SORBONNE UNIVERSITÉ

Auteurs : Geoffrey Copin
Yassine Ben Jemia
Zakaria Hamidi

31/10/2018

Type : Spécification Technique des Besoins et Exigences

Equipe : Geoffrey Copin, Yassine Ben Jemia, Zakaria Hamidi

Statut : En cours

Destinataire(s) : CEA List

Page(s) 25

Définition d'un éditeur d'argument de sûreté de fonctionnement

Spécification Technique des Besoins et Exigences

Version	Date	Modifications	Rédacteur(s)
1.0	10/10/2018	-	Geoffrey Copin
1.1	25/10/2018	-Cas d'utilisation -Planification -Définir les contraintes - Définition du projet	Geoffrey Copin Yassine Ben Jemia Zakaria Hamidi
1.2	27/10/2018	-Cas d'utilisation	Geoffrey Copin
1.3	30/10/2018	-Tests d'intégration - WBS - Modification du diagramme de PERTT -Diagramme de Package	Geoffrey Copin Yassine Ben Jemia Zakaria Hamidi
1.4	31/10/2018	-Corrections - Ajout de contrainte	Yassine Ben Jemia Geoffrey Copin

SORBONNE UNIVERSITÉ

Table des Matières

SORBONNE UNIVERSITÉ

Définition du projet

Une argumentation de sûreté de fonctionnement est un raisonnement étayé par des preuves visant à montrer que le fonctionnement d'un système est conforme à un ensemble d'exigences. L'usage d'une notation structurée pour exprimer l'argumentation facilite la réalisation d'un projet en permettant de communiquer de façon succincte et non ambigüe. La finalité de ce projet est de produire un éditeur graphique permettant de produire des diagrammes utilisant le langage Goal Structured Notation (GSN) pour représenter des argumentations de sûreté de fonctionnement.

Objectifs

1. Offrir à l'utilisateur la possibilité de démontrer que le système conçu fonctionne comme prévu.
2. Augmenter la confidence dans la réduction des risques et l'amélioration de la fiabilité logicielle.
3. Faciliter le travail de l'utilisateur en offrant une interface graphique et des patterns d'argumentation.

Périmètre

Durant notre projet, nous sommes chargés de mettre en place les fonctions opérationnelles suivantes:

- Un profil UML reprenant les éléments du langage GSN.
- Un éditeur graphique qui permet d'utiliser les stéréotypes définis dans le profil. L'éditeur sera doté d'une palette ainsi qu'une vue propriété.
- Création et intégration d'une librairie de patterns d'argumentation de sûreté réutilisables. L'utilisateur devra être en mesure d'enrichir la librairie en ajoutant de nouveaux patterns.
- Un manuel d'utilisation rédigé en anglais.

Contexte de l'organisation existante

La création de l'éditeur d'argumentation de sûreté de fonctionnement (Assurance Case) entre dans le cadre du projet
FACE qui définit une nouvelle architecture pour l'environnement informatique automobile ainsi que le projet européen "AMASS" qui a pour objectifs de développer des outils pour confirmer la sûreté des systèmes embarquées de point de vue sécurité , disponibilité ,

robustesse et fiabilité . Les deux projets ont pour but d'effectuer des assurances cases qui répondent aux norme afin d'approuver la fiabilité , la disponibilité , la sécurité et la sûreté de leurs systèmes.
Étant partenaire de ces projets , CEA List propose la réalisation d'un plugin permettant de concevoir une démarche d'argumentation basé sur le langage GSN afin de prouver la conformité d'un système à un ensemble d'exigences. Ce plugin sera réalisé avec la plateforme "Papyrus" qui a été développée au sein du CEA List afin de faciliter la spécification , la conception et la réalisation correcte de systèmes complexes.
L'environnement Papyrus est sous licence EPL " Eclipse public licence " . Il sera le berceau de notre projet .

Concepts du langage GSN

L'éditeur se base sur le langage GSN qui définit les concepts suivants :

Éléments de base

- **Goal :** Le *Goal* est l'élément à prouver dans une argumentation. Il peut être *Undevelopped* pour indiquer qu'il ne sera volontairement pas développé.
- **Strategy :** Afin de prouver un *Goal* une *Strategy* doit être adoptée. Elle décrit la relation entre un *Goal* et des sous-*Goal(s)*. Une *Strategy* peut également être *Undevelopped.*
- **Solution** : Une *Solution* représente une référence à un élément de preuve.
- **Context :** Un *Contex*t peut être lié à un stéréotype de type *Goal* ou de type *Strategy* afin d'apporter des informations contextuelles.
- **Assumption :** Une *Assumption* représente une hypothèse admise.
- **Justification :** Une *Justification* permet à l'auteur de l'argumentation case de justifier une affirmation ou une stratégie d'argumentation afin d'expliquer pourquoi il la considère comme acceptable .
- **SupportedBy :** Les relations de type *SupportedBy* permettent de définir un relation d'inférence entre plusieurs éléments de l'assurance case :
  " goal-to-goal", " goal-to-strategy", " goal-to-solution", "strategy-to-goal".
- **InContextOf :** Les relations de type *InContextOf* indiquent l'existence d'une relation contextuelle entre plusieurs éléments de l'argument :
  "goal-to-context", "goal-to-assumption", "goal-to-justification", "strategy-to-context", "strategy-to-assumption" et "strategy-to-justification".

Abstraction structurelle :

- **Mutliplicity:** Représentation de relations n-aires entre des éléments. Peut être appliquée à n'importe quel type de relation.

