LACOTE Raphaël

IUT Clermont-Ferrand
Département Informatique
Année 2022-2023

RAPPORT DE STAGE

Stage de 10 semaines réalisé dans l'entreprise Somab
du 23 Avril au 30 Juin 2023.

Sous la direction de Mr. DELABRE Pierre et de Mr DELAGE Raphaël

Refonte du logiciel Mab 2.0

REMERCIEMENTS

Je tiens à remercier Mr. DELABRE Pierre pour m’avoir accorder sa confiance pour la réalisation du projet.

Je souhaite également remercier Mr. DELAGE Raphaël pour m’avoir aidé et avoir apporter son expertise pour la mise en œuvre du projet.

Je souhaite dernièrement remercier Mr. WOHRER Adrien pour m’avoir suivi et conseillé durant ce stage.

J'autorise la diffusion de mon rapport sur l'intranet de l'IUT.

SOMMAIRE

INTRODUCTION

Du 23 Avril au 30 Juin 2023, J'ai réalisé un stage de 10 semaines dans le cadre de mon BUT Informatique afin de découvrir le monde professionnel, appliquer mes compétences et participer à la réalisation concrète d'un projet.

J'ai effectué mon stage dans l'entreprise SOMAB de Moulins. Il s'agit d'une entreprise de Machines-Outils. L'entreprise conçoit, fabrique et commercialise ces machines à destination de ces clients.

Ces machines dispose d'un logiciel intégré nommé MAB 2.0 permettant aux utilisateurs de donner des d'obtenir des informations et de paramétrer les machines.

Ce logiciel reçoit régulièrement des mises à jour et ajouts de fonctionnalités pour s'adaptés aux demandes des clients et aux modifications des machines. Ces modifications sont effectuées par l'un des informaticiens de l'entreprise Mr.DELAGE. Cependant, celui-ci devant également gérer une grande partie des systèmes informatiques de la SOMAB. Il lui était impossible par manque de temps d'effectuer des modifications en profondeur sur le logiciel.

Par conséquent, l'entreprise a été intéressée par ma candidature spontanée et ayant déjà recruté des stagiaires venant de l'IUT et m'a donc prit pour ce stage afin que je puisse refaire la structure du logiciel.

Je vais commencer par présenter l'entreprise et son organisation puis je présenterai le projet à réaliser. Je présenterai ensuite ma façon de m'organiser et de gérer ce projet pour ensuite comparer mes prévisions à son déroulement réel afin d'en tirer un bilan. Je ferai l'analyse du projet et présenterai ma conception de l'application pour ensuite exposer le développement du projet avec les défis et difficultés rencontrées Cela me permettra d'en tirer un bilan sur ce qui a été fait et sur ce qu'il reste à faire.

PRESENTATION DE L'ENTREPRISE

PRESENTATION GENERALE

La Somab (Société de mécanique et d'Automatisme du Bourbonnais) dispose d'une histoire riche puisque celle-ci trouve ses origines dans une entreprise fondée en 1861 : la SOMUA. Déjà à cette époque, celle-ci réalisé des machines-outils ainsi que des machines agricoles. La SOMAB est créer lorsque l'entreprise est rachetée en 1985.

L'entreprise est une PME d'une petite centaine de personnes dans le domaine des machines-outils à commande assisté ou numérique.

L'entreprise fournit un grand nombre de secteurs (Automobile, ferroviaire, aéronautique…) . Elle à une dimension internationale ayant des clients partout dans le monde et dispose de grandes entreprises françaises tel que Michelin.

Cette entreprise gère une grande partie de la chaine de production ce qui implique des domaines variés. Cela va de la conception de la machine à la conception des logiciels qui vont avec. Une partie de l'entreprise est consacrée aux domaines commerciaux et marketing afin de gérer les contrats avec les clients et avec les fournisseurs. D'autres personnes travaillent sur le service après-vente afin d'avoir de bons retour clients, et enfin la majorités des salariés de l'entreprise se consacre à la fabrication des machines.

Il y a donc des personnes aux travails extrêmement variés dans l'entreprise : Commerciaux, ouvriers, ingénieurs, informaticiens. Toutes ces personnes sont situées dans les mêmes locaux à Moulins. Il y a une partie où l'on trouve les machines et une autre avec les bureaux.

ORGANISATION DE L'ENTREPRISE

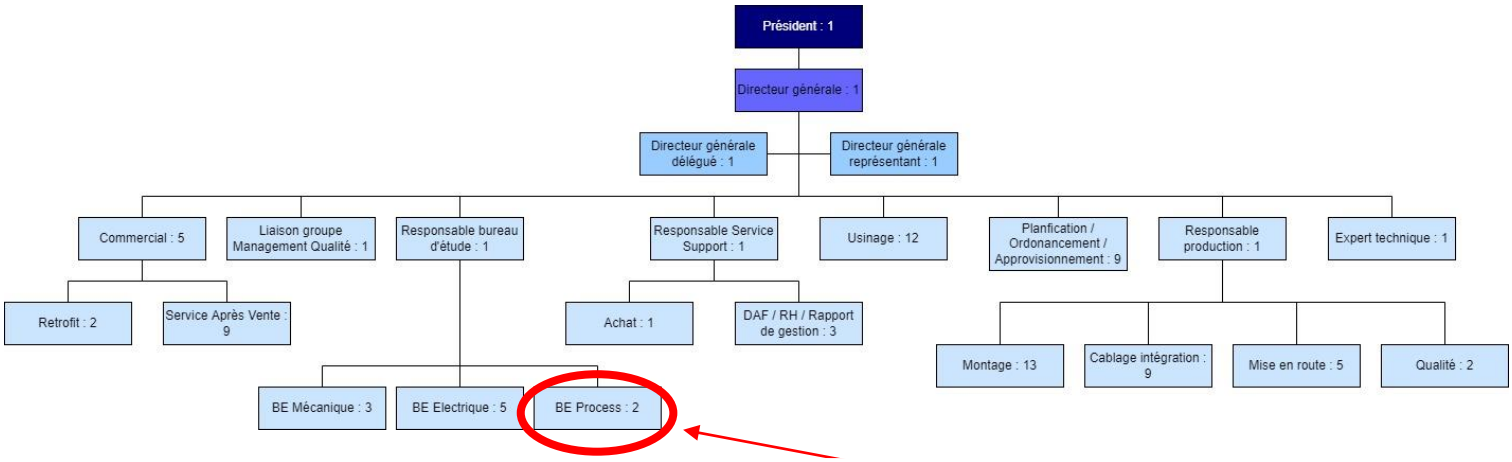


Figure 1 : Organisation de l'entreprise

Sur ce schéma, le nombre d'employés par poste est représenté par le nombre à droite. Mon tuteur se trouve dans le Bureau d'étude Process. Le but de ce bureau est d'améliorer les processus de l'entreprise et donc naturellement toute sa partie informatique.

Il gère aussi la maintenance des processus informatiques existant et en cas de problèmes ou de demandes particulières, les employés peuvent l'avertir dans sa salle pour qu'il puisse intervenir.

Je me trouve dans un petit open-space d'une dizaine de personnes des autres bureau d'étude Mécanique et Electrique qui vont intervenir dans la conception et l'amélioration des machines.

Figure 2 : Exemple de Exemple de brique

Exemple de machines fabriqués par l'entreprise. Vous pouvez voir la console où s'affichera le logiciel MAB 2.0 qui permettra d'utiliser la machine d'une manière simple et ergonomique. 3 simulateurs de la console sont mise à disposition dans l'open-space où je me trouve

PRESENTATION DU PROJET

OBJECTIF DU STAGE

Etant donné qu'il ne s'agit pas d'une entreprise d'informatique, l'entreprise ne dispose que d'un nombre restreint d'informaticiens. Ceux-ci ont pour principal mission de s'assurer de la maintenance du réseau informatique de l'entreprise, gérant des domaines tel que la sécurité informatique ou la gestion de problèmes réseaux . Ils ont également pour mission d'apporter des améliorations lorsque cela est demandé à un logiciel nommé MAB 2.0. Ce logiciel permet une utilisation simplifiée des machines.

Ce logiciel à cependant le problème d'être vieux et n'utilisent pas les dernières améliorations informatique. Celui-ci avait été créé par des personnes dont l'informatique n'est pas le domaine principale ce qui fait que certaines règles de programmations n'ont pas été respecté. A cause de cela, la structure est complexe rendant sa maintenabilité compliqué. Il y a donc une perte de temps réelle pour les programmeurs souhaitant ajouter des fonctionnalités au logiciel . Il y a de plus de nombreux bugs et problèmes d'ergonomie ce qui peut être générateur de couteuses erreurs. A cela s'ajoute des défauts d'optimisation qui peuvent entraîner une perte de temps pour les utilisateurs de la machine.

Les informaticiens n'ayant pas le temps de refonder le logiciel, l'entreprise m'a donné comme objectif de refaire le logiciel avec une meilleure structure plus adaptés aux évolutions apportés en informatique.

Le logiciel sera refait à l'aide d'un langage de programmation bien plus moderne est simple à utiliser ce qui exclut la réutilisation de l'ancien code.

Le logiciel doit respecter les contraintes suivantes :

- Le logiciel devra être capable de recevoir facilement des modifications et améliorations, par conséquent sa structure doit être clair et l'ajout d'éléments facile.

- La circulation dans le logiciel doit être la plus fluide possible dans un soucis d'efficacité et de confort d'utilisation.
- L'aspect graphique ne doit pas être modifier, il doit au contraire se rapprocher le plus possible de la version d'origine. Seul les bugs graphiques doivent être modifiés.
- Le logiciel devra naturellement avoir aucun bug qui entraînerait un problème majeur.
- De la documentation devra être effectuée pour permettre à d'autres développeurs de comprendre facilement le code du logiciel.

EXISTANT DU PROJET

Ce projet part d'un existant important : Le Logiciel Mab 2.0. Il s'agit d'une large bande de l'écran que l'utilisateur de la machine pourra utiliser à sa guise.

