Table des matières

Table des illustrations

Remerciements

Je tiens tout d'abord à remercier mon maître de stage, Jean-Pierre Rellier, qui m'a donné de son temps et de sa patience pour m'accompagner, et m'expliquer en détails les tenants et aboutissants du stage.

Mes remerciements vont également à tout le reste de l'unité BIA Toulouse qui m'a accueilli durant ces cinq mois, et notamment à Nathalie Dubois-Peyrard et Iadine Chadès qui ont partagé leur bureau avec moi, et à Régis Sabbadin qui m'a encouragé tout au long du stage.

Je remercie particulièrement Roger Martin-Clouaire, directeur de l'unité, et Pascale Faure, assistante de direction, qui m'ont guidés dès mon arrivée dans l'unité.

Un grand merci à Eve, Tara et Nicklas qui m'ont permis de pratiquer mon anglais dans des discussions sur de nombreux sujets.

Pour finir, je tiens à remercier tous mes professeurs qui m'ont appris les connaissances sans lesquelles je ne pourrai commencer ce stage, et qui nous ont fourni les aptitudes et les conseils nécessaires à la réussite de la recherche de stage.

Le stage de fin de licence Miage a plusieurs buts premiers : appliquer des connaissances théoriques à la pratique, appréhender les problématiques de l'entreprise.
Pour ma part, un de mes principaux objectifs était tout d'abord d'observer, de voir la palette des métiers existants, de tester certaines des missions confiées aux informaticiens ; ce afin d'améliorer mon projet professionnel, et de m'orienter plus tard vers la spécialité qui me conviendrait le mieux.

J'avais précédemment effectué un stage dans une très grande entreprise de l'agglomération Toulousaine, EADS Astrium*[1], où j'avais travaillé dans un environnement de bases de données Access*. Voulant connaître différentes facettes du travail d'un informaticien, j'ai orienté mes recherches vers un travail d''analyse et de développement d'applications qui auraient pour but d'aider les utilisateurs. Ceci me semblait totalement correspondre à ma formation, basée à la fois sur de la gestion (de l'analyse des systèmes d'information, des prises de décision) et de l'informatique (outils nécessaires à la mise en place d'une application).

C'est dans cette optique que j'ai intégré le grand institut qu'est l'Inra*, avec un stage ayant pour sujet l'identification et l'intégration d'une fonctionnalité importante à un logiciel déjà existant.

Ce logiciel est en fait une interface (nommée SOLFEGE), basée sur un environnement de travail nommé DIESE, qui permet de créer des simulateurs, et dont le but est de représenter des environnements et des processus de nature agronomique, comme l'élevage de bovins et porcins en Bretagne, ou encore la pousse de tomates sous serres en Avignon.

Le modèle utilisé par DIESE a pour vocation à être générique, dans le sens où il est applicable à tous les systèmes de nature agronomique. Il utilise pour cela un méta-modèle, notion que nous détaillerons plus tard dans ce document.

Cet outil de modélisation et de simulation a pour but d'amener à une capitalisation des connaissances dans le milieu agronomique.

Mon rôle a été de réfléchir et de mettre en place une interface pour modulariser les modèles développés par les agronomes, afin que des parties de ceux-ci (ou modules) puissent être réutilisés par d'autres agronomes, pour une autre simulation, sans avoir à recommencer toute une partie de travail déjà effectuée par ailleurs, par une tierce personne.

Afin d'expliquer le travail que j'ai effectué durant mon stage, je commencerai tout d'abord par décrire

1 Les mots ou expressions suivis d'une astérisque sont expliqués dans le glossaire en fin de mémoire.

l'environnement de travail dans lequel mon stage s'est effectué. Je détaillerai ensuite la phase d'analyse que j'ai réalisée par rapport au logiciel existant et aux besoins à satisfaire dans la partie que j'ai développée. Ensuite, j'expliquerai la manière dont j'ai conçu et livré le logiciel final.

J'établirai par la suite une réflexion sur le rôle d'un chef de projet, selon ce que j'ai pu observer durant ces cinq mois.

Je terminerai ce mémoire par un bilan technique, professionnel et personnel de ce stage.

I. *Environnement de travail*

A) Présentation de l'Inra.

Mon stage s'est déroulé dans l'Institut National de Recherche Agronomique (INRA) de Toulouse, plus précisément à Auzeville Tolosane*. Situé près de l'agrobiopôle, du lycée agricole et de l'ENSAT*, l'Inra Toulouse s'ancre dans un contexte omniprésent de recherche agronomique.

L'Inra est un organisme public de recherche scientifique finalisée*, placé sous la double tutelle du ministère de l'enseignement supérieur et de la recherche et du ministère de l'agriculture et de la pêche. Ses recherches concernent les questions liées à l'agriculture, à l'alimentation et à la sécurité des aliments, à l'environnement et à la gestion des territoires, dans une perspective de développement durable.

L'Inra a été fondé en 1946. Il est aujourd'hui le premier institut européen de recherche agronomique. Son rôle est de :

- produire et diffuser des connaissances scientifiques
- concevoir des innovations et des savoir-faire pour la société
- éclairer, par son expertise, les décisions des acteurs publics et privés
- développer la culture scientifique et technique,
- participer au débat science/société
- former à la recherche et par la recherche.

Cet institut mène une politique active de partenariat avec les acteurs socio-économiques (entreprises, organisations collectives agricoles), les collectivités territoriales, les pouvoirs publics qui sollicitent l'expertise de ses chercheurs aux plans national, européen et international, les représentants de la société pour conduire des travaux de prospective et de vision stratégique et bâtir ainsi des programmes de recherches en France et en Europe pertinents pour la société.

L'Inra occupe le 2e rang mondial et le 1er en Europe pour les publications en sciences agricoles et en sciences de la plante et de l'animal. Il entretient des partenariats scientifiques avec les grands instituts de recherche scientifique dans le monde, les universités, l'enseignement agronomique et vétérinaire, et s'engage dans la construction de l'espace européen de la recherche. Il développe également de multiples collaborations et échanges avec la communauté scientifique internationale dans de nombreux pays en Europe, Amérique, Asie et Afrique.

B) L'unité BIA

Dans ce grand organisme, structuré en plusieurs centres de recherches, dont celui de Toulouse (600 employés environ), j'ai travaillé dans le département **MIA*** (Mathématiques et Informatique Appliqués), et plus particulièrement dans l'unité **BIAT*** (Unité de Biométrie et d'Intelligence Artificielle de Toulouse).

Le département **MIA** est une entité nationale, qui regroupe plusieurs unités à travers la France, ayant un sujet de recherche commun le développement des recherches méthodologiques dans différents domaines des mathématiques et de l'informatique, pour répondre aux grandes priorités de l'Inra concernant l'agriculture, l'environnement et l'alimentation. Pour cette raison, beaucoup de chercheurs effectuent des déplacements à travers la France, pour rencontrer leurs collègues des autres unités, avec lesquels ils travaillent souvent en partenariat.

L'unité **BIAT** a pour mission de développer et d'appliquer des méthodes de statistiques et d'intelligence artificielle dans le cadre des grands axes de recherche de l'Inra. Ces missions s'accompagnent d'une activité de formation et de développement logiciel. L'objectif de l'unité est de développer et d'appliquer des méthodes de statistique et d'intelligence artificielle dans :

- L'analyse et la gestion des systèmes de production et des ressources physiques
- L'analyse de la structure et de la fonction des génomes.

Plusieurs groupes de personnes travaillent au sein de l'unité BIA : la direction et l'administration, et deux groupes de chercheurs :

- Les modélisateurs d'agro-systèmes, équipe dans laquelle s'ancre la problématique de mon stage.
- Les biologistes des méthodes informatiques, statistiques et informatiques.

Dans ces deux groupes cohabitent de nombreuses personnes, appelées "non-permanents", par opposition aux titulaires : les post-doctorants, les doctorants, qui effectuent leurs thèse en compagnie des chercheurs, les employés en contrat à durée déterminée et les stagiaires. Lors de mon stage, nous étions une dizaine de stagiaires à travailler dans l'unité.

Sur la page suivante est un schéma simplifié de la hiérarchie de l'unité BIA. Il existe des statuts de chercheurs, en fonction de leur expérience et des études qu'ils ont effectuées. Les deux équipes de chercheurs travaillent avec l'aide de l'équipe administrative et informatique, qui n'ont de cesse de les épauler et leur faciliter le travail. En plus de toutes ces personnes travaillant spécifiquement pour l'unité, de nombreuses autres, travaillant en partenariat avec un ou plusieurs chercheurs de l'unité, effectuent des séjours dans l'unité, d'une longueur variable d'une journée à un an.

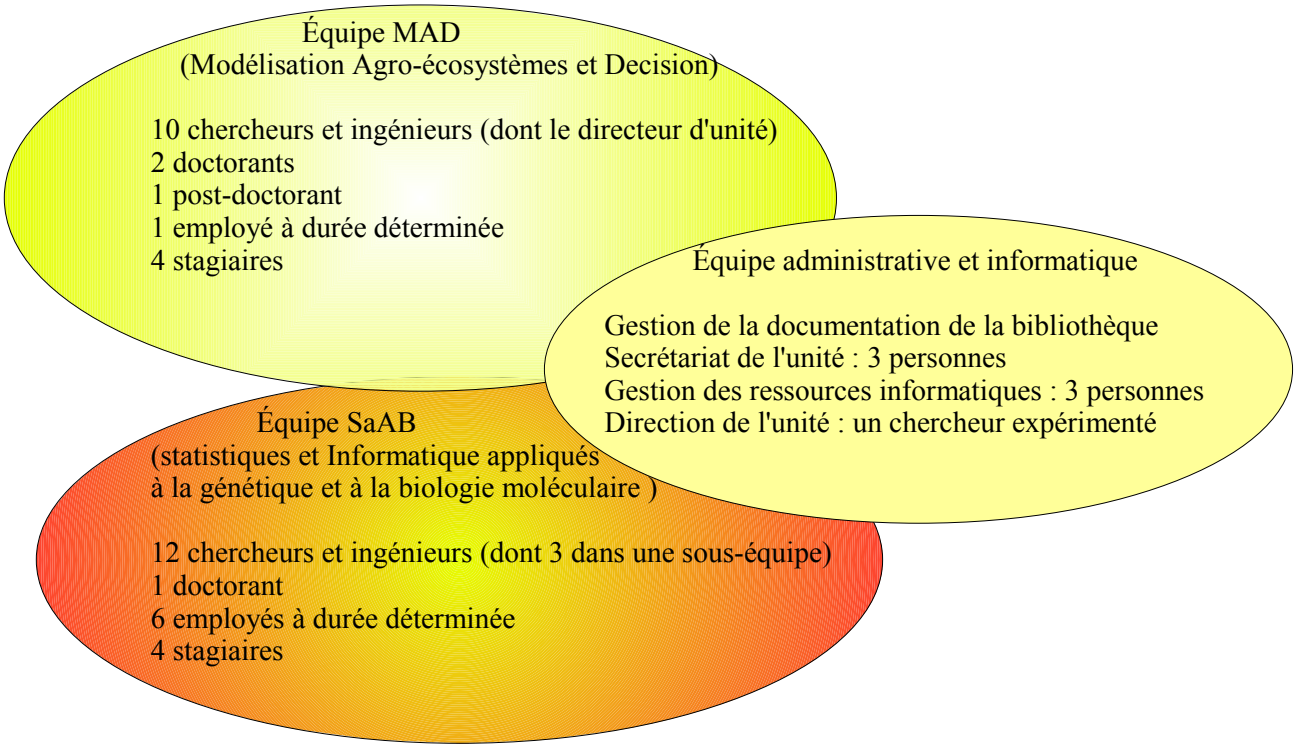


Fig 1 : Schéma des équipes de l'unité BIAT

C) Environnement matériel

Pour effectuer ce stage dans de bonnes conditions, il m'a été affecté un bureau, que j'ai partagé avec deux chercheuses de l'Inra, ainsi qu'un poste de travail.

J'ai travaillé sur un terminal léger, connecté sur le serveur d'applications de l'unité. Ce client fonctionnait sous le système d'exploitation Unix Debian Thunar (Thunar est un projet visant à développer un nouveau système de gestion de fichier pour l'environnement de bureau Xfce*) . Les données étaient elles, stockées de manière transparente sur un autre serveur, de sorte que l'on puisse y accéder du serveur d'applications. De ce client léger, je pouvais me connecter à un autre serveur pour exécuter les applications lourdes lorsque j'en avais besoin, afin d'éviter d'encombrer le serveur, grâce au protocole ssh*.

D) Planning

La première semaine de mon travail a été consacrée à l'étude de l'outil existant. J'ai relu les documents descriptifs de l'application et en ai compris la teneur. Après que mon maître de stage m'a montré le fonctionnement de l'interface SOLFEGE et de l'outil en lui-même sur un « cas d'école » - le fonctionnement d'un toboggan- , j'ai moi-même développé un exemple simple, pour me familiariser définitivement avec l'outil.

Durant le deuxième mois, j'ai commencé à étudier les besoins. Tout d'abord, en essayant d'analyser le concept de module, puis en essayant d'écrire une définition générale et exhaustive de cette notion. De cette analyse est issu un document de conception, joint en annexe.

Les trois derniers mois ont été consacrés au développement de la nouvelle interface. Le dernier mois a servi à développer les points plus "sensibles" de l'application, notamment une réflexion sur la mise en place d'une gestion de références, ainsi que des outils de transfert de classes du type copier/coller (cf p. 31)

La fin de ce dernier mois a été réservée pour les tests, le déboggage et l'écriture des documents à l'attention des utilisateurs et des développeurs, ainsi qu'à la finition des documents universitaires, développés tout au long du stage.

Au cours du stage, le planning fixé initialement a changé. En effet, malgré le travail de conception préliminaire, de nouvelles fonctionnalités, problèmes et débordements sont apparus au cours du temps, nous contraignant à faire glisser le plan d'avancement.