- **Optionnality:** Permet d'indiquer qu'une relation n'est pas obligatoire.
- **Option:** Permet de représenter le choix d'un sous-ensemble de relations parmi un ensemble de relations unies entre elles par un élément de type *Option*.
- **UninstanciatedElement:** N'importe quel élément peut être marqué comme étant *Uninstanciated* pour indiquer qu'il s'agit d'un élément générique faisant partie d'un pattern. Après l'instanciation du pattern, il faudra instancier cet élément en le remplaçant par un élément concret.

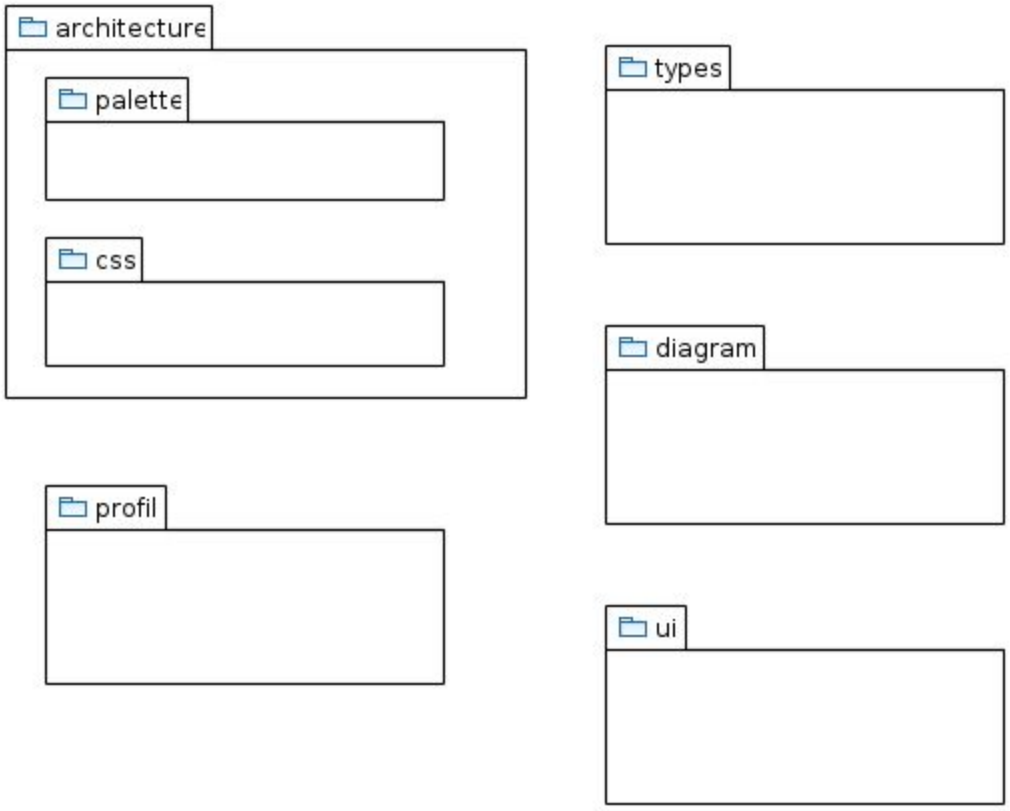
Abstraction modulaire:

- **Away Goal:** Un *Away Goal* référence un *Goal* défini dans un autre module.
- **ModuleReference:** Une *ModuleReference* fait référence à un module définit dans un autre diagramme.
- **ContractModuleReference:** Une *ContractModuleReference* fait référence à un *ContractModule* définit dans un autre diagramme.
- **AwaySolution:** Une *AwaySolution* référence des éléments de preuves présents dans un autre module.
- **AwayContext:** Un *AwayContext* référence des éléments contextuels définis dans un autre module.
- **PublicDecorator:** Il est possible d'ajouter un *PublicDecorator* à un élément d'un module pour le rendre accessible depuis d'autres modules.
- **ToBeSupportedByContract:** Le décorateur *ToBeSupportedByContract* peut être ajouté à un *Goal* pour indiquer qu'il s'appuie sur un argument défini dans un *ContractModule* encore inconnu.
- **Module et ContractModule:** Ces deux éléments représentent des modules.

Définition du système à réaliser

Architecture fonctionnelle

Diagramme de package:



Le diagramme ci-dessus comporte 5 packages et 2 sous-packages définis de la façon suivante:

- **Architecture:** il contiendra deux autres packages ainsi que le fichier ".architecture"
- **Palette:** ce package contiendra les différents éléments de la palette de notre éditeur
- **CSS**: tous les fichiers ".css" de l'éditeur graphique
- **Types:** les types des différents éléments ainsi que leurs correspondances avec la palette et cela sous forme de fichier XML
- **Profil:** cela contiendra les stéréotypes du modèle , GSN , le code généré à partir du profil et les contraintes entre les différents éléments .
- **Diagram:** il contiendra la définition du diagramme du profil ou bien une extension du diagramme de classe .
- **UI:** il contiendra le menu ainsi que les propriétés

SORBONNE UNIVERSITÉ

Cas d'utilisation

Un seul acteur -appelé l'utilisateur- interagira avec le programme.