Figure 3 : Console du logiciel MAB 2.0

Pour comprendre son utilité, il faut nous intéresser à un de ces principaux composant : les briques. Ces briques sont une série de 2 à 4 boutons permettant d'effectuer une action particulière ou d'indiquer une valeur.

Ils y en a une grosse vingtaine de créé, quelques-unes sont simple et n'interagissent pas avec la machine comme la brique Heure Date ayant 2 boutons : Le premier Affichant l'heure et l'autre la date. Certaines comme celles appelées ModeMachine communiquent avec la machine pour changer le mode de fonctionnement de la machine. D'autres comme la brique Métrologie lance un sous-menu lui-même composé de différentes briques et boutons permettant de faire d'autres actions spécifique.

Figure 4 : Exemple de brique

L'utilisateur peut choisir les briques dont il a besoin dans un onglet de configuration, certaines briques ne doivent cependant pas pouvoir être activer pour les machines n'en ayant pas l'utilité.

Une grande partie du logiciel est composé des briques. Le principale n'étant pas de toutes les faire ce qui prendrait un temps particulièrement conséquent mais de permettre un développement et un ajout facile de celle-ci grâce à une structure clair. Un échantillon de briques a été sélectionné parmi les plus différentes et les plus utilisés pour être développé durant le stage.

Pour pouvoir communiquer avec la machine, le logiciel utilise une API* : l'API Fagor. Il s'agit d'un logiciel tiers créé par une entreprise espagnol. L'API n'ayant pas été conçu par la SOMAB et n'étant pas maintenu par elle, il me sera impossible de faire des modifications directement sur celle-ci. De la documentation est fournit pour comprendre le fonctionnement de l'API et il est possible de communiquer avec l'entreprise de l'API pour obtenir plus de renseignements.

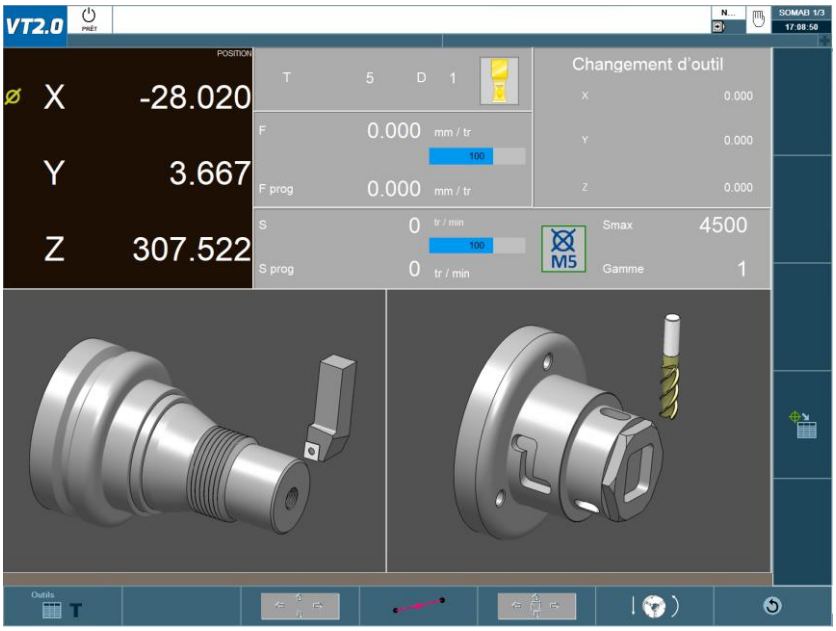


Figure 5 : Vue de l'API fagor

Le logiciel existant dispose d'options de configuration permettant de s'adapter à la clientèle comme par exemple la possibilité de changer de langues ce qui est très important lors de la vente de machines à une entreprise étrangère. Cela implique la nécessité de pouvoir facilement configurer une nouvelle langue disponible pour un nouveau client. Par exemple, dans le cas où l'entreprise commercialiserait des machines à l'inde., il doit être facilement faisable de créer des traductions pour l'Hindi et de l'appliquer à la machine

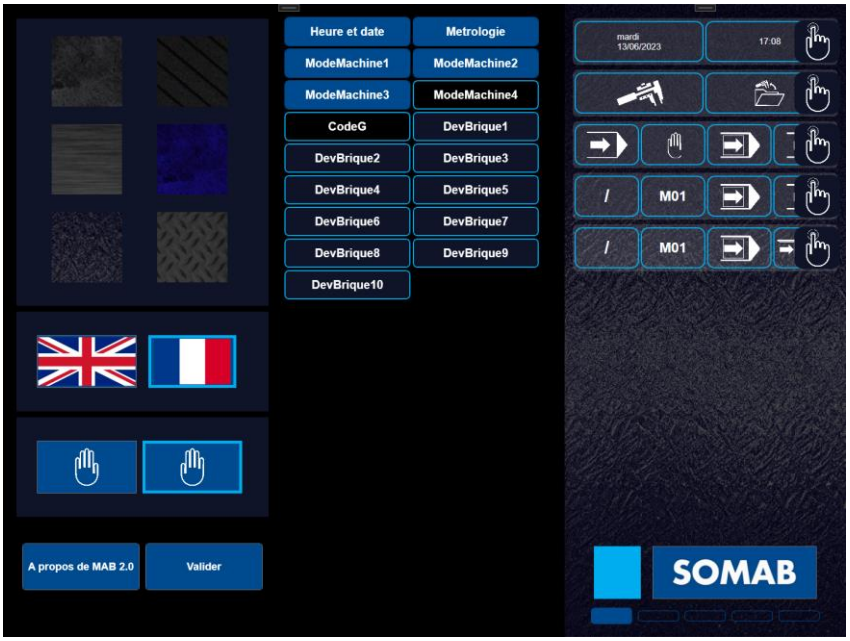


Figure 6 : Onglet de configuration du logiciel

Certains modes présentent des dysfonctionnements qu'il conviendra d'éliminer lors de l'élaboration du nouveau logiciel.

OUTILS UTILISES

Le précédent projet utilisé le langage de Programmation Visual Basic. Ce langage de programmation dispose de l'avantage d'être facilement accessible aux non-initiés. Cependant ce langage dispose de nombreux inconvénients tel que des performances moyennes et le manque de mise à jour majeur venant de Microsoft. Ceux-ci encouragent d'ailleurs les utilisateurs à se tourner vers des langages de programmation plus moderne tel que le c#.

Le C# est un autre langage de programmation de Microsoft. Il est par conséquent idéal pour une application fonctionnant sous Windows ce qui est le cas pour ce projet. Celui-ci à l'avantage de présenter un grand nombre de fonctionnalités. De plus, il est un langage fortement typé permettant une grande lisibilité du code facilitant ainsi la maintenance du logiciel Pour finir, il s'agit d'un langage de programmation moderne recevant très régulièrement des mises à jour ce qui le rend bien plus tourné vers l'avenir que le Visual Basic vieillissant.

Le XAML sera le langage utilisé pour la partie graphique du programme. Il présente l'avantage d'utiliser des balises ce qui rend aisés la disposition d'éléments graphiques.

L'IDE utilisé est Visual Studio. Il s'agit de l'IDE fournit par Microsoft pour le langage C#. Celui-ci à l'avantage de disposés de nombreuses fonctionnalités et possède des outils de débogage simple d'utilisation. Il possède également de nombreuses aides pour faciliter la création de la partie graphique.

Le framework WPF est un framework très utilisés pour les applications et langages utilisant le C# et le xaml. Il a l'avantage de séparer clairement la vue (partie graphique) du modèle (fonctionnalités) ce qui est un élément avantageux pour la clarté du code

Git est un outil servant aux développements de logiciels, il permet de sauvegarder ces avancements ou encore de revenir à une version antérieure en cas de problème sur la version en cours. Il permet de plus à mon tuteur de suivre l'avancée du projet.

ORGANISATION

MODE DE PREVISION

Avoir de bons moyens de vérifier que la qualité du projet est au rendez-vous est indispensable. De plus il est évident qu'il faut avoir des repères pour vérifier que le projet pourra être terminé dans les temps. Je vais maintenant décrire les outils de suivi de projet utilisés. Aucune méthode de suivi de projet ne m'a été imposé, j'ai donc décidé de choisir des procédés que je connaissais ayant pu les expérimenter durant les SAE.

J'ai décidé de créer un WBS qui me permet de décomposer mon projet en catégories, elles-mêmes divisées en plusieurs taches. Cela permet de visualiser facilement les différentes sous-tâches à réaliser pour obtenir un projet finit. Etant donné que des imprévus peuvent survenir durant le projet, je me suis réservé la possibilité de rajouter des tâches durant le projet.

L'utilisation du pert me permettra de réfléchir à la durée des différentes tâches du WBS ce qui m'aidera à visualiser les tâches les plus longues à réaliser. Cela me permet aussi d'analyser les dépendances entre les différentes tâches et ainsi de visualiser les tâches qui doivent être réalisé en priorité car d'autres tâches en dépendent.

Le calcul du chemin critique créé à partir du pert permet d'observer les tâches qui ne doivent pas souffrir de retard au risque de retarder le projet dans son ensemble. Il permet en plus d'avoir une bonne vision de la succession des différentes tâches

A partir de ces informations et afin de décider quelles tâches doivent être faites à quel moment, un Gantt a été réaliser pour planifier le projet.

