèle actuel, définis dans la dialogue 'Gérer les options Modèle ', ou
- Tous les modèles auxquels vous accédez (les autres utilisateurs peuvent définir leurs propres paramètres qui s'appliquent aux mêmes modèles), définis dans la dialogue « Préférences »

Vous pouvez aussi:

- Pour chaque langage de programmation utilisé dans le modèle, pour tous les utilisateurs travaillant sur le modèle, définissez des classes de collection pour générer du code à partir des connecteurs d'association où le rôle cible a un paramètre de multiplicité supérieur à 1.
- Définissez vous-même un chemin local, en utilisant la dialogue 'Chemin Local' ; ces paramètres s'appliquent à tous les modèles Enterprise Architect auxquels vous accédez
- Définir des macros de langage dans le modèle, qui sont utiles en rétro-ingénierie et peuvent être exportées et importées vers le modèle

Options d'ingénierie du code source

Les options « Ingénierie du code source » s'appliquent aux langages dans lesquels vous générez du code à partir Enterprise Architect . Elles sont divisées en options spécifiques au modèle et en options spécifiques à l'utilisateur, comme expliqué ici.

Options spécifiques au modèle

Ces options sont définies dans la dialogue 'Gérer les options Modèle '.

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source

Types d'options

Type d'option	Détail
Options de génération de code source	Vous pouvez définir un certain nombre de paramètres pour générer du code dans le modèle, tels que la langue par défaut dans laquelle générer le code et le jeu de caractères Unicode pour la génération de code.
Options - Durées de vie Object	Vous pouvez configurer diverses options concernant les durées de vie Object .
Options de langage de code	Pour chacun des langages de code supporte par Enterprise Architect , vous pouvez définir les options spécifiques au modèle et définir les classes de collection requises.

Options spécifiques à l'utilisateur

Ces options sont définies dans la dialogue 'Préférences'.

Accéder

Dans la dialogue « Préférences », cliquez sur « Source Code Engineering » dans la liste de gauche.

Ruban	Démarrer > Apparence > Préférences > Préférences
Raccourcis Clavier	Ctrl+F9

Types d'options

Type d'option	Détail
Options de génération de code source	Vous pouvez définir un certain nombre de paramètres pour générer du code dans n'importe quel modèle auquel vous accédez sous le même ID utilisateur.
Éditeurs de Code	Il s'agit d'options permettant d'accéder et de configurer l'éditeur de code source.
Attributes /Opérations	Utilisez ces options pour configurer les attributs et les opérations.
Options de langage de code	Pour chacun des langages de code supporte par Enterprise Architect , vous pouvez définir les options spécifiques à l'utilisateur qui s'appliquent à tout modèle auquel vous accédez sous votre ID utilisateur.

Options de génération de code

Lorsque vous générez du code pour votre modèle, vous pouvez définir certaines options. Ceux-ci inclus:

- La langue par défaut
- S'il faut générer des méthodes pour les interfaces implémentées
- Les options Unicode pour la génération de code

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source

Configurer les options de génération de code

Option	Action
Synchronisez toujours avec le fichier existant (recommandé)	Sélectionnez le bouton radio pour synchroniser le code importé avec un fichier existant.
Remplacer (écraser) le fichier source existant	Sélectionnez le bouton radio pour remplacer le fichier source existant par le code importé.
Types de composants	Cliquez sur ce bouton pour ouvrir la dialogue 'Importer les types de composants', pour paramétrer l'importation des types de composants.
Langue par défaut pour la génération de code	Cliquez sur la flèche déroulante et sélectionnez la langue par défaut pour la génération de code.
Nom DDL Gabarits	Cliquez sur le bouton ... pour définir les noms gabarit pour gabarits Primary Key , de contrainte unique, Foreign Key et de nom d'index Foreign Key .
Nom par défaut de l'attribut associé	Type un nom par défaut à générer à partir des attributs importés.
Générer des méthodes pour les interfaces implémentées	Cochez la case pour indiquer que les méthodes sont générées pour les interfaces implémentées.
Page de codes pour l'édition de la source	Cliquez sur la flèche déroulante et sélectionnez le format d'intégration de caractères Unicode approprié à appliquer.

Notes

- Il est intéressant de configurer ces paramètres, car ils servent de valeurs par défaut pour toutes les classes du modèle ; vous pouvez remplacer la plupart d'entre eux par classe en utilisant les paramètres personnalisés (à partir de la dialogue « Génération de code »)

Importer des types de composants

À l'aide de la dialogue « Importer les types de composants », vous pouvez configurer les éléments que vous souhaitez créer pour les fichiers de toute extension trouvée lors de l'importation d'un répertoire de code source.

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source : types de composants

Définir les types de composants d'importation

Option	Action
Extension	Type le nom d'extension d'un type de composant.
Type	Cliquez sur la flèche déroulante et sélectionnez le type de composant.
Stéréotype	Type n'importe quel nom de stéréotype qui identifie davantage un composant de ce type.
Liste des composants	Répertorie les types de composants actuellement définis.
Sauvegarder	Cliquez sur ce bouton pour enregistrer la définition du composant et l'ajouter à la liste des composants.
Nouveau	Cliquez sur ce bouton pour effacer le dialogue afin de pouvoir définir un nouveau type de composant.
Supprimer	Cliquez sur ce bouton pour supprimer le type de composant sélectionné de la liste des composants.

Notes

- Vous pouvez transporter ces types de composants d'importation entre les modèles, à l'aide des options du ruban « Paramètres > Modèle > Transférer > Exporter les données de référence » et « Importer les données de référence ».

Options du code source

Vous pouvez définir un large éventail d'options pour générer du code dans les modèles avec lesquels vous travaillez. Ceux-ci inclus:

- Comment formater le code généré
- Comment répondre à certains événements lors de la génération de code
- S'il faut générer un diagramme à partir du code

Accéder

Dans la dialogue « Préférences », sélectionnez l'option « Ingénierie du code source »

Ruban	Démarrer > Apparence > Préférences > Préférences
Raccourcis Clavier	Ctrl+F9

Configurer les options de génération de code

Champ	Action
Enveloppez les longues lignes de commentaires à	Type le nombre de caractères à autoriser dans une ligne de commentaire avant de passer le texte à la ligne suivante.
Diagramme Disposition automatique à l'importation	Cliquez sur la flèche déroulante et sélectionnez si et quand un diagramme est automatiquement généré lors de l'importation du code.
Type Diagramme Disposition par défaut	Cliquez sur la flèche déroulante et sélectionnez le type de disposition à appliquer aux diagrammes générés à partir du code.
Les fichiers de sortie utilisent à la fois CR et LF	Cochez la case pour inclure les retours chariot et les sauts de ligne ; définissez cette option en fonction du système d'exploitation actuellement utilisé, car le code pourrait ne pas s'afficher correctement.
Invite lors de la synchronisation (inversion)	Cochez la case pour afficher une prompt lorsque la synchronisation se produit.
Supprimer les pauses brutales des commentaires lors de l'importation	Cochez la case pour supprimer les coupures définitives des sections commentées lors de l'importation.
Générer automatiquement des noms de rôle lors de la création de code	Cochez la case pour générer des noms de rôle lors de la création du code.
Ne pas générer de membres lorsque la direction de l'association est « Non	Cochez la case pour empêcher la génération de membres si la direction de l'association n'est pas spécifiée.

spécifiée »	
Créer des dépendances pour les retours d'opération et les types de paramètres	Cochez la case pour générer des dépendances pour les retours d'opération et les types de paramètres.
Commentaires : Générer	Cochez la case pour générer des commentaires.
Commentaires : Inverse	Cochez la case pour générer des commentaires inversés.
Supprimer les préfixes lors de la génération des propriétés Get/Set	Type les préfixes, séparés par des points-virgules, utilisés dans les conventions de dénomination de vos variables, à supprimer dans les fonctions get/set correspondantes des variables.
Traiter comme des suffixes	Cochez la case pour utiliser les préfixes définis dans le champ « Supprimer les préfixes lors de la génération des propriétés Get/Set » comme suffixes.
Nom d'attribut en majuscule pour Propriétés	Cochez la case pour mettre en majuscule les noms d'attribut pour les propriétés.
Utilisez 'Est' pour la propriété booléenne Get()	Cochez la case pour utiliser le mot-clé Is pour la propriété booléenne Get().

Notes

- Il est intéressant de configurer ces paramètres, car ils servent de valeurs par défaut pour toutes les classes du modèle ; vous pouvez remplacer la plupart d'entre eux par classe en utilisant les paramètres personnalisés (à partir de la dialogue « Génération de code »)

Options - Éditeurs de Code

Vous accédez aux options de l'éditeur de code source via la page 'DDL' de la dialogue 'Préférences'. Sur cette page, vous pouvez configurer les options de l'éditeur interne d' Enterprise Architect , ainsi que l'éditeur par défaut pour les scripts DDL. Vous pouvez configurer des éditeurs externes pour les langages de code sur chaque page d'options de langue.

Accéder

Dans la dialogue 'Préférences', sélectionnez l'option 'Source Code Engineering > Éditeurs de Code '.

Ruban	Démarrer > Apparence > Préférences > Préférences
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Éditeur DDL	La valeur par défaut est vide, indiquant que l'éditeur de code Enterprise Architect est l'éditeur DDL utilisé. Vous pouvez sélectionner un autre éditeur par défaut si nécessaire ; cliquez sur le bouton ... pour rechercher et sélectionner l'éditeur DDL requis. Le nom de l'éditeur s'affiche alors dans le champ « Éditeur DDL ».
Base de données par défaut	Cliquez sur la flèche déroulante et sélectionnez la base de données par défaut à utiliser.
Moteur de stockage MySQL	Cliquez sur la flèche déroulante et sélectionnez le moteur de stockage MySQL à utiliser.
Utiliser l'éditeur intégré si aucun éditeur externe n'est défini	Cochez la case pour utiliser l'éditeur intégré pour le code dans n'importe quelle langue si aucun éditeur externe n'est défini pour cette langue dans les options spécifiques à l'utilisateur.
Afficher les numéros de ligne	Cochez la case pour afficher les numéros de ligne dans l'éditeur.
Afficher l'arborescence	Cochez la case pour afficher une arborescence avec les résultats de l'analyse du fichier ouvert (si le fichier est analysé avec succès).
Rétro-ingénierie automatique lors de l'enregistrement du fichier	Si vous cochez cette case, appuyez sur Ctrl+S pour enregistrer dans l'éditeur de code source, procédez automatiquement à une ingénierie inverse du code de la même manière que le fait le bouton Enregistrer la source et resynchroniser la classe.
N'analysez pas les fichiers plus grands que	Cliquez sur la flèche déroulante et sélectionnez la limite supérieure de la taille du fichier à analyser. La définition de cette option empêche la diminution des performances due à

	l'analyse de fichiers très volumineux.
Mise en évidence Police , du style et de la syntaxe	Cliquez sur le bouton ... pour afficher la dialogue ' Propriétés du langage de l'éditeur', dans laquelle vous pouvez définir les propriétés de langage de l'éditeur globales et spécifiques à la langue.
Configurer les associations de fichiers Enterprise Architect	Cliquez sur le bouton ... pour afficher la dialogue « Définir les associations pour un programme » et sélectionnez les extensions de fichier pour les fichiers que vous souhaitez ouvrir via le gestionnaire de documents Enterprise Architect .

Propriétés du langage de l'éditeur

À l'aide de la dialogue 'Editor Language Propriétés ', vous pouvez spécifier les propriétés de coloration syntaxique pour n'importe lequel des langages de programmation supporte Enterprise Architect lors de l'installation.

Accéder

Dans la dialogue « Préférences », sélectionnez « Source Code Engineering | Éditeurs de Code » et cliquez sur le bouton ... à côté de « Options de surbrillance de la syntaxe ».

Ruban	Démarrer > Apparence > Préférences > Préférences, sélectionnez « Source Code Engineering | Option Éditeurs de Code > cliquez sur le bouton ... à côté de « Options de surbrillance de la syntaxe »
Autre	Dans la fenêtre Éditeur de Code , cliquez sur l'icône de la barre d'outils | Options de coloration syntaxique

Possibilités

Panneau	Description
Panneau de langue	Le panneau à gauche de le dialogue répertorie les langues pour lesquelles vous pouvez définir des propriétés. En haut de la liste se trouvent trois options non linguistiques : • (Thème sombre) - attribue un arrière-plan sombre aux champs de propriétés et au panneau de code dans l'écran de l'éditeur de code (vous pouvez appliquer une couleur différente à des propriétés spécifiques) • (Thème clair) - attribue un arrière-plan pâle aux champs de propriétés et au panneau de code dans l'écran de l'éditeur de code (vous pouvez appliquer une couleur différente à des propriétés spécifiques) Vous pouvez également définir les thèmes d'arrière-plan dans la dialogue "Application Look". • (Global) fournit des propriétés que vous pouvez définir pour tous les langages de programmation ; cependant, vous pouvez réinitialiser une propriété globale à une valeur différente pour une langue particulière, dans les propriétés spécifiques à cette langue. La réinitialisation d'une propriété globale pour une langue n'affecte pas valeur de cette propriété pour les autres langues Cliquez sur la langue souhaitée dans la liste pour afficher les propriétés de cette langue : • Propriétés affichées en gras indiquent qu'il s'agit du niveau le plus élevé auquel cette propriété peut être définie (pour la plupart des options linguistiques autres que « Global », c'est effectivement le seul point auquel la propriété est définie) • Propriétés affichées en police normale sont généralement les propriétés globales que vous pouvez réinitialiser uniquement pour la langue actuelle.

Panneau Propriétés	Faites défiler les catégories de propriétés et les propriétés individuelles de la langue. Vous pouvez réduire et développer les catégories si nécessaire, à l'aide de la zone d'extension située à côté du nom de la catégorie ( ). Lorsque vous cliquez sur le nom d'une propriété, une explication de cette propriété s'affiche dans le panneau en bas à droite de le dialogue . Pour définir une propriété, cliquez sur le champ valeur suivant le nom de la propriété ; selon le type de propriété, soit le champ est activé pour l'édition directe, soit une flèche déroulante ou un bouton s'affiche (comme décrit pour l'onglet 'Tags' de la fenêtre Propriétés) afin que vous puissiez sélectionner les valeurs pour définir la propriété. . Sélectionnez ou saisissez les valeurs requises. Utilisez les icônes de la barre d'outils pour : • Enregistrez vos modifications dans les propriétés • Réinitialiser tous les champs de propriétés aux paramètres par défaut fournis avec Enterprise Architect • Réinitialiser le champ de style actuel au paramètre par défaut (non activé pour les champs sans style)
Attribuer des clés aux macros	Dans la catégorie 'Macros' des propriétés, vous pouvez attribuer des combinaisons de touches (Ctrl+Alt+<n>) aux macros de codage que vous avez créées vous-même dans la ' Visionneuse de code source '. Lorsque vous cliquez sur le bouton Parcourir dans un champ « Macro » sélectionné, la dialogue « Ouvrir la macro » s'affiche ; cette dialogue répertorie les macros existantes et, si une combinaison de touches a été attribuée à une macro, quelle est cette combinaison de touches. Cliquez sur le nom de la macro et sur le bouton Ouvrir pour attribuer les touches sélectionnées à la macro.

Notes

- Vous ne pouvez actuellement pas définir de propriétés pour les langues supplémentaires que vous incluez via une MDG Technologie
- Vous pouvez redimensionner cette dialogue , si nécessaire

Options - Durées de vie Object

Vous pouvez utiliser ces options pour configurer divers paramètres de durée de vie Object tels que :

- Définir les détails du constructeur lors de la génération du code
- Spécifier s'il faut créer un constructeur de copie
- Définir les détails du destructeur

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > Durées de vie Object

Possibilités

Option	Action
Constructeur	Si nécessaire, cochez les cases pour spécifier qu'un constructeur est généré et (pour C++) que le constructeur est en ligne. Cliquez sur la flèche déroulante et sélectionnez la visibilité appropriée du constructeur par défaut - Privé, Protégé ou Public.
Copier le constructeur	Si nécessaire, cochez les cases pour spécifier qu'un constructeur de copie est généré et (pour C++) que le constructeur de copie est en ligne. Cliquez sur la flèche déroulante et sélectionnez la visibilité appropriée du constructeur de copie par défaut : Privé, Protégé ou Public.
Destructeur	Si nécessaire, cochez les cases pour spécifier qu'un destructeur est généré et (pour C++) que le destructeur est en ligne et/ou virtuel. Cliquez sur la flèche déroulante et sélectionnez la visibilité appropriée du destructeur par défaut : Privé, Protégé ou Public.

Options - Attribut/Opérations

Votre utilisation des attributs et des opérations peut être configurée de plusieurs manières. Vous pouvez définir des options pour :

- Supprimer les attributs du modèle non inclus dans le code lors de la synchronisation inverse
- Supprimer les méthodes de modèle non incluses dans le code lors de la synchronisation inverse
- Supprimer le code des fonctionnalités contenues dans le modèle lors de la synchronisation directe
- Supprimer les associations et agrégations de modèles qui correspondent à des attributs non inclus dans le code lors de la synchronisation inverse
- Définir si les corps de méthodes sont inclus et enregistrés dans le modèle lors de la rétro-ingénierie
- Créez fonctionnalités en succession rapide, en effaçant la fenêtre Propriétés lorsque vous cliquez sur "Enregistrer" afin de pouvoir saisir un autre nom fonctionnalité

Vous configurez ces options sur la page 'Attributs/Opérations' de la dialogue 'Préférences'.

Accéder

Dans la dialogue « Préférences », sélectionnez l'option « Ingénierie du code source > Attributs/Opérations ».

Ruban	Démarrer > Apparence > Préférences > Préférences
Raccourcis Clavier	Ctrl+F9

Possibilités

Champ	Action
Lors de la synchronisation inverse, supprimez les attributs du modèle qui ne sont pas dans le code	Cochez la case pour indiquer que lors de la synchronisation inverse, les attributs du modèle qui ne sont pas inclus dans le code sont automatiquement supprimés du modèle.
Lors de la synchronisation inverse, supprimez les associations de modèles qui ne sont pas dans le code	Cochez la case pour indiquer que lors de la synchronisation inverse, les associations du modèle qui ne sont pas incluses dans le code sont automatiquement supprimées du modèle.
Lors de la synchronisation inverse, supprimez les méthodes de modèle qui ne sont pas dans le code	Cochez la case pour indiquer que lors de la synchronisation inverse, les méthodes du modèle qui ne sont pas incluses dans le code sont automatiquement supprimées du modèle.
Inclure les corps de méthode dans le modèle lors de la rétro-ingénierie	Cochez la case pour indiquer que dans le code de rétro-ingénierie, les corps de méthode du code sont inclus dans votre modèle.
Après l'enregistrement, resélectionnez l'élément	Cochez la case pour indiquer qu'après avoir enregistré un attribut ou une opération, la définition des propriétés continue d'afficher les détails de la fonctionnalité

modifié	sélectionnée. Si cette option est désélectionnée, cela indique que les champs de la définition des propriétés seront effacés afin que vous puissiez saisir immédiatement un autre nom d'attribut ou d'opération et des détails.
Lors de la synchronisation directe, prompt à supprimer fonctionnalités du code qui ne figurent pas dans le modèle	Cochez la case pour indiquer que, lors de la synchronisation directe, la dialogue 'Synchroniser l'élément < nom paquetage >.<nom de l'élément>' s'affiche, afin que vous puissiez soit ignorer, réaffecter ou supprimer fonctionnalités du code qui ne sont pas dans le modèle.

Conventions Modélisation

La synchronisation entre les modèles UML et le code de programmation est réalisée à l'aide d'un ensemble de conventions modélisation (mappings) entre les constructions UML et la syntaxe du code de programmation. Il est conseillé à l'ingénieur logiciel de se familiariser avec ces conventions afin de travailler avec le processus de génération de code pour les langages de programmation qu'il entend cibler. Il existe une gamme de constructions utilisées, notamment des éléments, fonctionnalités , des connecteurs, des extrémités de connecteur, des stéréotypes et Valeur Étiquetés . Le nouveau venu aura besoin d'un peu de temps pour se familiariser avec ces conventions, mais après peu de temps, il pourra traduire sans effort entre le code de programmation et les constructions UML .

Langues prises en charge

Langue
Script Action
Ada 2012 (éditions Unified et Ultimate)
C
C#
C++
Delphes
Java
PHP
Python
SystemC (éditions Unified et Ultimate)
Verilog (éditions Unified et Ultimate)
VHDL (éditions Unified et Ultimate)
Visual Basic
Visual Basic .NET

Notes

Enterprise Architect intègre un certain nombre d'indicateurs de visibilité ou de valeurs de portée pour ses langages pris en charge ; ceux-ci incluent, pour :

- Toutes les langues : Public (+), Protégé (#) et Privé (-)
- Java - Paquetage (~)
- Delphes - Publié (^)
- C# - Interne (~), Interne protégé (^)
- ActionScript - Interne (~)
- VB.NET - Ami (~), Ami protégé (^)
- PHP - Paquetage (~)
- Python - Paquetage (~)
- C - Paquetage (~)
- C++ - Paquetage (~)

Conventions ActionScript

Enterprise Architect supporte l'ingénierie round retour d'ActionScript 2 et 3, où ces conventions sont utilisées.

Stéréotypes

Stéréotype	S'applique à
littéral	Opération Correspond à : une méthode littérale référencée par une variable.
propriété obtenir	Opération Correspond à : Une propriété « lecture ».
ensemble de propriétés	Opération Correspond à : Une propriété « écriture ».

Valeur Étiquetés

Étiqueter	S'applique à
Nom d'attribut	Opération avec une propriété de stéréotype get ou un ensemble de propriétés Correspond à : Le nom de la variable derrière cette propriété.
dynamique	Classe ou interface Correspond à : Le mot-clé « dynamique ».
final	ActionScript 3 : fonctionnement Correspond à : Le mot-clé « final ».
intrinsèque	ActionScript 2 : classe Correspond à : Le mot-clé 'intrinsèque'.
espace de noms	ActionScript 3 : classe, interface, attribut, opération Correspond à : l'espace de noms de l'élément actuel.
passer outre	ActionScript 3 : fonctionnement Correspond à : Le mot-clé 'override'.
prototype	ActionScript 3 : attribut Correspond à : Le mot-clé 'prototype'.
repos	ActionScript 3 : paramètre Correspond à : Le paramètre rest (...)

Conventions communes

- Les qualificateurs Paquetage (ActionScript 2) et Paquetages (ActionScript 3) sont générés lorsque le Paquetage actuel n'est pas une racine d'espace de noms.
- Un type non spécifié est modélisé comme 'var' ou un champ ' Type ' vide

Conventions ActionScript 3

- La propriété Is Leaf d'une Classe correspond au mot clé scellé
- Si une balise d'espace de noms est spécifiée, elle remplace la portée spécifiée

Congrès Ada 2012

Enterprise Architect supporte l'ingénierie round retour d'Ada 2012, où ces conventions sont utilisées.

Stéréotypes

Stéréotype	S'applique à
adaPackage	Classe Correspond à : une spécification Paquetage dans Ada 2012 sans enregistrement balisé.
adaProcédure	Classe Correspond à : Une spécification de procédure dans Ada 2012.
déléguer	Opération Correspond à : Accès à un sous-programme.
énumération	Classe intérieure Correspond à : un type énuméré.
structurer	Classe intérieure Correspond à : Une définition d'enregistrement.
typedef	Classe intérieure Correspond à : une définition de type, une définition de sous-type, une définition de type d'accès, un changement de nom.

Valeur Étiquetés

Étiqueter	S'applique à
Aspect	Classe interne avec typedef de stéréotype Opération Correspond à : Spécification d'Aspect (Précondition et Postcondition de Sous-Programme de type 'invariant', sous-type 'prédicat').
Type d'unité instanciée	Classe interne avec typedef de stéréotype Correspond à : Le type de l'unité instanciée (Paquetage / Procédure / Fonction).
EstAccès	Paramètre Correspond à : Détermination si le paramètre est une variable d'accès.
EstAliasé	Paramètre de fonction

	Correspond à : Paramètre de fonction alias.
Discriminant	Classe interne avec typedef de stéréotype Correspond à : Le discriminant du type.
Type de pièce	Classe interne avec typedef de stéréotype Correspond à : Le type de pièce (« renomme » ou « nouveau »).
Type	Classe interne avec typedef de stéréotype Correspond à : Si 'Value' = 'SubType', définissez 'subtype' Si 'Valeur' = 'Accès', définissez le 'type d'accès'.

Autres congrès

- Type approprié de fichiers sources : fichier de spécification Ada, .ads
- Ada 2012 importe Paquetages définis comme « adaPackage » ou classe, en fonction des paramètres des options d'Ada 2012.
- Un Paquetage dans le fichier de spécification Ada est importé en tant que classe s'il contient un enregistrement balisé, dont le nom est régi par les options « Utiliser le nom de classe pour l'enregistrement balisé » et « Nom alternatif de l'enregistrement balisé » ; tous les attributs définis dans cet enregistrement balisé sont absorbés en tant qu'attributs de la classe
- Une procédure/fonction dans un fichier de spécification Ada est considérée comme la fonction membre de la classe si son premier paramètre satisfait aux conditions spécifiées dans les options « Style de paramètre de référence », « Ignorer le nom du paramètre de référence » et « Nom du paramètre de référence ».
- L'option « Définir la référence pour l'enregistrement balisé », si elle est activée, crée un type de référence pour la classe, dont le nom est déterminé par l'option « Nom Type référence » ; Par exemple:

HelloWorld.ads

paquetage HelloWorld est

tapez HelloWorld est étiqueté enregistrement

Att1 : Naturel ;

Att3 : Integer ;

terminer l'enregistrement ;

- Fonctions publiques

fonction MyPublicFunction (P: HelloWorld) return String ;

procédure MyPublicFunction (P1 : entrée sortie HelloWorld ; AFlag : Boolean) ;

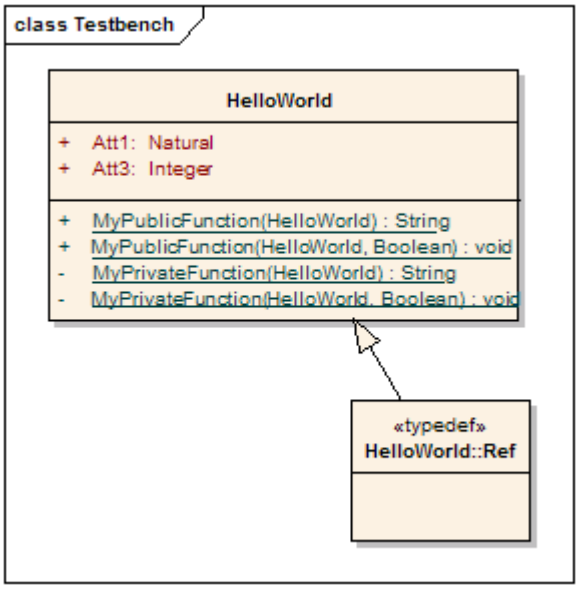
privé

- Fonctions privées

fonction MyPrivateFunction (P: HelloWorld) return String ;

procédure MyPrivateFunction (P1 : entrée sortie HelloWorld ; AFlag : Boolean) ;

mettre fin à HelloWorld ;



Notes

- support d'Ada 2012 est disponible dans les éditions Unified et Ultimate d' Enterprise Architect

Conventions C

Enterprise Architect supporte l'ingénierie round retour du C, où ces conventions sont utilisées :

Stéréotype

Stéréotype	S'applique à
énumération	Classe intérieure Correspond à : un type énuméré.
structurer	Classe intérieure Correspond à : Un type 'struct'.
Attribut	Une structure de mot-clé dans la définition de variable.
typedef	Classe intérieure Correspond à : Une instruction 'typedef', où le parent est le nom du type d'origine.
syndicat	Classe intérieure Correspond à : Un type d'union.
Attribut	Une union de mots clés dans une définition de variable.

Valeur Étiquetés

Étiqueter	S'applique à
anonyme	Classe contenant également le typedef Valeur Étiquetée Correspond à : Le nom de cette Classe étant défini uniquement par l'instruction typedef.
champ de bits	Attribut Correspond à : La taille, en bits, autorisée pour le stockage de cet attribut.
corpsEmplacement	Opération Correspond à : l'emplacement vers lequel le corps de la méthode est généré ; les valeurs attendues sont header, classDec ou classBody.
typedef	Classe avec un stéréotype autre que « typedef » Correspond à : Cette classe étant définie dans une instruction 'typedef'.
typeSynonymes	Classe Correspond à : Le nom 'typedef' et/ou les champs de ce type.

Génération de code C pour Modèle UML

UML	Code C
Une classe	Une paire de fichiers C (.h + .c) Notes : Le nom du fichier est le même que le nom de la classe
Exploitation (publique et protégée)	Déclaration de fonction dans le fichier .h et définition dans le fichier .c Notes :
Opération (privée)	Définition de fonction dans le fichier .c uniquement Notes :
Fonctionnement (statique)	Définition de fonction dans le fichier .c uniquement Notes : Les fonctions statiques n'apparaîtront que dans le fichier .c quelle que soit leur portée.
Attribut (public et protégé)	Définition de variable dans le fichier .h Notes :
Attribut (privé)	Définition de variable dans le fichier .c Notes :
Classe intérieure (sans stéréotype)	(N / A) Notes : Cette classe interne serait ignorée

Capturer #define valeur à générer dans le code C

Par exemple, #define PI 3.14.

Étape	Processus
1	Ajoutez un attribut à la classe, avec Name = PI et Initial Value = 3.14.
2	Dans le panneau des propriétés de la page ' Attributes ', mettez à jour les champs 'Static' et 'Const'.
3	Sur l'onglet ' Valeur Étiquetés ' de la page ' Attributes ', ajoutez une balise appelée 'define' avec la valeur True.

Notes

- Des conventions distinctes s'appliquent à la programmation orientée Object en C

Programmation orientée Object en C

Dans Enterprise Architect , vous appliquez un certain nombre de conventions pour la programmation orientée objet en C.

Pour configurer le système pour support la programmation orientée objet à l'aide de C, vous devez définir l'option « Support orientée Object » sur True sur la page « Spécifications C » de la dialogue « Préférences ».

Stéréotypes

Stéréotype	S'applique à
énumération	Classe Correspond à : un type énuméré.
structurer	Classe Correspond à : Un type 'struct'.
Attribut	Une structure de mot-clé dans la définition de variable.
typedef	Classe Correspond à : Une instruction 'typedef', où le parent est le nom du type d'origine.
syndicat	Classe Correspond à : Un type d'union.
Attribut	Une union de mots clés dans une définition de variable.

Valeur Étiquetés

Étiqueter	S'applique à
anonyme	Classe avec le stéréotype « énumération », « struct » ou « union » Correspond à : Le nom de cette Classe étant défini uniquement par l'instruction typedef.
corpsEmplacement	Opération Correspond à : l'emplacement vers lequel le corps de la méthode est généré ; les valeurs attendues sont « header », « classDec » ou « classBody ».
définir	Attribut Correspond à : instruction '#define'.
typedef	Classe avec le stéréotype « énumération », « struct » ou « union » Correspond à : Cette classe étant définie dans une instruction 'typedef'.

Génération de code C orienté objet pour UML Modèle

L'idée de base de l'implémentation d'une classe UML dans du code C est de regrouper la variable de données (attributs UML) dans un type de structure ; cette structure est définie dans un fichier .h afin qu'elle puisse être partagée par d'autres Classes et par le client qui y a fait référence.

Une opération dans une classe UML est implémentée dans le code C en tant que fonction ; le nom de la fonction doit être un nom complet composé du nom de l'opération, ainsi que du nom de la classe pour indiquer que l'opération concerne cette classe.

Un délimiteur (spécifié dans l'option « Délimiteur Namespace » sur la page « Spécifications C ») est utilisé pour joindre le nom de la classe et le nom de la fonction (opération).

La fonction dans le code C doit également avoir un paramètre de référence à l' object Class. Vous pouvez modifier les options « Référence en tant que paramètre d'opération », « Style de paramètre de référence » et « Nom du paramètre de référence » sur la page « Spécifications C » pour support cette référence. paramètre.

Limites de la programmation orientée objet en C

- Pas de mappage de portée pour un attribut : un attribut dans une classe UML est mappé à une variable de structure en code C, et sa portée (privée, protégée ou publique) est ignorée
- Actuellement, une classe interne est ignorée : si une classe UML est la classe interne d'une autre classe UML , elle est ignorée lors de la génération du code C.
- valeur initiale est ignorée : la valeur initiale d'un attribut dans une classe UML est ignorée dans le code C généré

Conventions C#

Enterprise Architect supporte l'ingénierie round retour de C# , où ces conventions sont utilisées.

Stéréotypes

Stéréotype	S'applique à
énumération	Classe Correspond à : un type énuméré.
événement	Opération Correspond à : Un événement.
extension	Opération Correspond à : une méthode d'extension de classe, représentée dans le code par un paramètre 'this' dans la signature.
indexeur	Opération Correspond à : une propriété agissant comme un index pour cette classe.
partiel	Opération Correspond à : Le mot-clé 'partial' sur une opération.
propriété	Opération Correspond à : une propriété contenant éventuellement du code de lecture et d'écriture.
enregistrer	Classe Correspond à : Un type « enregistrement ».
enregistrement_struct	Classe Correspond à : Un type 'record struct'.
structurer	Classe Correspond à : Un type 'struct'.

Valeur Étiquetés

Étiqueter	S'applique à
nomargument	Opération avec extension du stéréotype Correspond à : Le nom donné à ce paramètre.

Nom d'attribut	Opération avec une propriété ou un événement stéréotypé Correspond à : le nom de la variable derrière cette propriété ou cet événement.
nom du cours	Opération avec extension du stéréotype Correspond à : la classe à laquelle cette méthode est ajoutée.
const	Attribut Correspond à : Le mot-clé const.
définition	Opération avec stéréotype partiel Correspond à : Qu'il s'agisse de la déclaration de la méthode ou de la définition.
déléguer	Opération Correspond à : le mot-clé « délégué ».
enumType	Opération avec propriété de stéréotype Correspond à : le type de données sous lequel la propriété est représentée.
expressionCorps	Opération, Opération avec propriété de stéréotype ou indexeur Correspond à : « Vrai » si le « Code de comportement » provient d'un membre de fonction de corps d'expression.
extensionAttribute	Opération avec extension du stéréotype. Correspond à : L'attribut donné à ce paramètre.
externe	Opération Correspond à : Le mot-clé 'extern'.
fixé	Attribut Correspond à : le mot-clé « fixe ».
générique	Opération Correspond à : Les paramètres génériques de cette opération.
Contraintes génériques	Classe ou interface modélisée, opération avec la balise « générique » Correspond à : Les contraintes sur les paramètres génériques de ce type ou de cette opération.
Met en oeuvre	Opération Correspond à : le nom de la méthode implémentée, y compris le nom de l'interface.
ImplémenteExplicit	Opération Correspond à : la présence du nom de l'interface source dans cette déclaration de méthode.
initialiseur	Opération Correspond à : Une liste d'initialisation du constructeur.
nouveau	Classe, interface, fonctionnement

	Correspond à : Le mot-clé 'nouveau'.
passer outre	Opération Correspond à : Le mot-clé 'override'.
paramètres	Paramètre Correspond à : Une liste de paramètres utilisant le mot-clé 'params'.
partiel	Classe, Interface Correspond à : Le mot-clé 'partial'.
propriétéInitialiseur	Opération avec propriété de stéréotype Correspond à : un initialiseur de propriété.
lecture seulement	Opération, <<struct>>Classe Correspond à : Le mot-clé 'readonly'.
Paramètres positionnels	<<enregistrer>> Classe Correspond à : le paramètre de position dans la définition d'enregistrement.
réf	Opération, <<struct>>Classe Correspond à : Le mot-clé 'ref'.
scellé	Opération Correspond à : Le mot-clé « scellé ».
statique	Classe Correspond à : Le mot-clé 'static'.
peu sûr	Classe, interface, fonctionnement Correspond à : Le mot-clé « unsafe ».
virtuel	Opération Correspond à : Le mot-clé 'virtuel'.
écriture seule	Opération avec propriété de stéréotype Correspond à : Cette propriété définit uniquement le code « écriture ».

Autres congrès

- Namespaces sont générés pour chaque Paquetage sous une racine d'espace de noms
- La propriété Const d'un attribut correspond au mot-clé readonly, tandis que la balise const correspond au mot-clé const
- La valeur de inout pour la propriété Kind d'un paramètre correspond au mot clé ref
- La valeur de out pour la propriété Kind d'un paramètre correspond au mot clé out
- Les classes partielles peuvent être modélisées comme deux classes distinctes avec la balise partielle

- La propriété Is Leaf d'une Classe correspond au mot clé scellé

Conventions C++

Enterprise Architect supporte l'ingénierie round du C++, y compris les extensions Managed C++ et C++/CLI, où ces conventions sont utilisées.

Stéréotypes

Stéréotype	S'applique à
énumération	Classe Correspond à : un type énuméré.
ami	Opération Correspond à : Le mot-clé « ami ».
propriété obtenir	Opération Correspond à : Une propriété « lecture ».
ensemble de propriétés	Opération Correspond à : Une propriété « écriture ».
structurer	Classe Correspond à : Un type 'struct'.
typedef	Classe Correspond à : Une instruction 'typedef', où le parent est le nom du type d'origine.
alias	Classe Correspond à une déclaration 'Alias', où le parent est le nom du type d'origine.
syndicat	Classe Correspond à : Un type d’union.

Valeur Étiquetés

Étiqueter	S'applique à
afx_msg	Opération Correspond à : Le mot-clé afx_msg.
anonyme	Classe contenant également le typedef Valeur Étiquetée Correspond à : Le nom de cette Classe étant uniquement défini par l'instruction typedef.

Nom d'attribut	Opération avec une propriété de stéréotype get ou un ensemble de propriétés Correspond à : Le nom de la variable derrière cette propriété.
champ de bits	Attribut Correspond à : La taille, en bits, autorisée pour le stockage de cet attribut.
corpsEmplacement	Opération Correspond à : l'emplacement vers lequel le corps de la méthode est généré ; les valeurs attendues sont header, classDec ou classBody.
rappeler	Opération Correspond à : une référence à la macro CALLBACK.
constexpr	Attribut et fonctionnement Correspond à : Le mot-clé constexpr.
explicite	Opération Correspond à : Le mot-clé « explicite ».
initialiseur	Opération Correspond à : Une liste d'initialisation du constructeur.
en ligne	Attribut et fonctionnement Correspond à : le mot-clé 'inline' et la génération en ligne de la définition de variable membre et du corps de la méthode.
mutable	Attribut Correspond à : Le mot-clé 'mutable'.
portée	Classe avec énumération des stéréotypes Correspond à : le mot-clé 'class' ou 'struct'.
jette	Opération Correspond à : les exceptions levées par cette méthode.
typedef	Classe avec un stéréotype autre que « typedef » Correspond à : Cette classe étant définie dans une instruction 'typedef'.
typeSynonymes	Classe Correspond à : Le nom 'typedef' et/ou les champs de ce type.
volatil	Opération Correspond à : Le mot-clé 'volatile'.

Autres congrès

- Namespaces sont générés pour chaque Paquetage sous une racine d'espace de noms

- Les attributs By Reference correspondent à un pointeur vers le type spécifié
- La propriété Transient d'un attribut correspond au mot clé volatile
- La propriété Abstract d'un attribut correspond au mot clé virtual
- La propriété Const d'une opération correspond au mot-clé const, spécifiant un type de retour constant
- La propriété Is Query d'une opération correspond au mot clé const, spécifier la méthode ne modifie aucun champ
- La propriété Pure d'une opération correspond à une méthode virtuelle pure utilisant la syntaxe "= 0"
- La propriété Fixe d'un paramètre correspond au mot clé const

Conventions C++ gérées

Ces conventions sont utilisées pour les extensions gérées vers C++ antérieures à C++/CLI. Afin de configurer le système pour générer du C++ managé, vous devez modifier la version C++ dans les options C++.

Stéréotypes

Stéréotype	S'applique à
propriété	Opération Correspond à : Le mot-clé '__property'.
propriété obtenir	Opération Correspond à : le mot-clé '__property' et une propriété read.
ensemble de propriétés	Opération Correspond à : le mot-clé '_ _property' et une propriété 'write'.
référence	Classe Correspond à : Le mot-clé '__gc'.
valeur	Classe Correspond à : Le mot-clé '__value'.

Valeur Étiquetés

Étiqueter	S'applique à
Type géré	Classe avec référence stéréotypée, valeur ou énumération ; Interface Correspond à : Le mot clé utilisé dans la déclaration de ce type ; les valeurs attendues sont « class » ou « struct ».

Autres congrès

- Les balises typedef et anonymes du C++ natif ne sont pas prises en charge
- La propriété Pure d'une opération correspond au mot-clé __abstract

Conventions C++/CLI

Ces conventions sont utilisées pour modélisation des extensions C++/CLI vers C++. Afin de configurer le système pour générer du C++/CLI géré, vous devez modifier la version C++ dans les options C++.

Stéréotypes

Stéréotype	S'applique à
événement	Opération Description : définit un événement pour fournir l'accès au gestionnaire d'événements pour cette classe.
propriété	Opération, attribut Description : Il s'agit d'une propriété contenant éventuellement du code de lecture et d'écriture.
référence	Classe Description : Correspond au mot-clé 'ref class' ou 'ref struct'.
valeur	Classe Description : Correspond au mot-clé ' valeur class' ou ' valeur struct'.

Valeur Étiquetés

Étiqueter	S'applique à
Nom d'attribut	Opération avec une propriété ou un événement stéréotypé Description : Le nom de la variable derrière cette propriété ou cet événement.
générique	Opération Description : Définit les paramètres génériques de cette opération.
Contraintes génériques	Classe ou interface modélisée, opération avec balise générique Description : Définit les contraintes sur les paramètres génériques de cette opération.
initial uniquement	Attribut Description : Correspond au mot-clé 'initonly'.
littéral	Attribut Description : Correspond au mot-clé littéral.
Type géré	Classe avec référence stéréotypée, valeur ou énumération ; Interface Description : correspond au mot-clé 'class' ou 'struct'.

Autres congrès

- Les balises typedef et anonyme ne sont pas utilisées
- Les stéréotypes de propriété get/property set ne sont pas utilisés
- La propriété Pure d'une opération correspond au mot clé abstract

Conventions de Delphes

Enterprise Architect supporte l'ingénierie round retour de Delphi, où ces conventions sont utilisées :

Stéréotypes

Stéréotype	S'applique à
constructeur	Opération Correspond à : Un constructeur.
destructeur	Opération Correspond à : Un destructeur.
interface d'affichage	Classe, Interface Correspond à : Une interface de répartition.
énumération	Classe Correspond à : un type énuméré.
métaclasse	Classe Correspond à : un type de métaclasse.
object	Classe Correspond à : un type object .
opérateur	Opération Correspond à : Un opérateur.
propriété obtenir	Opération Correspond à : Une propriété « lecture ».
ensemble de propriétés	Opération Correspond à : Une propriété « écriture ».
structurer	Classe Correspond à : un type d'enregistrement.

Valeur Étiquetés

Étiqueter	S'applique à
Nom d'attribut	Opération avec une propriété de stéréotype get ou un ensemble de propriétés Correspond à : Le nom de la variable derrière cette propriété.

surcharge	Opération Correspond à : le mot-clé « surcharge ».
passer outre	Opération Correspond à : Le mot-clé 'override'.
emballé	Classe Correspond à : Le mot-clé 'packed'.
propriété	Classe Correspond à : Une propriété ; voir *Propriétés Delphi* pour plus d'informations.
réintroduire	Opération Correspond à : le mot-clé « réintroduire ».

Autres congrès

- La propriété Statique d'un attribut ou d'une opération correspond au mot-clé 'class'
- La propriété Fixe d'un paramètre correspond au mot-clé 'const'
- La valeur de inout pour la propriété Kind d'un paramètre correspond au mot clé 'Var'
- La valeur de out pour la propriété Kind d'un paramètre correspond au mot clé 'Out'

Conventions Java

Enterprise Architect supporte l'ingénierie round retour de Java - y compris les extensions AspectJ - lorsque ces conventions sont utilisées.

Stéréotypes

Stéréotype	S'applique à
annotation	Interface Correspond à : un type d'annotation.
CompactConstructeur	Opération Correspond à : Un constructeur canonique compact pour l'enregistrement.
enregistrer	Classe Correspond à : un type d'enregistrement.
défaut	Opération Correspond à : le mot-clé « par défaut ».
énumération	Attributes au sein d'une énumération stéréotypée de classe Correspond à : Une option énumérée, distinguée des autres attributs qui n'ont aucun stéréotype.
énumération	Classe Correspond à : un type énuméré.
opérateur	Opération Correspond à : Un opérateur.
propriété obtenir	Opération Correspond à : Une propriété « lecture ».
ensemble de propriétés	Opération Correspond à : Une propriété « écriture ».
statique	Classe ou interface Correspond à : Le mot-clé 'static'.

Valeur Étiquetés

Étiqueter	S'applique à

annotations	Rien Correspond à : Les annotations sur la fonctionnalité du code actuel.
arguments	Attribut avec énumération de stéréotype Correspond à : les arguments qui s'appliquent à cette valeur énumérée.
Nom d'attribut	Opération avec une propriété de stéréotype get ou un ensemble de propriétés Correspond à : Le nom de la variable derrière cette propriété.
dynamique	Classe ou interface Correspond à : Le mot-clé « dynamique ».
générique	Opération Correspond à : Les paramètres génériques de cette opération.
non scellé	Classe, Interface Correspond à : Le mot-clé « non-scellé ».
permis	Classe, Interface Correspond à : l'expression « permis ».
liste de paramètres	Paramètre Correspond à : Une liste de paramètres avec la syntaxe ....
scellé	Classe, Interface Correspond à : Le mot-clé « scellé ».
En-tête d'enregistrement	<<enregistrer>>Classe Correspond à : l'en-tête d'enregistrement de la définition d'enregistrement.
jette	Opération Correspond à : les exceptions levées par cette méthode.
transitoire	Attribut Correspond à : le mot-clé « transient ».

Autres congrès

- Les instructions Paquetage sont générées lorsque le Paquetage actuel n'est pas une racine d'espace de noms
- La propriété Const d'un attribut ou d'une opération correspond au mot-clé final
- La propriété Transient d'un attribut correspond au mot clé volatile
- La propriété Fixe d'un paramètre correspond au mot-clé final

Conventions AspectJ

Ce sont les conventions utilisées pour prendre en charge les extensions AspectJ vers Java.

Stéréotypes

Stéréotype	S'applique à
conseil	Opération Correspond à : Un conseil dans un aspect AspectJ.
aspect	Classe Correspond à : Un aspect AspectJ.
point de coupe	Opération Correspond à : Un « pointcut » dans un aspect AspectJ.

Valeur Étiquetés

Étiqueter	S'applique à
nom du cours	Attribut ou opération au sein d'un aspect stéréotypé de Classe Correspond à : les classes auxquelles appartient ce membre intertype AspectJ.

Autres congrès

- Les spécifications d'un pointcut sont incluses dans le champ 'Comportement' de la méthode

Conventions PHP

Enterprise Architect supporte l'ingénierie round retour de PHP 4 et 5, où ces conventions sont utilisées.

Stéréotypes

Stéréotype	S'applique à
trait	Classe Correspond à : Un « trait ».
propriété obtenir	Opération Correspond à : Une propriété « lecture ».
ensemble de propriétés	Opération Correspond à : Une propriété « écriture ».