Nom	Créer un nouveau projet
Description	L'utilisateur souhaite créer un nouveau projet depuis la page de démarrage Eclipse.
Acteurs	Utilisateur
Pré-conditions	Aucunes
Scénario nominal	1. L'utilisateur clique sur "New Papyrus Project" 2. Dans la rubrique "Architecture Contexts", l'utilisateur sélectionne l'option "GSN". 3. L'utilisateur saisit le nom du projet et sélectionne le type de diagramme souhaité, puis clique sur "Finish".
Post-conditions	Eclipse crée et ouvre un nouveau projet correspondant au paramètres saisis par l'utilisateur.
Alternatives / Exceptions	E1. À tout moment, l'utilisateur peut fermer l'assistant de création de projet, ce qui met fin au cas d'utilisation.

Titre	Test - A1
Cas d'utilisation	Créer un nouveau projet
Contexte	Le workspace courant ne contient aucun projet ayant le même nom que le projet de test.
Entrées(s)	- Nom du projet: "testProj"
Scénario	1. L'utilisateur clique sur le menu "File", puis "New", puis "New Papyrus Project" 2. Dans la rubrique "Architecture Contexts" l'utilisateur sélectionne l'option GSN. 3. L'utilisateur saisit le nom du projet puis clique sur "Finish".

Résultat attendu	Un nouveau projet a été créé.
Moyen de vérification	Le projet “testProj” doit figurer dans le workspace Eclipse. Les stéréotypes définis dans le profil doivent figurer dans la palette lorsque l’utilisateur édite un diagramme de classe.

Nom	Ajouter un nouvel élément via la palette
Description	L’utilisateur souhaite insérer un nouvel élément dans son diagramme
Acteurs	Utilisateur
Pré-conditions	La vue “Palette” a été ouverte en cliquant sur son icône ou via le menu “Window->Show View”.
Scénario nominal	1. L’utilisateur sélectionne l’élément souhaité dans la palette. 2. L’utilisateur instancie l’élément dans son diagramme par un glisser-déposer.
Post-conditions	L’élément souhaité a été ajouté au diagramme, à l’endroit où l’utilisateur a effectué le glisser-déposer.
Alternatives / Exceptions	E1. À l’étape 2, l’utilisateur appuie sur la touche “Escape” avant la fin du glisser-déposer, ce qui met fin au cas d’utilisation.

Titre	Test - B1
Cas d’utilisation	Ajouter un nouvel élément via la palette
Contexte	L’utilisateur est en train d’éditer un diagramme de classe vide. La vue “Palette” est ouverte
Entrées(s)	-
Scénario	1. L’utilisateur sélectionne l’élément “Goal” dans la palette. 2. L’utilisateur fait un glisser-déposer dans le diagramme.
Résultat attendu	Le diagramme contient un nouveau nœud “Goal”.

Moyen de vérification	Vérification visuelle. Dans la vue “Model Explorer”, un nouveau nœud “Goal” a été créé.

Nom	Ajouter un élément via le menu contextuel (optionnel)
Description	L’utilisateur souhaite insérer un nouvel élément dans son diagramme en utilisant le menu contextuel.
Acteurs	Utilisateur
Pré-conditions	Le plugin est chargé, un modèle est ouvert.
Scénario nominal	1. L’utilisateur affiche la vue “Model Explorer”. 2. L’utilisateur fait un clic droit à la racine de son modèle dans la vue “Model Explorer”. 3. Dans le menu qui apparaît, l’utilisateur sélectionne le sous-menu correspondant au profil. 4. Dans le sous-menu, l’utilisateur sélectionne l’option correspondant à l’élément qu’il souhaite créer.
Post-conditions	L’élément souhaité a été ajouté au diagramme.
Alternatives / Exceptions	E1. À partir de l’étape 2, l’utilisateur peut cliquer en dehors du menu contextuel et de ses sous-menus pour mettre fin au cas d’utilisation.

Titre	Test - B2
Cas d’utilisation	Ajouter un élément via le menu contextuel
Contexte	L’utilisateur est en train d’éditer un diagramme de classe vide. La vue “Model Explorer” est ouverte.
Entrées(s)	-
Scénario	1. L’utilisateur ouvre le menu contextuel en faisant un clic droit sur la racine du modèle dans la vue “Model Explorer”. 2. L’utilisateur sélectionne l’option “Goal”.

Résultat attendu	Le diagramme contient un nouveau nœud “Goal”.
Moyen de vérification	Vérification visuelle. Dans la vue “Model Explorer”, un nouveau nœud “Goal” a été créé.

Nom	Ajouter un élément à un module
Description	L’utilisateur souhaite ajouter un élément à un module.
Acteurs	Utilisateur
Pré-conditions	Le plugin est chargé, l’utilisateur est en train d’éditer un modèle contenant un module. La vue “Palette” est ouverte.
Scénario nominal	1. L’utilisateur sélectionne un élément dans la palette. 2. L’utilisateur instancie l’élément dans son diagramme par un glisser-déposer à l’intérieur du module choisi.
Post-conditions	L’élément a été ajouté au module.
Alternatives / Exceptions	E1. À l’étape 2, l’utilisateur clique sur “Escape” avant la fin du glisser-déposer, ce qui met fin au cas d’utilisation. A1. À l’étape 1 l’utilisateur sélectionne un élément du diagramme (en dehors du module), puis l’ajoute au module par un glisser-déposer.

Titre	Test - B3
Cas d’utilisation	Ajouter un élément à un module.
Contexte	L’utilisateur est en train d’éditer un diagramme de classe contenant un nœud “Module”. La vue “Palette” est ouverte.
Entrées(s)	-
Scénario	1. L’utilisateur sélectionne l’élément “Goal” dans la palette. 2. Fait un glisser-déposer à l’intérieur de la vue

	représentant le “Module”.
Résultat attendu	Le nœud “Goal” a été ajouté au “Module”.
Moyen de vérification	Vérification visuelle. Le nœud “Goal” doit apparaître en tant que fils du “Module” dans la vue “Model Explorer”.

