

Rapport de laboratoire

|  |  |
| --- | --- |
| **No de laboratoire** | 4 |
| **Étudiant(s)** | Sébastien Lago  Marc-André Allard  Israël Hallé |
| **Code(s) permanent(s)** | LAGS04128102  HALI17049101  ALLM09029106 |
| **Cours** | LOG430 |
| **Session** | Hiver 2014 |
| **Groupe** | 02 |
| **Professeur** | B. Galarneau |
| **Chargés de laboratoire** |  |
| **Date de remise** | avril 2014 |

Table des matières

[Liste des tableaux 3](#_Toc384823839)

[Liste des figures 3](#_Toc384823840)

[Introduction 4](#_Toc384823841)

[Conception architecturale 5](#_Toc384823842)

[Documents sources 5](#_Toc384823843)

[Règles et standards 5](#_Toc384823844)

[Prérequis 5](#_Toc384823845)

[Mission 6](#_Toc384823846)

[Sous-mission 7](#_Toc384823847)

[État 8](#_Toc384823848)

[Zone d’intérêt 8](#_Toc384823849)

[Bassin 9](#_Toc384823850)

[Système 11](#_Toc384823851)

[Scénarios de qualités 13](#_Toc384823852)

[Présentation de l’architecture 22](#_Toc384823853)

[Vue module 22](#_Toc384823854)

[Vue modèle-vue-contrôleur (C&C) 27](#_Toc384823855)

[Vue déploiement 34](#_Toc384823856)

[Analyse ATAM 38](#_Toc384823857)

[Documents sources 38](#_Toc384823858)

[Règles et standards 38](#_Toc384823859)

[Nature et mission du système 38](#_Toc384823860)

[Présentation de l'architecture et des approches architecturales 38](#_Toc384823861)

[Arbre d’utilité 39](#_Toc384823862)

[ATAM – Analyse 40](#_Toc384823863)

[Analyse de S05 40](#_Toc384823864)

[Analyse de S07 40](#_Toc384823865)

[Analyse de S08 40](#_Toc384823866)

[Analyse de S09 41](#_Toc384823867)

[Analyse de S10 41](#_Toc384823868)

[Analyse de S14 42](#_Toc384823869)

[Risques 42](#_Toc384823870)

[Non-risques 42](#_Toc384823871)

[Point de sensibilité 42](#_Toc384823872)

[Compromis 43](#_Toc384823873)

[Conclusion 44](#_Toc384823874)

# Liste des tableaux

[Tableau 1 Exigences fonctionnelles 6](#_Toc384821835)

[Tableau 2 Exigences fonctionnelles des sous-missions 7](#_Toc384821836)

[Tableau 3 Exigences fonctionnelles des états 8](#_Toc384821837)

[Tableau 4 Exigences fonctionnelles des zones d'intérêt 9](#_Toc384821838)

[Tableau 5 Exigences fonctionnelles des bassins 10](#_Toc384821839)

[Tableau 6 Contraintes 10](#_Toc384821840)

[Tableau 7 Exigences fonctionnelles du système 11](#_Toc384821841)

[Tableau 8 Contraintes du système 12](#_Toc384821842)

[Tableau 9 Responsabilités des éléments de la vue module 23](#_Toc384821843)

[Tableau 10 Correspondance des éléments de la vue avec les cas d'utilisation 24](#_Toc384821844)

[Tableau 11 Correspondance des éléments de la vue avec les scénarios de qualité 25](#_Toc384821845)

# Liste des figures

[Figure 1 vue module 22](#_Toc384821861)

[Figure 2 Vue composant et connecteur 27](#_Toc384821862)

[Figure 3 Diagramme de sequence illustrant le comportant MVC 32](#_Toc384821863)

[Figure 4 Vue déploiement 33](#_Toc384821864)

# Introduction

L’objectif de ce document est de présenter l’architecture du projet du club étudiant S.O.N.I.A, Mission Editor 2.0. Dans un premier temps, on présentera et documentera trois vues architecturales, une du style module, une du style composant et connecteur et une dernière du style affectation. Le but de cette documentation sera d’aider à bien comprendre le rôle des éléments, des relations et des propriétés de ces éléments et relations et comment ils peuvent contribuer à rencontrer les attributs de qualité exigés par le client. Par la suite, on présentera l’analyse de l’architecture à l’aide de la méthode ATAM dans le but de mettre en valeur les conséquences de décisions architecturales.

# Conception architecturale

## Documents sources

 Document de Vision

 Spécification des exigences logicielles (SRS)

## Règles et standards

« Description de ADD »

## Prérequis

« Besoins et attentes des lecteurs (Quelles qualités sont les plus importantes) »

« Vues architecturales à utiliser »

Les attributs de qualités les plus importants pour les utilisateurs du systèmes sont la disponibilité, la performance et la convivialité.

**Disponibilité**

Si le système (Mission Editor) tombe en panne pendant que l’utilisateur est en train de travailler dessus, il doit être capable de revenir à son état précédent le plus rapidement possible en minimisant les pertes de données. La disponibilité est la qualité la plus importante pour le système Mission Editor 2.0.

**Performance**

Les utilisateurs du système étant parfois contraints par le temps et la quantité de mémoire vive étant limitée à seulement 150 mégaoctets, le système Mission Editor 2.0 doit être conçu de manière à utiliser les ressources d’une manière efficace pour exécuter les tâches souhaitées dans des délais raisonnables tout en restant en deçà de la limite imposée par la quantité de mémoire vive.

**Convivialité**

Il est important que le logiciel soit conçu afin que les utilisateurs puissent l’utiliser efficacement. De plus, les nouveaux utilisateurs doivent être capables d’apprendre rapidement à utiliser le logiciel Mission Editor 2.0.

### Mission

**Exigence fonctionnelles**

Tableau 1 Exigences fonctionnelles

|  |  |
| --- | --- |
| **EF07 - Afficher une liste prédéfinie de zones d’intérêts** | Le système doit permettre de récupérer la liste de zones d’intérêt d’un bassin lorsqu’il est en édition de  mission. |
| **EF20 - Créer, afficher, modifier et supprimer des missions** | Le système doit permettre d’effectuer la création, l’affichage, la modification et la suppression de  missions. |
| **EF21 - Ajouter, modifier et supprimer des éléments de missions** | Le système doit permettre d’effectuer l’ajout, la modification et la suppression des éléments de missions.  Ces éléments de mission sont des états, des déclencheurs ou des cibles. |
| **EF22 - Afficher la gestion du temps pour une mission sélectionnée** | Le système doit permettre d’afficher les informations concernant les paramètres de la gestion du temps  d’une mission. |
| **EF31 - Définir le temps global d’une mission** | Le système doit permettre d’entrer une valeur numérique indiquant le temps que le sous-marin autonome  aura pour exécuter une mission complète. |
| **EF37 - Répartir le temps alloué lorsqu’on ajuste le temps global** | Lorsqu’on modifie le temps global d’une mission, le système doit ajuster automatiquement les temps  alloués de ses sous-missions d’après le rapport entre leur poids et la somme des poids des sousmissions  de la mission. |
| **EF46 - Afficher un bouton de validation de mission** | Le système doit posséder un bouton qui permet d’effectuer la validation de mission. |
| **EF47 - Valider la configuration d’une mission** | Le système doit effectuer la validation de configuration d’une mission lorsque l’utilisateur effectue une  sauvegarde ou clique sur le bouton de validation de configuration d’une mission. |
| **EF51 - Afficher le temps courant et le temps global d’une mission** | Le système doit afficher la somme des temps alloués des sous-missions d’une mission (Temps Courant)  et le temps global d’une mission. |
| **EF52 - Indiquer le manque ou l’excès du temps courant d’une mission** | Le système doit indiquer de façon visuelle le manque ou l’excès du temps courant d’une mission par  rapport au temps global. |