Valeur Étiquetés

Étiqueter	S'applique à
Nom d'attribut	Opération avec une propriété de stéréotype get ou un ensemble de propriétés Correspond à : Le nom de la variable derrière cette propriété.
final	Opérations en PHP 5 Correspond à : Le mot-clé « final ».

Conventions communes

- Un type non spécifié est modélisé comme var
- Méthodes renvoyant une référence sont générées en définissant le Type de retour sur var*
- Les paramètres de référence sont générés à partir de paramètres avec le paramètre Kind défini sur inout ou out

Conventions PHP 5

- Le modificateur Class final correspond à la propriété Is Leaf
- Le modificateur Abstract Class correspond à la propriété Abstract
- L'indication du type de paramètre est prise en charge en définissant le Type d'un paramètre.
- La valeur de inout ou out pour la propriété Kind d'un paramètre correspond à un paramètre de référence

Conventions Python

Enterprise Architect supporte l'ingénierie round retour de Python, où ces conventions sont utilisées.

Stéréotypes

Stéréotype	Correspond à
TypeAlias	Classe Correspond à : Alias Type explicite

Valeur Étiquetés

Étiqueter	S'applique à
asynchrone	Opération Correspond à : Le mot-clé 'async' dans la définition de la fonction.
Décorateurs	Classe, Opération Correspond à : Les décorateurs appliqués à cet élément dans la source.

Autres congrès

- Les membres Modèle avec une portée privée correspondent aux membres du code avec deux traits de soulignement en tête.
- Attributes ne sont générés que lorsque la valeur initiale n'est pas vide
- Tous les types font l'objet d'une ingénierie inverse en tant que var

Conventions SystemC

Enterprise Architect supporte l'ingénierie aller-retour de SystemC, où ces conventions sont utilisées.

Stéréotypes

Stéréotype	S'applique à
déléguer	Méthode Correspond à : Un délégué.
énumération	Classe intérieure Correspond à : un type d'énumération.
ami	Méthode Correspond à : Une méthode amie.
propriété	Méthode Correspond à : une définition de propriété.
sc_ctor	Méthode Correspond à : Un constructeur SystemC.
sc_module	Classe Correspond à : Un module SystemC.
port_sc	Attribut Correspond à : Un port.
sc_signal	Attribut Correspond à : Un signal.
structurer	Classe intérieure Correspond à : une structure ou une union.

Valeur Étiquetés

Étiqueter	S'applique à
gentil	Attribut (Port) Correspond à : Type de port (cadencé, fifo, maître, esclave, résolu, vectoriel).
mode	Attribut (Port) Correspond à : Mode port (in, out, inout).

remplacements	Méthode Correspond à : la liste d'héritage d'une déclaration de méthode.
lancer	Méthode Correspond à : la spécification d'exception d'une méthode.

Autres congrès

- SystemC hérite également de la plupart des stéréotypes et Valeur Étiquetés du C++

Pages de la boîte à outils SystemC

Pour modéliser une conception SystemC, faites glisser ces icônes sur un diagramme à partir de la page « Constructions SystemC » de la boîte à outils Diagramme .

Page	Icône
SystèmeC	Module Action : Définit un module SystemC. Un élément de classe stéréotypé sc_module.
Fonctionnalités SystemC	Port Action : définit un port SystemC. Un attribut stéréotypé sc_port.

Accéder

Ruban	Conception > Diagramme > Toolbox : > Spécifiez 'SystemC Constructs' dans les boîtes de dialogue ' Trouvez Item de Boîte à Outils '
Raccourcis Clavier	Ctrl+Shift+3 : > Spécifiez 'SystemC Constructs' dans la dialogue ' Trouvez Item de Boîte à Outils '
Autre	Vous pouvez afficher ou masquer la boîte à outils Diagramme en cliquant sur les icônes ou à l'extrémité gauche de la barre de légende en haut de la Vue Diagramme .

Conventions VB.NET

Enterprise Architect supporte l'ingénierie aller-retour de Visual Basic.NET, où ces conventions sont utilisées. Les versions antérieures de Visual Basic sont prises en charge dans un langage différent.

Stéréotypes

Stéréotype	S'applique à
événement	Opération Correspond à : Une déclaration d'événement.
importer	Opération Correspond à : Une opération à importer depuis une autre bibliothèque.
module	Classe Correspond à : Un module.
opérateur	Opération Correspond à : une définition de surcharge d'opérateur.
partiel	Opération Correspond à : Le mot-clé 'partial' sur une opération.
propriété	Opération Correspond à : une propriété contenant éventuellement du code de lecture et d'écriture.

Valeur Étiquetés

Étiqueter	S'applique à
Alias	Opération avec importation de stéréotypes Correspond à : l'alias de cette opération importée.
Nom d'attribut	Opération avec propriété de stéréotype Correspond à : le nom de la variable derrière cette propriété.
Jeu de caractères	Opération avec importation de stéréotypes Correspond à : la clause de jeu de caractères pour cette importation - l'une des valeurs "Ansi", "Unicode" ou "Auto".
déléguer	Opération Correspond à : le mot-clé « délégué ».

enumTag	Opération avec propriété de stéréotype Correspond à : le type de données sous lequel cette propriété est représentée.
Poignées	Opération Correspond à : la clause 'handles' sur cette opération.
Met en oeuvre	Opération Correspond à : la clause « implémente » sur cette opération.
Lib	Opération avec importation de stéréotypes Correspond à : la bibliothèque d'où provient cette importation.
DoitOverride	Opération Correspond à : le mot-clé « MustOverride ».
Rétrécissement	Opération avec opérateur stéréotype Correspond à : le mot-clé « Narrowing ».
Non remplaçable	Opération Correspond à : Le mot-clé 'NotOverrideable'.
Surcharges	Opération Correspond à : Le mot-clé 'surcharges'.
Remplacements	Opération Correspond à : Le mot-clé 'overrides'.
paramètreArray	Paramètre Correspond à : Une liste de paramètres utilisant le mot-clé 'ParamArray'.
partiel	Classe, Interface Correspond à : Le mot-clé 'partial'.
lecture seulement	Opération avec propriété de stéréotype Correspond à : Cette propriété définit uniquement le code « lu ».
ombres	Classe, interface, fonctionnement Correspond à : Le mot-clé 'Shadows'.
partagé	Attribut Correspond à : Le mot-clé « Partagé ».
Élargissement	Opération avec opérateur stéréotype Correspond à : Le mot-clé 'Élargissement'.
écriture seule	Opération avec propriété de stéréotype Correspond à : Cette propriété définit uniquement le code « écriture ».

Autres congrès

- Namespaces sont générés pour chaque Paquetage sous une racine d'espace de noms
- La propriété Is Leaf d'une Class correspond au mot clé NotInheritable
- La propriété Abstract d'une Class correspond au mot clé MustInherit
- La propriété Static d'un attribut ou d'une opération correspond au mot-clé Shared
- La propriété Abstract d'une opération correspond au mot clé MustOverride
- La valeur de in pour la propriété Kind d'un paramètre correspond au mot clé ByVal
- La valeur de inout ou out pour la propriété Kind d'un paramètre correspond au mot clé ByRef

Conventions Verilog

Enterprise Architect supporte l'ingénierie aller-retour de Verilog, où ces conventions sont utilisées.

Stéréotypes

Stéréotype	S'applique à
asynchrone	Méthode Correspond à : un processus simultané.
énumération	Classe intérieure Correspond à : un type d'énumération.
initialiseur	Méthode Correspond à : un processus d'initialisation.
module	Classe Correspond à : Un module.
partie	Attribut Correspond à : une instanciation de composant.
port	Attribut Correspond à : Un port.
synchrone	Méthode Correspond à : Un processus séquentiel.

Valeur Étiquetés

Étiqueter	S'applique à
gentil	Attribut (signal) Correspond à : le type de signal (tel que registre, bus).
mode	Attribut (Port) Correspond à : Le mode Port (in, out, inout).
Plan du port	Attribut (partie) Correspond à : la carte générique/port du composant instancié.
sensibilité	Méthode Correspond à : la liste de sensibilité d'un processus séquentiel.

taper	Attribut Correspond à : la plage ou valeur de type d'un attribut.

Pages de la boîte à outils Verilog

Accès : 'Conception > Diagramme > Boîte à outils : icône 'Hamburger' > HDL | Constructions Verilog

Faites glisser ces icônes sur un diagramme pour modéliser une conception Verilog.

Item	Action
Module	Définit un module Verilog. Un élément de classe stéréotypé par le module.
Énumération	Définit un Type énuméré. Un élément d'énumération.
Port	Définit un port Verilog. Un attribut stéréotypé par le port.
Partie	Définit une instanciation de composant Verilog. Un attribut en partie stéréotypé.
Attribut	Définit un attribut.
Procédure	Définit un processus Verilog : • Concurrent - Une méthode stéréotypée asynchrone • Séquentiel - Une méthode stéréotypée synchrone • Initializer - Une méthode stéréotypée d'initialisation

Conventions VHDL

Enterprise Architect supporte l'ingénierie aller-retour de VHDL, où ces conventions sont utilisées.

Stéréotypes

Stéréotype	S'applique à
architecture	Classe Correspond à : Une architecture.
asynchrone	Méthode Correspond à : un processus asynchrone.
configuration	Méthode Correspond à : Une configuration.
énumération	Classe intérieure Correspond à : un type énuméré.
entité	Interface Correspond à : Une entité.
partie	Attribut Correspond à : une instanciation de composant.
port	Attribut Correspond à : Un port.
signal	Attribut Correspond à : Une déclaration de signal.
structurer	Classe intérieure Correspond à : Une définition d'enregistrement.
synchrone	Méthode Correspond à : Un processus synchrone.
typedef	Classe intérieure Correspond à : une définition de type ou de sous-type.

Valeur Étiquetés

Étiqueter	S'applique à

estGénérique	Attribut (port) Correspond à : La déclaration 'port' dans une interface générique.
estSubType	Classe interne (typedef) Correspond à : une définition de sous-type.
gentil	Attribut (signal) Correspond à : le type de signal (tel que « registre », « bus »).
mode	Attribut (Port) Correspond à : Le mode Port (« in », « out », « inout », « buffer », « linkage »).
plan de port	Attribut (partie) Correspond à : la carte générique/port du composant instancié.
sensibilité	Méthode (synchrone) Correspond à : La liste 'sensibilité' d'un processus synchrone.
taper	Classe interne (typedef) Correspond à : L'indication 'type' d'une déclaration 'type'.
typeNameSpace	Attribut (partie) Correspond à : l'espace de noms 'type' du composant instancié.

Pages de la boîte à outils VHDL

Accéder

Pour modéliser une conception VHDL, faites glisser les icônes depuis les pages de la boîte à outils VHDL et déposez-les sur votre diagramme .

Ruban	Conception > Diagramme > Toolbox : > Spécifiez 'VHDL Constructs' dans la dialogue ' Trouvez Item de Boîte à Outils '
Raccourcis Clavier	Ctrl+Shift+3 : > Spécifiez 'VHDL Constructs' dans la dialogue ' Trouvez Item de Boîte à Outils '
Autre	Vous pouvez afficher ou masquer la boîte à outils Diagramme en cliquant sur les icônes ou à l'extrémité gauche de la barre de légende en haut de la Vue Diagramme .

Page de la boîte à outils VHDL

Item	Action
Architecture	Définit une architecture à associer à une entité VHDL. Un élément de classe stéréotypé par l'architecture.
Entité	Définit une entité VHDL pour contenir les définitions de port. Un élément d'interface stéréotypé par une entité.
Énumération	Définit un Type énuméré. Un élément d'énumération.
Structure	Définit un enregistrement VHDL. Un élément Class stéréotypé par la structure.
Typedef	Définit un type ou un sous-type VHDL. Un élément Class stéréotypé de typedef.

Page Boîte à Outils Fonctionnalités VHDL

Item	Action
Partie	Définit une instanciation de composant VHDL. Un attribut en partie stéréotypé.
Port	Définit un port VHDL. Un attribut stéréotypé par le port.
Signal	Définit un signal VHDL. Un attribut stéréotypé du signal.
Procédure	Définit un processus VHDL : • Concurrent - Une méthode stéréotypée asynchrone • Séquentiel - Une méthode stéréotypée synchrone • Configuration - Une méthode stéréotypée de configuration

Conventions Visual Basic

Enterprise Architect supporte l'ingénierie round retour de Visual Basic 5 et 6, où ces conventions sont utilisées. Visual Basic .NET est pris en charge comme langage différent.

Stéréotypes

Stéréotype	S'applique à
mondial	Attribut Correspond à : Le mot-clé 'Global'.
importer	Opération Correspond à : Une opération à importer depuis une autre bibliothèque.
propriété obtenir	Opération Correspond à : Une propriété « get ».
ensemble de propriétés	Opération Correspond à : Un « ensemble » de propriétés.
propriété louée	Opération Correspond à : Une propriété 'louer'.
avec des événements	Attribut Correspond à : Le mot-clé 'WithEvents'.

Valeur Étiquetés

Étiqueter	S'applique à
Alias	Opération avec importation de stéréotypes Correspond à : alias de cette opération importée.
Nom d'attribut	Opération avec la propriété de stéréotype get, la propriété set ou la propriété let Correspond à : le nom de la variable derrière cette propriété.
Lib	Opération avec importation de stéréotypes Correspond à : la bibliothèque d'où provient cette importation.
Nouveau	Attribut Correspond à : Le mot-clé 'nouveau'.

Autres congrès

- La valeur de in pour la propriété Kind d'un paramètre correspond au mot clé ByVal
- La valeur de inout ou out pour la propriété Kind d'un paramètre correspond au mot clé ByRef

Options linguistiques

Vous pouvez configurer diverses options sur la manière dont Enterprise Architect gère un langage particulier lors de la génération et de la rétro-ingénierie du code. Ces options sont soit spécifiques à :

- Votre ID , pour tous les modèles ou
- Le modèle dans lequel ils sont définis, pour tous les utilisateurs

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > <nom de la langue> Paramètres > Modèle > Options > Ingénierie du code source > <nom de la langue>
Raccourcis Clavier	Ctrl+F9 (dialogue 'Préférences')

Langues prises en charge

Langue
Script Action
Ada 2012 (dans les éditions Unified et Ultimate d' Enterprise Architect)
ArcGIS
ANSI C
C#
C++
Delphes
Java
PHP
Python
SystèmeC
Verilog (éditions Unified et Ultimate)
VHDL (éditions Unified et Ultimate)

Visual Basic
Visual Basic .NET

Options ActionScript - Utilisateur

Si vous avez l'intention de générer du code ActionScript à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications ActionScript » de la dialogue « Préférences » pour :

- Spécifiez le répertoire source par défaut
- Spécifier l'éditeur pour le code ActionScript

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > ActionScript
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code ActionScript. Cochez cette case pour désactiver support du code ActionScript.
Options pour l'utilisateur actuel	Dans les champs « Répertoire source par défaut » et « Éditeur », cliquez sur le bouton ... et recherchez le répertoire source et l'éditeur de fichiers externe que vous utiliserez.

Notes

- Ces options s'appliquent à tous les modèles auxquels vous accédez

Options ActionScript - Modèle

Si vous avez l'intention de générer du code ActionScript à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications ActionScript » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez la version ActionScript par défaut à générer (AS2.0 ou AS3.0)
- Spécifier les extensions de fichiers par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > ActionScript

Possibilités

Option	Action
Options pour le modèle actuel	Type la version ActionScript par défaut et l'extension de fichier par défaut à appliquer lors de la génération du code source ActionScript.
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options Ada 2012 - Utilisateur

Si vous avez l'intention de générer du code Ada 2012 à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Ada » de la dialogue « Préférences » pour :

- Indiquez au processus d'ingénierie inverse si le nom de l'enregistrement balisé est le même que le nom Paquetage
- Informer le moteur du nom alternatif de l'enregistrement balisé à localiser
- Spécifiez si le moteur doit créer un type de référence pour l'enregistrement balisé (si aucun n'est défini)
- Fournissez le nom du type de référence à créer (la valeur par défaut est Ref)
- Spécifier le paramètre de référence d'un type Référence / Accès
- Dites au moteur d'ignorer le nom du paramètre de référence
- Indiquer le nom du paramètre de référence à localiser

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Ada
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code Ada 2012. Cochez cette case pour désactiver support du code Ada 2012.
Options pour l'utilisateur actuel	Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède.

Notes

- support d'Ada 2012 est disponible dans les éditions Unified et Ultimate d' Enterprise Architect

Options Ada 2012 - Modèle

Si vous avez l'intention de générer du code Ada 2012 à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Ada » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut et
- Spécifier les définitions de classe de collection pour les connecteurs d'association

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > Ada

Possibilités

Option	Action
Options pour le modèle actuel	Type l'extension de fichier par défaut à appliquer lors de la génération du code source Ada.
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles
- support d'Ada 2012 est disponible dans les éditions Unified et Ultimate d' Enterprise Architect

Options ArcGIS - Utilisateur

Si vous avez l'intention de générer du code ArcGIS à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « ArcGIS » de la dialogue « Préférences » pour :

- Spécifiez le répertoire source par défaut
- Spécifier l'éditeur pour le code ArcGIS

ArcGIS doit être activé dans la dialogue « MDG Technologies » (« Spécialiser > Technologies > Gérer la technologie ») pour que la page « ArcGIS » soit disponible.

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > ArcGIS
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code ArcGIS. Cochez cette case pour désactiver support du code ArcGIS.
Options pour l'utilisateur actuel	Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède.

Options ArcGIS - Modèle

Si vous avez l'intention de générer du code ArcGIS à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « ArcGIS » de la dialogue « Gérer les options Modèle » pour :

- Spécifier les extensions de fichiers par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > ArcGIS

Possibilités

Option	Action
Options pour le modèle actuel	Type l'extension de fichier par défaut à appliquer lors de la génération du code source ArcGIS.
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options C - Utilisateur

Si vous avez l'intention de générer du code C à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications C » de la dialogue « Préférences ».

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > C
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code C. Sélectionnez cette option pour désactiver support du code C.
Options pour l'utilisateur actuel	Dans les champs valeur , spécifiez les options qui s'appliquent sous votre propre ID utilisateur dans tous les modèles auxquels vous accédez : • Le type d'attribut par défaut à créer (corrigé comme int) • Si une constante #define est importée en tant qu'attribut dans le code C importé (si 'Programmation orientée Object ' est défini sur True sur la page 'Spécifications C' de la dialogue 'Gérer les options Modèle ') • S'il faut générer des commentaires pour les méthodes C dans la déclaration et procéder à l'ingénierie inverse des commentaires à partir de la déclaration • S'il faut générer des commentaires pour les méthodes C pour l'implémentation et procéder à l'ingénierie inverse des commentaires de l'implémentation • S'il faut mettre à jour les commentaires lors de la régénération du code à partir du modèle • S'il faut mettre à jour le fichier d'implémentation lors de la régénération du code à partir du modèle • L'emplacement du répertoire de code source par défaut (cliquez sur le bouton ...) • Les extensions de fichiers par défaut à lire lors de l'importation d'un répertoire de code C • L' Éditeur de Code à utiliser (cliquez sur le bouton ...) • Le chemin de recherche du fichier d'implémentation par rapport au chemin du fichier d'en-tête

Options C - Modèle

Si vous avez l'intention de générer du code C à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications C » de la dialogue « Gérer les options Modèle » pour :

- Spécifier les extensions de fichier par défaut (en-tête et source)
- Définir support de la programmation orientée Object
- Définir les options d'ingénierie Statemachine
- Spécifier les définitions de classe de collection pour les connecteurs d'association

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > C

Possibilités

Option	Action
Options pour le modèle actuel	Dans les champs valeur , spécifiez ces options : • Les extensions d'en-tête et de fichier source par défaut pour les fichiers de code • Support en charge de la programmation orientée Object ; si c'est vrai, alors définissez : - Le caractère délimiteur Namespace - Si le premier paramètre d'une opération est une référence de classe - Le style de référence des paramètres dans le code C généré - Le nom du paramètre de référence dans le code généré - Le nom du constructeur par défaut dans le code généré - Le nom du Destructeur par défaut dans le code généré
Ingénierie Statemachine	Dans les champs valeur , utilisez les flèches déroulantes pour définir les options sur True ou False ; ces options s'appliquent uniquement à la génération de code à partir des modèles Statemachine dans le modèle actuel : • 'Utiliser le nouveau Statemachine Gabarit ' - défini sur True pour utiliser les gabarits de génération de code d' Enterprise Architect Release 11 et versions ultérieures, défini sur False pour appliquer les gabarits hérités d'EASL • Générer du code Trace - défini sur True pour générer du code Trace, False pour l'omettre
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options C# - Utilisateur

Si vous avez l'intention de générer du code C# à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications C# » de la dialogue « Préférences ».

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > C#
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code C# . Cochez cette case pour désactiver support du code C# .
Options pour l'utilisateur actuel	Dans les champs valeur , spécifiez les options qui s'appliquent sous votre propre ID utilisateur dans tous les modèles auxquels vous accédez : • Le type d'attribut par défaut à créer • Si Namespaces doivent être générés lors de la génération de classes C# • S'il faut supprimer les nouvelles lignes (retours chariot durs) de la balise récapitulative lors de l'importation de commentaires de style XML.NET • S'il faut générer une méthode Finalizer lors de la génération de code pour une classe C# • S'il faut générer une méthode Dispose lors de la génération de code pour une classe C# • L'emplacement du répertoire de code source par défaut (cliquez sur le bouton ...) • L' Éditeur de Code à utiliser (cliquez sur le bouton ...)

Options C# - Modèle

Si vous avez l'intention de générer du code C# à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications C# » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut
- Indiquez des classes de collection supplémentaires - pour définir des classes de collection personnalisées, qui peuvent être de simples substitutions (telles que CArray<#TYPE#>) ou un mélange d'autres chaînes et substitutions (telles que Cmap<CString,LPCTSTR,#TYPE#*,#TYPE #*>); ces classes de collection sont définies par défaut :
  - Liste<#TYPE#>;Pile<#TYPE#>;File d'attente<#TYPE#>;
- Définir les options d'ingénierie Statemachine
- Spécifier les définitions de classe de collection pour les connecteurs d'association

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > C#

Possibilités

Option	Action
Options pour le modèle actuel	Type l'extension de fichier par défaut à appliquer lors de la génération du code source C# , ainsi qu'une liste de toutes les classes de collection supplémentaires que vous souhaitez définir.
Ingénierie Statemachine	Dans les champs valeur , utilisez les flèches déroulantes pour définir les options sur True ou False ; ces options s'appliquent uniquement à la génération de code à partir des modèles Statemachine dans le modèle actuel : • 'Utiliser le nouveau Statemachine Gabarit ' - défini sur True pour utiliser les gabarits de génération de code d' Enterprise Architect Release 11 et versions ultérieures, défini sur False pour appliquer les gabarits hérités d'EASL • ' Générer Trace Code' - défini sur True pour générer du code Trace, False pour l'omettre
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options C++ - Utilisateur

Si vous avez l'intention de générer du code C++ à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications C++ » de la dialogue « Préférences ».

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > C++
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code C++. Sélectionnez cette option pour désactiver support du code C++.
Options pour l'utilisateur actuel	Dans les champs valeur , spécifiez les options qui s'appliquent sous votre propre ID utilisateur dans tous les modèles auxquels vous accédez : • Le type d'attribut par défaut à créer • Si Namespaces doivent être générés lors de la génération de classes C++ • Quel style appliquer lors de la génération et du traitement des commentaires pour C++ • Qu'il s'agisse de générer des commentaires pour les méthodes C++ dans la déclaration ou de procéder à une ingénierie inverse des commentaires à partir de la déclaration • Qu'il s'agisse de générer des commentaires pour les méthodes C++ de l'implémentation ou de procéder à une ingénierie inverse des commentaires de l'implémentation • S'il faut mettre à jour les commentaires lors de la régénération du code à partir du modèle • S'il faut mettre à jour le fichier d'implémentation lors de la régénération du code à partir du modèle • L'emplacement du répertoire de code source par défaut (cliquez sur le bouton ...) • Les extensions de fichiers par défaut à lire lors de l'importation d'un répertoire de code C++ • L' Éditeur de Code à utiliser (cliquez sur le bouton ...) • Le chemin de recherche du fichier d'implémentation par rapport au chemin du fichier d'en-tête

Options C++ - Modèle

Si vous avez l'intention de générer du code C++ à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications C++ » de la dialogue « Gérer les options Modèle » pour :

- Indiquez la version de C++ à générer ; cela contrôle l'ensemble des gabarits utilisés et la façon dont les propriétés sont créées
- Spécifiez le type de référence par défaut utilisé lorsqu'un type est spécifié par référence
- Spécifiez les extensions de fichiers par défaut
- Spécifier les préfixes Get/Set par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association
- Définir des classes de collection supplémentaires - pour définir des classes de collection personnalisées, qui peuvent être de simples substitutions (telles que CArray<#TYPE#>) ou un mélange d'autres chaînes et substitutions (telles que Cmap<CString,LPCTSTR,#TYPE#*,#TYPE #*>); ces classes de collection sont définies par défaut :
  - CArray<#TYPE#>;CMap<CString,LPCTSTR,#TYPE#*,#TYPE#*>;
- Définir les options d'ingénierie Statemachine

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > C++

Possibilités

Option	Action
Options pour le modèle actuel	Dans les champs valeur , précisez les options qui affectent tous les utilisateurs du modèle courant : • La version de C++ que vous utilisez (qui détermine les gabarits à utiliser lors de la génération du code) • Le type de référence par défaut à utiliser lors de la création de propriétés pour les attributs C++ par référence • Les extensions d'en-tête et de fichier source par défaut pour les fichiers de code • Le préfixe « Obtenir » par défaut • Le préfixe « Set » par défaut • Les classes de collection supplémentaires
Options d'ingénierie Statemachine	Dans les champs valeur , utilisez les flèches déroulantes pour définir les options sur True ou False ; ces options s'appliquent uniquement à la génération de code à partir des modèles Statemachine dans le modèle actuel : • 'Utiliser le nouveau Statemachine Gabarit ' - défini sur True pour utiliser les gabarits de génération de code d' Enterprise Architect Release 11 et versions ultérieures, défini sur False pour appliquer les gabarits hérités d'EASL • ' Générer Trace Code' - défini sur True pour générer du code Trace, False pour l'omettre

Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options Delphi - Utilisateur

Si vous avez l'intention de générer du code Delphi à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications Delphi » de la dialogue « Préférences » pour :

- Définir le type d'attribut par défaut
- Indiquer un répertoire source par défaut
- Définir l'éditeur de code par défaut à utiliser pour modifier le code source Delphi

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Delphi
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code Delphi. Sélectionnez cette option pour désactiver support du code Delphi.
Options pour l'utilisateur actuel	Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède.

Options Delphi - Modèle

Si vous avez l'intention de générer du code Delphi à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications Delphi » de la dialogue « Gérer les options Modèle » pour :

- Spécifier les extensions de fichier par défaut (en-tête et source)
- Spécifier les définitions de classe de collection pour les connecteurs d'association

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > Delphi

Possibilités

Option	Action
Options pour le modèle actuel	Type l'extension de fichier par défaut à appliquer lors de la génération du code source Delphi.
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Propriétés Delphi

Enterprise Architect offre support complète des propriétés Delphi. Celles-ci sont implémentées sous le nom Valeur Étiquetés , avec un éditeur de propriétés spécialisé pour aider à créer et modifier les propriétés de classe. En utilisant l'option de menu contextuel de l'élément ' Fonctionnalité Visibilité', vous pouvez afficher le compartiment 'tags' qui contient les propriétés. Les classes Delphi importées avec des propriétés ont cette fonctionnalité automatiquement rendue visible pour votre commodité.

Activer manuellement l'éditeur de propriétés

- Dans la classe sélectionnée, définissez le langage de génération de code sur « Delphi »
- Cliquez-droit sur la Classe et sélectionnez 'Delphi Propriétés ' pour ouvrir l'éditeur

À l'aide de l'éditeur Delphi Propriétés , vous pouvez créer des propriétés rapidement et simplement ; à partir de là, vous pouvez :

- Modifiez le nom et la portée (seuls Public et Publié sont actuellement pris en charge)
- Changer le type de propriété (la liste déroulante inclut toutes les classes définies dans le projet)
- Définissez les informations de lecture et d'écriture (les listes déroulantes contiennent tous les attributs et opérations de la classe actuelle ; vous pouvez également saisir du texte libre)
- Définissez « Stocké » sur True ou False
- Définir les informations sur les outils
- Définir la valeur par défaut, si elle existe

Notes

- Lorsque vous utilisez la dialogue « Créer une propriété » à partir de l'écran « Attribut », le système génère une paire de fonctions Get et Set ainsi que la définition de propriété requise comme Valeur Étiquetés ; vous pouvez modifier manuellement ces Valeur Étiquetés si nécessaire
- Les propriétés publiques sont affichées avec un préfixe de symbole « + » et publiées avec un « ^ »
- Lors de la création d'une propriété dans la dialogue « Créer une implémentation de propriété » (accessible via la dialogue « Attributes »), vous pouvez définir la portée sur « Publié » si le type de propriété est Delphi.
- Seuls « Public » et « Publié » sont pris en charge
- Si vous modifiez le nom d'une propriété et transférez l'ingénieur, une nouvelle propriété est ajoutée, mais vous devez supprimer manuellement l'ancienne du fichier source.

Options Java - Utilisateur

Si vous avez l'intention de générer du code Java à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications Java » de la dialogue « Préférences ».

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Java
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code Java. Cochez cette case pour désactiver support du code Java.
Options pour l'utilisateur actuel	Dans les champs valeur , précisez les options qui s'appliquent sous votre propre ID utilisateur dans tous les modèles auxquels vous accédez ; le: • Type d'attribut par défaut à créer (sélectionner dans la liste déroulante) • Emplacement du répertoire de code source par défaut (cliquez sur le bouton ...) • Éditeur de Code à utiliser (cliquez sur le bouton ...)

Options Java - Modèle

Si vous avez l'intention de générer du code Java à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications Java » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut
- Spécifiez un préfixe « Obtenir » par défaut
- Spécifiez un préfixe « Set » par défaut
- Définir les options d'ingénierie Statemachine
- Spécifier les définitions de classe de collection pour les connecteurs d'association
- Définir des classes de collection supplémentaires - pour définir des classes de collection personnalisées, qui peuvent être de simples substitutions (telles que CArray<#TYPE#>) ou un mélange d'autres chaînes et substitutions (telles que Cmap<CString,LPCTSTR,#TYPE#*,#TYPE #*>); ces classes de collection sont définies par défaut :
  - HashSet<#TYPE#>;Map< String ,#TYPE#>;

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > Java

Possibilités

Option	Action
Options pour le modèle actuel	Dans les champs valeur , précisez les options qui affectent tous les utilisateurs du modèle courant ; le: • Extension de fichier par défaut pour les fichiers de code • Les préfixes Get et Set par défaut • Les classes de collection par défaut et supplémentaires
Ingénierie Statemachine	Dans les champs valeur , utilisez les flèches déroulantes pour définir les options sur True ou False ; ces options s'appliquent uniquement à la génération de code à partir des modèles Statemachine dans le modèle actuel : • 'Utiliser le nouveau Statemachine Gabarit ' - défini sur True pour utiliser les gabarits de génération de code d' Enterprise Architect Release 11 et versions ultérieures, défini sur False pour appliquer les gabarits hérités d'EASL • ' Générer Trace Code' - défini sur True pour générer du code Trace, False pour l'omettre
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options MySQL - Utilisateur

Si vous avez l'intention de générer du code MySQL à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « MySQL » de la dialogue « Préférences » pour :

- Spécifier un type d'attribut par défaut
- Spécifier un répertoire source par défaut
- Spécifier les extensions de nom de fichier pour les fichiers à importer
- Spécifier un éditeur pour modifier le code
- Spécifier un propriétaire par défaut

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > MySQL
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code MySQL. Sélectionnez cette option pour désactiver support du code MySQL.
Options pour l'utilisateur actuel	Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède.

Options MySQL - Modèle

Si vous avez l'intention de générer du code MySQL à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « MySQL » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > MySQL

Possibilités

Option	Action
Options pour le modèle actuel	Type l'extension de fichier par défaut à appliquer lors de la génération du code source MySQL.
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options PHP - Utilisateur

Si vous avez l'intention de générer du code PHP à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications PHP » de la dialogue « Préférences » pour :

- Définir une liste d'extensions séparées par des points-virgules à examiner lors d'une importation de code de répertoire pour PHP
- Définir un répertoire par défaut pour ouvrir et enregistrer le code source PHP
- Spécifiez l'éditeur par défaut à utiliser lors de l'édition du code PHP

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > PHP
Raccourcis Clavier	Ctrl+F9 | Ingénierie du code source | PHP

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code PHP. Sélectionnez cette option pour désactiver support du code PHP.
Options pour l'utilisateur actuel	Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède.

Options PHP - Modèle

Si vous avez l'intention de générer du code PHP à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications PHP » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez la version PHP par défaut à générer
- Définir l'extension de fichier par défaut
- Spécifiez un préfixe « Obtenir » par défaut
- Spécifiez un préfixe « Set » par défaut

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > PHP

Possibilités

Option	Action
Options pour le modèle actuel	Type la version PHP par défaut, l'extension de fichier par défaut à appliquer lors de la génération du code source PHP et les préfixes par défaut « Get » et « Set ».

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options Python - Utilisateur

Si vous avez l'intention de générer du code Python à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications Python » de la dialogue « Préférences » pour :

- Spécifiez le répertoire source par défaut à utiliser
- Spécifiez l'éditeur par défaut utilisé pour écrire et modifier le code Python

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Python
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code Python. Sélectionnez cette option pour désactiver support du code Python.
Options pour l'utilisateur actuel	Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède.

Options Python - Modèle

Si vous avez l'intention de générer du code Python à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications Python » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > Python

Possibilités

Option	Action
Options pour le modèle actuel	Type l'extension de fichier par défaut à appliquer lors de la génération du code source Python.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options SystemC - Utilisateur

Si vous avez l'intention de générer du code SystemC à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « SystemC » de la dialogue « Préférences » pour :

- Spécifier un répertoire source par défaut
- Spécifier un éditeur pour modifier le code

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > SystemC
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code SystemC. Sélectionnez cette option pour désactiver support du code SystemC.
Options pour l'utilisateur actuel	Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède.

Options SystemC - Modèle

Si vous avez l'intention de générer du code SystemC à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « SystemC » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > SystemC

Possibilités

Option	Action
Options pour le modèle actuel	Type l'extension de fichier par défaut à appliquer lors de la génération du code source SystemC.
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options Teradata - Utilisateur

Si vous avez l'intention de générer du code Teradata à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Teradata » de la dialogue « Préférences » pour :

- Spécifier un type d'attribut par défaut
- Spécifier un répertoire source par défaut
- Spécifier un éditeur pour modifier le code

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Teradata
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code Teradata. Sélectionnez cette option pour désactiver support du code Teradata.
Options pour l'utilisateur actuel	Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède.

Options Teradata - Modèle

Si vous avez l'intention de générer du code Teradata à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Teradata » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > Teradata

Possibilités

Option	Action
Options pour le modèle actuel	Type l'extension de fichier par défaut à appliquer lors de la génération du code source Teradata.
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options VB.NET - Utilisateur

Si vous avez l'intention de générer du code VB.NET à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications VB.NET » de la dialogue « Préférences » pour :

- Spécifiez le type d'attribut par défaut
- Indiquer s'il faut générer des espaces de noms
- Spécifier un répertoire source par défaut
- Spécifier un éditeur pour modifier le code

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > VB.Net
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code VB.NET. Sélectionnez cette option pour désactiver support du code VB.NET.
Options pour l'utilisateur actuel	Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède.

Options VB.NET - Modèle

Si vous avez l'intention de générer du code VB.NET à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications VB.Net » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > VB.Net

Possibilités

Option	Action
Options pour le modèle actuel	Type l'extension de fichier par défaut à appliquer lors de la génération du code source VB.Net.
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options Verilog - Utilisateur

Si vous avez l'intention de générer du code Verilog à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Verilog » de la dialogue « Préférences » pour :

- Spécifier un répertoire source par défaut
- Spécifier un éditeur pour modifier le code

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Verilog
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code Verilog. Sélectionnez cette option pour désactiver support du code Verilog.
Options pour l'utilisateur actuel	Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède.

Options Verilog - Modèle

Si vous avez l'intention de générer du code Verilog à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Verilog » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > Verilog

Possibilités

Option	Action
Options pour le modèle actuel	Type l'extension de fichier par défaut à appliquer lors de la génération du code source Verilog.
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options VHDL - Utilisateur

Si vous avez l'intention de générer du code VHDL à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page 'VHDL' de la dialogue 'Préférences' pour :

- Spécifier un répertoire source par défaut
- Spécifier un éditeur pour modifier le code

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > VHDL
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code VHDL. Sélectionnez cette option pour désactiver support du code VHDL.
Options pour l'utilisateur actuel	Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède.

Options VHDL - Modèle

Si vous avez l'intention de générer du code VHDL à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page 'VHDL' de la dialogue 'Gérer les options Modèle ' pour :

- Spécifiez l'extension de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > VHDL

Possibilités

Option	Action
Options pour le modèle actuel	Type l'extension de fichier par défaut à appliquer lors de la génération du code source VHDL.
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options Visual Basic - Utilisateur

Si vous avez l'intention de générer du code Visual Basic à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications VB » de la dialogue « Préférences » pour :

- Spécifiez le type d'attribut par défaut
- Définir le répertoire source par défaut
- Définir les extensions de fichiers pour rechercher les fichiers de code à importer
- Définir l'éditeur par défaut à utiliser pour éditer le code source

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Visual Basic
Raccourcis Clavier	Ctrl+F9

Possibilités

Option	Action
Désactiver la langue	Laissez cette case décochée pour support la génération de code Visual Basic. Sélectionnez cette option pour désactiver support du code Visual Basic.
Options pour l'utilisateur actuel	Spécifie les options utilisées pour l'utilisation actuelle ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède.

Options Visual Basic - Modèle

Si vous avez l'intention de générer du code Visual Basic à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications VB » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez la version par défaut de Visual Basic à générer
- Indiquer l'extension de fichier par défaut lors de la lecture/écriture
- Indiquer le mode de transaction Microsoft Transaction Server (MTS) pour les objets MTS
- Spécifiez si une classe utilise Multi use (Vrai ou Faux)
- Spécifiez si une classe utilise la propriété Persistable
- Indiquer les comportements de liaison de données et de source de données
- Définir l'espace de noms global
- Définir l'attribut Exposé
- Indiquez si l'attribut Createable est True ou False
- Spécifier les définitions de classe de collection pour les connecteurs d'association

Accéder

Ruban	Paramètres > Modèle > Options > Ingénierie du code source > Visual Basic

Possibilités

Option	Action
Options pour le modèle actuel	Type l'extension de fichier par défaut à appliquer lors de la génération du code source Visual Basic, puis cliquez sur la flèche déroulante dans chacun des autres champs et sélectionnez la valeur appropriée.
Cours de collecte	Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association.

Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, ils ne s'appliquent pas aux autres modèles

Options linguistiques MDG Technologie

Si vous avez chargé une MDG Technologie qui spécifie un module de code dans votre dossier *Sparx Systems > EA > MDG Technologies* , le langage est inclus dans la liste « Source Code Engineering » de la dialogue « Préférences ». La langue n'est répertoriée dans la dialogue 'Préférences' que si un fichier MDG Technologie l'utilise réellement dans votre modèle.

Accéder

Ruban	Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > MDG
Raccourcis Clavier	Ctrl+F9

Possibilités

Champ	Action
Extension par défaut	Extension par défaut pour les fichiers sources générés ; affiché si l'option est dans la technologie. Ceci est enregistré par projet.
Importer des extensions de fichiers	Dossier par défaut à partir duquel importer les fichiers source ; affiché si la technologie supporte les espaces de noms. Ceci est enregistré une fois pour tous les projets.
Générer Namespaces	Indique si les espaces de noms sont générés ou non.
Répertoire source par défaut	Le répertoire par défaut pour enregistrer les fichiers source générés. Ceci est toujours affiché.
Éditeur	Indique l'éditeur utilisé pour modifier les fichiers source.
Type att	Indique le type d'attribut par défaut.

Notes

- Ces options sont définies dans la technologie à l'intérieur de la balise <CodeOptions> d'un module de code, comme indiqué :
  <CodeOption nom="DefaultExtension">.rb</CodeOption>

Options de réinitialisation

Enterprise Architect stocke certaines des options d'une classe lors de sa première création. Certains sont mondiaux ; par exemple, $LinkClass est stocké lorsque vous créez la classe pour la première fois, donc dans les classes existantes, la modification globale dans la dialogue « Préférences » ne sera pas automatiquement prise en compte. Vous devez modifier les options de la classe existante.

Modifier les options pour une seule classe

Étape	Action
1	Cliquez sur la classe à modifier et sélectionnez l'option de ruban 'Développer > Code source > Générer > Générer un seul élément'. La dialogue « Générer Code » s'affiche.
2	Cliquez sur le bouton Avancé. La dialogue « Options Object » s'affiche.
3	Cliquez sur l'option ' Attributes /Opérations'.
4	Modifiez les options et cliquez sur le bouton Fermer pour appliquer les modifications.

Modifier les options pour toutes les classes d'un Paquetage

Étape	Action
1	Cliquez sur le Paquetage dans la fenêtre Navigateur et sélectionnez l'option de ruban "Développer > Préférences > Options > Réinitialiser la langue source". La dialogue « Gérer la génération de code » s'affiche.
2	Dans le champ « Où se trouve la langue : », cliquez sur la flèche déroulante et sélectionnez la langue à partir de laquelle vous souhaitez changer.
3	Dans le champ « Convertir en : », cliquez sur la flèche déroulante et sélectionnez la langue vers laquelle vous souhaitez changer.
4	Cochez la case en regard de chaque option à appliquer aux éléments de classe modifiés dans le Paquetage : • Effacer Noms de fichiers des fichiers pour générer du code • Réinitialiser les options par défaut sur chaque classe • Traiter Paquetages enfants sous le Paquetage sélectionné
5	Cliquez sur le bouton OK pour appliquer les modifications.

Définir les classes de collecte

À l'aide Enterprise Architect , vous pouvez définir des classes de collection pour générer du code à partir de connecteurs d'association où le rôle cible a un paramètre de multiplicité supérieur à 1.

Tâches

Tâche	Détail
Définir des classes de collection	Dans la section 'Ingénierie du code source' de la dialogue 'Gérer les options Modèle ' (sélectionnez l'option de ruban 'Paramètres > Modèle > Options > Ingénierie du code source'), sur chaque page de langue, cliquez sur le bouton Classes de collection. La dialogue « Classes de collection pour les rôles d'association » s'affiche. Sur ce dialogue , vous pouvez définir : • La classe de collection par défaut pour les rôles 1..* • La classe de collection ordonnée à utiliser pour les rôles 1..* • La classe de collection qualifiée à utiliser pour les rôles 1..*
Définir des classes de collection pour une classe spécifique	Les classes de collection spécifiques à une classe peuvent être définies en cliquant sur le bouton Classes de collection dans la dialogue Classe ' Propriétés ' de l'élément.
Priorité de génération de code	Lorsque Enterprise Architect génère du code pour un connecteur qui a un rôle de multiplicité >1 : 1. Si le qualificatif est défini, utilisez la collection qualifiée : - pour la Classe si défini - sinon, utilisez la collection qualifiée en langage de code 2. Si l'option « Commander » est définie, utilisez la collection ordonnée : - pour la Classe si défini - sinon, utilisez la collection ordonnée du langage de code 3. Sinon, utilisez la collection par défaut : - pour la Classe si défini - sinon, utilisez la collection par défaut du langage de code
Utiliser des marqueurs	Vous pouvez inclure le marqueur #TYPE# dans le nom de la collection ; Enterprise Architect le remplace par le nom de la classe collectée au moment de la génération de la source (par exemple, Vector<#TYPE#> deviendrait Vector<foo>). À l'inverse, lors de l'ingénierie inverse, un connecteur Association est également créé si une entrée correspondante (par exemple, foo si foo est trouvé dans le modèle) est définie comme classe de collection.
Classes de collecte supplémentaires	Des classes de collection supplémentaires peuvent être définies dans les pages d'options de langage spécifiques au modèle pour C# , C++ et Java.
Type de membre	Dans l'onglet 'Rôle(s)' de la boîte dialogue ' Propriétés ' de l'Association (accessible depuis le menu contextuel cliquez-droit de n'importe quelle Association), il y a un champ ' Type de Membre' pour chacun des Rôles Source et Cible. Si vous définissez cela, la valeur que vous entrez remplace toutes les options

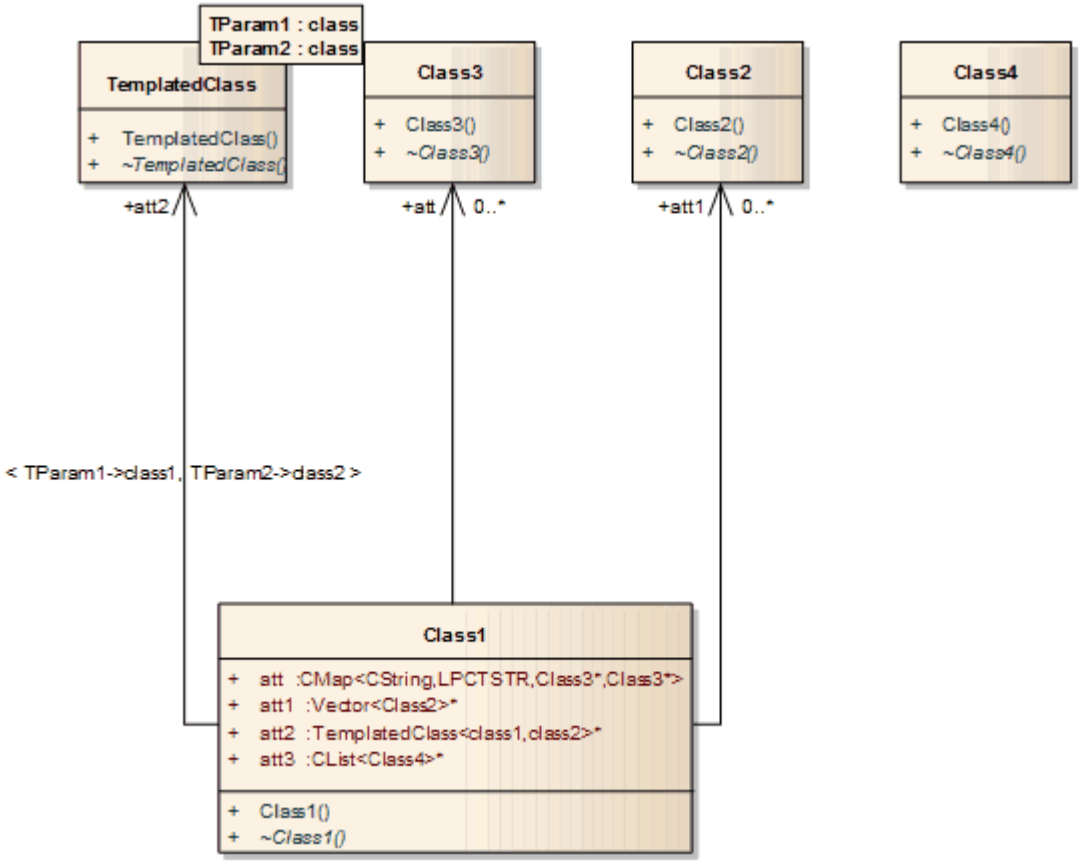
	répertoriées.