Nom	Éditer les propriétés d’un élément
Description	L’utilisateur souhaite éditer les propriétés d’un des éléments du diagramme.
Acteurs	Utilisateur
Pré-conditions	Le plugin est chargé, l’utilisateur est en train d’éditer un modèle contenant au moins un élément.
Scénario nominal	1. L’utilisateur sélectionne l’élément dont il veut éditer les propriétés. 2. L’utilisateur affiche la vue propriétés via un menu contextuel, ou en cliquant sur l’icône correspondante. 3. L’utilisateur modifie les champs correspondant au différentes propriétés de l’élément.
Post-conditions	Les propriétés de l’élément ont été modifiées.
Alternatives / Exceptions	E1. À tout moment, l’utilisateur peut désélectionner l’élément ou fermer la vue propriétés pour mettre fin au cas d’utilisation.

Titre	Test - C1
Cas d’utilisation	Éditer les propriétés d’un élément
Contexte	L’utilisateur est en train d’éditer un diagramme de classe contenant un nœud “Goal” (cf. Test - B1).
Entrées(s)	- statement: “test Statement”
Scénario	1. L’utilisateur sélectionne le nœud “Goal”.

	2. L'utilisateur ouvre la vue "Propriétés". 3. L'utilisateur saisit le statement dans le champ "statement".
Résultat attendu	La propriété "statement" du nœud "Goal" a été modifiée.
Moyen de vérification	La valeur modifiée sera visible si l'utilisateur ouvre à nouveau la vue "Propriétés".

Nom	Ajouter une relation entre deux éléments
Description	L'utilisateur souhaite ajouter une relation entre deux éléments d'un modèle
Acteurs	Utilisateur
Pré-conditions	Le plugin est chargé, l'utilisateur est en train d'éditer un modèle contenant au moins deux éléments. La vue "Palette" est ouverte.
Scénario nominal	1. Dans la palette, l'utilisateur clique sur le type de relation souhaité. 2. L'utilisateur clique sur la source de la relation. 3. L'utilisateur clique sur la cible de la relation.
Post-conditions	Une relation du type choisi a été ajoutée entre les deux éléments sélectionnées.
Alternatives / Exceptions	E1. À partir de l'étape 2, l'utilisateur peut appuyer sur "Escape", sélectionner un autre élément dans la palette ou cliquer sur la grille pour mettre fin au cas d'utilisation. E2. À l'étape 2, si l'utilisateur sélectionne une source invalide, l'opération est annulée. E3. À l'étape 3, si l'utilisateur sélectionne une destination invalide l'opération est annulée.

Titre	Test - D1
Cas d'utilisation	Ajouter une relation entre deux éléments

Contexte	L'utilisateur est en train d'éditer un diagramme de classe contenant deux nœuds "Goal". La vue "Palette" est ouverte.
Entrées(s)	-
Scénario	1. L'utilisateur sélectionne une relation de type "SupportedBy" dans la palette. 2. L'utilisateur clique sur le premier "Goal". 3. L'utilisateur clique sur le second "Goal".
Résultat attendu	Une relation de type "SupportedBy" a été ajoutée entre les deux nœuds "Goal". Le premier nœud en est la source et le second la destination.
Moyen de vérification	Vérification visuelle. La relation doit apparaître dans la vue "Model Explorer".

Titre	Test d'erreur - DE1
Cas d'utilisation	Ajouter une relation entre deux éléments
Contexte	L'utilisateur est en train d'éditer un diagramme de classe contenant un nœud "Context" et un nœud "Goal". La vue "Palette" est ouverte.
Entrées(s)	-
Scénario	1. L'utilisateur sélectionne une relation de type "SupportedBy" dans la palette. 2. L'utilisateur clique sur le nœud "Context".
Résultat attendu	La relation n'a pas été ajoutée.
Moyen de vérification	La vue représentant la relation doit disparaître lorsque l'utilisateur clique sur le "Context".

Titre	Test d'erreur - DE2
Cas d'utilisation	Ajouter une relation entre deux éléments
Contexte	L'utilisateur est en train d'éditer un diagramme de classe

	contenant un nœud “Context” et un nœud “Goal”. La vue “Palette” est ouverte.
Entrées(s)	-
Scénario	1. L’utilisateur sélectionne une relation de type “SupportedBy” dans la palette. 2. L’utilisateur clique sur le “Goal”. 3. L’utilisateur clique sur le “Context”.
Résultat attendu	La relation n’a pas été ajoutée.
Moyen de vérification	La vue représentant la relation doit disparaître lorsque l’utilisateur clique sur le “Context”.

Nom	Modifier la source ou la destination d’une relation
Description	L’utilisateur souhaite modifier la source ou la destination d’un élément.
Acteurs	Utilisateur
Pré-conditions	Le plugin est chargé, l’utilisateur est en train d’éditer un modèle contenant au moins deux éléments unis par une relation.
Scénario nominal	1. L’utilisateur clique sur l’une des extrémités de la vue graphique représentant la relation. 2. L’utilisateur déplace cette extrémité jusqu’à un autre élément du modèle.
Post-conditions	La source (ou la destination) de la relation a été modifiée.
Alternatives / Exceptions	E1. À l’étape 2, si l’utilisateur sélectionne un élément invalide ou déplace le pointeur jusqu’à une zone vide de l’éditeur la relation et supprimée et le cas d’utilisation prend fin. E2. À l’étape 2, si l’utilisateur sélectionne une source ou une destination invalide, la relation disparaît.

