

MASTER 2 – ISIDIS

Projet de synthèse

V.A.L.

**Spécifications Fonctionnelles**

Sommaire

[I. Glossaire 3](#_Toc343154453)

[II. Use Case Model 3](#_Toc343154454)

[1) Les acteurs 3](#_Toc343154455)

[2) Le diagramme 4](#_Toc343154456)

[3) Description des Use Cases 5](#_Toc343154457)

[III. Décomposition des Use Cases 6](#_Toc343154458)

[1) Use Case 1 : Collecter les informations terrains 6](#_Toc343154459)

[A. Use Case Detail : 6](#_Toc343154460)

[B. Diagramme de séquence système : 6](#_Toc343154461)

[C. Diagramme de classes participantes : 7](#_Toc343154462)

[D. Conception détaillée : 7](#_Toc343154463)

[2) Use Case 2 : Prioriser messages 8](#_Toc343154464)

[A. Use Case Detail : 8](#_Toc343154465)

[B. Diagramme de séquences système : 9](#_Toc343154466)

[C. Diagramme de classes participantes : 9](#_Toc343154467)

[D. Conception détaillée : 10](#_Toc343154468)

[3) Use Case 3 : Orchestrer les messages 10](#_Toc343154469)

[4) Use Case 4 : Configurer le démarrage des services 10](#_Toc343154470)

[5) Use Case 5 : Transmettre les messages au terrain 10](#_Toc343154471)

[6) Use Case 6 : Effectuer les interventions automatiques 10](#_Toc343154472)

[7) Use Case 7 : Expand messages 10](#_Toc343154473)

1. Glossaire

* RTDG : C’est ce qui représente notre [S.U.D](#SUD), c’est le système qui fait la transmission des messages entre le [RTDRS](#RTDRS) et le [composant embarqué](#composant_embarqué), il envoie aussi des contre-mesures au [composant embarqué](#composant_embarqué).
* RTDRS : C’est le système qui représente le centre de contrôle
* S.U.D : System Under Design, représente notre périmètre applicatif.
* Référentiel : Emplacement où sont stockées les informations dont a besoin notre système.
* Composant terrain : ce sont toutes les puces électroniques implantées sur les rames, sur les rails et sur les stations.
* composants embarqués : Les puces électroniques qui sont implantées sur les rames
* MOM : Message Oriented Middleware, c’est un framework qui permet la communication entre des applications via un réseau informatique, il permet un couplage faible entre les applications.
* ActiveMQ : Une implémentation du JMS (Java Message Service).
* framework de simulation : Système qui simulera le travail que font les [composants embarqués](#composant_embarqué)
* EFIC : [Composant embarqué](#composant_embarqué) qui collecte les messages des [composants embarqués](#composant_embarqué) et l’envoie au [RTDG](#RTDG)
* EHQMR : [Composant embarqué](#composant_embarqué) qui reçoit les messages envoyés par le [RTDG](#RTDG).
* Buffers : Structures où seront stockés les messages reçus des [composants embarqués](#composant_embarqué) selon la criticité.
* Criticité : critère qui nous dit quel message devons-nous traiter en premier lieu.

1. Use Case Model
2. Les acteurs

En lisant le sujet du projet de synthèse et en faisant la conception préliminaire, on a pu trouver 3 acteurs qui interviennent dans notre [S.U.D](#SUD):

* Admin\_RTDG : Cet acteur est celui qui administre notre système [RTDG](#RTDG), son rôle est de démarrer et arrêter le système et de configurer les règles de gestion de ce système, c’est donc à lui que revient la charge de mettre en place les [référentiels](#Réferentiel) qui seront dans la base de données, il peut à tout moment modifier, ajouter ou supprimer des informations sur ces [référentiels](#Réferentiel) tel les adresses MAC de chaque composant ou la contre-mesure à effectuer pour un message donné.
* [RTDRS](#RTDRS) : Cet acteur représente pour nous le centre de contrôle, c’est lui qui nous fournit les informations concernant l’état du trafic, on lui transmet quelques messages qu’on reçoit depuis les [composants terrain](#Composant_terrain), et il nous fournit les contre-mesures à renvoyer au terrain, on lui envoie aussi des notifications des contre-mesures qu’on fait sans le solliciter, la communication avec cet acteur se fait via un [MOM](#mom) appelé [ActiveMQ](#activemq).
* Composant\_Embarqué : Cet acteur représente pour nous les [composants embarqués](#composant_embarqué) contenus dans les rames des métros, ils nous enverront les messages que nous devons traiter, dans notre projet, à défaut d’avoir de vrais composants embarqués, nous réaliserons un [framework de simulation](#framework) qui réalisera ce travail, nous auront alors un composant appelé [EFIC](#efic) qui collecte les informations depuis les capteurs embarqués et les envoie à notre système [RTDG](#RTDG) et un autre composant appelé [EHQMR](#ehqmr) qui réceptionnera les messages qu’on lui enverra sous forme d’ordre pour qu’il les effectue.