ORGANISATION PREVISIONELLE

WBS

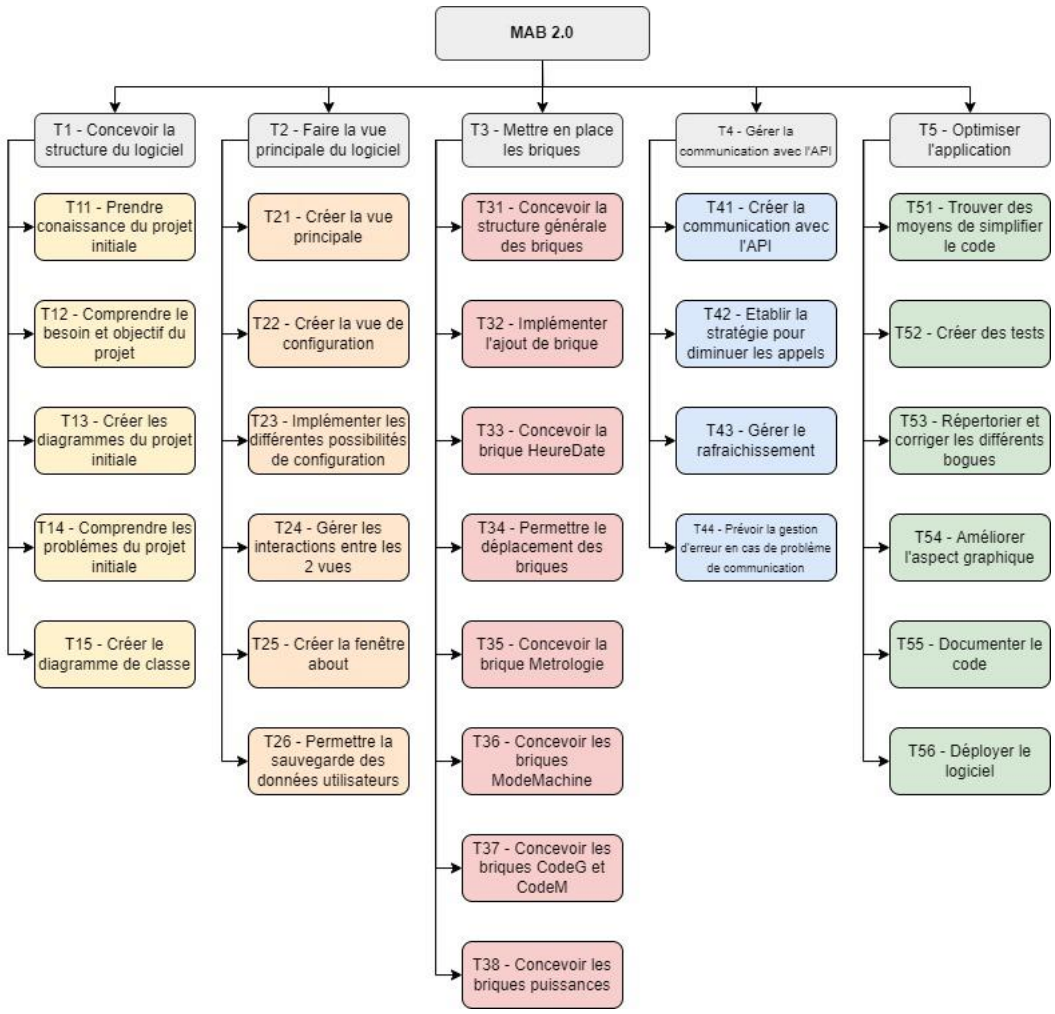


Figure 7 : WBS

Ce WBS répertorie les différentes tâches que j'ai pu identifier durant mon analyse du projet. Ces parties du projet sont :

- La conception de la structure du projet pour avoir un plan et une structure clair pour commencer à coder avec un plan et des objectifs.
- La programmation de la structure du projet, des menus principaux et des fonctionnalités basiques de l'application
- La mise en place des différentes briques que les utilisateurs utilise pour diriger la machine.
- La gestion de la communication avec l'API qui permettra au logiciel de communiquer avec la machine.
- Les finition et l'optimisation du programme afin de s'assurer de rendre un logiciel de qualité.

PERT

Code	Description	Durée Estimée	Durée Réele	Anteriorité
T11	Prendre conaissance du projet initiale	4	4	
T12	Comprendre le besoin et les objectifs du projet	1	1	T11
T13	Créer les diagrammes pour le projet initiale	3	3	T11
T14	Comprendre les problèmes du projet initiale	1	1	T13
T15	Créer le diagramme de classe du nouveau logiciel	3	3	T12 / T14
T21	Créer la vue principale	5	7	T15
T22	Créer la vue de configuration	3	2	T15
T23	Implémenter les différentes possibilités de configuration	6	5	T22
T24	Gérer les intéractions entre les deux vues	2	2	T21 / T23
T25	Créer la fenêtre about	1	1	T22
T26	Permettre la sauvegarde des données utilisateurs	2	1	T23 / T32
T31	Concevoir la structure générale des briques	4	4	T15
T32	Implémenter l'ajout de briques	1	1	T31 / T24
T33	Concevoir la brique HeureDate	2	1	T31
T34	Permettre le déplacement des briques	1	3	T32 / T33
T35	Concevoir la brique Métrologie	6	10	T33 / T41
T36	Concevoir les briques ModeMachines	2	1	T33 / T43
T37	*Concevoir les briques Code G et Code M*	*2*	*2*	*T33 / T43*
T38	*Concevoir les briques Puissance*	*3*	*1*	*T33 / T43*
T41	Créer la communication entre le logiciel et l'API	4	3	T21
T42	Etablir la stratégie pour diminuer les appels à l'API	1	1	T41
T43	*Gérer le rafraichissement*	*2*	*4*	*T42*
T44	*Prévoir la gestion d'erreur en cas de problème de communication*	*1*		*T43*
	Vérifier le fonctionnement de l'appli qur d'autres machines		2	
T51	*Trouver des moyens de simplifier le code*	*2*		*T2 / T3 / T4*
T52	*Créer des tests*	*4*		*T2 / T3 / T4*
T53	*Répertorier et corriger les différents bogues*	*4*		*T52*
T54	*Améliorer l'aspect graphique*	*3*		*T2 / T3 / T4*
T55	*Documenter le code*	*1*		*T51*
T56	*Déployer l'application*	*2*		*T51/T53/T54*

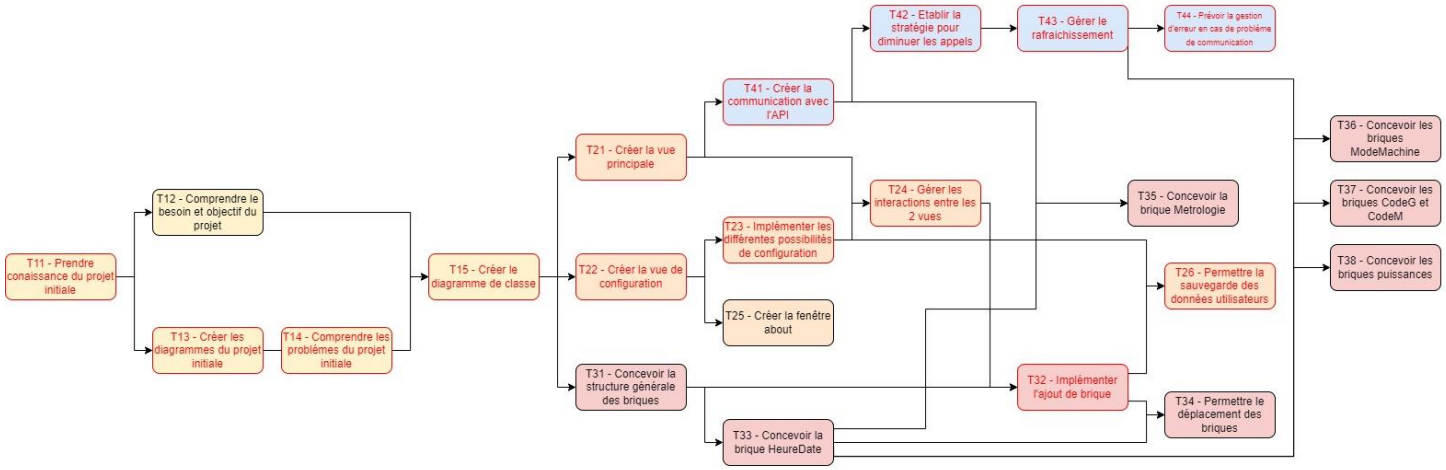
Figure 8 : PERT

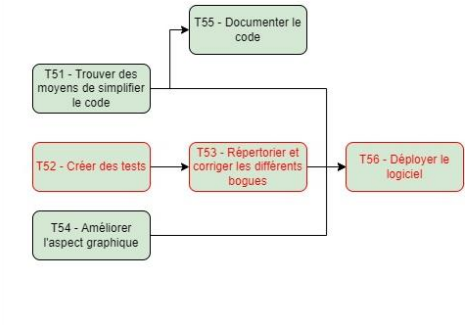
Ce pert a pour but de chercher à connaître les tâches qui sont les plus longues et celles qui dépendent d'autres tâches. Pour estimer leurs durées, je me suis basé sur mes évaluations lors des SAE afin de diminuer ma marge d'erreur. La durée des tâches a été estimé en demi-journée pour clarifier le pert. En tenant compte du fait que je travaille 9 demi-journées de durée similaire par semaine et en prenant en compte les jours fériés, je trouve 81 demi-journées durant mon stage. Le totale de la durée de toute les taches étant de 76 demi-journées, on peut considérer des imprévus pouvant aller jusqu'à 5 demi-journées.

On peut grâce à ce pert comparer la différence entre les durées estimées des durées réelles rentrées durant le projet. On peut voir que globalement, les durées sont similaires malgré quelques divergences, les durées réelles sont cependant légèrement moins bonnes qu'escomptés.
Le stage n'étant pas terminé, il me reste 18 demi-journées pour terminer les taches restantes (en gris sur le pert). Selon mes estimations, il me reste 17 demi-

journées pour les terminer, il y a donc de bonnes chances que le projet puisse être menée à son terme à condition de ne pas prendre de gros retard sur une ou plusieurs des tâches. Au cas ou un retard serait pris, des tâches comme l'amélioration de l'aspect graphique ou la documentation du code pourront être raccourcies car n'étant pas indispensable au fonctionnement basique de l'application.

