

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 4](#_Toc384203752)

[Liste des figures 4](#_Toc384203753)

[Introduction 5](#_Toc384203754)

[Conception architecturale 6](#_Toc384203755)

[Documents sources 6](#_Toc384203756)

[Règles et standards 6](#_Toc384203757)

[Prérequis 6](#_Toc384203758)

[Mission 7](#_Toc384203759)

[Sous-mission 8](#_Toc384203760)

[État 9](#_Toc384203761)

[Zone d’intérêt 10](#_Toc384203762)

[Bassin 11](#_Toc384203763)

[Système 12](#_Toc384203764)

[Scénarios de qualités 14](#_Toc384203765)

[Présentation de l’architecture 23](#_Toc384203766)

[Vue module : 23](#_Toc384203767)

[Vue C&C : 24](#_Toc384203768)

[Vue Allocation 25](#_Toc384203769)

[Analyse ATAM 26](#_Toc384203770)

[Documents sources 26](#_Toc384203771)

[Règles et standards 26](#_Toc384203772)

[Mission 26](#_Toc384203773)

[Présentation de l'architecture et des approches architecturales 26](#_Toc384203774)

[Arbre d’utilité 27](#_Toc384203775)

[ATAM – Analyse 28](#_Toc384203776)

[Risques 28](#_Toc384203777)

[Non-risques 28](#_Toc384203778)

[Compromis 28](#_Toc384203779)

[Raisonnement 28](#_Toc384203780)

[Conclusion 29](#_Toc384203781)

# Liste des tableaux

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

# Liste des figures

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

# Introduction

# 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**

|  |  |
| --- | --- |
| **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**

|  |  |
| --- | --- |
| **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**

|  |  |
| --- | --- |
| **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**

|  |  |
| --- | --- |
| **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**

|  |  |
| --- | --- |
| **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**

|  |  |
| --- | --- |
| **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**

|  |  |
| --- | --- |
| **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**

|  |  |
| --- | --- |
| **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.

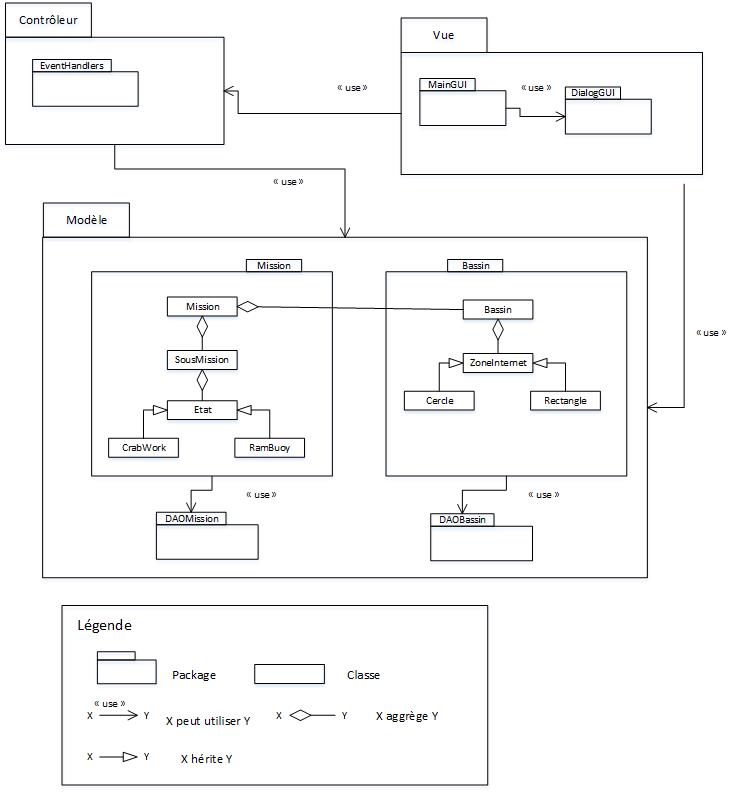
**S15 - Modification**

* **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 :