**Exigence non-fonctionnelles**

S2, S3, S5, S7, S11

### Sous-mission

**Exigence fonctionnelles**

Tableau 2 Exigences fonctionnelles des sous-missions

|  |  |
| --- | --- |
| **EF04 - Associer une zone d’intérêt à une sous-mission** | Le système doit permettre de spécifier à quelle zone d’intérêt une sous-mission est associée lors de la  création de la sous-mission. |
| **EF23 - Ajouter, afficher, modifier et supprimer des sous-missions** | Le système doit permettre d’effectuer l’ajout, l’affichage, la modification et la suppression de sousmissions  d’une mission. |
| **EF24 - Ajouter et supprimer des transitions entre les états de sous-mission** | Le système doit permettre d’ajouter et de supprimer des transitions entre les sous-missions d’une  mission. Ces transitions dépendent du succès ou de l’échec de l’état. |
| **EF25 - Identifier le premier état d’une sous-mission à l’aide d’une couleur différente** | Le système doit affecter une couleur différente au premier état par rapport aux autres états pour une  sous-mission. |
| **EF26 - Éditer le code de l’état d’une sous-mission** | Le système doit permettre de modifier le code du comportement d’un état d’une sous-mission. |
| **EF27 - Sauvegarder les changements de l’état d’une sous-mission** | Le système doit permettre de sauvegarder l’état d’une sous-mission après avoir effectué des  changements. |
| **EF32 - Définir le temps alloué à une sous-mission** | Le système doit permettre d’entrer le temps alloué au sous-marin autonome pour accomplir une sousmission. |
| **EF33 - Définir le temps minimum d’une sous-mission** | Le système doit permettre d’entrer un temps minimum pour une sous-mission. |
| **EF34 - Définir le temps maximum d’une sous-mission** | Le système doit permettre d’entrer un temps maximum pour une sous-mission. |
| **EF35 - Définir le poids d’une sous-mission** | Le système doit permettre d’attribuer une valeur numérique représentant le poids à une sous-mission. |
| **EF36 - Définir la nécessité d’une sous-mission** | Le système doit présenter un champ permettant d’indiquer l’obligation d’exécution d’une sous-mission. |
| **EF38 - Modifier le temps alloué selon le temps maximum** | Lorsque le temps alloué entré d’une sous-mission est plus grand que son temps maximum, le système  doit changer automatiquement le temps alloué entré pour qu’il soit égal à son temps maximum. |
| **EF41 - Exporter le plan des sous-missions en XML** | Le système doit permettre d’exporter le plan de la mission en format XML. |

**Exigence non-fonctionnelles**

S4, S14

### État

**Exigence fonctionnelles**

Tableau 3 Exigences fonctionnelles des états

|  |  |
| --- | --- |
| **EF28 - Choisir l’état d’une sous-mission à éditer** | Le système doit permettre d’obtenir la liste de tous les états d’une sous-mission et de choisir un état de  cette liste à éditer. |
| **EF29 - Rechercher un état d’une sous-mission** | Le système doit permettre d’effectuer une recherche d’un état avec son étiquette de nom ou son nom de  classe. |
| **EF30 - Obtenir le code source de l’état d’une sous-mission** | Lorsqu’on sélectionne un libellé d’état de mission à éditer, le système doit obtenir le code source de l’état. |

**Exigence non-fonctionnelles**

S13

### Zone d’intérêt

**Exigence fonctionnelles**

Tableau 4 Exigences fonctionnelles des zones d'intérêt

|  |  |
| --- | --- |
| **EF09 - Modifier la forme d’une zone d’intérêt** | Lorsqu’une zone d’intérêt est sélectionnée, le système doit permettre de changer sa forme soit en cercle  ou en rectangle. |
| **EF10 - Modifier la taille d’une zone d’intérêt** | Lorsqu’une zone d’intérêt est sélectionnée, le système doit permettre de changer sa taille. Lorsqu’il s’agit  d’un rectangle, la largeur et la longueur peuvent varier alors que s’il s’agit d’un cercle, son rayon peut  varier |
| **EF11 - Modifier l’angle d’une zone d’intérêt** | Lorsqu’une zone d’intérêt est sélectionnée, le système doit permettre de la faire pivoter. |
| **EF12 - Retirer une zone d’intérêt** | Le système doit permettre de retirer une zone d’intérêt du bassin en cours d’édition. |
| **EF13 - Modifier le point d’entrée et l’orientation d’une zone d’intérêt** | Lorsqu’une zone d’intérêt est sélectionnée, le système doit permettre de paramétrer l’endroit où se situe  le point d’entrée et l’orientation que le sous-marin autonome doit suivre. |
| **EF14 - Ajouter le point d’entrée et l’orientation à l’eau du sous-marin autonome** | Le système doit permettre de situer la position où le sous-marin autonome entrera à l’eau ainsi que son  orientation. |
| **EF16 - Définir une zone pour l’équipe à l’extérieur du bassin** | Le système doit permettre de définir une zone représentant la position à l’extérieur d’un bassin de  l’équipe opérant le sous-marin autonome. |
| **EF18 - Afficher la couleur d’une zone d’intérêt selon son type** | Le système doit afficher des couleurs distinctes pour chaque des zones d’intérêts d’un bassin. |
| **EF19 - Définir des zones inaccessibles dans un bassin** | Le système doit permettre de définir des zones inaccessibles pour sous-marin autonome. |
| **EF40 - Exporter le plan des zones d’intérêt en XML** | Le système doit permettre d’exporter le plan du bassin en format XML. |

**Exigence non-fonctionnelles**

N/A

### Bassin

**Exigence fonctionnelles**

Tableau 5 Exigences fonctionnelles des bassins

|  |  |
| --- | --- |
| **EF01 - Sélectionner un bassin** | Le système doit permettre de sélectionner un bassin parmi une liste prédéfinie par le club S.O.N.I.A. |
| **EF02 - Créer un nouveau gabarit de bassin** | Le système doit permettre de modéliser de nouveaux bassins. |
| **EF03 - Indiquer différentes profondeurs dans le bassin** | Le système doit permettre de configurer et d’afficher les profondeurs des différentes zones d’intérêt pour  un bassin sélectionné. |
| **EF05 - Afficher le nord sur le plan du bassin** | Le système doit afficher le nord à partir d’une boussole dans le haut de l’interface. |
| **EF06 - Pivoter un bassin** | Le système doit permettre de pivoter un bassin. |
| **EF08 - Ajouter une zone d’intérêt au bassin** | Lorsqu’un bassin a été chargé, le système doit permettre à l’utilisateur de sélectionner une zone d’intérêt  parmi une liste associée au bassin et d’ajouter à l’intérieur de celui-ci. |
| **EF15 - Modifier le point d’entrée et l’orientation à l’eau du sous-marin autonome** | Le système doit permettre de positionner le point d’entrée à l’eau du sous-marin autonome directement  sur le plan du bassin. |
| **EF16 - Définir une zone pour l’équipe à l’extérieur du bassin** | Le système doit permettre de définir une zone représentant la position à l’extérieur d’un bassin de  l’équipe opérant le sous-marin autonome. |
| **EF19 - Définir des zones inaccessibles dans un bassin** | Le système doit permettre de définir des zones inaccessibles pour sous-marin autonome. |

**Exigence non-fonctionnelles**

S1, S12

**Contraintes**

Tableau 6 Contraintes

|  |  |
| --- | --- |
| **CO07 - L’axe des X et celui des Y sont inversés dans le mode d’édition de bassin** | Afin de faciliter le travail et d’avoir des données positives pour représenter la profondeur, tout en suivant  la règle de la main droite, l’axe des ordonnées (X) et celui des abscisses (Y) doivent être inversés sur le  plan du bassin. |