CHEMIN CRITIQUE





Ce schéma représente le chemin critique des taches à réaliser pour le projet. Il me permet d'avoir une bonne vision d'ensemble sur les interactions entre les tâches. Les tâches du chemin critique (texte en rouge) permettent de visualiser les points critiques du projet sans quoi les autres ne pourront pas avancer correctement.

Figure 9 : Chemin critique

GANTT PREVISIONNEL

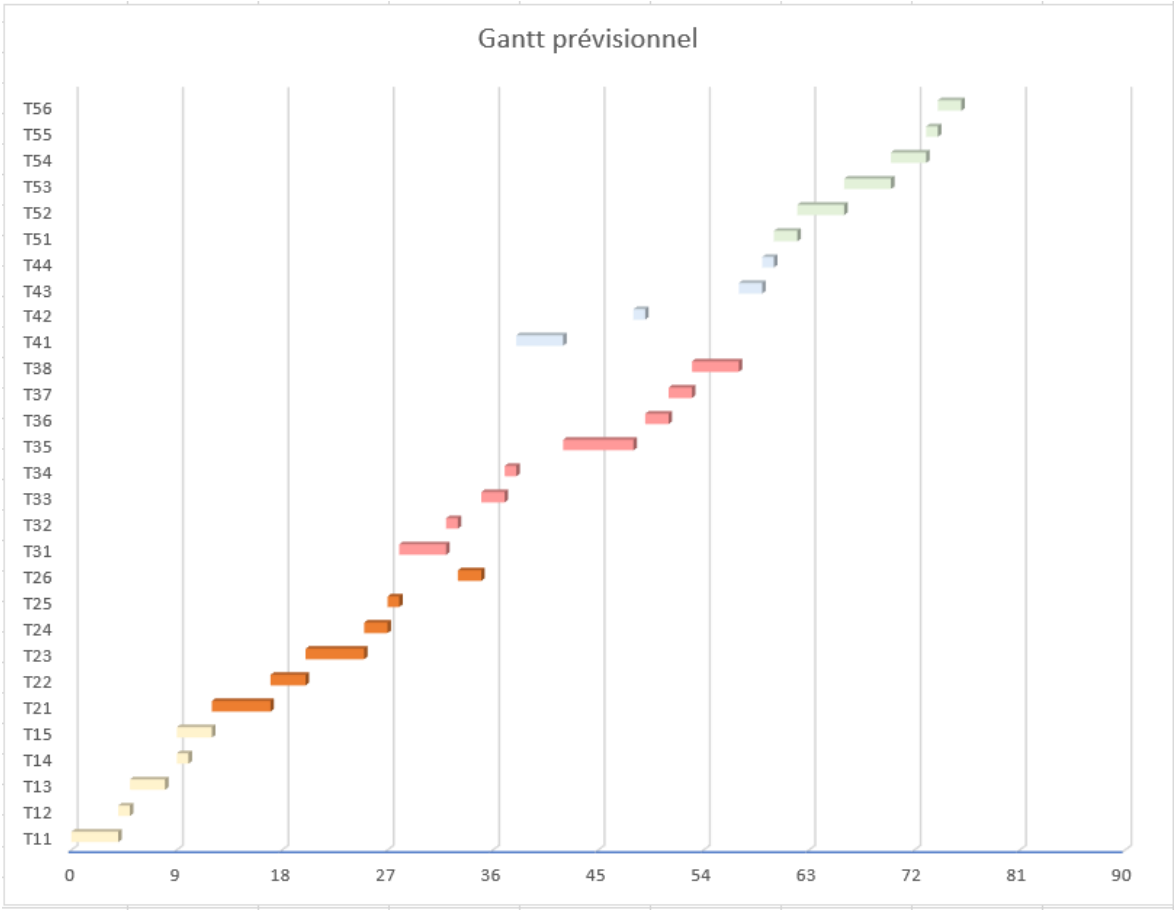


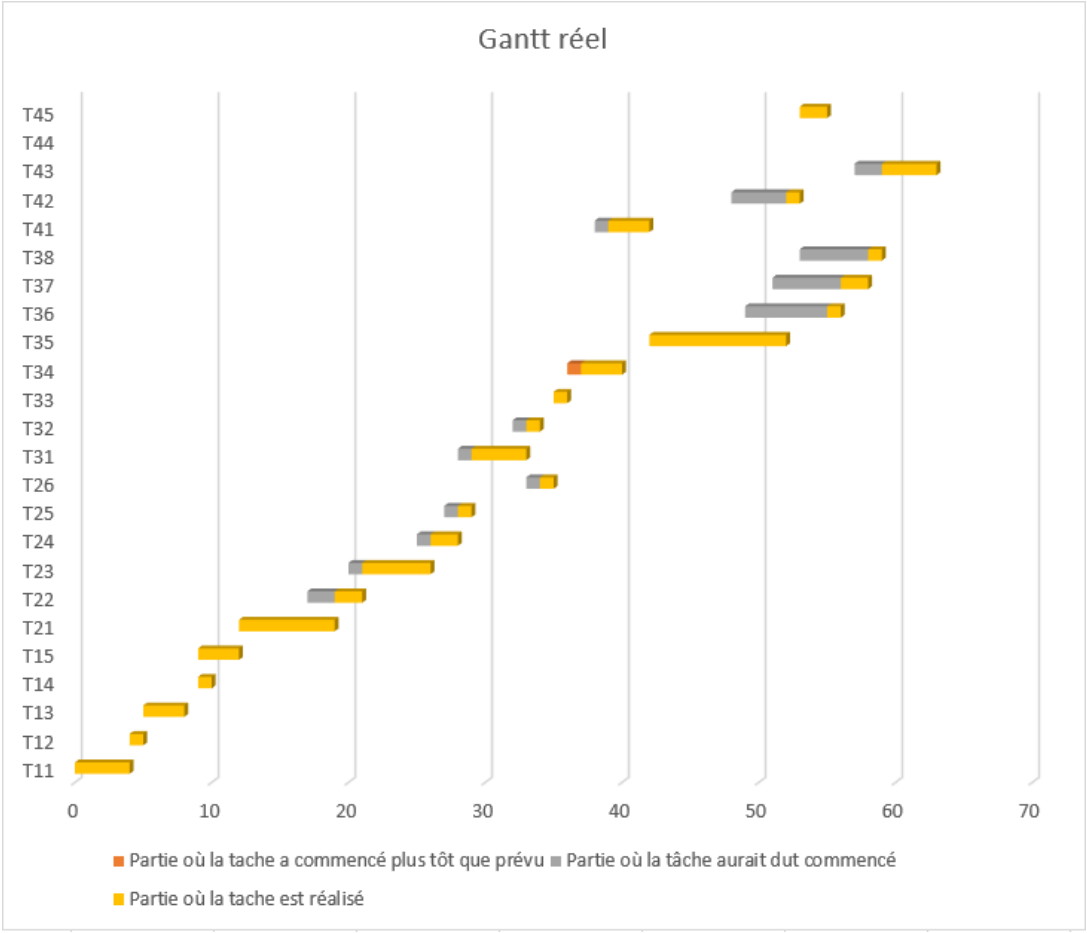
Figure 10 : GANTT Prévisionnel

Ce gantt montre mon plan pour le projet. Bien sûr, l'idée n'est pas de respecter absolument ce plan mais surtout d'avoir une idée de l'avancement du projet afin de vérifier qu'il n'y est pas de retard particulier.

La première semaine et le début de la seconde seront consacrés à la découverte du projet et à la construction de la structure, je ne coderai donc pas durant cette partie-là.
Les tâches oranges représentant la structure de base de l'application seront conçues en premiers pour avoir une bonne structure pour ensuite effectuer la partie rouge représentant le codage des briques qui sont les fonctionnalités réelle de l'application.

Les tâches en bleu ciel représentant la communication avec l'API seront effectuer en parallèle des taches rouges car une majorité des briques nécessite une communication avec l'API pour fonctionner.
Les tâches vertes étant des tâches d'optimisation, elles seront effectuées en dernier une fois que la majorité des fonctionnalités prévues seront terminées.

GANTT REEL



Sur le gantt réel, nous pouvons voir que la plupart des tâches ont été réalisés dans le temps imparti. La tâche T34 a même pu être commencée plus tôt que prévu.

Néanmoins le retard prit sur le développement de la brique métrologie (Tache 35) a entrainé des retards ce qui s'est répercuté sur les tâches s'effectuant après. On peut voir cela avec les barres grises montrant le retard prit sur la configuration initiale. Le retard de cette tâche s'explique par la refonte d'une partie de la structure de la brique pendant la programmation de celle-ci.

A ce retard s'est ajoutée la nécessité d'ajouter une tâche qui n'était pas prévu au départ :la tâche T45 pour tester le logiciel sur d'autres configurations de machines comme Windows 7 afin de vérifier que tout fonctionne correctement.

Dans l'ensemble le gantt à néanmoins été plutôt bien respectée, les décalages restants mineurs.

ANALYSE

ANALYSE DU BESOIN

L'analyse du besoin sert à identifier précisément les acteurs du systèmes ainsi que ces objectifs afin d'éviter de passer à coté des enjeux du logiciel.

Après l'analyse du besoin, j'ai put formalisés la bête à cornes suivante ainsi que le diagramme de cas d'utilisation du logiciel.

