Projet P2

*Réaliser une animation permettant la résolution d’un système de m équations à n inconnues*

Bastien Burri

Bandelier Matthieu

Gonin Nicolas

Vulliemin Kevin

Équipe 7

2 mars 2015

Table des matières

[1 Introduction 2](#_Toc416703312)

[1.1 Planning 2](#_Toc416703313)

[2 Spécifications des besoins 2](#_Toc416703314)

[2.1 Besoins utilisateurs 2](#_Toc416703315)

[2.1.1 Acteurs 3](#_Toc416703316)

[2.1.2 Cas d’utilisation 3](#_Toc416703317)

[2.2 Scénario 3](#_Toc416703318)

[2.2.1 Séparation 3](#_Toc416703319)

[2.2.2 UC X.X 4](#_Toc416703320)

[2.3 Exigences fonctionnels et non fonctionnels 4](#_Toc416703321)

[2.4 Maquette 4](#_Toc416703322)

[3 Spécifications techniques 5](#_Toc416703323)

[3.1 Analyse du domaine métier 5](#_Toc416703324)

[3.2 Diagramme de classe complet 5](#_Toc416703325)

[3.3 Dynamique de l’application 5](#_Toc416703326)

[3.3.1 UC X.X 5](#_Toc416703327)

[3.4 Architecture de déploiement 5](#_Toc416703328)

[3.5 Choix des librairies 6](#_Toc416703329)

[3.6 Tests fonctionnels 6](#_Toc416703330)

[4 Conclusion 6](#_Toc416703331)

# Introduction

Ce document présente les spécifications des besoins et spécifications techniques du projet « *Réaliser une animation d’un système de m équations à n inconnues*».

L’objectif du projet est de résoudre un système de m équations à n inconnues et de l’illustré à l’aide d’un exemple dans un but pédagogique. L’application sera utilisée pour expliquer la résolution d’un système de m équations à n inconnues à des étudiants de première année ayant des difficultés avec les maths. L’application aura une approche mathématique différente de celles enseignées habituellement en cours, elle contiendra une explication basée sur un problème naturel.

L’utilisateur spécifie à l’application le nombre d’équations ainsi que le nombre d’inconnues. Il devra également spécifier la manière dont il désire que ses équations soient résolues. En effet, ces équations pourront être résolues de deux manières différentes. La 1ère sera une solution ou toutes les étapes de la résolution seront affichées et la 2ème affichera uniquement la solution.

Une fois lancé, le programme résout les équations en fonction des paramètres précédemment saisis. Si les étapes de la résolution sont affichées, alors l’utilisateur aura la possibilité de naviguer entre ces étapes et il y aura une animation sur les lignes.

Les objectifs suivants seront réalisés dans le cadre du projet P2 Java :

* Illustration de la résolution d’un problème naturel à l’aide d’un système d’équations dans un but pédagogique.
* Résolution efficiente avec un algorithme performant sans animation d’un système de *m* équations à *n* inconnus.
* Résolution avec animation d'un système de maximum 5 équations à *n* inconnues.
* Configurations :
  + Entrer le nombre d'équation(s) et le nombre d'inconnue(s) du système d'équations;
  + Résolution étape par étape, avec la possibilité de naviguer entre les étapes, et animations sur les lignes.

## Planning

(screen du planning et mettre la page en paysage si besoin)

# Spécifications des besoins

Ce chapitre est dédié aux besoins du client par rapport au programme voulu. Il doit énumérer les fonctionnalités et service attendu par le programme. Pour ce faire, on commencera par identifier les acteurs sur chaque action, puis on définira ce qu’il peut faire dans notre programme. On pourra alors définir les possibilités des actions en effectuant des scénarios.

## Besoins utilisateurs

La demande du projet était telle que l’utilisateur voulait pouvoir créer des matrices puis résoudre celles-ci dans un cadre pédagogique. L’utilisateur a donc besoin d’avoir une résolution assisté mais afin de rendre ce programme à la fois pédagogique et utile, il a été décidé de créer 2 modes de résolution :

1. Résolution par étape : à des fins pédagogiques, avec affichage de l’historique des opérations ;
2. Résolution directe : à des fins utiles, pour résoudre des équations ou des matrices.

Afin que le programme soit utilisable de manière optimale par un professeur, il est préférable que celui-ci puisse sauvegarder un problème chez lui. Ainsi, celui-ci pourrait préparer son travail à la maison en configurant ses problèmes à sa guise et le présenter le lendemain sans erreur et sans perte de temps.