Voici le diagramme de Gantt montrant la planning prévisionnel et le planning effectué.

Plusieurs jalons et points de rencontres ont été effectués au début, au cours et à la fin de chaque étape, afin d'évaluer l'avancement au cours du temps. Ils ont consisté en de fréquentes réunions avec mon maître de stage - à raison d'au moins une par semaine - et en la rédaction de documentations techniques.

	temps	AVRIL				MAI				JUIN				JUILLET				AOUT				
Opération réalisée	estimé	1	2	3	4	5	6	7	8	9	1	1	12	13	14	15	16	17	18	19	20	21
Analyse existant	2										0	1										
Applications	1																					
Analyse futur	1																					
Conception	3																					
Squelette Module	1																					
Interface	2																					
Système Fichier	1																					
Sauvegarde Module	3																					
References	2																					
Tests unitaires	2																					
Tests intégration	1																					
Recette	0,5																					
Documentation	2																					
Mémoire de stage	20																					
Diaporama stage	2																					

Fig 2 : Diagramme de Gantt du planning du développement de l'application

LEGENDE

prévisionnel

réel

Le planning des réels s'arrête l'avant dernière semaine du stage, date de remise de ce présent dossier.

II. Bilan de l'étude des outils existants, DIESE et SOLFEGE

A) Existant

J'ai commencé par étudier le principe de mon nouvel environnement de travail, à l'aide d'un document résumant le fonctionnement de la bibliothèque DIESE. Il m'a fallu du temps, et de nombreuses relectures pour comprendre la teneur et l'intérêt de cet environnement. Pour pouvoir expliquer de manière simple l'intérêt de mon stage, je me dois de résumer et de décrire les principes de base de l'existant.

a) Intérêt de DIESE

L'outil DIESE est un outil permettant de modéliser et de simuler des systèmes d'intérêt agronomique. Basé sur la représentation à évènements discrets - c'est-à-dire un évènement qui survient à un moment donné-, cet outil permet également de représenter la continuité d'un processus dans un système -c'est-à-dire l'enchaînement de plusieurs actions répétitives. Ces actions se produisent régulièrement, selon un certain pas.- Ce sont donc deux types de processus qui cohabitent, le processus dits discrets, par le fait qu'il n'interviennent qu'à un moment donné, et les processus dits continus, qui interviennent durant une période sur le système. Pour lancer ces processus, il est nécessaire qu'un évènement en donne "l'ordre". Cet évènement peut être unique, ou bien engendrer des répétitions de lui même -c'est le cas notamment du lever du soleil-. Chacun de ces évènements influera sur la dynamique du système, c'est-à-dire fera évoluer un processus qui agira sur celui-là. A titre d'exemple, on pourra représenter et simuler la culture d'une céréale dans un certain environnement. Ainsi, cette culture est conditionnée par plusieurs processus ; la pousse de la céréale, la moisson. Alors que la moisson est un processus que l'on peut représenter de façon unique, donc discrète, la pousse du céréale est un ensemble d'avancements répétitifs, lancés par un premier évènement : ensemencement terminé.

Les chercheurs de L'Inra étudient certains systèmes et souhaitent les modéliser, c'est-à-dire les représenter sous forme de modèle informatique. L'environnement DIESE leur propose un outil pour effectuer l'analyse systémique de leur modèle, à travers une conception orientée objet et une représentation par classes. Ces classes seront des processus, des évènements et des entités, représentant les différents objets du système étudié. Le noyau principal de l'environnement DIESE est la bibliothèque de ces classes principales, et d'autres classes complémentaires. Il se nomme BASIC DIESE, et est codé en C++.

Pour reprendre l'exemple précédent, les chercheurs, souhaitant simuler la culture du blé dans la région Midi-Pyrénées, devront d'abord identifier les principaux éléments. Ainsi, ils pourront les représenter grâce aux classes de la bibliothèque C++ BASIC DIESE. D'une analyse préliminaire de cet exemple pourront apparaître plusieurs entités : les céréales, le sol sur lequel elles poussent, voire même le temps qu'il fait. Le chercheur mettra également en exergue les processus qu'il souhaite simuler : la récolte de ce blé, ou alors l'ensemencement... Il identifiera ces processus dans des classes, héritées des classes de DIESE. Enfin, il lui faudra étudier quels évènements sont la raison du déclenchement d'un processus. Il s'agira peut-être d'un instant (date ou moment induit par la conséquence d'un processus), ou du lever du soleil, etc.

D'après ce premier exemple on peut donc comprendre que DIESE est un modèle permettant de représenter tout système, notamment les systèmes agronomiques, dans le but d'en effectuer des simulations.

DIESE est le premier outil dédié à la modélisation de tous les types de systèmes agronomiques. Cet outil n'a de cesse d'évoluer depuis plus de dix ans, pour satisfaire toujours plus les clients. Durant ce même laps de temps ont été développées d'autres applications en relation avec DIESE. Il s'agit notamment d'une interface d'aide à la modélisation décision des chercheurs -nommée SOLFEGE -,mais également d'une interface de simulation -nommée MI_DIESE-. Une seconde bibliothèque de DIESE a également été créée, CONTROL DIESE, qui est un ensemble de concepts et d'algorithmes pour piloter un système.

Il existe actuellement plusieurs utilisateurs de l'application, travaillant sur des sujets variés :

- équipe de Rennes : impacts environnementaux des systèmes mixtes laitiers-porcins
- équipe AGIR de Toulouse : Systèmes d'élevage et diversité des prairies dans les territoires herbagers.
- équipe de Montpellier : Pratique de l'enherbement dans les systèmes de culture de la vigne.
- équipe de Grignon : Conditions de l'insertion durable du Pois dans les systèmes de grande culture.
- équipe EcoDeveloppement d'Avignon : Gestion du climat dans les serres agricoles.

DIESE a pour but final d'être utilisé par une plus grande communauté de chercheurs.

Toute la documentation de DIESE est accessible aux modélisateurs. Elle a été créée sous forme d'une page html, générée à l'aide de l'outil Doc++*. Pour mettre en place ce genre de documentation, il faut décrire sous forme de commentaires chaque fonction et attribut d'une classe. L'outil Doc++ analyse le code et utilise ces commentaires spéciaux, les met en page, de sorte que chaque modélisateur connaissant l'adresse de cette documentation puisse avoir accès aux définitions détaillées de toutes les classes de la bibliothèque DIESE. C'est de cette manière qu'il pourra les utiliser à bon escient.

b) *Fonctionnement*

L'approche de l'outil DIESE est basée sur un méta-modèle*. Un méta-modèle est une abstraction du modèle habituel classe/attribut/fonction, dans le cas du modèle objet sur lequel est basé DIESE. Le choix du méta-modèle apporte de nombreux avantages :

- création d'un formalisme dédié aux simulations d'évènements agronomiques.
- « filtre » sur le modèle considéré, extrait des éléments génériques selon le point de vue adopté.
- réutilisabilité et généricité des classes principales : on n'a que peu de choses a redéfinir dans chaque classe. En fait, de nombreuses méthodes sont prédéfinies dans les classes principales, que nous allons résumer ci-après. Par exemple, tous les accesseurs et modificateurs sont ainsi codés de sorte que le modélisateur n'ait pas à les redéfinir.

L'environnement DIESE tend donc à abstraire la notion de classe et se base sur le paradigme suivant :
Tout système de nature agronomique peut être représenté par un ensemble d'entités, de processus et d'évènements. Ces trois types d'objets représentent les classes principales de l'environnement DIESE.

Chacune de ces trois classes est composée d'un ensemble d'attributs statiques et fonctionnels. Ce sont ces mêmes attributs qui vont être abstraits du système : un attribut statique (ou descripteur) sera représenté, dans l'optique du méta-modèle, par une classe à part entière. De la même manière, une méthode, ou attribut fonctionnel, aura lui-même une classe le représentant.

Voici à quoi ressemble le diagramme de classes de ce méta-modèle :

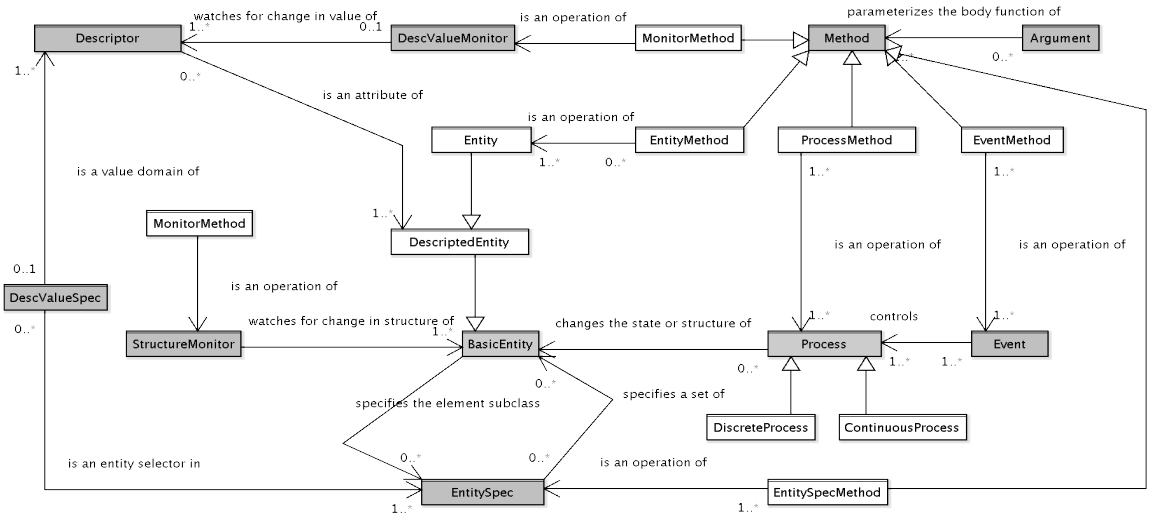


Fig 3 : diagramme des classes UML simplifié de la bibliothèque DIESE

C'est avec un diagramme de classes représenté avec la méthode OMT*, très ressemblant à un diagramme UML, que j'ai analysé la structure de la bibliothèque BASIC DIESE. Cependant, ce diagramme n'a pas été conçu pour concevoir l'application, mais pour la représenter. Il n'est ici qu'un outil de communication entre les différents développeurs de DIESE, et non pas un outil de modélisation de celui-ci.

Comme on peut le voir sur ce schéma de la figure 3, plusieurs classes cohabitent avec les trois classes principales, citées ci-dessus. Pour exemple, nous voyons qu'une méthode peut s'appliquer à la fois à une entité (entity), un processus (process), ou un évènement (event).

J'ai par la suite dû entrer dans le détail de ces classes pour en comprendre le fonctionnement. Un résumé du contenu de ces classes est nécessaire pour la compréhension de mon travail, aussi je simplifierai ce contenu.

Il est nécessaire de comprendre que le contenu de ces classes est fixé de telle sorte que n'importe quelle classe puisse en hériter. C'est ce qui fait le caractère générique de DIESE et toute la force de ce méta-modèle.

Une **entité**, c'est-à-dire un élément perceptible du modèle (une tomate, le sol, une serre...) est représentée par la classe Entity. Ce que les concepteurs analyseront comme étant un *élément structurel* sera modélisé dans cette classe.

Comme toute classe, la classe Entity possède un ensemble d'attributs et de méthodes propres. Ces informations génériques à la classe Entity sont donc réutilisables par tout élément héritant de la classe Entity. Les principaux attributs détenus par la classe, qui sont représentés par des objets à part entière, sont :

- une liste de descripteurs (ce qui pourrait correspondre aux attributs d'une classe dans le raisonnement objet sans méta-modèle).
- une liste de méthodes (ce qui correspondrait aux méthodes dans une classe normale).
- une liste d'éléments ensembliste, c'est-à-dire que tous les éléments de la liste sont d'une même nature (un sac de bille), ce qui implique que le nombre d'éléments n'affecte pas la nature du contenant .
- une liste de composants. Ce sont les composants qui donnent sa nature à l'élément (on pourra citer une voiture qui est composée de roues, d'un volant, d'un moteur, sans lesquels elle ne serait pas une voiture.)
- La classe mère, s'il y a lieu.
- la liste de toute les entités dont elle est un élément (vide s'il n'y en a aucun).
- La classe qu'il compose, si besoin (c'est-à-dire le sac si l'entité est la bille).

Un **processus** est une fonction du système, quelque chose qui fait évoluer son état, grâce au déclenchement d'un évènement. Il y a deux types de processus : les processus ponctuels (discrets) et les processus continus, suivant la manière de faire progresser le système (instantanée ou progressive). Pour les différencier, il existe deux classes, DiscreteProcess et ContinuousProcess, toutes deux héritées d'une classe-mère Process. Cette classe a pour attributs, entre autres, une liste de méthodes spécifiquement applicables aux processus; et telle que la fonction de transition d'état.

Un **évènement** est un élément dynamique permettant d'actionner un changement du système. Il intervient sur un processus. Sa date et les directives qu'il provoque sur le processus sont ses principaux attributs. Plusieurs évènements peuvent intervenir sur un processus, et un évènement peut intervenir sur plusieurs processus. Tous les éléments sont stockés dans un "agenda", qui exécute les tâches les unes après les autres, en fonction de l'instant programmé de lancement de chacun des évènements.

Une **fonction -** d'une de ces trois précédentes classes représentant le système - est modélisée par la classe Method. Elle a pour attributs son type de retour, sa liste de paramètres ainsi que le corps de la méthode. Elle a bien sûr comme méthodes des accesseurs aux arguments.

Un **attribut** est représenté par la classe Descriptor, qui possède comme attributs:

- un type
- un domaine de valeurs
- une valeur

Cette classe a une méthode qui vérifie que la valeur correspond au domaine de valeurs. Pour les Attributs susceptibles de changer au cours du temps, on pourra associer un démon* qui programmera une réaction spécifique à un dépassement de seuil, par exemple (si une voiture dépasse la vitesse de 150...).