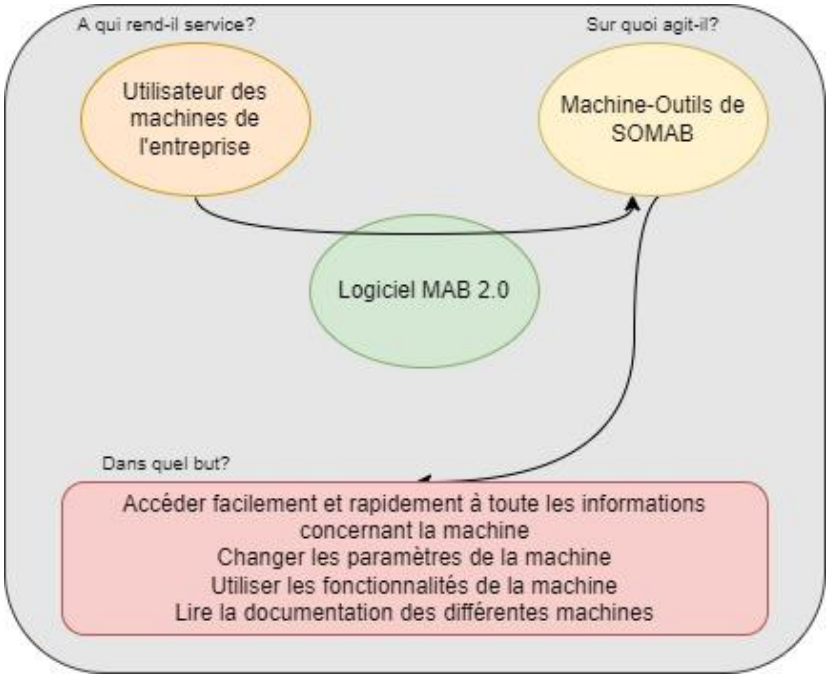


Figure 11 : Bête à cornes

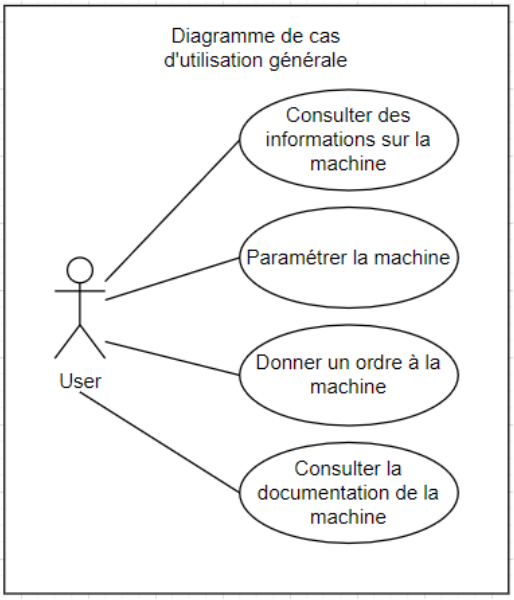


Figure 12 : Diagramme de cas d'utilisation

Sur le diagramme de cas d'utilisation, seul les buts généraux ont été définit. Paramétrer la machine implique par exemple de nombreux paramètres différents possible et tous les mettre aurait beaucoup trop alourdis le schéma.

On trouve un seul rôle pour les utilisateurs: celui qui va gérer la machine. S'Il n'y pas de différences de rôles entre les utilisateurs, il y en a en fonction de la machine sur lequel le logiciel est. En effet certaines machines peuvent utiliser des briques que d'autres ne peuvent pas utiliser. Les actions sont donc plus ou moins développés en fonction des machines.

COMMUNICATION ENTRE LES ELEMENTS

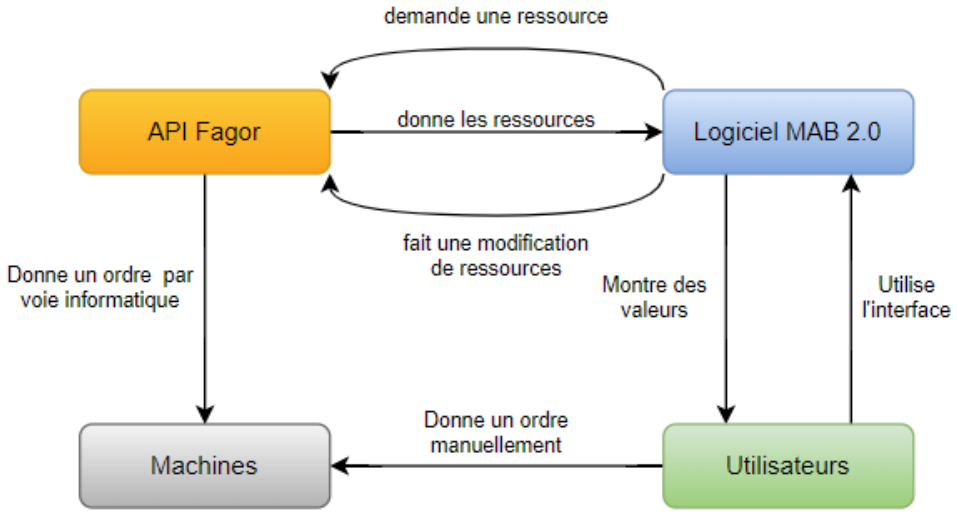


Figure 13 : Schéma des communications entre les éléments

Notre logiciel va interagir directement avec les utilisateurs qui se serviront du logiciel pour effectuer des modifications via l'API. Les utilisateurs pourront également connaître une valeur de la machine. Pour cela ils utiliseront le logiciel qui demandera une ressource à l'API qui pourra leurs fournir via l'interface du logiciel. Le logiciel sert de passerelle entre la machine et l'utilisateur pour que ces derniers puissent utiliser les machines de la manière la plus ergonomique possible.

DIAGRAMME D'ACTIVITE

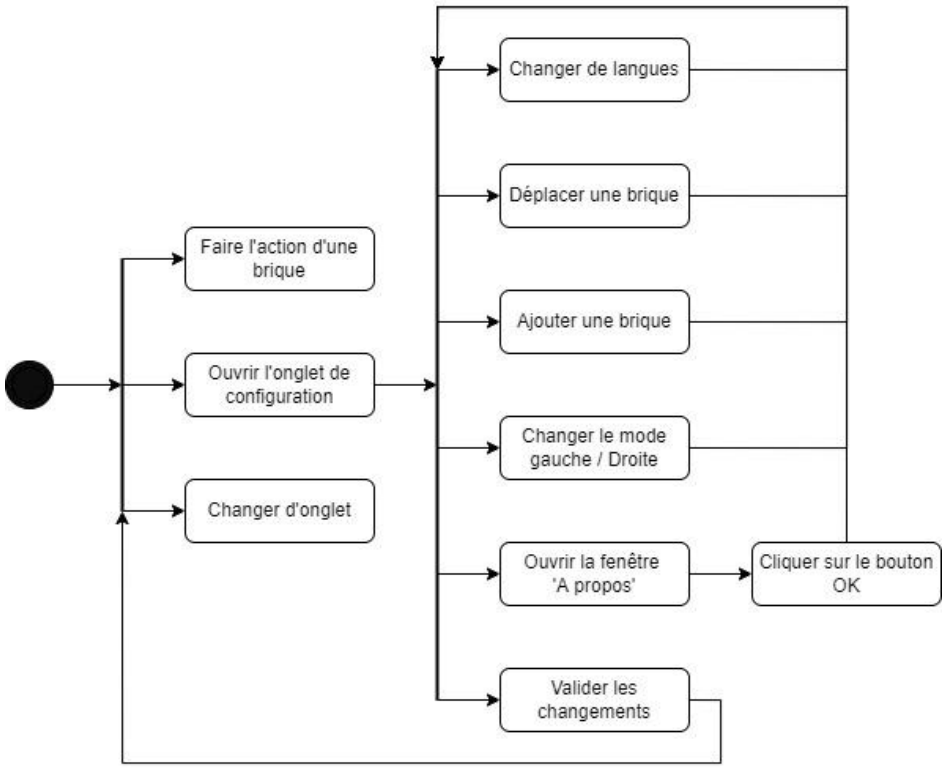


Figure 14 : Diagramme d'activité

Un diagramme d'activité a été réalisé pour comprendre comment les utilisateurs utilisent le logiciel. Ils entrent dans le logiciel (point noir) et arrivent dans la vue principale où ils ont le choix entre 3 possibilités.

- Faire l'action correspondant à un bouton d'une brique sur lequel ils ont cliqué.
- Changer l'onglet sur lequel ils se trouvent ce qui leur permet d'avoir des briques différentes et donc d'autres actions possibles.
- Ouvrir l'onglet de configuration pour changer les paramètres de celles-ci ce qui ouvre alors une seconde vue.

Cette seconde vue va permettre de changer diverses paramètres :

- Le changement de langues (Par défaut entre le français et l'anglais).
- L'activation du mode gauche ou droite (Le mode gauche met le logiciel sur la partie gauche de l'écran alors que le mode droite fait l'inverse)
- Le changement de la colorie du fond d'écran.
- L'utilisateurs peut déplacer à partir de cette écran changer à sa guise la position des briques dans la vue principale.
- L'utilisateur peut ajouter une brique lui permettant d'avoir accès à plus d'actions sur la vue principale.
- Ils peut ouvrir une popup contenant les informations légales du logiciel.
- Le dernier bouton permet de sauvegarder les changements et de quitter cette seconde vue.

Ce diagramme d'activité restera le même entre la première et la deuxième version du logiciel, les deux versions devant être les plus proche possible.

ANALYSE DE L'EXISTANT

Le logiciel existant présente un certaines nombre de défauts qu'il convient de détecter afin de trouver des solutions pour ne pas les reproduire.

Tout d'abord il y a un grand nombre de répétitions de code, cela nuit considérablement à la maintenabilité et à la lisibilité du code. Il conviendra donc de créer des classes et des fonctions adaptées pour résoudre ce problème.

L'organisation du code en elle-même présente de nombreux inconvénients. Par exemple habituellement on met des variables dans des classes qui les utilisent. Ici on trouve un énorme module contenant plus d'une centaine de variables qui peuvent être utilisés par n'importe quel partie de code. Il est donc très pénible de comprendre à quel partie du code correspond les variables. Celles-ci pouvant être modifié à n'importe quel endroit, cela provoque de nombreux problèmes de dépendances.

Ce module n'est pas le seul à poser problèmes. Il existe d'autres modules regroupant plus d'une trentaines de méthodes et disposant de plus d'un millier de lignes de code. Il serait bien plus pertinent de répartir ces méthodes dans des

classes les utilisant afin de rendre le code plus clair et par conséquent bien plus maintenable.