Exemple d'utilisation de classes de collection

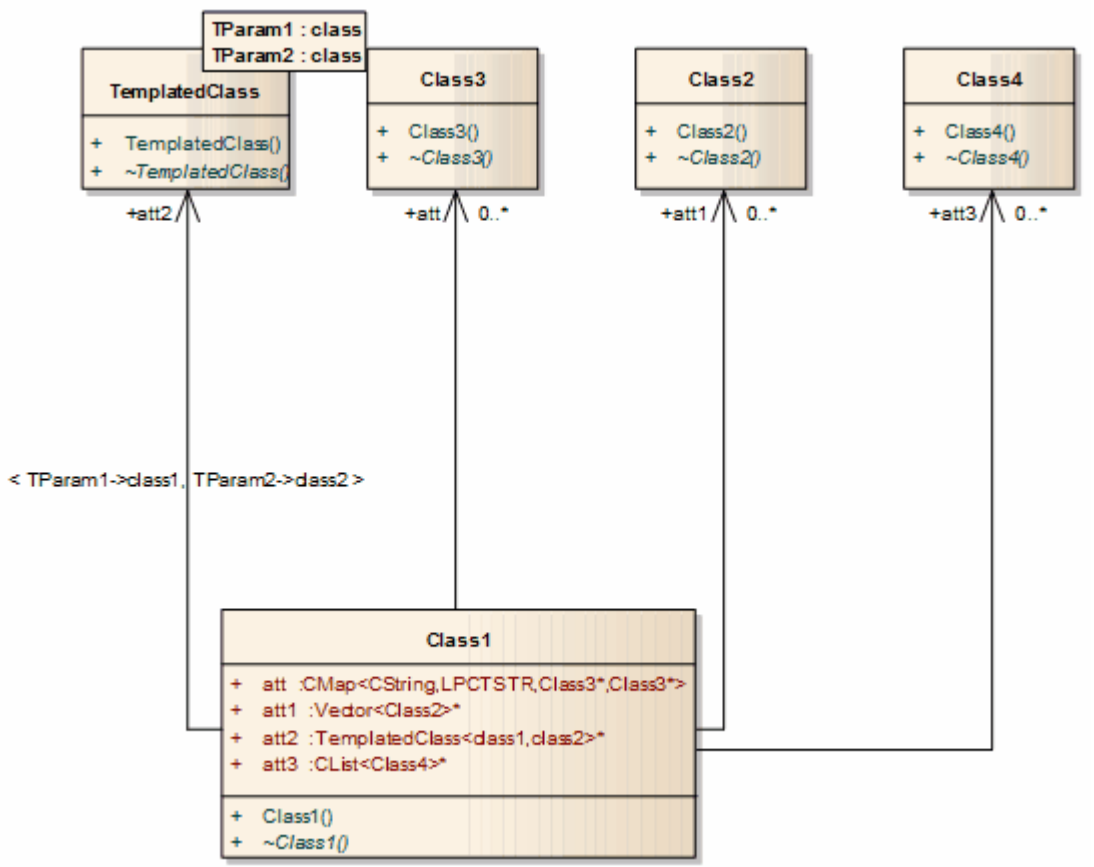
Considérez ce code source :

```
classe Classe1
{
publique:
Classe1();
virtuel ~Class1();
CMap<CString,LPCTSTR,Class3*,Class3*> att;
Vecteur<Class2> *att1;
TemplatedClass<class1,class2> *att2;
CListe<Class4> *att3;
} ;
classe Classe2
{
publique:
Classe2();
virtuel ~Class2();
} ;
classe Classe3
{
publique:
Classe3();
virtuel ~Class3();
} ;
classe Classe4
{
publique:
Classe4();
virtuel ~Class4();
} ;
modèle<classe TParam1, classe TParam2>
classe TemplatedClass
{
publique:
ClasseModèle() {
}
virtuel ~TemplatedClass() {
}
} ;
```

Si ce code est importé dans le système avec les options d'importation par défaut, ce diagramme est généré :



Toutefois, si vous entrez la valeur 'CList<#Type#>' dans le champ 'Classes de collection supplémentaires' de la page des options de langage spécifiques au modèle (C# , Java, C++), un connecteur d'association est également créé pour la classe 4 :


TParam1 : class
TParam2 : class
TemplatedClass
+ TemplatedClass()
+ ~TemplatedClass()
Class3
+ Class3()
+ ~Class3()
Class2
+ Class2()
+ ~Class2()
Class4
+ Class4()
+ ~Class4()
+att2
+att
0..*
+att1
0..*
+att3
0..*
< TParam1->class1, TParam2->class2 >
Class1
+ att :CMap<CString,LPCTSTR,Class3*,Class3*>
+ att1 :Vector<Class2>*
+ att2 :TemplatedClass<class1,class2>*
+ att3 :CList<Class4>*
+ Class1()
+ ~Class1()

Chemins locaux

Lorsqu'une équipe de développeurs travaille sur le même modèle Enterprise Architect , chaque développeur peut stocker sa version du code source dans son système de fichiers local, mais pas toujours au même emplacement que ses collègues développeurs. Pour gérer ce scénario dans Enterprise Architect , vous pouvez définir des chemins locaux pour chaque utilisateur, dans la dialogue 'Chemins locaux'.

Vous pouvez utiliser des chemins locaux pour générer du code et de l'ingénierie inverse, et dans Contrôle de Version , développer des schémas XML et générer des documents et des rapports Web.

La mise en place des chemins locaux peut prendre un peu de temps, mais si vous souhaitez travailler en collaboration sur la source et le modèle simultanément, l'effort en vaut la peine.

Par exemple, si :

- Le développeur A stocke ses fichiers .java dans un répertoire C:\Java\Source, tandis que le développeur B stocke les siens dans D:\Source, et
- Les deux développeurs souhaitent générer et procéder à l'ingénierie inverse dans le même modèle Enterprise Architect situé sur un lecteur réseau partagé (ou répliqué).

Le développeur A peut définir un chemin local de :

JAVA_SOURCE = "C:\Java\Source"

Toutes les classes générées et stockées dans le projet Enterprise Architect sont stockées sous :

%JAVA_SOURCE%\<xxx.java>

Le développeur B définit un chemin local comme :

JAVA_SOURCE ="D:\Source"

Désormais, Enterprise Architect stocke tous les fichiers Java dans ces répertoires comme :

%JAVA_SOURCE%\<nom de fichier>

Sur la machine de chaque développeur, le nom de fichier est étendu à la version locale correcte.

Accéder

Ruban	Développer > Code source > Options > Configurer les chemins locaux

Dialogue sur les sentiers locaux

À l'aide de la dialogue « Chemins locaux », vous pouvez configurer des chemins locaux pour un seul utilisateur sur une machine particulière. Pour une description de l'utilisation des chemins locaux, consultez la rubrique *Chemins locaux* .

Accéder

Ruban	Développer > Code source > Options > Configurer les chemins locaux

Possibilités

Option	Action
Chemin	Type ou dans le navigateur le chemin du répertoire local dans le système de fichiers (par exemple, d:\java\source).
ID	Type l' ID partagé qui remplace le chemin local (par exemple, JAVA_SRC).
Type	Cliquez sur la flèche déroulante et sélectionnez le type de chemin auquel appliquer (par exemple, Java).
Chemins relatifs	Répertorie les chemins actuellement définis pour le modèle, par défaut le plus récent en haut. Si vous souhaitez modifier la séquence des chemins dans la liste, cliquez sur un chemin et utilisez les boutons pour déplacer le chemin vers le haut ou vers le bas d'une position dans la liste.
Appliquer le chemin	Cliquez sur un chemin dans la liste « Chemins relatifs » et cliquez sur ce bouton pour mettre à jour tous les noms de chemin complets existants dans le modèle avec le nom de chemin relatif partagé. Par exemple: d:\java\source\main.java pourrait devenir %JAVA_SRC%\main.java
Développer le chemin	Cliquez sur un chemin dans la liste 'Chemins relatifs' et cliquez sur ce bouton pour supprimer le chemin relatif et remplacer le nom du chemin complet (l'effet inverse du bouton Appliquer le chemin).
Nouveau	Cliquez sur ce bouton pour effacer les champs de données afin de pouvoir définir un autre chemin local.
Sauvegarder	Lorsque vous avez défini un chemin local, cliquez sur ce bouton pour l'enregistrer et l'ajouter à la liste 'Chemins relatifs'.
Supprimer	Cliquez sur un chemin dans la liste « Chemins relatifs » et cliquez sur ce bouton pour supprimer complètement le chemin de la liste.
Fermer	Cliquez sur ce bouton pour fermer le dialogue et enregistrer les modifications

	apportées à la liste.

Notes

- Vous pouvez également configurer un lien hypertexte (pour une commande Enterprise Architect) sur un diagramme pour accéder à la dialogue « Chemins locaux », pour changer, mettre à jour ou développer votre chemin local actuel.
- Si le fait de développer ou d'appliquer un chemin pour un fichier lié crée un enregistrement en double, le processus ignorera cet enregistrement et affichera un message à la fin du processus.

Macros de langue

Lors de l'ingénierie inverse d'un langage tel que C++, vous pouvez trouver des directives de préprocesseur dispersées dans le code. Cela peut faciliter la gestion du code, mais peut gêner l'analyse du langage C++ sous-jacent.

Pour remédier à ce problème, vous pouvez inclure un nombre illimité de définitions de macro, qui sont ignorées lors de la phase d'analyse de la rétro-ingénierie. Il est toujours préférable, si vous disposez de la facilité , de prétraiter d'abord le code en utilisant le compilateur approprié ; de cette façon, les définitions et définitions de macros complexes sont développées et peuvent être facilement analysées. Si vous ne disposez pas de cette facilité , cette option constitue un substitut pratique.

Accéder

Ruban	Paramètres > Données de référence > Paramètres > Macros du préprocesseur ou Développer > Code source > Options > Configurer > Définir les macros du préprocesseur

Définir une macro

Étape	Action
1	Sélectionnez l'option de menu « Macros du préprocesseur ». La dialogue « Macros de langue » s'affiche.
2	Cliquez sur le bouton Ajouter un nouveau.
3	Entrez les détails de votre macro.
4	Cliquez sur le bouton OK .

Macros intégrées dans les déclarations

Les macros sont parfois utilisées dans la déclaration des classes et des opérations, comme dans ces exemples :

```
classe __declspec Foo
{
int __declspec Bar( int p);
} ;
```

Si declspec est défini comme une macro C++, comme indiqué, la classe et l'opération importées contiennent une Valeur Étiquetée appelée DeclMacro1 avec valeur __declspec (les macros suivantes seraient définies comme DeclMacro2, DeclMacro3 et ainsi de suite).

Lors de l'ingénierie avancée, ces Valeur Étiquetés sont utilisées pour régénérer les macros dans le code.

Définir des macros complexes

Il est parfois utile de définir des règles pour des macros complexes pouvant s'étendre sur plusieurs lignes ; Enterprise Architect ignore toute la section de code définie par la règle.

De telles macros peuvent être définies dans Enterprise Architect comme dans ces deux exemples ; les deux types peuvent être combinés dans une seule définition.

Bloc les macros

BEGIN_INTERFACE_PART ^ END_INTERFACE_PART

Le symbole ^ représente le corps de la macro - cela permet de passer d'une macro à une autre ; les espaces entourant le symbole ^ sont obligatoires.

Macros de fonctions

RTTI_EMULATION()

Enterprise Architect ignore le jeton, y compris tout ce qui se trouve entre parenthèses.

Les macros de fonction peuvent également inclure le corps de la fonction :

RTTI_EMULATION() {}

Dans ce cas, Enterprise Architect ignore le jeton, y compris tout ce qui se trouve entre parenthèses et entre accolades. Note que si la macro de fonction inclut le corps de la fonction, elle ne peut pas être combinée avec une macro Bloc .

Notes

- Vous pouvez transporter ces définitions de macro de langage (ou macro de préprocesseur) entre modèles, en utilisant les options « Paramètres > Modèle > Transférer > Exporter les données de référence » et « Importer les données de référence » ; les macros sont exportées sous forme de liste de macros

Développement de langages de programmation

Vous pouvez utiliser une gamme de langages de programmation établis dans Enterprise Architect , mais si ceux-ci ne conviennent pas à vos besoins, vous pouvez développer le vôtre. Vous l'appliqueriez ensuite à vos modèles via une MDG Technologie que vous pourriez développer uniquement dans ce but ou à des fins plus larges. Après avoir développé le langage, vous pouvez également écrire gabarits de transformation MDA pour convertir un Modèle indépendant de la plateforme ou un modèle dans un autre langage en un modèle pour votre nouveau langage, ou vice versa.

Accéder

Ruban	Développer > Code source > Options > Modifier le code Gabarits
Raccourcis Clavier	Ctrl+Maj+P

Développer un langage de programmation

Étape	Description
1	Dans l'éditeur Code Gabarit , cliquez sur le bouton Nouvelle langue et, dans la dialogue 'Types de données des langages de programmation', cliquez sur le bouton Ajouter un produit. Entrez le nom de votre nouveau langage de programmation et définissez les types de données correspondants. Vous ne pouvez pas accéder à la nouvelle langue dans l'éditeur de Code Gabarit tant qu'au moins un type de données n'a pas été ajouté à la langue.
2	Après avoir défini tous les types de données dont vous avez besoin, cliquez sur le bouton Fermer, sélectionnez la langue dans le champ « Langue » de l'éditeur de Code Gabarit et commencez à éditer ou à créer les gabarits de code pour la nouvelle langue. Les gabarits du code définissent comment le système doit fonctionner : • Ingénierie du code avancé de vos modèles dans le nouveau langage • Génération de code Comportementale (si cela est approprié)
3	Si vous préférez, vous pouvez également définir des options de code source pour votre nouvelle langue. Il s'agit de paramètres supplémentaires pour le langage qui ne sont pas fournis par les types de données ou gabarits de code et qui aident à définir la manière dont le système gère ce langage lors de la génération et de la rétro-ingénierie du code. Les options de codes sont mises à disposition de vos modèles uniquement via une MDG Technologie .
4	Définir une grammaire pour votre langue est une étape facultative qui offre deux avantages principaux : • Ingénierie inverse du code existant dans votre modèle • Synchronisation lors de la génération du code afin que les modifications apportées au fichier depuis sa dernière génération ne soient pas perdues. Pour accéder à l'éditeur de grammaire, sélectionnez l'option du ruban « Développer > Code source > Éditeur de grammaire ».
5	Si vous souhaitez que des transformations MDA soient effectuées vers (ou depuis) votre nouveau langage de programmation, vous pouvez également éditer et créer gabarits de transformation pour celui-ci. Le

	processus de création gabarits de transformation est très similaire à celui de création gabarits de code.
6	Après avoir créé les types de données, gabarits de code, les options de code, gabarits de grammaire et de transformation pour votre nouveau langage, vous pouvez les incorporer et les distribuer dans une MDG Technologie .

Cadre de code Gabarit

Lorsque vous utilisez Enterprise Architect pour générer du code à partir d'un modèle ou transformer le modèle, le système fait référence au Code Gabarit Framework (CTF) pour les paramètres qui définissent comment il doit :

- Concevoir un modèle UML
- Générer du code Comportementale
- Effectuer une transformation Model Driven Architecture (MDA)
- Générer du DDL en modélisation de base de données

Une gamme de gabarits standards est disponible pour la génération directe de code et pour la transformation ; si vous ne souhaitez pas utiliser les configurations CTF standard, vous pouvez les personnaliser pour répondre à vos besoins.

Gabarits de la FCDQ

Type Gabarit	Détail
Code Gabarits	Lorsque vous concevez un modèle de classe, les gabarits de code définissent la manière dont le code squelette doit être généré pour un langage de programmation donné. Les gabarits d'une langue sont automatiquement associés à la langue. Les gabarits sont écrits sous forme de texte brut avec une syntaxe qui partage certains aspects des langages de balisage et des langages de script.
Transformation du Modèle Gabarits	Transformation du Modèle Gabarits fournit une méthode entièrement configurable pour définir comment les transformations Model Driven Architecture (MDA) convertissent les éléments et fragments de modèle d'un domaine à un autre. Ce processus est à deux niveaux. Il crée un langage intermédiaire (qui peut être visualisé pour le débogage) qui est ensuite traité pour créer les objets.
Gabarits de génération de code Comportementale	Enterprise Architect supporte la génération de code définissable par l'utilisateur des modèles UML Comportementale . Cela applique le framework Code Gabarit standard mais inclut des macros de génération de code spécifiques Enterprise Architect Simulation Bibliothèque (EASL).
Gabarits DDL	Gabarits DDL sont très similaires aux gabarits de génération de code, mais ils ont été étendus pour support la génération DDL avec leur propre ensemble de gabarits de base, de macros, de macros de fonctions et d'options gabarit .

Personnalisation du code Gabarit

Enterprise Architect vous aide à générer du code source à partir de modèles UML pour un large éventail de langages de programmation. gabarits standard (mappages) sont fournis prêts à l'emploi, mais vous pouvez personnaliser la façon dont le code est généré à l'aide du Code Gabarit Framework (CTF), pratique et flexible. Ce cadre sophistiqué vous permet de personnaliser chaque détail de la façon dont le code est généré, y compris la facilité de créer de nouveaux gabarits pour les langues non prises en charge dans le produit de base. Par exemple, JavaScript ne fait pas partie des langages pris en charge mais une série de gabarits peuvent être écrits rapidement pour générer JavaScript à partir de modèles UML . Dans ces cas, gabarits existants servent de point de départ et de référence utiles pour de nouveaux langages.

Le framework code gabarit fournit également le mécanisme de génération de modèles comportementaux et est utilisé pour la transformation gabarits .

Fonctionnalités

Fonctionnalité	Détail
Gabarits par défaut	Gabarits de code par défaut sont intégrés à Enterprise Architect pour les langages pris en charge par l'ingénierie avancée.
Éditeur de code Gabarit	Un éditeur de code Gabarit est fourni pour créer et maintenir des codes Gabarits définis par l'utilisateur.
Personnalisation Gabarits de code	Descriptions de la syntaxe gabarit et des macros et fonctions que vous pouvez utiliser pour contrôler les effets des gabarits .
Synchroniser le code	Un sous-ensemble des Code Gabarits par défaut pour synchroniser le code.

Coder et transformer Gabarits

gabarits gabarits code et de transformation (Transformation du Modèle) définissent comment le système doit générer ou transformer le code dans l'un ou l'autre des langages de programmation supporte par Enterprise Architect . Chaque langage dispose d'une large gamme de gabarits de base, chacun définissant comment une structure de code particulière est générée. Vous pouvez utiliser ces gabarits de base tels quels, ou vous pouvez personnaliser et ajouter des gabarits pour mieux support votre utilisation des langues standard ou d'autres langues que vous pourriez définir dans le système. Vous révision , mettez à jour et créez gabarits via l'éditeur Code Gabarit ou l'éditeur Transformation Gabarit .

L'ordre dans lequel les gabarits de base sont répertoriés dans les deux éditeurs est lié à l'ordre hiérarchique des objets et de leurs parties à traiter. Des appels sont effectués depuis certains gabarits de base vers d'autres, et vous pouvez ajouter d'autres appels aux gabarits de base et à vos propres gabarits personnalisés. Par défaut, le Fichier gabarit est le point de départ d'un processus de génération de code à travers les gabarits ; un fichier est constitué de classes pouvant contenir Attributes et des opérations.

Accéder

Ruban	Développer > Code source > Options > Modifier le code Gabarits Conception > Paquetage > Transformation > Transformation Gabarits
Raccourcis Clavier	Ctrl+Maj+P (Gabarits de génération de code) Ctrl+Alt+H (Gabarits de transformation MDA)

Application des Gabarits

Action	Détail
Appel Gabarits	Dans n'importe quel gabarit , vous pouvez appeler d'autres gabarits en utilisant %TemplateName%. Les signes de pourcentage (%) encadrant indiquent une macro. Vous l'utiliseriez pour un seul appel au gabarit ClassBody , %ClassBody%, comme indiqué : % list = "TemplateName" @separator= " \n" @indent= " " % La macro %list effectue une passe itérative sur tous les objets dans le périmètre du gabarit courant et appelle le TemplateName pour chacun d'eux : % list = "ClassBody" @separator= " \n" @indent= " " % Après génération ou transformation, chaque macro est remplacée pour produire la sortie générée ; pour un langage tel que C++, le résultat du traitement de ce gabarit pourrait être : /** * Ceci est un exemple note de cours générée à l'aide de gabarits de code * @auteur Sparx Systems */ classe ClassA : classeB publique { ...

	}
Exécution du code Gabarits	Chaque gabarit peut agir uniquement sur un type d'élément particulier ; par exemple, le gabarit ClassNotes n'agit que sur les éléments UML Class et Interface. L'élément à partir duquel le code est actuellement généré est dit être dans la portée ; si l'élément concerné est stéréotypé, le système recherche un gabarit défini pour ce stéréotype. Si un gabarit spécialisé est trouvé, il est exécuté ; sinon, l'implémentation par défaut du gabarit de base est utilisée. Gabarits sont traités séquentiellement, ligne par ligne, en remplaçant chaque macro par sa valeur de texte sous-jacente issue du modèle.
Transférer Gabarits entre projets	Si vous modifiez un gabarit de génération ou de transformation de code de base, ou créez un gabarit personnalisé, vous pouvez les copier d'un projet à un autre en tant que données de référence.

Gabarits de base

Le Framework Code Gabarit se compose d'un certain nombre de gabarits de base. Chaque gabarit de base transforme des aspects particuliers de l' UML en parties correspondantes de langages orientés objet.

Les gabarits de base forment une hiérarchie qui varie légèrement selon les différents langages de programmation. Dans une hiérarchie gabarit typique pertinente pour un langage tel que C# ou Java (qui n'ont pas de fichiers d'en-tête), les gabarits peuvent être modélisés sous forme de classes, mais ne sont généralement que du texte brut. Cette hiérarchie serait légèrement plus compliquée pour des langages tels que C++ et Delphi, qui ont gabarits d'implémentation distincts.

Chacun des gabarits de base doit être spécialisé pour être utile en ingénierie de code ; en particulier, chaque gabarit est spécialisé pour les langues (ou 'produits') supportées. Par exemple, il existe un gabarit ClassBody défini pour C++, un autre pour C# , un autre pour Java, et ainsi de suite ; en spécialisant les gabarits , vous pouvez adapter le code généré pour l'entité UML correspondante.

Une fois les gabarits de base spécialisés pour une langue donnée, ils peuvent être spécialisés davantage en fonction de :

- Le stéréotype d'une classe, ou
- Le stéréotype d'une fonctionnalité (où la fonctionnalité peut être une opération ou un attribut)

Ce type de spécialisation permet, par exemple, à une opération C# stéréotypée comme « propriété » d'avoir un gabarit Operation Body différent d'une opération ordinaire ; le gabarit Operation Body peut ensuite être spécialisé davantage, sur la base du stéréotype de classe.

gabarits de base utilisés dans le CTF

Gabarit	Description
Attribut	Un gabarit de niveau supérieur pour générer des variables membres à partir d'attributs UML .
Déclaration d'attribut	Utilisé par l'attribut gabarit pour générer une déclaration de variable membre.
Notes d'attribut	Utilisé par l'attribut gabarit pour générer notes de variables membres.
Classe	Un gabarit de haut niveau pour générer des classes à partir de classes UML .
Base de classe	Utilisé par le gabarit de classe pour générer un nom de classe de base dans la liste d'héritage d'une classe dérivée, lorsque la classe de base n'existe pas dans le modèle.
Corps de classe	Utilisé par le gabarit de Classe pour générer le corps d'une Classe.
Déclaration de classe	Utilisé par le gabarit Class pour générer la déclaration d'une Classe.
Interface de classe	Utilisé par le gabarit de classe pour générer un nom d'interface dans la liste d'héritage d'une classe dérivée, lorsque l'interface n'existe pas dans le modèle.
Notes de cours	Utilisé par le gabarit de classe pour générer les notes de classe.
Déposer	Un gabarit de niveau supérieur pour générer le fichier source. Pour les langages tels que C++, cela correspond au fichier d'en-tête.
Section Importation	Utilisé dans le Fichier gabarit pour générer des dépendances externes.

Attribut lié	Un gabarit de niveau supérieur pour générer des attributs dérivés d'associations UML .
Notes sur les attributs liés	Utilisé par le gabarit Linked Attribute pour générer les notes d'attribut.
Déclaration d'attribut lié	Utilisé par le gabarit Linked Attribute pour générer la déclaration d'attribut.
Base de classe liée	Utilisé par le gabarit Class pour générer un nom de classe de base dans la liste d'héritage d'une classe dérivée, pour un élément Class du modèle qui est un parent de la classe actuelle.
Interface de classe liée	Utilisé par le gabarit de classe pour générer un nom d'interface dans la liste d'héritage d'une classe dérivée, pour un élément d'interface du modèle qui est parent de la classe actuelle.
Namespace	Un gabarit de niveau supérieur pour générer des espaces de noms à partir Paquetages UML (bien que tous les langages n'aient pas d'espaces de noms, ce gabarit peut être utilisé pour générer une construction équivalente, telle que Paquetages en Java).
Corps Namespace	Utilisé par le gabarit Namespace pour générer le corps d'un espace de noms.
Déclaration Namespace	Utilisé par le gabarit Namespace pour générer la déclaration de l'espace de noms.
Opération	Un gabarit de niveau supérieur pour générer des opérations à partir des opérations d'une classe UML .
Corps d’opération	Utilisé par le gabarit d'opération pour générer le corps d'une opération UML .
Déclaration d'opération	Utilisé par le gabarit d'opération pour générer la déclaration d'opération.
Notes d'opération	Utilisé par le gabarit d'opération pour générer la documentation d'une opération.
Paramètre	Utilisé par le gabarit de déclaration d'opération pour générer des paramètres.

Gabarits pour générer du code pour les langages avec des sections d'interface et d'implémentation séparées

Gabarit	Description
Implémentation de classe	Un gabarit de haut niveau pour générer l'implémentation d'une classe.
Implémentation du corps de classe	Utilisé par le gabarit Class Impl pour générer l’implémentation des membres de la classe.
Fichier Impl.	Un gabarit de niveau supérieur pour générer le fichier d'implémentation.
Notes de fichier	Utilisé par le gabarit File Impl pour générer notes dans le fichier source.

implémentées	
Importer la section Implémenter	Utilisé par le gabarit File Impl pour générer des dépendances externes.
Opération Implantation	Un gabarit de niveau supérieur pour générer des opérations à partir des opérations d'une classe UML .
Opération Corps Impl	Utilisé par le gabarit d'opération pour générer le corps d'une opération UML .
Déclaration d'opération Implémentation	Utilisé par le gabarit d'opération pour générer la déclaration d'opération.
Notes sur l'opération	Utilisé par le gabarit d'opération pour générer la documentation d'une opération.

Exporter Gabarits de génération et de transformation de code

Il est possible d'exporter gabarits de génération et de transformation de code de votre modèle vers un fichier .xml. Vous pouvez ensuite importer ce fichier - et donc les gabarits - dans d'autres modèles, comme données de référence. Vous pouvez exporter gabarits personnalisés, qui incluent ceux que vous ou d'autres utilisateurs avez créés et mis à jour, ainsi que gabarits de base (standard) qui ont été personnalisés. Vous n'avez pas besoin d'exporter gabarits de base qui n'ont pas été modifiés, car ceux-ci sont disponibles dans chaque installation d' Enterprise Architect .

Accéder

Ruban	Paramètres > Modèle > Transfert > Exporter les données de référence

Exporter un gabarit de Génération de Code ou gabarit de Transformation

Étape	Action
1	Dans la dialogue 'Exporter les données de référence', dans la liste 'Nom', sélectionnez les gabarits à exporter. La liste comprend tous les gabarits standard de génération ou de transformation de code qui ont été modifiés, ainsi que tous les gabarits personnalisés que vous avez créés ou modifiés. Vous pouvez sélectionner un ou plusieurs gabarits à exporter vers un seul fichier XML, en appuyant sur Ctrl ou Maj tout en cliquant sur les noms gabarit .
2	Cliquez sur le bouton Exporter.
3	Lorsque vous y êtes invité, entrez un nom de fichier valide avec une extension .xml.
4	Cliquez sur le bouton Enregistrer et sur le bouton OK . Cela exporte le(s) gabarit (s) vers le fichier ; vous pouvez utiliser n'importe quelle visionneuse de texte ou XML pour examiner le fichier.

Gabarits de génération et de transformation de code d'importation

Si vous avez exporté gabarits de génération de code et/ou de transformation à partir d'un modèle Enterprise Architect , vous pouvez les importer dans d'autres modèles Enterprise Architect comme données de référence.

Accéder

Ruban	Paramètres > Modèle > Transfert > Importer les données de référence

Importer des Gabarits de génération et/ou de transformation de code

Étape	Action
1	Dans la dialogue « Importer des données de référence », cliquez sur le bouton Sélectionner un fichier et accédez au fichier .xml contenant les gabarits de génération ou de transformation de code requis.
2	Sélectionnez le nom d'un ou plusieurs jeux de données gabarit et cliquez sur le bouton Importer.

Synchroniser le code

Enterprise Architect utilise gabarits de code lors de la synchronisation directe de ces langages de programmation :

- ActionScript
- C
- C++
- C#
- Delphes
- Java
- PHP
- Python
- VB
- VB.Net

Trois types de changements peuvent survenir dans la source lorsqu'elle est synchronisée avec le modèle UML :

- Les sections existantes sont synchronisées : par exemple, le type de retour dans une déclaration d'opération est mis à jour
- De nouvelles sections sont ajoutées aux fonctionnalités existantes : par exemple, Notes sont ajoutées à une déclaration Class là où il n'y en avait pas auparavant
- De nouvelles fonctionnalités et éléments sont ajoutés : par exemple, une nouvelle opération est ajoutée à une classe

Chacune de ces modifications a un effet différent sur le CTF et doit être gérée différemment par Enterprise Architect , comme décrit dans ces rubriques :

- *Synchroniser les sections existantes*
- *Ajouter de nouvelles sections aux Fonctionnalités existantes*
- *Ajouter de nouvelles Fonctionnalités et éléments*

Sections de code pouvant être synchronisées

Seul un sous-ensemble des gabarits de base CTF est utilisé lors de la synchronisation. Ce sous-ensemble correspond aux sections distinctes qu'Enterprise Enterprise Architect reconnaît dans le code source.

Code Gabarit	Section des codes
Notes de cours	Commentaires précédant la déclaration Class.
Déclaration de classe	Jusqu’aux parents de la classe inclus.
Notes d'attribut	Commentaires précédant une déclaration d'attribut.
Déclaration d'attribut	Jusqu'au caractère final inclus.
Notes d'opération	Commentaires précédant une déclaration d’opération.
Notes sur l'opération	Quant aux Notes d'opération.
Déclaration d'opération	Jusqu'au caractère final inclus.
Déclaration d'opération	

Implémentation	Jusqu'au caractère final inclus.
Corps d'opération	Tout entre et y compris les appareils orthodontiques.
Opération Corps Impl	Quant à l'opération Body.

Synchroniser les sections existantes

Lorsqu'une section existante dans le code source diffère du résultat généré par le gabarit correspondant, cette section est remplacée.

Considérons, par exemple, cette déclaration de classe C++ :

(asm) classe A : public B

Supposons maintenant que vous ajoutiez une relation d'héritage de la classe A à la classe C ; la déclaration Class entière serait remplacée par quelque chose qui ressemble à ceci :

(asm) classe A : public B, public C

Ajouter de nouvelles sections

Ces sections peuvent être ajoutées aux fonctionnalités existantes dans le code source, sous forme de nouvelles sections :

- Notes de cours
- Notes d'attribut
- Notes d'opération
- Notes sur l'opération
- Corps d'opération
- Opération Corps Impl

Supposons que, dans cet exemple, la classe A n'avait aucune note lorsque vous avez généré le code :

(asm) classe A : public B, public C

Si vous spécifiez maintenant une note dans le modèle pour la classe A, Enterprise Architect tente d'ajouter la nouvelle note du modèle lors de la synchronisation, en exécutant le gabarit Class Notes .

Pour laisser de la place à l'insertion de la nouvelle section, vous pouvez spécifier la quantité d'espace blanc à ajouter à la section via des macros de synchronisation.

Ajouter de nouvelles Fonctionnalités et éléments

Ces fonctionnalités et éléments peuvent être ajoutés au code source lors de la synchronisation :

- Attributes
- Classes intérieures
- Opérations

Ils sont ajoutés en exécutant les gabarits pertinents pour chaque nouvel élément ou fonctionnalité du modèle.

Enterprise Architect tente de préserver l'indentation appropriée des nouvelles fonctionnalités dans le code, en trouvant les indentations spécifiées dans les macros de liste de la classe ; pour les langages qui utilisent des espaces de noms, la macro 'synchNamespaceBodyIndent' est disponible.

Les classes définies dans un espace de noms (non global) sont indentées selon la valeur définie pour cette macro, lors de la synchronisation.

La valeur est ignorée :

- Pour les classes définies dans un Paquetage configuré en tant qu'espace de noms racine, ou
- Si l'option ' Générer Namespaces ' est définie sur False dans la page du langage approprié (C# , C++ ou VB.Net) sur la dialogue 'Préférences' (' Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > <langue>')

L'éditeur de Code Gabarit

L'éditeur Code Gabarit fournit les facilités de l' Éditeur de Code commun, y compris Intelli-sense pour les différentes macros. Pour plus d'informations sur Intelli-sense et Common Éditeur de Code , consultez la rubrique *Modification du code source* .

Accéder

Ruban	Développer > Code source > Options > Modifier le code Gabarits
Raccourcis Clavier	Ctrl+Maj+P

Possibilités

Option	Action
Langue	Sélectionnez le langage de programmation.
Nouveau langage	Affichez la dialogue 'Types de données des langages de programmation', qui vous permet d'inclure des langages de programmation autres que ceux pris en charge pour Enterprise Architect , pour lesquels créer ou modifier gabarits de code.
Gabarit	Affichez le contenu du gabarit actif, et ouvrez l'éditeur de modification gabarits .
Gabarits	Lister les gabarits du code de base ; le gabarit actif est mis en surbrillance. Le champ 'Modifié' indique si vous avez modifié le gabarit par défaut pour la langue actuelle.
Remplacements de stéréotypes	Lister les gabarits stéréotypés, pour le gabarit de base actif. Le champ 'Modifié' indique si vous avez modifié un gabarit stéréotypé par défaut.
Ajouter un nouveau Gabarit personnalisé	Invoquez une dialogue pour créer un gabarit stéréotypé personnalisé.
Ajouter un nouveau remplacement stéréotypé	Invoque une dialogue pour ajouter un gabarit stéréotypé, pour le gabarit de base actuellement sélectionné.
Obtenir Gabarit par défaut	Mettez à jour l'affichage de l'éditeur avec la version par défaut du gabarit actif.
Sauvegarder	Ecrasez les gabarits actifs par le contenu de l'éditeur.
Supprimer	Si vous avez remplacé le gabarit actif, le remplacement est supprimé et remplacé par le code gabarit par défaut correspondant.

Notes

- Les Code Gabarits modifiés et définis par l'utilisateur peuvent être importés et exportés en tant que données de référence (voir la rubrique *Partage des données de référence*) ; les gabarits définis pour chaque langue sont indiqués dans la dialogue 'Exporter les données de référence' par le nom de la langue avec le suffixe _Code_Templates - si aucun gabarits n'existe pour une langue, il n'y a pas d'entrée pour la langue dans le dialogue

Créer un nouveau Gabarit personnalisé

La dialogue Créer un nouveau Gabarit personnalisé offre la possibilité de créer un gabarit personnalisé pour le langage de programmation ou de système de gestion de base de données (SGBD) actuel, en fonction des informations modifiées avec l'éditeur de code Gabarit .

Lorsque cette dialogue est chargée, vous serez invité à entrer une valeur pour Type Gabarit et le nom Gabarit . Afin d'enregistrer un nouveau gabarit Type et le nom sont requis.

Possibilités

Option	Action
Type Gabarit	Choisissez le type de Gabarit pour le nouveau Gabarit personnalisé.
Nom Gabarit	Entrez un nom pour le nouveau Gabarit personnalisé.
OK	Enregistrez les détails du nouveau Custom Gabarit .
Annuler	Fermez la boîte dialogue Créer un nouveau Gabarit personnalisé et perdez toutes les modifications non enregistrées.

Note :

Tous gabarits de type " <none> " sont traités comme des fonctions, donc Enterprise Architect supprimera automatiquement tous les caractères d'espace saisis dans le nom.

Syntaxe du code Gabarit

Les codes Gabarits sont écrits à l'aide de l'éditeur de code Gabarit d' Enterprise Architect . L'éditeur Code Gabarit supporte la coloration syntaxique du langage Code Gabarit Framework.

Éléments de syntaxe

Éléments	Détail
Constructions de base	Gabarits peuvent contenir : • Texte littéral • Variables • Macro • Appels à d'autres gabarits
commentaires	Si vous souhaitez ajouter des commentaires aux gabarits , utilisez la commande : $COMMENT = "texte" où « texte » est le texte du commentaire ; cela doit être mis entre guillemets. La commande est sensible à la casse et doit être saisie en majuscules.

Texte littéral

Tout texte dans un gabarit donné qui ne fait pas partie d'une macro ou d'une définition/référence de variable est considéré comme du texte littéral. À l'exception des lignes vides, qui sont ignorées, le texte littéral est directement substitué du gabarit dans le code généré.

Considérez cet extrait du gabarit de déclaration de classe Java :

$bases = "Base"

classe % nom de classe % $bases

Sur la dernière ligne, le mot « class », y compris l'espace suivant, serait traité comme un texte littéral et donc, pour une classe nommée « foo », renverrait le résultat :

classe fooBase

Une ligne vide après la variable $bases n'aurait aucun effet sur la sortie.

Insertion de caractères système :

Les caractères %, $, " et \ ont une signification particulière dans la syntaxe gabarit et ne peuvent pas toujours être utilisés comme texte littéral. Si ces caractères doivent être générés à partir des gabarits , ils peuvent être reproduits en toute sécurité à l'aide de ces macros de substitution directe :

Macro	Action
%dl%	Produisez un caractère $ littéral.
%pc%	Produisez un caractère % littéral.
%qt%	Produisez un caractère " littéral.
%sl%	Produire un caractère \ littéral

Notes

Les opérateurs de conjonction String (« + », « += ») ne sont pas obligatoires mais peuvent être utilisés

Variables

Les variables Gabarit offrent un moyen pratique de stocker et de récupérer des données dans un gabarit . Cette section explique comment les variables sont définies et référencées.

Définitions des variables

Les définitions de variables prennent la forme de base :

$<nom> = <valeur>

où <nom> peut être n'importe quelle séquence alphanumérique et <valeur> est dérivé d'une macro ou d'une autre variable.

Un exemple simple de définition serait :

$foo = %nom de classe%

Les variables peuvent être définies à l'aide des valeurs de :

- Macros de substitution, de fonction ou de liste
- Littéraux String , placés entre guillemets doubles
- Références variables

Règles de définition

Ces règles s'appliquent aux définitions de variables :

- Les variables ont une portée globale dans le gabarit dans lequel elles sont définies et ne sont pas accessibles aux autres gabarits
- Chaque variable doit être définie en début de ligne, sans aucun espace intermédiaire
- Les variables sont désignées en préfixant le nom avec $, comme dans $foo
- Les variables n'ont pas besoin d'être déclarées avant d'être définies
- Les variables doivent être définies à l'aide de l'opérateur d'affectation (=) ou de l'opérateur d'addition-affectation (+=).
- Plusieurs termes peuvent être combinés dans une seule définition à l'aide de l'opérateur d'addition (+)

Exemples

Utilisation d'une macro de substitution :

$foo = %opTag : "bar"%

Utiliser une string littérale :

$foo = "barre"

En utilisant une autre variable :

$foo = $bar

Utilisation d'une macro de liste :

$ops = %list="Opération" @separator="\n\n" @indent="\t"%

Utilisation de l'opérateur d'addition-affectation (+=) :

$body += %list="Opération" @separator="\n\n" @indent="\t"%

Cette définition équivaut à :

$body = $body + %list="Opération" @separator="\n\n" @indent="\t"%

Utiliser plusieurs termes :

$templateArgs = %list="ClassParameter" @separator=", "%

$template ="modèle< " + $templateArgs + "> "

Références de variables

Les valeurs des variables peuvent être récupérées en utilisant une référence du formulaire :

$<nom>

où <name> peut être une variable préalablement définie.

Les références de variables peuvent être utilisées :

- Dans le cadre d'une macro, comme l'argument d'une macro de fonction
- En tant que terme dans une définition de variable
- En substitution directe de la valeur variable dans la sortie

Il est légal de référencer une variable avant qu'elle ne soit définie. Dans ce cas, la variable est supposée contenir une string valeur vide : " "

Références de variables - Exemple 1

Utiliser des variables dans le cadre d'une macro. Ceci est un extrait du gabarit ClassNotes C++ par défaut.

$wrapLen = %genOptWrapComment%

$style = %genOptCPPCommentStyle% (Définir des variables pour stocker les options de style et de longueur de retour à la ligne)

%if $style == "XML.NET"% (Référence à $style dans le cadre d'une condition)

%XML_COMMENT($wrapLen)%

%autre%

%CSTYLE_COMMENT($wrapLen)% (Référence à $wrapLen comme argument de la macro de fonction)

%fin si%

Références de variables - Exemple 2

Utilisation de références de variables dans le cadre d'une définition de variable.

$foo = "foo" (Définir nos variables)

$barre = "barre"

$foobar = $foo + $bar ($foobar contient maintenant la valeur foobar)

Références de variables - Exemple 3

Remplacement des valeurs de variables dans la sortie.

$bases=%classInherits% (Stocker le résultat du gabarit ClassInherits dans $bases)

Classe %className%$bases (affiche maintenant la valeur de $bases après le nom de la classe)

Macro

Les macros donnent accès aux champs d'éléments dans le modèle UML et sont également utilisées pour structurer la sortie générée. Toutes les macros sont entourées de signes de pourcentage (%), comme indiqué :

%<nom de la macro>%

En général, les macros (y compris les délimiteurs %) remplacent le texte littéral dans la sortie. Par exemple, considérons cet élément du gabarit de déclaration de classe :

... classe %className% ...

La macro de substitution de champ, %className%, entraînerait le remplacement du nom de classe actuel dans la sortie. Ainsi, si la classe générée était nommée Foo, le résultat serait :

... classe Foo ...

Le CTF contient un certain nombre de types de macro :

- Template Substitution Macros
- Field Substitution Macros
- Substitution Examples
- Attribute Field Substitution Macros
- Class Field Substitution Macros
- Code Generation Option Field Substitution Macros
- Connector Field Substitution Macros
- Constraint Field Substitution Macros
- Effort Field Substitution Macros
- File Field Substitution Macros
- File Import Field Substitution Macros
- Link Field Substitution Macros
- Linked File Field Substitution Macros
- Metric Field Substitution Macros
- Operation Field Substitution Macros
- Package Field Substitution Macros
- Parameter Field Substitution Macros
- Problem Field Substitution Macros
- Requirement Field Substitution Macros
- Resource Field Substitution Macros
- Risk Field Substitution Macros
- Scenario Field Substitution Macros
- Tagged Value Substitution Macros
- Template Parameter Substitution Macros
- Test Field Substitution Macros
- Function Macros
- Control Macros
- List Macro
- Branching Macros
- Synchronization Macros

- The Processing Instruction (PI) Macro
- EASL Code Generation Macros

Macros de substitution Gabarit

Les macros de substitution Gabarit correspondent aux gabarits de base, et aboutissent à l'exécution du gabarit nommé. Par convention, les macros gabarit sont nommées selon la casse Pascal.

Structure : %<TemplateName>%

où <TemplateName> peut être l'un des gabarits répertoriés dans cette rubrique.

Lorsqu'un gabarit est référencé depuis un autre gabarit , il est généré par rapport aux éléments actuellement dans la portée. Le gabarit spécifique est sélectionné en fonction des stéréotypes des éléments concernés.

Comme indiqué précédemment, il existe une hiérarchie implicite entre les différents gabarits . Certaines précautions doivent être prises afin de préserver une hiérarchie raisonnable de références de gabarit . Par exemple, cela n'a pas de sens d'utiliser la macro %ClassInherits% dans l'un des gabarits d'attribut ou d'opération. À l'inverse, les gabarits Operation et Attribute sont conçus pour être utilisés dans le gabarit ClassBody .

Macros de substitution Gabarit dans le CTF

- Attribut
- Déclaration d'attribut
- AttributeDeclarationImpl
- Notes d'attribut
- Classe
- ClasseBase
- Corps de classe
- ClassBodyImpl
- Déclaration de classe
- ClassDeclarationImpl
- ClasseImpl
- Héritage de classe
- Interface de classe
- Notes de cours
- Paramètre de classe
- Déposer
- FichierImpl
- ImportSection
- ImportSectionImpl
- Classe intérieure
- Classe intérieureImpl
- Attribut lié
- LinkedAttributeDeclaration
- LinkedAttributeNotes
- Base de classe liée
- Interface de classe liée
- Namespace
- Corps de l'espace de noms

- Déclaration d'espace de noms
- Espace de nomsImpl
- Opération
- OpérationCorps
- OpérationBodyImpl
- Déclaration d'opération
- OperationDeclarationImpl
- OpérationImpl
- Notes d'opération
- Paramètre

Macros de substitution de champs

Les macros de substitution de champ permettent d'accéder aux données de votre modèle. Ils sont notamment utilisés pour accéder aux champs de données depuis :

- Paquetages
- Des classes
- Attributes
- Opérations, et
- Paramètres

Les macros de substitution de champ sont nommées selon la casse Camel. Par convention, la macro est préfixée par une forme abrégée de l'élément de modèle correspondant. Par exemple, les macros liées aux attributs commencent par att, comme dans la macro %attName%, pour accéder au nom de l'attribut dans la portée.

Les macros qui représentent des cases à cocher renvoient une valeur de T si la case est cochée. Sinon la valeur est vide.

Ce tableau répertorie un petit nombre de macros de substitution de champs de projet. Les macros spécifiques à un type sont répertoriées dans les sous-thèmes de cette section *Macros de substitution de champ* .

Macros de projet

Nom de la macro	Description
eaDateHeure	L'heure actuelle au format : JJ-MMM-AAAA HH:MM:SS AM/PM.
eaGUID	Un GUID unique pour cette génération.
VersionEA	Version du programme (située dans la dialogue 'À propos Enterprise Architect ' en sélectionnant ' Démarrer > Aide > Aide > À propos d'EA').

Exemples de substitution

Les macros de substitution de champ peuvent être utilisées de deux manières :

- Substitution directe ou
- Substitution conditionnelle

Remplacement direct

Ce formulaire substitue directement la valeur correspondante de l'élément concerné dans la sortie.

Structure : %<macroName>%

Où <macroName> peut être l'une des macros répertoriées dans les tableaux Macros de substitution de champ.

Exemples

- %nom du cours%
- %opName%
- %attName%

Substitution conditionnelle

Cette forme de macro permet d'effectuer des substitutions alternatives en fonction de la valeur de la macro.

Structure : %<macroName> (== "<text> ") ? <subTrue> (: <subFalse>) %

Où:

- () indique que les valeurs entre parenthèses sont facultatives
- <text> est une string représentant une valeur possible pour la macro
- <subTrue> et <subFalse> peuvent être une combinaison de chaînes entre guillemets et du mot clé valeur ; lorsque la valeur est utilisée, elle est remplacée par la valeur de la macro dans la sortie

Exemples

- %classAbstract=="T" ? "pur" : " "%
- %opStereotype=="opérateur" ? "opérateur" : " "%
- %paramDefault != " " ? " = " valeur : " "%

Ces trois exemples ne génèrent rien si la condition échoue. Dans ce cas, la condition False peut être omise, ce qui entraîne l'utilisation suivante :

- %classAbstract=="T" ? "pur"%
- %opStereotype=="opérateur" ? "opérateur"%
- %paramDefault != " " ? " = "valeur%

Le troisième exemple des deux blocs montre une comparaison vérifiant une valeur ou une existence non vide. Ce test peut également être omis.

