

Ecole Polytechnique de l'Université de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
polytech.univ-tours.fr

Projet Recherche & Développement 2021-2022

Ordonnanceur 4.0

Ordonnancement des ordres de fabrication



Entreprise
GO SYSTEMES


Tuteur entreprise
Christophe BERTHON

Étudiant
Anthony ADELAIDE (DI5)

Tuteur académique
Yannick KERGOSIEN

Liste des intervenants

Entreprise

GO SYSTEMES
6 Bd Alfred Nobel
37540 Saint-Cyr-sur-Loire, FRANCE
cberthon@gosystemes.com



Nom	Email	Qualité
Anthony ADELAIDE	anthony.adelaide@etu.univ-tours.fr	Étudiant DI5
Yannick KERGOSIEN	yannick.kergosien@univ-tours.fr	Tuteur académique, Département Informatique
Christophe BERTHON	cberthon@gosystemes.com	Tuteur entreprise



Avertissement

Ce document a été rédigé par Anthony ADELAIDE susnommé l'auteur.

L'entreprise GO SYSTEMES est représentée par Christophe BERTHON susnommé le tuteur entreprise.

L'Ecole Polytechnique de l'Université de Tours est représentée par Yannick KERGOSIEN susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Anthony ADELAIDE, *Ordonnanceur 4.0: Ordonnancement des ordres de fabrication*, Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France, 2021-2022.

```
@mastersthesis{
  author={ADELAIDE, Anthony},
  title={Ordonnanceur 4.0: Ordonnancement des ordres de fabrication},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université de Tours},
  address={Tours, France},
  year={2021-2022}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	iv
1 Introduction	1
1 Acteurs, enjeux et contexte	1
2 Objectifs	1
3 Bases méthodologiques	2
2 Description générale	3
1 Environnement du projet	3
2 Caractéristiques des utilisateurs	3
3 Fonctionnalités du système	4
4 Structure générale du système	5
3 État de l'art / Veille technologique	7
1 Recherche opérationnelle	7
2 Définition	7
3 Problèmes d'ordonnancement classiques	7
4 Catégorie de machine (Ordonnancement d'atelier)	8
5 Notation de Graham	8
4 Analyse et conception	10
1 Analyse	10
1.1 Description du problème	10

1.2	Contraintes.....	12
1.3	Spécifications.....	12
2	Modélisation proposée.....	12
5	Mise en oeuvre	14
1	Outils et librairie utilisés.....	14
2	Déviation et choix d'implémentation	15
3	Analyse des résultats, évaluation, qualité	16
4	Principales IHM.....	16
6	Bilan et conclusion	19
1	Bilan du semestre 9	19
2	Bilan du semestre 10.....	19
3	Bilan sur la qualité	21
4	Bilan auto-critique.....	22
	Annexes	23
A	Planification, gestion de projet	24
1	Evolution du projet	24
B	Cahier de Spécifications	26
1	spécifications Fonctionnelles.....	26
1.1	Définition de la fonction 1 : Trie des OFs en fonction de la date au plus tôt	26
	Description de la fonction 1 :.....	26
1.2	Définition de la fonction 2 : Recherche des ressources.....	26
	Description de la fonction 2 :	26
1.3	Définition de la fonction 3 : Planification des ressources.....	26
	Description de la fonction 3 :	26
1.4	Définition de la fonction 4 : Importation de données	26
	Description de la fonction 4 :	26
1.5	Définition de la fonction 5 : Exportation des plannings	27
	Description de la fonction 5 :	27
2	Spécifications non fonctionnelles	27
2.1	Contraintes de développement et conception	27
2.2	Contraintes de fonctionnement et d'exploitation.....	27
2.2.1	Performances.....	27
2.2.2	Contrôlabilité	27
2.2.3	Sécurité.....	27

C	Cahier du développeur	28
1	Introduction	28
2	Diagrammes architecturaux et UML	29
3	Descriptions détaillées de données exploitées	31
4	Descriptions détaillées des classes, modules, réalisations	32
D	Document d'installation	33
1	Installation	33
E	Document d'utilisation	34
1	Utilisation du logiciel	34
F	Cahier de test	35
1	Tests unitaires	35

Table des figures

1	Introduction	
1.1	Méthode de gestion en cascade.....	2
2	Description générale	
2.1	Fonctionnalités du logiciel.....	4
2.2	Diagramme d'utilisation.....	5
2.3	Diagramme de classe et Diagramme d'utilisation.....	5
4	Analyse et conception	
4.1	Diagramme de bloc.....	13
5	Mise en oeuvre	
5.1	Intégration Continue.....	14
5.2	SonarCloud.....	15
5.3	Association des étapes aux consommables.....	16
5.4	Exécution sur jeu de données de 100 Etapes.....	17
5.5	Menu Principal.....	17
5.6	Menu de lancement de l'algorithme.....	18
6	Bilan et conclusion	
6.1	Planning Opérateur.....	21
A	Planification, gestion de projet	
A.1	Le diagramme de Gantt Initial.....	24
A.2	Le diagramme de Gantt Final.....	24
A.3	Tableau d'action.....	25

C Cahier du développeur

C.1 Package Algorithms	29
C.2 Package Checker	29
C.3 Package Data	30
C.4 Package Solution	30
C.5 Package Parser	31
C.6 Package View	31
C.7 Planning de sortie d'algorithme.....	32

1

Introduction

1 Acteurs, enjeux et contexte

GOSYSTEMES est une PME qui propose des solutions de gestion dédiée à l'industrie Chimie-Peinture. Plus particulièrement, les progiciels propose d'aider les utilisateurs dans l'administration des stocks, de la production, etc...

Dans le domaine de la production de peinture, il est nécessaire de planifier à l'avance l'ordre de passage des ordres de fabrication (OF) représentant les commandes de clients.

Généralement, cet ordonnancement est fait à la main sur un horizon de planification d'un à plusieurs jours, par une personne qui va concilier tous les paramètres et contraintes à prendre en compte (disponibilités des machines/opérateurs, exigences des clients, stock des ressources consommables, etc.).

La construction de cet ordonnancement prend du temps et si un élément imprévu survient (panne d'une machine, grève...), il faut tout replanifier. C'est dans ce contexte que l'expression d'un besoin d'un outil d'aide à la planification a été soulevée.

Acteurs :

- Client : M. Christophe BERTHON
- MOA : M. Yannick KERGOSIEN
- MOE : Anthony ADELAIDE

2 Objectifs

La société GOSystemes souhaite réaliser un nouvel outil qui travaillera sur les ordres de fabrication de peinture et fournira un document qui contiendra l'ordonnancement la plus optimale de celles-ci.

Cet outil va générer un planning pour les opérateurs, les machines et les ordres de fabrication sur un horizon de planification. Outre la planification via l'ordonnanceur, on souhaite également ajouter la possibilité d'établir ce planning selon un choix de fonction objectif que sélectionnera l'utilisateur.

3 Bases méthodologiques

Les outils :

- Trello : Un outil de gestion de projet qui offre la possibilité de séparer un projet en différentes tâches sous forme de carte. Il est possible de les attribuer à une personne et d'ajouter des deadlines et toutes autres informations utiles.
- Mail : Moyen de communication entre les différents membres du projet.
- Github : Il s'agit d'un gestionnaire de versioning qui va permettre à un développeur de sauvegarder le code qu'il produit.
- GitHub Action : Outil d'intégration continue qui permet d'informer le développeur en cas de problème avec le code rendu en ligne.
- SonarCloud / SonarLint : Complément de l'outil d'intégration continue qui permet au développeur de voir la qualité du code rendu et autres métriques.