On trouve à de nombreux endroits des boucles inutiles ce qui entrave les performances du logiciel car cela rajoute des calculs inutiles.

On remarque également beaucoup trop de comparaisons de chaines de caractères or il s'agit de calculs n'étant pas rapide à effectuer.

```
Case "Utilities"
    RendreFocusAppliCN()
    Retour = FagorVariables.WriteInt(FagorIndexMarqueur, 1, 6007, -1)

Case "Recall"
    RendreFocusAppliCN()
    Retour = FagorVariables.WriteInt(FagorIndexMarqueur, 1, 6008, -1)
Case "Custom"
    RendreFocusAppliCN()
    Retour = FagorVariables.WriteInt(FagorIndexMarqueur, 1, 6009, -1)
Case "MainMenu"
    RendreFocusAppliCN()
    Retour = FagorVariables.WriteInt(FagorIndexMarqueur, 1, 6010, -1)
Case "Switch"
    RendreFocusAppliCN()
    Retour = FagorVariables.WriteInt(FagorIndexMarqueur, 1, 6011, -1)
```

Figure 15 : Exemple de code mal optimisé

Ici on peut voir un exemple de code sous-optimale avec de nombreuses comparaisons de chaines de caractère. Cette partie de code se répète sur plus de 200 lignes dans un module en comptant plus d'un millier. Chaque Case est là pour un bouton. On peut supprimer ces problèmes en créant une fonction prenant en paramètre la seul valeur qui change dans cette partie de code.

Au niveau du ressenti utilisateur. On trouve comme problème une actualisation des données lente, de nombreux bugs et un comportement générale peu fluide.

DEVELOPPEMENT

BRIQUE

Pour montrer un exemple de brique, nous allons en prendre une d'entre-elles qui a été refaites dans le nouveau logiciel, la brique Métrologie.

La brique métrologie se compose de 2 boutons : le premiers boutons change un paramètre sur la machine, le second ouvre un nouveau menu.

Figure 16 : Brique Métrologie

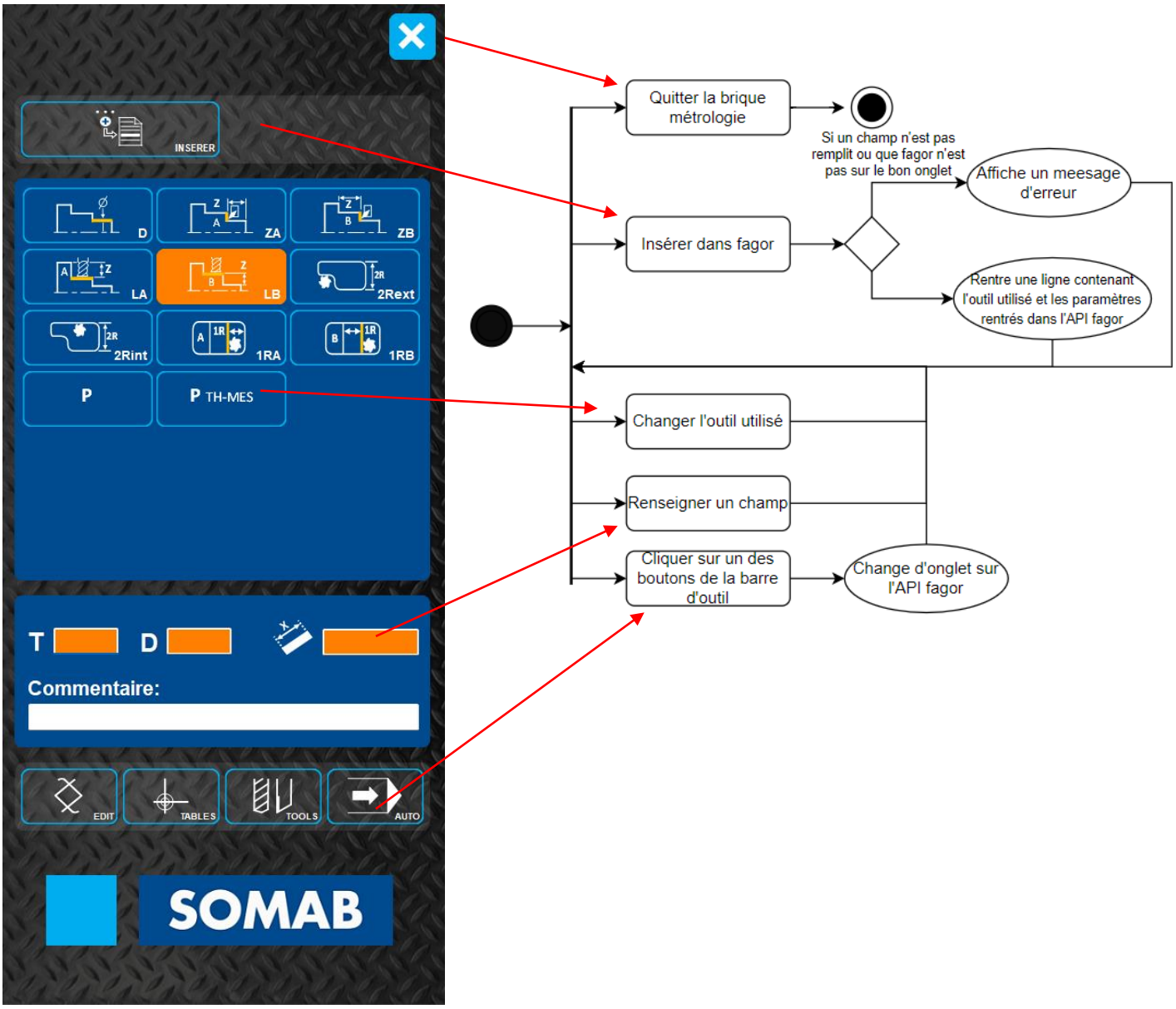


Figure 17 : Vue et digramme d'activité du sous-menu de la brique Métrologie

Ce menu secondaire contient elle-même 2 autres briques : Une permettant de changer d'onglet dans lequel l'API fagor se trouve avec 4 boutons correspondant chacun à un onglet.

Le cœur de cette brique réside dans la possibilité d'ajouté une ligne de commande pour la machine grâce au bouton d'Insertion. Il doit pour cela rentrer les paramètres dans les champs prévues et changer d'outils utilisés grâce aux boutons du dessus.

La brique fournit une interface clair permettant l'insertion d'une ligne en limitant grandement le risque d'erreurs pour l'utilisateur grâce à sa simplicité d'utilisation.

Certaines défauts apparaissent sur cette brique notamment du point de vue de l'intuitivité. En effet la brique contenant les différents boutons pour changer d'onglet est mal placé par rapport aux boutons d'insertion. En accord avec mon tuteur, certains changement d'organisations ont pu être réalisé.

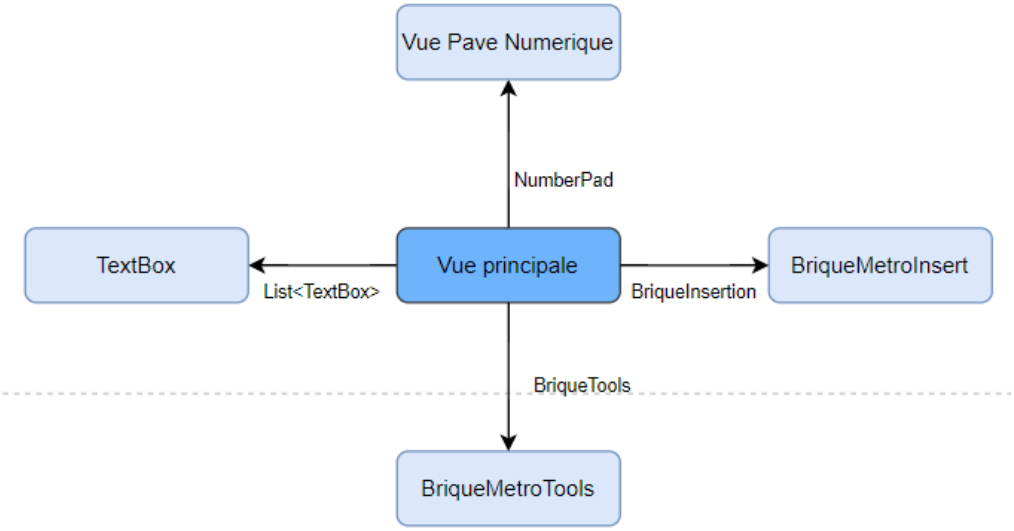


Figure 18 : Schéma des interactions du code de la briqué Métrologie

Du point de vue du codage, je me suis rendu compte que coder toutes les fonctionnalités de la vue dans la même classe allait considérablement complexifié le code et le rendre moins lisible. Etant donné que cela est une des plus importantes contraintes du logiciel pour faciliter la maintenance.

Pour les textbox par exemple (les champs à renseigner), une classe à été créée avec des paramètres pour la limite de caractères dans le champ, le texte affiché… ce qui permet une compréhension facile.

La vue principale contiendra les valeurs permettant de faire le lien entre les différentes parties de la vue.