Titre	Test - D2

Cas d'utilisation	Modifier la source ou la destination d'une relation
Contexte	L'utilisateur est en train d'éditer un diagramme de classe contenant deux nœuds "Goal" reliés entre eux par une relation de type "SupportedBy". (cf. Test - D1) Le diagramme contient également un nœud de type "Context".
Entrées(s)	-
Scénario	1. L'utilisateur sélectionne la source de la relation. 2. L'utilisateur la déplace vers le noeud "Context" par un glisser-déposer.
Résultat attendu	La relation est supprimée.
Moyen de vérification	Vérification visuelle. La relation ne doit plus apparaître dans la vue "Model Explorer".

Nom	Instancier un pattern
Description	L'utilisateur souhaite instancier l'un des patterns de la librairie.
Acteurs	Utilisateur
Pré-conditions	Le plugin est chargé, l'utilisateur est en train d'éditer un modèle et la vue "Model Explorer" est ouverte.
Scénario nominal	1. L'utilisateur fait un clic droit sur la racine de son modèle dans la vue "Model Explorer". 2. Dans le menu contextuel qui apparaît, l'utilisateur sélectionne le sous menu correspondant au profil. 3. Dans le sous-menu qui apparaît, l'utilisateur sélectionne l'option "Library". 4. Dans le sous-menu qui apparaît, l'utilisateur sélectionne le nom du pattern à instancier.
Post-conditions	Une copie du pattern sélectionné a été instanciée dans modèle.
Alternatives / Exceptions	E1. À partir de l'étape 2, l'utilisateur peut cliquer en dehors du menu contextuel et de ses sous-menus pour mettre fin au cas d'utilisation.

Titre	Test - E1
Cas d'utilisation	Instancier un pattern
Contexte	L'utilisateur est en train d'éditer un diagramme de classe. La vue "Model Explorer" est ouverte.
Entrées(s)	-
Scénario	1. L'utilisateur fait un clique-droit sur la racine du modèle dans la vue "Model Explorer". 2. Dans le menu contextuel qui apparaît, l'utilisateur sélectionne le sous menu correspondant au profil. 3. Dans le sous-menu qui apparaît, l'utilisateur sélectionne l'option "Library". 4. Dans le sous-menu qui apparaît, l'utilisateur sélectionne le pattern "Risk management".
Résultat attendu	Une copie du pattern a été instanciée dans le diagramme.
Moyen de vérification	Vérification visuelle. Les éléments constituant le pattern doivent être visibles dans la vue "Model Explorer".

Nom	Ajouter un nouveau pattern
Description	L'utilisateur souhaite sauvegarder un modèle en tant que pattern.
Acteurs	Utilisateur
Pré-conditions	Le plugin est chargé, l'utilisateur est en train d'éditer un modèle et la vue "Model Explorer" est ouverte.
Scénario nominal	1. L'utilisateur sélectionne les éléments du modèle qui feront partie du pattern à ajouter. 2. L'utilisateur fait un clic droit. 3. Dans le menu contextuel qui apparaît, l'utilisateur clique sur "Save as Pattern".

	4. Dans la boîte de dialogue qui apparaît l'utilisateur saisit le nom du pattern et clique sur "OK".
Post-conditions	Un pattern composé des éléments sélectionnées a été ajouté à la librairie de patterns.
Alternatives / Exceptions	E1. À l'étape 3, l'utilisateur peut cliquer en dehors du menu contextuel pour mettre fin au cas d'utilisation. E2. À l'étape 4, l'utilisateur peut cliquer sur "Cancel", ce qui met fin au cas d'utilisation.

Titre	Test - E2
Cas d'utilisation	Ajouter un nouveau pattern
Contexte	L'utilisateur est en train d'éditer un diagramme de classe non vide.
Entrées(s)	- nom: "testPattern"
Scénario	1. L'utilisateur sélectionne tous les éléments du diagramme. 2. L'utilisateur fait un clic droit et sélectionne l'option "Save as Pattern". 3. L'utilisateur saisit le nom dans la boîte de dialogue et clique sur "OK".
Résultat attendu	Le pattern a été créé.
Moyen de vérification	Le menu contextuel permettant d'instancier un nouveau pattern (cf. cas d'utilisation "Instancier un pattern") contient une nouvelle entrée nommée "testPattern" et correspondant au modèle sauvegardé.