La méthode de gestion de projet : On utilise le modèle en « cascade » (Figure 1.1) pour gérer ce projet. Les phases de modèle en « cascade » sont effectuées par l'ordre de haut en bas. Chaque étage, on retourne à l'étape précédente afin de le faire valider. Ce modèle correspondant donc à ce projet car il faut répéter les cycles de conceptions et codages dus aux différents algorithmes à développer.

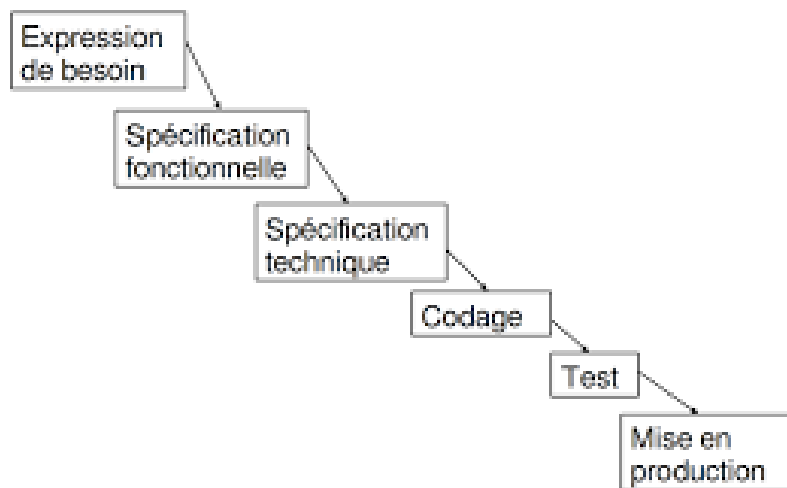


Figure 1.1 – Méthode de gestion en cascade

2

Description générale

1 Environnement du projet

L'environnement de développement consiste aux points suivants :

- Le système d'exploitation : Windows ;
- Le langage de programmation : dotnet ;
- L'IDE pour le développement : Visual Studio ;

2 Caractéristiques des utilisateurs

Utilisateur			
Type d'utilisateur or Champs	Connaissance de l'informatique	Expérience de l'application	Droits d'accès
Opérateur	Basique	Aucune	Tous les droits

3 Fonctionnalités du système

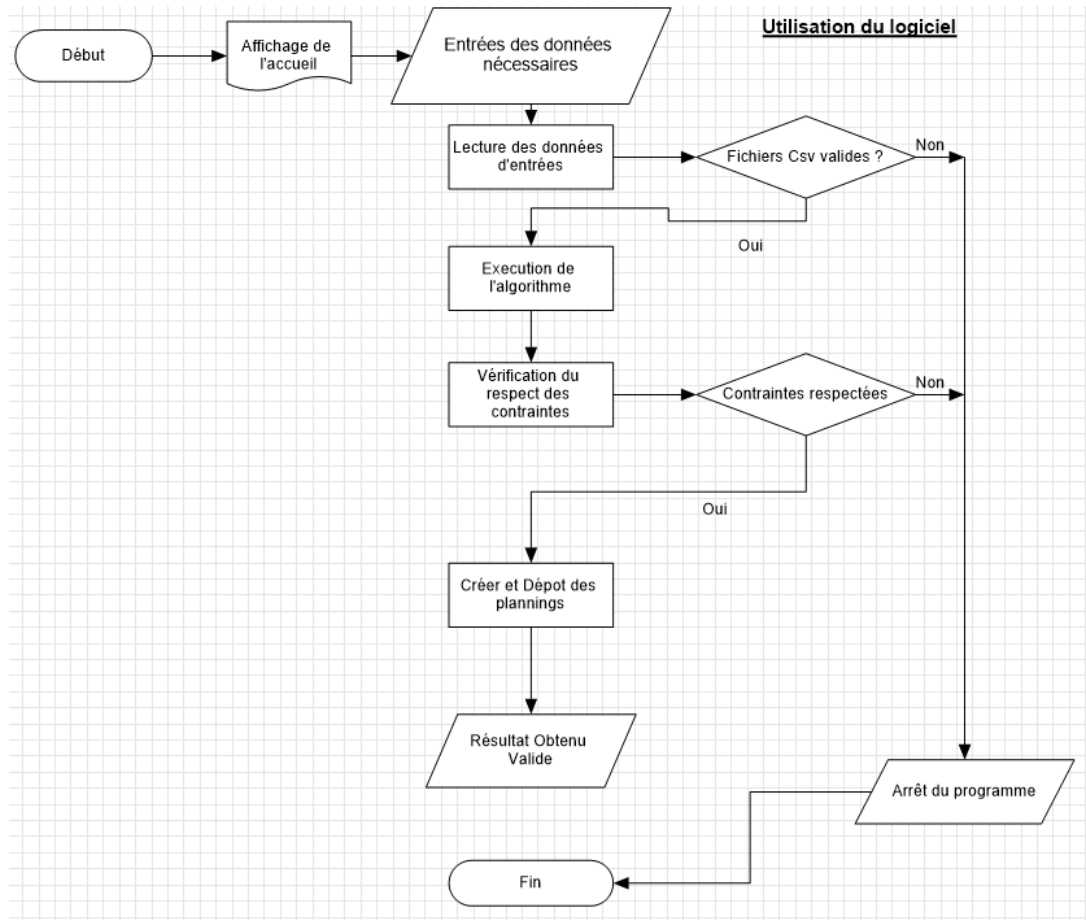


Figure 2.1 – Fonctionnalités du logiciel

L'utilisateur du logiciel doit insérer des données dans l'interface graphique telles que la localisation des données sources afin que l'algorithme puisse fonctionner.

4 Structure générale du système

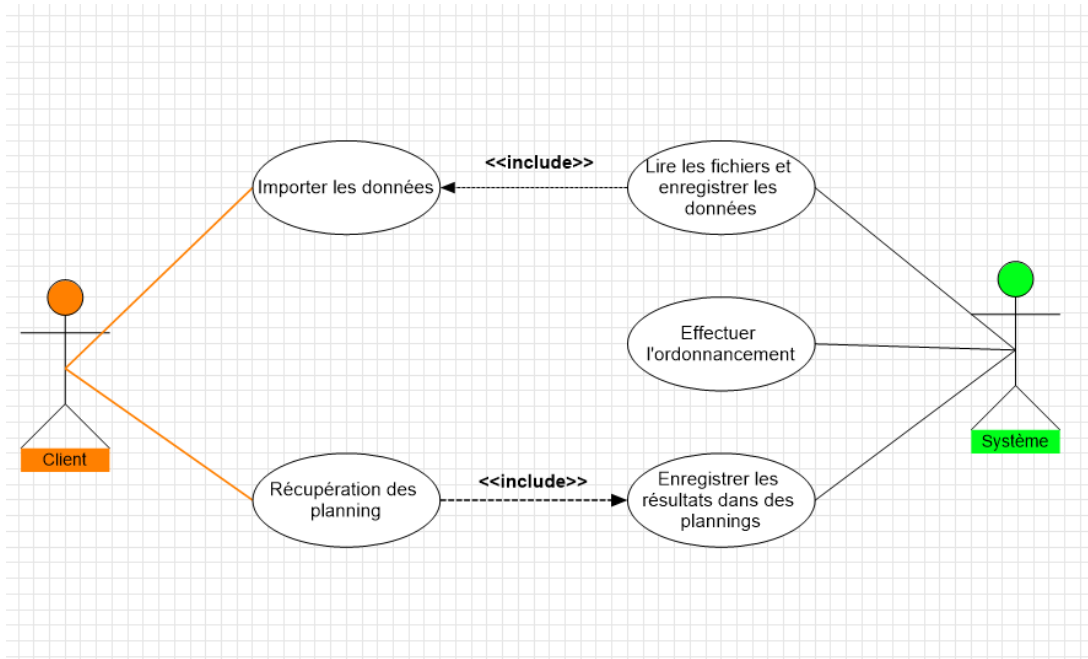


Figure 2.2 – Diagramme d'utilisation

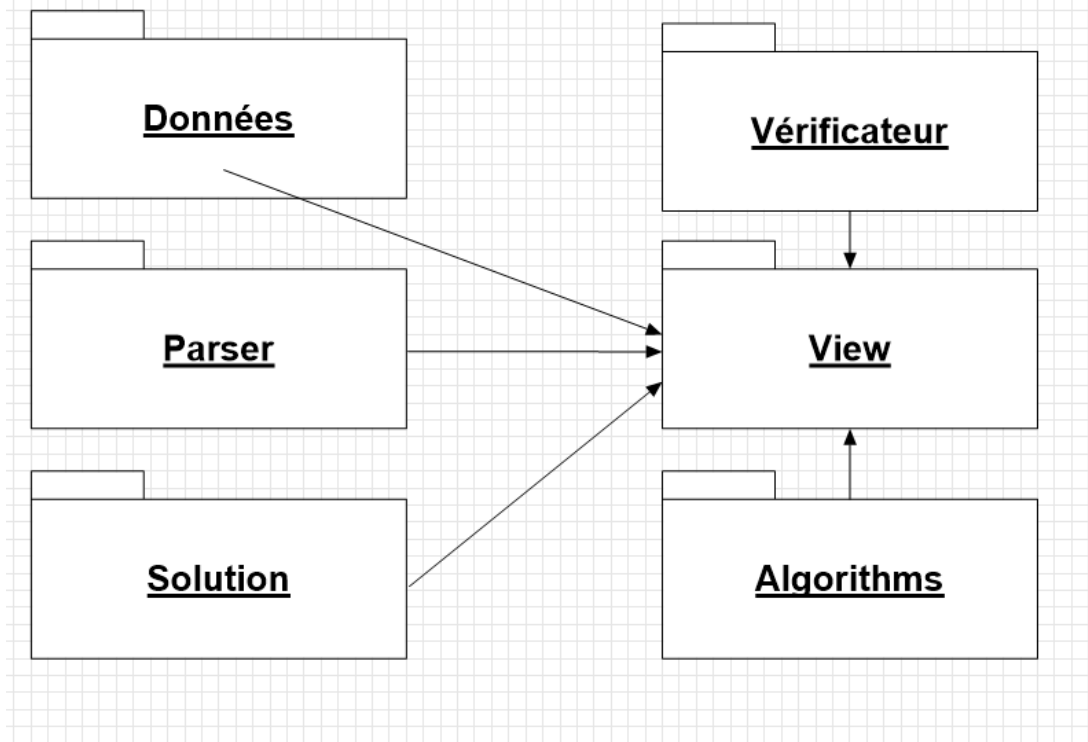
Diagramme de classe
UML

Figure 2.3 – Diagramme de classe et Diagramme d'utilisation

Le diagramme de classe présenté ci-dessus contient les différents modules que possède le logiciel qui seront expliqué ci-dessous. Le détail de la conception du logiciel sera expliqué dans le cahier du développeur.

Le Module Données réfère au module qui contient toutes les classes contenant les données du logiciel.

Le module Parseur contient les méthodes nécessaires à la récupération et transformation des données dans un langage que l'algorithme peut exploiter. Il permet à la fois de traduire et transformer ces données en fichiers Csv.

Le module Vérificateur contient les éléments capable de déterminer si la solution conçu par l'algorithme est réalisable. C'est à dire qu'elle va vérifier le respect de chaque contrainte de chaque ressource.

Le module Solution contient les données qui seront traduits sous forme de plannings(fichier Csv)

Le module View contient l'interface graphique que l'utilisateur va interagir au cours de l'utilisation du logiciel.

3

État de l'art / Veille technologique

Dans ce chapitre, nous allons identifier les différents problèmes d'ordonnancement classiques et comprendre comment les représenter sous une forme de notation simple.

1 Recherche opérationnelle

L'objectif de ce chapitre est de donner une vue d'ensemble de ce que l'on appelle Recherche Opérationnelle dans le monde académique et industriel. Il ne s'agit que d'une vision partielle et certainement subjective, pour plus de détails, voir des ouvrages plus exhaustifs.

2 Définition

On peut définir la Recherche Opérationnelle (RO) comme l'ensemble des domaines scientifiques, des outils et des problèmes touchant aux questions d'ordre décisionnel (dit aussi stratégique) ou d'optimisation de systèmes complexes.

L'expression systèmes complexes est à prendre ici au sens le plus littéral du terme, c'est-à-dire difficile à comprendre pour un individu sans l'aide d'un modèle ou d'un ordinateur. Les problèmes sont rendus complexes par leur dimension qui peut être importante, mais surtout par leur structure qui peut-être par exemple combinatoire, concurrentielle, stochastique etc. On peut citer en exemple pertinent : la recherche d'un itinéraire sur une carte, l'ordonnancement des tâches à accomplir en usine, la décision stratégique d'investissement d'une entreprise sur un marché concurrentiel, ...

3 Problèmes d'ordonnancement classiques

Problème de gestion de projet à contraintes de ressources et sa représentation graphique, le réseau PERT

Problème d'ordonnancement d'atelier

- Flow-shop
- Job-shop
- Open-shop

Problèmes d'ordonnancement dans les systèmes d'exploitation

Problèmes d'ordonnancement de tâches informatiques

4 Catégorie de machine (Ordonnancement d'atelier)

Machine Unique :

Dans ce cas, l'ensemble des tâches à réaliser est fait par une seule machine. Les tâches alors sont composées d'une seule opération qui nécessite la même machine. L'une des situations intéressantes où on peut rencontrer ce genre de configurations est le cas où on est devant un système de production comprenant une machine goulot qui influence l'ensemble du processus. L'étude peut alors être restreinte à l'étude de cette machine.

Machines parallèles :

Dans ce cas, on dispose d'un ensemble de machines identiques pour réaliser les travaux. Les travaux se composent d'une seule opération et un travail exige une seule machine. L'ordonnancement s'effectue en deux phases : la première phase consiste à affecter les travaux aux machines et la deuxième phase consiste à établir la séquence de réalisation sur chaque machine.

5 Notation de Graham

Avant de vous montrer des algorithmes pour différents problèmes d'ordonnancement, voici une notation pour ces problèmes qui va nous être utile. Elle a été introduite pour pouvoir classer plus facile la multitude de variantes de problèmes d'ordonnancement qui existent. C'est la notation $A|B|Y$.

A représente l'environnement d'exécution. Par exemple 1 est pour une machine, P3 pour trois machines parallèles identiques, Q pour m machines de vitesses différentes (m et les vitesses font alors partie de l'entrée du problème), etc.

B représente les contraintes sur les tâches. Par exemple prec est pour une contrainte de précédence donnée, ri veut dire que chaque tâche i est muni d'une date de relâchement ri donnée avant laquelle elle n'est pas disponible pour exécution. Le mot pmtn — pour préemption — veut dire que chaque tâche i peut être exécutée dans plusieurs intervalles séparés de durée totale pi. Dans le cas d'un ordonnancement sur plusieurs machines ces intervalles doivent être disjoints : une même tâche ne peut être exécutée à tout moment que sur une seule machine.

Y représente la fonction objective à minimiser. Par exemple Cmax demande à minimiser le makespan, PCi veut minimiser la somme des dates de terminaisons de toutes les tâches, aussi appelé flowtime en anglais.

Notation de Graham pour notre problème : $J, Px ; 0, prec, rj, dj, 0, no-wait ; Lmax$

A permet de spécifier l'environnement machine : $A = A1;A2$. $A1 = O$ pour l'open-shop, F pour le flow-shop et J pour le job-shop. $A2$ correspond au nombre de machines, o si ce nombre n'est pas fixé. B décrit les caractéristiques des tâches : $B = B1;B2;B3;B4;B5;B6$. $B1 = pmtn$ dans le cas d'un problème préemptif, o sinon. $B2 = prec$ si des contraintes de précédence existent entre les lots, tree si le graphe de précédences est une arborescence, o s'il n'existe pas de contraintes de précédence. $B3 = rj$ si des dates de disponibilité sont associées aux tâches, o sinon. $B4 = dj$ si des échéances sont associées aux tâches, o sinon. $B5 = (pj = p)$ si les tâches ont des durées identiques, $(pj = 1)$ si les tâches ont des durées unitaires, o sinon. $B6 = no-wait$ pour les problèmes d'atelier sans attente, o sinon.

$Y = Lmax$ si l'on veut minimiser le retard algébrique maximal.

4

Analyse et conception

1 Analyse