- %paramDefault ? " = " valeur : " "%

- %paramDefault ? " = " valeur%

Tous ces exemples contenant paramDefault sont équivalents. Si le paramètre dans la portée avait une valeur par défaut de 10, le résultat de chacun d'eux serait normalement :

= 10

Notes

- Dans une macro de substitution conditionnelle, tout espace blanc suivant <macroName> est ignoré ; si un espace blanc est requis dans la sortie, il doit être inclus dans les chaînes de substitution citées

Macros de substitution de champs d'attribut

Ce tableau répertorie chacune des macros de substitution de champ d'attribut.

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros d'attributs

Nom de la macro	Description
attAlias	dialogue ' Attributes ' : Alias.
attAllowDuplicates	Boîte dialogue 'Détail Attributes ' : case à cocher 'Autoriser les doublons'.
attClassifierGUID	GUID unique pour le classificateur de l'attribut actuel.
attCollection	dialogue 'Détail Attributes ' : case à cocher 'L'attribut est une collection'.
attConst	dialogue ' Attributes ' : case à cocher 'Const'.
attContainerType	dialogue 'Détail Attributes ' : Type de conteneur.
attConfinement	dialogue ' Attributes ' : Confinement.
attDérivé	dialogue ' Attributes ' : case à cocher 'Dérivé'.
attGUID	Le GUID unique pour l’attribut actuel.
attInitial	dialogue ' Attributes ' : Initiale.
attIsEnumLiteral	dialogue ' Attributes ' : case à cocher 'Est littéral'.
attIsID	dialogue 'Détail Attributes ' : case à cocher 'isID'.
attLongueur	dialogue 'Colonne' : Durée.
attLowerBound	dialogue 'Détail Attributes ' : Limite inférieure.
attName	dialogue ' Attributes ' : Nom.
attNotes	dialogue ' Attributes ' : Notes .
attOrderedMultiplicity	dialogue 'Détail Attributes ' : case à cocher 'Multiplicité ordonnée'.
attPropriété	Boîte dialogue ' Attributes ' : case à cocher ' Propriété '.
attQualType	Le type d'attribut qualifié par le chemin de l'espace de noms (si vous générez des espaces de noms) et le chemin du classificateur (délimité par des points). Si le classificateur d'attribut n'a pas été défini, équivaut à la macro attType.

attScope	dialogue ' Attributes ' : Portée.
attStatique	dialogue ' Attributes ' : case à cocher 'Statique'.
attStéréotype	dialogue ' Attributes ' : Stéréotype.
attType	dialogue ' Attributes ' : Type .
attUpperBound	dialogue 'Détail Attributes ' : Limite supérieure.
attVolatile	dialogue 'Détail Attributes ' : case à cocher 'Transient'.

Macros de substitution de champs de classe

Ce tableau fournit une liste de méthodes pour accéder à chaque propriété de classe disponible dans les gabarits de génération et de transformation de code.

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros de classe

Nom de la macro	Description
Type d'élément	Le type d'élément : Interface ou Classe.
classeRésumé	dialogue Classe ' Propriétés ' : case à cocher 'Résumé' (onglet 'Détails').
Alias de classe	dialogue de la classe ' Propriétés ' : champ 'Alias'.
Arguments de classe	dialogue de classe 'Détail' : Gabarits C++ : Arguments.
classeAuteur	dialogue de la classe ' Propriétés ' : champ 'Auteur'.
nomBaseClasse	dialogue ' Type Hiérarchie' : Nom de classe (à utiliser là où aucun connecteur n'existe entre les classes enfants et de base).
classeBaseScope	La portée de l'héritage en tant qu'ingénierie inverse. (À utiliser là où aucun connecteur n’existe entre les classes enfant et de base.)
classeBaseVirtuelle	La propriété virtuelle de l'héritage en ingénierie inverse. (À utiliser là où aucun connecteur n’existe entre les classes enfant et de base.)
classeComplexité	dialogue de la classe ' Propriétés ' : champ 'Complexité'.
classeCréée	La date et l'heure de création de la classe.
GUID de classe	GUID unique pour la classe actuelle.
classeHasConstructeur	Examine la liste des méthodes de l' object actuel et, en fonction des conventions du langage actuel, renvoie T s'il s'agit d'un constructeur par défaut. Généralement utilisé avec la macro genOptGenConstructor.
classeHasCopyConstructor	Examine la liste des méthodes de l' object actuel et, en fonction des conventions du langage actuel, renvoie T s'il s'agit d'un constructeur par copie. Généralement utilisé avec la macro genOptGenCopyConstructor.
classeHasDestructor	Examine la liste des méthodes de l' object actuel et, selon les conventions du langage actuel, renvoie T si l'une est un destructeur. Généralement utilisé avec la macro genOptGenDestructor.
classeHasParent	True, si la classe dans la portée comporte une ou plusieurs classes de base.

classHasStéréotype	True, si la classe dans la portée a un stéréotype qui correspond à un nom de stéréotype (que vous pouvez éventuellement spécifier comme pleinement qualifié). Il vérifie donc tous les stéréotypes d'une classe et renvoie « T » si l'un d'entre eux correspond au stéréotype spécifié ou à une spécialisation de celui-ci. Par exemple: • %classHasStereotype : "block"% renverra 'T' pour toute classe stéréotypée par bloc à partir de n'importe quelle version de SysML, y compris associationBlock • %classHasStereotype:"SysML1.4::block"% correspondra spécifiquement aux versions SysML 1.4 Comparez cela avec classStereotype, plus tard.
importations de classes	dialogue 'Code Gen' : Importations.
la classeEstActive	dialogue de classe 'Avancé' : case à cocher 'Est Actif '.
classIsAssociationClass	True, si l'Association est un connecteur AssociationClass.
classeEstInstanciée	Vrai, si la Classe est une Classe gabarit instanciée.
classeEstFeuille	dialogue de classe 'Avancé' : case à cocher 'Is Leaf'.
classIsRoot	Boîte dialogue de classe 'Avancé' : case à cocher 'Est Root'.
classIsSpecification	dialogue de classe 'Avancé' : case à cocher 'Is Spécification '.
mots-clés de classe	Boîte dialogue Classe ' Propriétés ' : champ 'Mots clés'.
classeLangue	dialogue de la classe ' Propriétés ' : champ 'Langue'.
classeMacros	Une liste de macros séparées par des espaces définies pour la classe.
classeModifiée	La date et l'heure de la dernière modification du cours.
classeMultiplicité	dialogue de classe 'Avancé' : Multiplicité.
nom du cours	Boîte dialogue Classe ' Propriétés ' : champ 'Nom'.
notes de cours	Boîte dialogue Classe ' Propriétés ' : champ ' Note '.
classeParamDefault	dialogue de classe « Détail ».
nomParamclasse	dialogue de classe « Détail ».
classeParamType	dialogue de classe « Détail ».
classePersistance	Boîte dialogue Classe ' Propriétés ' : Champ 'Persistance' (onglet 'Détails')
phase de classe	Boîte dialogue Classe ' Propriétés ' : champ 'Phase'.
nomQualClasse	Le nom de la classe précédé de ses classes externes. Les noms de classe sont séparés par des doubles deux-points (::).

classeScope	Boîte dialogue Classe ' Propriétés ' : champ 'Scope'.
classeStéréotype	dialogue de la classe ' Propriétés ' : champ 'Stéréotype'. Récupère le nom du premier stéréotype appliqué à la classe. Lorsqu'il est utilisé dans une comparaison, il vérifie si ce premier stéréotype correspond exactement à une string . Par exemple : %classStereotype=="enumeration" ? "énumération" : "classe"% Comparez cela avec classHasStereotype, plus tôt.
état de classe	Boîte dialogue Classe ' Propriétés ' : champ 'Statut'.
version de classe	Boîte dialogue Classe ' Propriétés ' : champ 'Version'.

Macros de substitution de champs d'option de génération de code

Les macros de substitution de champ d'option de génération de code fonctionnent sur les options de génération de code source définies dans les pages « Ingénierie du code source » de :

- dialogue 'Préférences' (' Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source') pour les options spécifiques à l'utilisateur, ou
- dialogue « Gérer les options Modèle » (« Paramètres > Modèle > Options ») pour les options spécifiques au modèle

Pour plus d'informations sur la répartition des options, consultez la rubrique *Options d'ingénierie du code source* .

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide. Ce tableau répertorie chacune des macros de substitution de champ d'option de génération de code.

Macros d'options de génération de code

Nom de la macro	Description
genOptActionScriptVersion	Page Spécifications ActionScript : version par défaut.
genOptCDefaultAttributeType	C Page Spécifications : Type d'attribut par défaut .
genOptCGenMethodNotesInBody	Page Spécifications C : Notes de méthode en cours de mise en œuvre.
genOptCGenMethodNotesInHeader	Page Spécifications : Notes de méthode dans l'en-tête.
genOptCSynchNotes	Page Spécifications C : Synchroniser Notes dans la génération.
genOptCSynchCFile	Page Spécifications C : Synchroniser le fichier d'implémentation dans la génération.
genOptCDefaultSourceDirectory	Page Spécifications : Répertoire source par défaut.
genOptCNamespaceDelimiter	Page Spécifications C : Délimiteur Namespace .
genOptCOperationRefParam	C Page Spécifications : référence comme paramètre de fonctionnement.
genOptCOperationRefParamStyle	C Page Spécifications : Style de paramètre de référence.
genOptCOperationRefParamName	C Page Spécifications : Nom du paramètre de référence.
genOptCConstructorName	Page Spécifications C : Nom du constructeur par défaut.

genOptCDestructorName	Page Spécifications C : Nom du destructeur par défaut.
genOptCPPCommentStyle	Page Spécifications C++ : Style de commentaire.
genOptCPPDefaultAttributeType	Page Spécifications C++ : Type d'attribut par défaut .
genOptCPPDefaultReferenceType	Page Spécifications C++ : Type de référence par défaut .
genOptCPPDefaultSourceDirectory	Page Spécifications C++ : Répertoire source par défaut.
genOptCPPGenMethodNotesInHeader	Page Spécifications C++ : case à cocher « Notes de méthode dans l'en-tête ».
genOptCPPGenMethodNotesInBody	Page Spécifications C++ : case à cocher Notes de méthode dans le corps.
genOptCPPGetPrefix	Page Spécifications C++ : Obtenir le préfixe.
genOptCPPHeaderExtension	Page de spécifications C++ : extension d'en-tête.
genOptCPPSetPrefix	Page Spécifications C++ : Définir le préfixe.
genOptCPPSourceExtension	Page Spécifications C++ : Extension source.
genOptCPPSynchNotes	Page Spécifications C++ : Synchroniser Notes .
genOptCPPSynchCPPFile	Page Spécifications C++ : Synchroniser le fichier CPP.
genOptCSDefaultAttributeType	Page Spécifications C# : Type d'attribut par défaut .
genOptCSSourceExtension	Page Spécifications C# : extension de fichier par défaut.
genOptCSGenDispose	Page Spécifications C# : Générer Dispose.
genOptCSGenFinalizer	Page Spécifications C# : Générer Finalizer.
genOptCSGenEspace de noms	Page Spécifications C# : Générer Namespace .
genOptCSDefaultSourceDirectory	Page Spécifications C# : Répertoire source par défaut.
genOptDefaultAssocAttName	Page Ingénierie du code source : nom par défaut de l'attribut associé.
genOptDefaultConstructor	Page Durées de vie Object : visibilité du constructeur par défaut.

Scope	
genOptDefaultCopyConstructorScope	Page Durées de vie Object : visibilité du constructeur de copie par défaut.
genOptDefaultDatabase	Page Éditeurs de Code : Base de données par défaut.
genOptDefaultDestructorScope	Page Durées de vie Object : visibilité du constructeur de destructeur par défaut.
genOptGenCapitalizedPropriétés	Page « Ingénierie du code source » : case à cocher « Mettre en majuscule les noms d'attributs pour Propriétés ».
genOptGenComments	Page 'Ingénierie du Code Source' : case à cocher 'Commentaires - Générer '.
genOptGenConstructor	Page Durées de vie Object : case à cocher ' Générer Constructeur'.
genOptGenConstructorInline	Page Durées de vie Object : case à cocher "Constructeur en ligne".
genOptGenCopyConstructor	Page Durées de vie Object : case à cocher ' Générer un constructeur de copie'.
genOptGenCopyConstructorInline	Page Durées de vie Object : case à cocher "Copier le constructeur en ligne".
genOptGenDestructor	Page Durées de vie Object : case à cocher ' Générer Destructeur'.
genOptGenDestructorInline	Page Durées de vie Object : case à cocher "Destructeur en ligne".
genOptGenDestructorVirtuel	Page Durées de vie Object : case à cocher "Destructeur virtuel".
genOptGenImplementedInterfaceOps	Page 'Génération de Code' : case à cocher ' Générer des méthodes pour les interfaces implémentées'.
genOptGenPrefixBoolProperties	Page 'Ingénierie du code source' : case à cocher 'Utiliser 'Est' pour la propriété booléenne Get()'.
genOptGenRoleNames	Page « Ingénierie du code source » : case à cocher « Générer automatiquement les noms de rôle lors de la création de code ».
genOptGenUnspecAssocDir	Page « Ingénierie du code source » : case à cocher « Ne pas générer de membres là où la direction de l'association n'est pas spécifiée ».
genOptJavaDefaultAttributeType	Page Spécifications Java : type d'attribut par défaut.
genOptJavaGetPrefix	Page Spécifications Java : Obtenir le préfixe.
genOptJavaDefaultSourceDirectory	Page Spécifications Java : Répertoire source par défaut.

genOptJavaSetPrefix	Page Spécifications Java : Définir le préfixe.
genOptJavaSourceExtension	Page de spécifications Java : extension du code source.
genOptPHPDefaultSourceDirectory	Page Spécifications PHP : Répertoire source par défaut.
genOptPHPGetPrefix	Page Spécifications PHP : Obtenir le préfixe.
genOptPHPSetPrefix	Page de spécifications PHP : définir le préfixe.
genOptPHPSourceExtension	Page de spécifications PHP : extension de fichier par défaut.
genOptPHPVersion	Page de spécifications PHP : version PHP.
genOptPropertyPrefix	Page « Ingénierie du code source » : supprimez les préfixes lors de la génération des propriétés Get/Set.
genOptVBMultiUse	Page Spécifications VB : case à cocher "Multi-usage".
genOptVBPersistable	Page Spécifications VB : case à cocher « Persistable ».
genOptVBDataBindingBehavior	Page Spécifications VB : case à cocher « Comportement de liaison de données ».
genOptVBDataSourceBehavior	Page Spécifications VB : case à cocher "Comportement de la source de données".
genOptVBGlobal	Page Spécifications VB : case à cocher « Espace de noms global ».
genOptVBCcréable	Page Spécifications VB : case à cocher « Créable ».
genOptVBExposé	Page Spécifications VB : case à cocher « Exposé ».
genOptVBMTS	Page Spécifications VB : Mode de transaction MTS.
genOptVBNetGenEspace de noms	Page Spécifications VB.Net : Générer Namespace .
genOptVBVersion	Page Spécifications VB : version par défaut.
genOptWrapComment	Page « Ingénierie du code source » : longueur de retour à la ligne pour les lignes de commentaires.

Macros de substitution de champ de connecteur

Ce tableau répertorie chacune des macros de substitution de champ de connecteur.

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros de connecteur

Nom de la macro	Description
connecteurAlias	Boîte dialogue ' Propriétés ' du connecteur : champ 'Alias'.
connecteurAssociationClassElemGUID	GUID de l'élément Association Class du connecteur.
connecteurAssociationClassElemName	Nom de l'élément Association Class du connecteur.
connecteurDestAccess	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Accès.
connecteurDestAggregation	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Agrégation.
connecteurDestAlias	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Alias.
connecteurDestAllowDuplicates	dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : case à cocher 'Autoriser les doublons'.
connecteurDestChangeable	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Modifiable.
connecteurDestConstraint	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Contrainte(s).
connecteurDestContainment	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Confinement.
connecteurDestDerived	dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : case à cocher 'Dérivé'.
connecteurDestDerivedUnion	dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : case à cocher ' DerivedUnion '.
connecteurDestElem*	Ensemble de macros qui accèdent à une propriété de l'élément à l'extrémité cible d'un connecteur. Le * (astérisque) est un caractère générique qui correspond à n'importe quelle macro de substitution de classe dans la liste des macros de classe. Par exemple: • connecteurDestElemAlias (classAlias) • connecteurDestElemAuthor (classAuthor)
connecteurDestElemType	Type d'élément de l'élément de destination du connecteur. (Séparé des macros ConnectorDestElem* car il n'existe pas de macro de substitution classType.)

connecteurDestFeature*	Un ensemble de macros qui accèdent à une propriété de la fonctionnalité à l'extrémité cible d'un connecteur. Le * (astérisque) est un caractère générique qui correspond à n'importe quelle macro de substitution d'attribut ou d'opération dans la liste Macro d'attribut ou Macro d'opération, en fonction du connecteurDestFeatureType. Par exemple: • ConnectorDestFeatureReturnClassifierGUID - GUID du classificateur de retour d'une opération • ConnectorDestFeatureContainment - le confinement d'un attribut
connecteurDestFeatureType	Le type de fonctionnalité de destination du connecteur. • ConnectorDestFeatureType="Attribut" ou "Opération"
connecteurDestMemberType	dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Type de membre .
connecteurDestMultiplicité	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Multiplicité.
connecteurDestNavigabilité	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Navigabilité.
connecteurDestNotes	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Notes sur le rôle .
connecteurDestOrdonné	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : case à cocher 'Ordonné'.
connecteurDestOwned	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : case à cocher 'Possédé'.
connecteurDestQualifier	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Qualificateur(s).
connecteurDestRole	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Rôle.
connecteurDestScope	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Périmètre cible.
connecteurDestStéréotype	Connecteur Boîte dialogue ' Propriétés ', onglet ' Rôle cible ' : Stéréotype.
connecteurDirection	Propriétés du Connecteur : Direction.
connecteurEffet	dialogue 'Contraintes de transition' : champ 'Effet'.
connecteurGuard	Boîtes de dialogue « Flux Object » et « Contraintes de transition » : champ « Garde ».
connecteurGUID	GUID unique pour le connecteur actuel.
connecteurIsAssociationClass	True, si le connecteur est un connecteur AssociationClass.
Nom du connecteur	Propriétés du Connecteur : Nom.
connecteurNotes	Propriétés du Connecteur : Notes .

connecteurSourceAccess	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Accès.
connecteurSourceAgrégation	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Agrégation.
connecteurSourceAlias	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Alias.
connecteurSourceAllowDuplicates	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : case à cocher Autoriser les doublons.
connecteurSourceModifiable	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Modifiable.
connecteurSourceContrainte	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Contrainte(s).
connecteurSourceContainment	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Confinement.
connecteurSourceDérivé	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : case à cocher 'Dérivé'.
connecteurSourceDerivedUnion	dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : case à cocher ' DerivedUnion '.
connecteurSourceElem*	Ensemble de macros qui accèdent à une propriété de l'élément à l'extrémité source d'un connecteur. Le * (astérisque) est un caractère générique qui correspond à n'importe quelle macro de substitution de classe dans la liste des macros de classe. Par exemple: • connecteurSourceElemAlias (classAlias) • connecteurSourceElemAuthor (classAuthor)
connecteurSourceElemType	Type d'élément de l'élément source du connecteur. (Séparé des macros ConnectorSourceElem* car il n'existe pas de macro de substitution classType.)
connecteurSourceFeature*	Un ensemble de macros qui accèdent à une propriété de la fonctionnalité à l'extrémité source d'un connecteur. Le * (astérisque) est un caractère générique qui correspond à n'importe quelle macro de substitution d'attribut ou d'opération dans la liste Macro d'attribut ou Macro d'opération, en fonction du type de connecteurSourceFeatureType. Par exemple: • ConnectorSourceFeatureCode - Code de l'opération • ConnectorSourceFeatureInitial - Initiale de l'attribut
connecteurSourceFeatureType	Le type de fonctionnalité source du connecteur. • ConnectorSourceFeatureType="Attribut" ou "Opération"
connecteurSourceMemberType	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Type de membre .
connecteurSourceMultiplicité	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Multiplicité.

connecteurSourceNavigabilité	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Navigabilité.
connecteurSourceNotes	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Notes sur le rôle .
connecteurSourceOrdonné	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : case à cocher 'Ordonné'.
connecteurSourcePossédé	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : case à cocher 'Détenu'.
connecteurSourceQualifier	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Qualificateur(s).
connecteurSourceRole	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Rôle.
connecteurSourceScope	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Périmètre cible.
connecteurSourceStéréotype	Boîte dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Stéréotype.
connecteurStéréotype	Boîte dialogue ' Propriétés ' du connecteur : champ 'Stéréotype'.
connecteurTrigger	dialogue 'Contraintes de Transition' : champ ' Déclencheur '.
Type de connecteur	Le type de connecteur ; par exemple, Association ou Généralisation.
connecteurPoids	dialogue 'Contraintes de flux Object ' : champ 'Poids'.

Macros de substitution de champs de contraintes

Ce tableau répertorie chacune des macros de substitution de champ « Contrainte ».

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros de contraintes

Nom de la macro	Description
nom_contrainte	Boîte dialogue 'Classe', onglet 'Contraintes' : Nom.
contrainteNotes	Boîte dialogue 'Classe', onglet 'Contraintes' : Notes .
état de contrainte	Boîte dialogue 'Classe', onglet 'Contraintes' : Statut.
type de contrainte	dialogue 'Classe', onglet 'Contraintes' : Type .
contraintePoids	dialogue 'Classe', onglet 'Contraintes' : ordre des touches (main haut/bas).

Macros de substitution de champ d'effort

Ce tableau répertorie chacune des macros de substitution de champ « Effort ».

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros d'effort

Nom de la macro	Description
effortName	Fenêtre d’effort : Effort.
effortNotes	Fenêtre d'effort : Notes (sans étiquette).
effortTemps	Fenêtre d’effort : temps.
Type d'effort	Fenêtre d'effort : Type .

Macros de substitution de champs de fichier

Ce tableau répertorie chacune des macros de substitution de champ de fichier.

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros de fichiers

Nom de la macro	Description
extension de fichier	L'extension du type de fichier du fichier en cours de génération.
nom de fichier	Le nom du fichier en cours de génération.
nomfichierImpl	Le nom du fichier d'implémentation pour cette génération, le cas échéant.
en-têtes de fichiers	dialogue 'Code Gen' : En-têtes.
fichiersImports	dialogue 'Code Gen' : Importations. Pour les langages pris en charge, cela inclut également les dépendances dérivées de ces types de relations : • Agrégation • Association • Classificateur d'attributs • Type de retour de méthode • Classificateur de paramètres de méthode • Généralisation • Réalisation (à interfacer) • Liaison Gabarit (C++) • Dépendance
chemin du fichier	Le chemin complet du fichier en cours de génération.
filePathImpl	Le chemin complet du fichier d'implémentation pour cette génération, le cas échéant.

Macros de substitution de champs d'importation de fichiers

Ce tableau répertorie chacune des macros de substitution de champ d'importation de fichier.

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de T si la case est cochée. Sinon la valeur est vide.

Macros d'importation de fichiers

Nom de la macro	Description
importClassName	Le nom de la classe importée.
importFileName	Le nom de fichier de la classe en cours d'importation.
importFilePath	Le chemin complet de la classe en cours d'importation.
importFromAggregation	T si la Classe possède un connecteur Agrégation vers une Classe dans ce fichier, F sinon.
importerDeAssociation	T si la Classe possède un connecteur Association à une Classe dans ce fichier, F sinon.
importFromAtt	T si un attribut d'une Classe dans le fichier courant est du type de cette Classe, F sinon.
importFromDependency	T si la Classe possède un connecteur de Dépendance vers une Classe dans ce fichier, F sinon.
importFromGeneralization	T si la Classe possède un connecteur de Généralisation vers une Classe dans ce fichier, F sinon.
importerDeMeth	T si le type de retour d'une méthode d'une Classe dans le fichier courant est le type de cette Classe, F sinon.
importFromParam	T si un paramètre de méthode d'une Classe dans le fichier courant est du type de cette Classe ; sinon F.
importFromPropertyType	T si la Classe a une propriété (Partie/Port) typée vers une autre Classe, F sinon.
importFromRealization	T si la Classe possède un connecteur de Réalisation vers une Classe dans ce fichier, F sinon.
importFromTemplateBinding	T si la classe possède un connecteur TemplateBinding vers une classe dans ce fichier, F sinon.
importerDansFichier	T si la Classe est dans le fichier courant, F sinon.
importPackagePath	Le chemin Paquetage avec un '.' séparateur de la classe importée.

ImportRelativeFilePath	Le chemin de fichier relatif de la classe en cours d'importation à partir du chemin de fichier du fichier en cours de génération.

Macros de substitution de champ de lien

Si vous souhaitez donner accès aux données concernant les connecteurs du modèle, notamment les Associations et Généralisations, vous pouvez utiliser les macros 'Substitution de champ de lien'. Les noms des macros sont en boîtier Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée ; sinon la valeur est vide.

Lier les macros

Nom de la macro	Description/Résultat
lienAttAccess	dialogue ' Propriétés ' de l'association, Rôle cible : champ 'Accès'.
linkAttAggregation	Association dialogue ' Propriétés ', Rôle Source ou Cible : Agrégation.
linkAttCollectionClass	Collection appropriée pour l'attribut lié dans la portée.
linkAttContainment	Association ' Propriétés ' dialogue , Rôle cible : Confinement.
lienAttName	dialogue 'Association Propriétés ' : Cible.
lienAttNotes	dialogue ' Propriétés ' de l'association, Rôle cible : Notes sur le rôle .
lienAttOwnedByAssociation	Vrai, si la case « Propriété » de la page « Rôle(s) » de la dialogue « Propriétés » de l'association n'est pas cochée.
lienAttOwnedByClass	Vrai, si la case « Propriété » de la page « Rôle(s) » de la dialogue « Propriétés » de l'association est cochée.
lienAttQualName	La cible d'association qualifiée par le chemin de l'espace de noms (si vous générez des espaces de noms) et le chemin du classificateur (délimité par des points).
lienAttRole	dialogue ' Propriétés ' de l'association, Rôle cible : Rôle.
linkAttRoleAlias	dialogue 'Association Propriétés Rôle Cible' : Alias
linkAttStéréotype	Association dialogue ' Propriétés ', Rôle cible : Stéréotype.
lienAttTargetScope	dialogue ' Propriétés ' de l'association, Rôle cible : Périmètre cible.
lienCarte	Lien dialogue ' Propriétés ', Rôle cible : Multiplicité.
lienGUID	GUID unique pour le connecteur actuel.
linkIsAssociationClass	True, si l'Association est un connecteur AssociationClass.
le lien est lié	Renvoie T si des TemplateBindings sont spécifiés sur le connecteur.
lienParamSubs	Renvoie une liste des arguments spécifiés, séparés par des virgules.

lienNomParent	Généralisation dialogue ' Propriétés ' : champ 'Cible'.
lienParentQualName	La cible de généralisation qualifiée par le chemin de l'espace de noms (si vous générez des espaces de noms) et le chemin du classificateur (délimité par des points).
lienStéréotype	Le stéréotype du connecteur actuel.
lienHéritageVirtuel	Généralisation dialogue ' Propriétés ' : champ 'Héritage Virtuel'.

Macros de substitution de champs de fichiers liés

Ce tableau répertorie chacune des macros de substitution de champ « Fichier lié ».

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros de fichiers liés

Nom de la macro	Description
linkedFileLastWrite	dialogue Classe ' Propriétés ' : onglet 'Fichiers', champ 'Dernière écriture'.
linkedFileNotes	Boîte dialogue Classe ' Propriétés ' : onglet ' Fichiers ', champ ' Notes '.
chemin de fichier lié	Boîte dialogue Classe ' Propriétés ' : onglet 'Fichiers', champ 'Chemin du fichier'.
taille du fichier lié	Boîte dialogue Classe ' Propriétés ' : onglet 'Fichiers', champ 'Taille'.
Type de fichier lié	Boîte dialogue Classe ' Propriétés ' : onglet ' Fichiers ', champ ' Type '.

Macros de substitution de champs métriques

Ce tableau répertorie chacune des macros de substitution de champ métrique.

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros métriques

Nom de la macro	Description
métriqueName	Écran Métriques : champ « Métrique ».
métriqueNotes	Écran Métriques : champ (Notes).
Typemétrique	Écran Métriques : champ « Type ».
métriquePoids	Écran Métriques : champ « Poids ».

Macros de substitution de champ d'opération

Les macros 'Substitution de champ d'opération' permettent d'accéder aux données concernant les opérations dans le modèle. Les noms des macros sont en boîtier Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée ; sinon la valeur est vide.

Macros de substitution de champ d'opération

Nom de la macro	Description/Résultat
opRésumé	dialogue 'Opération' : case à cocher 'Virtuel'.
opAlias	dialogue 'Opération' : Alias.
opBehavior	dialogue 'Opération Comportement' : Comportement.
Code Opérateur	dialogue 'Opération Comportement' : Code de Comportement.
opConcurrency	dialogue 'Opération' : Concurrence.
opConst	dialogue 'Opération' : case à cocher 'Const'.
opGUID	GUID unique pour l'opération en cours.
opHasSelfRefParam	Analyse la liste des paramètres de l'opération en cours, en renvoyant « T » si un type est la référence de classe (cela peut être ClassA* ou ClassA&, en fonction de la valeur de la macro de substitution de champ d'option de génération de code genOptCOperationRefParamStyle).
opImplMacros	Une liste de macros séparées par des espaces définies dans l'implémentation de cette opération.
opIsQuery	dialogue 'Opération' : case à cocher 'IsQuery'.
opMacros	Une liste de macros séparées par des espaces définies dans la déclaration pour cette opération.
nomOp	dialogue 'Opération' : Nom.
opNotes	dialogue 'Opération' : Notes .
opPure	dialogue 'Opération' : case à cocher 'Pure'.
opReturnArray	dialogue 'Opération' : case à cocher 'Retourner le tableau'.
opReturnClassifierGUID	GUID unique pour le classificateur de l'opération en cours.
opReturnQualType	Type de retour de l'opération qualifié par le chemin de l'espace de noms (si vous générez des espaces de noms) et le chemin du classificateur (délimité par des points). Si le classificateur de type de retour n'a pas été défini, il est équivalent à la

	macro opReturnType.
opReturnType	dialogue 'Opération' : Type de retour .
opScope	dialogue « Opération » : Portée.
opStatique	dialogue 'Opération' : case à cocher 'Statique'.
opStéréotype	dialogue 'Opération' : Stéréotype.
opSynchronisé	dialogue 'Opération' : case à cocher 'Synchronisé'.

Macros de substitution de champs Paquetage

Ce tableau répertorie les macros de substitution de champ Paquetage .

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros Paquetage

Nom de la macro	Description
packageRésumé	dialogue ' Paquetage ' : Résumé.
packageAlias	dialogue ' Paquetage ' : Alias.
packageAuteur	dialogue ' Paquetage ' : Auteur.
packageComplexité	dialogue ' Paquetage ' : Complexité.
GUIDdupaquet	Le GUID unique pour le Paquetage actuel.
packageKeywords	dialogue ' Paquetage ' : Mots-clés.
paquetLangue	dialogue ' Paquetage ' : Langue.
nom du paquet	dialogue ' Paquetage ' : Nom.
chemindupaquet	La string représentant la hiérarchie des Paquetages , pour la Class dans la portée. Chaque nom Paquetage est séparé par un point (.).
packagePhase	dialogue ' Paquetage ' : Phase.
packageScope	dialogue ' Paquetage ' : Portée.
Statut du package	dialogue ' Paquetage ' : Statut.
packageStéréotype	dialogue ' Paquetage ' : Stéréotype.
version du package	dialogue ' Paquetage ' : Version.

Macros de substitution de champs de paramètres

Ce tableau répertorie chacune des macros de substitution de champ de paramètre.

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros de paramètres

Nom de la macro	Description
paramClassifierGUID	GUID unique pour le classificateur du paramètre actuel.
paramDefault	dialogue 'Paramètres' de l'opération : champ 'Par défaut'.
paramFixe	dialogue 'Paramètres' de l'opération : case à cocher 'Fixe'.
paramGUID	GUID unique pour le paramètre actuel.
paramIsEnum	True, si le paramètre utilise le mot-clé enum (C++).
paramKind	dialogue 'Paramètres' de l'opération : champ 'Kind'.
nomParam	dialogue 'Paramètres' de l'opération : champ 'Nom'.
paramNotes	dialogue 'Paramètres' de l'opération : champ ' Notes '.
paramQualType	Le type de paramètre qualifié par le chemin de l'espace de noms (si vous générez des espaces de noms) et le chemin du classificateur (délimité par des points). Si le classificateur de paramètres n’a pas été défini, est équivalent à la macro paramType.
paramType	dialogue 'Paramètres' de l'opération : champ ' Type '.

Macros de substitution de champs problématiques

Ce tableau répertorie chacune des macros de substitution de champ Problème.

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros problématiques

Nom de la macro	Description
problèmeCompletedBy	Boîte dialogue « Maintenance », onglet « Problèmes d'éléments » : complété par.
problèmeCompletedDate	Boîte dialogue « Maintenance », onglet « Problèmes d'éléments » : terminé.
problèmeHistorique	Boîte dialogue 'Maintenance', onglet 'Problèmes d'éléments' : Historique.
Nom du problème	Boîte dialogue 'Maintenance', onglet 'Problèmes d'élément' : Nom.
problèmeNotes	Boîte dialogue 'Maintenance', onglet 'Problèmes d'éléments' : Description.
problèmePriorité	Boîte dialogue 'Maintenance', onglet 'Problèmes d'éléments' : Priorité.
problèmeRaisedBy	Boîte dialogue « Maintenance », onglet « Problèmes d'élément » : soulevé par.
problèmeRaisedDate	Boîte dialogue « Maintenance », onglet « Problèmes d'élément » : soulevé.
état du problème	Boîte dialogue 'Maintenance', onglet 'Problèmes d'éléments' : Statut.
versionduproblème	Boîte dialogue 'Maintenance', onglet 'Problèmes d'éléments' : Version.

Macros de substitution de champs d’exigence

Ce tableau répertorie chacune des macros de substitution de champ d'exigence avec une description du résultat.

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros d'exigence

Nom de la macro	Description
exigenceDifficulté	dialogue ' Propriétés ' : Onglet 'Exiger' : Difficulté.
exigenceLastUpdated	Boîte dialogue ' Propriétés ' : Onglet 'Exiger' : Dernière mise à jour.
nomexigence	Boîte dialogue ' Propriétés ' : Onglet 'Exiger' : Brève description.
exigenceNotes	Boîte dialogue ' Propriétés ' : Onglet 'Exiger' : Notes .
exigencePriorité	Boîte dialogue ' Propriétés ' : Onglet 'Exiger' : Priorité.
exigenceStatus	Boîte dialogue ' Propriétés ' : Onglet 'Exiger' : Statut.
Type d'exigence	Boîte dialogue ' Propriétés ' : Onglet 'Exiger' : Type .

Macros de substitution de champs de ressources

Ce tableau répertorie chacune des macros de substitution de champ de ressource.

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros de ressources

Nom de la macro	Description
ressourceAllocatedTime	Fenêtre Allocation des ressources : Alloué Time.
dateFinressource	Fenêtre Allocation de ressources : Date de fin.
ressourceExpectedTime	Fenêtre Allocation des ressources : Temps prévu.
ressourceExpendedTime	Fenêtre Allocation des ressources : Temps dépensé.
ressourceHistorique	Fenêtre Allocation de ressources : Historique.
nom de la ressource	Fenêtre Allocation de ressources : Ressource.
ressourceNotes	Fenêtre Allocation des ressources : Description.
ressourcePourcentageCompleted	Fenêtre Allocation de ressources : Terminé (%).
ressourceRôle	Fenêtre Allocation de ressources : Rôle.
dateDébut de la ressource	Fenêtre Allocation des ressources : Date Démarrer .

Macros de substitution de champs de risque

Ce tableau répertorie chacune des macros de substitution du champ Risque.

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros de risque

Nom de la macro	Description
Nom du risque	Fenêtre Risques : Risque.
risqueNotes	Fenêtre Risques : (Notes).
Type de risque	Fenêtre Risques : Type .
Poids du risque	Fenêtre Risques : Poids.

Macros de substitution de champs de scénario

Ce tableau répertorie chacune des macros de substitution de champ Scénario avec une description du résultat.

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Macros de scénario

Nom de la macro	Description
GUID du scénario	L' ID unique d'un scénario. Identifie le scénario sans ambiguïté dans un modèle.
Nom du scénario	Boîte dialogue ' Propriétés ', onglet 'Scénario' : Scénario.
scénarioNotes	Boîte dialogue ' Propriétés ', onglet 'Scénario' : (Notes).
Type de scénario	Boîte dialogue ' Propriétés ', onglet 'Scénario' : Type .

Macros de substitution Valeur Étiquetée

Les macros Valeur Étiquetée sont une forme spéciale de macros de substitution de champ, qui donnent accès aux balises d'éléments et aux Valeur Étiquetés correspondantes. Ils peuvent être utilisés de deux manières :

- Remplacement direct
- Substitution conditionnelle

Remplacement direct

Cette forme de macro remplace directement la valeur de la balise nommée dans la sortie.

Structure : %<macroName>:"<tagName>"%

<macroName> peut être l'un des éléments suivants :

- attTag
- baliseclasse
- connecteurDestElemTag
- connecteurDestTag
- connecteurSourceElemTag
- connecteurSourceTag
- connecteurTag
- lienAttTag
- lienTag
- baliseop
- packageTag
- paramTag

Cela correspond aux balises des attributs, Classes, opérations, Paquetages , paramètres, connecteurs aux deux extrémités, éléments aux deux extrémités des connecteurs et connecteurs incluant la fin de l'attribut.

<tagName> est une string représentant le nom de balise spécifique.

Exemple

%opTag : "attribut"%

Substitution conditionnelle

Cette forme de macro imite la substitution conditionnelle définie pour les macros de substitution de champ.

Structure : %<macroName>:"<tagName> " (== "<test> ") ? <subTrue> (: <subFalse>) %

Note :

- <macroName> et <tagName> sont tels que définis ici
- (<text>) indique que <text> est facultatif
- <test> est une string représentant une valeur possible pour la macro
- <subTrue> et <subFalse> peuvent être une combinaison de chaînes entre guillemets et du mot clé valeur ; lorsque la

valeur est utilisée, elle est remplacée par la valeur de la macro dans la sortie

Exemples

%opTag : "opInline" ? "en ligne" : " "%

%opTag : "opInline" ? "en ligne"%

%classTag : "unsafe" == "vrai" ? "peu sûr" : " "%

%classTag : "unsafe" == "vrai" ? "peu sûr"%

Les macros Valeur Étiquetée utilisent la même convention de dénomination que les macros de substitution de champs.

Macros de substitution de paramètres Gabarit

Si vous souhaitez donner accès dans un gabarit de transformation aux données concernant la transformation de substitution de paramètre de liaison d'un connecteur Gabarit Binding dans le modèle, vous pouvez utiliser les macros de substitution de paramètre Gabarit . Les noms des macros sont en boîtier Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée ; sinon la valeur est vide.

Macros de substitution de paramètres Gabarit

Nom de la macro	Description
paramètreSubstitutionFormal	dialogue « Propriétés de liaison Gabarit », onglet « Paramètre de liaison », panneau « Substitution(s) de paramètres » : Nom formel du paramètre Gabarit .
paramètreSubstitutionRéel	dialogue « Propriétés de liaison Gabarit », onglet « Paramètre de liaison », panneau « Substitution(s) de paramètre(s) » : nom/expression du paramètre réel.
paramètreSubstitutionActualClassifier	dialogue ' Gabarit Binding Propriétés ', onglet 'Paramètre de liaison', panneau 'Substitution(s) de paramètres' : classificateur de paramètres réel.

Macros de substitution de champ Test

Ce tableau répertorie chacune des macros de substitution de champ Test avec une description du résultat.

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon la valeur est vide.

Test les macros

Nom de la macro	Description
testAcceptanceCritères	dialogue Tester : Critères d'acceptation.
testCheckedBy	Fenêtre Cas Test : Vérifié par.
testDateExécuter	Fenêtre Cas Test : Dernier Exécuter .
Classe de test	Fenêtre Cas Test : Classe Test (le type de test défini : Unité, Intégration, Système, Acceptation, Inspection, Scénario)
testEntrée	dialogue Tester : Entrée.
nom du test	Fenêtre Scénarios Test : Test .
Notes de test	Fenêtre Cas Test : Description.
résultats de test	dialogue Tester : Résultats.
testRunBy	Fenêtre Cas Test : Exécuter By. (Les valeurs sont dérivées des définitions des auteurs du projet dans la dialogue « Personnes » - « Paramètres > Données de référence > Types Modèle > Personnes > Auteurs du projet ».)
Statut du test	Fenêtre Cas Test : Statut.
Type de test	Fenêtre Scénarios Test : Type .

Macros de fonctions

Les macros de fonctions constituent un moyen pratique de manipuler et de formater divers éléments de données d'éléments. Chaque macro de fonction renvoie une string de résultat . Il existe deux manières principales d'utiliser les résultats des macros de fonctions :

- Substitution directe de la string renvoyée dans la sortie, telle que : %TO_LOWER(attName)%
- Stockage de la string renvoyée dans le cadre d'une définition de variable telle que : $name = %TO_LOWER(attName)%

Les macros de fonctions peuvent prendre des paramètres, qui peuvent être transmis aux macros sous la forme :

- Littéraux String , placés entre guillemets doubles
- Macros de substitution directe sans les signes de pourcentage englobants
- Références variables
- Littéraux numériques

Plusieurs paramètres sont transmis à l'aide d'une liste séparée par des virgules.

Les macros de fonction sont nommées selon le style Tout en majuscules, comme dans :

%CONVERT_SCOPE(opScope)%

Les macros de fonctions disponibles sont décrites ici. Les paramètres sont indiqués par des crochets, comme dans :

FUNCTION_NAME([param]).

CONVERT_SCOPE([umlScope])

À utiliser avec les langues prises en charge, pour convertir [umlScope] en mot-clé de portée approprié pour la langue générée. Ce tableau montre la conversion de [umlScope] par rapport à la langue donnée.

Langue	Conversions
C++	Paquetage ==> public Publique ==> publique Privé ==> privé Protégé ==> protégé
C#	Paquetage ==> interne Publique ==> publique Privé ==> privé Protégé ==> protégé
Delphes	Paquetage ==> protégé Publique ==> publique Privé ==> privé Protégé ==> protégé
Java	Paquetage ==> {vide} Publique ==> publique Privé ==> privé Protégé ==> protégé

PHP	Paquetage ==> public Publique ==> publique Privé ==> privé Protégé ==> protégé
VB	Paquetage ==> Protégé Publique ==> Publique Privé ==> Privé Protégé ==> Protégé
VB.Net	Paquetage ==> Ami Publique ==> Publique Privé ==> Privé Protégé ==> Protégé

COLLECTION_CLASS([langue])

Donne la classe de collection appropriée pour la langue spécifiée pour l'attribut lié actuel.

CSTYLE_COMMENT([wrap_length])

Convertit les notes de l'élément actuellement dans la portée en commentaires simples de style C, en utilisant /* et */.

DELPHI_PROPERTIES([portée], [separator] , [retrait])

Génère une propriété Delphi.

DELPHI_COMMENT([wrap_length])

Convertit les notes de l'élément actuellement dans la portée en commentaires Delphi.

EXEC_ADD_IN(, [nom_fonction],, ...,)

Appelle une fonction Add-In Enterprise Architect , qui peut renvoyer une string de résultat.

[addin_name] et [function_name] spécifient les noms du Add-In et de la fonction à appeler.

Les paramètres de la fonction Add-In peuvent être spécifiés via les paramètres [prm_1] à [prm_n].

$result = %EXEC_ADD_IN("MyAddin", "ProcessOperation", classGUID, opGUID)%

Toute fonction qui doit être appelée par la macro EXEC_ADD_IN doit avoir deux paramètres : un object EA.Repository et un tableau Variant qui contient tous les paramètres supplémentaires de l'appel EXEC_ADD_IN. Le type de retour doit être Variant.

Fonction publique ProcessOperation(Référentiel As EA.Repository , args As Variant) As Variant

TROUVER([src], [sousChaîne])

Position de la première instance de [subString] dans [src] ; -1 si aucun.

GET_ALIGNMENT()

Renvoie une string dans laquelle tout le texte de la ligne de sortie actuelle est converti en espaces et tabulations.

JAVADOC_COMMENT([wrap_length])

Convertit les notes de l'élément actuellement dans la portée en commentaires de style javadoc.

GAUCHE([src], [nombre])

Les premiers [count] caractères de [src].

LONGUEUR([src])

Longueur de [src]. Renvoie une string .

MATH_ADD(x,y) MATH_MULT(x,y) et MATH_SUB(x,y)

Dans un code gabarit ou DDL gabarit , ces trois macros réalisent respectivement les fonctions mathématiques de :

- Ajout (x+y)
- Multiplication (x*y) et
- Soustraction (xy)

Les arguments x et y peuvent être des entiers ou des variables, ou une combinaison des deux. Considérez ces exemples, tels qu'utilisés dans un gabarit de « Classe » pour la génération de code C++ :

- $a = %MATH_ADD(3,4)%
- $b = %MATH_SUB(10,3)%
- $c = %MATH_MULT(2,3)%
- $d = %MATH_ADD($a,$b)%
- $e = %MATH_SUB($b,$c)%
- $f = %MATH_MULT($a,$b)%
- $g = %MATH_MULT($a,10)%
- $h = %MATH_MULT(10,$b)%

Ceux-ci calculent, dans la même séquence, à :

- une = 3 + 4 = $une
- b = 10 - 3 = $b
- c = 2 * 3 = $c
- d = a + b = $d
- e = b - c = $e
- f = a * b = $f
- g = une * 10 = $g
- h = 10 * b = $h

Lorsque le code est généré, le fichier .h (pour C++) contient ces chaînes correspondantes :

- une = 3 + 4 = 7
- b = 10 - 3 = 7
- c = 2 * 3 = 6
- d = une + b = 14
- e = b - c = 1
- f = une * b = 49
- g = une * 10 = 70
- h = 10 * b = 70

MID([src], [début]) MID([src], [début], [compte])

Sous-chaîne de [src] commençant à [start] et incluant [count] caractères. Où [count] est omis, le reste de la string est inclus.

PI([option], [valeur], {[option], [valeur]})

Ensembles le PI pour le gabarit actuel à [valeur]. Les valeurs valides pour [valeur] sont :

- " \n"
- " \t "
- " "
- ""

<option> contrôle le moment où le nouveau PI prend effet. Les valeurs valides pour <option> sont :

- I, Immédiat : le nouveau PI est généré avant la prochaine ligne gabarit non vide
- N, Next : le nouveau PI est généré après la prochaine ligne gabarit non vide

Plusieurs paires d'options sont autorisées en un seul appel. Un exemple de situation dans laquelle cela serait utilisé est celui où un mot-clé est toujours sur une nouvelle ligne, comme illustré ici :

%PI=" "%

%classRésumé ? "abstrait"%

%if classTag:"macro" != " "%

%PI("I", " \n", "N", " ")%

%classTag : "macro"%

%fin si%

classe

%nom du cours%

Pour plus de détails, voir *La macro d'instruction de traitement (PI)* .