Un **paramètre** d'une fonction est représentée par une classe Argument, dont la syntaxe est proche de la classe Descriptor. Un paramètre s'applique à une méthode.

Pour résumer, on abstrait le fonctionnement du système, de sorte qu'on se situe au dessus du modèle objet, selon la vision que l'on en a classiquement. Pour bien comprendre cette notion, voici une représentation schématique de l'exemple développé : la culture du blé.

Dans cet exemple, nous représentons d'abord l'entité Blé. Elle a pour descripteurs une taille, une race et un état. Ces deux descripteurs font partie de la liste d'attributs statiques de l'entité Blé. Cette entité a également pour méthode une fonction qui à partir de ces attributs, calcule l'état dans lequel est la pousse de blé, afin de savoir s'il est temps ou non de le moissonner. L'entité Blé est une spécialisation de la classe-mère Entity. On identifie ensuite un processus simple, la pousse de ce blé, déclenché chaque jour par le lever du soleil (en simplifiant). Ainsi, l'architecture conceptuelle de ce système avec l'aide de DIESE peut être représenté ainsi.

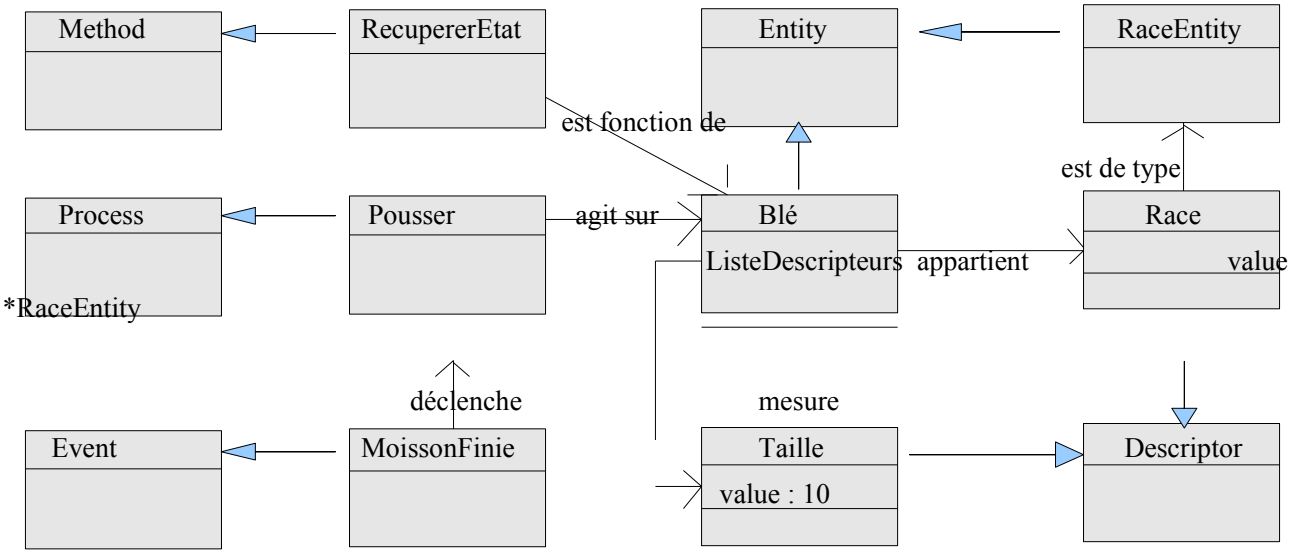


Fig 4 : Diagramme des classes UML du système "pousse du blé"

c) *L'interface* SOLFEGE

DIESE est doté d'une interface graphique, appelée SOLFEGE, permettant aux concepteurs d'un système de créer leur simulateur très facilement.
Cette interface permet aux modélisateurs une approche simplifiée de DIESE et un développement accéléré. Lorsqu'il souhaite définir les différentes classes formant son simulateur, le développeur est en effet aidé par SOLFEGE. Il lui suffit de pousser des boutons, de choisir dans des listes, pour créer un premier squelette structurel de son simulateur.

Au final, le développeur peut certes être amené à développer du code, notamment pour les changements d'états caractérisant un processus, mais ce développement est final, et s'effectue donc dans un univers dont la structure est déjà établie. En effet, SOLFEGE génère le code architectural du simulateur (déclarations, en-têtes, pattern des réalisations, constructeurs).
Il existe trois types de développement : le développement structurel, dont la totalité est effectué par SOLFEGE, le développement fonctionnel, qui correspond au codage des méthodes. Enfin, il existe un niveau dynamique, c'est-à-dire les éléments qui font évoluer le système (tels que les processus). Ces éléments ne sont pas donnés dans la modélisation, mais plus tard, lors de la simulation.

Voici par exemple une des fenêtres de SOLFEGE, qui est en fait sa fenêtre principale. Elle présente plusieurs menus pour développer, générer et préparer la livraison d'un simulateur.

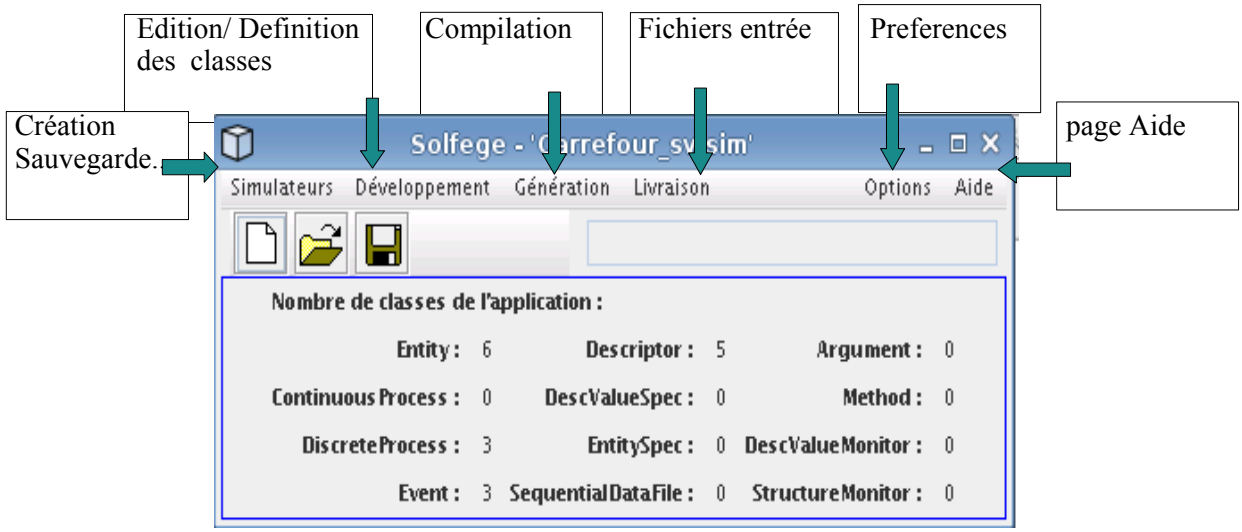


Fig 5: Fenêtre principale de l'outil SOLFEGE*, lorsqu'un simulateur est ouvert*

Ces menus permettent de visualiser et modifier le contenu d'un simulateur.

Le travail de développement et d'utilisation du simulateur est prévu pour être effectué par deux personnes différentes. Le développeur modélise et développe le simulateur du système, grâce à l'interface. Lorsque le système est modélisé, un nouvel acteur utilise le simulateur. Ce nouvel acteur, le client de l'application, est le plus souvent un chercheur qui souhaite utiliser le simulateur pour faire des tests sur le cas particulier du système qu'il souhaite étudier. Il peut donc exister plusieurs clients qui travaillent sur des cas différents.

Pour effectuer une simulation, deux moyens existent. On peut d'abord lancer la simulation avec une commande effectuée sous un terminal. On peut également, pour avoir une meilleure vue sur l'évolution du système, et pouvoir agir en temps réel sur les évènements ayant lieu durant la simulation, utiliser une autre interface graphique, appelée MI_DIESE*. Cette interface graphique est d'applicabilité générale (c'est-à-dire peut manipuler tout simulateur créé dans le cadre de DIESE).

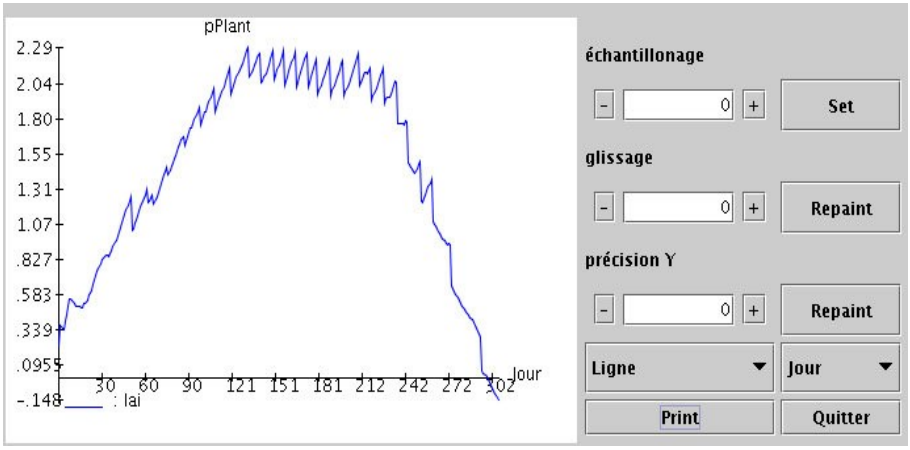


Figure 6 : Exemple d'utilisation de l'interface MI_DIESE

Quel que soit le mode de simulation choisi par le chercheur, il est important de faire passer au simulateur des données dites d'entrée. En effet, pour que le simulateur créé soit utilisable pour n'importe quel sous-système de celui modélisé, c'est seulement les données structurelles et fonctionnelles, qui sont générales à tout le système, qui sont des éléments du simulateurs. Il est donc nécessaire de faire passer les données spécifiques au début et au cours de la simulation.

Pour exemple, si l'on souhaite simuler la culture du blé, il sera nécessaire de donner en paramètre du simulateur le nombre de pieds/m² de blé, leur race, leur taille de départ, etc. On peut aisément comprendre que ces données ne sont pas génériques, car elles peuvent changer d'une région à une autre, voire même d'un champ à l'autre.

Pour passer ces données, ces "paramètres" au simulateur, des fichiers d'entrée sont lus lors de la simulation. Ce sont ces fichiers d'entrée qui représentent les données, écrites dans un pseudo-code et compréhensibles à la fois par le chercheur (qui n'est normalement pas un développeur) et par le simulateur.
Ce pseudo-code est un langage défini à l'aide de l'outil lex* et interprétable par un compilateur généré par yacc*.

d) Objectifs et contexte de développement

On souhaite remanier l'interface SOLFEGE, qui sert à modéliser un système pour créer un simulateur. L'objectif de ce stage est de fournir aux chercheurs des moyens de regrouper par modules des morceaux d'un simulateur. On veut garantir la réutilisabilité de ce code. En effet, de nombreuses fonctionnalités peuvent servir dans des simulations pourtant bien différentes, et à différents endroits (que ce soit dans un département de l'Inra Toulouse ou dans d'autres unités). Les modules créés auront donc pour but d'être échangés et réutilisés par ailleurs.

Il a donc fallu réfléchir non seulement sur ce concept de module, mais également sur la manière d'adapter cette nouvelle fonctionnalité à l'interface SOLFEGE existante, ainsi qu'à la manière dont ces modules seraient stockés dans le système de fichiers.

L'interface est codée en Java, ainsi que l'extension que j'ai développée, après une analyse complète, afin de trouver la solution la plus ergonomique et simple pour les utilisateurs.

B) Applications Véhiculaires et Conclusions

A la suite de la première analyse des documents et des outils fournis par DIESE, j'ai effectué le travail d'un modélisateur, utilisateur de l'environnement DIESE, et plus spécifiquement l'interface SOLFEGE. Cette mise en application m'a permis de me familiariser avec l'outil, de ne plus le voir d'un simple point de vue théorique, mais de voir son utilisation et de comprendre le fonctionnement qu'en font et qu'en feraient par la suite les utilisateurs, ce afin d'envisager plus tard l'ergonomie de ce que j'allais mettre en place. Cela m'a également permis de comprendre les derniers points flous concernant mon analyse de l'existant, car il est difficile de voir et comprendre les moindres détails d'une application et d'un concept si dense en un temps si court de manière uniquement théorique.

a) Le toboggan

Tout d'abord, j'ai observé Jean-Pierre Rellier, mon maître de stage, développer une première application, afin que je prenne note de la phase d'analyse et de conception auxquels sont soumis les modélisateurs d'une simulation. Nous avons pour cela étudié le fonctionnement d'un « toboggan ». En fait, il s'agit d'une route sur laquelle évolue une voiture. Nous étudions ses différentes positions, ainsi que sa vitesse au cours du temps. La voiture peut se déplacer de droite à gauche et de gauche à droite, comme on peut le voir sur ce schéma.

Le sujet complet de l'exercice effectué est donné en annexe 1.

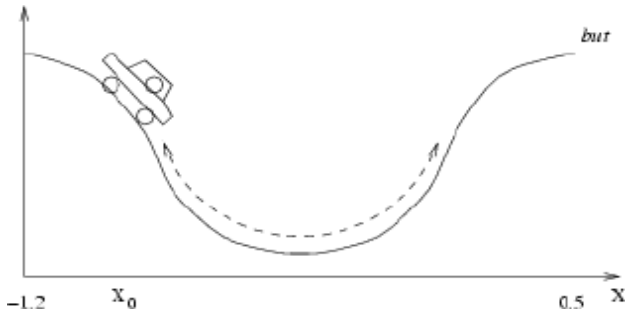


Fig 7 : Représentation du système étudié : un toboggan

Grâce au modèle de la bibliothèque DIESE. la méthode de modélisation du système étudié a été très simple. Nous avons tout d'abord commencé à extraire de la lecture du sujet (voir annexe) les entités, leurs attributs (nommés descripteurs dans l'environnement DIESE), les processus et les évènements agissant sur le système. A plusieurs reprises, nous avons dû faire des choix de modélisation, car plusieurs solutions se proposaient à nous. Il est donc à noter que, bien que DIESE aide et guide la conception, il appartient au modélisateur de faire les choix stratégiques concernant sa future application, ce qu'il est le plus à même de faire puisqu'il s'agit d'informations sur son domaine de compétences.