Dans ce chapitre, nous allons identifier les différentes ressources liés à notre problème d'ordonnancement, ensuite lister les contraintes liés à ces dernières, après nous allons déterminer le type d'algorithme approprié afin de résoudre ce problème et enfin proposer une modélisation du logiciel à développer.

1.1 Description du problème

Le problème consiste à planifier un ensemble d'OF sur un horizon de planification de plusieurs jours à définir par l'utilisateur. Cet outil va générer un planning pour les opérateurs, les machines et les ordres de fabrication sur un horizon de planification.

Nous décrirons dans cette section, la liste de toutes les données à prendre en compte afin de trouver une solution pertinente au problème d'ordonnancement.

Chaque OF est caractérisé par :

- Un identifiant : numéro permettant de distinguer les OF.
- Une date de début de production au plus tôt : date à laquelle la production de l'OF peut commencer.
- Une date de fin de production au plus tard : date pour laquelle la production de l'OF doit être terminée. Selon l'objectif choisi, un retard de production sera possible ou non.
- Une quantité à produire : cette information permettra de sélectionner la bonne cuve pour la production. La quantité ne variera pas de la première à la dernière étape.
- Un statut : valeur binaire qui indique si l'OF est en cours (1) ou non (0). Un OF en cours signifie qu'au moment de la planification, un certain nombre d'étape a déjà été réalisé.
- Une liste ordonnée d'étapes : à réaliser pour produire (ou finir de produire si statut=1) à respecter.
- Une heure de début de production au plus tôt : dans le cas où statut=1 qui indique quand l'OF en cours de production termine son étape courante.
- Une liste de types de ressources consommables et leurs quantités : cette liste définit l'ensemble des ressources consommables nécessaires sur toutes les étapes pour la production de l'OF.
- Un produit : cette information nous permettra d'identifier la durée de nettoyage de la cuve.
- Une cuve : ressource qui contient le produit pendant tout l'OF.

Une étape de production concerne un OF et indique les ressources nécessaires pour la réaliser ainsi que des durées de mobilisation des ressources. Chaque étape est caractérisée par :

- Un identifiant : afin de distinguer les différentes étapes.
- Un type de machine : le type de machine nécessaire à la réalisation de l'étape
- Un type d'opération : le type d'opération effectuée dans l'étape (mélanger, écraser, etc...).
- Une durée totale : elle indique le temps de mobilisation de la machine de toute l'étape.
- Une durée de présence de l'opérateur au début de l'étape : temps durant lequel l'opérateur doit être présent sur la machine pour lancer l'étape.
- Une durée de présence de l'opérateur à la fin de l'étape : temps durant lequel l'opérateur doit être présent sur la machine pour finaliser l'étape.
- Une durée maximum d'attente pour la prochaine étape : temps maximum qui peut s'écouler entre cette étape et l'étape suivante.
- Une production à la journée ou sur deux jours (0 ou 1) : Cette information permet d'indiquer si cette étape peut être reportée au lendemain. A noter qu'une étape dans la liste des étapes à réaliser pour un OF peut être reportée.

Chacune de ces étapes sont associés à des opérateurs qui participent au début et/ou à la fin de l'opération. Le début et la fin d'une même étape peut être affecté à un opérateur différent. Un même opérateur peut intervenir pour plusieurs étapes différentes tant qu'il possède les compétences requises. Ces opérateurs sont caractérisés par :

- Un identifiant de l'opérateur : numéro permettant de distinguer les opérateurs.
- Une liste de type de machines : elle indique quel type de machines l'opérateur est compétant pour intervenir.
- Une liste de type d'opérations : elle indique quel type d'opérations l'opérateur peut réaliser.
- Un horaire de travail : chaque opérateur possède une heure de début et de fin de travail.

Ces opérateurs travaillent sur des machines qui effectuent le processus des étapes. Chaque machine est caractérisée par :

- Un identifiant : numéro permettant de distinguer la machine.
- Un type : numéro associé au type de machine (broyeur, disperseur, etc...)
- Un calendrier d'indisponibilité jour et heure : une liste de de jour et de plage horaire (heure de début et heure de fin) durant laquelle la machine n'est pas disponible (pour cause de pannes, de maintenances, ou de réalisation d'OF en cours).
- Une durée de nettoyage : temps nécessaire du nettoyage de la machine. Chaque OF possède une cuve qui lui associé pendant toute la durée de production.

Cette cuve est caractérisée par :

- Un identifiant : numéro permettant de distinguer les différentes cuves.
- Une Matrice des durées de nettoyage : définit la durée de nettoyage entre deux OFs. Cette durée est le temps minimum à respecter entre deux OFs utilisant une même cuve. La durée maximale de cette matrice sera le temps de nettoyage par défaut en dans le cas où la cuve ne sera pas utilisé plus tard.

Un OF peut être ordonnancé sur une journée que si toutes les ressources consommables nécessaires pour sa réalisation sont disponibles. Chaque type de ressource consommable est caractérisé par :

- Un identifiant : numéro permettant de distinguer les types de ressources consommables.
- Une quantité en stock : représente la quantité actuellement en stock.
- Un délai de réapprovisionnement : nombre de jours minimum avant une livraison de cette ressource. Sur un horizon de planification plus grand que ce délai, on peut considérer qu'à partir de ce nombre de jours minimum, la quantité en stock est infinie (car elle peut être commandée).
- Un calendrier d'approvisionnement prévisionnels : indique un ensemble de jour et de quantité de produit réceptionnée et intégrée au stock en début de journée.

1.2 Contraintes

Une démarche qualité logiciel sera à effectuer durant toute la durée du projet. Le logiciel devra être capable de fonctionner sans être maintenu et facilement améliorable.

Un opérateur ne peut pas effectuer de tâche en simultanée. Les consommables ne peuvent pas avoir de nombre négatif.

La langue principale du programme et des documents liés sera le français.

Il n'y a pas de contrainte (architecture) en dehors de celles découlant des besoins exprimés ci-dessus. Le langage utilisé sera le dotnet. Il s'agit d'un langage polyvalent possédant de bonnes performances connu par l'étudiant. Les temps de résolution de l'outil doivent être raisonnables.

1.3 Spécifications

Vous trouverez les spécification en ANNEXE

2 Modélisation proposée

Aux vus des différentes contraintes et ressources liés à ce problème, nous avons conclu qu'il s'agissait d'un problème Jobshop. C'est à dire que nous possédons un nombre n de machine, une tâche ne peut commencer que si la précédente est terminée et une machine ne peut faire qu'une tâche à la fin.

Nous pouvons qualifier notre problème de la notation suivante : Notation de Graham pour notre problème : $J, P_x; 0, prec, r_j, d_j, 0, no-wait; L_{max}$

A l'aide de l'encadrant, on a pu modéliser très simplement sous forme algorithmique le problème.