### Système

**Exigence fonctionnelles**

Tableau 7 Exigences fonctionnelles du système

|  |  |
| --- | --- |
| **EF17 - Calculer la distance entre deux points** | Le système doit permettre de mesurer la distance séparant deux points. |
| **EF39 - Vérifier les temps alloués** | Le système doit afficher une alerte lorsque le temps courant des sous-missions d’une mission ne  correspond pas au temps global de la mission à laquelle il est associé. |
| **EF42 - Sauvegarder les travaux** | Le système doit permettre de sauvegarder dans un délai de 2 à 5 secondes le travail sur des bassins ou  des missions en cours. |
| **EF43 - Sauvegarder automatique des travaux** | Le système doit effectuer des sauvegardes périodiques toutes les 5 minutes lors de l’édition des bassins  ou des missions. |
| **EF44 - Annuler une opération** | Le système doit permettre d’annuler une opération qui a été effectuée lors de l’édition de bassins ou de  missions. |
| **EF45 - Refaire une opération** | Le système doit permettre de refaire une opération qui a été annulée lors de l’édition de bassins ou de  missions. |
| **EF48 - Afficher une icône d’avertissement et un message d’erreur lors d’une validation erronée** | Le système doit afficher un message d’avertissement d'une durée de 4 à 5 secondes lorsque la  sommation du temps des sous-missions est différente du temps global ou, lors de changement au temps  global, si la réallocation du temps global cause qu’un temps alloué à une sous-mission descend en bas  du temps minimum de cette sous-mission. |
| **EF49 - Sauvegarder temporairement les informations** | Le système doit effectuer une sauvegarde temporaire des informations lors d’édition de bassin ou de  missions dans un fichier séparé pour des fins de récupération du système. |
| **EF50 - Transférer la sauvegarde au sous-marin autonome** | Le système doit permettre d’envoyer une sauvegarde au sous-marin autonome par TCP/IP. |

**Exigence non-fonctionnelles**

S6, S8, S9, S10, S15

**Contraintes**

Tableau 8 Contraintes du système

|  |  |
| --- | --- |
| **CO01 - L’unité de base pour la longueur doit être le mètre** | Toutes les mesures (bassins, zone d’intérêt...) des longueurs doivent être en mètre. L’application doit donc afficher et enregistrer toutes les informations dans ce format. |
| **CO02 - Le logiciel doit être développé en Java ou en SWT** | Étant donné que ce projet implique la fusion entre le Pool Editor et le Mission Editor, il est important qu’il soit développé en Java ou en SWT afin de permettre un maximum de réutilisabilité de code possible. |
| **CO03 - Le format XML avec balises prédéfinies doit être utilisé lors de la sauvegarde d’un plan** | Afin que le sous-marin puisse interpréter le plan des missions et la situation des zones d’intérêt, il est  absolument essentiel que Mission Editor 2.0 sauvegarde les informations dans un document XML où  chacun des éléments et des noeuds respectent la mise en forme imposée par le club. |
| **CO04 - Les unités de base pour le temps sont la minute et la seconde** | Tous les temps définis dans le système (temps global, temps alloué, temps minimum, temps maximum)  doivent être basés sur des minutes et secondes. |
| **CO05 - Le poids d’une sous-mission doit être défini par un entier** | Le poids n’a pas d’unité de mesure, mais il doit être entré en tant qu’entier. |
| **CO06 - Le système doit être développé pour accommoder une résolution de 1024x768 pixels** | Toutes les interfaces présentes dans le système doivent être visibles adéquatement dans une résolution  de 1024x768 pixels. Il faut également développer le système afin de pouvoir éventuellement  accommoder une résolution de 1920x1080 pixels. |
| **CO08 - Hiérarchie des missions et des sous-missions** | La création de missions et sous-missions doit conserver une hiérarchie. Une mission peut posséder zéro  à plusieurs sous-missions et une sous-mission ne peut contenir de missions ni d’autres sous-missions. La  structure entre mission et sous-mission se limite à un seul niveau. |
| **CO09 - Installer depuis un fichier JAR** | Le système doit être installé depuis un fichier de type JAR. |

## Scénarios de qualités

**S1 - Convivialité**

* **Scénario complet** : L’utilisateur veut rajouter une zone d’intérêt au bassin. Il veut pouvoir le faire en utilisant la souris le moins de clics de souris possible.
* **Source** : L’utilisateur du Mission Editor 2.0.
* **Stimulus** : L’utilisateur souhaite ajouter une zone d’intérêt au bassin.
* **Artefact** : Le logiciel Mission Editor 2.0
* **Environment**: En execution.
* **Réponse** : Possibilité de sélectionner le champ suivant en utilisant le contrôle de tabulation. Les champs sont placés selon l’ordre dans lesquels ils doivent être remplis.
* **Mesure** : Prend au maximum 5 clics de souris pour ajouter une zone d’intérêt au bassin.

**S2 - Convivialité**

* **Scénario complet** : Un utilisateur veut valider la configuration d’une mission pour vérifier sa conformité
* **Source** : Un utilisateur
* **Stimulus** : Un utilisateur clique sur le bouton de validation de mission
* **Artefact** : Mission Editor 2.0
* **Environnement** : En execution.
* **Réponse** : Message indiquant si la mission est conforme ou non.
* **Mesure** : La satisfaction de l’utilisateur face à la validation effectué.

**S3 - Convivialité**

* **Scénario complet** : Un nouvel utilisateur essais de créer une nouvelle mission à partir du « Mission Editor 2.0.
* **Source** : Nouvel utilisateur.
* **Stimulus** : Volonté de créer une mission
* **Artefact** : Mission Editor 2.0
* **Environnement** : En execution.
* **Réponse** : Des valeurs par défauts sont fournis pour les champs où c’est possible.
* **Mesure** : La création d’une mission prend au maximum 15 minutes.

**S4 - Convivialité**

* **Scénario complet** : Un utilisateur cherche un état sans connaitre exactement le nom de l’étiquette ou de la classe.
* **Source** : Un utilisateur Mission Editor 2.0.
* **Stimulus** : Entre du texte dans la barre de recherche
* **Artefact** : Le système Mission Editor 2.0.
* **Environnement** : En mode normal ou surchargé.
* **Réponse** : Afficher les 5 premiers états qui commencent par les lettres entrés dans la barre de recherche.
* **Mesure** : Satisfaction de l’usager

**S5 - Convivialité**

* **Scénario complet** : Un utilisateur veut voir les zones d’intérêts des bassins quand il est en mode « édition de mission ».
* **Source** : Un utilisateur du Mission Editor 2.0.
* **Stimulus** : Modifie une mission
* **Artefact** : Le système Mission Editor 2.0
* **Environnement** : En mode normal ou surchargé.
* **Réponse** : Affiche les zones d’intérêts de chaque bassin associé à la mission.
* **Mesure** : Temps gagné par l’usager

**S6 - Convivialité**

* **Scénario complet** : Un développeur en charge de l’interface utilisateur veut pouvoir facilement la modifier. L’interface utilisateur et le code devrait être séparé.
* **Source** : Un développeur
* **Stimulus** : Modifier l’interface utilisateur
* **Artefact** : Le système Mission Editor 2.0
* **Environnement** : En design
* **Réponse** : L’architecture MVC est implémentée pour séparer l’interface utilisateur du code.
* **Mesure** : Temps gagné par le développeur.

**S7 - Convivialité**

* **Scénario complet** : Un utilisateur veut revenir en arrière après avoir apporté une modification à une mission, sans avoir à refaire ce qu’il avait fait avant.
* **Source** : Un utilisateur
* **Stimulus** : Modifier une mission
* **Artefact** : Le système Mission Editor 2.0
* **Environnement** : Normal, en exécution
* **Réponse** : Commande undo et redo mise à la disposition de l’usager
* **Mesure** : Satisfaction de l’usager

**S8 - Disponibilité**

* **Scénario complet** : L’utilisateur modifie une mission et une panne survient. Il doit pouvoir reprendre son travail le plus rapidement possible.
* **Source** : Interne. L’éditeur de mission Mission Editor 2.0
* **Stimulus** : Crash. Le système ne répond plus.
* **Artefact** : L’éditeur de mission Mission Editor 2.0
* **Environnement** : Système en opération normale.
* **Réponse** : Lancer une exception contenant l’identifiant et le message de l’erreur.
* **Mesure** : Temps moyen pour réparer le système est moins de 30 secondes (MTTR). Temps moyen entre deux pannes (MTTF) ne dépasse pas une panne au trois heures.

**S9 - Disponibilité**

* **Scénario complet** : L’utilisateur essais de modifier une mission et le système plante. Il veut pouvoir continuer à travailler sans perdre toutes les modifications qu’il avait fait.
* **Source** : Interne. L’éditeur de mission Mission Editor 2.0
* **Stimulus** : Crash. Le système ne répond plus.
* **Artefact** : L’éditeur de mission Mission Editor 2.0
* **Environnement** : Système en opération normale.
* **Réponse** : Restoration du système.
* **Mesure** : Temps moyen pour continuer le travail après une panne est de 20 secondes.