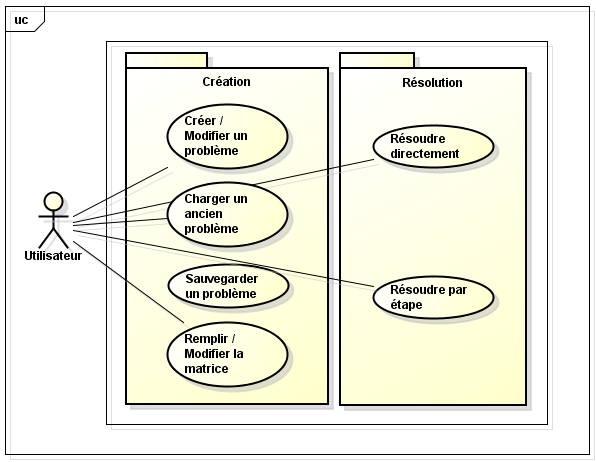
Pour terminer, la configuration du problème et le remplissage de la matrice ont été placée dans des fenêtres annexes qui peuvent être affichées pour modifier un problème, ou modifier les chiffres de la matrice, à tout instant. L’utilisateur peut donc changer entre le mode directe et par étape sans toucher le problème.

### Acteurs

Le seul acteur dans le programme est l’utilisateur courant. Le programme peut être utilisé par un professeur lors d’une présentation mais également par un élève qui veut essayer de comprendre par lui-même les étapes de résolution d’une équation avec le rendu graphique. Néanmoins, même si les utilisateurs sont différents, ils disposent des mêmes fonctions et des mêmes droits car le programme est purement local.

### Cas d’utilisation

Les cas d’utilisation définissent les différentes possibilités que l’utilisateur attend de notre application.



## Scénario

Afin d’identifier le déroulement de chaque cas d’utilisation défini avant, il est possible de faire une sorte de description qu’on appelle scénario. On va enfaite imaginer la suite de communication entre le système, sous forme de boite noire, et les actions utilisateurs. Cela va nous permettre de distinguer toutes les possibilités de l’application.

### Séparation

On peut voir la séparation effectué dans le graphique des cas d’utilisation en package :

* Création : qui va permettre de générer les objets tels que l’équation ou la matrice.
* Résolution : qui permet de résoudre les objets qu’on aura générés ou plutôt d’afficher les solutions.

La description textuelle des packages est présente dans le fichier annexe « Besoins ».

### UC X.X

Ce sous chapitre présente la description textuelle ainsi que les diagrammes de séquence de chaque cas d’utilisation. Le but est de préciser le comportement de chaque cas afin de voir quels sont les possibilités de celles-ci.

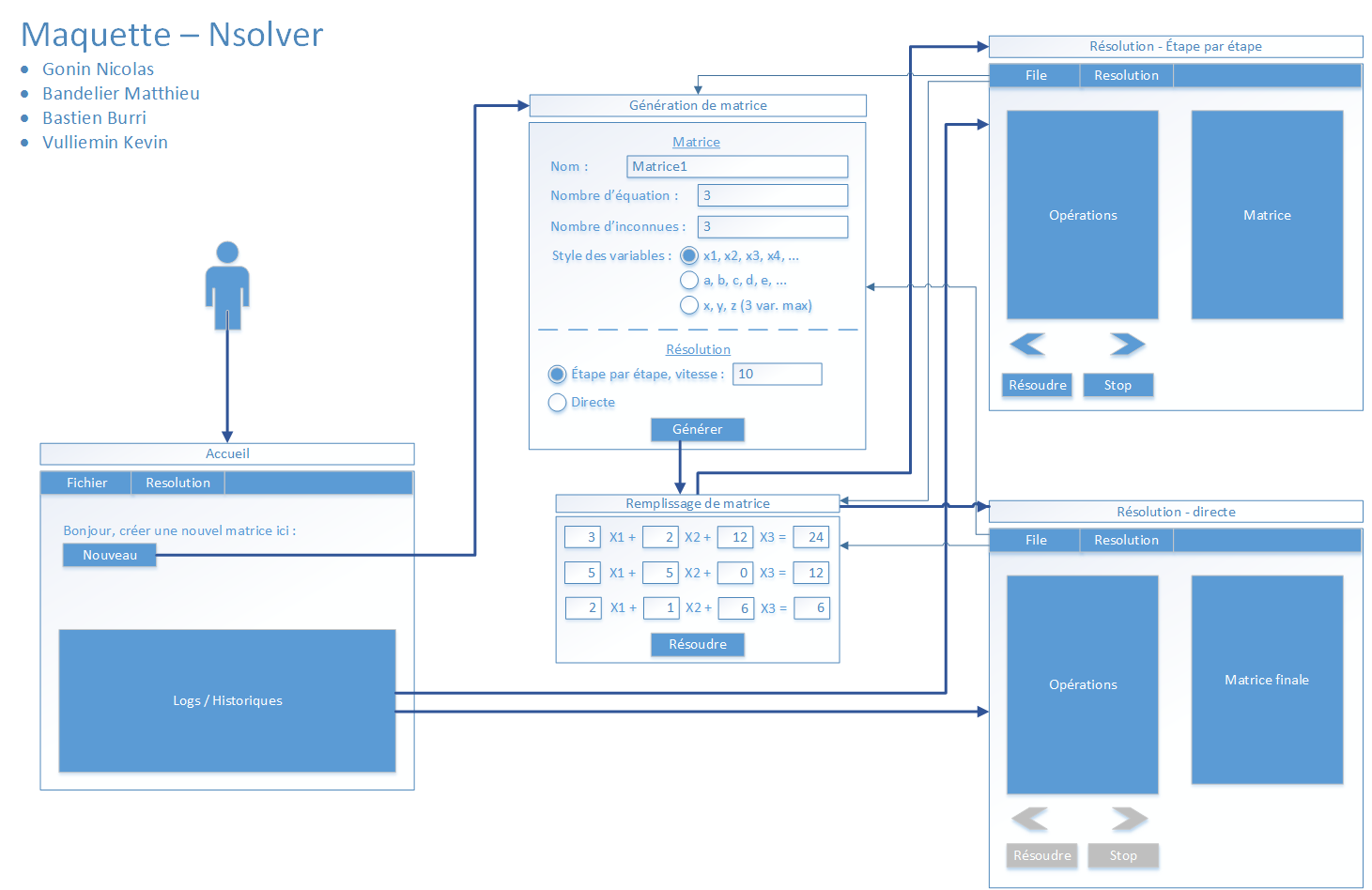
Ceci est disponible dans le fichier annexe « Besoins » au vu de la quantité d’information.