PROCESS_END_OBJECT([template_name])

Permet aux Classes qui sont une Classe plus éloignée de la Classe de base d'être transformées en objets (tels que les attributs, les opérations, Paquetages , les paramètres et les colonnes) de la Classe de base. [template_name] fait référence au gabarit de travail qui stocke temporairement les données.

REMOVE_DUPLICATES([source], [separator])

Où [source] est une liste séparée par [separator] ; cela supprime toutes les chaînes en double ou vides.

REMPLACER([string], [ancien], [nouveau])

Remplace toutes les occurrences de [old] par [new] dans la string donnée <string>.

RESOLVE_OP_NAME()

Résout les conflits dans les noms d'interface où deux interfaces de méthode d'origine portent le même nom.

RESOLVE_QUALIFIED_TYPE() RESOLVE_QUALIFIED_TYPE([separator]) RESOLVE_QUALIFIED_TYPE([separator] , [par défaut])

Génère un type qualifié pour l'attribut actuel, l'attribut lié, le parent lié, l'opération ou le paramètre. Permet la spécification d'un séparateur autre que. et une valeur par défaut lorsqu'une certaine valeur est requise.

DROITE([src], [compte])

Les derniers [count] caractères de [src].

TO_LOWER([string])

Convertit [string] en minuscules.

TO_UPPER([string])

Convertit [string] en majuscules.

TRIM([string]) TRIM([string], [trimChars])

Supprime les espaces blancs de fin et de début de [string]. Si [trimChars] est spécifié, tous les caractères de début et de fin de l'ensemble de <trimChars> sont supprimés.

TRIM_LEFT([string]) TRIM_LEFT([string], [trimChars])

Supprime les premiers caractères spécifiés de <string>.

TRIM_RIGHT([string]) TRIM_RIGHT([string], [trimChars])

Supprime les caractères de fin spécifiés de <string>.

VB_COMMENT([wrap_length])

Convertit les notes de l'élément actuellement dans la portée en commentaires de style Visual Basic.

WRAP_COMMENT([commentaire], [wrap_length], [indent], [start_string])

Enveloppe le texte [commentaire] à la largeur [wrap_length] en mettant [indent] et [start_string] au début de chaque ligne.

$behavior = %WRAP_COMMENT(opBehavior, "40", " " , " // ")%

<wrap_length> doit toujours être transmis sous forme de string , même si WRAP_COMMENT traite ce paramètre comme un integer .

WRAP_LINES([texte], [wrap_length], [start_string] {, [end_string] })

Enveloppe [text] comme désigné par [wrap_length], en ajoutant [start_string] au début de chaque ligne et [end_string] à la fin de la ligne si cela est spécifié.

XML_COMMENT([wrap_length])

Convertit les notes de l'élément actuellement dans la portée en commentaires de style XML.

Macros de contrôle

Des macros de contrôle permettent de contrôler le traitement et le formatage des gabarits . Les types de base de macro de contrôle comprennent :

- La macro de liste, pour générer plusieurs fonctionnalités d'éléments, telles que des attributs et des opérations
- Les macros de branchement, qui forment des constructions if-then-else pour exécuter conditionnellement des parties d'un gabarit
- La macro PI pour formater les nouvelles lignes dans la sortie, qui prend effet à partir de la prochaine ligne non vide
- Une macro de fonction PI qui permet de définir PI sur une variable et ajoute la possibilité de définir le PI généré avant la ligne suivante
- Les macros de synchronisation

En général, les macros de contrôle sont nommées selon la casse Camel.

Liste des macros

Si vous devez parcourir ou parcourir un ensemble d'objets contenus dans ou sous l' object actuel, vous pouvez le faire à l'aide de la macro *%list* . Cette macro effectue une passe itérative sur tous les objets dans la portée du gabarit actuel et appelle un autre gabarit pour traiter chacun d'entre eux.

La structure de base est la suivante :

%list=<TemplateName> @separator=<string> @indent=<string> (<conditions>) %

où <string> est une string littérale entre guillemets et <TemplateName> peut être l'un de ces noms gabarit :

- Attribut
- AttributImpl
- Classe
- ClasseBase
- ClasseImpl
- ClassInitializer
- Interface de classe
- Contrainte
- Gabarit personnalisé (gabarits personnalisés vous permettent de définir vos propres gabarits)
- Effort
- Classe intérieure
- Classe intérieureImpl
- Fichier lié
- Métrique
- Namespace
- Opération
- OpérationImpl
- Paramètre
- Problème
- Exigence
- Ressource
- Risque
- Scénario
- Test

<conditions> est facultatif et ressemble aux conditions des instructions « if » et « elseIf ».

Exemple

Dans une transformation Classe, la Classe peut contenir plusieurs attributs ; cet exemple appelle la transformation Attribut et génère le résultat du traitement de la transformation pour chaque attribut de la classe concernée. La liste résultante sépare ses éléments par une seule nouvelle ligne et les indente respectivement de deux espaces. Si la classe concernée avait des attributs stéréotypés, ils seraient générés à l'aide du gabarit spécialisé approprié.

%list="Attribut" @separator="\n" @indent=" "%

L'attribut separator, noté @separator, spécifie l'espace qui doit être utilisé entre les éléments de la liste, à l'exclusion du dernier élément de la liste.

L'attribut indent, noté @indent, spécifie l'espace par lequel chaque ligne de la sortie générée doit être indentée.

Cas spéciaux

Il existe quelques cas particuliers à prendre en compte lors de l'utilisation de la macro %list :

- Si le gabarit d'attribut est utilisé comme argument de la macro %list, cela génère également des attributs dérivés des associations en exécutant le gabarit LinkedAttribute approprié.
- Si le gabarit ClassBase est utilisé comme argument de la macro %list, cela génère également des bases de classes dérivées des liens dans le modèle en exécutant le gabarit LinkedClassBase approprié.
- Si le gabarit ClassInterface est utilisé comme argument de la macro %list, cela génère également des bases de classes dérivées des liens dans le modèle en exécutant le gabarit LinkedClassInterface approprié.
- Si InnerClass ou InnerClassImpl est utilisé comme argument de la macro %list, ces Classes sont générées en utilisant respectivement les gabarits Class et ClassImpl ; ces arguments indiquent que les gabarits doivent être traités en fonction des classes internes de la classe concernée

Macros de branchement

Les macros de branchement fournissent des constructions if-then-else. Le CTF supporte une forme limitée de branchement via ces macros :

- si
- sinonSi
- autre
- fin si
- endTemplate (qui sort du gabarit courant)

La structure de base des macros if et elseIf est :

%if <test> <opérateur> <test>%

où <operator> peut être l'un des suivants :

- ==
- !=
- < (comparaison mathématique, moins que)
- > (comparaison mathématique, supérieur à)
- <= (comparaison mathématique, inférieur ou égal à)
- >= (comparaison mathématique, supérieur ou égal à)

et <test> peut être l'un des éléments suivants :

- une string littérale, entourée de guillemets doubles
- une macro de substitution directe, sans les signes de pourcentage englobants
- une référence variable

Note que si vous utilisez l'un des opérateurs de comparaison mathématiques, <test> doit être un nombre décimal au format string .

Les branches peuvent être imbriquées et plusieurs conditions peuvent être spécifiées à l'aide de l'une des options suivantes :

- et, ou
- ou

Lorsque vous spécifiez plusieurs conditions, « et » et « ou » ont le même ordre de priorité et les conditions sont traitées de gauche à droite.

Si les instructions conditionnelles sur les chaînes sont sensibles à la casse, 'a String ' n'est pas égal à 'A STRING'. Par conséquent, dans certaines situations, il est préférable de définir la variable $str=TO_LOWER(variable) ou TO_UPPER(variable) puis de comparer à un cas spécifique.

Les macros ne sont pas prises en charge dans les instructions conditionnelles. Il est préférable d'attribuer les résultats d'une macro (string) à une variable, puis d'utiliser la variable dans la comparaison.

$fldType = % TO_LOWER ($paramètre1)%

$COMMENT = "Utiliser les 4 premiers caractères pour les types de champs Date et Heure"

$fldType4 = % GAUCHE ($fldType, 4)%

%if $fldType4 == "date"%

Dateheure

%fin si%

Celui-ci prend un paramètre de valeur « Datetime », « DATETIME » ou « Date », et renvoie « Datetime ».

Les macros endif ou endTemplate doivent être utilisées pour signifier la fin d'une branche. De plus, la macro endTemplate provoque le retour immédiat du gabarit , si la branche correspondante est en cours d'exécution.

Exemple 1

%if elemType == "Interface"%

;

%autre%

%OpérationCorps%

%fin si%

Dans ce cas:

- Si l'elemType est "Interface", un point-virgule est renvoyé
- Si l'elemType n'est pas "Interface", un gabarit appelé Operation Body est appelé

Exemple 2

$bases="ClasseBase"

$interfaces=""%

%if $bases != " " et $interfaces != ""%

: $bases, $interfaces

%elseSi $bases != ""%

: $bases

%elseSi $interfaces != ""%

: $interfaces

%fin si%

Dans ce cas, le texte renvoyé est « : ClassBase ».

Conditions utilisant une valeur booléenne

Lors de la configuration d'un branchement à l'aide de conditions impliquant une case à cocher système (champs booléens), telles que Attribute.Static (attStatic), l'instruction conditionnelle serait écrite comme suit :

%if attStatic == "T"%

Par exemple:

% si attCollection == "T" ou attOrderedMultiplicity == "T" %

% modèle de fin %

Macros de synchronisation

Les macros de synchronisation sont utilisées pour fournir des conseils de formatage à Enterprise Architect lors de l'insertion de nouvelles sections dans le code source, lors de la synchronisation directe. Les valeurs des macros de synchronisation doivent être définies dans le Fichier gabarits .

La structure de configuration des macros de synchronisation est la suivante :

%<nom>=<valeur>%

où <nom> peut être l'une des macros répertoriées ici et <valeur> est une string littérale entourée de guillemets doubles.

Macros de synchronisation

Nom de la macro	Description
syncNewClassNotesEspace	Espace pour ajouter une nouvelle note de cours. valeur par défaut : \n.
syncNewAttributeNotesSpace	Espace à ajouter à une nouvelle note d'attribut. valeur par défaut : \n.
syncNewOperationNotesSpace	Espace pour ajouter une nouvelle note d'opération. valeur par défaut : \n.
syncNewOperationBodySpace	Espace à ajouter à un nouveau corps d’opération. valeur par défaut : \n.
syncNamespaceBodyIndent	Retrait appliqué aux classes dans des espaces de noms non globaux. valeur par défaut : \t.

La macro d'instruction de traitement (PI)

La macro PI (Processing Instruction) fournit un moyen de définir le texte séparateur à insérer entre les morceaux de code (qui représentent des entités) générés à l'aide d'un gabarit .

La structure de définition de l'instruction de traitement est :

%PI=<valeur>%

Dans cette structure, <value> est une string littérale entourée de guillemets doubles, avec ces options :

- " \n" - Nouvelle ligne (valeur par défaut)
- " " - Espace
- " \t" - Onglet
- " " - Nul

Par défaut, le PI est configuré pour générer une nouvelle ligne (\n) pour chaque substitution non vide, dont le comportement peut être modifié en réinitialisant la macro PI. Par exemple, la déclaration d'attribut d'une classe en code VB simple serait générée sur une instruction sur une seule ligne (sans nouvelles lignes). Ces propriétés sont dérivées des propriétés Class-Attribute du modèle à générer, par exemple :

Const privé PrintFormat As String = "Portrait"

Le gabarit pour générer cela commence avec le PI défini sur un espace plutôt que sur une nouvelle ligne :

% PI = " " %

% CONVERT_SCOPE (attScope)%

% fin si %

% si attConst == "T" %

Const

% fin si %

Lors de la transformation, attscope renvoie le mot-clé VB 'Private' et attConst renvoie 'Const' sur la même ligne espacée d'un seul espace (convenant à l'exemple de définition VB Class.Attribute précédent).

Alternativement, lors de la génération d'une classe, vous souhaiterez peut-être que la déclaration de classe, les notes et le corps de la classe soient tous séparés par des lignes doubles. Dans ce cas, %PI est défini sur « /n/n » pour renvoyer un interligne double :

% PI = " \n\n" %

% Déclaration de classe %

% Notes de cours %

% Corps de classe %

Caractéristiques de l'IP

- Les lignes vides n'ont aucun effet sur la sortie
- Toute ligne contenant une macro qui produit un résultat vide n'entraîne pas de séparateur PI (espace/nouvelle ligne)
- La dernière entrée ne renvoie pas de PI ; par exemple, %Classbody% n'a pas de double ligne ajoutée après le corps

Macros de génération de code pour Statemachines Exécutables

Les gabarits répertoriés ici sont disponibles via l'éditeur de Code Gabarit (l'option de ruban 'Développer > Code source > Options > Modifier le Code Gabarits ') ; sélectionnez 'STM_C++_Structured' dans le champ 'Langue'.

Les gabarits sont structurés comme indiqué :

StmContextStateMachineEnum

StmStateMachineEnum

StmContextStateEnum

StmAllStateEnum

StmContextTransitionEnum

StmTransitionEnum

StmContextEntryEnum

StmAllEntryEnum

StmContextStateMachineStringToEnum

StmStateMachineStringToEnum

StmContextStateEnumToString

StmStateEnumToString

StmContextTransitionEnumToString

StmTransitionEnumToString

StmContextStateNameToGuid

StmStateNameToGuid

StmContextTransitionNameToGuid

StmTransitionNameToGuid

StmContextDefinition

StmStateMachineEnum

StmAllStateEnum

StmTransitionEnum

StmAllEntryEnum

StmAllRegionVariableInitialize

StmStateWithDeferredEvent

StmDeferredEvent
StmTransitionProcMapping
StmTransitionProc
StmTransitionSortie
Entrée StmTransition
StmTargetOutgoingTransition
StmTargetParentSubmachineState
StmStateProcMapping
StmStateProc
StmStateEntry
StmSortantTransition
StmConnectionPointReferenceEntry
StmParameterizedInitial
StmSubMachineInitial
StmRegionInitial
StmRegionDésactivé
StmStateExitProc
StmStateTransition
StmStateEvent
StmStateTriggeredTransition
StmStateCompletionTransition
StmStateIncomingTransition
StmStateOutgoingTransition
StmSubmachineStateExitEvent
StmVertexOutgoingTransition
StmConnectionPointReferenceExitEvent
StmStateExitEvent
StmVertexOutgoingTransition
StmAllRegionVariable
StmStateMachineStringToEnum
StmStateMachineRun
StmStateInitialData
Entrée StmStateMachine
StmSortantTransition
StmStateMachineRunInitial
StmStateMachineInitial
StmStateMachineRuns

StmContextManager

StmSimulationManager
StmContextInstanceDeclaration

StmContextInstance
StmContextVariableRunstate
StmContextInstanceAssociation
StmContextInstanceEffacer

StmEventProxy
StmSignalEnum
StmContextJoinEventEnum
StmJoinEventEnum
StmEventEnum
StmSignalDéfinition
StmSignalAttributeAssignment
StmSignalAttribute
StmSignalInitialize
StmEventStringToEnum
StmEventEnumToString
StmEventNameToGuid

StmConsoleManager
StmContextInstanceDeclaration
StmContextInstance
StmContextVariableRunstate
StmContextInstanceAssociation
StmContextInstanceEffacer

StmStateMachineStrongToEnum

StmInitialForTransition

StmVertextOutgoingTransition

StmSendEvent

StmBroadcastEvent

StmContextRef

Signal et événement

Nom de la macro	Description
stmEventEnum	Le nom de l'événement avec le préfixe 'ENUM_', tout en majuscules.

StmEventGuid	Le GUID de l'événement.
stmEventName	Le nom de l'événement avec les espaces et les astérisques supprimés.
stmEventVariable	Le nom de l'événement avec le préfixe 'm_' en minuscules.
stmIsSignalEvent	Est « T » si l'élément est un SignalEvent.
stmSignalEnum	Le nom du Signal avec le préfixe 'ENUM_', tout en majuscules.
stmSignalFirstEvent	Le nom de l'événement avec le préfixe 'ENUM_', tout en majuscules.
stmSignalGuid	Le GUID du signal.
stmSignalName	Le nom du signal avec les espaces et les astérisques supprimés.
stmSignalVariable	Le nom du Signal avec le préfixe 'm_' en minuscule.
stmTriggerName	Transition Propriétés : Le nom du Déclencheur .
stmTriggerSpecification	Propriétés de Transition : La spécification du Déclencheur .
stmTriggerType	Propriétés de Transition : Le type du Déclencheur .

Contexte

Nom de la macro	Description
stmContextName	Le nom de la classe avec les espaces et les astérisques supprimés.
stmContextQualName	Nom qualifié de la classe pour laquelle le code est généré.
stmContextVariableName	
stmContextFileName	Le nom du fichier de sortie pour la classe pour laquelle le code est généré.

Écriture de l'état d'exécution Object dans l'initialisation Statemachine

Nom de la macro	Description
stmContextVariableRunstateName	
stmContextVariableRunstat	

eValue	
stmContextHasStatemachine	Est 'T' si le contexte actuel a un ou plusieurs Statemachines .
stmHasHistoryPattern	Est 'T' si la Statemachine a un History Motif .
stmHasTerminatePattern	Est 'T' si la Statemachine a un Terminate Motif .
stmHasDeferredEventPattern	Est 'T' si la Statemachine a un Motif d'événement différé.
stmHasSubmachinePattern	Est 'T' si la Statemachine a un Submachine Motif .
stmHasOrthogonalPattern	Est 'T' si la Statemachine a un Motif orthogonal.

Statemachine

Nom de la macro	Description
stmStatemachineName	Le nom de la Statemachine avec les astérisques et les espaces supprimés.
stmStatemachineEnum	Le nom de la Statemachine plus 'ENUM_' plus le nom de la Statemachine en majuscules.
stmStatemachineGuid	Le GUID de l'élément Statemachine .
stmStateCount	Le nombre d'éléments State dans la Statemachine .
stmSubmachineInitialCount	Nombre d'éléments initiaux dans l'élément Sub Machine State .
stmStatemachineHasSubmachineState	Est « T » si la Statemachine a au moins un SubMachine State .
stmStatemachineInitialCount	Le nombre d'éléments initiaux dans la Statemachine .

Région

Nom de la macro	Description
stmRegionEnum	Le nom de la région State plus « ENUM_ » plus le nom de la région State en majuscules.
stmRegionFQName	Le nom complet de la région State .

stmRegionName	Le nom de la région State sans les espaces et les astérisques.
stmRegionVariable	Le nom de la région State avec le préfixe « m_ » en minuscules.
stmRegionFQVariable	Le nom complet de la région State avec le préfixe « m_ » en minuscules.
stmRegionGuid	Le GUID de la Région.
stmRegionInitial	
stmRegionIsOwnedByStateMachine	Est « T » si la région appartient à un Statemachine .

Transition

Nom de la macro	Description
stmTransitionEnum	Le nom de la Transition avec le préfixe 'ENUM_', plus le nom de la Transition en majuscules.
stmTransitionGuid	Le GUID de la transition.
stmTransitionName	Le nom de la transition avec les espaces et les astérisques supprimés.
stmTransitionSourceGuid	Le GUID de l'élément Source dans la transition.
stmTransitionTargetGuid	GUID de l'élément Target dans la transition.
stmTransitionVariable	Le nom de la Transition avec le préfixe 'm_' en minuscule.
stmTransitionSourceVariable	
stmTransitionTargetVariable	
stmTransitionFQVariable	
stmSourceVertexEnum	Le nom du sommet source de la transition plus '_ENUM' plus le nom du sommet source de la transition en majuscules.
stmTargetVertexEnum	Le nom du sommet cible de la transition plus '_ENUM' plus le nom du sommet cible de la transition en majuscules.
stmSourceIsInitial	Est « T » si la source de la transition est une initiale.
stmSourceIsState	Est 'T' si la source de la transition est un State .

stmSourceIsEntryPoint	Est « T » si la source de la transition est un point d'entrée.
stmSourceIsExitPoint	Est « T » si la source de la transition est un point de sortie.
stmSourceIsFork	Est « T » si la source de la transition est un fork.
stmSourceIsJoin	Est « T » si la source de la transition est un élément Join.
stmTargetIsFinalState	Est « T » si la cible de la transition est un élément State final.
stmTargetIsExitPoint	Est « T » si la cible de la transition est un élément de point de sortie.
stmTargetIsState	Est « T » si la cible de la transition est un élément State .
stmTargetIsChoice	Est « T » si la cible de la transition est un élément Choice.
stmTargetIsJunction	Est « T » si la cible de la transition est un élément de jonction.
stmTargetIsEntryPoint	Est « T » si la cible de la transition est un élément de point d'entrée.
stmTargetIsConnectionPoi ntReference	Est « T » si la cible de la transition est un élément de référence de point de connexion.
stmTargetIsFork	Est « T » si la cible de la transition est un élément Fork.
stmTargetIsJoin	Est « T » si la cible de la transition est un élément Join.
stmTransitionEffet	L'effet de la transition.
stmTransitionGuard	La Garde de la Transition.
stmTransitionKind	Le type ou le type de transition.
stmTargetInitialTransition	
stmTargetIsSubmachineSta te	Est « T » si la cible de la transition est un State de sous-machine.
stmSourceStateEnum	Le nom de l'état source de la transition avec le préfixe '_ENUM' en majuscule.
stmTargetStateEnum	Le nom de l'état cible de la transition, avec le préfixe « _ENUM » en majuscule.
stmTargetVertexFQName	Le nom complet du sommet cible de la transition.
stmTargetIsDeepHistory	Est « T » si la cible de la transition est un State d'histoire profonde.
stmTargetIsShallowHistory	Est « T » si la cible de la transition est un State d'histoire peu profonde.
stmTargetIsTerminate	Est « T » si la cible de la transition est un élément Terminate.
stmParentIsStateMachine	Est 'T' si le sommet est un point d'entrée ou un point de sortie, ou si le conteneur est un Statemachine .

stmSourceParentStateEnum	
stmTargetParentStateEnum	
stmTargetSubmachineEnum	
stmTargetRegionIndex	
stmIsSelfTransition	Est « T » si la source de la transition est la même que sa cible.
stmHistoryOwningRegionInitialTransition	
stmDefaultHistoryTransition	

Sommet et State

Nom de la macro	Description
stmVertexName	Le nom du sommet.
stmStateName	Le nom de l' State .
stmVertexGuid	Le GUID du sommet.
stmVertexFQName	Le nom complet du sommet.
stmStateFQName	Le nom complet de l' State .
stmVertexType	Le type du sommet ; l'un des « State », « FinalState », « Pseudostate », « ConnectionPointReference » ou « » (vide).
stmPseudostateKind	Le genre de pseudo-état ; l'un des éléments suivants : "initial", "deepHistory", "shallowHistory", "join", "fork", "junction", "choice", "entryPoint", "exitPoint" ou "terminate".
stmPseudostateName	Le nom du pseudo-état.
stmPseudostateVariable	Le nom du pseudo-état avec le préfixe « m_ » en minuscules.
stmPseudostateStateMachineName	Le nom de la Statemachine pseudo-étatique.
stmPseudostateStateMachineVariable	Le nom de la Statemachine pseudo-étatique avec le préfixe « m_ » en minuscules.

stmVertexVariable	Le nom du sommet avec le préfixe « m_ » en minuscule.
stmVertexEnum	Le nom du sommet plus '_ENUM' plus le nom du sommet en majuscules.
stmStateEnum	Le nom de l' State plus '_ENUM' plus le nom de l' State en majuscules.
stmConnectionPointReferenceStateName	Le nom de la référence du point de connexion.
stmConnectionPointReferenceStateVariable	Le nom de la référence du point de connexion avec le préfixe « m_ » en minuscules.
stmConnectionPointReferenceEntryCount	
stmParameterizedInitialCount	
stmInitialCountForTransition	
stmStateVariable	Le nom de l' State avec le préfixe « m_ » en minuscule.
stmStateEntryBehavior	Le comportement défini pour une opération Action 'entrée' pour un State (le texte sur l'onglet 'Comportement' pour l'opération Action 'entrée' sur la fenêtre Fonctionnalités de l'élément).
stmStateEntryCode	Le code initial défini pour une opération Action 'entrée' pour un State (le texte de l'opération Action 'entrée' dans l'onglet 'Code' du comportement).
stmStateDoBehavior	Le comportement défini pour une opération Action 'do' pour un State (le texte sur l'onglet 'Comportement' pour l'opération Action 'do' sur la fenêtre Fonctionnalités de l'élément).
stmStateDoCode	Le code initial défini pour une opération Action « faire » pour un State (le texte de l'opération Action « faire » dans l'onglet « Code » du comportement).
stmStateExitBehavior	Le comportement défini pour une opération Action 'exit' pour un State (le texte sur l'onglet 'Comportement' pour l'opération Action 'exit' sur la fenêtre Fonctionnalités de l'élément).
stmStateExitCode	Le code initial défini pour une opération Action « sortie » pour un State (le texte de l'opération Action « sortie » dans l'onglet « Code » du comportement).
stmStateSubmachineName	Le nom de la sous-machine.
stmStateSubmachineVariable	Le nom de la sous-machine avec le préfixe « m_ » en minuscules.
stmStateIsFinal	Est « T » si l' State est un FinalState.
stmStateIsSubmachineState	Est 'T' si l' State est un State de sous-machine (page ' Propriétés ' | Avancé | propriété 'isSubmachineState').

stmSubMachineEnum	Le nom de la sous-machine suivi de « _ENUM » plus le nom de la sous-machine en majuscules.
stmStateHasChildrenToJoin	
stmStateIsTransitionTarget	
stmThisIsSource	
stmThisIsSourceState	
stmStateParentIsSubmachine	Est 'T' si le conteneur du State est un Statemachine .
stmStateContainerMatchTransitionContainer	
stmVertexRegionIndex	
stmStateRegionCount	Le nombre de régions dans l' State .
stmStateInitialCount	Le nombre d'éléments initiaux dans la Statemachine .
stmVertexContainerVariable	
stmVertexParentEnum	
stmStateHasUnGuardedCompletionTransition	
stmStateEventHasUnGuardedTransition	
stmInitialTransition	

Association d'instances

Nom de la macro	Description
stmSourceInstanceName	
stmTargetInstanceName	
stmSourceRoleName	
stmTargetRoleName	

Macros de génération de code EASL

Enterprise Architect fournit un certain nombre de macros de génération de code Enterprise Architect Simulation Bibliothèque (EASL) pour générer du code à partir de modèles comportementaux. Ceux-ci sont:

- EASL_INIT
- EASL_GET
- EASLListe et
- EASL_END

EASL_INIT

La macro EASL_INIT est utilisée pour initialiser un modèle de comportement EASL. La génération du code du modèle de comportement dépend de ce modèle.

Aspect	Description
Syntaxe	%EASL_INIT(<<GUID>>)% où: • <<GUID>> est le GUID de l' Object (généralement un élément Class) qui est le propriétaire du modèle de comportement

EASL_GET

La macro EASL_GET est utilisée pour récupérer une propriété ou une collection d'un object EASL. Les objets EASL ainsi que les propriétés et collections de chaque object sont identifiés dans les rubriques *Collections EASL* et *Propriétés EASL* .

Aspect	Description
Syntaxe	$result = %EASL_GET(<<Propriété>>, <<ID du propriétaire>>, <<Nom>>)% où: • <<Property>> est l'un des éléments suivants : "Property", "Collection", "At", "Count" ou "IndexOf". • <<OwnerID>> est l' ID de l' object propriétaire pour lequel la propriété/collection doit être récupérée • <<Nom>> est le nom de la propriété ou de la collection à laquelle vous accédez • $result est la valeur renvoyée ; c'est "" si ce n'est pas une propriété valide Si « Propriété » est : • "At", alors <<OwnerID>> est l' ID d'une collection et <<Name>> est l'index de la collection pour laquelle l'élément doit être récupéré • "Count", alors <<Owner ID>> est l' ID d'une collection et <<Name>> n'est pas utilisé ; il récupérera le numéro d'article dans la collection • "IndexOf", alors <<Owner ID>> est l' ID d'une collection et <<Name>> est l' ID de l'élément de la collection ; il récupérera l'index (format string) de l'élément dans la collection

Exemple	$sPropName = %EASL_GET("Propriété", $context, "Nom")%

Liste EASL

La macro EASLList est utilisée pour restituer chaque object d'une collection EASL en utilisant le gabarit approprié.

Aspect	Description
Syntaxe	$result = %EASLList=<<TemplateName>> @separator=<<Separator>> @indent=<<indent>> @owner=<<OwnedID>> @collection=<<CollectionName>> @option1=<<OPTION1>> @option2=<<OPTION2>>..... @optionN=<<OPTIONN>>% où: • <<TemplateName>> est le nom de tout gabarit de modèle comportemental ou gabarit personnalisé • <<Separator>> est un séparateur de liste (tel que « \n ») • <<indent>> est toute indentation à appliquer au résultat • <<OwnedID>> est l' ID de l' object qui contient la collection requise • <<CollectionName>> est le nom de la collection requise • <<OPTION1>...<<OPTION99>> sont des options diverses qui peuvent être transmises sur le gabarit ; chaque option est donnée comme paramètre d'entrée supplémentaire au gabarit • $result est la valeur résultante ; c'est "" si ce n'est pas une collection valide
Exemple	$sStates = %EASLList=" State " @separator="\n" @indent="\t" @owner=$StateMachineGUID @collection=" States " @option=$sOption%

EASL_END

La macro EASL_END est utilisée pour libérer le modèle de comportement EASL.

Aspect	Description
Syntaxe	%EASL_END%

Comportementale Modèle Gabarits

- Action
- Affectation Action
- Pause Action
- Appel Action
- Action Créer

- Action Détruire
- Action si
- Boucle Action
- Action Opaque
- Action parallèle
- Action RaiseÉvénement
- Action RaiseException
- Commutateur Action
- Comportement
- Comportement du corps
- Déclaration de comportement
- Paramètre de comportement
- Argument d'appel
- Décision Action
- Condition Décision
- Logique Décision
- Tableau de Décision
- Garde
- Déclaration de propriété
- Notes sur les propriétés
- Object de propriété
- State
- Rappel State
- Énumération State
- Nom énuméré State
- Statemachine
- Historique Statemachine
- Transition
- Effet de transition
- Déclencheur

Collections EASL

Cette rubrique répertorie les collections EASL pour chacun des objets EASL, telles que récupérées par la macro de génération de code EASL Code Generation Macros .

Action

Nom de la collection	Description
Arguments	Les arguments de Action .
Sous-actions	Les sous-actions de l' Action .

Comportement

Nom de la collection	Description
Actions	Les actions du comportement.
Nœuds	Les nœuds du comportement.
Paramètres	Les paramètres du comportement.
Variables	Les variables du comportement.

Classificateur

Nom de la collection	Description
Toutes les machines d'état	Toutes Statemachines pour le classificateur.
Propriétés asynchrones	Les propriétés asynchrones du classificateur.
Déclencheurs asynchrones	Les déclencheurs asynchrones du Classificateur.
Comportements	Les comportements du classificateur.
Propriétés	Les propriétés du classificateur.
Propriétés chronométrées	Les propriétés chronométrées du classificateur.
Déclencheurs chronométrés	Les déclencheurs temporisés du Classificateur.

Déclencheurs	Tous déclencheurs du Classificateur.

Construction

Nom de la collection	Description
Tous les enfants	Les enfants du Construct.
Dépendances Client	Les dépendances du client sur Construct.
StéréoTypes	Les stéréotypes du Construct.
Dépendances du fournisseur	Les dépendances du fournisseur sur le Construct.

Nœud

Nom de la collection	Description
Bords entrants	Les bords entrants du nœud.
Bords sortants	Les bords sortants du nœud.
Sous-nœuds	Les sous-nœuds du Node.

State

Nom de la collection	Description
Faire des comportements	Les comportements de l' State .
Comportements d'entrée	Les comportements d'entrée de State .
Comportements de sortie	Les comportements de sortie de l' State .

Statemachine

Nom de la collection	Description

Tous les états finaux	Les States finaux de Statemachine .
Tous les États	Tous States au sein de la Statemachine , y compris ceux au sein States Submachine .
Transitions dérivées	Les transitions dérivées du Statemachine avec l'effet valide associé.
States	Les States au sein de la Statemachine .
Transitions	Les transitions au sein de la Statemachine .
Sommets	Les sommets de la Statemachine .

Transition

Nom de la collection	Description
Effets	Les effets de la Transition.
Gardes	Les gardes de la Transition.
Déclencheurs	Les déclencheurs de la Transition .

Déclencheur

Nom de la collection	Description
Transitions déclenchées	Les transitions déclenchées associées au Déclencheur .

Sommet

Nom de la collection	Description
Transitions sortantes dérivées	Transitions sortantes dérivées du sommet après avoir traversé les pseudo-nœuds.
Transitions entrantes	Les transitions entrantes du Vertex.
Transitions sortantes	Les transitions sortantes du Vertex.

Propriétés EASL

Cette rubrique répertorie les propriétés EASL pour chacun des objets EASL, telles que récupérées par la macro de génération de code EASL Code Generation Macros .

Action

Nom de la propriété	Description
Comportement	Le comportement associé à l' Action (Call Behaviour Action ou Call Operation Action).
Corps	Le corps de l' Action .
Contexte	Le contexte de Action .
Garde	La garde de Action .
EstFinal	Une vérification pour savoir si l'action est une Action finale.
EstGardé	Une vérification pour savoir si l'action est une Action gardée.
EstInitial	Une vérification pour savoir si l'action est une Action initiale.
Gentil	L' Action est du genre.
Suivant	L'action suivante de l' Action .
Nœud	Le nœud associé à Action dans le graphique.

Argument

Nom de la propriété	Description
Paramètre	L' ID du paramètre associé à l'argument.
Valeur	La valeur par défaut de l'argument.

Comportement

Nom de la propriété	Description
Action initiale	L'action initiale du comportement.

estReadOnly	Le isReadOnly du comportement.
estSingleExecution	L'isSingleExecution du comportement.
Gentil	Le genre de comportement.
Type de retour	Le type de retour du comportement.
Spécification	La spécification du comportement.

AppelÉvénement

Nom de la propriété	Description
Opération	Le fonctionnement du CallEvent.

ChangementÉvénement

Nom de la propriété	Description
ChangerExpression	L’expression de changement du ChangeEvent.

Classificateur

Nom de la propriété	Description
A des comportements	Une vérification pour savoir si le classificateur dispose de modèles comportementaux (activité et interaction).
Langue	Le langage du classificateur.
Statemachine	La Statemachine du classificateur.

Condition

Nom de la propriété	Description

Expression	L'expression de la Condition.
Inférieur	La valeur inférieure de la Condition.
Supérieur	La valeur supérieure de la Condition.

Construction

Nom de la propriété	Description
ObtenirTaggedValue	La Valeur Étiquetée de la Propriété.
EstStéréotypeAppliqué	Une vérification pour savoir si un stéréotype particulier est appliqué à la propriété.
Notes	Notes sur la propriété.
Type UML	Le type UML de la propriété.
Visibilité	La visibilité de la Propriété.

Bord

Nom de la propriété	Description
Depuis	L' ID du nœud à partir duquel le Edge est issu.
À	L' ID du nœud sur lequel le dispositif Edge est ciblé.

ObjetÉvénement

Nom de la propriété	Description
Type d'événement	Type d'événement de l' Object événement.

Exemple

Nom de la propriété	Description

Classificateur	Le classificateur de l'instance.
Valeur	La valeur de l'Instance.

Paramètre

Nom de la propriété	Description
Direction	La direction du paramètre.
Type	Le type du paramètre.
Valeur	La valeur du paramètre.

Primitif

Nom de la propriété	Description
Nom FQ	Le nom FQ du Primitif.
ID	L' ID du Primitif.
Nom	Le nom du Primitif.
ObjectType	Le type object de la primitive.
Parent	L'IDParent du Primitif.

PropriétéObjet

Nom de la propriété	Description
Taille liée	La taille liée du PropertyObject (s'il s'agit d'une collection).
ClassificateurStereoType	Le stéréotype du classificateur du PropertyObject.
EstAsynchProp	Une vérification pour savoir si le PropertyObject est une propriété asynchrone.
EstCollection	Une vérification pour savoir si le PropertyObject est une collection.
IsOrdered	Une vérification pour savoir si le PropertyObject est ordonné (s'il s'agit d'une collection).

EstTimedProp	Une vérification pour savoir si le PropertyObject est une propriété chronométrée.
Gentil	Le genre du PropertyObject.
Valeur inférieure	La valeur inférieure du PropertyObject (s'il s'agit d'une collection).
Type	Le type du PropertyObject.
Valeur supérieure	La valeur supérieure du PropertyObject (s'il s'agit d'une collection).
Valeur	La valeur du PropertyObject.

Événement de signal

Nom de la propriété	Description
Signal	Le signal du SignalEvent.

State

Nom de la propriété	Description
HasSubMachine	Une vérification pour savoir si l' State est un état sous-machine.
EstFinalState	Une vérification pour savoir si l' State est un État final.
Sous-machine	Obtenez l' ID de la sous-machine contenue par l' State (le cas échéant).

Statemachine

Nom de la propriété	Description
HasSubMachineState	Une vérification pour savoir si la Statemachine a un état Submachine.
Etat initial	L'état initial de la Statemachine .
État de la sous-machine	L' State de la sous-machine de Statemachine .

Événement de temps

Nom de la propriété	Description
Quand	La propriété « quand » de TimeEvent.

Transition

Nom de la propriété	Description
A un effet	Une vérification pour savoir si la transition a un effet valide.
IsDerived	Une vérification pour savoir si la transition est une transition dérivée.
EstTranscend	Une vérification pour savoir si la transition transcende d'une Statemachine (Submachine State) à une autre.
Est déclenché	Une vérification pour savoir si la transition est déclenchée.
Source	La source de la Transition.
Cible	La cible de la Transition.

Déclencheur

Nom de la propriété	Description
AsynchDestinationState	L'état de destination asynchrone du Déclencheur (s'il s'agit d'un déclencheur asynchrone).
Propriété dépendante	L' ID de la propriété associée au Déclencheur .
Événement	L'événement du Déclencheur .
Nom	Le nom du Déclencheur .
Type	Le type du Déclencheur .

Sommet

Nom de la propriété	Description

EstHistoire	Une vérification pour savoir si le sommet est un état historique.
EstPseudoState	Une vérification pour savoir si le sommet est un pseudo-état.
PseudoStateKind	Le type de pseudo-état du Vertex.

Appeler Gabarits depuis Gabarits

À l'aide d'appels de fonction avec des paramètres, vous pouvez appeler gabarits à partir d'autres gabarits , qu'il s'agisse gabarits standards ou gabarits définis par l'utilisateur créés dans votre projet. De plus, gabarits appelés peuvent renvoyer une valeur et peuvent être appelés de manière récursive.

Exemples

Une instruction d'appel renvoyant un paramètre à une variable :

$sSource = %StateEnumeratedName($Source)%

Une instruction d'appel à un gabarit qui a des paramètres :

%Tâche de règle($GUID, $index)%

Utilisation de l'instruction $parameter dans le gabarit appelé :

$GUID = $paramètre1

$index = $paramètre2

Gabarits support les appels récursifs, comme cet appel récursif sur le gabarit RuleTask :

$GUID = $paramètre1

$index = $paramètre2

% PI = " " %

$nul = "Initialiser la condition et object d'action"

$count = %BR_GET("RuletCount")%

% si $count == " " ou $count == $index %

%ComputeRègle($GUID)%

\n

% modèle de fin %

%Règle($index)%

\n

$index = %MATH_ADD($index, "1")%

%Tâche de règle($GUID, $index)%

L'éditeur de Code Gabarit dans MDG Development

Ces rubriques décrivent comment utiliser la fenêtre Code Gabarit Editor pour créer gabarits personnalisés :

- Create Custom Templates
- Customize Base Templates
- Add New Stereotyped Templates

L'éditeur Code Gabarit fournit les facilités de l' Éditeur de Code commun, y compris Intelli-sense pour les macros gabarit de génération de code. Pour plus d'informations sur Intelli-sense et Common Éditeur de Code , consultez la rubrique *Modification du code source* .

Créer Gabarits personnalisés

Enterprise Architect fournit une large gamme de gabarits qui définissent la manière dont les éléments de code sont générés. Si ceux-ci ne suffisent pas à vos besoins - par exemple, si vous souhaitez générer du code dans un langage qui n'est actuellement pas pris en charge par Enterprise Architect - vous pouvez créer de tout nouveaux gabarits personnalisés . Vous pouvez également ajouter des remplacements de stéréotypes à vos gabarits personnalisés ; par exemple, vous pouvez répertorier tous vos paramètres et leurs notes dans vos notes de méthode.

Accéder

Ruban	Développer > Code source > Options > Modifier le code Gabarits Conception > Paquetage > Transformation > Transformation Gabarits
Raccourcis Clavier	Ctrl+Shift+P (gabarits de génération de code) Ctrl+Alt+H (gabarits de transformation MDA)

Créez gabarits personnalisés à l'aide de l'éditeur Code Gabarits

Étape	Description
1	Dans le champ « Langue », cliquez sur la flèche déroulante et sélectionnez le langage de programmation approprié.
2	Cliquez sur le bouton Ajouter un nouveau Gabarit personnalisé. La dialogue « Créer un nouveau Gabarit personnalisé » s'affiche.
3	Dans le champ ' Gabarit Type ', cliquez sur la flèche déroulante et sélectionnez l' object modélisation approprié. L'option '<Aucun>' nécessite un traitement spécial ; il permet la définition d'une macro de fonction qui ne s'applique réellement à aucun des types, mais doit être appelée en tant que fonction pour définir les variables $parameter1, $parameter2 et ainsi de suite pour chaque valeur transmise.
4	Dans le champ « Nom Gabarit », saisissez un nom approprié. Cliquez sur le bouton OK .
5	Dans l'onglet 'Code Gabarits Editor', le nouveau gabarit est inclus dans la liste ' Gabarits ', avec la valeur 'Oui' dans le champ 'Modifié'. Le gabarit s'appelle < Type Gabarit >__< Nom Gabarit >. Note le double trait de soulignement entre le type gabarit et le nom gabarit .
6	Sélectionnez le gabarit dans la liste Gabarits et modifiez le contenu dans le champ Gabarit pour répondre à vos besoins.
7	Cliquez sur le bouton Enregistrer. Ceci stocke le nouveau gabarit , qui est maintenant disponible dans la liste des gabarits à utiliser. Vous

	pouvez également ajouter un remplacement de stéréotype au gabarit , si nécessaire.

Notes

- Pour un langage personnalisé, vous devez définir le gabarit du fichier afin qu'il puisse appeler les gabarits de la section d'importation, Namespace et de la classe, ainsi que tout autre gabarits que vous jugez applicable.

Personnaliser Gabarits de base

Enterprise Architect fournit une large gamme de gabarits qui définissent la manière dont les éléments de code sont générés. Si vous souhaitez modifier la façon dont un élément de code est généré, vous pouvez personnaliser les gabarits existants appropriés fournis par le système. Vos modifications peuvent concerner l'effet du gabarit lui-même ou ses appels à d'autres gabarits . Vous pouvez également ajouter des remplacements de stéréotypes à vos gabarits personnalisés ; par exemple, vous pouvez répertorier tous vos paramètres et leurs notes dans vos notes de méthode.

Lorsque vous personnalisez un gabarit (de base) fourni par le système, vous créez effectivement une copie du gabarit qui est utilisée de préférence à l'original. Toutes les modifications ultérieures concernent cette copie et le gabarit de base d'origine est masqué. Si vous supprimez ultérieurement la copie, elle ne peut plus remplacer l'original, qui est ensuite réutilisé.

Accéder

Ruban	Développer > Code source > Options > Modifier le code Gabarits
Raccourcis Clavier	Ctrl+Maj+P

Personnaliser un gabarit de base

Étape	Description
1	Sur l'Editeur de Code Gabarit , dans le champ 'Langue', cliquez sur la flèche déroulante et sélectionnez le langage de programmation pour lequel vous souhaitez personnaliser les gabarits de base.
2	Dans la liste Gabarits , cliquez sur le gabarit de base à modifier.
3	Mettre à jour le gabarit .
4	Cliquez sur le bouton Enregistrer pour enregistrer vos modifications.
5	Répétez les étapes 2 à 4 pour chacun des gabarits de base pertinents que vous souhaitez personnaliser.
6	Si vous préférez, ajoutez un ou plusieurs remplacements de stéréotypes à l'un des gabarits .

Ajouter de nouveaux Gabarits stéréotypés

Parfois, il est utile de définir un gabarit de génération de code spécifique à utiliser avec des éléments d'un stéréotype donné. Cela permet de générer un code différent pour les éléments, en fonction de leur stéréotype. Enterprise Architect fournit des gabarits par défaut, spécialisés pour les stéréotypes couramment utilisés dans les langues prises en charge. Par exemple, le gabarit « Operation Body » pour C# a été spécialisé pour le stéréotype de propriété, de sorte qu'il génère automatiquement ses méthodes constitutives « get » et « set ». Vous pouvez remplacer les gabarits stéréotypés par défaut comme décrit dans la rubrique *Remplacer Gabarits par défaut* . De plus, vous pouvez définir gabarits pour vos propres stéréotypes, comme décrit ici.

Accéder

Ruban	Développer > Code source > Options > Modifier le code Gabarits
Raccourcis Clavier	Ctrl+Maj+P

Ajouter un nouveau gabarit stéréotypé à l'aide de l'éditeur Code Gabarit

Étape	Description
1	Sélectionnez la langue appropriée dans la liste Langue.
2	Sélectionnez un des gabarits de base, dans la liste Gabarits .
3	Cliquez sur le bouton « Ajouter un nouveau remplacement stéréotypé ». La dialogue « Nouveau remplacement Gabarit » s'affiche.
4	Sélectionnez le stéréotype Fonctionnalité et/ou de Classe requis. Cliquez sur le bouton OK .
5	Le nouveau remplacement gabarit stéréotypé s'affiche dans la liste Remplacements de stéréotype, marqué comme modifié.
6	Apportez les modifications requises dans l'éditeur de Code Gabarits .
7	Cliquez sur le bouton Enregistrer pour stocker le nouveau gabarit stéréotypé dans le fichier du projet. Enterprise Architect peut maintenant utiliser le gabarit stéréotypé lors de la génération de code pour les éléments de ce stéréotype.

Notes

- Les stéréotypes de classe et fonctionnalité peuvent être combinés pour fournir un niveau supplémentaire de spécialisation pour fonctionnalités ; par exemple, si les propriétés doivent être générées différemment lorsque la

classe a un stéréotype MyStereotype, alors la propriété et MyStereotype doivent être spécifiés dans la dialogue New Gabarit Override

Remplacer Gabarits par défaut

Enterprise Architect dispose d'un ensemble de gabarits de génération de code intégrés ou par défaut. L'éditeur Code Gabarits vous permet de modifier ces gabarits par défaut, personnalisant ainsi la manière dont Enterprise Architect génère le code. Vous pouvez choisir de modifier tout ou partie des gabarits de base pour obtenir le style de codage souhaité.

Tous gabarits que vous avez remplacés sont stockés dans le fichier de projet. Lors de la génération du code, Enterprise Architect vérifie d'abord si un gabarit a été modifié et si c'est le cas, utilise ce gabarit . Sinon, le gabarit par défaut approprié est utilisé.