**S10 - Disponibilité**

* **Scénario complet** : Un utilisateur change le temps alloués à une sous-mission. Le total des temps alloués n’est plus égal au temps global.
* **Source** : Un utilisateur
* **Stimulus** : La somme de tous les temps alloués n’est plus égale au temps global.
* **Artéfact** : Mission editor 2.0
* **Environnement** : Fonctionnement normal
* **Réponse** : Une alerte est affichée à l’écran
* **Mesure** : Erreur détectée après un maximum de 1 seconde.

**S11 - Performance**

* **Scénario complet** : Un utilisateur du système veut créer une mission et il dispose de peu de temps.
* **Source** : Un utilisateur Mission Editor 2.0.
* **Stimulus** : Demande de création de mission.
* **Artefact** : Le système Mission Editor 2.0.
* **Environnement** : En mode normal ou surchargé.
* **Réponse** : Traitement des demandes de mise à jour des fichiers de configuration.
* **Mesure** : Temps de latence maximal de 3 secondes.

**S12 - Performance**

* **Scénario complet** : Un utilisateur du système fait pivoter un bassin. La boussole doit être mise à jour en temps réel.
* **Source** : Un utilisateur Mission Editor 2.0.
* **Stimulus** : Pivoter un bassin
* **Artefact** : Le système Mission Editor 2.0.
* **Environnement** : En mode normal ou surchargé.
* **Réponse** : Mise à jour de la boussole en temps réel
* **Mesure** : Temps de latence maximal de 100 millisecondes.

**S13 - Performance**

* **Scénario complet** : Un utilisateur veut rapidement trouver un état en utilisant la barre de recherche.
* **Source** : Un utilisateur Mission Editor 2.0.
* **Stimulus** : Rechercher un état
* **Artefact** : Le système Mission Editor 2.0.
* **Environnement** : En mode normal ou surchargé.
* **Réponse** : Afficher le résultat de la recherche
* **Mesure** : Temps pris pour la recherche maximal est 1 seconde.

**S14 - Modification**

* **Scénario complet** : Un utilisateur veut modifier le code source de l’état d’une sous-mission. Ce code ne doit pas avoir d’impact sur autre chose que l’état modifié.
* **Source** : Un utilisateur
* **Stimulus** : Modification du code source de l’état d’une sous-mission.
* **Artefact** : Mission Editor 2.0
* **Environnement** : Normal ou surchargé
* **Réponse** : La modification ne doit pas avoir d’impact sur autre chose que l’état modifié. Le code doit être modulaire.
* **Mesure** : Aucun impact sur les autres fonctions du système.

**S15 - Modification**

* **Scénario complet** : Un développeur veut pouvoir modifier le système pour qu’il puisse accommoder une résolution 1920x1080 pixels.
* **Source** : Un développeur
* **Stimulus** : Veut modifier le système pour rajouter une nouvelle résolution.
* **Artefact** : Mission Editor 2.0
* **Environnement** : Design
* **Réponse** : Le système doit être conçu pour pouvoir facilement rajouter une nouvelle résolution.
* **Mesure** : Temps maximum de 1 h pour rajouter une nouvelle résolution.

## Présentation de l’architecture

### Vue module

Cette vue (voir figure 1) représente le style architectural utilise et décomposition modulaire. Le système est décomposé en trois packages principaux (modèle, vue et contrôleur). Le package vue contient chacun des éléments qui concernent l’interface utilisateur. Le package modèle contient les classes de la logique d’affaire et de l’accès à la source de données de l’application. Les sous-packages Mission et Bassin contiennent les classes de la logique d’affaire et les sous-packages DAOMission et DAOBassin contiennent les classes de l’accès à la source de données (fichiers XML des missions et des bassins). Le package contrôleur contient les gestionnaires d’événement, qui sont des classes qui traitent les interactions de l’utilisateur avec les composants GUI de la vue.

#### Présentation primaire

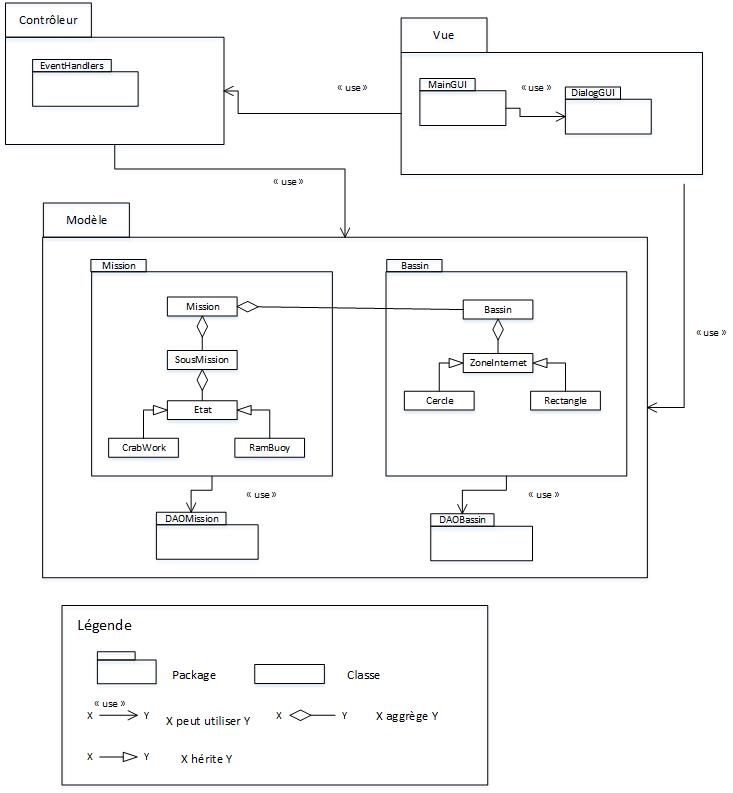


Figure 1 vue module

#### Responsabilités

Tableau 9 Responsabilités des éléments de la vue module

|  |  |
| --- | --- |
| **Élément de la vue** | **Responsabilité(s)** |
| Package contrôleur | Package qui contient les classes liées au contrôleur MVC. Ces classes seront surtout des gestionnaires d’événements. |
| Package EventHandlers | Package qui contient les classes qui permettent de gérer les événements de l’utilisateur (clic de bouton, sélection dans une liste, etc.). |
| Package modèle | Package qui contient les classes du modèle MVC. Ces classes contiennent les données qui sont utilisées par l’application. |
| Package Mission | Package qui contient les classes liées aux missions. |
| Package Bassin | Package qui contient les classes liées aux bassins. |
| Package DAOMission | Package qui contient les classes qui permettent l’accès(en lecture ou en écriture) au fichier XML des missions. |
| Package DAOBassin | Package qui contient les classes qui permettent l’accès(en lecture ou en écriture) au fichier XML des bassins. |
| Package vue | Package qui contient les classes des vues MVC. Ces classes sont surtout des fenêtres et des boîtes de dialogue dont le rôle est d’afficher les données du modèle. |
| Package MainGUI | Package qui contient les classes associées aux fenêtres. |
| Package DialogGUI | Package qui contient les classes associées aux boîtes de dialogue. |
| Classe Mission | Classe qui représente une mission. |
| Classe SousMission | Classe qui représente une sous-mission d’une mission. |
| Classe Etat | Classe qui représente un état d’une sous-mission. |
| Classe CrabWork | Spécialisation de la classe Etat |
| Classe RamBuoy | Spécialisation de la classe Etat |
| Classe Bassin | Classe qui représente un bassin. |
| Classe ZoneInteret | Classe qui représente une zone d’intérêt d’un bassin |
| Classe Cercle | Spécialisation de la classe ZoneInteret qui représente une zone d’intérêt en forme de cercle. |
| Classe Rectangle | Spécialisation de la classe ZoneInteret qui représente une zone d’intérêt en forme de rectangle. |
| Relation utilise | Un module peut utiliser les classes d’un autre module vers où la flèche de la relation pointe. |
| Relation hérite | Une classe est la spécialisation d’une autre en héritant ses propriétés et méthodes. |
| Relation agrège | Une classe contient une ou plusieurs instances d’une autre classe. |