```

1 NbCteMaxViolé <- 0
2 OF par OF, dans l'ordre du trie :
3   d <- dit; t <- ou hi (cas ou OF en cours)
4   Etape par Etape de l'OF actuelle :
5     (d', t', Ensemble de ressource) <- RechercheRessource(d, t, etape)
6     Si (t' - t) > Maxh alors NbCteMaxViolé++
7     Planifier : l'étape courante de l'OF à t e jour d et
8     MAJ de la solution
9     t <- t' Pah+Ph+Paph
10
11   MAJ opération nettoyage cuve

```

Ce type d'algorithme ne remet jamais en cause les actions qui ont été faite. C'est a dire qu'une fois que l'algorithme planifier un OF, on ne peut plus modifier ce planning.

Afin d'augmenter les performances de l'algorithme, il a été décider de faire des heuristiques afin de trie les OFs de façon optimale et aider l'algorithme à trouver les meilleurs solutions.

Ci-dessous vous trouverez un diagramme de bloc qui résume la modélisation du logiciel.

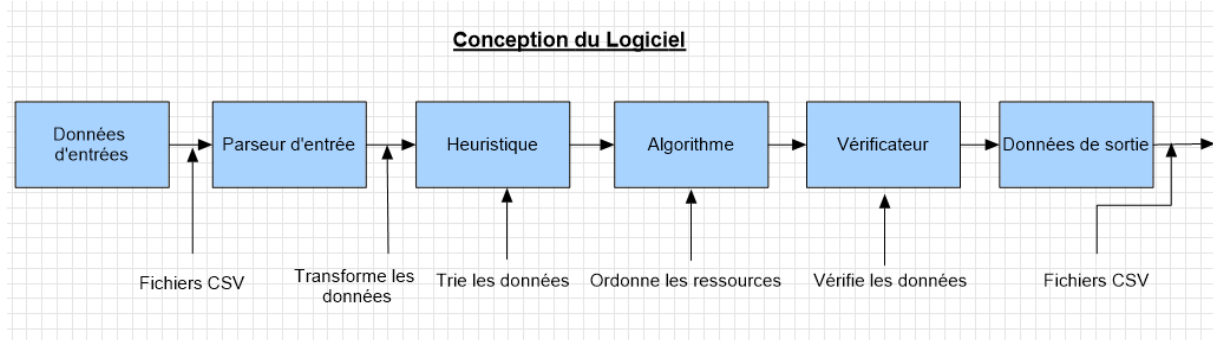


Figure 4.1 – *Diagramme de bloc*

5

Mise en oeuvre

Ce chapitre va couvrir les livrables fourni par l'étudiant ainsi que les différents éléments concernant les résultat obtenus.

le cahier de développement donne plus de détails en ANNEXE

1 Outils et librairie utilisés

Le livrable fourni sera sous la forme d'un executable. Il a donc aucune installation ou de dépendences à télécharger avec le logiciel. Cependant, pour un développeur qui souhaite continue le projet, il faudra qu'il utilise les librairies(intégré de base dans le projet) :

- NUnit.Framework(librairie de test)
- using System.Collections.Generic
- using System.Diagnostics ;
- using System.Windows.Forms ;
- using System.Globalization ;
- using CsvHelper ;

Pour ce projet, j'ai utilisé de l'intégration continue afin de garder le contrôle sur le dépôt de code. C'est à dire que le dépôt Github ou ce situe le code du logiciel est contrôler par la CI/CD qui va executer les tests unitaires présent dans le code et interpellé le développeur en cas de probleme. De plus, il vérifie le fonctionne du logiciel dans un environnement Windows qui sera l'environnement final du logiciel.

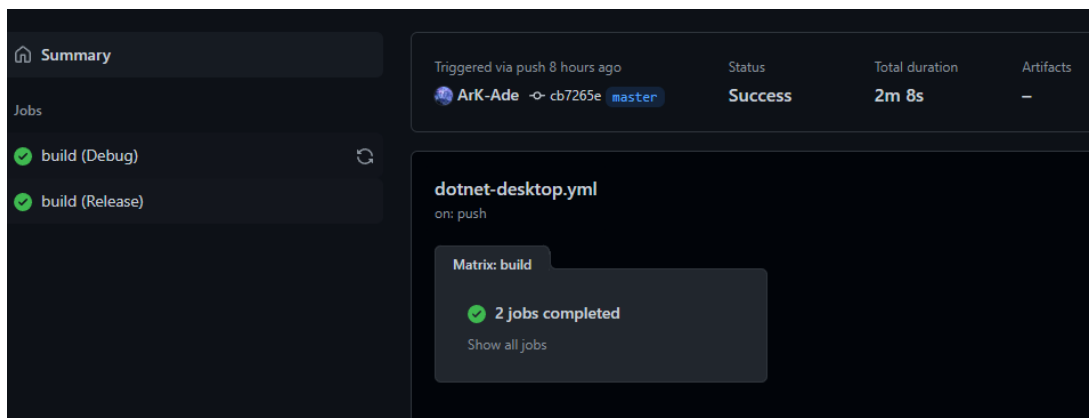


Figure 5.1 – Intégration Continue

De plus a ce système d'intégration continue, il a été rajouter un module de contrôle Qualité. A l'aide de SonarCloud, l'étudiant à pu analyser en profondeur les différents points telles que la qualité, la maintenabilité du code à chaque instant.

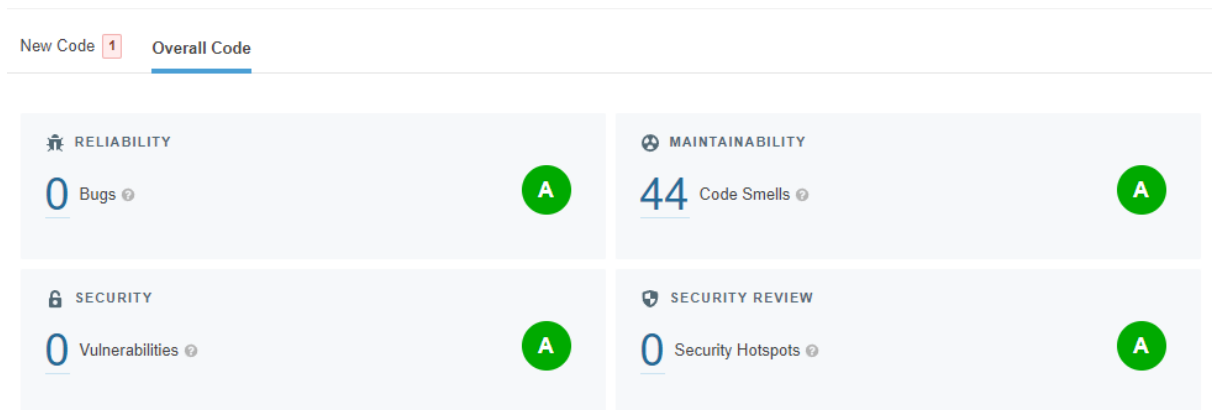


Figure 5.2 – SonarCloud

2 Déviation et choix d'implémentation

// Données

Afin que l'algorithme puisse être tester et corriger, il est nécessaire de lui fournir des données qui correspond à la modélisation proposée. Cependant, du à un manque de données crucial, il a été décider par l'étudiant de modifier les données reçu par le client afin de pouvoir finalisé l'algorithme et le parseur.

Les données totalement manquantes sont listés ci-dessous

- Informations sur les opérateurs
- Informations sur les machines
- Informations sur les cuves

Au vu du manque de données, j'ai modifier les données afin d'avoir un jeu de données valide pour une exécution

Dans l'exemple suivant, nous avons attribué aux différents composants des étapes qui constituent des OFs.

// Interface graphique

Au départ, il avait été décider que l'algorithme serait une application console Windows. Cependant, l'étudiant, en ayant analyser les besoins de l'algorithme, a décider qu'il serait judicieux d'ajouter une interface graphique qui non seulement sera plus facile d'utilisation mais permettrait d'integrer les données d'entrées plus facilement. Voir IHM

// Choix de librairie CsvHelper

Il a fallut choisir une librairie afin de pouvoir transformer des fichiers de données en données que l'algorithme puissent exploiter. La librairie CsvHelper a été choisi pour les raisons suivantes :

- Facilité d'utilisation
- Librairie de base de visual studio
- Capable de lire et écrire des fichiers CSV

// Algorithme L'algorithme proposé réponds certe au probleme cependant elle reste très vague puisqu'elle ne décrit pas certaines fonctions qui sont compliqués(recherche de ressource). De plus,

A	B	C	D	E	F
COMPOSE	DocNum	COMPOSANT	QTE_T	ETAPES	
SR50RAL7045	1021616	AK210BASE.A	1	1021616,1	
SR50RAL7045	1021616	AK210BASE.A	1	1021616,2	
AK210RAL7045	1021614	AK210BASE.A	1	1021614,1	
AK210RAL7045	1021614	AK210BASE.A	1	1021614,2	
AK210RAL7045	1021614	AK210BASE.A	1	1021614,3	
AK210RAL7045	1021614	AK210BASE.A	1	1021614,4	
AK210RAL7045	1021614	AK210BASE.A	1	1021614,5	
SR21RAL6025	1021611	AK210BASE.A	1	1021611,1	
SR60RAL6025	1021610	AK210BASE.A	1	1021610,1	
