

Architecture logiciel VTT

Le 22/03/2024

Objet du document

Architecture logicielle

Niveau de confidentialité du document

0 – Public	1 – Interne	2 – Restreint	3 – Confidentiel
	X		

Fiche de révision

Version	Date	Auteur	Nature
V0.4	18/03/2024	MJB, TCA, LSA, MAC	Réalisation d'un document d'architecture du projet VTT

L'incrémement de la version s'effectue :

- en version (0.x) pour les versions de travail, non diffusées à l'extérieur,
- en version majeure (1.0) pour les modifications fonctionnelles,
- en version mineure (x.1) pour les modifications de forme.

PRESENTATION VTT SCRIPT.....	3
INTRODUCTION.....	3
OBJECTIFS.....	3
ARCHITECTURE	4
ARCHITECTURE GLOBALE	4
COMPOSANTS PRINCIPAUX	4
PRISE DE DECISION ARCHITECTURALE.....	4
CONCLUSION	4

DEPENDANCE SU PROJET	5
DEPENDANCES DU PROJET VTTSCRIPT :	5
ANNEXES.....	6
LE PROJET « VTT.SCRIPT » DEPEND DE CES PROJETS :	6
SCHEMA DES DEPENDANCES DES CLASSES.....	8
TABLEAU ORDRE D'APPEL, OPÉRATIONS ET CHARGEMENTS DES DÉPENDANCES	9
VUE ARCHITECTURALE	11
6.1 VUE LOGIQUE	11
VUE ARCHITECTURALE	13
6.3 VUE COMPORTEMENTALE/PROCESSUS	13

Présentation VTT Script

INTRODUCTION

Le projet VTT.Script est une application complexe qui repose sur une architecture modulaire. Ce qui signifie que chaque module est plus ou moins autonome, il peut fonctionner indépendamment des autres composants. Ce dossier d'architecture logicielle vise à expliquer en détail la structure et le fonctionnement de ce projet, mettant en lumière ses dépendances, son ordre de démarrage et ses interactions entre les différentes parties.

OBJECTIFS

Décrire l'architecture modulaire du projet VTT.Script.

Expliquer l'ordre d'appel des dépendances lors du démarrage de l'application.
Analyser les interactions entre les différentes parties du projet pour en comprendre le fonctionnement global.

Architecture

ARCHITECTURE GLOBALE

L'architecture du projet VTT.Script est basée sur un ensemble de parties interdépendantes qui coopèrent pour assurer le bon fonctionnement de l'application. Les parties essentielles comprennent VTT.Common, ServiceBase et Tools, tandis que d'autres parties plus spécifiques telles que VTT.Model, EncryptDecrypt et KeyMouseEngine remplissent des fonctions spécialisées. Ces parties sont activées dans un certain ordre lors du démarrage de l'application, ce qui permet de configurer les paramètres et de démarrer les écrans nécessaires. Le projet reste cohérent grâce à l'utilisation d'éléments de base et d'informations spéciales.

COMPOSANTS PRINCIPAUX

- VTT.Common
- ServiceBase
- Tools
- VTT.Model
- EncryptDecrypt
- KeyMouseEngine
- VTT.DAL
- VTT.UI.Capture
- WindowsInput

PRISE DE DECISION ARCHITECTURALE

Utilisation d'une architecture modulaire pour permettre une évolutivité et une maintenance aisées du projet.

Séparation claire des responsabilités entre les différentes parties du projet pour garantir sa cohérence et sa robustesse.

Conclusion

Ce dossier d'architecture logicielle offre un aperçu approfondi de l'architecture du projet VTT.Script, en mettant en évidence ses dépendances, son ordre de démarrage et

ses interactions. Il fournit un guide précieux pour les développeurs et les architectes logiciels impliqués dans le développement et la maintenance de cette application.