#### Cas d’utilisation

Tableau 10 Correspondance des éléments de la vue avec les cas d'utilisation

|  |  |
| --- | --- |
| **Élément de la vue** | **Cas d’utilisation** |
| Package contrôleur | Tous les cas d’utilisation |
| Package EventHandlers | Tous les cas d’utilisation |
| Package modèle | Tous les cas d’utilisation |
| Package Mission | CU11, CU12, CU13, CU14, CU15, CU16, CU17, CU18, CU19 |
| Package Bassin | CU01, CU02, CU03, CU04, CU05, CU06, CU07, CU08, CU09, CU10, CU14, CU20 |
| Package DAOMission | CU14 |
| Package DAOBassin | CU14 |
| Package vue | Tous les cas d’utilisation |
| Package MainGUI | Tous les cas d’utilisation |
| Package DialogGUI | CU02, CU04, CU05, CU06, CU12, CU14, CU15, CU17, CU18, CU19 |
| Classe Mission | CU01, CU02, CU03, CU11, CU12, CU13, CU14, CU15 |
| Classe SousMission | CU12, CU13, CU17, CU18, CU19 |
| Classe Etat | CU16 |
| Classe CrabWork | CU16 |
| Classe RamBuoy | CU16 |
| Classe Bassin | CU04, CU05, CU06, CU07, CU08, CU09, CU14, CU20 |
| Classe ZoneInteret | CU10 |
| Classe Cercle | CU10 |
| Classe Rectangle | CU10 |

#### Documentation des interfaces

Voir la documentation des interfaces de la vue modèle-vue-contrôleur.

#### Scénarios de qualité

Tableau 11 Correspondance des éléments de la vue avec les scénarios de qualité

|  |  |
| --- | --- |
| **Élément de la vue** | **Scénario(s) de qualité** |
| Package contrôleur | S6 |
| Package EventHandlers |  |
| Package modèle | S6 |
| Package Mission | S2 |
| Package Bassin |  |
| Package DAOMission | S11 |
| Package DAOBassin | S11 |
| Package vue | S1, S2, S3, S4, S5, S6, S7, S8, S10, S12, S13, S15 |
| Package MainGUI | S2, S4, S5, S7, S12, S13 |
| Package DialogGUI | S1, S2, S3, S8, S10 |
| Classe Mission | S2, S10 |
| Classe SousMission | S4, S13 |
| Classe Etat | S14 |
| Classe CrabWork |  |
| Classe RamBuoy |  |
| Classe Bassin |  |
| Classe ZoneInteret |  |
| Classe Cercle |  |
| Classe Rectangle |  |

#### Contraintes de conception

* Pas de boucles permises dans la décomposition des package du graphe.
* Un package ne peut avoir qu’un seul parent.

### Vue modèle-vue-contrôleur (C&C)

Dans toute application de manipulation de données à partir d’une interface graphique, un facteur très important dans le choix des tactiques est de séparer les données de l’affichage de celle-ci. Étant donné que les données du « Mission Editor 2.0 » sont relativement bien définies, une architecture où la vue dépend des données et non vice-versa a été favorisée. Finalement, un dernier critère important est de pouvoir garder la vue synchronisée avec les données. Ainsi, un bassin est modifié, les changements devraient être reflétés par la vue.

Notre vue C&C représente l’architecture « Publisher/Subscriber » du « Mission Editor 2.0 ». Cette architecture permet de construire l’interface utilisateur par-dessus les modèles de données et synchroniser cette interface avec les changements aux données. Tout d’abord, « EventHandler » est un composant qui permet de gérer le signalement de changement. Ainsi, si un modèle de données est modifié, « EventHandler » s’occupe de notifier la vue associée. « Views » représente une vue qui affiche sous forme d’interface graphique les données provenant des modèles. Ces vues sont associées aux modèles via la relation « s’abonne » et « notifie ». Un abonnement permet de s’enregistrer pour recevoir les futures notifications de changement. La relation « notifie » est un message indirect qu’un modèle envoie à la vue pour la notifier de son changement. Les deux « Modèle manager » représentent les données logiques manipulées par l’application. Finalement, ces deux modèles de données sont synchronisés avec leur fichier XML qui permet de garder ces données sur le disque.

#### Présentation primaire

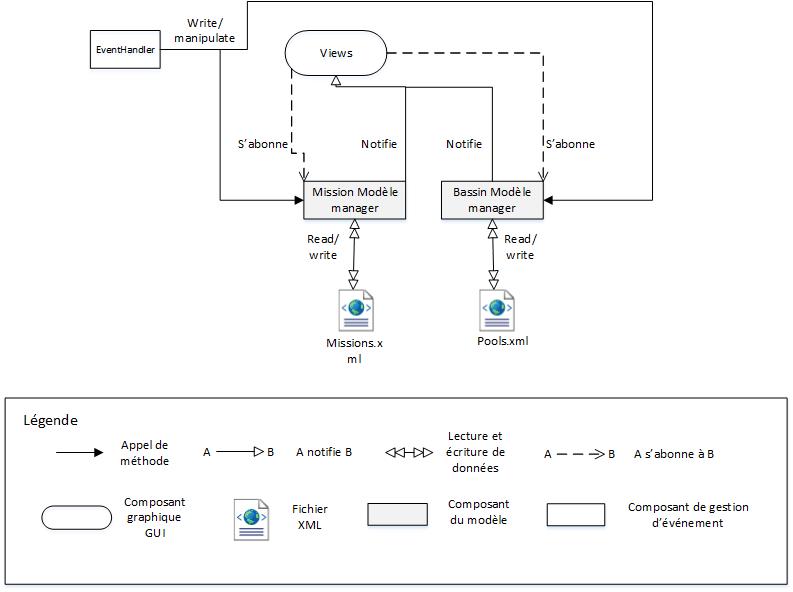


Figure 2 Vue composant et connecteur

#### Liste d’éléments

##### Éléments

**Composant graphique views**

Composant graphique (GUI) dont le rôle est d’afficher les données de l’application à l’écran. Ce composant s’abonne aux événements du modèle. Lorsqu’un changement survient dans les données du modèle, ce composant reçoit une notification de la part de celui-ci. Cette notification permet de mettre à jour les informations affichées à l’écran.

**Composant du modèle Mission modèle manager**

Composant qui contient les données relatives à une mission (nom, temps global, etc.). Lorsque les données subissent une modification, ce composant envoie une notification au composant View pour que celui-ci puisse se mettre à jour.

**Composant du modèle Bassin modèle manager**

Composant qui contient les données relatives à un bassin (angle, échelle, angle avec le nord, etc.). Lorsque les données subissent une modification, ce composant envoie une notification au composant View pour que celui-ci puisse se mettre à jour.

**Composant de gestion d’événements EventHandler**

Composant dont le but est de gérer les interactions de l’utilisateur avec la vue (clic de bouton, clic de souris, mouvement de glisser-déposer, etc.). C’est via ce composant que les données du modèle sont manipulées.

**Fichier XML Missions.xml**

Fichier au format de données .xml qui sert de support de données persistantes pour toute information relative au(x) mission(s).

**Fichier XML Pools.xml**

Fichier au format de données .xml qui sert de support de données persistantes pour toute information relative au(x) bassin (s).

##### Relations

Voir la figure de la section présentation primaire.

#### Interfaces

**Identité de l’interface**

Cette interface se nomme MissionManagerService. Le rôle de cette interface est de gérer des missions.

**Ressources**

ajouterSousMission(SousMission smObject)

Ajouter une sous-mission à la mission en cours d’édition.

**Préconditions**

* L’argument smObject ne doit pas être null.

**Postconditions**

* En cas de succès, une sous-mission est ajoutée à la mission.

**Restrictions de l’usage**

N/A

**Gestion des erreurs**

* InvalidSMException. Cette exception est levée si smObject est null.

**Ressources**

supprimerSousMission(SousMission smObject)

Supprimer une sous-mission à la mission en cours d’édition.

**Préconditions**

* L’argument smObject ne doit pas être null.