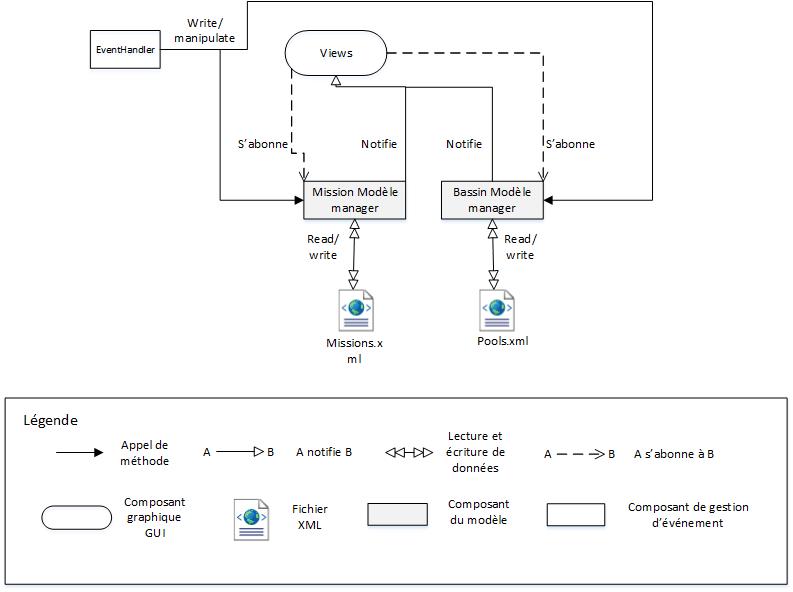
« Sommaire de la justification et des tactiques»



«Insérer description, voir plan. »

### Vue C&C :

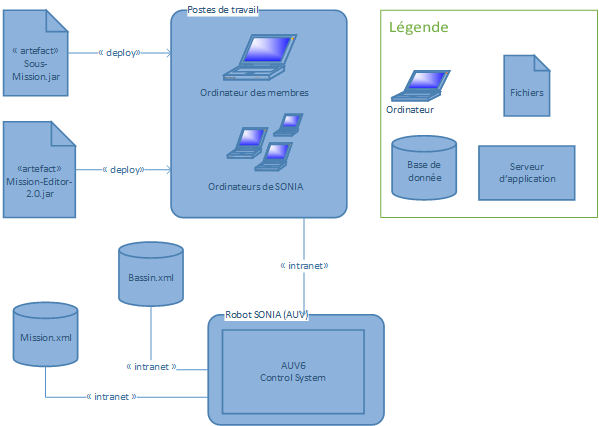
Dans toute application de manipulation de données à partir d’une interface graphique, un facteur très important dans le choix des tactiques est de séparer les données de l’affichage de celle-ci. Étant donné que les données du « Mission Editor 2.0 » sont relativement bien définies, une architecture où 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.

### Vue Allocation

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.

# Analyse ATAM

## Documents sources

Voir section documentation architecturale.