Accéder

Ruban	Développer > Code source > Options > Modifier le code Gabarits
Raccourcis Clavier	Ctrl+Maj+P

Référence

Remplacez un gabarit de génération de code par défaut à l'aide de l'éditeur Code Gabarits .

Lors de la génération du code, Enterprise Architect utilise maintenant le gabarit de remplacement au lieu du gabarit par défaut.

Champ/Bouton	Description
Langue	Sélectionnez la langue appropriée dans la liste.
Gabarits	Sélectionnez l'un des gabarits de base dans la liste.
Remplacement des stéréotypes	Si le gabarit de base comporte des remplacements stéréotypés, vous pouvez en sélectionner un dans la liste.
<Autres champs>	Apportez toute autre modification requise.
Sauvegarder	Cliquez sur ce bouton pour stocker la version modifiée du gabarit dans le fichier projet. Le gabarit est marqué comme modifié.

Cadre de grammaire

Enterprise Architect fournit support de l'ingénierie inverse pour un certain nombre de langages de programmation populaires. Cependant, si la langue que vous utilisez n'est pas prise en charge, vous pouvez écrire votre propre grammaire à l'aide de l'éditeur de grammaire intégré. Vous pouvez ensuite incorporer la grammaire dans une MDG Technologie pour fournir à la fois support de l'ingénierie inverse et de la synchronisation du code pour votre langue cible.

Le framework permettant d'écrire une grammaire et de l'importer dans Enterprise Architect est le complément direct du Framework Code Gabarit . Alors que gabarits de code servent à convertir un modèle en forme textuelle, des grammaires sont nécessaires pour convertir du texte en modèle. Les deux sont nécessaires pour synchroniser les modifications dans vos fichiers source.

Un exemple de fichier source de langue et un exemple de grammaire pour cette langue sont fournis dans le répertoire Code Samples, auquel vous pouvez accéder à partir de votre répertoire d'installation (l'emplacement par défaut est C:\Program Files\ Sparx Systems \EA). Deux autres fichiers de grammaire sont également fournis, illustrant des aspects spécifiques du développement de grammaires.

Composants

Composant	Description
Syntaxe de la grammaire	Les grammaires définissent la manière dont un texte doit être divisé en structure, ce qui est nécessaire lorsque vous convertissez du code en représentation UML . Au niveau le plus simple, une grammaire est constituée d'instructions permettant de diviser une entrée pour former une structure. Enterprise Architect utilise une variante du formulaire Backus-Naur (nBNF) pour inclure des instructions de traitement, dont l'exécution renvoie des informations structurées à partir des résultats analysés sous la forme d'un arbre de syntaxe abstraite (AST), qui est utilisé pour générer une représentation UML .
Éditeur de grammaire	L'éditeur de grammaire est un éditeur intégré que vous pouvez utiliser pour ouvrir, modifier, valider et enregistrer des fichiers de grammaire.
Débogage de la grammaire	Vous pouvez déboguer les fichiers de grammaire que vous créez en utilisant deux facilités : • Le Parser , qui génère l'AST pour la Grammaire • Le Profiler, qui analyse également la grammaire et génère l'AST mais qui expose le chemin de profilage pour montrer exactement ce qui s'est passé à chaque étape du processus

Syntaxe de la grammaire

Les grammaires définissent la manière dont un texte doit être divisé en structure, ce qui est exactement ce qui est nécessaire lorsque vous convertissez du code en représentation UML . Au niveau le plus simple, une grammaire n'est que des instructions permettant de diviser une entrée pour former une structure. Enterprise Architect utilise une variante de la forme Backus-Naur (BNF) pour exprimer une grammaire de manière à lui permettre de convertir le texte en représentation UML . Ce que la grammaire d' Enterprise Architect offre par rapport à un BNF pur, c'est l'ajout d'instructions de traitement, qui permettent de renvoyer des informations structurées à partir des résultats analysés sous la forme d'un arbre syntaxique abstrait (AST). À la fin de l'AST, Enterprise Architect le traitera pour produire un modèle UML .

Syntaxe

Syntaxe	Détail
commentaires	Les commentaires ont la même forme que dans de nombreux langages de programmation. // Vous pouvez commenter la fin d'une ligne en ajoutant deux /s. /* Vous pouvez commenter plusieurs lignes en ajoutant un / suivi d'un *. Le commentaire se termine lorsque vous ajoutez un * suivi d'un /. */
Instructions	Les instructions précisent les détails clés du fonctionnement de la grammaire. Ils sont généralement inclus en haut de la grammaire et ressemblent aux appels de fonction dans la plupart des langages de programmation.
Règles	Les règles constituent le corps d'une grammaire. Une règle peut avoir une ou plusieurs définitions séparées par des délimiteurs de barres verticales (|). Pour qu'une règle soit acceptée, n'importe quelle définition complète doit être acceptée. Les règles se terminent par le caractère point-virgule (;).
Définitions	Une définition est l'un des chemins qu'une règle peut emprunter. Chaque définition est composée d'un ou plusieurs termes.
Listes de définitions	Une liste de définitions correspond à un ou plusieurs ensembles de termes. Ceux-ci seront évalués dans l'ordre jusqu'à ce que l'on réussisse. Si aucune ne réussit, la règle contenant échoue. Chaque paire de définitions est séparée par un | personnage. Il s'agit d'une règle simple avec trois définitions : <salutation> ::= "bonjour" | "salut" | ["bonjour";
Termes	Un terme peut être une référence à une règle, une valeur spécifique, une plage de valeurs, une sous-règle ou une commande.
Commandes	Comme les instructions, les commandes ressemblent à des appels de fonction. Ils répondent à deux objectifs principaux : • Pour traiter les jetons d'une manière spécifique ou • Pour fournir un résultat à l'appelant

Instructions de grammaire

Les instructions précisent les détails clés du fonctionnement de la grammaire. Ils sont généralement inclus en haut de la grammaire et ressemblent aux appels de fonction dans la plupart des langages de programmation.

Instructions

Instruction	Description
sensible aux majuscules et minuscules()	L'une de ces deux instructions est censée spécifier si la correspondance des jetons doit être sensible à la casse ou non. Par exemple, les langues de la famille BASIC ne sont pas sensibles à la casse, tandis que les langues de la famille C sont sensibles à la casse.
caseInsensible()	
délimiteurs (DelimiterRule : Expression)	L'instruction délimiteurs indique à l'analyseur lexical quelle règle utiliser pour la découverte des délimiteurs. Les délimiteurs sont utilisés lors de l'analyse des mots-clés et peuvent être définis comme les caractères pouvant être utilisés immédiatement avant ou après les mots-clés de la langue.
lex (TokenRule : Expression)	L'instruction lex indique à l'analyseur lexical le nom de la règle racine à utiliser pour son analyse.
analyser (RootRule : Expression) analyser (RootRule : Expression, SkipRule : Expression)	L'instruction parse indique à l'analyseur le nom de la règle racine à utiliser pour son traitement. Le deuxième argument facultatif spécifie une règle de saut (ou d'échappement), qui est généralement utilisée pour gérer les commentaires.

Règles de grammaire

Les règles sont exécuter pour diviser le texte en structure. Une règle est composée d'une ou plusieurs définitions, chacune étant composée d'un ou plusieurs termes.

Types de règles

Règle	Description
Règles nommées	Un nom, suivi d'une liste de définitions. Par exemple: <règle> ::= <term1> <term2> | "-" <terme1> ;
Règles en ligne	À l'intérieur d'une définition, une règle définie entre parenthèses. Celles-ci agissent exactement de la même manière que s'il s'agissait d'une règle nommée appelée par un terme. Par exemple: <rule> ::= (<en ligne>);
Règles facultatives	À l'intérieur d'une définition, une règle définie entre crochets. Cette règle réussit même si le contenu échoue. Par exemple: <rule> ::= [<en ligne>];
Règles répétitives	Dans une définition, un terme suivi d'un signe plus. Cette règle correspond à la règle interne une ou plusieurs fois. Par exemple: <rule> ::= <en ligne>+; règle ::= (<term1> <term2>)+;
Règles de répétition facultatives	A l'intérieur d'une définition, une règle suivie d'une étoile. Cette règle correspond à la règle interne zéro ou plusieurs fois, ce qui signifie qu'elle réussit même si la règle interne ne réussit jamais. Par exemple: <rule> ::= <en ligne>*; règle ::= (<term1> <term2>)*;

Termes de grammaire

Les termes identifient où les jetons sont consommés.

Types de terme

Type	Description
Termes concrets	Chaînes entre guillemets. Par exemple, « classe »
Caractères Unicode	Un terme uniquement lexer, ayant le préfixe U+0x suivi d'un nombre hexadécimal. Par exemple : U+0x1234
Gammes	Un terme uniquement lexer, correspondant à n'importe quel caractère entre les deux caractères spécifiés. Par exemple, "a".."z" ou U+0x1234..U+2345
Les références	Le nom d'une autre règle, entre crochets. Le jeton correspondra si cette règle réussit. Par exemple, <une autreRègle>
Commandes	Un appel à une commande spécifique.

Commandes de grammaire

Les commandes, comme les instructions, ressemblent à des appels de fonction. Ils répondent à deux objectifs principaux :

- Pour traiter les jetons d'une manière spécifique ou
- Pour fournir un résultat à l'appelant

Commandes

Commande	Description
attribut (Nom : String , Valeur : Expression)	Crée un attribut sur le nœud AST actuel. L'attribut sera créé avec le nom spécifié dans la source grammaticale et recevra la valeur de tous les jetons consommés dans le cadre de l'exécution de l'expression de valeur. Cette commande produit les attributs de nœud AST sur lesquels Enterprise Architect opère dans l'ingénierie de code.
attributEx(Nom : String) attributEx (Nom : String , Valeur : String)	Crée un attribut sur le nœud AST actuel sans consommer de jetons. L'attribut sera créé avec le même nom que celui spécifié dans la source grammaticale, et avec soit une valeur vide, soit la valeur spécifiée par l'argument facultatif Value. Cette commande produit les attributs de nœud AST sur lesquels Enterprise Architect opère dans l'ingénierie de code.
nœud (Nom : String , Cible : Expression)	Crée un nœud AST sous le nœud AST actuel (les nœuds sur lesquels Enterprise Architect opère dans l'ingénierie de code). Le nœud sera créé avec le nom spécifié dans la source grammaticale.
jeton (cible : expression)	Crée un jeton lors de l'analyse lexicale pour le traitement lors de l'analyse. La valeur du jeton sera la valeur de tous les caractères consommés suite à l'exécution de l'expression cible.
mots clés()	Correspond à n'importe quelle string littérale utilisée comme terme de grammaire ; autrement dit, si vous saisissez une string explicite que vous recherchez, elle devient un mot clé.
sauter (Cible : Expression) sauter (Cible : Expression, Échap : Expression)	Consomme les données d'entrée (caractères lors du lexing et jetons lors de l'analyse) jusqu'à ce que l'expression « cible » corresponde. L'expression facultative « Escape » peut être utilisée pour gérer des instances telles que des guillemets échappés dans des chaînes.
skipBalanced (Origine : Expression, Cible : Expression) skipBalanced (Origine : Expression, Cible : Expression, Évasion : Expression)	Consomme les données d'entrée (caractères ou jetons) jusqu'à ce que l'expression « cible » corresponde et que le niveau d'imbrication atteigne zéro. Si l'expression « Origine » correspond au cours de ce processus, le niveau d'imbrication est augmenté. Si l'expression « Cible » correspond, le niveau d'imbrication est diminué. Lorsque le niveau d'imbrication atteint zéro, la commande se termine avec succès. Une expression facultative « Escape » peut être fournie.
sauterEOF()	Consomme toutes les données restantes (caractères ou jetons) jusqu'à la fin du fichier.

échouer()	Fait échouer l'analyseur avec la règle actuelle, y compris les définitions restantes.
avertissement()	Insère un avertissement dans l'AST résultant.
except(Cible : Expression, Exception : Expression)	Consomme les données d'entrée qui correspondent à l'expression Target, mais échoue sur les données qui correspondent à l'expression Exception. Cela fonctionne de manière quelque peu similaire, mais exactement à l'opposé, de la commande skip.
préProcess (Cible : Expression)	Évalue une expression et utilise ces données prétraitées dans plusieurs définitions. Ceci est particulièrement utile dans l'analyse d'expressions, où la même expression de gauche sera évaluée par rapport à un certain nombre d'opérateurs. Cette commande réduit le travail que l'analyseur doit effectuer pour que cela se produise.

Nœuds AST

En définissant une grammaire, vous utiliserez des nœuds AST et des attributs de nœud AST qui peuvent être reconnus en ingénierie de code dans Enterprise Architect , dans les résultats AST renvoyés par les commandes d'attribut, d'attributEx et de nœud. Les nœuds et attributs sont identifiés dans ces tableaux . Tous les autres seront ignorés dans l'ingénierie du code.

Nœud FICHIER

Le nœud FILE représente un fichier. Il n'est mappé à rien, mais contient toutes les informations requises.

Multiplicité / Nœuds	Description
0..* / FORFAIT	Voir *Nœud PACKAGE* .
0..* / CLASSE	Voir *Noeud CLASS* .
0..* / IMPORTATION	Le nœud pour représenter l'espace de noms/ Paquetage importé ou équivalent. L'attribut 'NAME' du nœud sera le nom de l'espace de noms/ Paquetage importé ou équivalent.
0..* / COMMENTAIRE	Les étiquettes de champ faisant partie d'une règle de saut seront au niveau racine ; le générateur de code recherche des commentaires de ce type par position par rapport au nœud.
0..1 / INSERT_POSITION	Cela donne la position où les nouvelles classes, Paquetages et implémentations de méthodes peuvent être insérés dans le fichier. S'il n'est pas trouvé, le générateur de code insérera automatiquement de nouveaux éléments immédiatement après que le dernier ait été trouvé dans le code.

Nœud FORFAIT

Le nœud PACKAGE correspond à un espace de noms ou équivalent dans le fichier. Lors de l'importation avec « paquetage par espace de noms », Enterprise Architect créera un Paquetage directement sous l'importation pour cela et y placera toutes les classes. Lorsqu'il n'importe pas d'espaces de noms, Enterprise Architect recherchera les classes sous ce point, mais il ne fera rien avec ce nœud.

De plus, si vous générez avec les espaces de noms activés (voir les rubriques d'aide *des options de code* pour les langages génériques), une classe générée ne correspondra pas à une classe dans le code à moins qu'elle ne se trouve sous la même structure Paquetage .

Contenu dans les nœuds : FILE

Multiplicité / Nœuds	Description
1/ NOM	Voir *Nœud NOM* .
0..* / CLASSE	Voir *Noeud CLASS* .
0..* / FORFAIT	Le nœud Paquetage enfant.
0..1 /	Donne la position où s'ouvre le corps du Paquetage . Cela peut également être

OUVERT_POSITION	utilisé comme position d'insertion.
0..1 / INSERT_POSITION	Donne la position où les nouvelles Classes et Paquetages peuvent être insérés dans le fichier. S'il n'est pas trouvé, le générateur de code insérera automatiquement de nouveaux éléments immédiatement après que le dernier ait été trouvé dans le code.
0..1 / SUPPRESSION	Empêche l'indentation lors de l'insertion dans ce Paquetage .

Nœud CLASSE/INTERFACE

Le nœud CLASS (ou INTERFACE) est le plus important dans la génération de code. Il est introduit sous forme d'objets de classe (ou d'interface).

Voir *Classe DECLARATION* et *Classe BODY* .

Contenu dans les nœuds : FILE, PACKAGE, classe BODY

Déclaration CLASSE

Contenu dans les nœuds : CLASSE/INTERFACE

Multiplicité / Nœuds	Description
1/ NOM	Voir *Nœud NOM* .
0..* / PARENT	Voir *Nœud PARENT* .
0..* / ÉTIQUETTE	Voir *Nœud TAG* .
0..1 / DESCRIPTION	Voir *DESCRIPTION Nœud* .
1/ NOM	Le nom de la classe. S'il existe un NOM de nœud, cela écrasera cet attribut.
0..1 / PORTÉE	La portée UML de la classe - Publique, Privée, Protégée ou Paquetage .
0..1 / RÉSUMÉ	S'il est présent, indique qu'il s'agit d'une classe abstraite.
0..1 / VERSION	La version de la classe.
0..1 / STÉRÉOTYPE	Le stéréotype Enterprise Architect devrait attribuer à la classe. Cela ne prend pas support plusieurs stéréotypes.
0..1 / ÎLEAF	S'il est présent, indique qu'il s'agit d'une classe feuille/finale/scellée dont aucune sous-classe ne peut hériter.
0..1 / MULTIPLICITÉ	S'il est présent, représente la multiplicité de la classe.
0..1 / LANGUE	En règle générale, vous n'avez pas besoin de définir cela.
0..1 / REMARQUE	Généralement non utilisé car il est abordé dans les commentaires au-dessus de la

	classe.
0..1 / ALIAS	S'il est présent, représente l'alias de tout identifiant, tel qu'un Namespace , une classe ou une variable.
0..* / MACRO	Ajoute une Valeur Étiquetée numérotée qu'Enterprise Enterprise Architect peut utiliser pour effectuer round entre les macros.

Nœud CORPS de classe

Contenu dans les nœuds : CLASSE/INTERFACE

Multiplicité / Nœuds	Description
0..* / MÉTHODE	Voir *Nœud MÉTHODE* .
0..* / ATTRIBUT	Voir *Nœud ATTRIBUT* .
0..* / CHAMP	Voir *Nœud FIELD* .
0..* / CLASSE	Voir *Noeud CLASS* .
0..* / PORTÉE	Voir *Nœud SCOPE* .
0..* / PROPRIÉTÉ	Ce nœud représente la définition de propriété dans le corps de classe.
0..* / ÉTIQUETTE	Voir *Nœud TAG* .
0..* / PARENT	Voir *Nœud PARENT* .
0..1 / OUVERT_POSITION	Donne la position où le corps de la classe s'ouvre. Cela peut également être utilisé comme position d'insertion.
0..1 / INSERT_POSITION	Donne la position où les nouveaux membres de la classe peuvent être insérés dans le fichier. S'il n'est pas trouvé, le générateur de code insérera automatiquement de nouveaux éléments immédiatement après que le dernier ait été trouvé dans le code.

Nœud PORTÉE

Il s'agit d'une fonctionnalité facultative pour les langages ressemblant au C++ qui ont des blocs qui spécifient la portée des éléments. La langue doit avoir un nom spécifié qui est utilisé pour la portée de tous les éléments du Bloc . À tous autres égards, il se comporte de manière identique au nœud Class BODY.

Contenu dans les nœuds : classe BODY

Multiplicité / Nœuds	Description
1/ NOM	Utilisé comme portée pour toutes les méthodes et attributs contenus dans la portée.

Nœud MÉTHODE

Contenu dans les nœuds : classe BODY, SCOPE

Multiplicité / Nœuds	Description
1/ Méthode DÉCLARATION	Voir *Méthode DECLARATION Noeud* .

Méthode DECLARATION Noeud

Contenu dans les nœuds : MÉTHODE

Multiplicité / Nœuds	Description
0..1 / TYPE	Voir *Nœud TYPE* .
0..* / PARAMÈTRE	Voir *Nœud PARAMETRE* .
0..* / ÉTIQUETTE	Voir *NŒUD D'ÉTIQUETTE* .
0..1 / DESCRIPTION	Voir *DESCRIPTION Nœud* .
0..1 / MULTI PARAMÈTRE	Prend en charge le style de déclaration de liste de paramètres de Delphi. C'est l'équivalent de FIELD.
1/ NOM	Le nom de la méthode.
0..1 / TYPE	Le type de retour de la méthode.
0..1 / PORTÉE	La portée UML de la méthode - Public, Private, Protected ou Paquetage .
0..1 / RÉSUMÉ	S'il est présent, indique que la méthode est abstraite.
0..1 / STÉRÉOTYPE	Le stéréotype Enterprise Architect doit attribuer à la méthode. Cela ne prend pas support plusieurs stéréotypes.
0..1 / STATIQUE	S'il est présent, indique que la méthode est statique.
0..1 / CONST ou CONSTANTE	S'il est présent, indique que la méthode est constante.
0..1 / PUR	S'il est présent, indique que la méthode est une méthode pure.
0..1 / ISQUERIE	S'il est présent, indique que la méthode est en requête/lecture seule.
0..1 / TABLEAU	S'il est présent, indique que le type de méthode (type de retour) est un tableau.

0..1 / SYNCHRONISÉ	S'il est présent, indique que la méthode est une méthode synchronisée.
0..* / MACRO	La macro spécifiée dans la déclaration de méthode.
0..1 / CSHARPIMPLEMENTS	Spécifie un comportement spécial pour C# .
0..1 / COMPORTEMENT	Fournit support en charge de l'aspect J, en utilisant le comportement.
0..1 / MONTRER LE COMPORTEMENT	Fournit support de l'aspect J, en utilisant le comportement, et montre le comportement de rétro-ingénierie sur le diagramme .

Nœud ATTRIBUT

Contenu dans les nœuds : classe BODY, SCOPE

Multiplicité / Nœuds	Description
1 / TYPE	Voir *Nœud TYPE* .
0..* / ÉTIQUETTE	Voir *Nœud TAG* .
0..1 / DESCRIPTION	Voir *DESCRIPTION Nœud* .
1/ NOM	Le nom de l'attribut.
0..1 / TYPE	Le type de l'attribut.
0..1 / PORTÉE	La portée UML de l'attribut - Public, Private, Protected ou Paquetage .
0..1 / PAR DÉFAUT	La valeur par défaut de l'attribut.
0..1 / CONTENEUR ou ARRAY	S'il est présent, indique le conteneur de l'attribut.
0..1 / CONFINEMENT	Référence ou valeur .
0..1 / STÉRÉOTYPE	Stéréotype qu'Enterprise Enterprise Architect doit attribuer à l'attribut. Cela ne prend pas support plusieurs stéréotypes.
0..1 / STATIQUE	S'il est présent, indique qu'il s'agit d'un attribut statique.
0..1 / CONST ou CONSTANTE	S'il est présent, indique qu'il s'agit d'un attribut constant.
0..1 / COMMANDÉ	S'il est présent, indique que l'attribut (valeur) est ordonné.
0..1 / LIMITE FAIBLE	Si présent, représente la bordure inférieure de l'attribut valeur .
0..1 / LIMITE HAUTE	Si présent, représente la bordure supérieure de la valeur de l'attribut.

0..1 / TRANSITOIRE ou VOLATILE	S'il est présent, indique que l'attribut est transitoire ou volatile.

Nœud CHAMP

Un champ correspond à plusieurs déclarations d'attributs en une seule. Tout ce qui n'est pas défini dans les déclarateurs mais défini dans le champ lui-même sera défini pour chaque déclarateur. Tout ce qui est pris en charge dans un attribut est pris en charge dans le champ. Si aucun déclarateur n'est trouvé, cela fonctionne de la même manière qu'un attribut.

Contenu dans les nœuds : classe BODY, SCOPE

Multiplicité / Nœuds	Description
0..* / DÉCLARATEUR	Voir *Nœud ATTRIBUT* .

Nœud PARAMETRE

Contenu dans les nœuds : méthode DECLARATION, TEMPLATE

Multiplicité / Nœuds	Description
1 / TYPE	Voir *Nœud TYPE* .
0..* / ÉTIQUETTE	Voir *Nœud TAG* .
0..1 / DESCRIPTION	Voir *DESCRIPTION Nœud* .
0..1 / NOM	Le nom du paramètre.
0..1 / TYPE	Le type du paramètre.
0..1 / GENRE	On s'attend à ce qu'il soit entrant, sortant, sortant ou revenant.
0..1 / PAR DÉFAUT	La valeur par défaut du paramètre.
0..1 / FIXE	S'il est présent, indique que le paramètre est fixe/constant.
0..1 / TABLEAU	S'il est présent, indique que le type de paramètre est un tableau.

Nœud NOM

Contenu dans les nœuds : PACKAGE, DÉCLARATION de classe

Multiplicité / Nœuds	Description

1/ NOM	La partie nom.
0..* / QUALIFICATEUR	La partie qualificative.
0..* / PARTIENOM	Une alternative à l'utilisation de NAME et QUALIFIER. Une string de valeurs, toutes sauf la dernière prises comme qualificatifs. Le dernier est considéré comme le Nom.

TYPE Nœud

Contenu dans les nœuds : méthode DÉCLARATION, ATTRIBUT, PARAMÈTRE

Multiplicité / Nœuds	Description
0..1 / MODÈLE	Le texte entier du gabarit est le nom du type. Utilisé uniquement si NAME n'est pas défini. Voir *Nœud MODÈLE* .
1/ NOM	La partie nom.
0..* / QUALIFICATEUR	La partie qualificative.
0..* / PARTIENOM	Une alternative à l'utilisation de NAME et QUALIFIER. Une string de valeurs, toutes sauf la dernière prises comme qualificatifs. Le dernier est considéré comme le Nom.

Nœud MODÈLE

Contenu dans les nœuds : TYPE

Multiplicité / Nœuds	Description
0..* / PARAMÈTRE	Voir *Nœud PARAMETRE* .
1/ NOM	

Nœud PARENT

Contenu dans les nœuds : classe DÉCLARATION

Multiplicité / Nœuds	Description
0..1 / TYPE	A la valeur Parent, Implements ou VirtualP.
1/ NOM	La partie nom du parent.

0..* / QUALIFICATEUR	La partie qualificative du Parent.
0..* / PARTIENOM	Une alternative à l'utilisation de NAME et QUALIFIER. Une string de valeurs, toutes sauf la dernière prises comme qualificatifs. Le dernier est considéré comme le Nom.
0..1 / INSTANTIATION	S'il est présent, indique l'instanciation d'un paramètre gabarit .

Nœud ÉTIQUETTE

Contenu dans les nœuds : Classe DECLARATION, Méthode DECLARATION, ATTRIBUTE, PARAMETER

Multiplicité / Nœuds	Description
1/ NOM	Le nom de la Valeur Étiquetée (le Tag).
0..* / VALEUR	La valeur de la Valeur Étiquetée .
0..1 / MÉMO	S'il est présent, indique que le type de la Valeur Étiquetée est <memo>.
0..1 / NOMÉMO	S'il est présent, indique que le type de la Valeur Étiquetée n'est pas <mémo>.
0..1 / GROUPE	S'il est présent, indique que la valeur est un groupe Valeur Étiquetée .

DESCRIPTION Nœud

Contenu dans les nœuds : Classe DECLARATION, Méthode DECLARATION, ATTRIBUTE, PARAMETER

Multiplicité / Nœuds	Description
0..* / VALEUR	Le texte Enterprise Architect doit attribuer à la Note .

Modification des grammaires

Si vous devez écrire et modifier une grammaire pour du code importé dans un nouveau langage de programmation, vous pouvez le faire à l'aide de l'éditeur de grammaire intégré.

Accéder

Ruban	Développer > Code source > Éditeur de grammaire

Créer et modifier la grammaire

Champ/Bouton	Action
Grammaire ouverte	Affichez un navigateur à travers lequel vous pouvez localiser et ouvrir le fichier contenant la grammaire que vous souhaitez modifier.
Récent	Les grammaires récemment utilisées sont rapidement accessibles à l'aide de cette zone de liste déroulante.
Sauvegarder	Enregistrez le fichier actuel.
Enregistrer sous	Enregistre une copie du fichier actuel
Valider la grammaire	La validation grammaticale exécuter une série de tests sur la grammaire actuelle pour garantir sa validité. Des erreurs et des avertissements seront affichés pour vous informer à la fois des erreurs qui rendront la grammaire inutilisable et des conditions dans lesquelles vous pourriez obtenir des résultats inattendus.
Aide	Affichez cette rubrique d'aide.

Options Menu Contexte

Champ/Bouton	Action
Fichier ouvert	Affichez un navigateur à travers lequel vous pouvez localiser et ouvrir le fichier contenant la grammaire que vous souhaitez modifier.
Valider	La validation grammaticale exécuter une série de tests sur la grammaire actuelle pour garantir sa validité. Des erreurs et des avertissements seront affichés pour vous informer à la fois des erreurs qui rendront la grammaire inutilisable et des conditions dans lesquelles vous pourriez obtenir des résultats inattendus.
Langue	L'éditeur de grammaire utilise par défaut la forme Backus-Naur normale (nBNF).

	L'option mBNF est également disponible.
Numéros de ligne	Activez ou désactivez les numéros de ligne dans l'éditeur de grammaire.

Analyse des résultats AST

L'arbre de syntaxe abstraite (AST) est le code qu'Enterprise Enterprise Architect voit lorsqu'il traite une grammaire.

Vous analysez le texte dans la moitié inférieure de la fenêtre de l'éditeur de grammaire et révision ce qui est affiché en conséquence. Vous pouvez soit ouvrir un fichier, soit y coller du texte. Si vous avez collé du texte qui correspond à quelque chose qui ne peut pas apparaître au niveau du fichier (comme les paramètres d'opération), vous pouvez sélectionner une règle alternative à utiliser comme point de départ. L'analyse commencera alors à partir de cette règle.

Accéder

Ruban	Développer > Code source > Éditeur de grammaire > Débogueur de grammaire > Résultats AST

Options barre d'outils

Option	Action
Fichier ouvert	Ouvrez un exemple de fichier d'entrée à tester.
Récent	Les fichiers source récemment ouverts peuvent être sélectionnés dans cette liste déroulante.
Analyser	Effectuez l'opération d'analyse. Si l'analyse réussit, l'onglet « Résultats AST » contiendra l'AST résultant.
Sélectionnez une règle	Cette liste déroulante vous permet de sélectionner une règle racine alternative pour traiter votre exemple de source.
Aide	Affichez cette rubrique d'aide.

Profilage de l'analyse grammaticale

Lorsque vous analysez une grammaire que vous avez créée, des erreurs que vous ne pouvez pas diagnostiquer immédiatement peuvent apparaître. Pour vous aider à résoudre de telles erreurs, vous pouvez révision le processus suivi par l'analyseur pour générer l'AST que vous pouvez voir, à l'aide du Grammar Profiler.

Vous analysez à nouveau le texte dans la moitié inférieure de la fenêtre de l'éditeur de grammaire, mais cette fois, l'arborescence montre chaque règle tentée par l'analyseur, où elle est arrivée et si elle a réussi ou non. Les règles d'ouverture d'un fichier, de collage d'un fichier et de définition de la règle de démarrage restent les mêmes.

Accéder

Ruban	Développer > Code source > Éditeur de grammaire > Débogueur de grammaire > Résultats du profileur

Options barre d'outils

Option	Action
Fichier ouvert	Affichez un navigateur à travers lequel vous pouvez localiser et ouvrir le fichier contenant la grammaire que vous souhaitez modifier.
Analyser	Effectuez l'opération d'analyse. Si l'analyse réussit, l'onglet « Résultats AST » contiendra l'AST résultant et l'onglet « Résultats du profil » contiendra des informations de débogage concernant le chemin emprunté par l'analyseur dans votre grammaire. Les données de profil sont extrêmement utiles lors du débogage d'une nouvelle grammaire.
Sélectionnez une règle	Si vous souhaitez utiliser une règle racine différente pour traiter votre source d'échantillon, cliquez sur la flèche déroulante et sélectionnez la règle alternative.
Aide	Affichez cette rubrique d'aide.

Notes

- Étant donné que le profilage peut prendre très longtemps pour les fichiers volumineux, l'onglet « Résultats du profil » n'est pas rempli si vous n'affichez pas cet onglet lorsque vous commencez l'analyse.

Éditeur de macros

L'éditeur de macros permet à un utilisateur de compléter la grammaire avec une liste de mots-clés et de règles pour exclure les macros lors des opérations d'analyse grammaticale. La liste de définitions de macros est particulièrement utile lors du développement de grammaires pour des langages support les macros tels que C++. Il évite la nécessité de décrire ces règles dans la grammaire elle-même et peut être utilisé avec plusieurs grammaires.

Cette fonctionnalité est disponible à partir d' Enterprise Architect version 14.1.

Accéder

Ruban	Développer > Code source > Éditeur de grammaire > Éditeur de macros

Modification des macros

Fichier ouvert	Ouvrir une liste de définitions de macro existante
Récent	Les listes de définitions de macro récemment ouvertes peuvent être sélectionnées dans cette liste déroulante
Sauvegarder	Enregistre les modifications apportées à la liste de définitions de macro ouverte
Enregistrer sous	Enregistre une copie de la liste de définitions de macro existante
Valider	Valide la grammaire de la liste de définitions de macro

Exemples de grammaires

Le répertoire Code Samples configuré par le programme d'installation Enterprise Architect contient un exemple de grammaire que vous pouvez charger dans l'éditeur de grammaire pour révision et dans le Débogueur de grammaire pour analyser et profiler.

L'exemple de grammaire se compose de deux fichiers :

- test.ssl - un exemple de fichier source de langage simple, dans le style C, et
- ssl.nbnf - une grammaire pour un exemple de langage simple

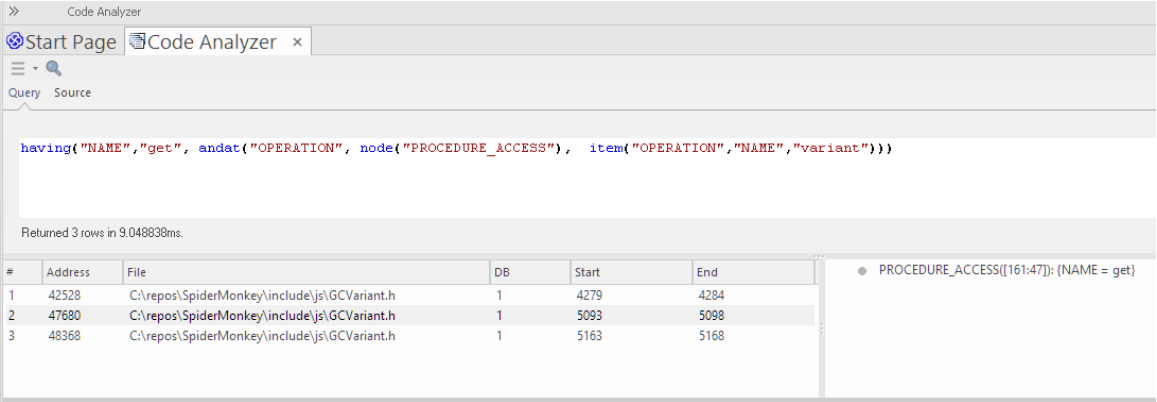
L'exemple illustre :

- Tokenisation (à l'aide du Lexer)
- Création d'un Paquetage
- Création d'une classe ou d'une interface
- Création d'un attribut
- Création d'une opération (avec paramètres)
- Importer des commentaires

Le répertoire Code Samples contient également deux autres fichiers Grammar que vous pouvez examiner :

- Expressions Sample.nBNF - ceci illustre comment l'analyse des expressions est configurée et traitée, avec un texte de commentaire détaillé fournissant des explications
- CSV Sample.nBNF - un exemple de grammaire pour le traitement des fichiers CSV

Analyseur de code



L'analyseur de code est un outil essentiel pour quiconque traite quotidiennement du code source.

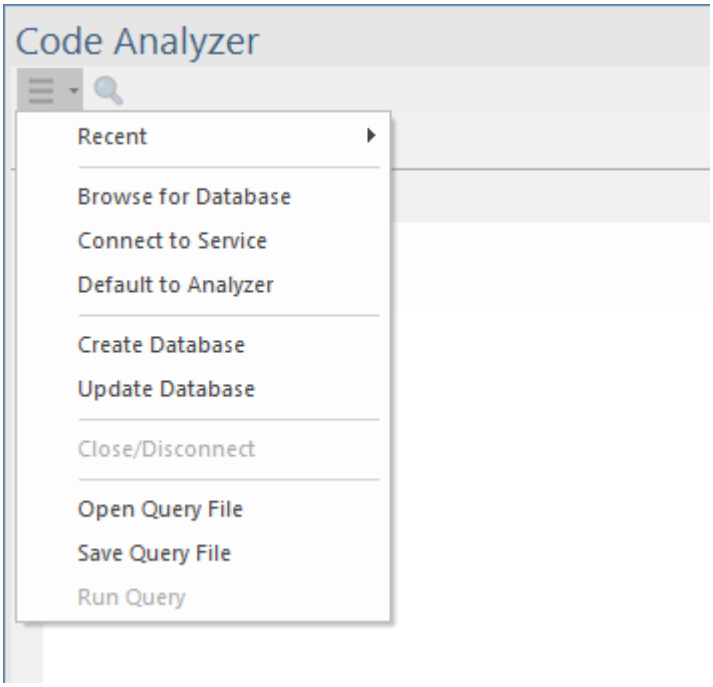
Il peut effectuer des requêtes très complexes sur les référentiels de code source à une vitesse fulgurante, soit localement, soit sur un service cloud Sparx Intel. Les requêtes sont composées à l'aide d'un langage de haut niveau développé par Sparx System. Le langage utilise un vocabulaire restreint mais expressif, facile à apprendre et permettant d'interroger les métriques de code beaucoup plus rapidement que les méthodes conventionnelles.

Accéder

Ruban	Développer > Code source > Analyseur de code

Menu de l'analyseur de code

Le menu Code Analyzer s'affiche lorsque vous cliquez sur l'icône  dans le coin supérieur gauche de la fenêtre.



Le menu propose diverses commandes pour les activités associées à l'utilisation de Code Analyzer, notamment le choix d'une base de données Code Miner à utiliser, la mise à jour de la base de données Code Miner et l'ouverture d'un fichier Query pour modification.

Ce tableau décrit chacune des commandes de menu.

Commande	Description
Récent	Affiche un sous-menu qui fournit une liste des connexions récentes aux services et aux fichiers de base de données locale .
Rechercher une base de données	Affiche une dialogue « Sélecteur de fichiers », vous permettant de rechercher une base de données Code Miner sur votre ordinateur.
Se connecter au service	Affiche la dialogue « Connexion à la base de données Code Miner », dans laquelle vous spécifiez les détails de connexion pour une (liste de) services de base de données Code Miner .
Par défaut sur l'analyseur	La sélection de cette option entraîne la connexion automatique de Code Analyzer au service Code Miner configuré pour le script Analyseur d'Exécution actif, lorsque Code Analyzer est démarré.
Créer une base de données	Affiche la dialogue « Créer une base de données Code Miner », qui vous permet de créer une base de données Code Miner à partir d'un référentiel de code source dans le système de fichiers.
Mettre à jour la base de données	Affiche la dialogue « Mise à jour de la base de données Code Miner », qui vous permet d'effectuer une mise à jour incrémentielle d'une base de données Code Miner existante, pour incorporer les modifications récentes aux fichiers de code source.
Fermer/Déconnecter	Se ferme ou se déconnecte de la bibliothèque ou du service de base de données Code Miner .
Ouvrir le fichier Query	Affiche une dialogue « Ouvrir un fichier » vous permettant de choisir un fichier de requête mFQL dans le système de fichiers.

Enregistrer le fichier Query	Affiche une dialogue « Enregistrer le fichier » vous permettant d'enregistrer la requête mFQL actuelle dans un fichier nommé .
Exécuter Query	Exécute l'intégralité de la requête ou une sélection du contenu de la requête saisie dans l'éditeur de l'onglet ' Query '. Raccourci F6.

Avant d'utiliser l'analyseur

Avant de pouvoir utiliser Code Analyzer, vous devez d'abord créer une base de données Code Miner ou en localiser une existante à laquelle Code Analyzer peut accéder. La création d'une base de données Code Miner est résumée ici, ou vous pouvez lire une description détaillée dans la rubrique d'aide *Création d'une nouvelle base Code Miner* .

Selon l'emplacement de la bibliothèque que vous utiliserez, vous devez soit :

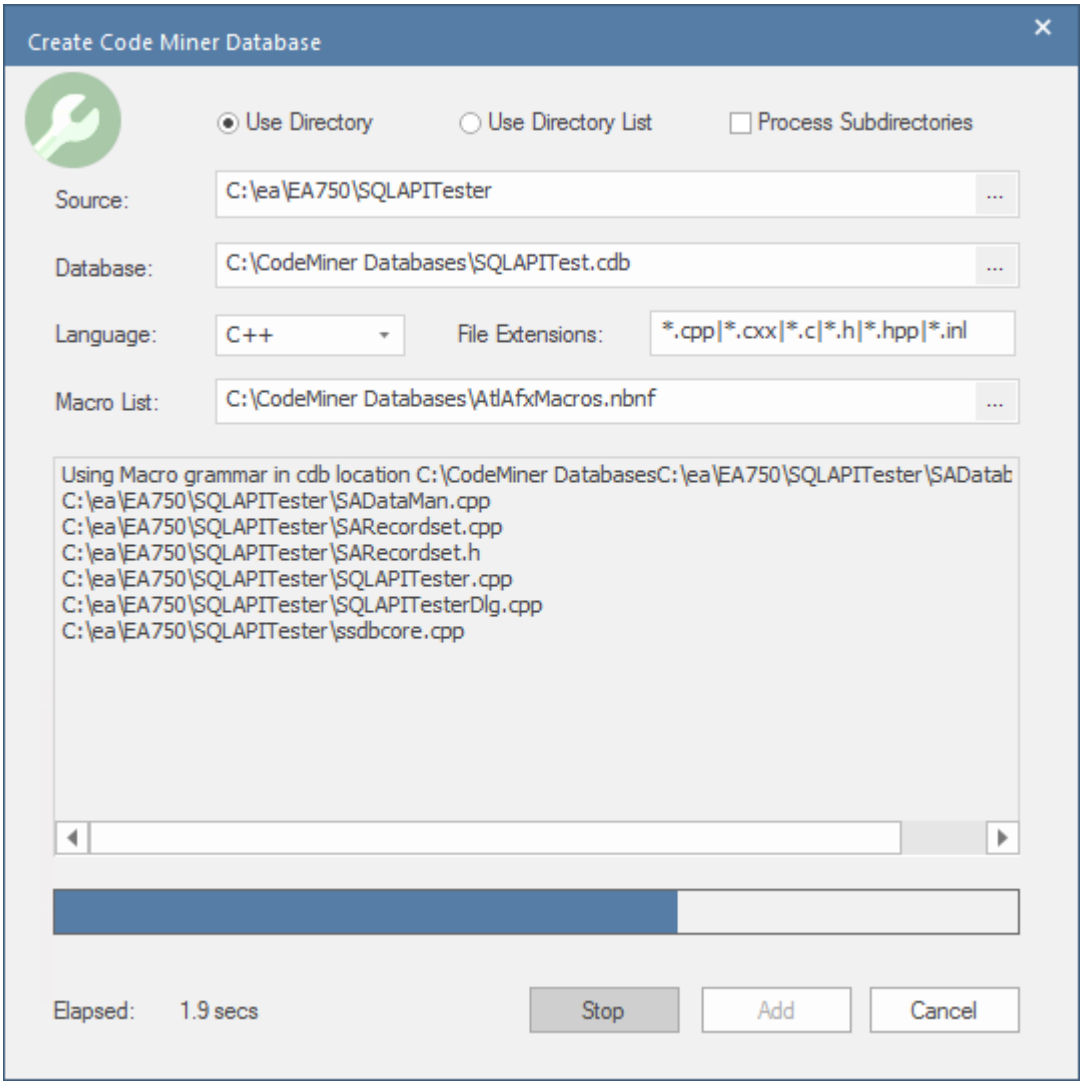
- Sélectionnez un fichier de bibliothèque Code Miner à utiliser, ou
- Connectez-vous à un service qui héberge une base de données Code Miner .

Une fois que vous avez terminé ces étapes, vous êtes prêt à commencer à écrire et à exécuter des requêtes dans Code Analyzer.

Création d'une base de données Code Miner

Les bases de données Code Miner sont construites à partir de référentiels de code source. Le processus est similaire à la compilation de code, utilisant la grammaire du langage pour analyser des fichiers individuels.

Il existe deux types de build : complet et incrémentiel. La build complète initiale peut prendre un certain temps, mais les builds incrémentielles suivantes sont incroyablement rapides.



Utiliser un répertoire comme entrée

Vous pouvez sélectionner un seul dossier comme racine du code source que vous souhaitez compiler. Avec cette option, vous pouvez choisir d'inclure des sous-répertoires

Utilisation d'une liste de répertoires

Parfois, vous souhaitez utiliser plusieurs projets, mais tous les projets ne se trouvent pas dans un seul répertoire. Dans ce cas, vous pouvez créer un fichier texte répertoriant le chemin complet de chaque dossier que vous souhaitez inclure et spécifier ce fichier texte dans le champ « Source ». Chaque chemin de répertoire doit être répertorié sur une ligne distincte.

c:\mesprojets\project1\tools\scintilla

c:\mesprojets\project2\src

d:\meslibs\lib1\src

Si vous souhaitez traiter de manière récursive les sous-répertoires d'un répertoire, faites précéder le chemin d'un point d'exclamation comme celui-ci :

!d:\meslibs\lib1\src

Toute ligne commençant par un caractère # est traitée comme un commentaire.

inclure la scintilla

c:\mesprojets\project1\tools\scintilla

Langue

Dans ce champ, vous spécifiez la langue utilisée dans le code source à partir duquel cette base de données Code Miner est construite.

Les langages disponibles sont : C++, C# , Java, XML, MDGTechnology et Custom.

Liste des macros

Lorsque le langage sélectionné est 'C++', le champ de sélection 'Liste de macros' s'affiche. Pour C++, le succès et la profondeur des informations compilées dans la base de données peuvent être inextricablement liés à l'utilisation de macros. Ce champ peut être utilisé pour sélectionner un fichier de macro nBNF qui sera utilisé comme composant de grammaire auxiliaire pour la compilation.

Par défaut, le fichier de macro sera par défaut le fichier de macro dans le dossier d'installation Enterprise Architect . Vous êtes libre de modifier ou d'étendre le contenu de ce fichier en fonction de vos besoins, par exemple lorsque vous devez corriger des erreurs signalées dans le fichier log de compilation.

Grammaire

Sparx Systems a développé des grammaires pour toutes les langues répertoriées dans la liste de sélection déroulante ; C++, C# , Java, XML et aussi MDGTechnology. Pour ces langues, un fichier de grammaire intégré est utilisé.

Il existe également une option permettant de sélectionner une langue « personnalisée ». Lorsque « Personnalisé » est sélectionné, le champ « Grammaire » s'affiche. Ce champ est utilisé pour spécifier un fichier contenant la grammaire de votre langue personnalisée. Le Code Miner utilisera ensuite cette grammaire pour analyser le code source écrit dans ce langage.