Une fois ce travail théorique effectué, nous avons commencé à utiliser SOLFEGE afin de créer les classes correspondantes. L'interface, couplée à la bibliothèque DIESE, nous a "mâché" le travail, c'est-à-dire qu'elle a retranscrit en code structurel les informations que nous lui avions données par l'intermédiaire de l'interface graphique, c'est-à dire qu'elle a créé toutes les classes nécessaires au fonctionnement de la simulation. Ne restait donc plus qu'à gérer l'enchaînement des évènements, le choix des données sur lesquelles nous souhaitions travailler. Nous nous sommes donc mis dans le rôle de l'utilisateur du simulateur et avons choisi quelles données nous mettrions dans ces fichiers d'entrée. Cela se résuma simplement, dans notre cas où l'application est simple, à une voiture, un pilote, et les évènements qui agiraient sur les processus afin de faire évoluer le système. Nous avons instancié des objets avec des valeurs simples, en écrivant en pseudo-code la création de ces instances, puis nous avons lancé la simulation.

Comme j'ai pu le constater lors de cette mise en situation, les outils DIESE et SOLFEGE permettent un réel gain de temps lors de la programmation d'une simulation, gain qui pourrait encore être amélioré en permettant la modularisation du travail des concepteurs.

Cette première approche d'une conception avec SOLFEGE m'a permis de connaître les commandes importantes pour simuler, mais également d'observer le fonctionnement de l'interface et de commencer à visualiser une solution au problème mis en exergue dans mon sujet de stage. Cela m'a aussi permis de comprendre les derniers détails du fonctionnement du système.

b) Le carrefour

Durant quelques jours, je me suis appliquée à réfléchir sur un second exemple, pour lequel je devais cette fois-ci trouver seule la solution.
L'énoncé de l'exercice (cf Annexe 2) se résume ainsi :

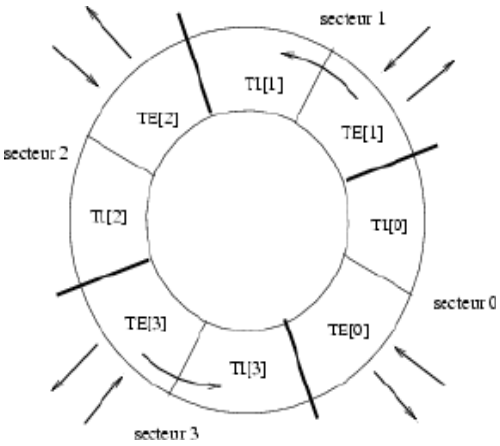


Fig 8 : Schéma d'un carrefour, système que j'ai étudié et modélisé avec SOLFEGE

Sur un rond-point à N entrées, découpé en N secteurs circulent des voitures. Chaque secteur est divisé en un tronçon interne et un tronçon d'échange (par où les voitures entrent et sortent). Chaque voiture a une vitesse qui leur est propre.

Il faut prendre en compte l'état actuel du carrefour (les voitures déjà présentes sur le rond point par exemple) pour faire circuler chaque voiture de son point de départ à son point d'arrivée. En fait, une voiture ne peut évoluer (entrer / avancer) dans le carrefour que si la totalité du secteur précédent son entrée (tronçon interne et le tronçon d'échange par lequel elle rentrera) sont libres. De même, une voiture peut avancer, ou doubler, s'il reste de la place dans le tronçon suivant (soit il est vide, soit il a plusieurs voies avec une seule voiture).

Grâce à SOLFEGE, j'ai pu modéliser ce problème:

- -Création de sous-classes de Entity:
    - Carrefour
      éléments : N secteurs
    - Secteur (composé d'1 tronçon d'échange et d'un tronçon interne
      attributs : numéro
    - Tronçon
      éléments : m voitures
    - Tronçon d'échange (hérite de tronçon)
    - Tronçon interne (hérite de tronçon)
    - Voiture Attributs :
      vitesse
      tronçon d'entrée (de type tronçon échange)
      tronçon de sortie (idem)
      tronçon courant (de type tronçon)

Comme expliqué plus tôt, les attributs sont définis dans une sous-classe de Descriptor (cf. page 15). Leur type et leur domaine y sont donnés.

J'ai représenté ma réflexion, pour plus de clarté, par un diagramme de classes UML. Je me suis par la suite axée sur ce diagramme pour réaliser mon simulateur.

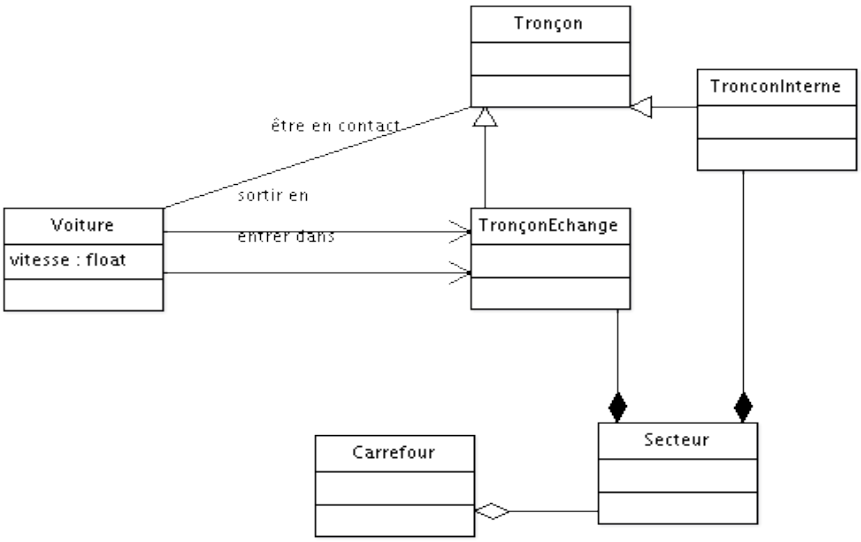


Figure 9 : Diagramme des classes de l'application "Carrefour" avec ma propre conception

J'ai décidé de modéliser ce fonctionnement par processus discrets, c'est-à-dire qu'on ne simule pas chaque instant de la situation mais seulement les moments "importants". Ils se résument ici au changement de contenant de la voiture (tronçon), c'est-à-dire lorsque la voiture se déplace (entrée, passage d'un tronçon du rond point à l'autre, sortie).

- Création des évènements :
    - arrivée d'une nouvelle voiture
    - voiture doit avancer
    - voiture devant sa sortie

- Création des processus :
    - entrée Voiture
    - avancement de la voiture
    - sortie de la voiture

Chaque voiture aura un évènement d'arrivée, de sortie, et k évènements d'avancement en fonction de sa voie d'entrée et de sortie. Chaque évènement déclenche le processus correspondant et provoque le changement du système.

Je n'ai dans cette application utilisé que les classes les plus courantes de SOLFEGE et me suis familiarisée non seulement avec leur contenu, mais aussi avec leur logique de codage. J'ai également pu percevoir et comprendre comment travaillaient les modélisateurs, pour visualiser le type d'interface que je devais mettre en place.

Il faut bien comprendre que pour arriver à ce stade j'ai travaillé en trois phases :

- Une première phase d'identification du système, qui constitue en une réflexion et/ou une représentation. Durant cette phase, le concepteur est libre de ses choix de modélisation.
- Une seconde phase de création du modèle, informatiquement.
- Une dernière phase de simulation.

Ces trois phases ne sont pas forcément linéaires, dans le sens où l'on peut être amené à simuler le système pour détecter des possibles erreurs de création ou de modélisation.

Cette application a donc été bénéfique pour ma compréhension de l'outil, et a clôturé la phase d'analyse; mais elle a également été utile au développeur de SOLFEGE, mon maître de stage Jean-Pierre Rellier, puisqu'il a pu voir en temps réel les soucis que pouvaient rencontrer les utilisateurs en effectuant des manipulations non répertoriées amenant à des situations non prévues. Ainsi, le fait que je sois une débutante dans l'utilisation de l'interface, tout comme le sont certains des utilisateurs de celle-ci, a permis de corriger et d'améliorer certains aspects , afin de rendre le logiciel plus robuste.

c) Réflexion préliminaire au sujet de la modularisation

Pour finir cette première phase d'analyse, j'ai réfléchi à la façon dont il serait possible de mettre la modularisation en place sur ces précédents exemples.

Je me suis très rapidement rendue compte que de nombreuses modélisations d'un système étaient possibles, qu'elles fonctionnement bien ou non, qu'elle aient une sémantique juste ou pas, et que tout cela appartenait finalement au modélisateur de faire le choix le plus juste en fonction de ce qu'il souhaitait représenter.

Une des premières contraintes du développement de cette interface est donc la liberté de développement du modélisateur qu'il faut préserver.

J'ai cependant retenu des solutions pour aller plus en profondeur du concept de module et voir comment ceux-ci pourraient fonctionner. J'ai ainsi identifié dans l'application "Carrefour" trois grands modules :

- un module "voiture", avec ses descripteurs
- un module "carrefour", avec les secteurs qui le comprennent
- un module "circulation", avec les divers processus et évènements.

On se rend très vite compte que chacun de ces modules a besoin d'interactions externes pour pouvoir faire changer le système. Ainsi le module circulation s'applique à un véhicule (que ce soit une voiture ou autre) sur une voie circulatoire (qu'il s'agisse d'un rond point ou non).

Il faut bien sûr que chaque module soit doté de sens. Parfois certains modules nous semblent justes, mais ne peuvent cependant pas s'adapter à n'importe quelle situation, du fait de la façon dont ils ont été développés. C'est le rôle du modélisateur de bien définir ce que fait un module et son domaine d'application.

III. Réflexion et nouvelle conception de SOLFEGE

A) Notion de Module

Après l'analyse des deux exemples d'utilisation précédents, il a fallu commencer à envisager d'écrire une définition du concept du module. Armée de mes premières expériences, et d'un document préliminaire que Jean-Pierre Rellier avait réalisé, j'ai confronté les théories, puis j'ai présenté une approche à mon maître de stage.

Au cours de cette analyse des besoins et des réunions d'échanges d'idées avec mon maître de stage, nous sommes passés d'un première définition qui était à mes yeux très théorique à une approche plus mathématique et systématique. Cette définition s'est aiguisée jusqu'à ce que nous soyons d'accord avec une conception qui soit à la fois représentative de la réalité, et ayant une mise en oeuvre aisée.
Nous nous sommes basés sur des modules qui avaient été envisagés et représentés, pour tester l'efficacité de notre définition. Je me suis ensuite appliquée à modéliser de façon plus physique une maquette de cette modularisation, pour voir concrètement les bases de mon futur travail.

Tout d'abord, j'ai dû prendre en compte dans ma conception les diverses contraintes qui m'étaient fixées, que ce soit par l'entreprise ou par mon environnement de travail. Les contraintes données par mon maître de stage ont permis d'assurer la cohérence de développement avec la version actuelle de SOLFEGE. Il s'agissait dans un premier temps d'étendre l'interface Java existante, ce qui implique que le développement s'est effectué avec le langage Java. L'outil a continué à se baser sur le même méta-modèle codé en C++.

La seconde contrainte, importante pour que les utilisateurs voient un intérêt à la modularisation, est la compatibilité inter-modules : il faut qu'il soit rapide pour un développeur d'adapter un module à son application, car si ce temps dépasse le temps qu'il aurait mis à créer lui-même les classes correspondantes, la modularisation ne remplit pas son rôle premier. Enfin, l'application devra rester portable, pour être utilisée par n'importe quel chercheur, ayant les connaissances nécessaires, sur n'importe quel système (En fait, SOLFEGE n'est actuellement compatible qu'avec les systèmes d'exploitations Unix et Microsoft, mais ceux-ci sont utilisés sur la quasi-totalité des ordinateurs au sein de l'Inra).

Pour répondre à l'objectif de mon stage, après avoir identifié les modules, j'ai dû intégrer cette notion et livrer le nouvel outil créé.

B) Besoins d'utilisation

On veut permettre aux utilisateurs de créer des modules. Pour exemple, dans le cas des applications du précédent chapitre, on pourrait voir apparaître dans un module « voiture », l'entité voiture, les descripteurs relatifs à l'entité, ses méthodes, des processus qui pourraient agir dessus. Le but de la mise en place de ces modules est bien sûr la réutilisabilité du code, mais également une plus grande clarté dans l'organisation pour le développeur.

Les principaux besoins des utilisateurs résultants de l'analyse effectuée (cf Annexe 3) sont :

- L'accès aux modules, à leur contenu, la modification, suppression, création de ceux-ci.
- L'importation et l'exportation de modules à sa base de connaissances

De nouveaux besoins, ou facilités d'utilisation sont apparus par la suite, lors du maquettage de l'interface. Ces nouveaux besoins sont notamment un besoin de "copier" un module ou une classe afin de l'intégrer à une autre base de connaissances. La liste de ces besoins, plus détaillée, apparaît dans le document de conception fourni en annexe.

Deux types d'utilisateurs seront amenés à utiliser, via l'interface SOLFEGE, cette modularité :

- Les premiers, que nous appellerons "fournisseurs" dans la suite de ce document, sont ceux qui créent des modules, pour leur propre usage ou dans le but de les partager.
- Les seconds sont ceux qui utilisent des modules pour les intégrer à leurs travaux. Nous les appellerons "collecteurs", pour les différencier des clients d'une application, qui sont ceux qui utilisent le simulateur une fois terminé. Il est nécessaire de comprendre qu'au fil du temps, les développeurs peuvent passer par les deux statuts.