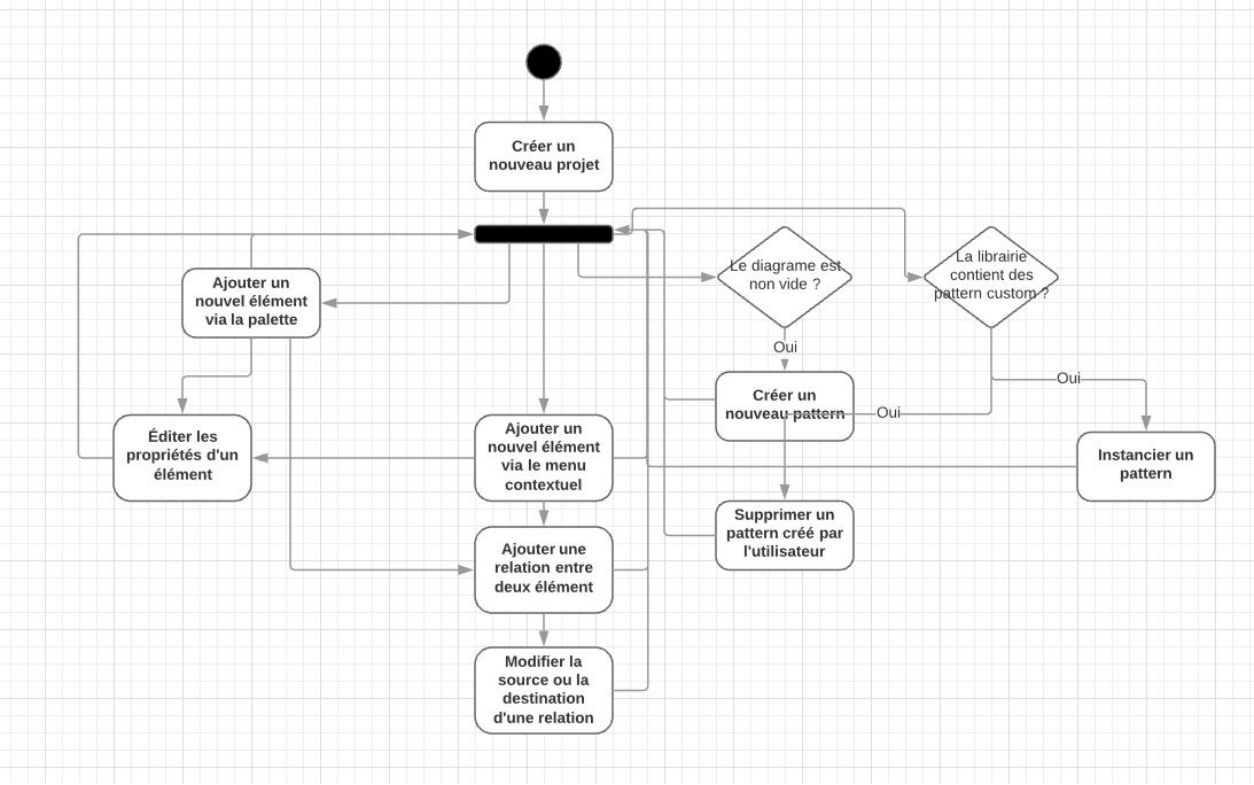
Nom	Supprimer un pattern créé par l'utilisateur
Description	L'utilisateur souhaite supprimer l'un des patterns qu'il a ajouté à la librairie.
Acteurs	Utilisateur

Pré-conditions	Le plugin est chargé, l'utilisateur est en train d'éditer un modèle. La librairie de patterns contient au moins un pattern créé par l'utilisateur.
Scénario nominal	1. L'utilisateur fait un clic droit sur la racine de son modèle dans la vue "Model Explorer". 2. Dans le menu contextuel qui apparaît, l'utilisateur sélectionne le sous menu correspondant au profil. 3. Dans le sous-menu qui apparaît, l'utilisateur sélectionne l'option "Library". 4. Dans le sous-menu qui apparaît, l'utilisateur sélectionne l'option "Delete pattern". 5. Dans la boîte de dialogue qui apparaît, l'utilisateur sélectionne le(s) pattern(s) à supprimer puis clique sur "OK".
Post-conditions	Un pattern composé des éléments sélectionnées a été ajouté à la librairie de patterns.
Alternatives / Exceptions	E1. À l'étape 3, l'utilisateur peut cliquer en dehors du menu contextuel pour mettre fin au cas d'utilisation. E2. À l'étape 4, l'utilisateur peut cliquer sur "Cancel", ce qui met fin au cas d'utilisation.

Titre	Test - E3
Cas d'utilisation	Supprimer un pattern créé par l'utilisateur
Contexte	L'utilisateur est en train d'éditer un diagramme de classe. La librairie de patterns contient le pattern "testPattern". (cf. Test - E2)
Entrées(s)	-
Scénario	1. L'utilisateur fait un clic droit sur la racine de son modèle dans la vue "Model Explorer". 2. Dans le menu contextuel qui apparaît, l'utilisateur sélectionne le sous menu correspondant au profil. 3. Dans le sous-menu qui apparaît, l'utilisateur sélectionne l'option "Library". 4. Dans le sous-menu qui apparaît, l'utilisateur sélectionne l'option "Delete pattern".

	5. Dans la boîte de dialogue qui apparaît, l'utilisateur sélectionne "testPattern" puis clique sur "OK".
Résultat attendu	Le pattern "testPattern" a été supprimé.
Moyen de vérification	Le menu contextuel permettant d'instancier un nouveau pattern (cf. cas d'utilisation "Instancier un pattern") ne contient plus l'entrée "testPattern".

Interactions entre les cas d'utilisation



Contraintes

Contraintes fonctionnelles :

- Une correspondance entre les concepts GSN et les éléments de la norme SACM 2 doit être établie.
- A l'issue du projet, le plugin devra intégrer le dépôt Papyrus.
- Le plugin devra permettre aux utilisateurs de modéliser des assurances cases en utilisant le langage UML .
- Le plugin devra comporter des vues graphiques.
- La définition des patterns dépend des normes GSN et des contraintes de la norme SACM 2

Contraintes techniques :

- La conception et l'implémentation du projet se fait dans l'environnement "Papyrus"
- Le plugin doit être compatible avec la dernière version de l'IDE Eclipse : "Eclipse photon"
- Le projet doit être implémenté sous JAVA SE 8
- Le plugin se base sur un profil UML
- La mise en place des vues personnalisés des stéréotypes seront basées sur le langage CSS

Analyse des risques :