SR60RAL6025	1021610	AK210BASE.A	1	1021610,2	
SR60RAL6025	1021610	AK210BASE.A	1	1021610,3	

Figure 5.3 – Association des étapes aux consommables

des contraintes non mentionnés dans l'algorithme ont été prise en compte dans l'algorithme. Voir Cahier du Développeur.

// Intégration du design pattern strategy Ayant perdu du temps de développement, il a été décider q'une seule heuristique serait développer. N'ayant aucun moyen de pouvoirs les développer, l'étudiant a décider d'intégrer le design pattern strategy au logiciel afin de faciliter le développement de nouvelle heuristique. Voir Cahier du Développeur.

3 Analyse des résultats, évaluation, qualité

Afin de juger de la qualité du logiciel, j'ai utilisé les outils sonarlint et sonarCloud. SonarCloud est capable d'analyser les porjets de tous types et permet d'évaluer la qualité sur plusieurs critères.

Bugs(erreurs majeurs) Code Smell (erreurs mineurs) Sécurité Maintenabilité

Les Code Smells sont des mauvaises pratiques de conception logicielle qui peuvent conduire à mettre votre application en défaut et peut produire des bugs. Comme indiquer sur la figure 5.1, le code peut être considéré d'une bonne qualité aux vu des différents critères mentionnés.

Comme l'indique la figure ci dessous, le programme a réussi a planifier un jeu de données composé de 10 OFs composés de 10 étapes chacune en 9 millis secondes. Il est difficile de créer des jeu de données aussi long et aléatoire à la main. Il est donc possible de conclure que l'algorithme à l'air de bien s'exécuter sur des petites instances et qu'il sera nécessaire de posséder un outils capable de générer de lourdes données.

4 Principales IHM

Voici les différentes IHM que vous rencontrerez lorsque vous utiliserez l'application

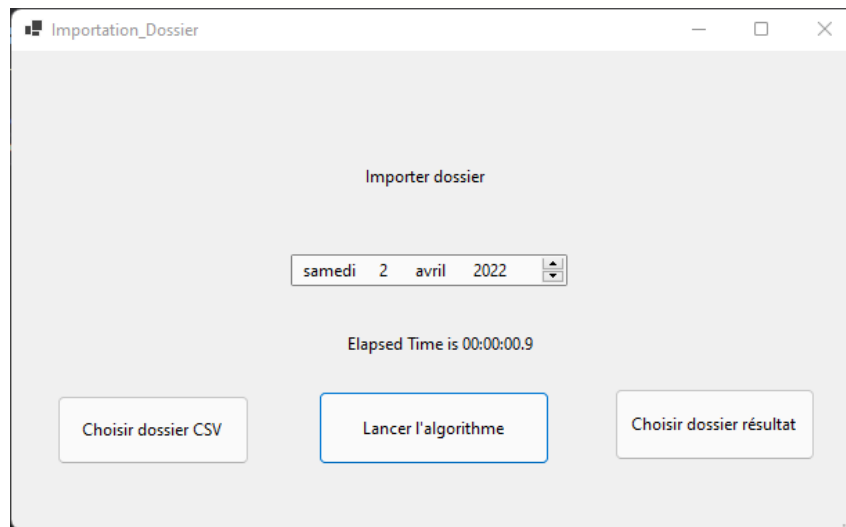


Figure 5.4 – *Exécution sur jeu de données de 100 Etapes*

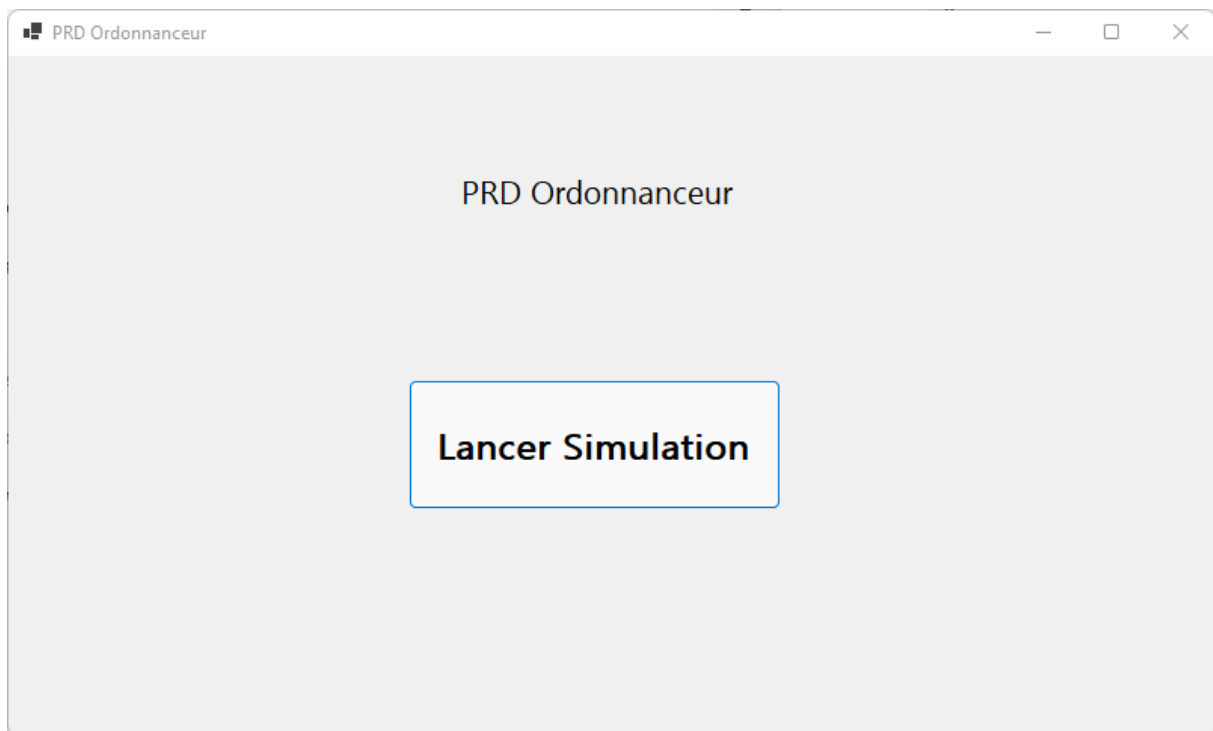


Figure 5.5 – *Menu Principal*

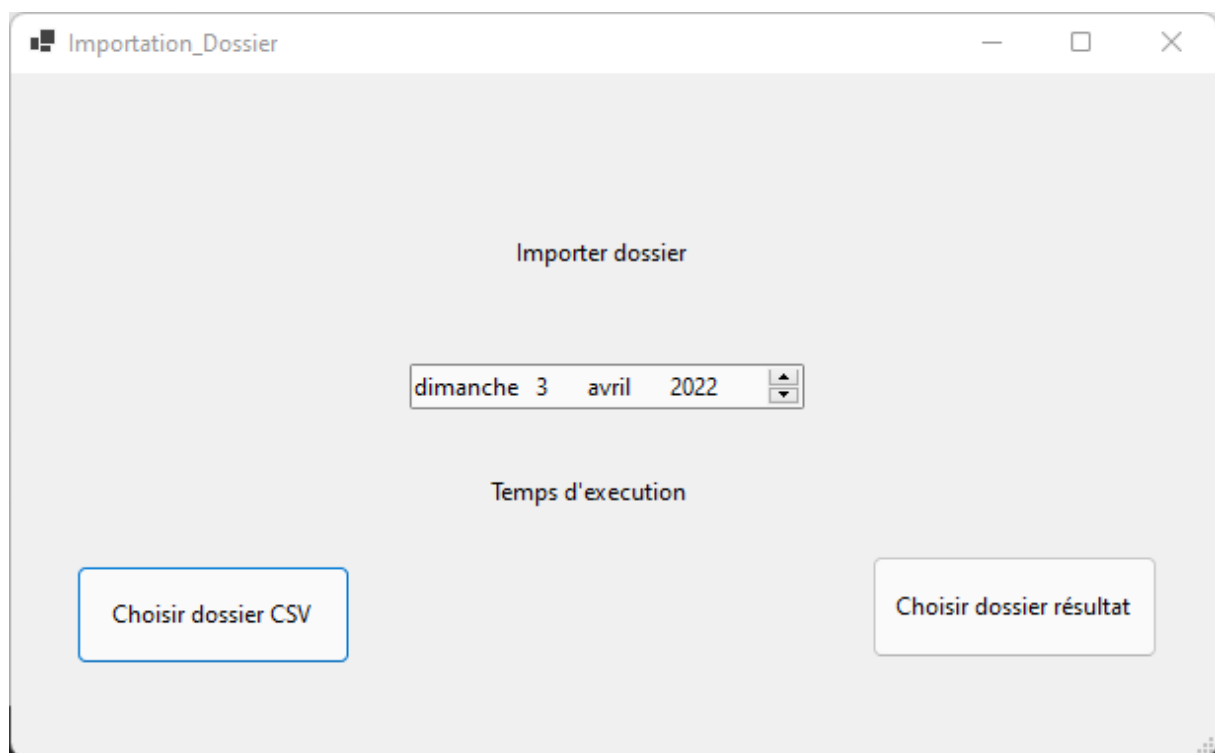


Figure 5.6 – Menu de lancement de l’algorithme

6

Bilan et conclusion

1 Bilan du semestre 9

C'est la première fois que je travaille sur un projet d'aussi grande envergure seul. En fin de semestre 9, j'ai pris beaucoup de retard sur les tâches à faire due à ma négligence sur la taille des tâches à effectuer.

Je remercie M. Kergosien d'avoir soulever le problème très tôt dans le projet et d'avoir pris le temps de m'expliquer mes erreurs.

Je prends conscience du travail restant à faire. Afin de remettre le projet en temps et en heure, je vais changer de façon de travailler et de me coordonner avec mon tuteur académique afin de combler le retard. Il s'agit d'une expérience nécessaire qui me sera indispensable face aux projets que je veux accomplir dans le futur.

CF Planning S9 et S10 à fournir en annexe

2 Bilan du semestre 10

Dans cette section, nous allons résumer dans un tableau récapitulatif la liste des implémentations et contraintes respectés par l'étudiant.

Liste des fonctions implémentés	
Liste d'implémentation	État d'avancement
Affichage d'une IHM, Sélection des données de données d'entrées	Fonctionnel
Sélection d'un dossier de sortie pour les plannings	Fonctionnel
Sélection d'une date de lancement de l'algorithme	Fonctionnel
Recherche de ressources(opérateur, machines, etc...)	Fonctionnel et partiellement testé
Sauvegarde des ressources dans les plannings	Fonctionnel
Transformation des données d'entrées en données pour l'algorithme	Respectés et testé
Trie des OFs (heuristique)	Fonctionnel et testé
Récupérer et afficher le nombre de contrainte non respectés	Fonctionnel
Transformation des données de l'algorithme en planning	Fonctionnel
Intégrer un vérificateur qui valide la solution proposée	Partiellement fonctionnel(voir liste contrainte)

Liste des contraintes respectés	
Contraintes	État d'avancement
Opérateur qui ne travaille que durant les heures de travail	Respectés et testé
Opérateur qui ne travaille pas durant les weekend	Respecté et testé
Opérateur qui n'effectue qu'une tâche à la fois	Respecté et testé
Opérateur qui travaille sur une machine que s'il est qualifié	Respecté et testé
Machine qui n'effectue qu'une tâche à la fois	Respecté et testé
Machine qui ne travaille pas si son calendrier d'indisponibilité lui indique	Non implémenté
Machine qui ne travaille que sur une étape qui demande le type de cette dernière	Implémenter et non testé
Cuves qui n'effectue qu'une tâche à la fois	Respecté et testé
Une Cuve est assigné à un OF que s'il possède le bon type	Non implémenter
Une étape utilise un nombre disponible de consommable	Respecté et testé