STRUCTURE DES BRIQUES

Les briques représentent une grande partie de l'application étant données que celles-ci apportent l'utilité principale du logiciel qui comme noté plus haut permettent aux utilisateurs d'interagir avec la machine. Celles-ci sont composées de 1 à 4 boutons différents. La structure précédente du logiciel pour les briques apporte de nombreux problèmes

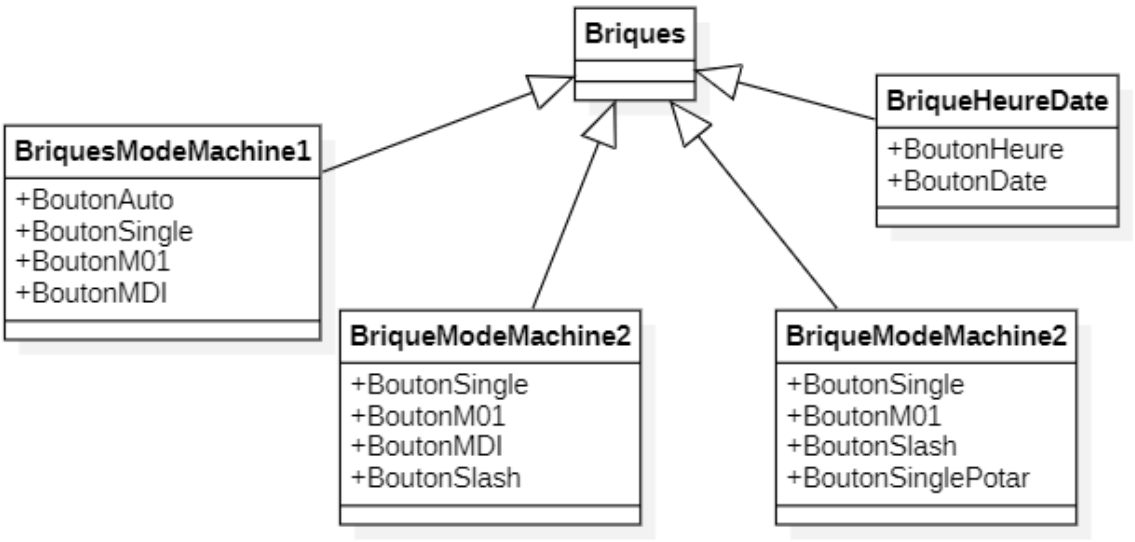


Figure 19 : Diagramme initiale de la structure des briques

Pour des raisons de simplification, uniquement 4 briques sont représentés sur le diagramme.

Comme vous pouvez le voir sur ce schéma simplifié, Les briques héritent tous d'une classe mère qui permet de rassembler les fonctionnalités communes entre les briques. Ceci est une bonne chose car cela permet d'éviter des répétitions de code et de manier facilement les briques différentes car étant toutes du type Brique.

Chaque briques dispose de sa partie graphique ainsi que de sa partie fonctionnalités

Cependant le problème réside dans le fait que les boutons soit directement codés dans celles-ci or comme vous pouvez le voir sur le diagramme de nombreux boutons sont disponibles pour des briques différentes. Leurs codes et leurs utilisations (ce qu'il se passe quand on appuie dessus) se retrouvent dans plusieurs endroits à la fois. Cela pose plusieurs problèmes :

- Créer une nouvelle brique utilisant un bouton préexistant demande de recopier l'intégralité du bouton.
- Des problèmes de cohérence peuvent survenir si le code d'un même bouton est différent d'un endroit à l'autre.
- Le code est bien plus long et complexe ce qui le rend bien moins lisible et donc plus difficilement maintenable.
- La modification du bouton implique de modifier tous les endroits où il se trouve rendant l'opération plus longue et est plus susceptible de provoquer des bugs.

Pour éliminer ces inconvénients, il était donc nécessaire de réfléchir à une nouvelle structure du code des briques.

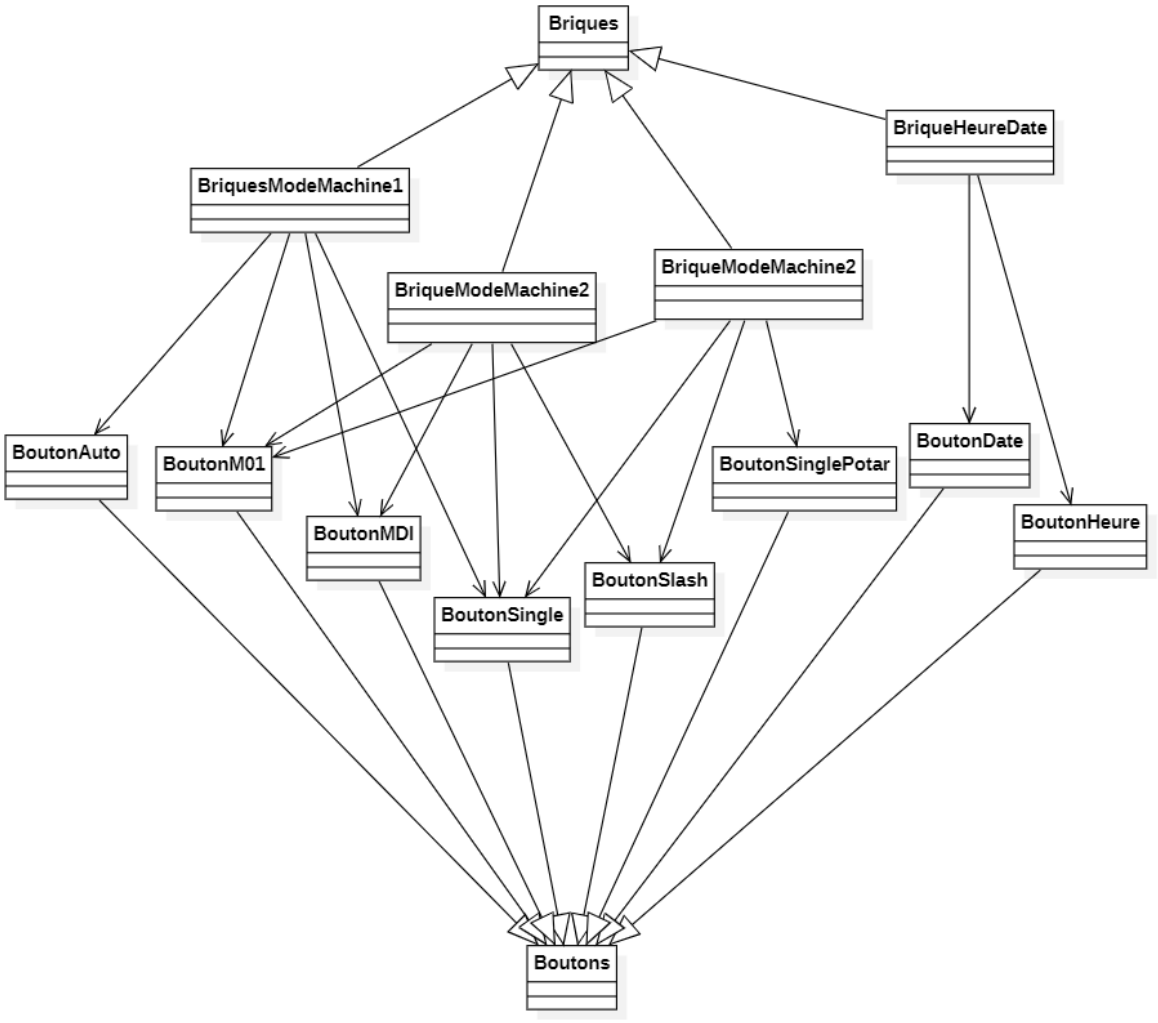


Figure 20 : Diagramme intermédiaire de la structure des briques

La première idée est de créer une classe bouton auquel tous les diverses types de boutons pourront hériter. Chaque boutons disposent d'une partie graphique (en xaml) et d'une partie fonctionnalités (en C#). La classe mère n'a que du code C#. Chaque briques contient quant à elles les boutons qui corresponde.

Cela permet de réduire drastiquement la duplication de code améliorant ainsi la structure. Cependant, beaucoup de boutons ont des graphismes en commun et il reste donc de la duplication de code. De plus, comme vous pouvez le constater, la structure devient plus complexe et pas plus clair, j'ai donc décidé d'encore la retravailler afin de supprimer ces inconvénients avant de commencer à coder la structure des briques.

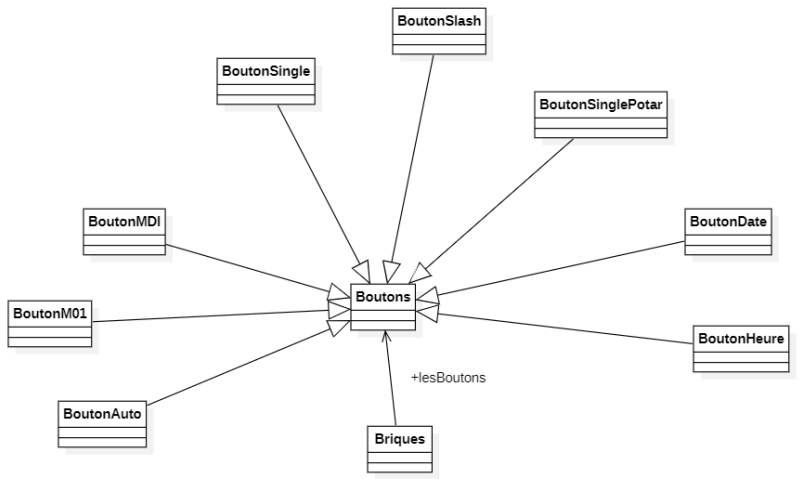


Figure 21 : Diagramme finale de la structure des briques

Cette structure est celle que j'ai retenu pour le logiciel. La création de différentes classes pour les briques était devenu inutile car la seul différenciation entre elles était les boutons contenus dedans. La classe Briques aura donc une liste de boutons, la liste étant différente pour les diverses briques. Cela permet donc de simplifier considérablement la structure du code sans engendrer de répétitions. La création de briques reprenant des boutons existant devient donc très facile à réaliser.

Pour les boutons, seul la classe mère Boutons obtient du code xaml pour la partie graphique. Cela sert à condenser le code en un seul endroit le code graphique éliminant les dernières répétitions. Le code devient plus simple car chaque boutons n'a qu'une partie C# et pas de parties graphique. L'inconvénient est que les particularités graphiques de chaque boutons sont indiqués dans le code C# nuisant quelques peu à la lisibilité du code mais cela reste une amélioration par rapport à l'ancienne structure.