Risques liés aux contraintes fonctionnelles :

Difficulté de compréhension de la norme SACM 2	
Description	SACM 2 est une spécification qui définit un métamodèle pour la représentation des cas d'assurances structurés.
Causes	- Complexité de la norme. - Manque de familiarité des membres de l'équipe avec la lecture et l'analyse des normes.
Conséquences	- Perte de temps à la compréhension des notions cités dans la norme - Mise en place de stéréotypes qui ne répondent pas aux règles décrites dans la norme SACM 2

Moyens de prévention	- Demander l'avis du client sur les moyens de pallier à ce risque - Validation du profil par notre client.
Gravité	Moyenne : le risque peut entraîner des corrections sur le projet afin qu'il soit conforme ce qui peut engendrer des retards

Risques liés aux contraintes techniques :

Appréhension de l'environnement "Papyrus"	
Description	Papyrus est un environnement intégré pour l'édition de modèles UML.
Causes	- Les membre de l'équipe n'ont jamais utilisé Papyrus.
Conséquences	- La prise en main de certaines fonctionnalitées peut engendrer des pertes de temps.
Moyens de prévention	- Demander l'avis du client. - Consulter des tutoriels et lire la documentation fournie par le client.
Gravité	Moyenne : le risque peut entraîner des retards dans la livraison du projet.

Définition des éléments graphiques	
Description	L'éditeur doit comporter des vues graphiques redimensionnables.
Causes	- Le comportement par défaut des vues personnalisées n'est pas toujours correct.
Conséquences	- Les vues sont susceptibles de ne pas réagir correctement au redimensionnement du layout des modèles et des éléments du modèle.
Moyens de prévention	- Définir les vues en utilisant le langage CSS et des images vectorielles. - Tester avec attention le comportement des vues lors d'un redimensionnement.
Gravité	moyenne : Le risque peut entraîner des corrections sur le projet afin qu'il soit conforme ce qui peut engendrer des retards