Dépendance su projet

DEPENDANCES DU PROJET VTTSCRIPT :

VTT.Common	ServiceBase	Tools
VTT.Model	EncryptDecrypt	KeyMouseEngine
VTT.DAL	VTT.UI.Capture	WindowsInput

Le projet VTTScript dépend de ces projets pour son bon fonctionnement.

Dans le projet VTT.Scripting, qui est un projet avec des dépendances mais aucun autre projet ne dépend de lui, les dépendances suivantes sont requises :

- ServiceBase
- Tools
- VTT.Common
- VTT.DAL
- EncryptDecrypt
- VTT.Mode
- VTT.UI.Capture
- WindowsInput

Ordre d'appel des dépendances lors du démarrage du projet VTTScript jusqu'à l'affichage de MainWindow :

VTT.Script :

- Lancement du projet avec app.xaml.cs en exécutant dans l'ordre : App(), OnStartup()
- Tools (appelé plusieurs fois après)
- Utilisation de différents outils qui vont servir au projet (LogsManagement, ...)

VTT.Common

- Affichage de l'écran de chargement au démarrage (Splashscreen.cs), puis initiation de l'écran principal (AppLauncherViewModelBase.cs, FichierIni.cs)

ServiceBase

- Paramétrage des données avec des fichiers de configuration comme Main-Config.config, les valeurs JSON d'un projet, et utilisation des dépendances Tools et VTT.Model (GlobalConfiguration.cs, SettingTools.cs, Common.ParamsEditingData.cs)

VTT.Model

- Appel d'objets (_application, _data)

VTT.UI.Capture

- Enregistrement des actions et des changements sur l'écran (InputRecorder, EventMonitor)

EncryptDecrypt

- Conversion de la base 64 à une chaîne de caractères et encodage en UTF8 lors du passage sur la clé SqlConnection et Client-Password (Program.cs)

KeyMouseEngine

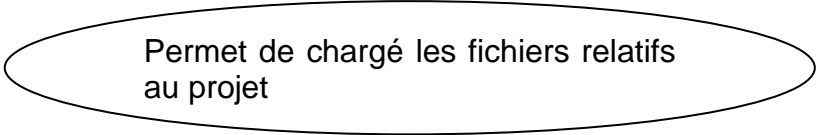
- Méthode InterOpWindow (fonction indéterminée)

Le projet utilise des objets présents dans Models ainsi que des clés de configuration pour son fonctionnement.

Le projet VTT.Script utilise différentes parties qui s'entraident pour fonctionner. On a des parties essentielles comme VTT.Common, ServiceBase, et Tools, et d'autres plus spécifiques comme VTT.Model, EncryptDecrypt et KeyMouseEngine. Quand on lance l'application, ces parties sont activées dans un certain ordre pour faire des choses comme configurer des paramètres et démarrer les écrans. Le projet utilise également des éléments de base et des informations spéciales pour rester cohérent. Cette analyse nous aide à comprendre comment toutes ces parties travaillent ensemble pour que le projet fonctionne correctement.