## Exigences fonctionnels et non fonctionnels

Les exigences fonctionnelles fournissent des détails d’implémentation sur les fonctionnalités de l’application. Les exigences non fonctionnelles représentent les facteurs extérieurs imposés à l’application devant être pris en compte lors du développement.

Le tableau des exigences est fourni à la fin du document annexe « Besoins ».

## Maquette



# Spécifications techniques

## Analyse du domaine métier

## Diagramme de classe complet

## Dynamique de l’application

Ce chapitre décrit le comportement dynamique de l’application pour chaque fonction définis dans la partie besoin. Le diagramme de collaboration va permettre de déterminer les objets (les constituants) qui vont être utilisés durant l’action alors que le digramme de séquences détaillés ajoutera une notion de temps et d’ordre dans les liens collaboratifs

### UC X.X

Le diagramme de collaboration de chaque cas d’utilisation ainsi que le diagramme de séquence détaillé sont disponibles dans le document « Dynamique ».

## Architecture de déploiement

Le déploiement de l’application se fera sur des PC de l’école ou des PC personnels. Comme l’application est codée en Java, le terminal sur lequel l’application sera exécutée requiert l’installation d’une machine virtuelle Java si elle n’est pas déjà installée.

L’application sera distribuée sous forme de classes Java compilées et zippées dans des librairies JAR (Java Archive).

Deux packages de déploiement seront distribués :

* Package contenant uniquement les librairies JAR du projet, pouvant être exécutées avec une JVM (Java Virtual Machine) déjà installée sur un PC ;
* Package contenant les librairies JAR du projet et une installation du JRE (Java Runtime Environment) contenant une machine virtuelle Java, pour un PC qui ne serait pas encore équipé de Java.



## Choix des librairies

Pour l'affichage 3D, nous utilisons la librairie externe de java "java3d", qui nécessite *j3dcore-ogl.dll* et les JARs suivants :

* *j3dcore.jar*
* *j3dutils.jar*
* *vecmath.jar*

Pour le look & feel :

* *seaglasslookandfeel-0.2.jar*

Pour la résolution directe, nous avons utilisé la classe *QRDecomposition.java* de la librairie JAMA.  
Nous avons directement inclue cette classe dans notre projet pour permettre sa modification.  
Cette librairie est libre de droit.

## Tests fonctionnels

# Conclusion

Comme il a été expliqué précédemment, le but du projet est de créer un programme de support pour étudier la résolution d’un problème contenant *m* équations à *n* inconnues.

L’utilisateur disposera de deux possibilités pour résoudre ces équations:

* La 1ère  sera une solution ou toutes les étapes de la résolution seront affichées, avec une possibilité de naviguer entre les étapes;
* La 2ème affichera uniquement les solutions.

Comme le programme a premièrement un but pédagogique, les premières itérations de développement porteront essentiellement sur la résolution par étape et la création d’une matrice. Seront ensuite implémentés les fonctions de sauvegardes et de résolution directe.