Le collecteur, tout comme le fournisseur, développe un simulateur et a donc des connaissances poussées sur le fonctionnement de la bibliothèque DIESE. Par opposition, un client n'est qu'un utilisateur du produit final, qui est presque opaque concernant son fonctionnement interne (il n'a qu'une vision boite noire de l'application). La modularisation ne changera donc que peu la vision du client du simulateur qui lui est livré.

Pour répondre aux besoins précédemment édictés, définis plus en détail dans les annexes, nous avons réfléchi à la mise en place d'un module. J'ai tout d'abord effectué un maquettage des différentes nouveautés de l'interface de modélisation. Ce maquettage figure également dans le document de conception.

Le diagramme suivant résume les différents besoins mis en exergue ci-dessus.

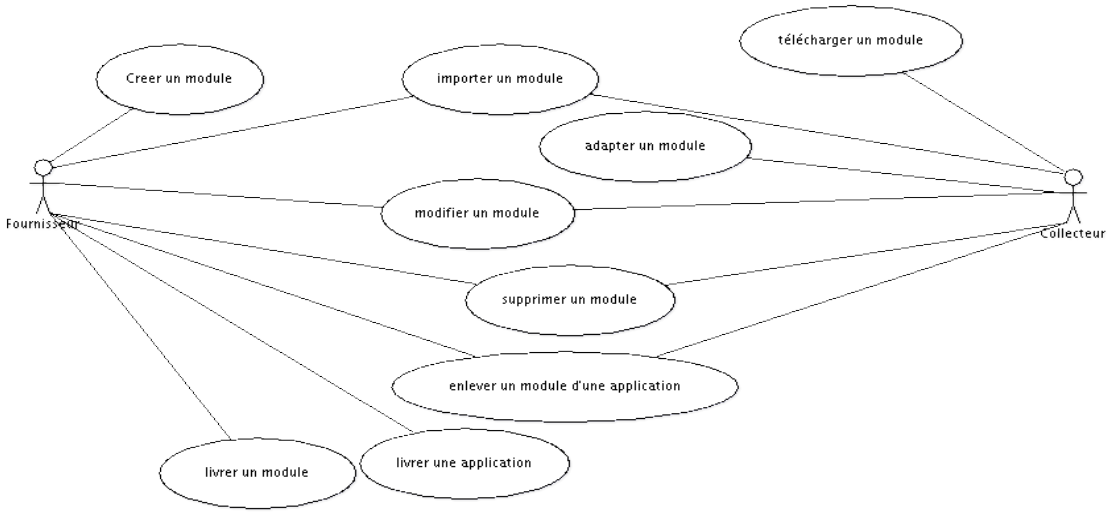


Figure 10 : Diagramme de cas d'utilisation de la modularisation

C) Architecture et représentation d'un module

Une base de connaissances est un ensemble d'informations sur un système. Ces connaissances peuvent être contenues dans un module, un simulateur complet, ou bien les deux. Toute base de connaissance (ou KBS) est donc composée de classes dérivées de l'environnement DIESE, mais également de modules.

On voit apparaître dans cette définition une récursivité : un module peut être composé d'autres modules. Ceci permet une grande faculté de la modélisation en modules : on encapsule des données, on les réutilise dans d'autres modules.

Chaque module est composé de deux parties : un Corps et un Support.
Le Corps du module correspond à ses classes "principales", d'un point de vue sémantique, ainsi que toutes les variables et fonctions globales qui le définissent.
Le support du module A est :

- Une liste de modules (B et C) dont le contenu est donné en complément, car sémantiquement ils ne doivent pas faire partie du module A (exemple : la vitesse d'une voiture).
- Une liste de classes, dont le code est également donné en complément.
- Une liste de références à des objets externes au module, c'est-à-dire l'ensemble des variables non reconnues par le module A, c'est à dire que les items sont déclarés mais non réalisés.

Cette définition du contenu d'un module peut être représentée de façon schématique :

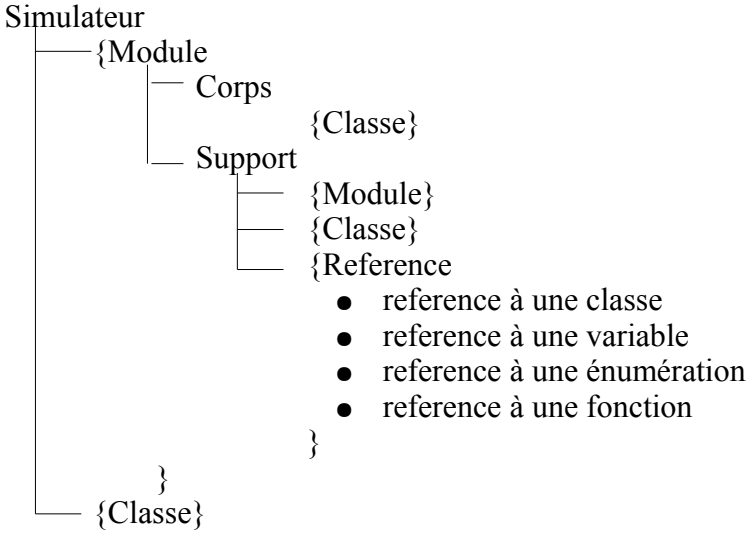


Figure 11: Schéma de la hiérarchie des nouvelles classes de SOLFEGE

Ce schéma reprend les différentes associations contenant/contenu mise en évidence dans le chapitre d'analyse.

Il faut comprendre que du fait qu'une partie du support est incomplète, le modules ne sont pas compilables. Il faut que le module soit intégré à un système pour pouvoir exister. Ce sont finalement ces références et cette notion d'incomplétude qui différencie le module d'un simulateur. On peut même établir la propriété qu'un simulateur est une base de connaissances n'ayant pas de références.

De plus, lorsque l'on souhaite intégrer un module à son environnement, il faut comprendre qu'il peut être excédentaire (c'est-à-dire qu'on n'a besoin que de certaines parties de son contenu) ou au contraire incomplet par rapport à nos attentes. De la même manière, le module peut proposer des informations redondantes par rapport au système auquel il est intégré. Il est donc important de pouvoir proposer des solutions à ces problèmes pour obtenir un modèle cohérent.

J'ai ensuite analysé le système de fichiers actuel, pour voir de quelle manière nous allions être amenés à la remanier. En effet, il est important que les répertoires où sont stockés les bases de connaissance dans la mémoire de l'ordinateur soient bien définis, car SOLFEGE demande à ce que l'arborescence soit claire pour enregistrer les items au bon endroit. A plusieurs reprises, la nouvelle architecture a été repensée, et même après le début de la conception, pour des raisons techniques, de petits détails ont encore changé.

La façon dont les échanges ont lieu entre fournisseur et collecteur ne sera pas étudiée durant le stage : il est considéré que les moyens actuels (internet, bouche à bouche) suffisent pour promouvoir et échanger un module avec un collègue. Cependant, et ce pourra être l'effet d'un autre stage, on pourra par la suite envisager un espace de partage entre chercheurs de ces modules, une communauté, pour garantir un plus grand partage de connaissances.

Le développement de la modularisation a amené à la mise en place de nouvelles classes dans SOLFEGE. Les cinquante classes actuelles ont donc été augmentées d'une trentaine de nouvelles classes. Les nouvelles classes sont tout d'abord la représentation sémantique des nouvelles notions (module, support, référence), ainsi que d'une classe par nouvelle fenêtre créée (fenêtre de modification, de choix, de visualisation ...). En plus de ces créations de classes, j'ai dû modifier les classes existantes pour les adapter à la modularisation. En effet, SOLFEGE a été initialement prévu pour ne gérer qu'un seul type de base de connaissances, et en un unique exemplaire. C'est donc sur ces deux fronts que j'ai été amenée à travailler, c'est-à-dire l'introduction d'un nouveau type de base de connaissance, et la prise en compte de la multiplicité de ces bases.

Après analyse, j'ai décidé de créer une classe mère à toutes les bases de connaissances, étant donné qu'un simulateur et un module ont de nombreuses choses en commun. Cet héritage m'a permis par la suite de surcharger certaines fonctions dans ces deux classes, de sorte que pour la classe principale, le fait qu'il gère un simulateur ou un module lui soit très opaque.

C) Conception de la nouvelle interface

Après avoir défini dans un premier temps les principaux besoins, j'ai maquetté l'adaptation de l'interface existante pour prendre en compte les nouvelles fonctions. Ces maquettes ont été soumises à l'appréciation de mon tuteur. L'ergonomie proposée dans un premier temps a été revue pour être moins lourde et plus proche de l'ergonomie actuelle. Cependant, ce même maquettage a été remis en cause par la suite, lors du développement. En effet, lors de la mise en place, de nombreux problèmes pratiques et techniques nous ont amenés à corriger certains détails.

Il a été décidé de garder au maximum le principe de l'interface existante, SOLFEGE . Pour cela on a utilisé la même charte de visualisation pour les classes d'un module que pour les classes d'un simulateur (entité, descripteur,...), et la même charte graphique pour un module que pour un simulateur.
Ces interfaces doivent être utilisables à la fois pour le fournisseur et pour le collecteur. il faut donc proposer l'ensemble des fonctionnalités sus-nommées à tout utilisateur.

Pour exemple, voici le maquettage proposé initialement, à comparer avec la figure 5 de la fenêtre principale actuelle.

Fig 12 :Interface du menu principal lorsque le simulateur Carrefour est ouvert, menu Modules propres

Toute cette maquette a bien sûr été bien plus détaillée dans le livret de conception, de sorte que tout le développement se base sur ce document. De cette manière, le document de conception agit comme un cahier des charges, comme un contrat précisant le travail à effectuer.

Cependant, des modifications, infimes ont été effectuées pour coller aux réalités de l'application et des techniques utilisées. C'est ainsi que nous avons finalement ré-envisagé la fenêtre principale avec un menu à onglets. Mais la plupart des directives du document ont été respectées.

La modularisation et son intégration dans l'interface graphique a par la suite amené de nouveaux besoins. Celui, par exemple, de transférer une classe ou un module d'une base de connaissances à une autre. Pour remédier à cela, on a imaginé une sorte d'équivalent à un "copier-coller", qui transférerait ou copierait la classe à l'endroit voulu.

Enfin, la partie la plus importante et longue du stage était planifiée pour être la mise en place du système de références. Des fonctionnalités pour identifier les références devaient être mises en place, afin de pouvoir intégrer un module à son environnement. Cela comprend notamment le changement de nom d'une référence pour coller à un nom existant dans l'environnement. Une interface doit permettre de faire ce travail, notamment en permettant de chercher les objets ayant la même sémantique que ce que déclare la références (même type, par exemple...).

Cette interface a pour but d'être utilisée à la fois par le développeur du module, pour définir l'identité de ces références, mais également par le collecteur, afin d'adapter le module acquis à un nouvel environnement.

D) Développement de l'application

Pour développer l'application en concordance avec les besoins et les solutions proposées, j'ai décidé dans un premier temps d'œuvrer par étape, créant d'abord les classes d'ossature des modules.

Cependant, il est très vite apparu le fait que, contrairement au planning que j'avais prévu, ce développement ne pouvait se faire de façon linéaire. Sur un grand projet comme celui-ci, la quantité d'informations est gigantesque, et la moindre erreur peut se répercuter dans de nombreuses classes existantes.

Après avoir développé les classes Module, Corps et Support, je me suis rendue compte que le plus grand changement à faire dans un premier temps se situait dans la classe principale. Il a en effet fallu prendre en compte que celle-ci ne permettait de gérer qu'un seul simulateur à la fois. J'ai donc fait en sorte que cette classe puisse non seulement gérer un simulateur ou un module, ce qui implique un remaniement de chaque fonction de cette classe, mais également que l'on puisse gérer plusieurs bases de connaissances à la fois. Pour ce faire, il a finalement fallu revoir encore une fois la conception actuelle, en séparant la classe principale -qui avait pour but non seulement d'être la classe gérant les bases de connaissances et les appels aux autres classes, mais également l'interface graphique de la fenêtre principale. Nous avons donc séparé ces deux rôles en deux classes pour qu'il soit possible de créer autant de fenêtres principales que de bases de connaissances ouvertes. A ces deux nouvelles classes il a fallu rajouter des méthodes, comme des écouteurs d'évènements sur une fenêtre.

Mais le remaniement conceptuel de SOLFEGE s'est également étendu à la quasi-totalité de ses classes. En effet, comme évoqué précédemment, ces classes ne s'appliquant qu'à une unique base de connaissances, une entière reconception, à la fois théorique mais également du point de vue du développement, a été nécessaire.

Pour exemple, il existe des classes équivalentes aux classes de la bibliothèque DIESE. Prenons la classe Event. Étant donné que tous les évènements instanciés avaient pour but de s'appliquer au simulateur en cours, c'est la classe en elle-même qui détenait la liste de ses instances. Avec la modularisation, il devient nécessaire que cette liste d'instance dépende également de la base de connaissance utilisée. Cette liste d'instance est notamment utilisée dans la classe principale de l'interface, nommée Solfege.

AVANT	**APRES**

class Event
{
static Vector vEv ; // liste des instances
....
}
 	
class Solfege
{
...
// Reference à la liste des instances
Event.vEv <-- ref à une variable de classe
}
	class KBS (base de connaissances)
	{
	Vector vEv ; // liste des instances évent
	...
	}
	
	class Solfege
	{
	...
	Kbs_en_cour.vEv <--variable d'instance
	}

Figure 13: Exemple de modification des classes de SOLFEGE

Cette simple modification, appliquée aux 40 classes raisonnant ainsi, peut amener de nombreuses autres modifications, et une énorme perte de temps par rapport au planning.