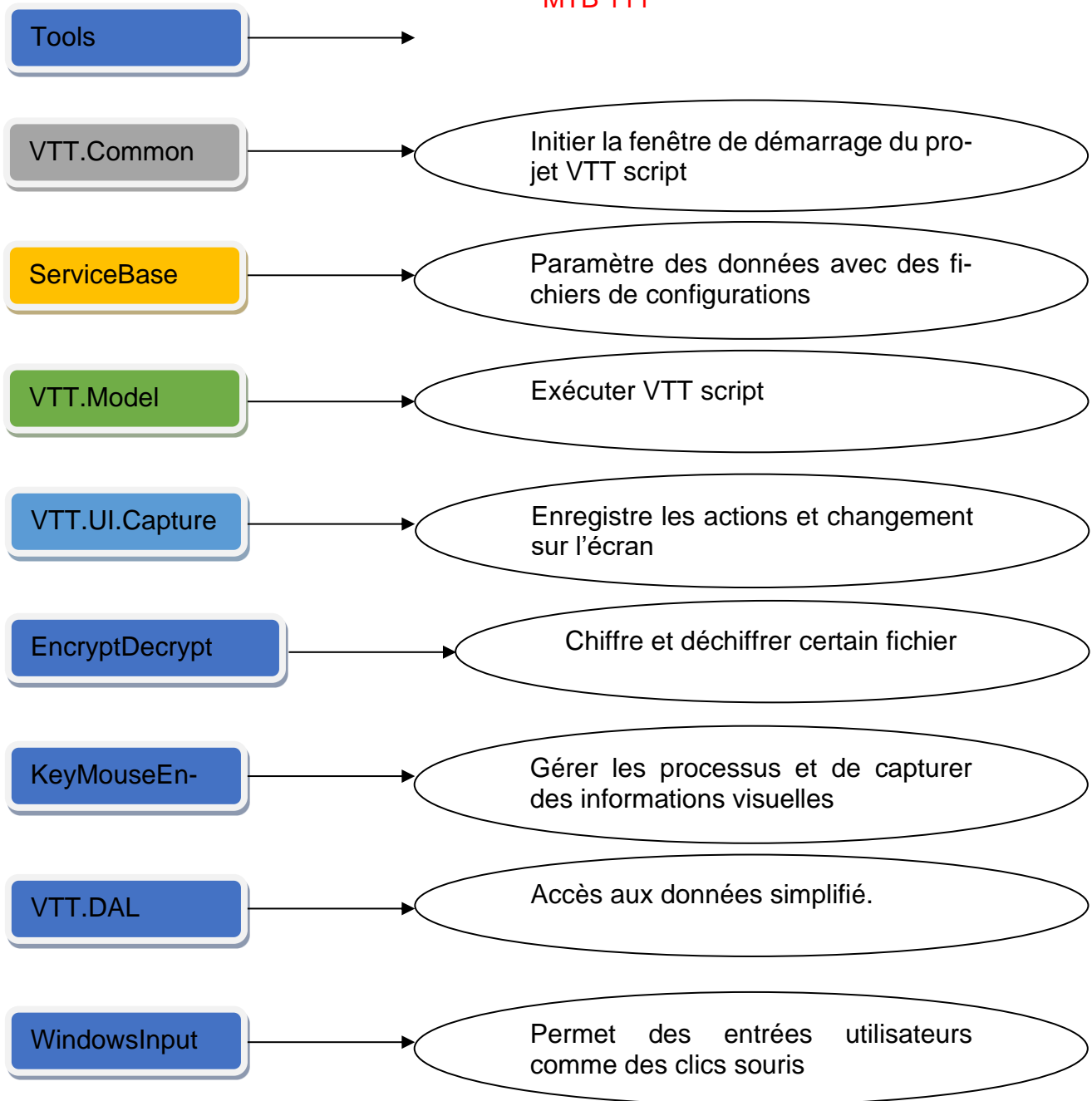
Annexes

LE PROJET « VTT.SCRIPT » DEPEND DE CES PROJETS :