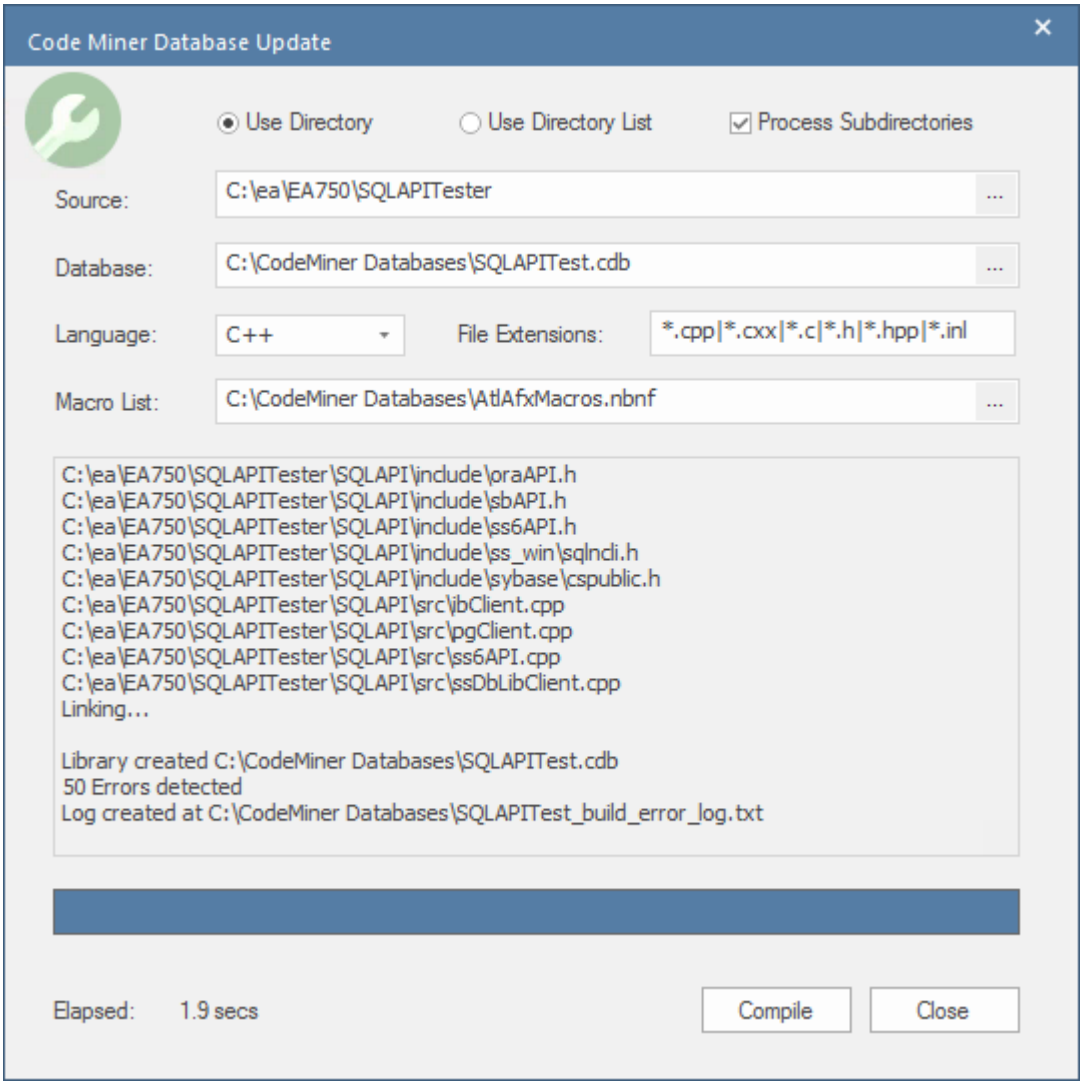
Les utilisateurs qui développent une langue personnalisée devront spécifier les règles de grammaire pour cette langue et les enregistrer dans un fichier nBNF. L'éditeur de grammaire d' Enterprise Architect est conçu spécifiquement à cet effet.

Le Help Topic *Grammar Framework* fournit des informations détaillées sur l'écriture d'une grammaire nBNF.

Mise à jour d'une base de données Code Miner

De temps en temps, vous souhaiterez mettre à jour votre base de données Code Miner . Généralement, lorsque vous avez apporté des modifications à votre code source, mais également après avoir mis à jour un fichier de grammaire ou étendu un fichier de macro.

Le processus de mise à jour d'une base de données est très similaire à la création d'une nouvelle base de données, mais plus rapide car vous ne partez pas de zéro. Choisissez simplement l'option de menu « Mettre à jour la base de données ». La dialogue « Mise à jour de la base de données Code Miner » s'affichera. Les champs de saisie seront remplis avec les valeurs de la dernière build. Procédez comme pour « Création d'une base de données Code Miner ».



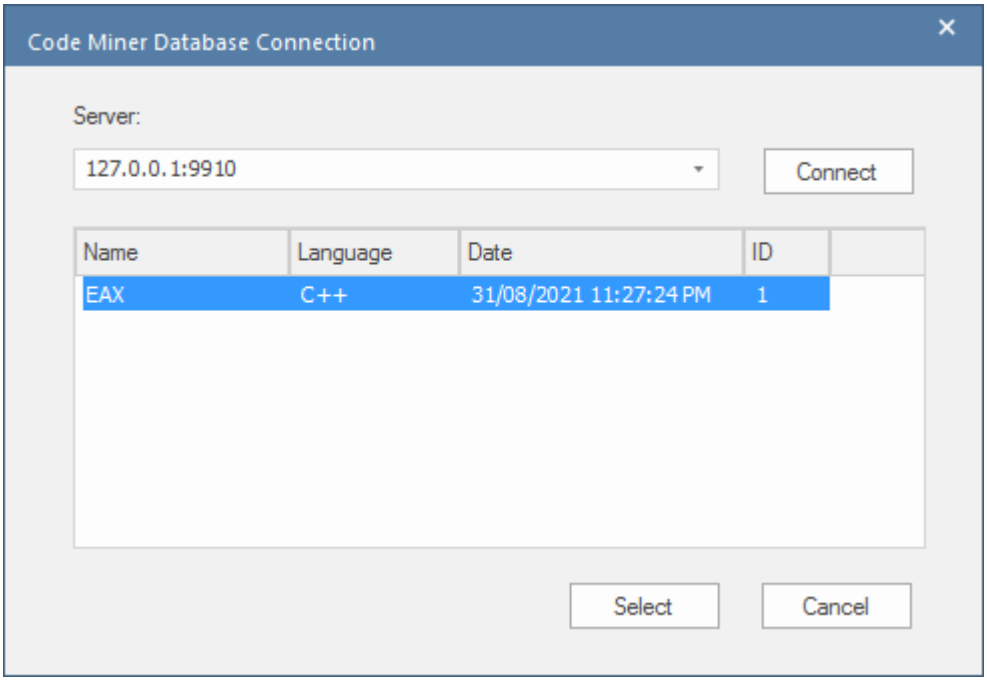
Sélection d'un fichier de base de données Code Miner

Si vous choisissez d'utiliser un fichier de bibliothèque pour votre base de données Code Miner , choisissez l'option de menu « Rechercher la base de données ». Cela affichera un « Sélecteur de fichiers », dans lequel vous pourrez rechercher et sélectionner un fichier *.cdb.

Connexion à un service

Lors de la connexion à un service, le dialogue liste toutes les bases de données hébergées par le service.

Vous pouvez choisir de sélectionner une base de données individuelle dans la liste, ou simplement cliquer sur le bouton Sélectionner, auquel cas les requêtes seront exécutées sur toutes les bases de données répertoriées par le service.

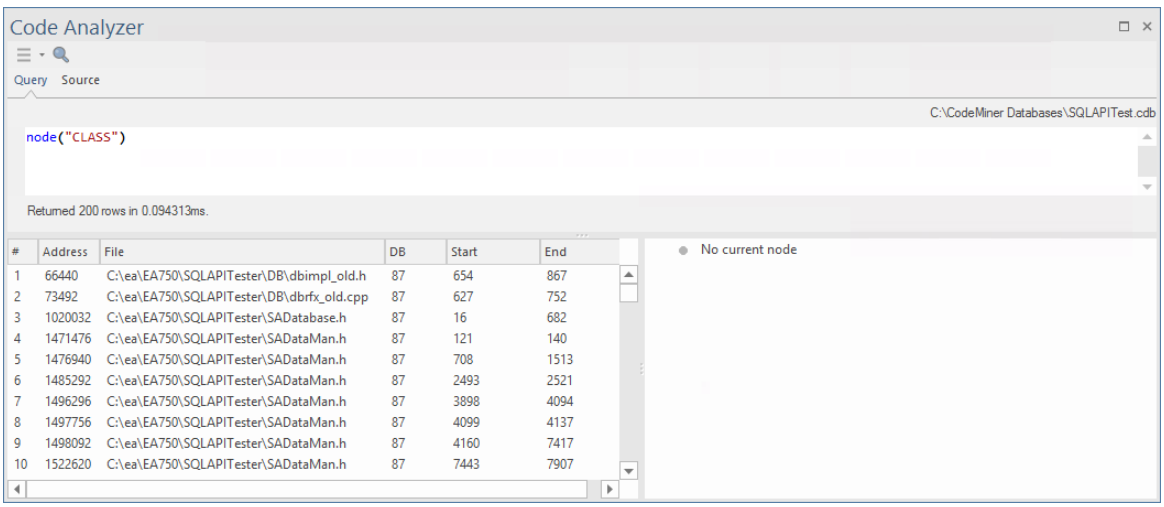


Exécution de requêtes

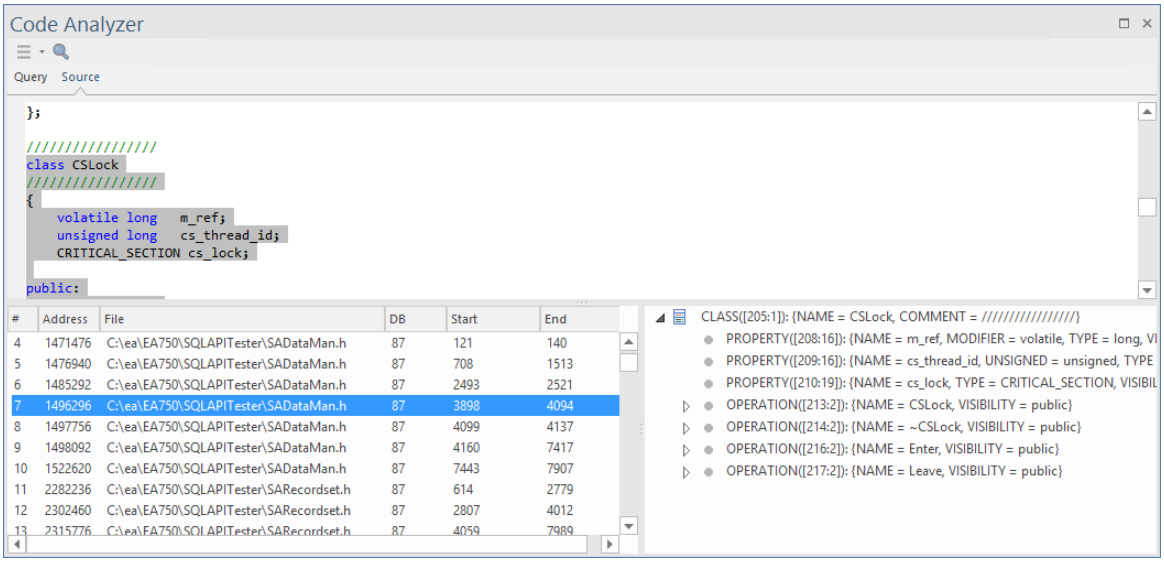
Une fois connecté à une base de données Code Miner , vous êtes prêt à commencer à exécuter des requêtes.

Pour exécuter une requête, sélectionnez l'onglet Query dans la fenêtre Code Analyzer, saisissez votre requête, puis cliquez sur l'icône pour exécuter la requête.

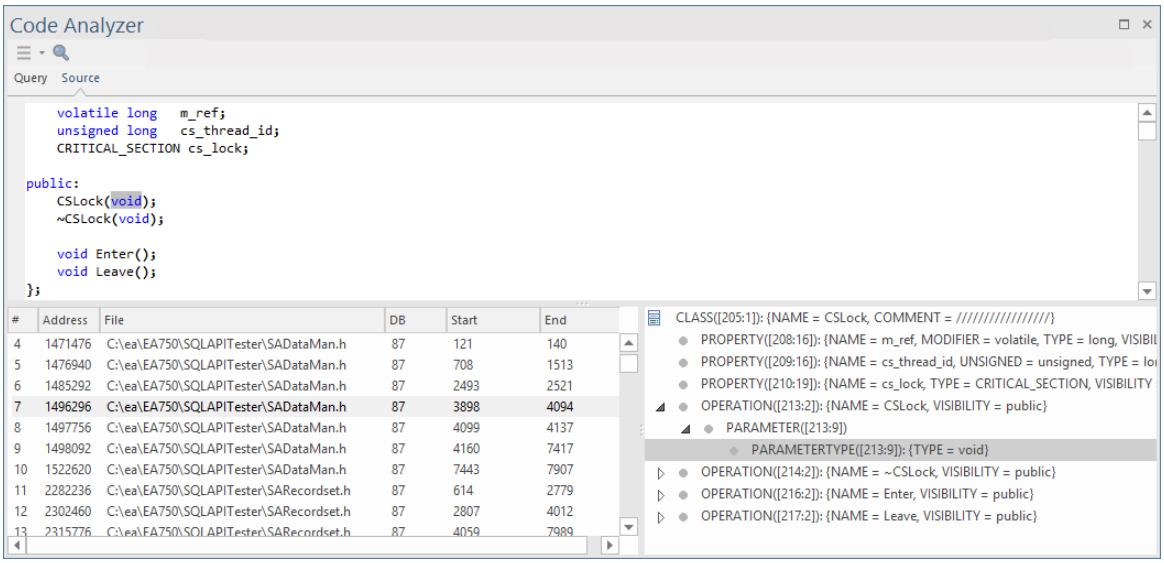
Dans cet exemple, nous avons exécuter un nœud de requête simple ("CLASS"), qui renverra tous les nœuds 'Class' trouvés dans la base de données Code Miner .



En sélectionnant un résultat dans le panneau inférieur gauche, l'onglet 'Source' est activé et affiche le code source correspondant au nœud sélectionné. Les détails de ce nœud de classe sont affichés dans le panneau inférieur droit.



La sélection d'un élément de détail dans le panneau inférieur droit entraîne un rétrécissement de la sélection dans le code source, comme indiqué ici.



Exemple Query - Intersection

As an example, this mFQL query finds all the classes that have an operation named GetOption.

andat("CLASS", item("OPERATION", "NAME", "GetOption"), node("CLASS"))

This clause returns a set of operations for which the 'NAME' value is "GetOption":

item("OPERATION", "NAME", "GetOption")

This clause returns a set of all Class nodes:

node("CLASS")

Formal syntax:

andat(string:rule, set:left, set:right)

'andat' takes the set of operations (left), applies the rule "CLASS" (only include rows that have a CLASS parent), then intersects that set with the set of all known classes (right). If the intersection succeeds, the operation node is added to the result set, otherwise it is excluded.

Le langage Query - mFQL

Le langage de requête utilisé avec Code Analyzer est décrit dans son intégralité dans la rubrique d'aide *Code Miner Query Language (mFQL)* .

Une brève description et quelques exemples sont également présentés ici.

Le langage mFQL est basé sur des ensembles. Chaque instruction fonctionne en utilisant les différents types d'opérations sur les ensembles, dont il n'en existe que quelques-unes.

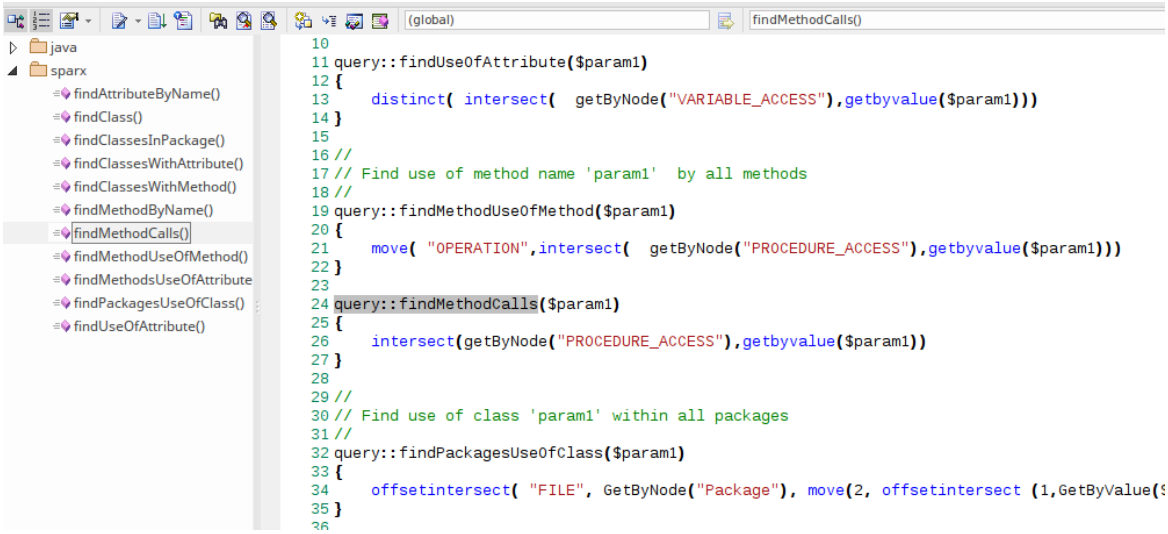
Cadre Code Miner

Le système Code Miner offre un accès rapide et complet aux informations du code source existant. En analysant tout le code source et en stockant l'arbre de syntaxe abstraite résultant dans une base de données optimisée en lecture, le système fournit un accès complet à tous les aspects du code source original, dans un format compréhensible par machine.

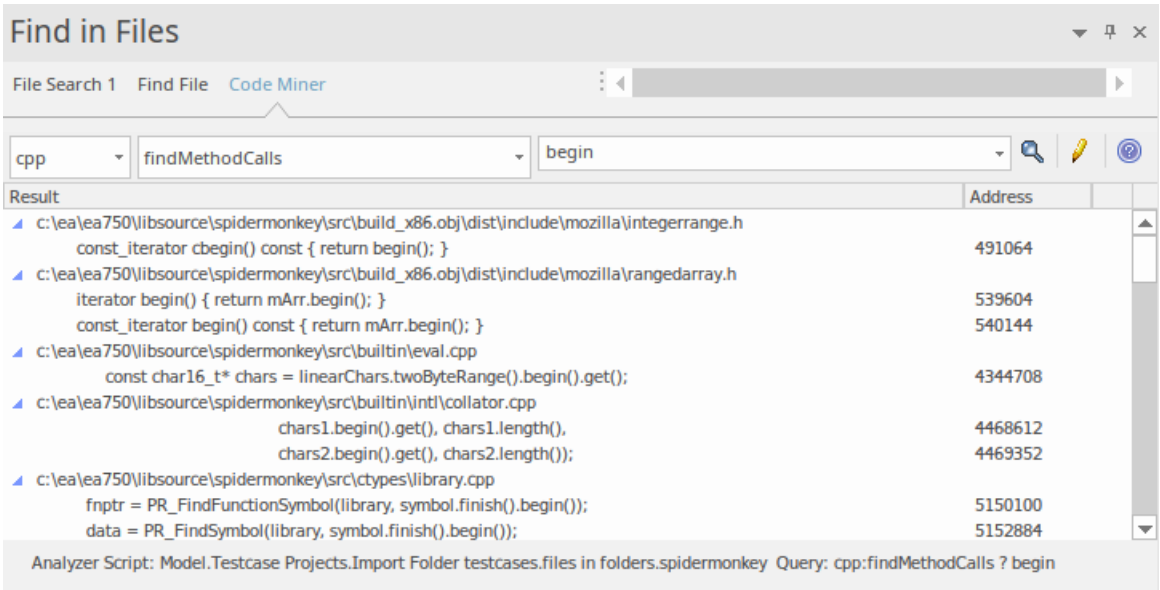
L'objectif principal du système est de fournir un accès rapide et efficace aux données cachées dans le code source. De grands efforts ont été déployés pour garantir des performances maximales, tout en fournissant les interfaces les plus simples possibles. En conséquence, le système peut être utilisé pour analyser la structure du programme, calculer des métriques, tracer des relations et même effectuer une refactorisation.

Les informations des bases de données Code Miner sont récupérées à l'aide de requêtes écrites en langage Query Code Miner NBNF (mFQL), le propre langage de Code Miner . Le langage lui-même est raisonnablement simple, fournissant un petit nombre de commandes. Aussi simple que soit le langage, il supporte des requêtes de taille et de complexité arbitraires. La conception offre des performances extrêmes pour toutes les requêtes, grandes et petites.

Cette fonctionnalité est disponible à partir d' Enterprise Architect version 14.1.



Le Code Analyzer d' Enterprise Architect , ses outils de recherche et les fonctionnalités Intelli-sense de ses éditeurs de code utilisent tous les informations extraites de ces bases de données.

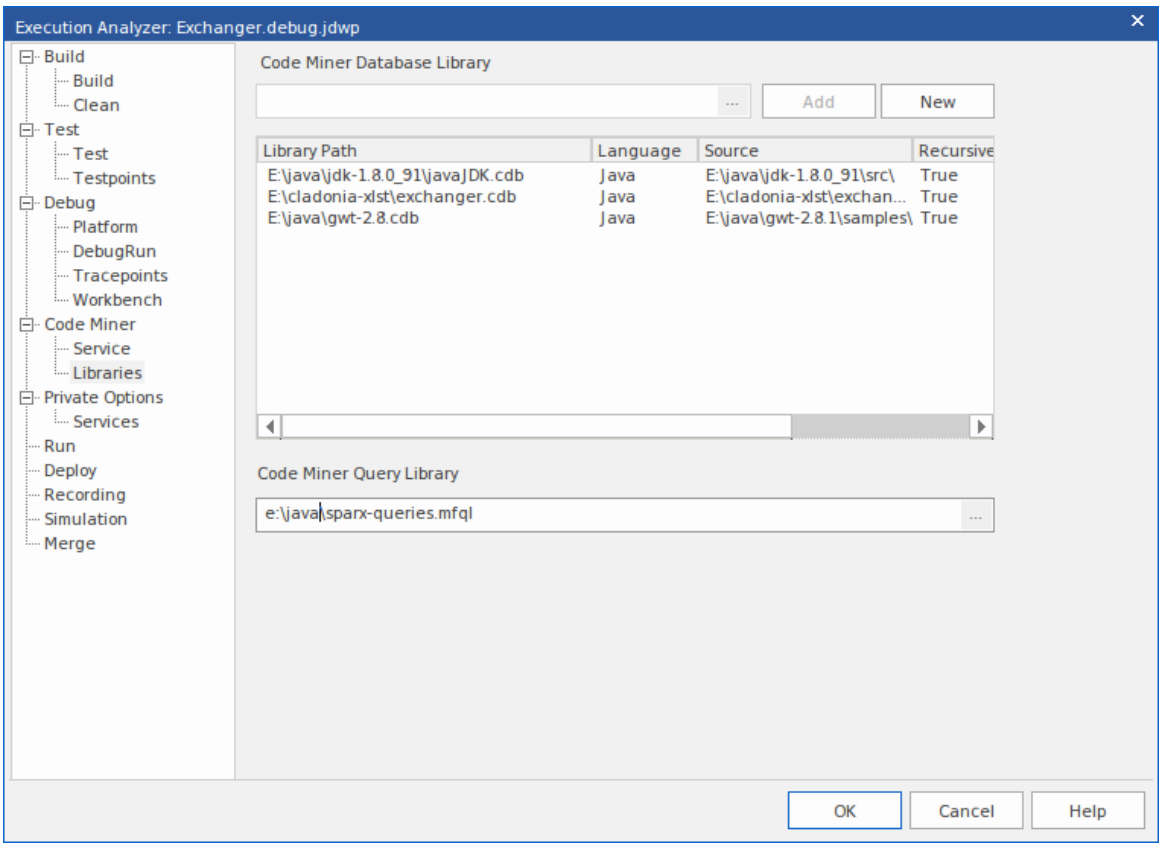


Le script Analyzer actuellement actif, ainsi que les paramètres de requête, sont indiqués en bas de la page « Code Miner » de l'outil de recherche.

Bibliothèques Code Miner

Les bibliothèques Code Miner sont gérées dans Enterprise Architect à l'aide de l' Éditeur de Script Analyseur . Ces bibliothèques sont une collection de bases de données Code Miner , dont une existe normalement pour chaque framework ou projet. L' Éditeur de Script Analyseur permet de créer de nouvelles bases de données et d'ajouter, de mettre à jour ou de supprimer des bases de données existantes. Ensemble, ces bases de données forment la Bibliothèque Code Miner utilisée par les fonctionnalités Code Analyser et Intelli-sense d' Enterprise Architect . La bibliothèque peut être utilisée localement ou déployée sur un emplacement de serveur où elle peut desservir plusieurs clients. Vous sélectionnez le scénario à utiliser sur la page « Sparx Intel Service » du script de l'analyseur.

Cette fonctionnalité est disponible à partir d' Enterprise Architect version 14.1.



Accéder

Dans la fenêtre Analyseur d'Exécution , localisez et double-cliquez sur le script requis - la dialogue de l'éditeur de script s'affichera. Dans cette dialogue , sélectionnez la page « Code Miner > Bibliothèques ».

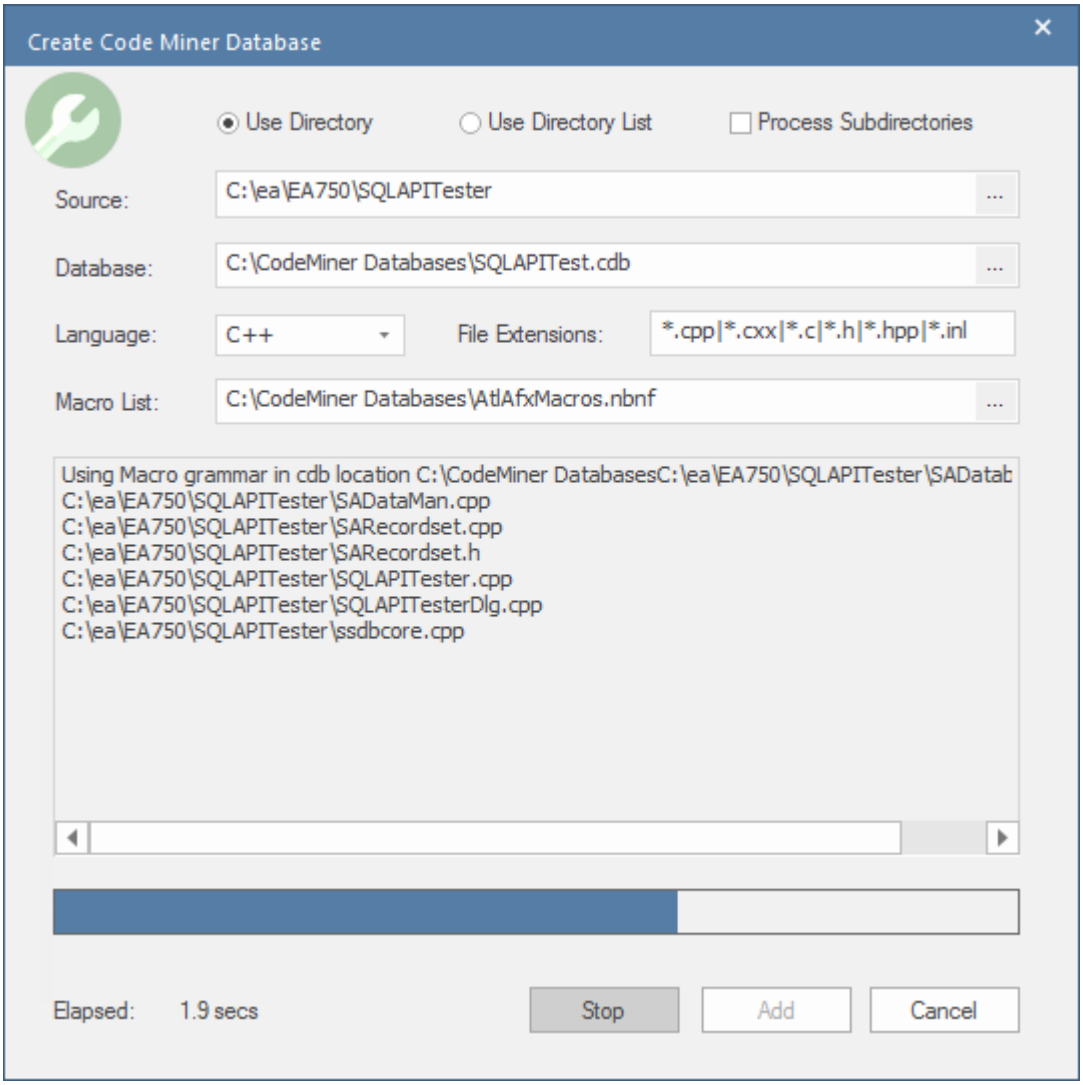
Ruban	Exécuter > Outils > Analyseur, ou Develop > Source Code > Analyseur d'Exécution > Edit Scripts d'Analyseur

Création d'une nouvelle base de données

Sur le ' Code Miner | Sur la page Bibliothèques de Éditeur de Script d'Analyzer, cliquez sur le bouton 'Nouveau' pour créer une nouvelle base de données.

Dans la dialogue 'Créer une base de données Code Miner ', spécifiez le(s) dossier(s) contenant le code source du projet,

sélectionnez le langage de programmation et entrez le chemin de destination de la base de données Code Miner . Lorsque vous cliquez sur le bouton « Compiler », les détails de la construction sont affichés dans la fenêtre log .



Une fois le processus terminé, cliquez sur le bouton « Ajouter » pour ajouter la base de données nouvellement créée à la bibliothèque.

Pour des informations détaillées sur la création de nouvelles bases de données, veuillez consulter la rubrique d'aide *Création d'une nouvelle base de données Code Miner* .

Ajout d'une base de données existante

Sélectionnez une base de données Code Miner existante à l'aide du bouton de sélection " ... " dans le champ du chemin de la base de données.

(Les bases de données Code Miner ont l'extension de fichier .CDB), puis cliquez sur le bouton Ajouter. Les détails sur la base de données sont répertoriés dans la bibliothèque. Les informations présentées affichent la grammaire du langage de programmation utilisée pour créer la base de données. Le chemin de base du code analysé lors de la construction est également indiqué et si le processus d'analyse a été appliqué de manière récursive via des sous-répertoires.

Mise à jour d'une base de données

De temps en temps, lorsque vous mettez à jour le code source d'un projet, vous souhaiterez mettre à jour la base de données Code Miner créée à partir de ce code source.

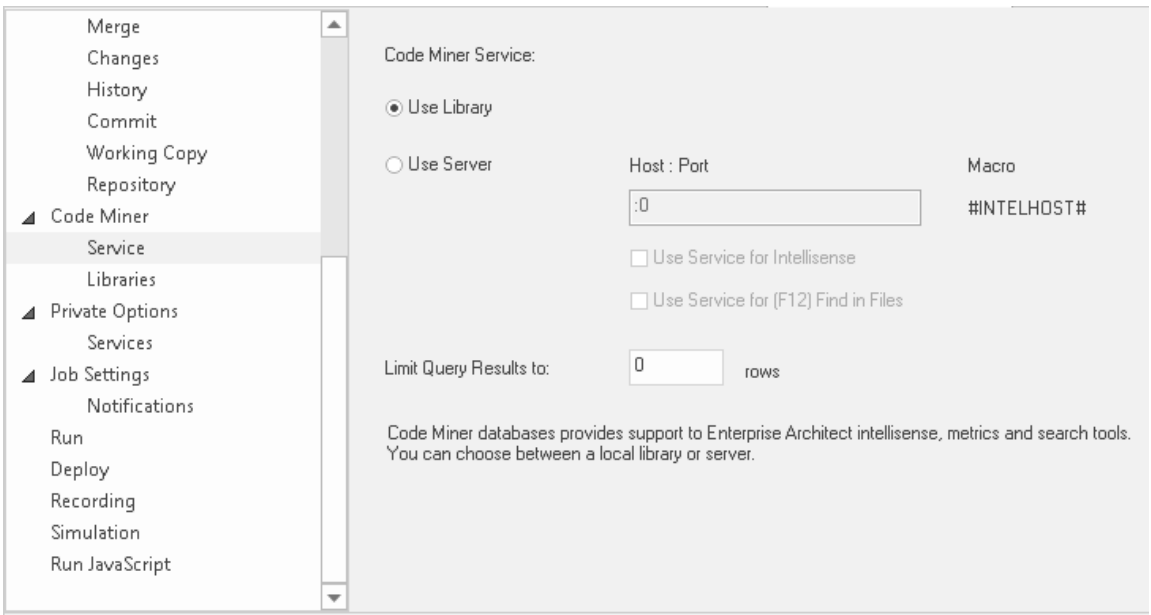
Pour mettre à jour une seule base de données Code Miner , sélectionnez-la dans la liste, cliquez-droit et choisissez 'Mettre à jour la sélection' dans son menu contextuel. Une dialogue similaire à la dialogue « Créer une base de données » s'affichera. Cliquez sur le bouton 'Compile', le Code Miner recréera la base de données à partir de la base de code mise à jour.

Suppression d'une base de données

Pour supprimer une seule base de données Code Miner , sélectionnez-la dans la liste et choisissez « Supprimer la sélection » dans son menu contextuel.

Configuration Enterprise Architect pour utiliser une Bibliothèque Code Miner

Dans un script Enterprise Architect Analyzer, choisissez la page 'Sparx Intel Service' et sélectionnez 'Utiliser Bibliothèque '. Enterprise Architect obtient ensuite ses informations Intelli-sense à partir des bases de données répertoriées dans la section « Bibliothèques » du script Analyzer actuellement actif.



Création d'une nouvelle base de données Code Miner

Code Analyzer d' Enterprise Architect , les fonctionnalités Intelli-sense de ses éditeurs de code et ses outils de recherche utilisent tous des bases de données Code Miner .

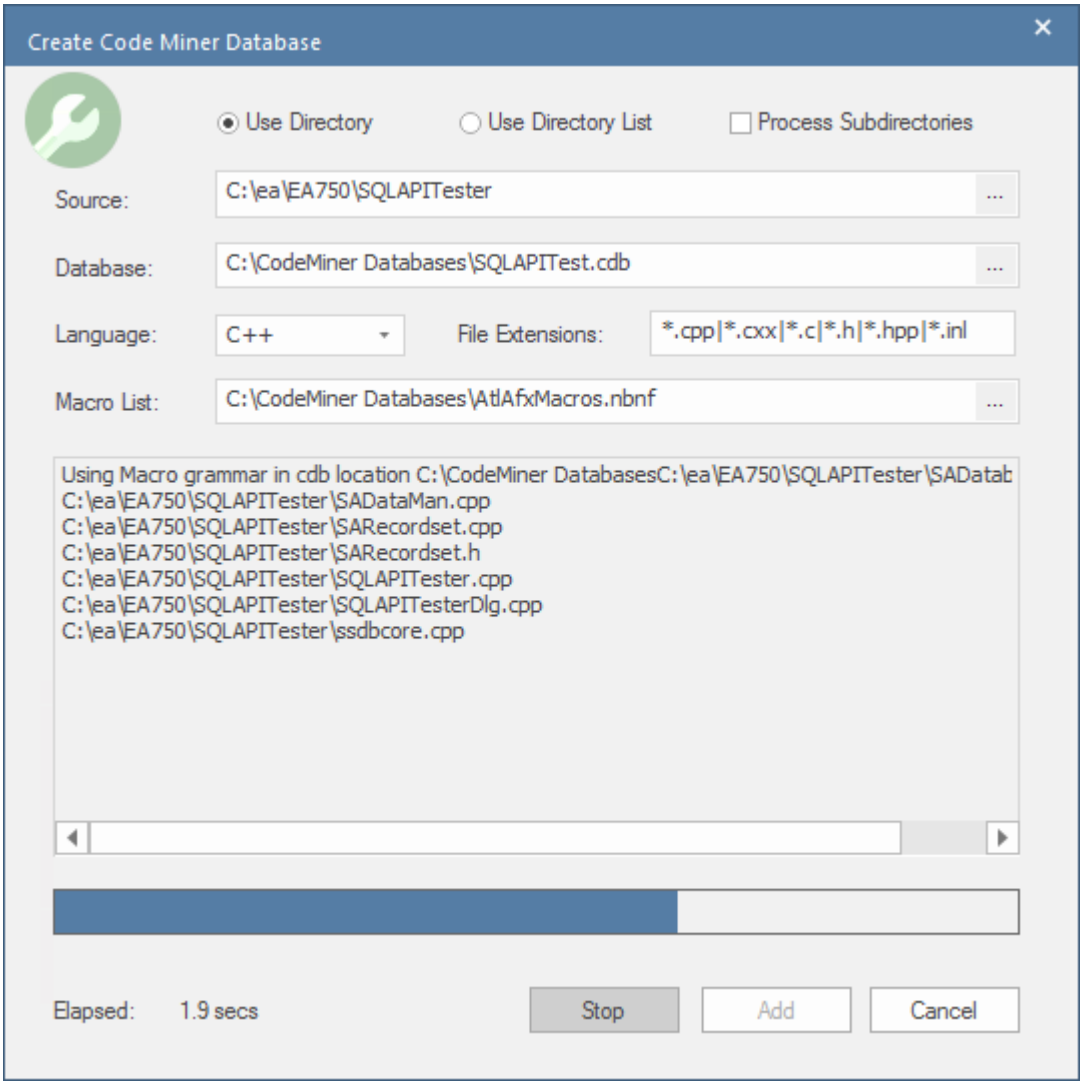
Une base de données Code Miner est créée en analysant les fichiers de code source selon les règles de grammaire du langage sélectionné et en stockant l'arbre de syntaxe abstraite résultant, dans une base de données optimisée en lecture. Une ou plusieurs bases de données peuvent être combinées pour former une Bibliothèque Code Miner .

Accéder

Fenêtre de l'analyseur de code	Dans la fenêtre Code Analyzer, cliquez sur le bouton de menu, , dans la barre d'outils, puis choisissez l'option de menu « Créer une base de données ».
Exécution Éditeur de Script Analyseur	La fenêtre Éditeur de Script de Analyseur d'Exécution étant ouverte, sélectionnez la page ' Code Miner > Bibliothèques', puis cliquez sur le bouton 'Créer'.

Créer Dialogue de base de données Code Miner

La dialogue « Créer une base de données Code Miner » est utilisée pour lancer le processus d'analyse des fichiers de code source afin de créer une base de données Code Miner . Sur le dialogue , vous spécifiez une plage d'entrées utilisées par le processus, telles que le dossier de code source, le fichier de langue et de liste de macros, ainsi que le nom du fichier de sortie. Les champs dialogue sont décrits dans le tableau présenté ci-dessous.



Champ	Description
Utiliser le répertoire	Sélectionnez cette option lorsque tous les fichiers source à traiter résident dans un seul répertoire. Lorsque cette option est sélectionnée, la case à cocher « Traiter les sous-répertoires » est activée.
Utiliser la liste des répertoires	Sélectionnez cette option lorsque le code source de votre projet réside dans plusieurs répertoires distincts. Dans ce cas, vous utilisez le champ 'Source' pour spécifier un fichier contenant une liste de répertoires contenant le code source à traiter.
Sous-répertoires de processus	Cette case à cocher est activée lorsque l'option « Utiliser le répertoire » est sélectionnée. Lorsqu'il est sélectionné, le fichier de code source résidant dans n'importe quel sous-répertoire du répertoire « Source » spécifié sera également traité.
Source	Ce champ permet de préciser le ou les répertoires contenant les fichiers de code source qui seront traités pour créer la base de données Code Miner . Lorsque l'option 'Utiliser le répertoire' est sélectionnée, ce champ est utilisé pour spécifier le dossier racine dans lequel rechercher les fichiers de code source. Lorsque l'option « Utiliser la liste des répertoires » est sélectionnée, ce champ est

	utilisé pour spécifier un fichier créé par l'utilisateur contenant une liste de noms de chemin d'accès aux répertoires contenant les fichiers sources à traiter. Cliquer sur le bouton ... ouvre une dialogue « Sélecteur de fichiers », qui vous permet de rechercher et de choisir un fichier avec l'extension « .ssdirlist ». Pour plus d'informations, consultez la section *Fichier de liste de répertoires* ci-dessous.
Base de données	Ce champ spécifie le nom de chemin complet du fichier de base de données Code Miner qui sera créé. L'extension de nom de fichier « .cdb » est utilisée pour ce fichier.
Langue	Il s'agit d'une liste déroulante dans laquelle vous spécifiez la langue utilisée dans les fichiers de code source en cours de traitement. Il existe un certain nombre de langages pour lesquels Enterprise Architect fournit support « intégrée ». (Il existe des grammaires intégrées utilisées pour analyser les langues prises en charge). Il existe également une option permettant de choisir une langue « personnalisée ». Si vous choisissez d'utiliser une langue personnalisée, vous devrez créer votre propre grammaire pour support l'analyse de cette langue. Lorsque l'option 'Personnalisée' est sélectionnée, le champ 'Fichier de grammaire' s'affichera, vous permettant de spécifier le fichier qui définit votre grammaire personnalisée.
Extensions de fichiers	Ce champ répertorie un certain nombre d'extensions de nom de fichier généralement associées aux fichiers de code source de la langue choisie. Seuls les fichiers dont les extensions de nom de fichier correspondent à celles de la liste seront traités par l'analyseur. Vous pouvez ajouter ou supprimer des extensions de nom de fichier en fonction de vos besoins.
Liste des macros	Lorsque le langage sélectionné est 'C++', le champ de sélection 'Macro List' s'affiche. Le champ Liste de macros vous permet de spécifier un fichier qui fournit une liste de macros que l'analyseur doit ignorer lorsqu'il les rencontre. Pour le langage C++, les macros posent un problème à l'analyseur car elles masquent les constructions du langage natif. L'ajout du nom d'une macro au fichier de liste de macros et la mise à jour de la base de données effaceront généralement toutes les erreurs liées à cette macro. Pour plus d'informations, consultez la section *Extension du fichier de liste de macros* ci-dessous.
Fichier de grammaire	Sparx Systems a développé des grammaires pour toutes les langues répertoriées dans la liste de sélection déroulante. C++, C# , Java, XML et aussi MDGTechnology. Il existe également une option permettant de sélectionner une langue « personnalisée ». Les utilisateurs qui développent un langage personnalisé devront spécifier les règles de grammaire pour ce langage et les enregistrer dans un fichier nBNF, afin que Code Miner puisse analyser correctement le code source écrit dans ce langage. L'éditeur de grammaire d' Enterprise Architect est conçu spécifiquement à cet effet. Lorsque vous sélectionnez « Personnalisé » comme langue, vous devez ensuite spécifier le fichier de grammaire que vous avez créé pour cette langue, afin que Code Miner puisse analyser correctement votre code source. Le Help Topic *Grammar Framework* fournit des informations détaillées sur l'écriture d'une grammaire nBNF.
Fenêtre de sortie	La fenêtre de sortie montre la progression de l'analyse des fichiers de code source. Une fois terminé, il affiche également les noms du fichier de base de données et du fichier log créés ainsi que le nombre d'erreurs rencontrées.

Bouton Compiler/Arrêter	Le bouton 'Compiler' permet de lancer le traitement. Ce bouton se transforme en bouton « Stop » une fois le traitement commencé, permettant à l'utilisateur d'abandonner l'opération.
Bouton Ajouter	Une fois qu'une base de données a été compilée, le bouton « Ajouter » peut être utilisé pour ajouter cette base de données à une Bibliothèque Code Miner . Plusieurs bases de données peuvent être ajoutées pour créer une bibliothèque couvrant de nombreux projets de code source. **Note** : Lorsque la dialogue 'Créer une base de données Code Miner ' est ouverte depuis la fenêtre Code Analyzer, le bouton 'Ajouter' ne s'affiche pas.

Fichier de liste de répertoires

Si vous choisissez de spécifier un fichier de liste de répertoires, vous devrez créer un fichier texte simple utilisant l'extension de nom de fichier « .ssdirlist », qui répertorie le chemin complet de chaque répertoire que vous souhaitez traiter, avec un chemin par ligne. Par exemple:

c:\mesprojets\project1\tools\scintilla

c:\mesprojets\project2\src

d:\meslibs\lib1\src

Si vous souhaitez traiter de manière récursive les sous-répertoires d'un répertoire répertorié, faites précéder ce chemin d'un point d'exclamation comme celui-ci :

!d:\meslibs\lib1\src

Toute ligne commençant par un caractère # est traitée comme un commentaire :

inclure la scintilla

c:\mesprojets\project1\tools\scintilla

Extension du fichier de liste de macros

Pour le langage C++, les macros présentent un problème pour les grammaires car elles cachent les constructions du langage natif. L'analyseur ne peut pas effectuer de substitution sur les macros car elles sont souvent définies de manière conditionnelle et l'analyseur n'a aucune idée de l'architecture. Le fichier Macro List fournit une liste de macros que l'analyseur doit ignorer lorsqu'il les rencontre.

Lorsque vous créez une base de données Code Miner pour un référentiel de code source C++, des erreurs peuvent s'afficher. Lorsqu'une erreur se produit, utilisez le log des erreurs pour rechercher et inspecter la ligne de code à l'origine de l'erreur. Cela identifie presque toujours une macro à l'origine de l'échec grammatical. L'ajout de ce nom à la liste des macros et la mise à jour de la base de données effaceront généralement toutes les erreurs liées à cette macro.

Par exemple, le log des erreurs affiche cette erreur :

C:\ea\EA750\SQLAPITester\SQLAPI\include\asa\sqlfuncs.h, ligne : 12, col : 18, symbole inattendu ','.

Après inspection, la ligne de code à l'origine de l'erreur est la suivante :

FUNC_INFO(externe , vide , _esqlentry_, sqlstop, (SQLCA *))

(Il existe également de nombreuses autres lignes similaires utilisant la macro 'FUNC_INFO'.)

Nous éditons donc le fichier de liste de macros par défaut, 'AtxAflMacros.nbnf', en ajoutant cette ligne :

"FUNC_INFO" " (" skipBalanced(" (" , ") ") ") " |

Cette ligne demande à l'analyseur, lorsqu'il rencontre la macro "FUNC_INFO" , d'appliquer la fonction skipBalanced(" (" , ") "), qui prend deux paramètres ; dans ce cas, ce sont les parenthèses ouvrantes et fermantes. Ainsi, l'analyseur est invité à ignorer tout ce qui se trouve entre les parenthèses ouvrantes et fermantes.

Lorsque la modification du fichier Macro List est enregistrée et que la base de données est recompilée (mise à jour), toutes les erreurs relatives à la macro "FUNC_INFO" ont été éliminées.

Apprenez Plus

- Grammar Framework
- Code Analyzer

Requêtes Code Miner

Code Miner queries are best considered as functions written in the Code Miner NBNF Query Language (mFQL). As such, they have unique names, can be grouped by namespace and can take one or more parameters. Queries are bundled together into one source file. This source file is identified to Enterprise Architect by naming it in your Analyzer Script.

When specified, the queries it contains are available in the Code Miner control. Parameters to these queries can be taken from selected text in a code editor, the model context or typed directly into the search field of the control.

This feature is available from Enterprise Architect Release 14.1.

```
188
189 namespace java
190 {
191 //
192 // Find all references
193 //
194 query::findByName($param1)
195 {
196     distinct(GetByValue( $param1 +))
197 }
198
199 query::findMethodByName($name)
200 {
201     move( 1, "METHOD", intersect( GetByNode("NAME"), GetByValue( $name ) )  )
202 }
203
204 query::findMethodCall($name)
205 {
206     filter( "METHOD_ACCESS", intersect(GetByNode("NAME"), GetByValue( $name )) )
207 }
208
```

This image illustrates an mFQL query from the Sparx Queries file distributed with Enterprise Architect installations. The syntax for composing an mFQL query and the mFQL language itself is described here.

Query Syntax

The syntax for composing mFQL queries is:

namespace

{

query:*name*([$*param1 [*, $*param2]]*)

{

mfql-expression

}

}

where:

- *namespace* names the collection of queries
- *name* is the 'function' name of the query
- *$param1 and $param2* are placeholders for argument substitutions at runtime
- *mfql-expression* is an mFQL expression

Langage Query Code Miner (mFQL)

The Code Miner system provides fast and comprehensive access to the information in existing source code. By parsing all source code and storing the resulting Abstract Syntax Tree (AST) in a read-optimized database, the system provides complete access to all aspects of the original source code, in a machine understandable format.