Une étape dite reportable peut avoir des étapes planifiées sur plusieurs jours différents	implémenter non testé

Voici un extrait d'un planning de sortie

A1 : X ✓ fx UID Operator						
	A	B	C	D	E	F
1	UID Operato	UID OF	Job Start	Job End	Code	
2	1	1021610	#####	#####	OPBefore	
3	3	1021610	#####	#####	OPAfter	
4	3 /		#####	#####	OPNetMachine	
5	2	1021610	#####	#####	OPBefore	
5	3	1021610	#####	#####	OPAfter	
7	3 /		#####	#####	OPNetMachine	
8	2	1021610	#####	#####	OPBefore	

Figure 6.1 – Planning Opérateur

3 Bilan sur la qualité

Compte tenu du temps imparti, je me suis focalisé sur les fonctionnalités les plus importantes du projet et la conception du code. SonarCloud m'a permis de confirmer que le projet rendu est maintenable et de bonne qualité. C'est à dire que les erreurs sont extrêmement rare par rapport à la taille du code et qu'il sera facile pour un développeur de le reprendre et de l'améliorer (voir figure 5.2)

4 Bilan auto-critique

Je suis assez satisfait de ma progression. Ma motivation m'a permis de rediriger le projet dans le bon sens à mon avis. Je pense que la communication avec mon encadrant a été un grand atout durant le projet. Ma curiosité a été selon moi d'une grande utilité durant ce projet, car elle m'a poussé à utiliser des outils (intégration continu) qui vont apporter une meilleure qualité au projet.

Je dois néanmoins améliorer ma capacité à prévoir la durée d'une tâche afin de pouvoir mieux évaluer les risques liés à celle-ci. Ma motivation est aussi un point à améliorer, car c'est à cause de celle-ci que j'ai pris beaucoup de retard au début du projet.

Je reste convaincu que le bilan de ce projet sera positif et j'ai acquis des compétences qui me seront essentielles dans le monde de l'informatique et de la gestion de projet.

Annexes

Planification, gestion de projet

1 Evolution du projet

Voici les diagrammes montrant l'évolution du projet

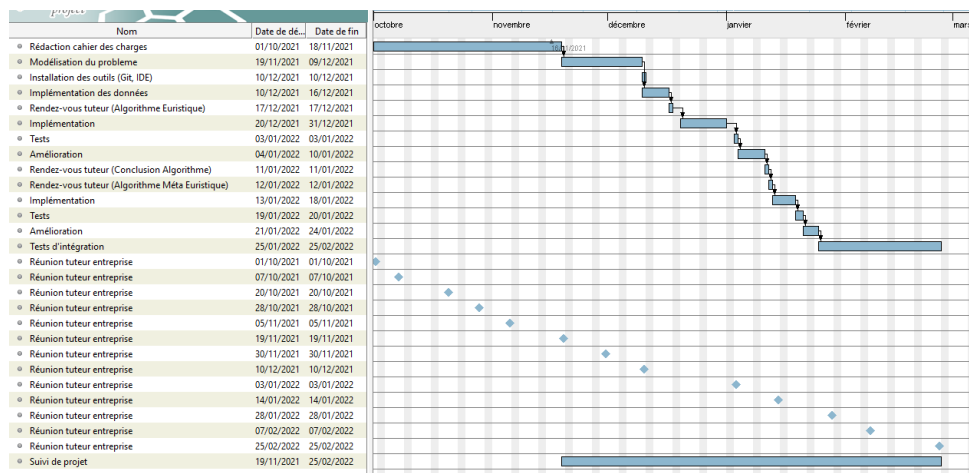


Figure A.1 – *Le diagramme de Gantt Initial*

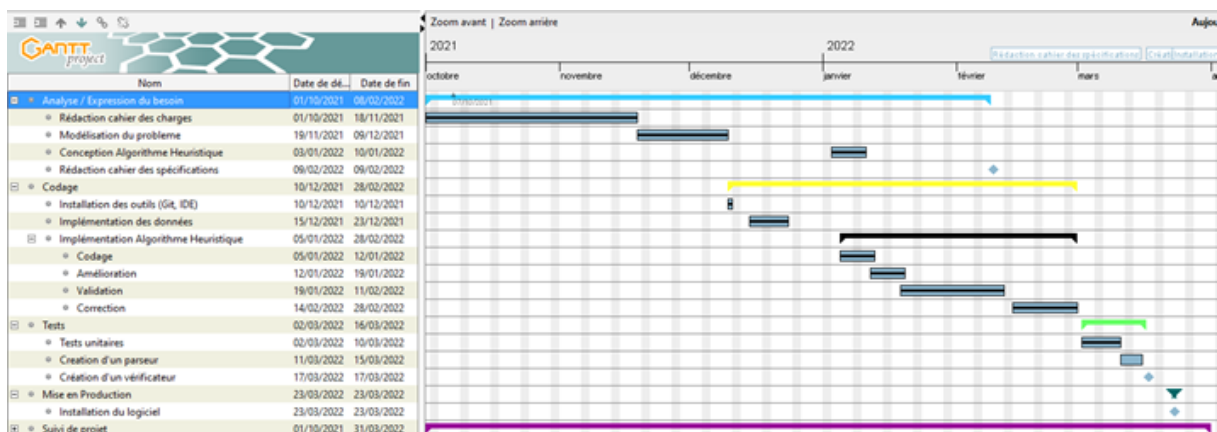


Figure A.2 – *Le diagramme de Gantt Final*

Problème rencontré	Risque(s) identifié(s)	Alerte(s) levée(s)	Action(s) menée(s)
Temps de travail effectué au début du projet	Risque de retard sur le projet		J'ai augmenté la quantité d'heure de travail sur le projet de plus de 6 heures par semaine afin de compenser le temps perdu.
Mauvaise estimation du temps prévu pour les tâches	Risque de retard sur le projet		Il a fallu faire une réorganisation des tâches en fonction de leur priorité. De plus, le nombre de tâches non prioritaire a été réduit afin de respecter les jalons.

Figure A.3 – Tableau d'action

1 spécifications Fonctionnelles

1.1 Définition de la fonction 1 : Trie des OFs en fonction de la date au plus tôt

Description de la fonction 1 :

L'heuristique doit permettre de trier une liste d'OFs en fonction de la date au plus tôt. C'est à dire que le premier élément sera l'OF qui pourra être commencer en premier par rapport aux autres.

- Données d'entrée : liste d'OFs
- Données de sortie : liste d'OFs triée

1.2 Définition de la fonction 2 : Recherche des ressources

Description de la fonction 2 :

Cette fonction faisant par de l'algorithme doit permettre à ce dernier de vérifier que les ressources telles que les opérateurs, machines sont disponibles afin de passer à l'étape supérieure

- Données d'entrée : temps actuel dans l'algorithme
- Données de sortie : un tableau des différentes ressources disponibles à l'instant t

1.3 Définition de la fonction 3 : Planification des ressources

Description de la fonction 3 :

Cette fonction faisant par de l'algorithme doit permettre à ce dernier de planifier que les ressources telles que les opérateurs, machines dans des données formatées.

- Données d'entrée : temps actuel dans l'algorithme

1.4 Définition de la fonction 4 : Importation de données

Description de la fonction 4 :

Cette fonction doit permettre au logiciel de capturer et transformer des fichiers csv en données compréhensibles par l'algorithme. Ces données csv auront un certain format.

- Données d'entrée : fichiers csv

1.5 Définition de la fonction 5 : Exportation des plannings

Description de la fonction 5 :

Cette fonction doit permettre à l'algorithme de transformer les données issues de la planification en planning pour chaque ressource existant.

— Données de sortie : fichiers csv

2 Spécifications non fonctionnelles

2.1 Contraintes de développement et conception