Notamment, parmi les changements induits par cette différence de perception d'une base de connaissance, c'est plus de la moitié des méthodes de chaque classe, qui étaient statiques puisqu'elles ne s'appliquaient toujours qu'à la même base de connaissances, qui sont devenues des méthodes simples et non plus des méthodes de classe.

Pour ces diverses raisons, j'ai été amenée à développer en parallèle, et non pas en continu, les différentes parties de la conception à coder.

Pour déboguer mes erreurs, j'ai rassemblé les erreurs par type de classes, beaucoup étant identiques. J'avais parfois plus de mille erreurs, mais le compilateur java ne m'affichait que les 100 premières, ce qui m'empêchait d'anticiper la totalité des futures erreurs.

Enfin, je me suis attelée à déboguer les bugs non visibles à la compilation par des tests systématiques. Cela m'a permis de faire les tests unitaires au cours du temps, cependant, notamment pendant le changement de conception, ces tests ont dû être refaits puisque les fonctions avaient quelque peu changé.

Je dois admettre que la chose qui m'a fait défaut est ce manque d'anticipation de la nouvelle conception, mais je pense que je n'aurai pas été à même de penser à ce changement dans les premières semaines de développement. Je me suis en effet familiarisée à ce moment-là avec le code, que je ne connaissais que jusque-là dans son ensemble. Ce n'est qu'en commençant à développer que j'ai pu comprendre la teneur de ce que j'avais observé lors de mon analyse. Il est pour moi impossible de comprendre toute la teneur de ce logiciel dense en se contentant de l'observer, donc les premiers pas dans le développement, bien qu'au final peu productifs, ont été pour moi un effort nécessaire pour que par la suite ce développement avance plus rapidement.

E) Livraison de la nouvelle version de SOLFEGE

A chaque livraison de l'outil SOLFEGE (environ tous les six mois) sont fournis de la documentation spécifique aux modélisateurs. Dans cette documentation sont disponibles un manuel d'utilisateur, mais également des scripts d'installation de tout l'environnement DIESE pour que l'interface mette en place son arborescence de manière opaque pour l'utilisateur. Ce travail de documentation et de rédaction des scripts Shell d'installation, concernant la modularisation m'incombe.

SOLFEGE n'est pas livré seul. Les autres outils dont j'ai rapidement parlé dans la présentation de DIESE sont développés en continu également par l'équipe de travail de DIESE. A chaque livraison, des modifications sont effectuées sur plusieurs des quatre outils. Cet ensemble qui est livré doit donc avoir une cohérence. Je dois de plus, fournir un document à l'attention des développeurs de l'application, qui fasse un récapitulatif et une explication du travail effectué, des besoins encore insatisfaits...

F) Bilan du travail effectué

Après la phase de conception de la future interface, dont les principaux tenants et aboutissants sont détaillés ci-dessus, je me suis attelée à la tâche du développement de cette interface. Cela a tout d'abord commencé par une analyse plus profonde du code des classes déjà existante, pour comprendre l'architecture mise en place, mais également pour me familiariser avec la création d'une interface graphique. C'est durant ce long travail que j'ai également dû réfléchir à la possible adaptation de ces classes à la modularisation.

A la suite de cette réflexion et après certains essais infructueux, il m'est apparu que la tâche serait plus ardue que prévu, le code actuel n'ayant pas été prévu pour être adapté à un nouveau type de bases de connaissances. C'est ainsi une grande partie des classes Java existantes qui ont dû être remaniées, modifiées et adaptées à cette nouveauté. Cela m'a permis d'avoir un aperçu des grands et moyens systèmes d'informations dans une entreprise, et de comprendre pourquoi les grands projets pouvaient s'avérer si longs et complexes.

Le travail effectivement fini est :

- L'analyse des besoins.
- La conception du modèle de module.
- Le maquettage de l'interface.
- Le développement du squelette des nouvelles classes.
- La création des interfaces pour ces nouveautés (références, menus).
- La mise à jour de l'interface principale (menu à onglet).
- La création de nouveaux noms dans le dictionnaire (pour garantir l'universalité de DIESE, ce produit est actuellement disponible en deux langues : l'anglais et le français).
- La définition des nouveautés dans le "manuel" des développeurs.

Les travaux non terminés lors de la rédaction de ce mémoire sont :

- Le deboggage de certains points graphiques, dont la priorité était moindre
- La réflexion et le développement sur l'adaptation des références
- Les tests d'intégration et de recette.

IV. Réflexion sur le rôle de chef de projet

J'ai, durant ces cinq mois, pu observer le fonctionnement de plusieurs équipes de projet, puisque j'ai eu la chance de travailler dans un très grand organisme. Au sein même de mon département sont développés de nombreux projets différents, et j'ai ainsi pu observer et améliorer la vision que j'avais d'un chef de projet. Bien que n'ayant pas distingué au premier abord quels sont les « chefs », puisque les employés sont presque tous sur un même pied d'égalité, j'ai assez vite pu distinguer les leaders, les preneurs de décision.

Le métier de chef de projet n'est pas accessible à tous. Il faut du charisme, du caractère, de la patience, savoir écouter mais aussi savoir faire avancer les choses.

Mon analyse se basera sur l'observation des chefs de projet, d'équipe et de département dans l'environnement de l'Inra. Je mettrai mes constatations en parallèle avec mes a priori, que j'ai acquis à la fois pendant ma formation, et mon précédent stage.

Cette nouvelle vision de chef de projet sera avant tout le résultat de mon vécu, et de la relation tuteur de stage-stagiaire que j'ai vécue, mais également sur l'écoute et la réflexion que j'ai menée auprès d'autres personnes de l'institut, qu'ils soient chefs de projet, ou bien des chercheurs connaissant cette fonction.

Je commencerai par établir la liste des principales qualités qui selon moi sont nécessaires à un chef de projet, puis je montrerai comment elles sont mises en application à l'Inra, afin de souligner les différences avec mes connaissances théoriques en la matière. Je détaillerai ce qui selon moi fait la différence entre un chef de projet et un employé, j'illustrerai pour cela mes propos de cas réels que j'ai pu observer.

A) Les compétences du chef de projet

Traditionnellement, la notion de chef de projet est apparue dans les secteurs travaillant "au projet", c'est-à-dire le bâtiment, l'ingénierie, ou encore l'industrie aéronautique. Depuis, cette notion de chef de projet s'est étendue à de nombreux secteurs, c'est-à-dire que de nombreuses entreprises où les employés travaillaient de manière répétitive commencent à aborder ce travail à la manière d'un projet.

Le chef de projet est à la tête de la maîtrise d'oeuvre. Il est responsable de la solution apportée à la maîtrise d'ouvrage (ceux qui ont commandé le projet). Dans la recherche, cette notion de maîtrise d'ouvrage et de maîtrise d'oeuvre reste très vague. En effet, une recherche n'est pas faite "sur demande". C'est chaque chercheur, qui, au vu de ses précédentes recherches, de ses intérêts, propose de nouveaux sujets de recherche, qui seront par la suite acceptés ou non par sa hiérarchie. Dans mon unité, qui produit des logiciels, les commanditaires de nos projets peuvent être des collaborateurs, ou des collègues. Dans ce contexte-là il est vrai que le lien entre maîtrise d'oeuvre et maîtrise d'ouvrage peut être très infime.

Quand on pense aux compétences d'un chef de projet, on pense surtout à celle de pilotage d'un projet, c'est-à-dire non seulement l'allocation des ressources matérielles et temporelles, mais également de veiller à ce que celles-ci soient respectées. L'anticipation est donc un maître mot. Cette qualité est notamment mise en application lors de la planification du projet. La seconde compétence qui apparaît chez le chef de projet est la communication, qui reste primordiale lors de la tenue de réunions.

Le chef de projet est ensuite censé maîtriser les techniques utilisées dans son projet, c'est-à dire être capable d'effectuer et de comprendre le travail de son équipe. Cette qualité est également de mise à l'Inra, notamment auprès du chef d'unité qui a le devoir de connaître, du moins en globalité, les sujets de recherche de tous les chercheurs de l'unité. Enfin, pour qu'un projet se passe bien, on demande souvent à ce qu'un chef de projet le soit du début à la fin du projet. Cette contrainte est malheureusement souvent difficile à mettre en place, à cause des départs des employés, qu'il s'agisse de congés ou de départs définitifs. C'est cependant plus simple à l'Inra, le travail de chercheur étant un travail stable, et les chercheurs étant eux-mêmes passionnés par leurs recherches, ils souhaitent les finir avant de commencer un nouveau projet.

Pour terminer, je tiens à souligner le fait qu'un chef de projet peut avoir d'autres fonctions au sein de l'entreprise, qu'il travaille sur d'autres projets, ou soit un représentant de ses pairs auprès de sa hiérarchie. Il a également son propre travail à assurer, sa propre vie à gérer. Il est donc essentiel de savoir faire la part entre les diverses vies de ce chef, de ne pas les mélanger, pour être ainsi reconnu en tant que "bon" chef par son équipe.

B) La planification

L'étape de planification est pour beaucoup primordiale au bon avancement d'un projet. Il permet, tout comme le cahier des charges, de poser des bases stables sur lesquelles l'équipe devra travailler. Souvent, c'est le chef de projet, qui a une formation ou une expérience de gestionnaire, qui est apte à calculer le temps nécessaire à l'accomplissement de chacune des tâches qu'il a précédemment mises en exergue. On considère souvent que ce même chef de projet doit être capable d'effectuer le travail de son équipe, pour pouvoir être à même de bien déléguer et superviser ces tâches.

Dans la réalité, des deux expériences de stage que j'ai vécues, dans le privé comme dans le public, ce n'est jamais tout à fait le cas. Certes, les grandes lignes du travail à accomplir sont édictées par le chef d'équipe, mais ce sont les membres de l'équipe qui vont accomplir cette tâche, qui vont définir les détails de ce travail. Ce sont en effet les plus à même de connaître leurs capacités de travail, car il s'agit en général d'un travail qui concerne leur spécialité, à laquelle ils sont attachés et pour laquelle ils sont des experts.

A l'Inra, ce phénomène est encore décuplé par le fait que certains employés travaillent sur plusieurs projets à la fois. Ils doivent dans ce cas répartir leur force de travail sur ces différents projets pour rester dans les temps impartis et pouvoir fournir les documents (quand il y en a à fournir) demandés aux jalons fixés.

Il existe donc en effet des notions de "date finale" d'un projet. Pour exemple, les employés de l'Inra doivent fournir régulièrement des "papiers", qui sont la synthèse d'un travail effectué, d'une recherche, d'une étude, souvent en partenariat avec d'autres membres de la recherche mondiale. Bien qu'il arrive que ces dates de livraison puissent être flexibles, cette notion de temps ne peut être oubliée. Notamment, certaines activités des chercheurs passent parfois en priorité car leur "date finale" est fixe. C'est le cas par exemple des cours et conférences que de nombreux chercheurs donnent à travers le monde. Ils doivent donc préparer ces tâches avant toute autre.

Pour conclure, bien que la planification ne soit pas absente dans les esprits de tous, il est rare que celle-ci soit détaillée et préparée dans un diagramme de Gantt. Cela n'empêche pas chacun d'être responsable de son temps et de le gérer au mieux.

Il faut entendre le terme planification pour son équipe comme pour soi-même : de nombreuses obligations incombant au chef de projet, il est de son devoir de faire parfois certains choix pour assister aux évènements auxquels il est d'un réel recours, et d'en oublier d'autres où il ne serait finalement que spectateur impassible.

C) La conduite de réunion

Une des activités qui incombe généralement au chef de projet est la préparation et la conduite de réunions. Celles-ci sont nécessaires pour se tenir au courant de l'avancement des travaux. A l'Inra, au sein même d'une équipe, les informations circulent vite sans qu'une réunion formelle soit nécessaire. Cependant, il est important de temps en temps que tous les membres de l'équipe se réunissent pour que tout le monde ait le même niveau d'information. En effet, des informations divulguées oralement ne peuvent en aucun cas être considérées comme contractuelles et acquises par tous. Absences, incompréhensions peuvent se produire. C'est donc le rôle de la réunion de projet que de clarifier la situation et l'avancement du groupe. Cette réunion permet au chef de projet de faire passer clairement ses directives. Il manifeste également son approbation quand aux déroulements des projets dans son équipe.

A l'Inra, en plus de ces réunions de projet, auxquelles je n'ai pas assisté étant externe aux projets en cours, a lieu régulièrement (une fois par semaine), une réunion d'unité ou tous les membres se réunissent. Durant cette réunion, les sujets abordés vont du travail effectué par une équipe (nouveaux projets, acceptation d'un article ...), à des sujets plus globaux à l'unité (nouveau doctorant, séminaire, récapitulatif de réunions Inra...).

Cette réunion est animée par le chef d'unité. Son rôle est d'annoncer les grands points de discussion de la réunion et de veiller au partage de la parole. Il est également le modérateur en cas de conflit minime. Mais dans cette réunion, on ne sent pas de hiérarchie apparente. Chacun peut prendre la parole, au même titre que le chef d'unité, ce qui rend la réunion conviviale et que chacun s'intéresse aux sujets abordés. Parfois, lorsqu'une polémique éclate, c'est le rôle du chef d'unité de calmer la discussion et d'avancer sur les points de la réunion. En effet, la réunion doit être courte. Parfois, quand le programme est chargé, la discussion continue à la cantine, dans les bureaux.

Enfin, pour clôturer une réunion, il est nécessaire d'établir un compte rendu, pour que les absents, comme les présents puissent en permanence se tenir au courant ou se rappeler d'évènements importants à survenir. La rédaction de ce compte rendu est effectuée soit par le chef d'unité lui-même, soit par un autre employé qui s'est désigné pour le faire. Aucune notion d'obligation n'est donc en jeu, le rédacteur du compte rendu étant volontaire. Étant donné que les chercheurs se sentent concernés par leur avenir et leur travail, il leur paraît donc tout naturel de s'informer et de tenir leurs collègues informés. Le compte rendu est établi en temps réel, sur un ordinateur portable la plupart du temps, et est ensuite publié sur le blog de l'unité. De cette manière, quiconque veut se tenir au courant des activités de l'unité peut, en passant par l'intranet du département BIAT, avoir connaissance des dernières nouvelles.