**Postconditions**

* En cas de succès, une sous-mission est retirée de la mission en cours d’édition.

**Restrictions de l’usage**

N/A

**Gestion des erreurs**

* InvalidSMException. Cette exception est levée si smObject est null.

**Ressources**

CreerBassin(Bassin objBassin)

Un nouveau bassin est ajouté à la mission en cours d’édition.

**Préconditions**

* L’argument objBassin ne doit pas être null.

**Postconditions**

* En cas de succès, un nouveau bassin est ajouté à la mission en cours d’édition.

**Restrictions de l’usage**

N/A

**Gestion des erreurs**

* InvalidSMException. Cette exception est levée si objBassin est null.

**Ressources**

supprimerBassin(Bassin objBassin)

Un bassin est supprimé de la mission en cours d’édition.

**Préconditions**

* L’argument objBassin ne doit pas être null.

**Postconditions**

* En cas de succès, un bassin est suppriméde la mission en cours d’édition.

**Restrictions de l’usage**

N/A

**Gestion des erreurs**

* InvalidSMException. Cette exception est levée si objBassin est null.

**Ressources**

Boolean valider()

Valider la configuration de la mission en cours d’édition.

**Préconditions**

N/A

**Postconditions**

* Retourne une valeur booléenne indiquant si la configuration de la mission en cours d’édition est valide ou non.

**Restrictions de l’usage**

N/A

**Gestion des erreurs**

N/A

**Ressources**

modifierTempsGlobal(tempGlobal objTempsGlobal)

Modifier le temps global d’une mission. La proportion du temps de chacune des sous-missions est conservée.

**Préconditions**

* L’argument smObject ne doit pas être null.

**Postconditions**

* En cas de succès, le temps alloué de chacune des sous-missions est réajustée.

**Restrictions de l’usage**

N/A

**Gestion des erreurs**

* InvalidTGException. Cette exception est levée si objTempsGlobal est null.

#### Comportement

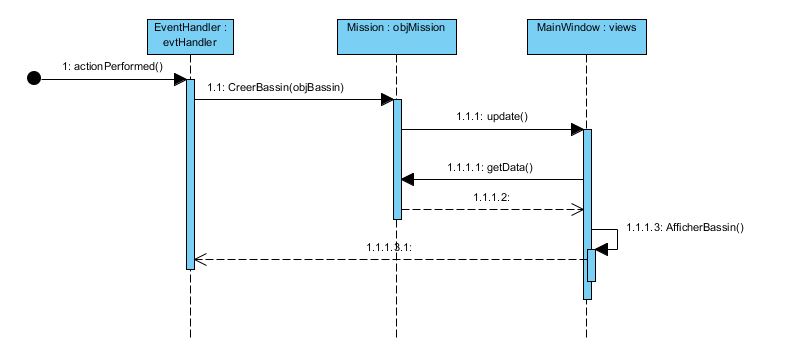


Figure 3 Diagramme de sequence illustrant le comportant MVC

Ce diagramme de séquence illustre le comportement des objets lorsqu’un utilisateur interagit avec les composants GUI de la vue. Lorsqu’un utilisateur crée un nouveau bassin, le gestionnaire d’événements ajoute le nouveau bassin à la mission en cours d’édition. Ensuite, la mission envoie une notification à la vue (fenêtre principale où est représenté le bassin) pour que celle-ci puisse mettre à jour les données à afficher à l’écran.

#### Diagramme de contexte

N/A

#### Variability Guide

N/A

#### Exposé des motifs

Le patron architectural MVC a été choisi afin de séparer l’interface usager du code et ainsi rencontrer l’attribut de qualité de la convivialité et de la modificabilité.

### Vue déploiement

La vue d’Allocation doit pouvoir refléter les différents fichiers utilisés par le système ainsi que l’environnement du club SONIA, c’est-à-dire un mélange d’ordinateurs du club et des membres.

Tout d’abord, la vue utilisée est la vue de déploiement afin de bien représenter l’environnement dans lequel le système opère. Le système est représenté sous forme de fichier « jar », sois l’artefact « Mission-Editor-2.0.jar ». Les sous-missions développées indépendamment sont aussi représentées sous forme d’archive Java « Sous-Mission.jar ». Ces deux fichiers sont déployés sur les postes du club ou les ordinateurs personnels afin que les membres puissent créer de nouvelle mission ou bassin. Les missions et bassins sont par la suite enregistrés sous forme de fichier XML, sois respectivement « Mission.xml » et « Bassin.xml ». Finalement, ces fichiers sont envoyés via une connexion interne à l’AUV.

#### Présentation primaire

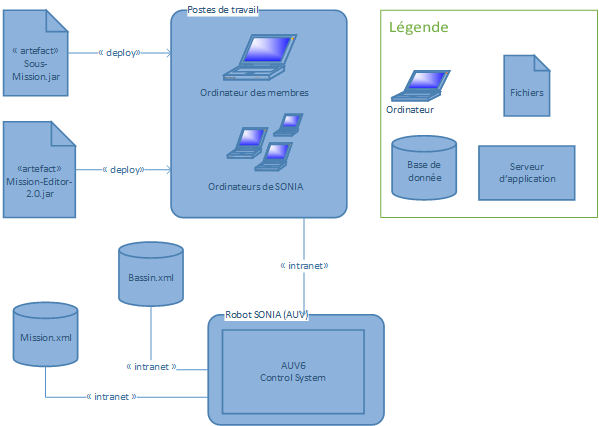


Figure 4 Vue déploiement

#### Liste d’éléments

##### Éléments

**Ordinateur des membres**

Ordinateur personnel d’un des membres du club sur lequel l’application est installée.

**Ordinateurs de SONIA**

Ordinateur qui appartient au club sur lequel l’application est installée.

**Fichier Sous-Mission.jar**

Fichier d’archive Java qui contient les ressources en lien avec les sous-missions.

**Fichier Mission-Editor2.0.jar**

Fichier d’archive Java qui contient les principales ressources de l’application.

**Base de données Bassin.xml**

Fichier au format xml sur lequel on y sauvegarde les données des bassins.

**Base de données Mission.xml**

Fichier au format xml sur lequel on y sauvegarde les données des missions.

**AUV6 Control System**

Serveur d’application sur lequel on y enregistre les fichiers XML.

##### Relations

Voir la figure de la représentation primaire.

#### Interfaces

**Identité de l’interface**

Cette interface se nomme MissionManagerService. Le rôle de cette interface est de gérer des missions.

**Ressources**

sauvegarderEnXML()

Sauvegarder la mission en cours d’édition en format XML.

**Préconditions**

N/A

**Postconditions**

* En cas de succès, la mission en cours d’édition est sauvegardée en format XML.

**Restrictions de l’usage**

N/A

**Gestion des erreurs**

* InvalidMissionException. Cette exception est levée si la configuration de la mission n’est pas valide.

#### Type de données et constantes

N/A

#### Comportement

N/A

#### Diagramme de contexte



#### Variability Guide

N/A

#### Exposé des motifs

La vue de déploiement a été choisie afin de bien représenter l’environnement de développement sur lequel le système opère.

# Analyse ATAM

## Documents sources

Voir section documentation architecturale.