## Règles et standards

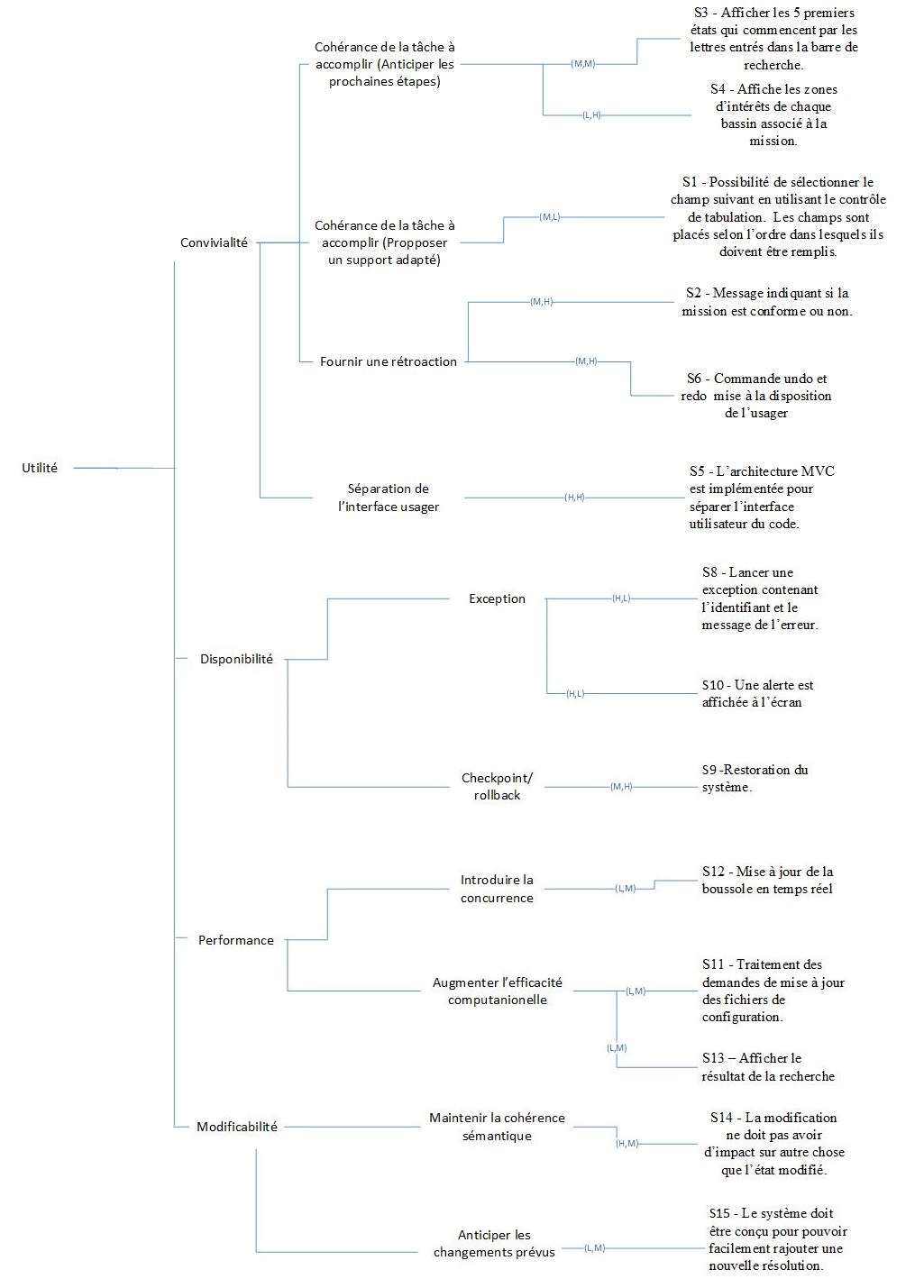
« Description de ATAM »

## 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 dédier au chargement de ses librairies et fournir la liste de sous-mission au système.Arbre d’utilité



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

Note : Réarranger l’arbre d’utilité

## ATAM – Analyse

« 6 scénarios détaillés, incluant références aux 4 tableaux suivants »

### Risques

|  |  |  |
| --- | --- | --- |
| R01 |  |  |
|  |  |  |
|  |  |  |

### Non-risques

|  |  |  |
| --- | --- | --- |
| NR01 |  |  |
|  |  |  |
|  |  |  |

### Compromis

|  |  |  |
| --- | --- | --- |
| C01 |  |  |
|  |  |  |
|  |  |  |

### Raisonnement

|  |  |  |
| --- | --- | --- |
| Ra01 |  |  |
|  |  |  |
|  |  |  |

# Conclusion