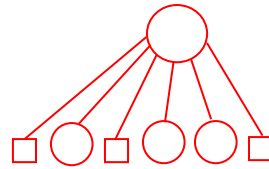


Permet de chargé les fichiers relatifs
au projet

MTB 111



MTB111



SCHEMA DES DEPENDANCES DES CLASSES

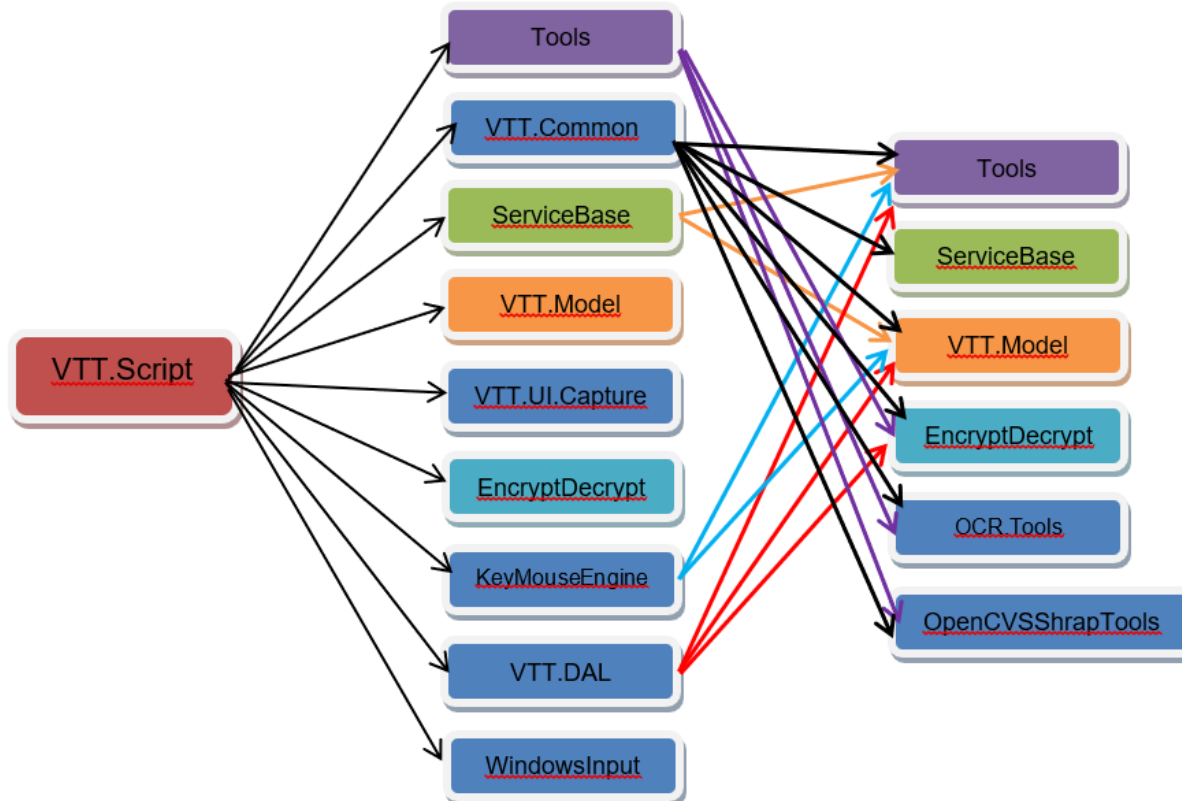


TABLEAU ORDRE D'APPEL, OPÉRATIONS ET CHARGEMENTS DES DÉPENDANCES

Ordre d'appel des dépendances lors du démarrage du projet VTT.Script jusqu'à l'affichage de MainWindow :	Opérations et chargements des dépendances :
VTT.Script	Lancement du projet avec app.xaml.cs en exécutant dans l'ordre : App() , OnStartup(),
Tools (appelé plusieurs fois après)	Utilisation de différents outils qui vont servir au projet (LogsManagement,
VTT Common	Va d'abord afficher l'écran de chargement au démarrage (Splashscreen.cs) puis plus tard va initier l'écran principale (AppLauncherViewModelBase.cs, FichierIni.cs)
ServiceBase	Va paramétrer des données avec des fichiers de configuration comme MainConfig.config, les valeurs Json d'un projet et va lui-même se servir de la dépendance Tools et VTT.Model (GlobalConfiguration.cs, SettingTools.cs, Common.ParamsEditingData.cs)
VTT Model	Appel d'objet (_application, _data)
VTT UI Capture	Le programme y accède depuis VTT-Script.AdminViewModel.cs. Il va lui permettre d'enregistrer les actions et les changements sur l'écran (InputRecorder, EventMonitor)
EncryptDecrypt	Va convertir de la base 64 a un String et aussi encoder en UTF8 quand il passe

MTB 111

	sur la clé SqlConnection et ClientPassword (Program.cs)
KeyMouseEngine	Aucune idée de ce que ça fait car la méthode InterOpWindow est vide

Vue architecturale

6.1 VUE LOGIQUE

La vue logique permet d'organiser les éléments comme des classes ou des objets en catégorie. Elle permet de décomposer le système en abstractions et constitue le cœur de la réutilisation.