The core goal behind the system is to provide access to the data hidden within source code in a timely and effective manner. Great pains have been taken to ensure maximal performance, while providing the simplest interfaces possible. As a result the system can be used to analyze program structure, calculate metrics, trace relationships and even perform refactoring.

mFQL

mFQL is the query language of the Code Miner. The language itself is reasonably simple, providing a small number of commands. Simple as the language is, it supports queries of arbitrary size and complexity. The design provides extreme performance for all queries, great and small.

The language is set-based; it operates primarily on sets of abstract data obtained through discrete vertical indices. For our purposes, a set is an ordered array of numbers, each of which is a pointer to a node in the AST Store. A discrete vertical index provides a mechanism to retrieve sets by discrete value.

The language includes the three basic set-joining operations. These are 'intersect', 'union', and 'except'. The 'except' join is, more precisely, a 'symmetric difference' join. A 'complement' join can be achieved by using a short sub-query; this is detailed in the 'except' join documentation. The 'offsetIntersect' join is also discussed in detail there.

The Code Miner database provides three discrete vertical indices in its AST Store. These indices are 'node name', 'attribute name', and 'attribute value'. Each vertical index can be queried for a discrete value, which will return a set of all nodes where that value is present. The three vertical indices are queried using the functions 'getByNode', 'getByName' and 'getByValue', respectively.

Set 'traversal routines' provide mechanisms to filter sets based on patterns in the AST. The traversal routines are either destructive (move) or non-destructive (filter). Destructive traversals modify the set member values to point to the target node; non-destructive traversals ensure the target node exists. In both cases, nodes that cannot complete the traversal are removed.

Please note that all traversals in mFQL are upwards. Downwards traversals are technically complex, as a node could have any number of child nodes. Conversely, upward traversals are much simpler, with every node having zero or one parent node. For these reasons, downward traversals are not supported in the query language.

Although there are only a small number of operations in mFQL, the language is capable of expressing very finely grained and complex queries. The language is functional in design, and supports arbitrary nesting calls.

mFQL queries execute at lightning speed. The backend database was designed from the ground up for read performance. The query parser was hand optimized. Knowing that it always has pure ordered sets, the low-level code takes several shortcuts to perform joins with minimal work effort.

In order to use nBNF effectively one must possess a working knowledge of the target language, and an intimate knowledge of the grammar used to parse it.

Le langage mFQL

This section provides a list of Code Miner NBNF Query Language (mFQL) queries with explanations and comments.

The queries shown here demonstrate different capabilities and different approaches to exploring and extracting data using mFQL and the Code Analyzer in Enterprise Architect. The mFQL queries help make the syntax human-readable and intuitive, and have been extended in Enterprise Architect to include additional functions necessary to do real things with Code Miner databases.

The Query Language

String parameters are indicated by **string**, set parameters are indicated by **set** and number parameters are indicated by **numbers**.

Notes

1. Case sensitivity is defined by the case sensitivity of the language of the source code used to populate the database. If the source language is case sensitive (such as C++) all string literal parameters are case sensitive. If the source language is case insensitive (such as SQL) all string literal parameters are case insensitive.
2. Hierarchical traversals in mFQL are generally upwards. Downwards traversals are not optimal, as a node might have any number of child nodes. Upward traversals are much simpler, with every node having zero or one parent node. Downward-looking queries such as 'children' only query one level down.
3. Synonyms of some keywords are provided to better express a query intent or action in particular circumstances, and to support legacy queries. Synonyms are simple alternatives for the base function keyword. For example, 'type(str)' can be written as 'node(str)' or 'byNode(str)' or 'getByNode(str)'. The current specified version is the preferred one, with the synonyms only intended for use in exceptional circumstances.

Statement	Description
type(value)	type(value) Extracts a set based upon node name. The exact name for a node is defined by the grammar used to parse the original source. In this example, find all nodes within the database of type "CLASS". type("CLASS") *Synonyms:* • node • byNode • getByNode
with(name)	with(name) Searches the database for any element that has a named attribute matching the search string. The value of the attribute is ignored - this is a query for the attribute NAME only. All nodes with one or more attributes of the specified name are returned. If a single node has two attributes of the same name, one instance of that node is returned. This example will find all elements in the database that have an attribute named "Type": with("Type") *Synonyms:* • name

	• byName • getByName
find(value) find([+] value [+ value] [+])	find(value) find([+] value [+ value] [+]) Search the database for any element having an attribute value with the provided search term. The match is case sensitive and must match the whole word. You can extract a set based upon an attribute value; when extracting nodes by attribute value, the values of all attributes for the node are considered. Wildcards allow for specifying a subset of attribute values for a node. Wildcards can be used at either the beginning or end of a value specification: • A leading concatenation symbol allows for any number of attributes preceding the first matched attribute • A trailing concatenation symbol allows for arbitrary trailing attributes In both cases, if the node would match without wildcards, it will match with them – the wildcard specifies any number of leading/trailing attributes, including none. In this example, we retrieve a set of nodes that have their last two attributes being "." and "sun". The leading concatenation symbol specifies that any number of attributes (including none), with any value, can exist before the matched attributes, but none can follow. find(+ "." + "sun") The next example has a trailing wildcard. Any node with "com", "." and "sun" as the first three attributes will be returned. Any number of trailing attributes can exist. find("com" + "." + "sun" +) Both wildcards can be used together. In this example nodes with attributes with the three specified values as names, in order, regardless of leading or trailing attributes, will be returned. find(+ "com" + "." + "sun" +) Example: Find all nodes in the database that have any attribute with a value of "CString": find("CString") Example: Find all nodes in the database with a set of attributes having these values in this order: find("com" + "." + "sun") *Synonyms:* • value • byValue • getByValue
has(name,value)	has(name, value) Finds all elements that have a named attribute with the value supplied. Unlike the intersection of 'find' and 'with', this query will only return rows with an exact name/value pair. has("Type","CString")
having(name, value, set)	having (name, value, set) Finds all elements within the supplied set that have a named attribute with the given value. Similar to 'has' but supplies a predefined input set to search. Whether to use

	'has' or 'having' is generally determined by the kind of query structure being used, its depth and its readability.
	
	Example 1: Find all Property elements with a name of "m_strName" that have a Type attribute of CString:
	`having("Type","CString",this("PROPERTY","NAME","m_strName"))`
	
	Example 2: Extend Example 1 to only include those that store a CString *:
	`having("Reference","*",`
	`having("Type","CString",this("PROPERTY","NAME","m_strName")))`

this(type,name,value)	this(type, name, value) Function finds one or more elements that have a matching TYPE, and WITH a named attribute having the specified VALUE. Example: Find all operations named "Import Solution": `this("OPERATION","NAME","ImportSolution")` *Synonyms:* • object • item
like(name,like,set)	like(name, like, set) like(name, like, set, caseSensitivity) Finds a set of elements that have an attribute that starts with the search sub-string. Note that this is not a fully wild-carded search but is case sensitive and must be an exact match for the length of the search string. caseSensitivity: specify one of: • "CaseSensitive" • "CaseInsensitive" • "Default" - uses the code language to determine which to use Note that case-insensitive searching can be slower. Example: Find all Classes in the database whose NAME attribute starts with "CMapStr": `like("NAME","CMapStr",gettype("CLASS"))`
contient (nom, contient, ensemble)	contient (nom , terme_recherche , ensemble) contient (name , search_term , set , caseSensitivity) Recherche un ensemble d'éléments dont l'attribut contient la sous-chaîne de recherche. Semblable à 'like' sauf que cela recherchera toute la string , pas seulement depuis le début. caseSensitivity : spécifiez l'un des éléments suivants : • "Sensible aux majuscules et minuscules" • "Insensible à la casse" • "Par défaut" - utilise le langage de code pour déterminer lequel utiliser Note que la recherche insensible à la casse peut être plus lente. Note que cette recherche peut être plus lente que l'utilisation de "J'aime". Exemple : Rechercher toutes les classes dans la base de données dont l'attribut

	NAME contient "MapStr" : contient ("NOM" , "MapStr" , gettype ("CLASSE"))
and(set1,set2,...)	and(set1, set2, ...) Returns the intersection of nodes between two or more sets. To be included in the final set, an element must exist in ALL the input sets. *Synonyms:* • intersect(set, set,...) • {set, set, ...}
union(set1,set2,...)	union(set1,set2, ...) Returns the distinct union of ALL nodes present in the input sets. *Synonyms:* • or(set, set ...) • [set, set ...]
ancestor(str,set)	ancestor(str, set) ancestor(num, set) ancestor(num, str, set) The ancestor function traverses each node in a set of a number of parent nodes, excluding any nodes that fail the traversal. The number of nodes to traverse, the name of the target node for the traversal, or both can be provided as parameters. When the number of nodes is provided, but the target node name is not, any nodes with the specified number of parents will pass the traversal. Any node that runs out of parents will be dropped from the set. When the name of the target is specified, but the number of nodes to traverse is not, any nodes with a parent with a matching name, at any point in the hierarchy, will pass the traversal. Any node with no matching parent is excluded. When both the number of nodes and the target name are provided, only nodes that have a parent node with the specified name, at the specified offset, pass the traversal. All other nodes are removed from the set. In this example the set hasParameter("CString","&",1) is moved up to an ancestor node named "OPERATION". If the move fails the node is dropped from the result. ancestor("OPERATION",hasParameter("CString","&",1)) In this example the set is moved up one rung to its parent. If there is no parent, the node is dropped from the result. ancestor(1,hasParameter("CString","&",1)) In this example the set is moved up three steps to its parent->parent->parent . If there is no such node, the node is dropped from the result. ancestor(3,hasParameter("CString","&",1)) *Synonyms*: • move
filter(str,set)	filter(str, set) filter(num, set)

	filter(num, str, set) The filter function is the same as the 'ancestor' function, except that it returns nodes from the original child set rather than new ancestor nodes. If a node is unable to pass the specified traversal, it is removed from the set. Nodes that pass the traversal are left in place, unmodified. In this example the set hasParameter("CString","&",1) is tested for an ancestor node named "OPERATION". If the move fails the node is dropped from the result. The result set is a set of parameter types that meet the criteria. filter("OPERATION",hasParameter("CString","&",1))
match(NameA,setA,Name B,setB)	match(NameA,setA, NameB,setB) 'Match' takes two input sets and two attribute names and returns all those in 'setA' that have a matching record in 'setB', as determined by comparing the values of the named attributes 'strA' and 'strB'. That is, a 'setA' row is included if the value of attribute 'strA' in 'setA' exists in 'setB' as the value of an attribute of name 'strB'. 'Match' is useful for finding where one element feature is used in a different context elsewhere in the database. For example, where a unique element name or GUID is referenced by another element. In this example, we match the attribute named 'TYPE' from the right set to the attribute 'NAME' in the left set. The result will be all "CLASS" type objects from the left set with NAME == TYPE(s) as specified in the right set. match("NAME",type("CLASS"),"TYPE",this("PROPERTY","NAME","m_pLink"))
graph(targetType, targetName, linkType, linkName, start)	graph(targetType, targetName, linkType, linkName, start) Find a recursive set of elements that form some kind of graph when linked by attribute pairs, in a manner similar to 'match'. The starter set is queried for all owned instances of the linkType with link Name and these are matched against a new query based on the targetType with targetName. The new set is filtered in a manner similar to 'match', and all elements in the new query that share the same NAME/VALUE pair as from the starter set are kept; all others are discarded. The resultant set is then fed back into the original set as the starter for the next iteration, with the results at each stage being added together to form the final result set. Example: Return the Class hierarchy for a Class named "Car". graph("CLASS","NAME","GENERALIZATION","GENERAL",this("CLASS","NAME","Car"))
prune(set_test,str,set_base)	prune(set_test, str, set_base) prune(set_test, num, set_base) For two sets of nodes, temporarily move one set UP to the named or numeric position in its ancestry and filter out any nodes that do not exist by strict intersection in the TEST set. The first set is the TEST set, the right or last set is the BASE set. The set returned is all the elements in the BASE set that, when moved to the TEST position, matched something in the TEST set. The returned nodes are the original nodes from the BASE set and are not moved up when returned. Example 1 finds the set of parameter types used for operation parameters named "CustomerName" across the whole database. prune(this("PARAMETER","NAME","CustomerName"),"PARAMETER",type("PARAMETERTYPE"))

	Example 2 finds all Properties of a Class named Customer, assuming the grammar used to compile the database placed the Property definition two hierarchy levels below the Class definition. prune(this("CLASS","NAME","Customer"),2,type("PROPERTY"))
andat(str,test,base)	andat(str, base, test) andat(num, base, test) andat(num, str, base, test) For two sets of nodes, temporarily move one set UP to the named or numeric position in its ancestry and filter out any nodes that do not exist by strict intersection in the TEST set. The first set is the TEST set, the right or last set is the BASE set. The set returned is all the elements in the BASE set that, when moved to the TEST position, matched something in the TEST set. The returned nodes are the original nodes from the BASE set and are not moved up when returned. Similar to 'prune', this query supports additional options and structures the inputs in a different order to facilitate different kinds of stacked searches. The 'andat' function performs both a non-destructive tree traversal and an intersect join in one operation. Each node in the left set is traversed according to parameters provided, then the result of the traversal is intersected with the right set. If the intersect passes, the original node is added to the result set. If the intersect fails, the node is excluded from the result set. The traversal parameters for 'andat' are the same as for 'ancestor' and 'filter'. For more information about the traversal parameters, see the 'ancestor' function. Example: For the set of all "PROPERTY" nodes in the database, move them up to a parent node of type CLASS and then intersect the result with the right hand set - in this case a CLASS named CDiagram. All nodes that pass this test are returned as PROPERTY nodes, effectively giving the set of all properties of the Class CDiagram. andat("CLASS",type("PROPERTY"),this("CLASS","NAME","CDiagram")) *Synonyms:* • offsetIntersect • offsetx
unique(left,right) / except(left,right)	unique(left, right) except(left, right) Except joins return sets that contain any nodes from either set that do not appear in both sets. This join is similar to a bitwise XOR operation. In set theory, this type of join is referred to as a 'symmetric difference join'. {1, 2, 3} excepted with {2, 3, 4} results in {1, 4}
omit(left,right) / exclude(left,right)	omit(left, right) exclude(left, right) Exclude joins return a set that contains all nodes from the left set that do not appear in the right set. In set theory, this type of join is referred to as a 'relative complement join'. {1, 2, 3} complemented with {2, 3, 4} results in {1}
differ(name,set,name,set)	differ(name, set, name, set) Return a set of nodes that do not have a matching row in another set, using a NAME/VALUE pair from each set to match on.

	Example: This more complex example tests the complete set of Generalizations for a Class hierarchy and identifies missing or unresolved Class names in the total inheritance hierarchy. Like the 'match()' function discussed later, this function iterates over attribute name/value pairs as specified in the left and right input sets, but only includes rows in the final set where there is NO match. differ("GENERAL", children("GENERALIZATION", graph("CLASS","NAME","GENERALIZATION","GENERAL", this("CLASS","NAME","CMainFrame"))), "NAME", graph("CLASS","NAME","GENERALIZATION","GENERAL", this("CLASS","NAME","CMainFrame")))
children(type,set)	children(type, set) Return a set of child nodes of a specified type for one or more parents in the source set. For all children regardless of type, use an empty string. For example, in the first query we return ALL first level children of the CMainFrame Class. In the second query we restrict the nodes returned to be of type "REGION" only. children("",this("CLASS","NAME","CMainFrame")) children("REGION",this("CLASS","NAME","CMainFrame"))

childcount(num,type,set)	childcount (num,type,set)
	Return nodes that exactly match the number of specified children of a specified type. For example, only return operations that have 5 parameters.
	An example usage is in specifying an exact operation signature, so we check firstly that parameter1 and parameter2 match the type we are querying for, then move those to their operation ancestor and intersect the result with the operation name "GetFromCache" we are interested in. To rule out spurious hits with operations having more than 2 parameters, we explicitly add childcount(2, ...) to ensure we only get operations that have 2 parameters.
	
	childCount(2,"PARAMETER",
	{
	ancestor("OPERATION",hasParameter("CString","&",1)),
	ancestor("OPERATION",hasParameter("CString","&",2)),
	this("OPERATION","NAME","GetFromCache")
	}
	)

byAddress(num)	byAddress(num) The byAddress function is used in applying the results of one query to another. For example, we might have a node of particular interest, and want our query to return only nodes that join (in some way) to the specified node. byAddress(node: number)

	This example builds a set containing the single node related to the address specified: byAddress(11256)
byPosition(File, Offset)	byPosition(File, Offset) The byPosition function is used to return the inner-most node that covers a certain position in a file. This function is useful for locating a position in the AST based upon a file position.
distinct(set)	distinct(set) The distinct function ensures that a set has no duplicate values. All duplicate values are excluded from the result set.

Définir l'extraction

These procedures extract sets from discrete vertical indices. There are three indices available, each with a specific extraction function. String literal parameters to these functions could be case sensitive. Case sensitivity is defined by the language of the source code used to populate the database. If the source language is case sensitive (as C++ is) all string literal parameters are case sensitive. If the source language is case insensitive (as SQL is) all string literal parameters are case insensitive.

type

type(value: string)

Extract a set based upon a node name. The exact name for a node is defined by the grammar used to parse the original source. In this example, all nodes with the name "OPERATION" are returned.

type("OPERATION")

–

with

with(value: string)

Extract a set based upon attribute name. All nodes with one or more attributes of the specified name are returned. If a single node has two attributes of the same name, one instance of that node is returned. This example returns all nodes with one or more attributes named "NAMEPART".

with("NAMEPART")

find

find([+] value: string [+ value: string] [+])

Extract a set based upon an attribute value. When extracting nodes by attribute value, the value of all attributes for the node are considered. Wildcards allow for specifying a subset of attribute values for a node.

When a single value is provided, all nodes that have a single attribute with the value specified are returned. If a node has any other attributes, it is excluded. In this example, all nodes with exactly one attribute with the value of 'i' are returned.

find("i")

More than one value can be specified by using a concatenation symbol. When more than one value is specified, the resulting set will contain all nodes that have attributes with exactly the values specified, in the order specified. Any node with extra leading or trailing attributes is excluded. This example retrieves a set of all nodes with a set of three attributes with the values “com”, “.” and “sun”, in that order.

find("com" + "." + "sun")

Wildcards can be used at either the beginning or end of a value specification. A leading concatenation symbol allows for any number of attributes preceding the first matched attribute. A trailing concatenation symbol allows for arbitrary trailing attributes. In both cases, if the node would match without wildcards, it will match with them – the wildcard specifies any number of leading/trailing attributes, including none.

In this example, we retrieve a set of nodes that have their last two attributes being “.” and “sun”. The leading concatenation symbol specifies that any number of attributes (including none), with any value, can exist before the matched attributes, but none can follow.

find(+ “.” + “sun”)

The next example has a trailing wildcard. Any node with attributes “com”, “.” and “sun” as the first three attributes will be returned. Any number of trailing attributes can exist.

find(“com” + “.” + “sun” +)

Both wildcards can be used together. In this example, nodes with attributes named as the three values specified, in order, regardless of leading or trailing attributes, will be returned.

find(+ “com” + “.” + “sun” +)

—

Définir le parcours

ancestor

ancestor(count: number, source: set)

ancestor(value: string, source: set)

ancestor(count: number, value: string, source: set)

The 'ancestor' function traverses each node in a set up a number of parent nodes, excluding any nodes that fail the traversal. The number of nodes to traverse, the name of the target node for the traversal, or both can be provided as parameters.

- When the number of nodes is provided, but the target node name is not, any nodes with the specified number of parents will pass the traversal; any node that runs out of parents will be dropped from the set
- When the name of the target is specified, but the number of nodes to traverse is not, nodes with a parent with a matching name at any point in the hierarchy will pass the traversal; any node with no matching parent is excluded
- When both the number of nodes and the target name are provided, only nodes that have a parent node with the specified name at the specified offset pass the traversal; all other nodes are removed from the set

It is possible - even likely - that these calls will generate sets having duplicate values. This is by design, as the concrete rules for sets do not define them as being discrete. If (as in most cases) you want your set to be discrete, use the 'distinct' function described in the *The mFQL Language* Help topic.

This sample extracts a set of all nodes named 'OPERATION', then traverses each node up one level to its immediate parent. Any 'OPERATION' node with no parent is excluded.

ancestor(1, getByNode("OPERATION"))

This sample extracts a set of all nodes named 'OPERATION', then traverses each node up to the first 'CLASS' parent node. Any 'OPERATION' node with no 'CLASS' parent is excluded.

ancestor("CLASS", getByNode("OPERATION"))

This sample extracts a set of all nodes named 'OPERATION', then traverses each node up one level to its immediate parent. If the parent node is not a 'CLASS' node, or the node fails to traverse though a lack of parent nodes, it is excluded.

ancestor(1, "CLASS", getByNode("OPERATION"))

–

filter

filter(count: number, source: set)

filter(value: string, source: set)

filter(count: number, value: string, source: set)

The 'filter' function is the same as the 'ancestor' function, except that it does not modify nodes – it is non-destructive. If a node is unable to pass the specified traversal, it is removed from the set. Nodes that pass the traversal are left in place, unmodified.

It is often desirable to filter a set by the current node name. This can be used to ensure that the nodes returned from a 'with' or 'find' call are of a particular node type. This example returns all nodes with an attribute with the value of “CFoo”, where the resulting node is a “TYPE” node.

filter(0, “TYPE”, find(“CFoo”))

For more details on the use of the 'filter' function, see the 'ancestor' function.

—

Définir la jonction

and

and(left: set, right: set)

An 'and' join will return a set containing all nodes that exist in both the left and right set. This join is comparable to a bitwise AND operation. In set theory, this type of join is called an 'intersection'.

{1, 2, 3} intersected with {2, 3, 4} results in {2, 3}

This example returns a set that contains all nodes that have a single attribute with the name of "TYPE" and the value of "int".

```
and(
find("int"),
with("TYPE")
)
```

union

union(left: set, right: set [, right: set])

'Union' joins return a set that includes all nodes found in either the left or the right set. This join is used to combine the results of two or more sub-queries into a single set. A 'union' join is similar to a logical OR operation. In set theory, the 'union' join is known as a union.

The 'union' join is able to operate on more than two sets. The result is a set that contains all nodes from all supplied sets. The 'union' join is the only join able to operate on more than two sets.

The result of a 'union' join is always a discrete set, unless one of the source sets contained duplicates. This means that duplicates in source sets will be preserved, but the 'union' join itself will not generate duplicates.

{1, 2, 3} unioned with {2, 3, 4} results in {1, 2, 3, 4}

This sample creates a set containing all nodes with an attribute named “TYPE” or a single attribute with the value of “int”.

```
union(
find("int"),
with("TYPE")
)
```

except

except(left: set, right: set)

'except' joins return sets that contain any nodes from either set that do not appear in both sets. This join is similar to a bitwise XOR operation. In set theory, this type of join is referred to as a 'symmetric difference' join.

{1, 2, 3} excepted with {2, 3, 4} results in {1, 4}

For more information on the 'symmetric difference' join in set theory, see https://en.wikipedia.org/wiki/Symmetric_difference

This sample returns a set of all nodes with an attribute named "TYPE" but no single attribute with the value of "int", plus

all nodes with an attribute with the value of "int" that are not named "TYPE".

```
except(
 find("int"),
 with("TYPE")
)
```

exclude

exclude(left: set, right: set)

'exclude' joins return a set that contains all nodes from the left set that do not appear in the right set. In set theory, this type of join is referred to as a relative complement join.

{1, 2, 3} complemented with {2, 3, 4} results in {1}

This sample returns a set of all nodes with a value of “int” that are not “TYPE” nodes:

```
Exclude(
find(“int”),
with(“TYPE”)
)
```

andat

andat(count: number, left: set, right: set)

andat(value: string, left: set, right: set)

andat(count: number, value: string, left: set, right: set)

The andat function performs both a non-destructive tree traversal and an intersect join in one operation. Each node in the left set is traversed according to parameters provided, then the result of the traversal is intersected with the right set. If the intersect passes, the original node is added to the result set. If the intersect fails, the node is excluded from the result set.

The traversal parameters for andat are the same as for 'ancestor' and 'filter'. For more information about the traversal parameters, see the 'ancestor' function described in the *Set Traversal* Help topic.

This sample takes all “NAME” nodes, traverses them up one parent, and intersects them with a set of all “CLASS” nodes. If a “NAME” node passes both the traversal and intersect join, it is added to the result set. The result is a set of all “NAME” nodes whose immediate parent is a “CLASS” node.

```
andat(1,
type(“NAME”),
type(“CLASS”)
)
```

—

Service Intel Sparx

Le programme de service Sparx Intel offre aux projets de développement et aux acteurs un moyen d'obtenir des informations précieuses sur les bases de code et les cadres logiciels avec lesquels ils travaillent. Le service agit en tant que fournisseur pour les clients Enterprise Architect , permettant l'accès à Intelli-sense pour l'édition de code et des résultats de recherche perspicaces dans les outils de recherche.

Le service Sparx Intel fait partie du groupe Sparx Satellite Services. Le service peut exécuter sur un réseau local ou Cloud exécutant Microsoft Windows . Le service Sparx Intel Satellite peut être installé en tant que service Windows ou exécuter en tant que processus autonome. Le service permet à plusieurs clients Enterprise Architect d'accéder et d'interroger les mêmes informations à partir de nombreux domaines et frameworks logiciels différents.

Cette fonctionnalité est disponible à partir Enterprise Architect version 16.0

Configuration du service Intel Sparx

Le programme SparxIntelService.exe exécute un ou plusieurs services Intel pour Enterprise Architect . Le programme se trouve dans le même dossier d'installation Enterprise Architect et utilise un fichier de configuration qui nomme les services pouvant être exécuter sur la machine locale.

Dans les exemples de cette rubrique, le programme tentera d'utiliser le fichier c:\mystuff\myservices.config. Je rechercherai un service nommé *EA* et, s'il le trouve, je le démarrerai.

SparxIntelService.exe service d'écoute = EA config = c:\mystuff\myservices.config

The Config File Format

The configuration file has this format:

```
# comment
# comment
# comment
{                # start of service definition
   ...           # list of directives as pairs
}                # end of service definition
{                # start of service definition
   ...           # list of directives as pairs
}                # end of service definition
```

Comments are indicated by the # character.

If the config directive is omitted (not recommended), the program will look for a config file of the same name as the program, in the same directory as the program.

In this example the program will attempt to use the file SparxIntelService.config in the same folder:

SparxIntelService.exe listen service:EA

Directif	Description
nom	Lorsqu'un service est nommé sur la ligne de commande, le service avec l' attribut de nom correspondant sera démarré.
statut	Lorsque l'état
chargement paresseux	Lorsque lazyload est « true », toute base de données Code Miner sera chargée en retard jusqu'à ce qu'une requête Intel soit adressée au service.
niveau de journalisation	Définit le niveau d'informations enregistrées, sous la forme d'une combinaison de mots-clés { information, avertissement, erreur} séparés par un '|'. Par exemple: loglevel=Information|avertissement|erreur
sortie de déconnexion	Spécifie le chemin d'accès complet du fichier log dans lequel écrire. Par exemple: logoutput =c:\logfiles\intel-service-project1. log
base de données	Spécifie le nom de chemin complet de la base de données Code Miner à charger. Par exemple:

	base de données =c:\intel--service\project1.cdb Plusieurs directives « base de données » sont autorisées, chacune spécifiant une base de données différente.
permettre	Identifie l'adresse IP autorisée à se connecter au service sur le port. Par exemple: autoriser = hôte local autoriser = 127.0.0.1 Allow=172.160.* (les caractères génériques sont autorisés lorsque le « réseau » la directive a une valeur de « réseau » ou de « public », mais pas « local »)
réseau	Permet de restreindre les connexions de service. • local - le service n'écoutera sur aucune connexion autre que localhost • réseau - lorsqu'il est utilisé avec des directives génériques « autoriser », permet aux clients sur un caractère générique d'adresse IP autorisée de se connecter • public - autorise toute connexion
montrer	Lorsque « true », la fenêtre de console du service sera affichée ; Le défaut est faux'.
port	Le port sur lequel le service écoutera.

The Service Configuration Template

When choosing the 'Execute > Tools > Services > Code Miner Service > Edit Configuration File' ribbon option you display the Windows 'Save As' browser through which you can choose either the config file to open or where a file should be created.

If no config file is recorded in the registry and you specify a non-existent filename, that file is created, filled with a 'bare bones' configuration skeleton and saved. The selected/new configuration is then shown in the Enterprise Architect default editor.

The 'bare bones' template is shown here.

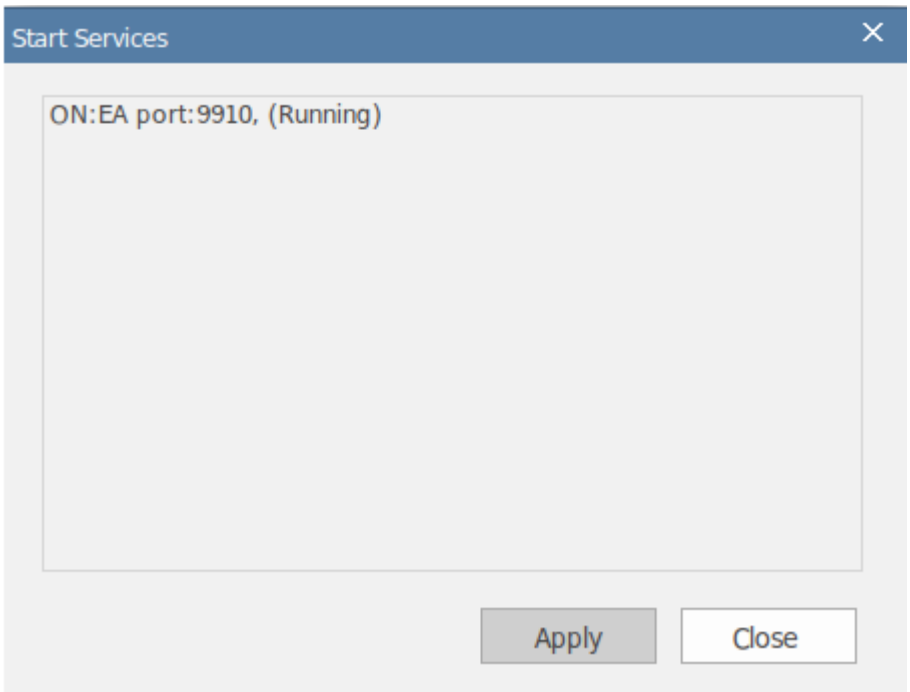
```
#-----------------------------------------------------------------------------
# Sparx Intel Service Configuration File
# -----------------------------------------------------------------------------
# This file is used to describe one or more intel services and the code miner databases that they support
# This file can be used in EA to manage a number of services on the local machine
# -----------------------------------------------------------------------------
# Service Attributes
# -----------------------------------------------------------------------------
# name                    The unique name of the service in this file
# status                  "ON" - service can run, "OFF" service will never run
# lazyload                "true" - databases are loaded n demand, "false" - databases are loaded when service
starts
# port                    Unique Port number that service will listen on and EA will connect to
# network                 [optional,default=local] Restricts service to listening to locahost only (local), to a range
```

```
of addresses (network) or any address (public)
# allow                    Allows a specific IP address or wildcard IP address to connect (if network is NOT local)
#                                (There can be multiple allow directives present)
# autoupdate          "true" - will detect updates to listed databases and reload them, "false" default, changes are not
detected
# show                     [optional,default=false] shows the console window for the service
# logoutput                [optional] The path of a log file which service can write to
# loglevel                 [optional] The levels of information logged. Combine with '|' character, e.g.: {
information|warning|error }
# database                 [Required] The full path to a codeminer database which usually has the .cdb file
extension
#                                (There can be multiple database directives present)
#
# ---------------------------------------------------------------------------------
# Attribute Values
# ---------------------------------------------------------------------------------
#
# <string>   - text. (do not include quotes)
# <boolean>  - text, { true, false, ON, OFF }
# <path>     - fully specified file path to codeminer database
# <number>   - digits
# ---------------------------------------------------------------------------------
#
{
        name=<string>,
        status=<boolean>,
        lazyload=<boolean>,
        port=<number>,
        allow=<string>,
        allow=<string>,
        network=<string>,
        autoupdate=<string>,
        show=<boolean>,
        logoutput=<string>,
        loglevel=<string>,
        database=<path>,
        database=<path>,
        database=<path>
},
{
        name=Project1,
        status=ON,
        lazyload=TRUE,
```

```
        allow=localhost,
        allow=127.0.0.1,
        port=9999,
        autoupdate=true,
        database=c:\Project1\Project1.cdb
}
```

Les options du ruban de service Sparx Intel

Lorsqu'un fichier de configuration de service existe, vous pouvez le modifier ou l'exécuter à l'aide d'un certain nombre d'options disponibles dans l'option de ruban « Exécuter > Outils > Services » dans le groupe d'options de menu Code Miner .

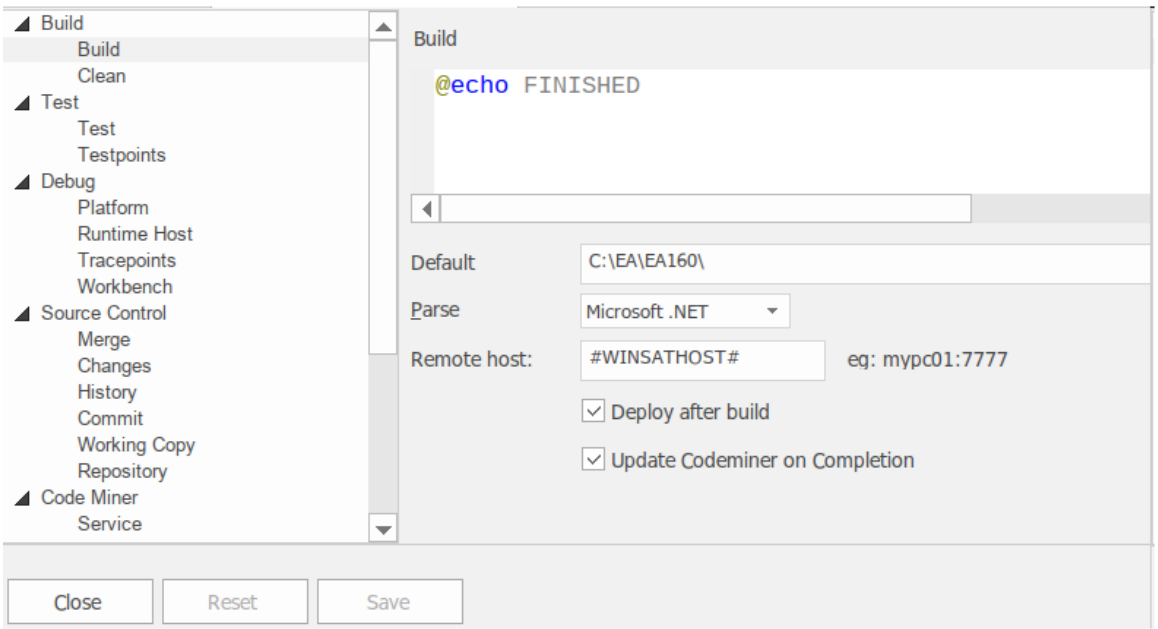
Option	Description
Statut Vue de tous les services	(Au-dessus de toutes les catégories de service.) Cette option affiche une vue qui répertorie l'état de chaque service Enterprise Architect nommé dans le fichier de configuration actuel, ainsi que son état.
Démarrer	Cette option lit le fichier de configuration de service actuel et démarre les services configurés pour exécuter , et arrête l'exécution des services qui ne sont pas configurés pour exécuter . Un service est configuré si : 1. Il est nommé dans le fichier de configuration. 2. Il a le statut d'attribut : ON. 
Arrête tout	Cette option arrête tous les services en cours d'exécution.
Modifier le fichier de	Cette option prompts à utiliser le fichier de configuration du service, puis ouvre ce

configuration	fichier dans un éditeur de texte Enterprise Architect . Le système se souvient de l'endroit où se trouve le fichier. 31 # <number> - digits 32 # ---------- 33 # 34 { 35 name=project1, 36 status=ON, 37 lazyload=true, 38 port=9910, 39 allow=localhost, 40 network=local, 41 autoupdate=true, 42 show=true, 43 logoutput=c:\My Documents\project1.txt, 44 loglevel=information|warning|error, 45 database=c:\My Documents\project1\project1.cdb 46 } 47
Démarrer automatique avec EA	Cette option démarre automatiquement les services ayant l'attribut « status:ON » à l'ouverture du modèle. System Output System Debug AuditLog Broadcast VCLog Job History Script Full load on demand package: 1 EA00-0000-2F00 Auto Start Services Enabled Service Configuration: C:\ea\intelservice\SparxServices.config Service EA is running Les messages enregistrés dans la fenêtre Sortie système ici lorsque le modèle est ouvert indiquent que le service était déjà en cours d'exécution.
Arrêt automatique à la fermeture	Cette option arrête automatiquement l'exécution des services lorsque Enterprise Architect est fermé.

Mise à jour automatique du service Sparx Intel

Lorsque vous exécutez la commande Build pour un script Analyzer, une tâche est ajoutée à la file d'attente des tâches.

Si la case « Mettre à jour Codeminer à la fin » du script Build est cochée dans l' Éditeur de Script Analyseur , une tâche supplémentaire est ajoutée au travail pour mettre à jour chacune des bases de données Codeminer répertoriées dans le script.



Les bibliothèques peuvent être vues dans Code Miner | Section Bibliothèques du script.

Comment s'exécute la tâche

La tâche de mise à jour Code Miner exécute le programme SSCodeMiner.exe avec deux arguments.

Le premier argument spécifie la base de données sur laquelle effectuer la construction incrémentielle et a la forme suivante :

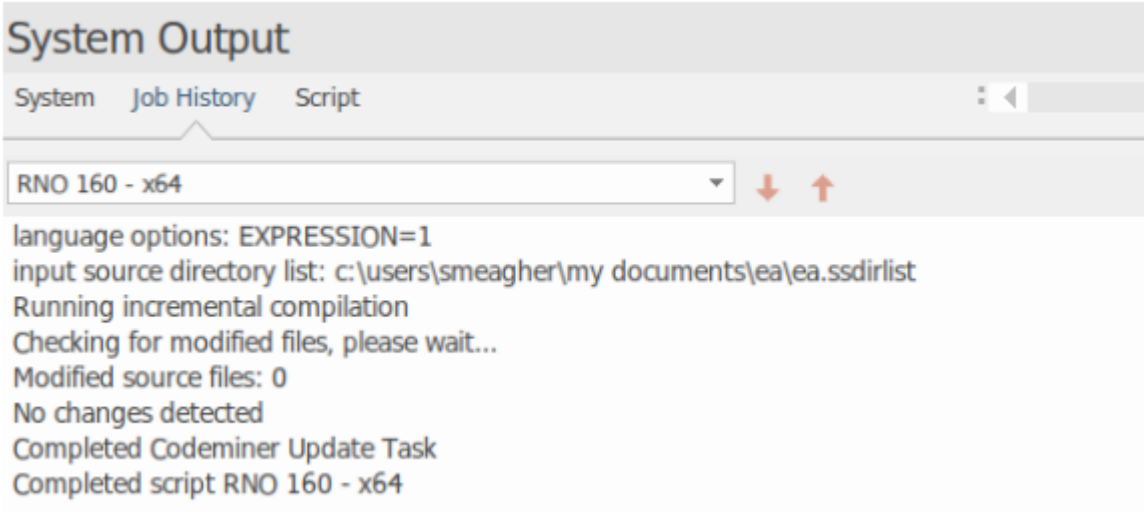
update="c:\chemin\ea.cdb"

Le deuxième argument est facultatif et spécifie un fichier de grammaire de macro auxiliaire à utiliser lors de la compilation de la base de données ; il a cette forme :

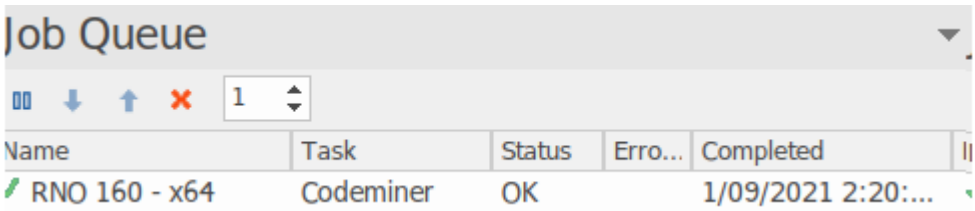
macros="c:\ea\ea160\config\CodeMiner\SparxProjectMacros.nbnf"

Résultat du travail

Pendant l'exécution de la tâche de mise à jour Code Miner , la sortie du processus de mise à jour SSCodeMiner.exe capturé est envoyée à l'onglet « Historique des tâches » de la fenêtre Sortie système, sous la même forme que celle affichée lors de l'exécution d'une mise à jour manuelle d'une base de données Code Miner dans Enterprise Architect . Dans cette illustration, nous pouvons voir que l'analyseur Script RNO 160 -x64 s'est terminé avec succès.



La fenêtre File d'attente des travaux indique que le travail est terminé. La dernière tâche à exécuter était la mise à jour Code Miner .



L'onglet « Historique des tâches » indiquait qu'aucun fichier de code source n'avait été modifié. Si des modifications modifiées du code source sont détectées (c'est-à-dire que le service Code Miner a détecté une nouvelle version de ea.cdb et l'a automatiquement mise à jour), ces informations s'affichent :

```
C:\RNO160\SparxIntelService.exe
Running service EA
Auto update enabled, interval is 30 seconds
Loaded database c:\users\smeagher\My Documents\ea\ea.cdb
Listening on port 9910
Sparx Intel Service Version 2.0.0.0 started 2021-08-27 10:15AM [pid:367]
Created log file c:\ea\logs\eaintel.txt
Logging levels: information,warning,error
Sparx Intel Service Version 2.0.0.0 ready 2021-08-27 10:15AM
Enabling SO_REUSEADDR
SO_REUSEADDR enabled
27/08/2021 10:26:06 auto update of c:\users\smeagher\My Documents\ea\ea.cdb
27/08/2021 11:26:36 auto update of c:\users\smeagher\My Documents\ea\ea.cdb
client connected localhost (127.0.0.1) at 2021-08-27 11:46AM

client connected localhost (127.0.0.1) at 2021-08-27 11:46AM
```

Configuration des services

Programme de services

Le nom du programme de service est SparxintelService.exe.

Fichier de configuration

Le service est configuré par le fichier SparxIntelService.config.

Le fichier doit se trouver dans le même répertoire que le programme de service.

Le fichier contient un certain nombre de directives et répertorie également les bases de données Code Miner à servir.

Le fichier est lu une fois au démarrage du service.

Directives	Description
port	Le numéro de port sur lequel le service écoutera.
permettre	Nomme un domaine ou une adresse IP dont l'accès est autorisé : 198.* ou 127.0.0.1
réseau	Les valeurs peuvent être « publiques », « réseau » ou « privées ». • Utilisez "privé" lorsque les directives d'autorisation spécifient une ou plusieurs adresses IP uniques • Utilisez « réseau » lorsque les directives d'autorisation spécifient un domaine générique : 198* • Utilisez "public" pour autoriser tous les clients
base de données	Nomme le chemin d'accès physique complet d'une base de données Code Miner sur le serveur.

Exécuter le programme de manière autonome

Depuis une console normale, entrez la commande : SparxIntelService -listen

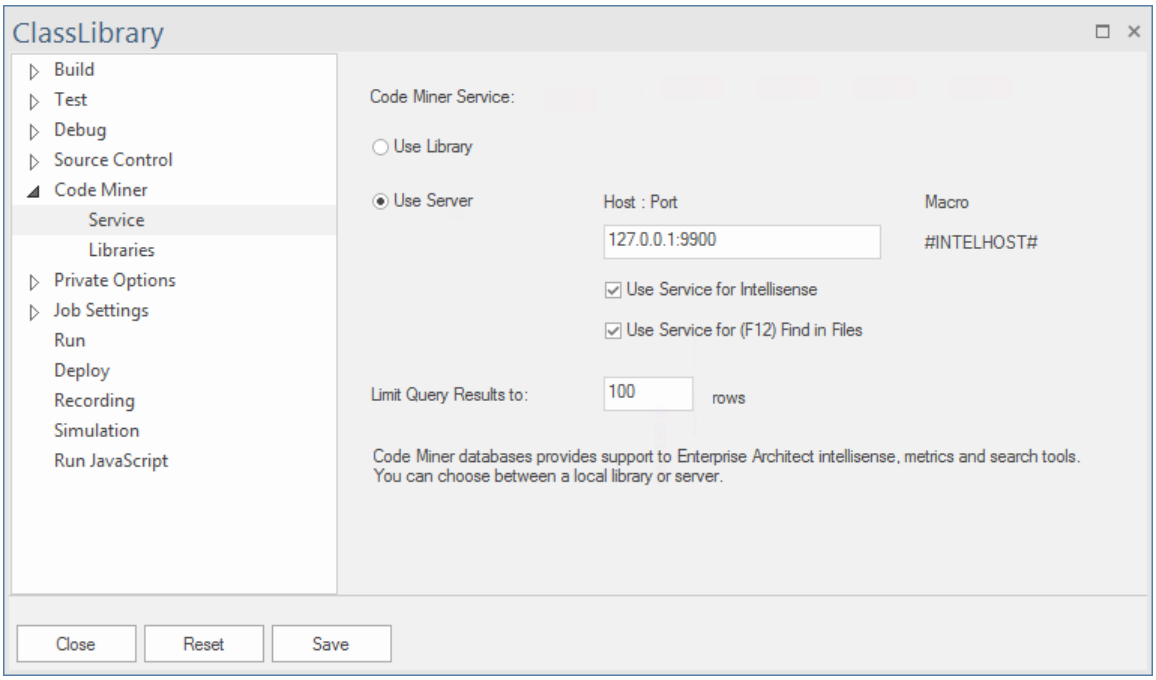
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\sparxsys>cd C:\servers

C:\servers>SparxIntelService -listen
Listening on port 9000
Loaded database e:\codeminer\jdk1.cdb
Loaded database e:\codeminer\bcgsoft10.cdb
Loaded database e:\codeminer\atlmfc.cdb
```

Installation en tant que service Windows

Depuis une console d'administration, entrez la commande : SparxIntelService -install

Configuration client - Configuration Enterprise Architect pour utiliser un service Code Miner

Enterprise Architect utilise des composants appelés Scripts d'Analyseur pour la configuration de nombreux systèmes support . C'est ici que l'emplacement du serveur est spécifié. Cette image montre la page « Code Miner Service » d'un script.



Accéder

Ruban	Développer > Source Code > Analyseur d'Exécution > Edit Scripts d'Analyseur > Double-cliquez sur un Script > Code Miner > Service

Champs de configuration

Utiliser le serveur	Sélectionnez ce bouton radio pour configurer le serveur Code Miner à utiliser.
Port hôte	Type le numéro du port via lequel le service fonctionnera.
Utiliser le service pour Intelli-sense	Cochez la case pour utiliser le service Intel pour la complétion des champs Intelli-sense.
Utiliser le service pour [F12] Rechercher dans les fichiers	Cochez la case si vous souhaitez utiliser le service au lieu de la fenêtre Rechercher dans les fichiers pour exécuter les requêtes de recherche, lorsque vous appuyez sur F12.

Limiter les résultats Query aux lignes	Type le nombre de lignes de résultats de requête à afficher par page.
Sauvegarder	Cliquez sur ce bouton pour enregistrer les détails de configuration que vous avez saisis.