Ce projet ne possède aucune contrainte au niveau du développement. Néanmoins, la conception du logiciel est basé sur l'algorithme lui même issues des contraintes aux niveaux des différentes ressources mise en disposition.

2.2 Contraintes de fonctionnement et d'exploitation

2.2.1 Performances

Concernant les performances, rien n'est précisé par le client concernant l'algorithme ou la création de plannings. Cependant, on exigera une exécution dans l'ordre de la seconde voire minute.

2.2.2 Contrôlabilité

L'utilisateur doit être capable de choisir la date de démarrage de l'ordonnancement.

2.2.3 Sécurité

On utilisera un logiciel de contrôle de qualité afin de minimiser les risques de vulnérabilités.



Cahier du développeur

1 Introduction

Cette section est destinée aux futurs développeurs souhaitant reprendre l'application. Il comprend les diagrammes UMLs nécessaires à la compréhension du programme.

2 Diagrammes architecturaux et UML

Voici les diagrammes UML représentant l'intégralité du logiciel

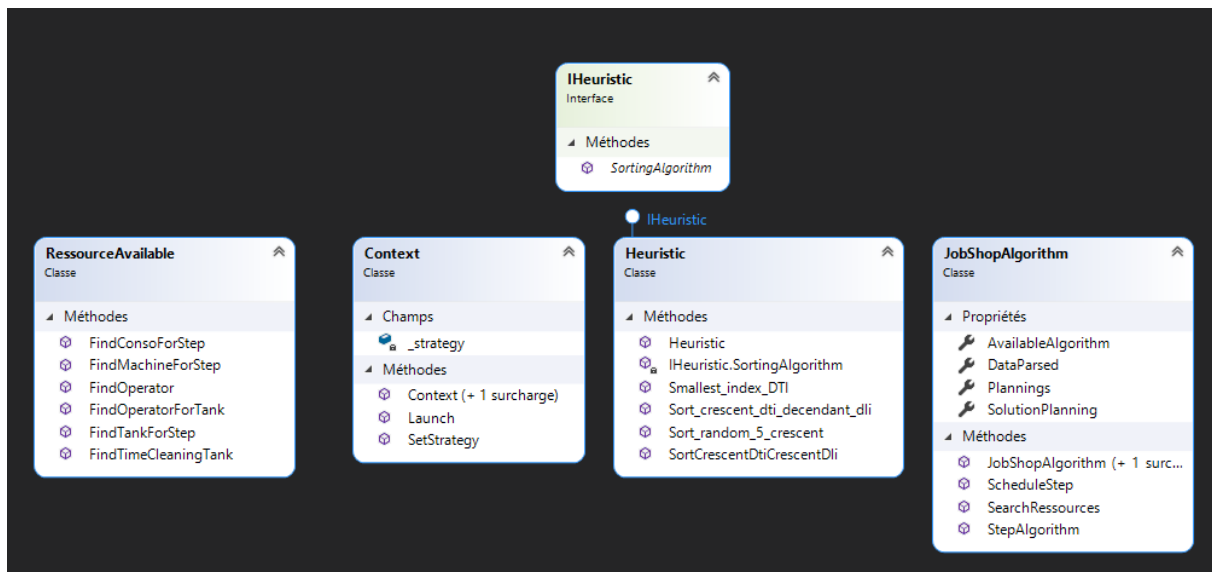


Figure C.1 – Package Algorithms

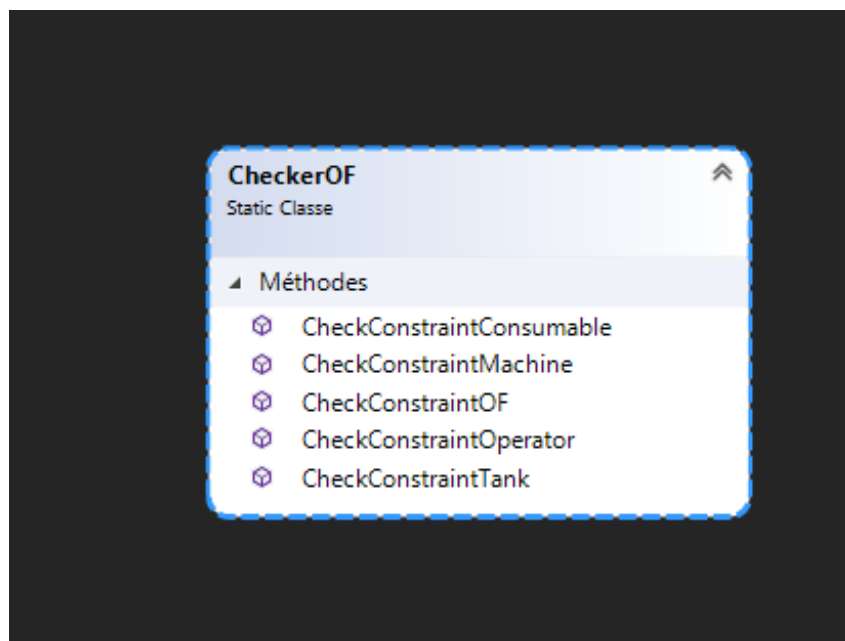


Figure C.2 – Package Checker

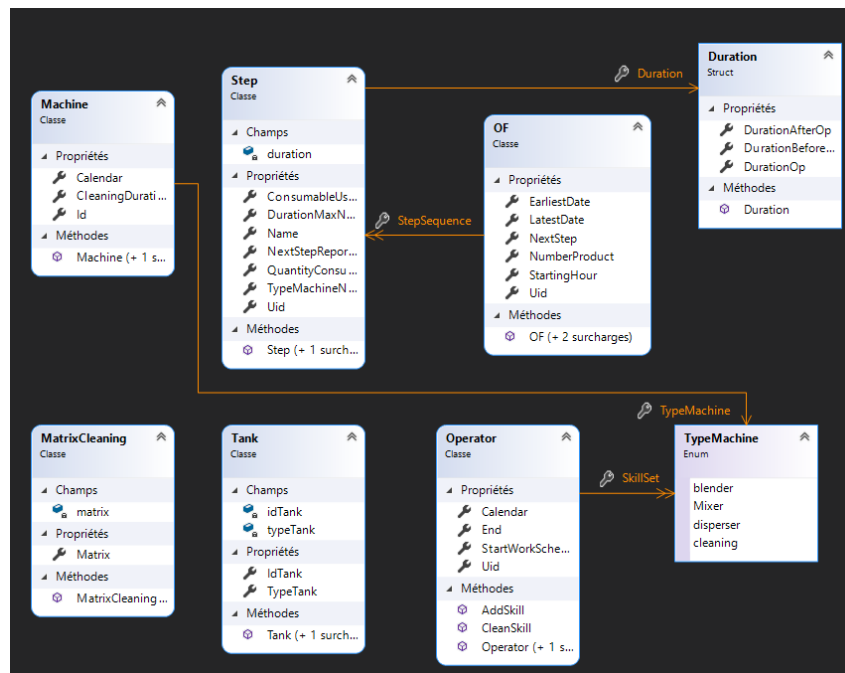


Figure C.3 – Package Data

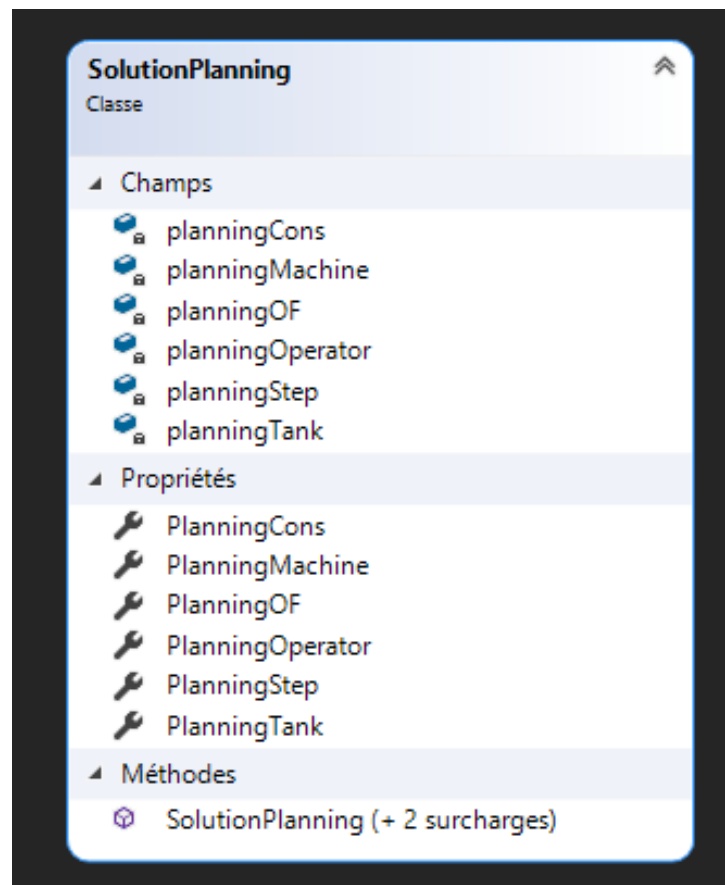


Figure C.4 – Package Solution

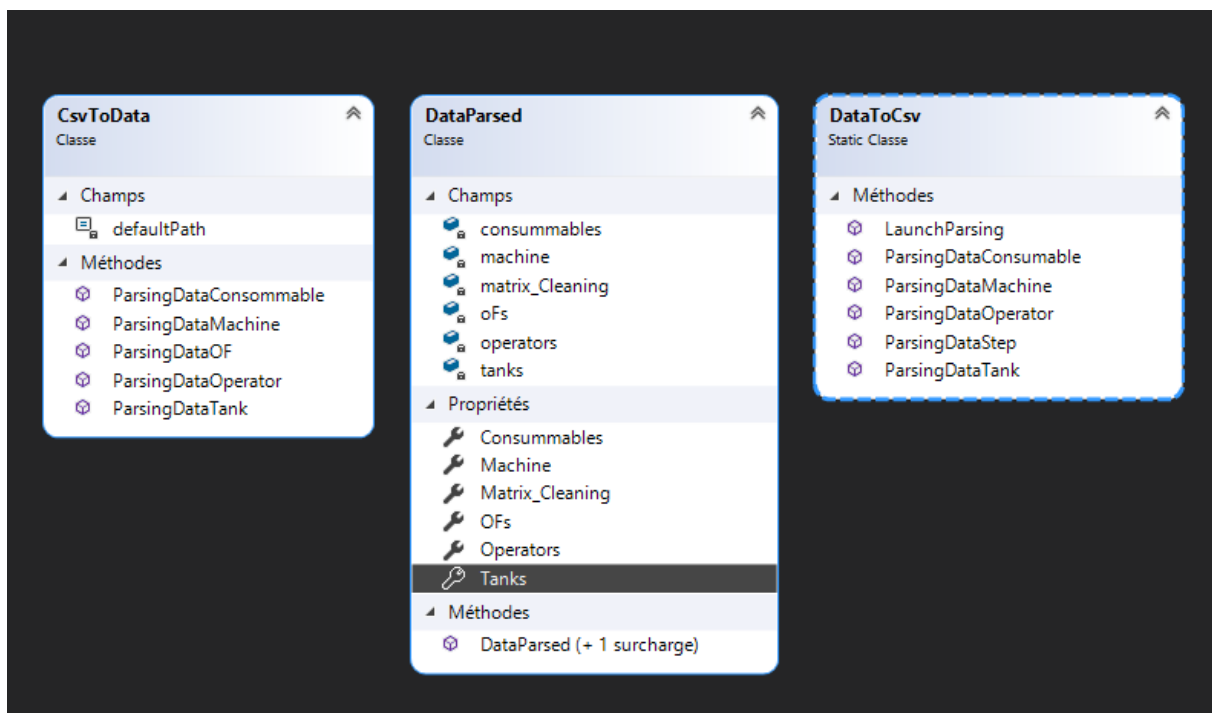


Figure C.5 – Package Parser

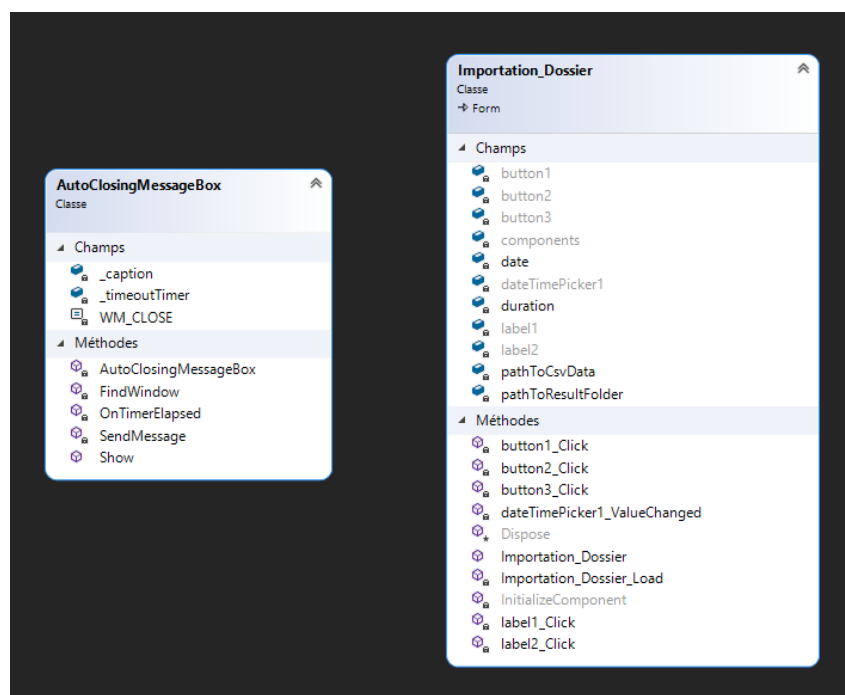


Figure C.6 – Package View

3 Descriptions détaillées de données exploitées

L'algorithme doit accéder aux fichiers csv afin de pouvoir commencer le traitement. Ces données sont constituer des fichiers suivantes

- Cuves.csv
- Etapes.csv
- Machines.csv

- OFS.csv
- Operateurs.csv
- Stocks.csv

4 Descriptions détaillées des classes, modules, réalisations

La description détaillées des classes et modules sera disponible sur le git de dépôt via une documentation Doxygen.

Lors du l'exécution de l'algorithme, vous trouverez les fichiers ci-dessous vers le chemin que vous avez spécifié pendant le lancement du programme.

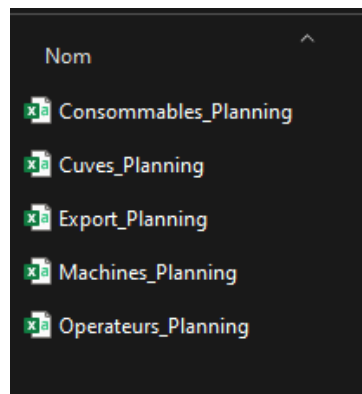


Figure C.7 – *Planning de sortie d'algorithme*

D

Document d'installation

Ce document regroupe toutes les informations nécessaires pour l'installation du projet sur les machines, ainsi que pour sa mise en production.

1 Installation

Le programme étant un exécutable, la seule condition nécessaire à sa bonne exécution est l'obligation d'utiliser un ordinateur possédant un système d'exploitation Windows.

Pour installer le programme dans un logiciel de développement, vous devez vous munir de :

- Visual Studio
- Git

Lorsque vous détenez ces outils, il vous suffira d'ouvrir le projet via le dépôt Git du projet.

E

Document d'utilisation

1 Utilisation du logiciel

Lorsque vous avez téléchargé l'exécutable du logiciel, il vous suffira de l'exécuter sur un ordinateur Windows. Si vous ne possédez pas l'avez pas, vous pouvez générer l'exécutable avec le code sur Visual Studio.

Ci dessous vous trouverez les étapes à suivre afin d'exécuter l'algorithme sans problème.

- Appuyer sur le bouton Lancer la simulation
- Choisir le dossier d'entrée des fichiers
- Choisir le dossier de sortie des fichiers
- Choisir la date de commencement de la planification

Après avoir suivi ces étapes, le logiciel vous notifiez de la présence d'erreur dans l'exécution. Si c'est le cas, il est possible que vos données ne possède pas le bon format.

Si l'exécution s'est bien déroulé, vous trouverez les plannings des différentes ressources au dossier que vous avez spécifié(dossier de sortie)

Les tests visent à garantir l'exactitude, l'intégrité, la sécurité et les performances du logiciel.

1 Tests unitaires

La majorité des tests sur l'application se fait par l'intermédiaire du module checker de l'application. Veuillez vous référer au cahier de développeur pour en savoir davantage.

SortCrescentDtiCrescentDliTest
Trie 3 OFs
EXPECTED RESULTS
Liste triés dans l'ordre croissant
OBTAINED RESULTS
Liste triés dans l'ordre croissant

Smallest_index_DTITest
Retourne l'index de l'OF pouvant être commencer au plus tôt dans une liste
EXPECTED RESULTS
Index du plus l'OF pouvant commencer le plus tôt possible
OBTAINED RESULTS
Index du plus l'OF pouvant commencer le plus tôt possible

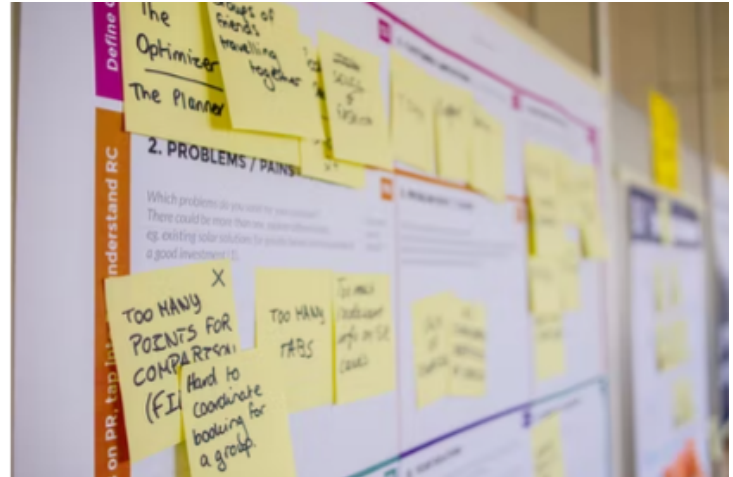
FindOperator
Verifie qu'un operator est bien retourné lorsqu'il y a un operateur disponible
EXPECTED RESULTS
Tableau contenant un opérateur
OBTAINED RESULTS
Tableau contenant un opérateur

FindZeroOperatorIfHeIsNotAvailable
Verifie qu'un tableau vite est bien retourné lorsqu'il y a pas d'opérateur disponible
EXPECTED RESULTS
Tableau vide
OBTAINED RESULTS