## Règles et standards

« Description de ATAM »

## Nature et mission du système

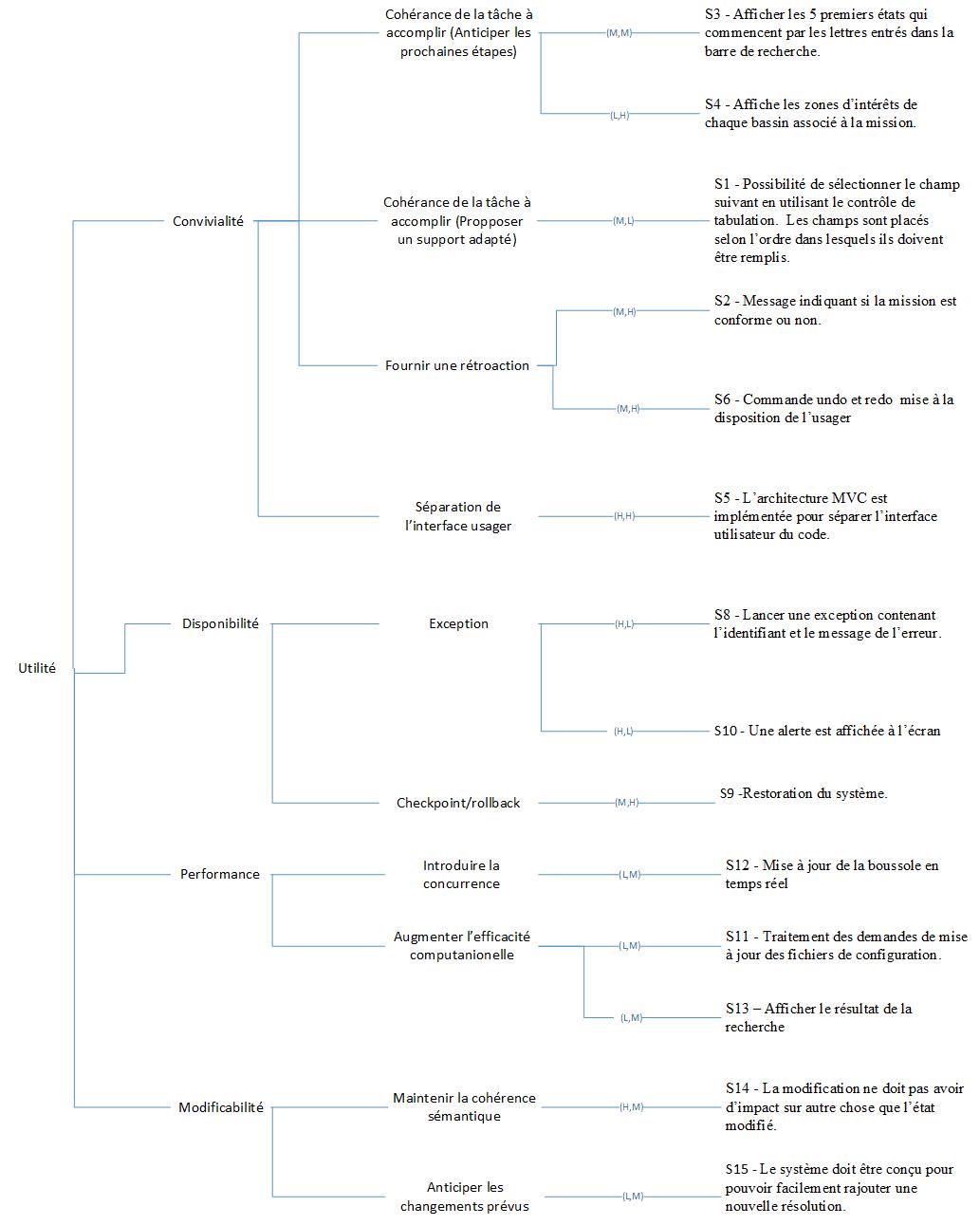
Le système a comme mission de fournir les fonctionnalités nécessaires à un utilisateur afin de pouvoir créer et modifier les bassins ainsi que les missions utilisé par le sous-marin du club SONIA. Le système est basé sur deux logiciels présentement utilisé par SONIA, soit le « Mission Editor » et le « Pool Editor ». Ainsi, le nouveau système est contraint de fonctionner avec le format présent des bassins et missions. Les parties prenantes sont le club SONIA ainsi que les architectes du système. Ce système doit particulièrement être fiable et résistant aux pannes puisqu’il est utilisé lors des tests où le temps est très limité et chaque seconde mise sur l’édition d’un bassin ou d’une mission est du temps perdu pour les tests.

## Présentation de l'architecture et des approches architecturales

L’architecture est contrainte par les systèmes déjà existant. D’abord, le système doit pouvoir fonctionner au moins sur Windows. Étant donnée la contrainte d’utilisation du langage portable JAVA, L’architecture devrait pouvoir fonctionner sur tous les systèmes sans avoir besoin de tactique spéciale. Le « Mission Editor 2.0 » doit pouvoir interagir avec l’AUV en créant des fichiers représentant les bassins et les missions qui seront envoyé à l’AUV. Ainsi, l’architecture devrait utiliser un modèle par couche afin de pouvoir isoler l’écriture et la lecture de données dans le format d’origine dans une couche inférieur. Le système doit aussi récupérer les sous-missions disponibles à partir de librairies Java externe. Une architecture sous forme de service permet de créer un service dédié au chargement de ses librairies et fournir la liste de sous-mission au système.

Ainsi, comme vu dans la documentation de l’architecture, le système sépare clairement les données à d’une division des modules sous forme de Modèle-Vue-Contrôleur. Cette approche permet d’aider le développement d’une interface conviviale et fluide. En y ajoutant une architecture « Publisher/Subscriber » il nous est possible de synchroniser la vue aux changements de données. De plus, la détection des changements nous permettent de créer facilement un système de sauvegarde automatique pour mieux gérer la fiabilité du système.

## Arbre d’utilité



« Justifier le choix des scénarios utilisés à la prochaine section + commentaires»

## ATAM – Analyse

Les 6 scénarios analysés ont été extraits de l’arbre d’utilité en choisissant les scénarios à plus hautes priorité. Chaque scénario a été analysé en suivant la méthode ATAM.

### Analyse de S05

|  |  |  |  |
| --- | --- | --- | --- |
| **Scénario** : Un développeur en charge de l’interface utilisateur veut pouvoir facilement la modifier. L’interface utilisateur et le code devrait être séparé. | | | |
| **Attribut** : Convivialité | | | |
| **Environnement** : Environnement de développement | | | |
| **Stimulus** : Un développeur veut modifier l’interface graphique | | | |
| **Décision d’architecture** | **Risques** | **Points de sensibilité** | **Compromis** |
| Découplage de la vue des données à l’aide du patron MVC. | R03, NR03 | PS03 |  |
| **Raisonnement** :   * Une architecture MVC permet de découpler la logique de domaine de la vue et de centraliser le lien entre ces deux concepts dans un contrôleur. * En découplant l’interface utilisateur du domaine, il est beaucoup plus facile de changer l’interface sans affecter la logique du domaine et donc de pouvoir réitérer plus facilement sur le design de l’interface afin d’avoir une interface utilisateur plus conviviale. | | | |

### Analyse de S07

|  |  |  |  |
| --- | --- | --- | --- |
| **Scénario** : Un utilisateur veut revenir en arrière après avoir apporté une modification à une mission, sans avoir à refaire ce qu’il avait fait avant. | | | |
| **Attribut** : Convivialité | | | |
| **Environnement** : Système en production | | | |
| **Stimulus** : L’utilisateur veut retourner en arrière | | | |
| **Décision d’architecture** | **Risques** | **Points de sensibilité** | **Compromis** |
| Garder un historique de chaque action | R06 | PS10 | C05 |
| Utilisation de transaction avec « Rollback » |  | PS09 |  |
| **Raisonnement** :   * L’utilisation d’un historique de chaque action permet de garder une trace de l’évolution du système et donc de retourner dans un état passé en inversant les actions des actions. * En utilisant un système de transaction pour les actions, il est possible d’annuler une action en retourner à l’état initial du système avant que l’action soit exécuté. | | | |

### Analyse de S08

|  |  |  |  |
| --- | --- | --- | --- |
| **Scénario** : L’utilisateur modifie une mission et une panne survient. Il doit pouvoir reprendre son travail le plus rapidement possible. | | | |
| **Attribut** : Disponibilité | | | |
| **Environnement** : Système en production | | | |
| **Stimulus** : Le système ne répond plus | | | |
| **Décision d’architecture** | **Risques** | **Points de sensibilité** | **Compromis** |
| Utilisation d’exception | R04 | PS04 | C03 |
| Utilisation de transaction avec « Rollback » |  | PS05 |  |
| **Raisonnement** :   * L’utilisation d’exceptions permet d’interrompre une action à n’importe quel moment et retourner à un point défini de l’application en prenant soin de libérer les ressources allouées. * Un système de transaction permet de revenir à un état stable en cas d’exception lors d’une opération non atomique. | | | |