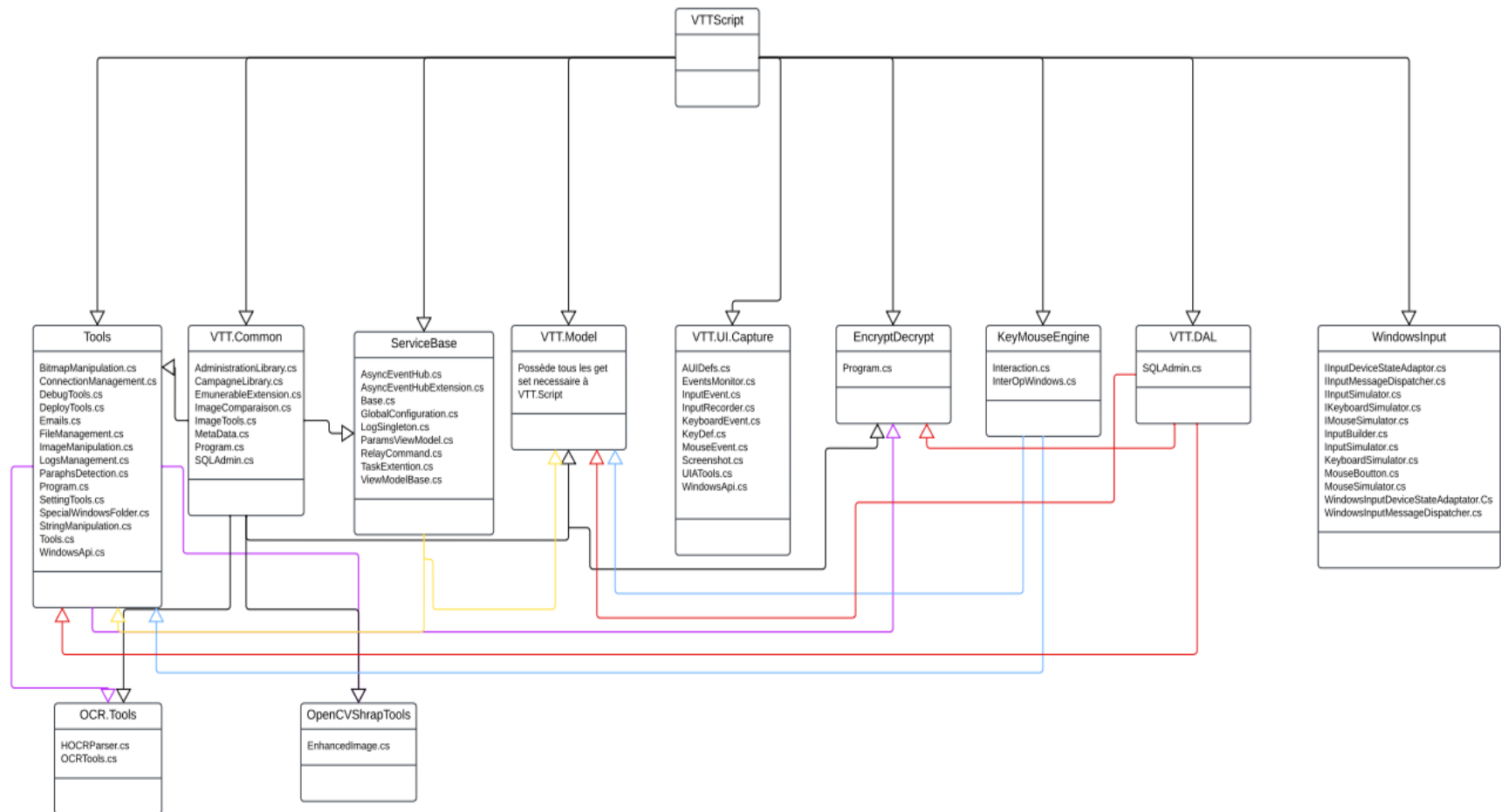
Cette vue est divisée en deux digrammes : le diagramme de classe et le diagramme d'objet.