Le chef de projet ne doit pas s'arrêter à diriger une réunion : il doit d'abord la préparer avec ingéniosité pour que celle-ci se passe du mieux possible. S'assurer de la disponibilité des participants est primordial pour que tous soient au courant. C'est pourquoi il vaut parfois mieux faire des réunions à personnel réduit, pour être sûr d'avoir toutes les personnes concernées par le sujet de la réunion, et de n'avoir que ces personnes-là. Il serait en effet vain de convier des employés n'étant pas en contact direct avec l'ordre du jour. En ce cas, un bilan de la réunion peut suffire pour tenir au courant les membres moins impliqués. Cette préparation est toujours de mise à l'Inra, mais c'est, suivant les cas, le rôle du chef d'unité, du chef d'équipe, du chef de projet ou encore d'un des responsables de la communication d'un projet qui se charge de cela.

L'observation de ces diverses réunions m'a permis de comparer ma vision théorique des compétences d'un chef de projet à la réalité. J'ai également, en petit comité avec mon encadrant, participé à des petites réunions pour clarifier l'avancement de mon travail. J'ai donc du mettre quelques unes de ces compétences de communication en oeuvre.

Il apparaît comme primordial que pour pouvoir conduire une réunion et tenir tout le monde au courant, le chef de projet doit savoir communiquer, faire passer un message. Il doit pour cela être charismatique, être en somme un leader de nature. Le chef d'unité étant désigné, on peut en effet penser que ce sont ses qualités de communication, notamment, qui l'ont désigné à ce poste.

Lors d'une réunion, il peut également arriver qu'un conflit éclate. C'est alors le rôle du chef de projet d'éviter que le conflit s'envenime. Je n'ai pas été amenée à voir la résolution de grands conflits de l'entreprise, car ils étaient réglés dans l'oeuf par la volonté de tous de rester dans une atmosphère sereine. De ce fait, les employés eux-mêmes tentent de s'auto-réguler et de trouver des consensus, ou à défaut des compromis, pour continuer d'avancer ensemble. Cette bonne ambiance est selon moi primordiale pour que chaque employé se sente bien dans son travail. Lorsque ces derniers entrent en conflit, il est donc nécessaire que les problèmes soient étalés et réglés. C'est, selon moi, au chef de projet de faire ce travail. Et c'est aussi son devoir d'éviter de rentrer en conflit avec les membres de son équipe. Il ne serait sinon plus respecté et cela ralentirait grandement le travail. Pour éviter les conflits et les régler, un chef de projet doit donc être à la fois ouvert, compréhensif, mais aussi diplomate.

Toutefois, ces conflits sont peu fréquents à l'Inra, du fait non seulement de l'ambiance chaleureuse est moins stressante qui y règne, mais également du fait qu'il est facile de s'isoler des autres : les bureaux sont simples ou doubles, et les portes qui sont la plupart du temps ouvertes vers le monde extérieur peuvent facilement se fermer pour être au calme.

D) La délégation

Tous les chefs de projets ont un grand point fort : alors qu'ils ont souvent de nombreuses choses à gérer en même temps (un stagiaire, un doctorant, deux projets, des réunions d'unité....), il réussissent haut la main à être bons dans chacune de leurs attributions. Pour arriver à cette organisation, cette efficacité, je pense qu'il est nécessaire d'avoir beaucoup d'expérience.
Il apparaît clair qu'un chef de projet junior, qui n'aura d'abord pas la même densité d'activités à gérer, pourra être dans un premier temps dépassé par les évènements.

C'est à ce point qu'apparaît une notion très importante dans la vie du chef de projet : il est normal qu'il ne puisse pas effectuer seul toutes les tâches qu'il supervise. Il doit alors être capable de déléguer le travail de manière équitable et juste entre les salariés de son équipe, notamment. Il sait quand ceux-ci ne suffiront pas à effectuer une tâche précise, aussi peut-il prévoir une personne externe travaillant sur cette tâche.
Pour que la délégation soit efficace, il faut qu'elle soit intelligemment répartie : avec la pratique, on connaît mieux son équipe, les points forts et les points faibles de chacun. Il faut alors attribuer à chacun la tâche qui lui incombe le mieux : celle qu'il sait le mieux faire, mais aussi celle qui saura le passionner assez pour qu'il se sente motivé par le travail. Pour toute entreprise, il subsiste en effet toujours cette notion de rentabilité : il faut travailler bien, et vite. Cette capacité d'organisation de l'équipe et de délégation se retrouve également dans la planification du travail de son équipe.

A l'Inra, les membres d'un projet sont regroupés par spécialité, et chaque chercheur effectue, seul ou en groupe avec d'autres personnes, sa recherche. Dans ce cas, le chef de projet ne doit pas planifier le travail du groupe, mais doit cependant être au courant de ce qui évolue.
La délégation a plutôt lieu lorsqu'il s'agit de petites tâches administratives liées à l'équipe. Il s'agit de répartir celles-ci entre les chercheurs en fonction de ce qu'ils savent faire et du temps qui leur est disponible. Certains assisteront à des réunions à l'étranger, d'autres établiront un compte rendu pour la hiérarchie...

E) Bilan et relation chef-stagiaire

Je conclurai ce compte rendu de mes observations par l'analyse des rapports entre maître de stage et stagiaire, qu'il s'agisse de mon propre vécu ou des observations des autres stagiaires.
Tout d'abord, il apparaît clair que c'est une relation privilégiée qui est mise en place entre un maître de stage et son stagiaire. De cette collaboration naît tout ou partie d'un projet.

Dans une optique d'avancement du projet, chacun des deux acteurs apporte son savoir. Le maître de stage apporte son expérience et sa connaissance, profonde ou non, des problématiques du projet ; alors que le stagiaire amène dans son travail les technologies et raisonnements qu'il a récemment acquis.
C'est le rôle du maître de stage, qui s'avère souvent être également le chef du projet, de motiver son stagiaire, tout comme un chef de projet doit amener son équipe à travailler avec enthousiasme. Il peut également réorienter son travail s'il arrive que celui-ci dérive de son but premier.
Souvent, le travail produit par le stagiaire a un réel but pour l'entreprise qui l'emploie, qu'il s'agisse d'un simple test avant de commencer un grand projet, ou un travail ayant pour but d'être utilisé rapidement.

Pour finir, je tiens à dire qu'il n'existe pas de profil "type" de chef de projet. Certains ont de grandes capacités, d'autres moins, et ce sont ces capacités qui leurs permettent de se faire reconnaître de leur équipe et de leur hiérarchie. Cependant, on peut voir que certaines qualités sont plus qu'appréciées chez un chef de projet: sa disponibilité, sa façon de communiquer. Lorsqu'un leader soude une équipe, celle-ci est alors beaucoup plus performante. Le rôle du chef de projet est donc central : il est celui qui est responsable des tensions, du travail. Cela ne l'empêche pas d'entretenir des relations amicales avec son équipe. Il sera dans ce cas simplement perçu différemment. Mais il lui appartient de trouver un rôle de chef entre "laisser-faire" et autocratique. Et parfois, il arrive même que le chef ne soit pas le leader. Il peut arriver que celui qui est désigné comme tel soit supplanté par un collègue plus charismatique dans son rôle de leader.

V. Bilan de stage

Au terme de ce stage, je me dois de faire la synthèse des apports très nombreux que l'opportunité de ce stage m'a offerts. Je commencerai par dresser un récapitulatif de mon travail, de ce qu'il a pu apporter à l'entreprise, des perspectives que mon travail de stage ouvre... Je ferai ensuite un bilan sur les différentes aptitudes que j'ai entrevues, acquises, améliorées, ainsi que les conclusions que j'ai tirées de mes observations durant ces cinq mois.

A) Bilan du travail au sein de l'entreprise

Mon travail s'est effectué en plusieurs phases : une phase d'analyse de l'existant, durant laquelle je me suis familiarisée avec l'environnement de travail, les modèles et outils utilisés, ainsi que le logiciel que j'allais devoir étendre. Durant cette phase, j'ai été soumise à diverses difficultés, dues à une approche que je n'avais jamais envisagée. C'est cette approche de méta-modèle que j'ai dû appréhender. Ce n'est qu'en ayant compris le fonctionnement et l'intérêt de ce paradigme que je pouvais comprendre la teneur du projet.

L'application de mes toutes nouvelles connaissances à des exemples concrets a été très intéressante et m'a permis d'avoir une vision moins théorique. Je pense avoir compris assez rapidement, avec l'aide de Jean-Pierre Rellier, l'intérêt et le fonctionnement, ou tout du moins les bases, de la bibliothèque DIESE, nécessaires à la bonne poursuite du stage.

Mais il ne faut pas oublier que le travail que j'ai effectué avait une réelle raison d'être pour l'entreprise, et que mon travail, même s'il n'est pas clos lors de la rédaction de ce présent bilan, sera tout au moins une base de travail solide pour mes successeurs.

Ce stage m'a également permis d'apporter, en plus de mon travail pour l'entreprise, des prises de conscience : il a permis de revenir sur des points acquis depuis longtemps, afin d'améliorer des fonctionnalités avec des contrôles auxquels il n'avait pas été pensé sans voir un utilisateur utiliser le logiciel. De plus, le logiciel ayant évolué avec le temps, et continuant d'évoluer, il a parfois mis en lumière certaines des modifications à effectuer.

Il m'a aussi été possible de voir en temps réel l'utilisation que peuvent faire les clients de l'application, puisqu'ils n'ont pas une connaissance aussi poussée de l'outil que peut en avoir son concepteur, et peuvent donc ignorer certaines astuces d'utilisation, tout comme moi, en tant que débutante, pouvait les ignorer.

J'ai enfin pu apporter mon point de vue sur mon futur travail, conforter ma propre vision à celle de mon maître de stage. Plus qu'un simple travail, ce stage m'a été très enrichissant de tous points de vue : professionnel, technique et personnel.

B) Bilan technique

J'ai ensuite pu profiter de ce stage pour améliorer, voire même apprendre de nouvelles compétences : Tout d'abord, travailler dans un environnement Linux, duquel je n'étais que très peu familière me permettra plus tard de travailler sur n'importe quel système d'exploitation. Travaillant en ligne de commande, je suis maintenant beaucoup plus familière avec ce genre de questionnement de machine. J'ai également travaillé sur des logiciels qui m'étaient inconnus, comme Emacs, qui est très puissant, notamment pour la programmation en C++. Je suis également beaucoup plus douée avec la manipulation de la suite OpenOffice, qui est utilisée sur toutes les plate-formes.

Pour modéliser mes applications, j'ai utilisé l'outil ArgoUML*, qui est un logiciel de modélisation d'un système. Je n'ai pas découvert toutes les fonctionnalités de cet outil, aussi le trouve-je moins puissant que l'outil Win'Design, utilisé en cours. Mais apprendre à utiliser d'autres outils est nécessaires pour pouvoir s'adapter rapidement à un nouvel environnement et être plus efficace.

La mise en pratique de connaissances théoriques a été également intéressante : j'ai pu visualiser quelles sont les attentes du monde du travail vis-à-vis des informaticiens, mais j'ai également pu étendre mes savoir-faire et les faire "coller" aux besoins du moment. Cela m'a fait réaliser qu'à chaque moment, un informaticien doit apprendre, réapprendre pour pouvoir avancer toujours plus. La veille technologique est nécessaire et efficace pour être plus performant dans son travail, et on ne peut considérer son savoir comme acquis. Le travail d'informaticien est donc un travail en constante évolution.

Parmi les nombreuses connaissances que j'ai appliquées et améliorées, pour lesquelles j'ai souvent été amenée à effectuer des recherches supplémentaires, je peux citer le langage Java, et particulièrement les interfaces graphiques, lesquelles m'étaient très peu familières. Mais également, et bien que je n'ai pas à eu à développer dans ce langage, le C++ pour comprendre le logiciel existant et la bibliothèque du méta-modèle. J'ai également été amenée à me représenter le monde informatique d'une manière différente, à cause de la nouvelle approche de méta-modèle sur laquelle j'ai travaillé. J'ai de cette manière pu concevoir qu'il y avait un monde "au-dessus" des langages comme le Java, le C++.

Enfin, au delà des compétences techniques que j'ai pu développer, j'ai également appris à rédiger de la documentation selon les modèles préétablis dans l'entreprise. J'ai en effet dû établir un document de conception et je dois également rédiger des récapitulatifs des innovations de la nouvelle version de SOLFEGE, à la fois pour les développeurs suivants, mais également, et ce de manière plus compréhensible, pour les utilisateurs.

C) Bilan professionnel

Le bilan professionnel que je peux dresser commence bien avant le début du stage, lors de la recherche de celui-ci. Il est en effet très formateur de solliciter de soi-même les entreprises, de préparer son entretien puis d'y assister. J'ai durant ma recherche de stage effectué deux entretiens, et la préparation, la gestion du stress et l'organisation des réponses aux questions en temps réel a été très formateur.

En ce qui concerne le stage en lui-même, il m'a tout d'abord permis d'avoir une meilleure vision du monde de l'entreprise, à ne pas me focaliser sur le seul exemple de l'entreprise que j'avais observé précédemment. Je me suis ainsi rendue compte qu'il n'y a pas un environnement de travail "type", que chaque expérience est différente de la précédente. Travailler dans une secteur qui m'était totalement inconnu, la recherche, d'autant plus que le domaine de recherche, vaste et intéressant qu'est l'agronomie, m'a permis d'élargir grandement mes perspectives sur le monde du travail, sur la diversité des métiers et des secteurs existants.

J'ai pu voir que le monde de l'entreprise ne peut se résumer au travail individuel de chacun : c'est toute une communauté qui évolue, débat, partage, se réunit autour d'un but commun.
Les employés partagent de nombreuses activités annexes, en dehors du travail, mais au centre de ces nombreux points d'intérêts, tous ces employés restent attachés à leur travail.