1. Le diagramme



1. Description des Use Cases

|  |  |
| --- | --- |
| **Nom** | **Description** |
| Collecter les informations terrains | Cet UC a comme rôle la réception de tout type d'informations terrain à partir des différents [capteurs embarqués](#composant_embarqué) et qui nous seront envoyés par le composant [EFIC](#efic) |
| Prioriser messages | Dans cet UC on gère les messages selon un [référentiel](#Réferentiel), on met chaque message dans le buffer adéquat, sachant qu'on a plusieurs [buffers](#buffer) selon la [criticité](#criticité) du message. Après ceci, on priorisera nos messages pour les traiter selon une stratégie qu'on aura préalablement défini. |
| Orchestrer les messages | Cet UC permet la gestion de la communication des messages provenant des centres de contrôles vers notre système d'information. |
| Configurer le démarrage des services | Cet UC permet de faire un chargement total des informations de la base de données en mémoire pour démarrer notre système d'information, ces informations  seront mises dans des [référentiels](#Réferentiel), tel celui des adresses MAC de chaque composant ou celui des contre-mesures relatives à chaque incident. |
| Transmettre les messages au terrain | Cet UC a comme rôle la transmission des messages vers un [composant embarqué](#composant_embarqué) identifié par son adresse MAC suivant un [référentiel](#Réferentiel). |
| Effectuer les interventions automatiques | Permet de traiter les messages et de définir les contre mesure et sous actions sur les évènements critiques en utilisant un [référentiel](#Réferentiel) qu'on va définir. |
| Expand messages | C’est au niveau de cet UC qu'on transforme un message fonctionnel en un message technique, car le message qu'on obtiendra du côté du [RTDRS](#RTDRS) reste fonctionnel et il nous faudra alors  l'interpréter pour pouvoir savoir quelle contre-mesure y affecter. |

1. Décomposition des Use Cases
2. Use Case 1 : Collecter les informations terrains
3. Use Case Detail :

Acteur principal : Composant\_Embarqué

Acteur secondaire : -

Objectifs :

Cet UC a pour objectif, la récupération des informations depuis les [composants embarqués](#composant_embarqué) et leur enregistrement dans un [buffer](#buffer) pour  pouvoir les traiter par notre système [RTDG](#RTDG).

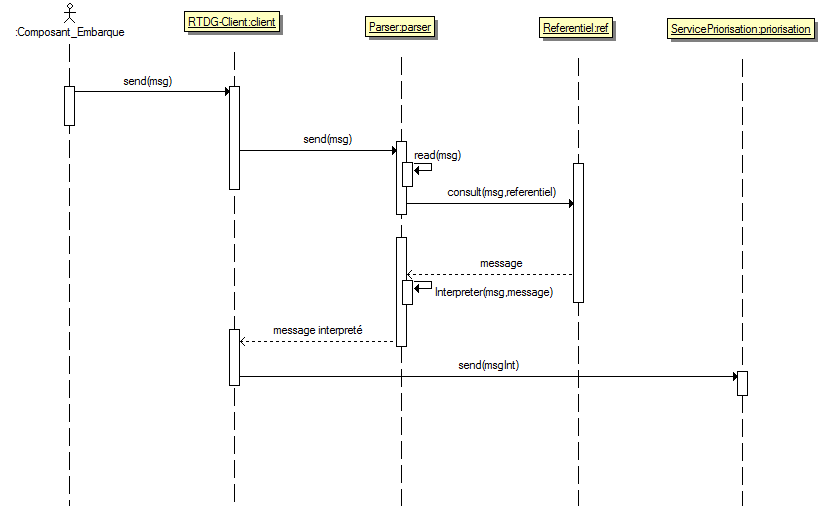
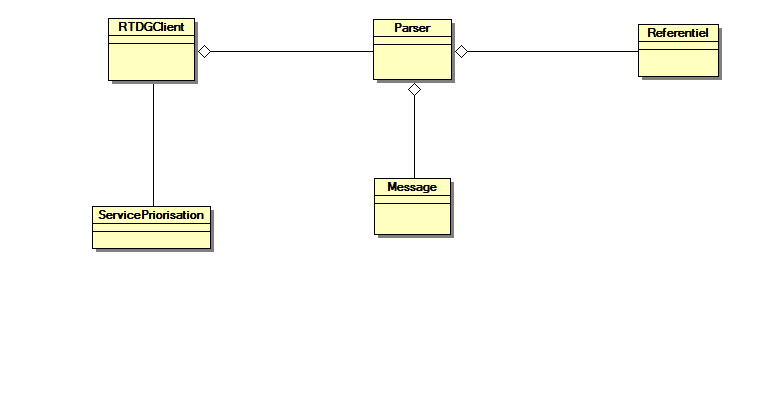
Pré-conditions :