Dans VTT.Script et même dans toute la solution les objets respectent une nomenclature. Les objets vont venir prendre le nom de leur utilité précédé de leur type quand c'est nécessaire. Par exemple dans un code où il y aura besoin d'un mot de passe il serait déclaré ainsi.

```
public string Password ;
```

Si l'objet est privé il sera écrit avec un « _ » avant.

```
private string _Password ;
```

Diagramme de classe :

Vue architecturale

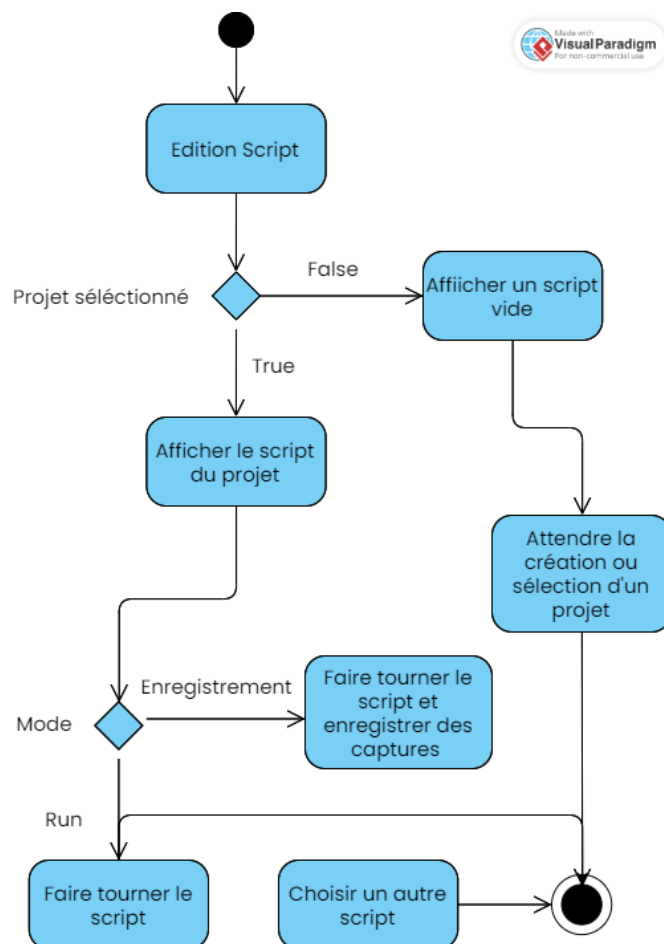
6.2 VUE COMPORTEMENTALE/PROCESSUS

La vue des processus décrit les interactions entre les différents processus, threads (fils d'exécution) ou tâches, elle permet également d'exprimer la synchronisation et l'allocation des objets. Cette vue permet avant tout de vérifier le respect des contraintes de fiabilité, d'efficacité et de performances des systèmes multitâches.

Les diagrammes utilisés dans la vue des processus sont exclusivement dynamiques : il y'a des diagrammes d'activités, de séquence, diagrammes de communication ou d'états-transitions.

Nous allons représenter ici les diagrammes sous forme extrêmement simplifiés.

Voici le diagramme d'activité de l'onglet « Edition Scripts » pour exemple :



Vue architecturale

6.3 VUE DE DEPLOIEMENT

La vue de déploiement visualise la vue topologique d'un système entier. Il représente le déploiement d'un système.

Les vues de déploiements sont généralement utilisées pour visualiser le matériel physique et les logiciels d'un système. On peut ainsi comprendre comment le système sera physiquement déployé sur le matériel.

- Vue de déploiement lorsqu'un client utilise VTT :

