

|  |
| --- |
| SIT213 BE2  IMT Atlantique |
| Chaire M@D - Maintien à domicile | IMT Atlantique |
| 14 septembre  Créé par :  LE DUC Elouan MAQUIN Philippe LE GRUIEC Clément LE JEUNE Matthieu FRAIGNAC Guillaume |

Table des matières

[Intentions du projet 3](#_Toc50977016)

[BE 2 : Test & Validation 4](#_Toc50977017)

[Conclusion 6](#_Toc50977024)

# Intentions du projet

Il s’agit de réaliser, par équipe de 4 ou 5 élèves, une maquette logicielle (en Java) simulant un système de transmission numérique élémentaire. On intégrera donc dans la chaîne un bloc de modulation numérique.

Le système sera assemblé suivant une bibliothèque de modules comportant des ports d’entrée, des ports de sortie et des paramètres physiques. Ces derniers pourront être déterminés à partir des activités du module SIT 212. Le système global sera mis au point progressivement sur 5 séances au cours desquelles les modules seront raffinés, complétés, validés et connectés selon un schéma de transmission de type « point-à-point ».

Outre la qualité technique de la réalisation, on insistera sur les points suivants :

1. La qualité de documentation de la maquette logicielle (notamment la Javadoc).
2. Les efforts de validation des résultats de simulation produits par la maquette.
3. La maîtrise du processus de travail : gestion des versions successives de la maquette logicielle et du dossier technique afférent, synergie de l'équipe, démarche qualité. Concernant ce tout dernier critère, le respect des exigences de mise en forme du livrable sera primordial.

# BE 2 : Test & Validation

Dans le cadre du projet SIT213, vous élaborez un simulateur. L'objet de la présente séance est de réfléchir à la problématique du test et de la validation de ce système, et de choisir une mise en œuvre.

### Quelles parties peut-on vraiment tester avec un programme de test ?

Le programme de test pourra uniquement tester la cohérence des résultats en les comparant avec un modèle théorique. C’est donc le code utile du programme qui sera réellement testé.

### Comment améliorer la testabilité ?

Pour mieux améliorer la testabilité, il est nécessaire de bien décrire dans la documentation les méthodes qui sont utilisées. Ainsi, le programme de test pourra prendre en compte un plus grand nombre de cas d’usage possible.

### Comment faire du test de non-régression à chaque nouvelle itération du projet ?

Vérifier que le nouveau code passe également les anciens tests.

### Quelle confiance accorder aux résultats de simulation, et comment accroître cette confiance ?

Les résultats de la simulation sont un critère nécessaire mais pas suffisant. Si le simulateur respecte le cahier des charges, on peut faire confiance à ses résultats. Mais pour tester si ce code respecte ce dit cahier des charges, il faut le tester avec de nouveaux tests. Ainsi pour augmenter notre confiance dans notre programme il va falloir rajouter de nombreux tests.

### Comment comparer des simulations lorsqu'elles reposent sur des comportements aléatoires (message, bruit, ...) ?

Pour comparer de manière équitable il faut trouver un "seed" qui génère un message aléatoire intéressant.

Il doit posséder tous les changements d'états (ex : 00, 01, 11, 10). Ensuite, il faut réutiliser cette seed pour chaque simulation, ainsi on peut étudier les différences de comportements de la simulation entre les différentes étapes.

### Comment organiser l'équipe spécifiquement pour les tests (p. ex. développeurs et testeurs différents) ?

Les fonctionnalités développées par un membre de l’équipe seront d’abord testées par ce dernier. Dans un second temps, le code devra être testé rigoureusement par un ou plusieurs autres membres de l’équipe. Ainsi, cela imposera une vérification des fonctionnalités développées en plusieurs passes. Dans l’idéal, cette procédure de test devra être réalisée à chaque merge d’une sous branche dans la branche master du projet git.

# Conclusion

blablabla