- [Framework de simulation](#framework) doit être démarré

-  [RTDG](#RTDG) doit être en écoute pour pouvoir recevoir les messages depuis les [composants embarqués](#composant_embarqué)

Scénario nominal: S-01017 : Interpréter un message reçu depuis un composant embarqué (parser)

1. le composant embarqué envoie un message à RTDG-Client
2. le RTDG-Client lit le message et consulte le référentiel
3. le RTDG-Client interprète le message
4. le RTDG-Client envoie le message interprété au service de priorisation
5. Diagramme de séquence système :



1. Diagramme de classes métier :
2. Conception détaillée :
   * Diagramme de classes détaillé
   * Diagrmme de séquence détaillé
3. Use Case 2 : Prioriser messages
4. Use Case Detail :

Acteur principal : -

Acteur secondaire : Composant\_Embarqué

Objectifs :

L’objectif de ce Use Case et de prioriser les messages reçus pour pouvoir déterminer lequel devra être traité en premier.

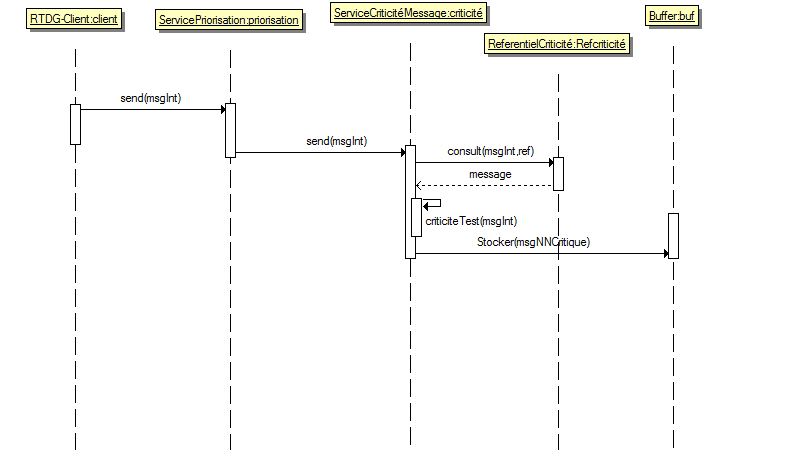
Pré-conditions :

- [Framework de simulation](#framework) doit être démarré

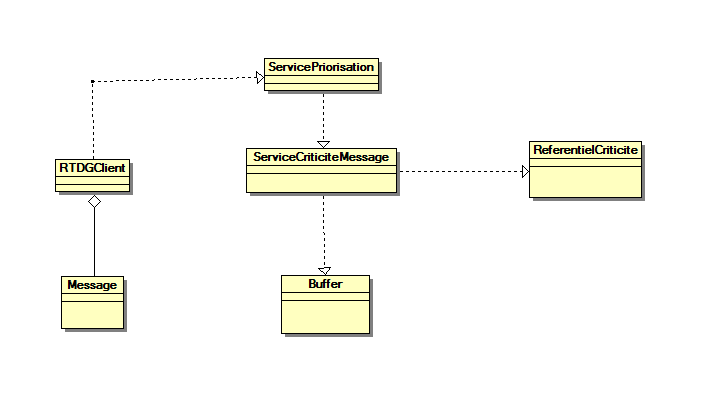
- Les [référentiels](#Réferentiel) doivent avoir été chargés en mémoire