Planification

WBS

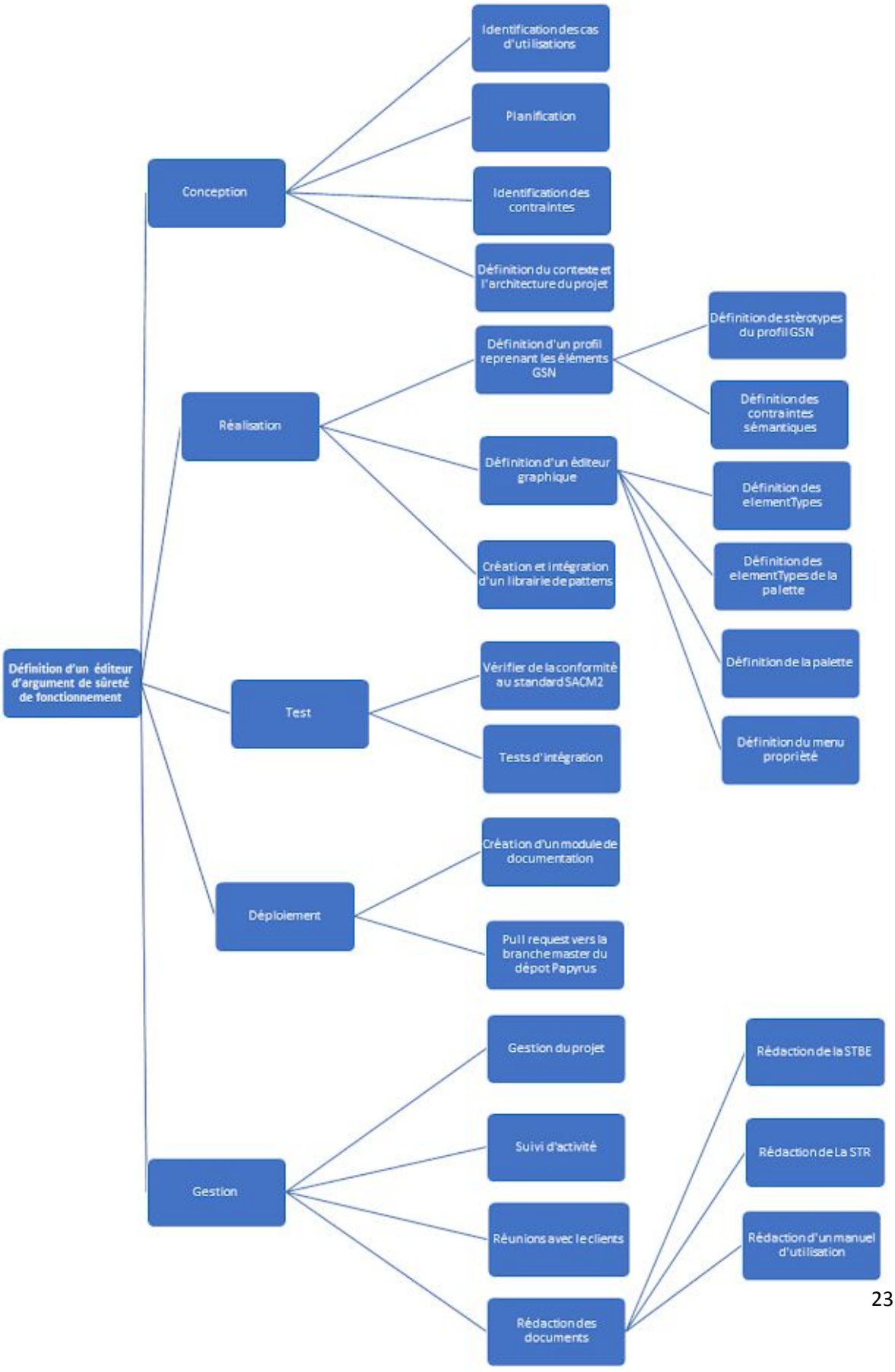

Définition d'un éditeur d'argument de sûreté de fonctionnement
Conception
Identification des cas d'utilisations
Planification
Identification des contraintes
Définition du contexte et l'architecture du projet
Réalisation
Définition d'un profil reprenant les éléments GSN
Définition de stéréotypes du profil GSN
Définition des contraintes sémantiques
Définition d'un éditeur graphique
Définition des elementTypes
Définition des elementTypes de la palette
Définition de la palette
Définition du menu propriété
Création et intégration d'un librairie de patterns
Test
Vérifier de la conformité au standard SACM2
Tests d'intégration
Déploiement
Création d'un module de documentation
Pull request vers la branche master du dépot Papyrus
Gestion
Gestion du projet
Suivi d'activité
Réunions avec le clients
Rédaction des documents
Rédaction de la STBE
Rédaction de La STR
Rédaction d'un manuel d'utilisation

Diagramme de PERT :

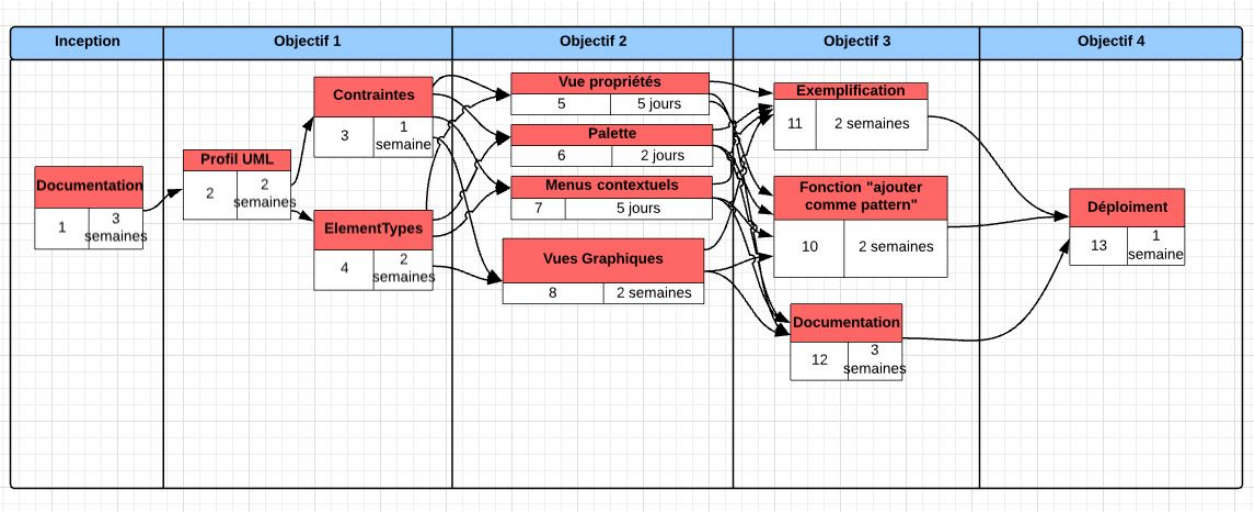
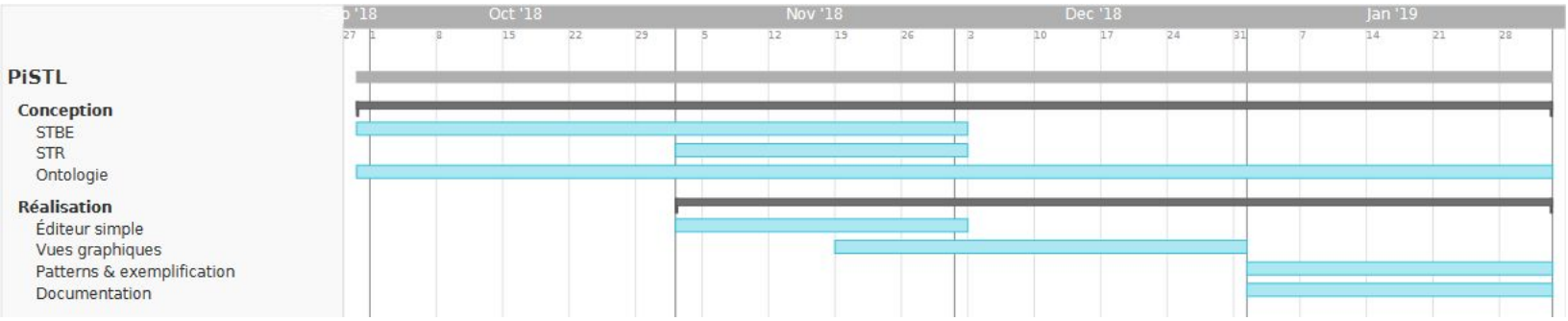


Diagramme de Gantt :



Références :

GSN Community Standard. Version 2. January 2018 :
https://scsc.uk/scsc-141B

Structured Assurance Case Metamodel (SACM) Version 1.1 :
https://www.omg.org/spec/SACM/1.1/About-SACM/

Projet AMASS :
https://www.amass-ecsel.eu/

Papyrus :
https://en.wikipedia.org/wiki/Papyrus_(software)