Tableau vide

FindZeroMachineIfItIsNotAvailable
Verifie qu'un tableau vite est bien retourné lorsqu'il y a pas de machine disponible
EXPECTED RESULTS
Tableau vide
OBTAINED RESULTS
Tableau vide

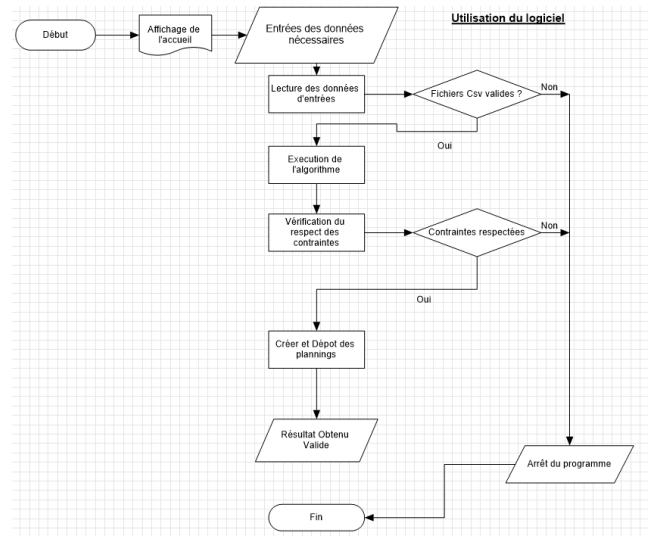
Objectifs

Le projet a pour objectif d'ordonnancer les différentes ressources d'une usine de peinture. Ces données seront transmises via des fichiers CSV à l'application qui va ensuite les ordonnancer et enfin les retransmettre sous forme de planning.



Mise en œuvre

1. Modélisation du problème
2. Développement de l'importation des données
3. Développement d'une heuristique
4. Développement de l'algorithme
5. Extraction des plannings



Résultats attendus

Le principe est trouver une solution réalisable sous la forme de planning.

A1	A	B	C	D	E	F
1	UID Operato	UID OF	Job Start	Job End	Code	
2	1	1021610	#####	#####	OPBefore	
3	3	1021610	#####	#####	OPAAfter	
4	3 /		#####	#####	OPNetMachine	
5	2	1021610	#####	#####	OPBefore	
5	3	1021610	#####	#####	OPAAfter	
7	3 /		#####	#####	OPNetMachine	
3	2	1021610	#####	#####	OPBefore	

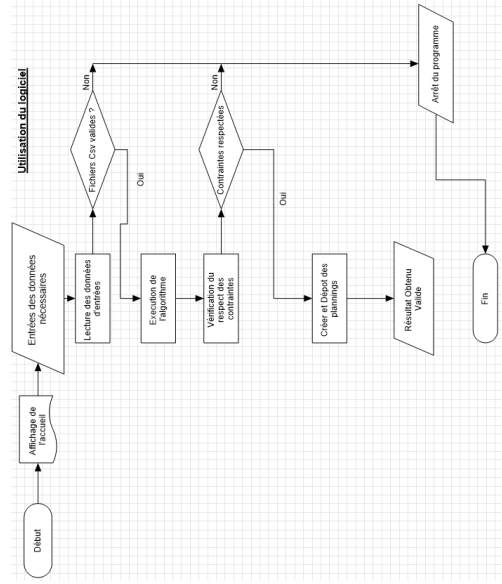
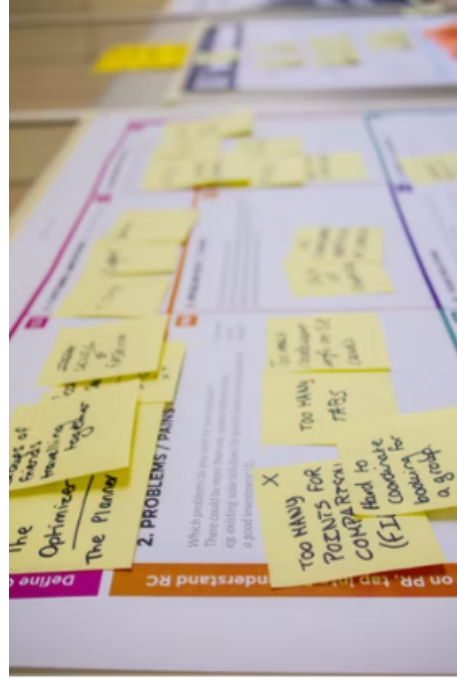
Le projet a pour objectif d'ordonnancer les différentes ressources d'une usine de peinture. Ces données seront transmises via des fichiers CSV à l'application qui va ensuite les ordonner et enfin les retransmettre sous forme de planning.

1. Modélisation du problème
2. Développement de l'importation des données
3. Développement d'une heuristique
4. Développement de l'algorithme
5. Extraction des plannings

Le principe est trouver une solution réalisable sous la forme de planning.



En collaboration avec GO SYSTEMES



A1	A	B	C	D	E	F
UID Operato	UID OF	Job Start	Job End	Code		
1	1021610	#####	#####	OPBefore		
2	1021610	#####	#####	OPAFTER		
3	3 /	#####	#####	OPNetMachine		
4	2	1021610	#####	OPBefore		
5	3	1021610	#####	OPAFTER		
7	3 /	#####	#####	OPNetMachine		
3	2	1021610	#####	OPBefore		



Ordonnanceur 4.0

Ordonnancement des ordres de fabrication

Résumé

GO SYSTEMES accompagne les entreprises de son savoir-faire dans l'évolution de leur système d'information.

Elle se spécialise dans la formation externe en informatique et l'ingénierie en informatique et notamment dans la production de PGI. Les entreprises dans le secteur de la Peinture et Chimie sont les principales concernées.

La production de peinture demande de planifier en avance l'ordre de passage des ordres de fabrication (OF). Ces OFs représentent les commandes de clients.

Généralement, l'ordonnancement est fait à la main sur un horizon de planification d'un à plusieurs jours, par une personne qui va concilier tous les paramètres et contraintes à prendre en compte.

La réalisation de cet ordonnancement prend du temps et l'arrivée d'un élément imprévu (panne d'une machine, grève. . .) provoque une ré-ordonnancement. C'est dans ce contexte que l'expression d'un besoin d'un outil d'aide à la planification a été soulevé.

Le logiciel, à son aboutissement, va ordonnancer les OFs sur un horizon, fournir en sortie diverses plannings pour les entités et permettre à l'utilisateur de choisir le critère d'optimisation de l'ordonnanceur.

Mots-clés

Ordonnancement, Développement

Abstract

GO SYSTEMES supports companies with its know-how in the evolution of their information systems.

It specializes in external training in computer science and computer engineering and in particular in the production of integrated management software. The companies in the sector of Paint and Chemistry are the main ones concerned.

The production of paint requires to plan in advance the order of passage of the orders of orders (OF). These POs represent customer orders.

Generally, the scheduling is done by hand on a planning horizon of one to several days, by a person who will days, by a person who will reconcile all the parameters and constraints to be taken into account. The realization of this scheduling takes time and the arrival of an unforeseen element (breakdown of a machine, strike of a machine, strike. . .) causes a rescheduling. It is in this context that the expression of a need for a planning aid tool was raised.

The software, at its completion, will schedule the OFs on a horizon, provide various output schedules for the entities and allow schedules for the entities and allow the user to choose the optimization criterion of the the scheduler.

Keywords

Scheduling, Development

Entreprise

GO SYSTEMES



Tuteur entreprise

Christophe BERTHON

Étudiant

Anthony ADELAIDE (DI5)

Tuteur académique

Yannick KERGOSIEN