Scénario nominal: S-01020 : affecter un message non critique dans un buffer (nom de scénario à discuter)

1. le service de priorisation reçoit le message interprété depuis RTDG-Client
2. le service de priorisation envoie le message interprété au service de gestion des messages « ServiceCriticiteMessage »
3. Stocker les messages non critique dans un buffer spécifique pour cela
4. Diagramme de séquences système :

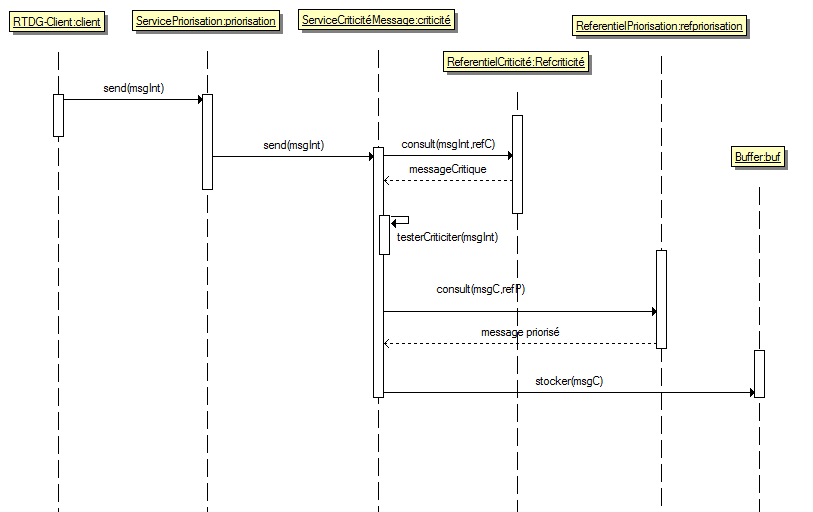


1. Diagramme de classes métiers :

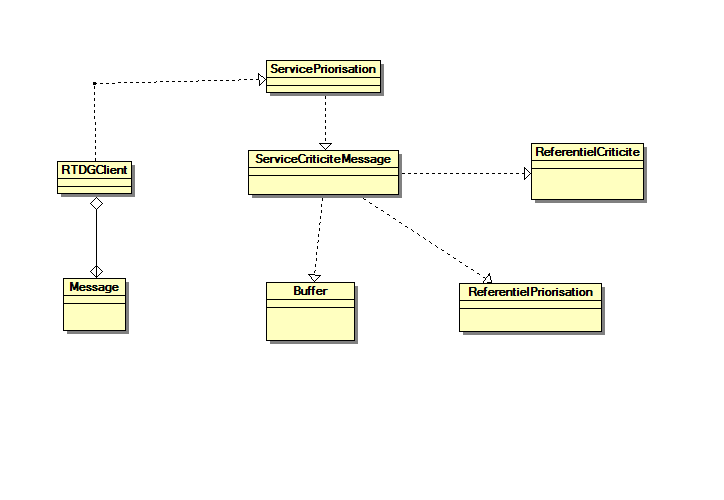


Scénario Alternatif 1: S-01029: Affecter un message critique dans un buffer par ordre de priorité (nom de scénario à discuter)

1. le service de priorisation reçoit le message interprété depuis RTDG-Client
2. le service de priorisation envoie le message interprété au service de gestion des messages « ServiceCriticiteMessage »
3. Prioriser le message selon les référentiels
4. Stocker les messages critique dans un buffer spécifique
5. Diagramme de séquences système :



1. Diagramme de classes métiers :



1. Conception détaillée :
   * Diagramme de classes détaillé :
   * Diagramme de séquences détaillé :
2. Use Case 3 : Orchestrer les messages
3. Use Case 4 : Configurer le démarrage des services
4. Use Case 5 : Transmettre les messages au terrain
5. Use Case 6 : Effectuer les interventions automatiques
6. Use Case 7 : Expand messages