### Analyse de S09

|  |  |  |  |
| --- | --- | --- | --- |
| **Scénario** : L’utilisateur essaie de modifier une mission et le système plante. Il veut pouvoir continuer à travailler sans perdre toutes les modifications qu’il avait faites. | | | |
| **Attribut** : Disponibilité | | | |
| **Environnement** : Système en production | | | |
| **Stimulus** : Un utilisateur effectue une action à partir de l’interface utilisateur | | | |
| **Décision d’architecture** | **Risques** | **Points de sensibilité** | **Compromis** |
| Copie de sauvegarde automatique | R02, NR01, NR02 | PS01 | C01, C02 |
| Restauration d’une copie de sauvegarde après panne | R01, R02 | PS02 |  |
| **Raisonnement** :   * Avec les sauvegarde automatique, il est peu probable de perdre plus de temps que l’intervalle de sauvegarde automatique en cas de panne. * Les sauvegardes sont déjà implémentées sous forme de sauvegarde manuelle. Le système est donc déjà disponible et testé. * Transparent pour l’utilisateur. | | | |

### Analyse de S10

|  |  |  |  |
| --- | --- | --- | --- |
| **Scénario** : Une alerte est affichée à l’écran en cas d’exception. | | | |
| **Attribut** : Disponibilité | | | |
| **Environnement** : Système en production | | | |
| **Stimulus** : Une exception est lancée lors d’une action | | | |
| **Décision d’architecture** | **Risques** | **Points de sensibilité** | **Compromis** |
| Utilisé un « aspect » pour chaque action qui s’occupe d’attraper les exceptions et d’afficher une erreur. | R04 | PS06 |  |
| **Raisonnement** :   * En utilisant un gestionnaire d’exception qui affiche les détails de l’erreur dans une boîte de dialogue, l’utilisateur a un retour sur l’erreur qui vient de se produire et peut donc prendre une décision informé pour la corriger. * En utilisant la tactique d’architecture « aspect », l’aspect qui gère les exceptions peut être conçu une seule fois et réutiliser pour chaque action. | | | |

### Analyse de S14

|  |  |  |  |
| --- | --- | --- | --- |
| **Scénario** : Un utilisateur veut modifier le code source de l’état d’une sous-mission. Ce code ne doit pas avoir d’impact sur autre chose que l’état modifié. | | | |
| **Attribut** : Modificabilité | | | |
| **Environnement** : Environnement de développement | | | |
| **Stimulus** : Un développeur veut modifier le code d’une sous-mission. | | | |
| **Décision d’architecture** | **Risques** | **Points de sensibilité** | **Compromis** |
| Désigner un module qui gère les sous-missions et les archives de sous-mission. | NR04 | PS07 | C04 |
| Un Pipe-and-filter permet de valider et compiler la nouvelle sous-missions et de mettre à jour la librairie JAR. | R05 | PS08 |  |
| **Raisonnement** :   * Les sous-missions doivent être extraite à partir d’une librairie JAR ou d’un dossier avec l’ensemble des class Java représentant les sous-missions. Isoler cette logique dans un module permet de la découpler du reste de l’application. * Pipe and Filter permet de piper seulement dans le cas où une sous-missions a été valider et compiler avant d’être ajouté à la bibliothèque de sous-mission. Ainsi, les sous-missions erronées peuvent être pipées vers un autre filtre qui affiche un message d’erreur. Cette architecture permet aussi l’ajout facile de nouvelle validation de sous-mission tel que des tests unitaires. | | | |

### Risques

|  |  |
| --- | --- |
| R01 | La restauration d’une sauvegarde de copie lors du redémarrage peut ralentir le démarrage de l’application et causer une frustration chez l’utilisateur. |
| R02 | La panne a eu lieu de l’écriture de la sauvegarde de copie résultant en une sauvegarde corrompue. |
| R03 | L’interface utilisateur dépend du modèle de donnée des bassins et missions. |
| R04 | Un module du système peut causer une erreur à tout moment et lancer une exception. |
| R05 | La modification du code des sous-missions peut contenir des erreurs. |
| R06 | L’annulation d’une action peut causer une exception. |

### Non-risques

|  |  |
| --- | --- |
| NR01 | Les erreurs de sauvegarde de copie sont bien gérées et ne produisent pas de panne. |
| NR02 | La sauvegarde de copie se fait silencieusement en arrière-plan et n’affecte pas les performances du système. |
| NR03 | Les modèle de données pour les missions et bassin sont déjà définis par le système présentement en place et ne devrait pas changer. |
| NR04 | Chaque sous-mission est représentée par une classe indépendante des autres sous-missions. |
| NR05 | Chaque action peut être représentée en mémoire et être réversible. |

### Point de sensibilité

|  |  |
| --- | --- |
| PS01 | Une panne cause la perte totale des données depuis la dernière sauvegarde manuelle de l’utilisateur. |
| PS02 | Après une panne, le système récupère la dernière sauvegarde et la restaure après avoir vérifié si elle est corrompue. |
| PS03 | Si le code de l’interface et de la logique de domaine est trop dépendant, il sera difficile de changer l’un sans affecter l’autre. |
| PS04 | Une exception non attrapé peut résulter en un plantage irréparable du système. |
| PS05 | Si une exception survient lors d’une transaction incomplète le système reste dans un état incohérent. |
| PS06 | L’utilisateur doit pouvoir comprendre pourquoi son action n’a pu être complétée, sinon il lui sera impossible de corriger l’erreur. |
| PS07 | Une erreur de syntaxe dans le code d’une sous-mission modifié peut empêcher la création d’une nouvelle archive JAR avec toute les sous-missions. |
| PS08 | L’archive des sous-missions ne doit pas être supprimé ou remplacé par une nouvelle version non-fonctionnelle. |
| PS09 | Une exception lors de l’annulation d’une action peut laisser le système dans un état incohérent. |
| PS10 | Pour annuler une action, le système doit garder une trace de l’évolution des états au travers du temps. |

### Compromis

|  |  |
| --- | --- |
| C01 | Si une panne survient lors de l’enregistrement il est possible que cet enregistrement soit corrompus ou incomplet. L’alternative proposée est de garder plusieurs sauvegardes, ainsi il est possible de restaurer la sauvegarde la plus récente fonctionnelle. Par contre cela peut résulter en plusieurs sauvegardes. Il est donc nécessaire de trouver le bon nombre de sauvegarde garder optimal par rapport au risque de perdre toute les sauvegarde. |
| C02 | La sauvegarde de copie ne peut-être effectuer lors de chaque modification des données puisque cela risquerait d’engorger les écritures de disque. Il est donc nécessaire de trouver un intervalle de temps raisonnable pour effectuer les sauvegardes automatiques. |
| C03 | Chaque action doit être enveloppée dans un bloc d’exception. Ainsi, chaque une action peut être interrompue par une exception sans faire planter le système. |
| C04 | Si les sous-missions sont garder dans une archive JAR, il n’est pas possible de modifier une classe sans modifier l’ensemble de l’archive, et donc l’ensemble des sous-missions. Par contre, il est possible de s’assurer de la validité d’une sous-mission, de la recompiler et de recréer l’archive avec la nouvelle sous-mission. |
| C05 | Étant donnée la taille limité en mémoire, il est impossible de garder une infinité d’état du système afin de revenir en arrière. Il est donc nécessaire de déterminé un nombre raisonnable d’état garder en mémoire permettant ainsi de pouvoir annuler autant d’action que d’état garder en mémoire. |

# Conclusion

Pour conclure, notre documentation se décline en trois vues. La vue de type module, qui est composée du style architectural utilise et décomposition modulaire, permettra surtout de rencontrer les attributs de qualité relatifs à la modificabilité. La deuxième vue de type composant et connecteur, qui représente le patron d’architecture logiciel MVC, contribuera surtout à rencontrer les attributs de qualité en lien avec la convivialité. La troisième vue de type affectation de notre documentation sert à représenter l’environnement de notre système durant le déploiement. Quant à notre analyse ATAM, les scénarios de qualité à analyser ont été choisis en fonction du niveau de priorité de ces scénarios, soit des scénarios de qualité liés à la convivialité et à la disponibilité.