COMMUNICATION AVEC L'API

L'un des principales problèmes du logiciel préexistant est au niveau de sa communication avec l'API, il était donc important de revoir la conception de cela. Le problème est le nombre trop élevé de demandes à l'API ce qui entraîne une surcharge diminuant sa réactivité .

Il y a 2 types de demandes à l'API : celles ayant pour but d'obtenir une ressource, par exemple la valeur d'une variable et celles ayant pour but de modifier une ressources par exemple modifier le mode de fonctionnement de la machine.

Ce dernier cas ne pose pas de problème particulier, on appel l'API uniquement lorsque on clique sur un bouton adapté pour changer la valeur. Même en cliquant à répétition sur le bouton ce qui ne sert à rien, cela ne fait pas assez d'appels pour surcharger l'API.

Le problème intervient lors de l'obtention des ressources. Des boutons de nombreuses briques affichent des valeurs qui peuvent changer à tout moment. Pour cela le logiciel initiale utilise un chronomètre de 0.75 secondes. Au bout de ce laps de temps, pour chaque valeurs d'affichées le logiciel envoie une requête à l'API. Lorsque de nombreuses briques contenant ces valeurs sont affichées à l'écran cela provoque de nombreux appels surchargeant l'API alors qu'un grand nombre de valeurs n'ont pas eu de changement. A cause de cela, l'actualisation des valeurs ne peut pas être plus rapide pour ne pas générer trop de demandes à l'API. C'est donc au niveau des appels de lecture de valeurs qu'il faut apporter des modifications à la structure.

Ce cas de figure étant un problème arrivant souvent en informatique. Un patron de conception* existe pour ce problème-là. Le patron de conception existant pour ce problème s'appelle le patron observateur

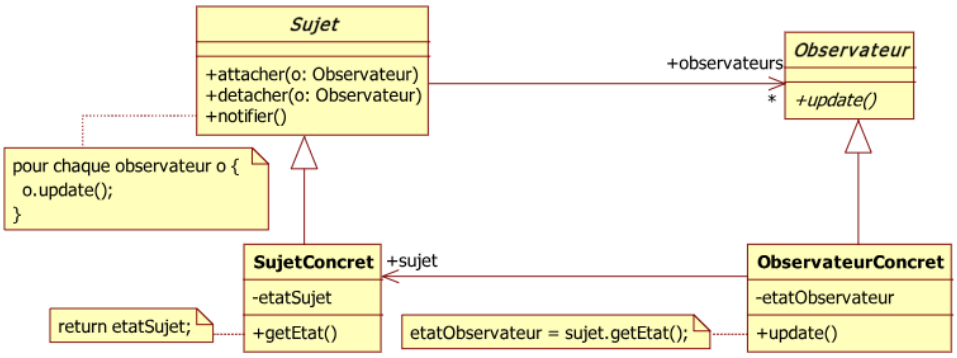


Figure 22 : Patron observateur

Ce patron a pour but de changer la façon d'obtenir les ressources. Au lieu de demander une ressource toutes les secondes, la ressource elle-même va avertir un observateur lorsque elle change. Il y a donc des requêtes à l'API que lorsqu'il y a un changement de valeurs, éliminant de ce fait les requêtes inutiles pour obtenir une valeur qui n'a pas changé.

Le problème est que dans notre cas, cela signifie que l'API doit nous prévenir or nous n'avons pas le contrôle de cette dernière puisque il s'agit d'une entreprise externe qui l'a gère.

Pour savoir si cette conception était possible, nous avons organisé un rendez-vous par visio-conférence.

Lors de cette communication, il s'est avéré que cette conception était possible pour certaines valeurs mais pas toutes. En effet certaines valeurs changent extrêmement rapidement, si ces valeurs notifier le logiciel lorsque elle change, cela surchargerait l'API au lieu de la décharger. Nous avons donc convenu d'une solution alternative avec l'entreprise gérant l'API fagor.

La solution retenue est de faire un mixe des 2 solutions : le chronomètre et l'observateur. Pour les valeurs ou il est possible de faire avec un patron observateur, nous ferons avec. Pour les autres, nous mettrons un chronomètre mais au lieu d'effectuer une demande à l'API pour chaque valeurs. Nous ferons une seul demande avec un array (structure de données composées de plusieurs éléments) contenant toutes les valeurs nécessaires. Faire une seul grosse demande étant moins couteux qu'effectuer de nombreuses petites demandes.

SAUVEGARDE DES PARAMETRES

Dans un souci de gain de temps et d'ergonomie, le logiciel doit être capable d'enregistrer les paramètres choisit par l'utilisateur (langue, fond d'écran du logiciel, briques sélectionnées…).

Pour cela, l'une des possibilités est l'utilisation d'une base de données permettant de stocker les données de manière organisé. Cependant le nombre de données à sauvegarder est faible et ne permet pas de justifier une telle mise en place et cela nécessiterait une connexion entre la machine et le serveur contenant la base de donnée.

J'ai donc décidé de faire comme le logiciel initiale, de sauvegarder les données dans des fichiers texte étant donné que cela est une solution fiable et rapide à mettre en place.

Un fichier texte de configuration sera également créer pour sauvegarder les paramètres inhérents à chaque machine (Certaines briques étant désactivé pour certaines machines).

MANAGER

Certaines valeurs doivent être connu par plusieurs parties du logiciel (Par exemple le fond d'écran doit être connu par toutes les vues en ayant besoin). Pour celles qui ne sont pas susceptible de changer, on utilise une classe statique contenant toutes ces valeurs définit comme des constantes.

Pour les autres variables on va utiliser une classe appelée Manager contenant les valeurs et fonctions devant être accessible partout. Celle-ci est définit dans la classe App qui est la classe principale de toutes les applications WPF responsable de la gestion globale de l'application.

Une partie de la communication avec l'API a été mit dans le Manager mais cela ne respectait pas le principe S des principes SOLID (Une seul responsabilité par classe. La classe Manager devenait donc peu lisible et donc peu maintenable.

Pour palier ce problème, une classe Gateway sera mise en place dans les 2 semaines de stage restant pour palier ce problème. Cela permettra également de pouvoir lancer l'application sans la communication avec l'API ce qui n'est pas possible dans la configuration actuelle.

BILAN TECHNIQUE

CE QUI A ETE REALISE

La structure du logiciel a pu être entièrement conçu et programmer durant les 8 semaines de stage passées ce qui permet au logiciel d'avoir une base solide sur lequel des ajouts ont pu être fait.

Des briques ont été conçus permettant de vérifier que la structure du logiciel et sa communication avec l'API est fonctionnelle.

Des tests réalisés ont correspondu aux attentes notamment au niveau de la fluidité du logiciel. Certains détails graphiques peuvent encore être améliorer.

Le logiciel n'est pas encore prêt à utiliser notamment parce que peu de briques ont été réaliser et que de nombreux tests doivent encore être fait pour éliminer tous bugs ou dysfonctionnements.

CE QU'IL RESTE A FAIRE

Durant les 2 dernières semaines du stage, les points suivant restent à développer ou à améliorer :

- La communication avec l'API n'est pas encore complètement terminé, comme dit précédemment une classe Gateway doit être mise en place pour améliorer la qualité du code. De plus la gestion du rafraichissement n'est pas encore terminé ce qui est nécessaire pour un grand nombre de briques.
- Des tests et de nombreuses corrections de bugs restent à effectuer pour éviter tout comportement imprévues
- De la documentation sera à réaliser pour permettre aux personnes reprenant le projet de comprendre plus facilement le projet.
- Des corrections graphiques peuvent encore être apportées pour que les graphismes du logiciel soit vraiment conforme à ceux de la précédente version.
- Si à l'issue du stage, tous les principaux éléments de cette liste on été terminées. Des briques supplémentaires seront ajoutées. Etant donné, que la structure des briques à été conçu leurs temps de développement est considérablement réduit.

-

CONCLUSION

Ce stage a constitué pour moi une première vrai expérience professionnelle. Tout d'abord il m'a permit de découvrir le milieu d'une entreprise et de découvrir la collaboration qu'il existe entre les différents secteurs (informatique, ouvriers, marketing…).

Ensuite ce stage m'a permit de découvrir certains points à améliorer. Notamment de prendre plus de temps à concevoir sur papier avant de coder. J'ai dut en effet à quelques reprise reprendre une partie de mon travail car la qualité du code n'était pas au rendez-vous, ce qui créer des bugs et complexifié le reste du développement. Cela m'a appris la rigueur que nécessite un projet tel que celui-ci.

Enfin, ce stage m'a permis de développer mon autonomie. En effet, si les projets et SAE de l'IUT m'ont apprit à travailler en équipe avec d'autres personnes du même domaine, il n'y avait pas de long projet où je devais travailler en solitaire dessus. Cela m'a apprit à m'organiser et à gérer mon temps.

Je suis dans l'ensemble satisfait de mon travail réalisé durant cette période étant dans les temps pour finir le projet.

ENGLISH SUMMARY

From April 23rd to June 30th, I did an internship at SOMAB company in Moulins. It is a company specialized in the design, production, and sale of machine tools.

My goal was to remake a software integrated into the machines built by the company: the Mab 2.0 software. The original software had major design issues, making the code difficult to maintain and causing a lot of bugs.

The software is used by machine users to simplify the utilization of the machines. They can change machine parameters, give an order or consult data or documentation about the machine.

A significant part of the project's objective was to achieve an efficient structure that can easily had improvements. The new software was developed using the WPF framework, with two languages: XAML for the graphical part and C# for the rest of the software.

During the first part of the internship, I have create the main pages of the software and design the structure. After this, I worked on the communication with the machine and implemented the corresponding functionalities.

The internship is not yet finished, so the software is not fully functional. There are still some bugs and graphical improvements to be upgrade. However, a significant part of the project has been completed, and the remaining time should be sufficient to finish it.

LEXIQUE

API :

Une API est un ensemble de protocole permettant à diverses applications de communiquer et d'échanger des données entre elles.

Patron de conception :

Les patrons de conception étant la meilleure solution donné à un problème de conception récurrent.