Je dois avouer que c'est ceci qui m'a tout d'abord étonné : voir des gens travailler sur un sujet qui les passionne, donner de leur temps encore et encore, pour sauver et faire avancer leurs recherches. Même durant leurs pauses, c'est toujours cette même recherche qui anime les chercheurs, cette même passion de trouver des solutions à des problèmes qui leurs sont chers.

Dans cette ambiance chaleureuse, j'ai également pu voir évoluer des personnes s'apparentant à des chefs de projet, et j'ai vu comment ils mettaient leurs compétences en application pour empêcher des conflits, animer une réunion... Car il est plus qu'important, en plus des connaissances théoriques que l'on peut nous apprendre à la Miage, de voir comment dans la pratique ces savoirs sont mis en place. C'est également sur une clarification des rapports hiérarchiques dans une entreprise que s'achève ce stage. Je sais à présent que la hiérarchie est présente pour diriger, aiguiller, et qu'elle ne doit pas être considérée comme une ennemie, mais comme une alliée permettant d'évoluer dans le monde de l'entreprise.

J'ai durant ce stage travaillé sur un projet de grande ampleur, si l'on considère le temps consacré à celui-ci, les différents développeurs au cours du temps, ainsi que la quantité de code effectuée. Cela me permet donc d'imaginer plus aisément le genre de tâches qui peuvent être effectuée au sein d'un grand projet. En effet, au début du stage, je ne m'imaginais pas à quel point un projet comme celui-ci pouvait prendre de temps, et l'organisation du travail qu'il exigeait pour pouvoir avancer au mieux.

Dans un tel projet, pour que les différents développeurs puissent se comprendre, il est nécessaire d'adopter des normes d'écriture, de commentaires. J'ai donc tenté, du mieux que j'ai pu, et avec les moyens qui m'étaient fournis, de travailler de la façon que mon encadrant m'avait montré. J'ai certes dans un premier temps dû m'adapter à une écriture qui n'était pas tout à fait la mienne, mais il est important que je garde les normes existantes sans chercher à imposer ma façon de faire. Cette période de transition passée, j'ai pu me rendre compte que je pouvais aisément me faire à un nouvel environnement à et à des nouvelles contraintes, ce qui me sera une grande qualité par la suite.

Enfin, et l'on n'y pense pas assez souvent, travailler durant ces cinq mois à l'Inra m'a permis de rencontrer moults personnes de différents horizons, puisque beaucoup ne sont que de passage à l'Inra, et d'agrandir le cercle de mes contacts professionnels, qui pourront m'être d'une grande aide par le futur, que ce soit dans ma recherche d'un futur projet, d'un stage ou encore d'un emploi. C'est dans la bonne ambiance que se développent les meilleurs contacts, et que, selon moi, le meilleur travail est effectué.

D) Bilan personnel

Pour finir, ce stage m'a également été bénéfique au niveau humain : en plus d'avoir eu la chance de connaître des personnalités incroyables et ouvertes, j'ai découvert deux mondes qui m'étaient totalement étrangers : l'agronomie et la recherche. En plus de côtoyer chaque jour ces personnes intéressantes et passionnées par leurs travaux, j'ai pu participer à des discussions sur de nombreux sujets de société, comme l'écologie, les organismes génétiquement modifiés, et bien d'autres encore, touchant de près ou de loin la recherche agronomique.

Les deux nouveaux mondes que j'ai découverts, bien qu'intéressants, ne seront pas selon moi des voies vers lesquelles je m'orienterai plus tard, car ils ne me semblent pas correspondre totalement à ma formation ni à mes ambitions. Mais cette expérience me permet justement de restreindre mes choix de futurs stages et emplois, afin de savoir dans quelle filière je souhaite spécifiquement travailler.

Ensuite,ce stage m'a permis de travailler une de mes grandes faiblesses, la communication. J'ai en effet dû exprimer les problèmes que j'ai rencontrés à mon maître de stage, afin que celui-ci m'aide à les résoudre. Cela n'a pas toujours été facile car j'ai tendance à remplacer un mot par un autre, et à penser plus vite que je ne parle, ce qui devient un langage très difficilement compréhensible pour autrui. J'ai donc tenté de travailler ceci, en expliquant posément mes problèmes, parfois en les écrivant d'abord sur papier. Je me devais en effet de tenir un discours organisé, chose qui n'est pas aisée lorsque cela n'a pas été préparé.

J'ai par la suite été amenée à faire une présentation générale devant tous les membres de mon unité, ce qui implique d'expliquer d'un point de vue simple les principes du projet, qui était peu connu pour nombre de mes collègues.

J'ai également appris à ravaler parfois mon orgueil, à admettre que je ne pouvais pas tout savoir. Les premiers temps, demander de l'aide à mon maître de stage, sur des sujets que j'aurais dû -selon moi- maîtriser, m'apparaissait comme un échec. Ce n'est heureusement pas le cas, et ces informations que m'a fournies Jean-Pierre Rellier m'ont été d'une grande aide dans les blocages techniques que j'ai rencontrés. C'est finalement de cette manière que l'on apprend le plus, et je suis heureuse d'avoir pu acquérir ainsi de nouvelles connaissances.

De même qu'un projet comme celui-ci forme au travail en équipe, puisque je travaillais en referrant régulièrement mon travail à mon encadrant, j'ai également du être plus autonome dans mon travail. Il est certes important de ne pas s'auto-bloquer et de ne pas hésiter à demander de l'aide, mais il paraît également évident qu'un excès de demandes peut vite faire perdre du temps à tous. J'ai donc dû, en cas de problèmes, réfléchir par moi-même, chercher des informations et solutions. Ce n'est qu'en dernier recours que je demandais de l'aide à mon maître de stage.

Personnellement, un stage comme celui-ci peut apporter beaucoup : une remise en cause, un sentiment de réconfort, ou bien encore une passion naissante pour une spécialité inconnue. Pour ma part, je peux avouer que ce stage m'a apporté la confirmation de mon choix de carrière : la partie que j'ai trouvée la plus intéressante dans ce stage a été la conception et la planification du travail à effectuer, plus que le développement de l'outil en lui-même. Je sais que je veux être une actrice de grands projets, je veux pouvoir me dire que mon travail sert à d'autres, car cela apporte à mes yeux une sensation d'être utile.

C'est enfin sur un sentiment de satisfaction que je rédige ce bilan. Bien que tous les objectifs du stage ne soient pas complètement atteints, j'ai appris, j'ai travaillé, j'ai essayé d'arriver à un résultat, aussi incomplet soit-il, est cela est déjà une fin en soi. Je sais que mon travail sera réutilisé et amélioré par la suite, et j'espère pouvoir laisser les renseignements nécessaires à ce que ceci s'accomplisse dans les meilleures conditions.

Bibliographie et Références

site national de l'Inra : http://www.inra.fr/

site de l'Inra de Toulouse : http://www.toulouse.inra.fr/

site du extranet de l'unité BIAT : http://mia.toulouse.inra.fr/

approche de la méthode OMT :

http://www-ic2.univ-lemans.fr/~alissali/Enseignement/Polys/GL/node78.html

Fonctionnement du système de fichiers Xfce : http://fr.wikipedia.org/wiki/Xfce

Utilisation de Lex et Yacc : http://www.linux-france.org/article/devl/lexyacc/minimanlexyacc.html

Précédent projet autour de DIESE, effectué par Emmanuel Pannier (Miage) : MODERATO

record.toulouse.inra.fr/publications/stage_pannier_2006.pdf

Utilisation de doc++ : http://www.zib.de/visual/software/doc++/

Logiciel argoUML : http://argouml.tigris.org/

Introduction à la stratégie événementielle :

Structure of discrete event simulation. John B. Evans. Editions Ellis Horwood / Halsted Press

Méthode de modélisation objet :

Modélisation objet UML. Pierre-Alain MULLER et Nathalie GAERTNER. Editions Eyrolles.

Glossaire

- **Access** : Logiciel de bases de données de la suite Microsoft, il permet notamment, grâce à son propre langage, le Visual Basic, de créer des applications en relation avec des bases de données.

- **Auzeville Tolosane** : Ville du Sud-est de l'agglomération Toulousaine, située entre Castanet Tolosan, Ramonville et Labège. Elle fait partie de de la communauté d'agglomération du Sicoval.

- **BIA Toulouse(unité)** : Unité Biométrie et intelligence artificielle. Sous-ensemble du département MIA, national à l'Inra. Cette unité fait des recherches poussées dans deux voies :
  - l'analyse et la gestion des systèmes de production et des ressources physiques
  - l'analyse de la structure et de la fonction des génomes.

- **Demon :** Processus chargé d'une mission. Ce programme n'est pas sous le contrôle de l'utilisateur, c'est-à-dire qu'il travaille de manière opaque. Dans DIESE, un démon est un moniteur, attaché à un descripteur, qui a pour but de vérifier que le descripteur reste dans les valeurs de son domaine, par exemple.

- **DIESE** (DIscrete Event Simulation Environment) : collection d'outils permettant le développement de simulateurs. Cet environnement est fondé sur deux bibliothèques : la bibliothèque "basique" (BASIC DIESE pour DIscrete Event Simulation Engine) et la bibliothèque CONTROL DIESE, qui est spécialisée dans le pilotage des systèmes. L'environnement est également complété par deux interfaces graphiques, SOLFEGE et MI_DIESE, qui ont respectivement pour but d'aider au développement d'un simulateur et de faciliter le déroulement d'une simulation.

- **Doc++ :** DOC++ est un système de documentation pour C, C++, IDL et Java. Il peut générer au choix un document TeX pour produire une documentation imprimée de qualité, ou un fichier HTML pour un parcours en-ligne aisé de la documentation. La documentation est extraite directement du fichier d'en-tête C/C++/IDL ou du fichier des classes Java.

- **EADS Astrium :** Grande entreprise de l'agglomération toulousaine, elle est une filiale de l'entreprise EADS, spécialisée dans les systèmes spatiaux civils et militaires.

- **ENSAT (Ecole Nationale Supérieure Agronomique de Toulouse)** : forme des ingénieurs agronomes et propose 6 options dans des domaines comme les biotechnologies, l'environnement, la qualité ou la gestion.

- **INRA** : Institut national de la recherche agronomique. Cette entité française est basée sur de nombreuses villes, telle que Toulouse, Rennes, Bordeaux, Montpellier... et a pour but d'étudier les comportements au niveau de l'alimentation, l'environnement, l'agriculture, la valorisation des territoires. Plus d'informations sur le site de L'Inra, donné en bibliographie.

- **Lex/ Yacc :** outils très populaires de génération d'analyseurs lexicaux (Lex) et syntaxiques (Yacc) en langage C. « Yacc » est l'acronyme de Yet Another Compiler Compiler. Lex traite les langages de types 3 (ou réguliers), alors que Yacc fournit le code nécessaire à l'analyse de langages de type 2 (ou non-contextuels).

- **MIA (département) :** l'un de 14 département de recherche de l'Inra. L'objectif du département est de développer des recherches méthodologiques dans différents domaines des mathématiques et de l'informatique pour répondre aux grandes priorités de l'Inra concernant l'agriculture, l'environnement et l'alimentation.

- **MI_DIESE :** interface graphique, alternative du lancement par l'utilisateur lui-même d'une ligne de commande au système d'exploitation avec en paramètres des fichiers préparés avec un simple éditeur de texte. C'est par une option de la compilation du simulateur qu'on décide si celui-ci pourra être utilisé sous l'interface graphique ou non.

- **Meta-modèle :** Représentation d'un point de vue particulier sur des modèles. On admet qu'un méta modèle est le modèle d'un langage de description de modèles.

- **OMT :** Object Modeling Technique : Ancêtre de L'UML, cette méthode de modélisation d'un système orientée Objet a été créée par James Rimbaugh. Ce projet tendait à trouver une technique de modélisation des systèmes orientés objets, mais a finalement été remplacé par l'UML, qui est une formalisme issu de cette technique.

  La méthode est basée sur trois étapes :
  - L'analyse : on décrit le but du système et non pas la façon dont il sera réalisé. Il ne faut prendre aucune décision d'implantation.
  - La conception système : on définit le découpage du système en sous-systèmes, à partir des résultats de l'analyse et dans l'objectif de définir le choix de l'architecture
  - La conception des objets : on raffine les structures de données et les algorithmes en y ajoutant les détails d'implantation et en tenant compte de l'environnement.

- **Recherche scientifique publique finalisée :** Il s'agit de prendre en compte dans les questionnements scientifiques les enjeux socio-économiques. Ces organismes de recherche font progresser les connaissances, débouchent sur des innovations pour la société et permettent d'éclairer les décisions, publiques ou privées

- **SOLFEGE :** interface graphique de modélisation d'un simulateur dans l'environnement DIESE, dans un contexte de résolution de problèmes agronomiques.

- **Ssh (secure shell):** programme informatique et protocole de communication sécurisé. A la connexion, le protocole échange des clés de chiffrement, de sorte que par la suite chaque message qui transite avec l'aide de ce protocole soit chiffré.

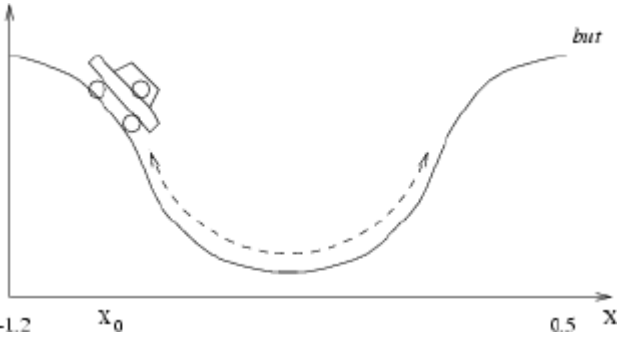
- **Thunar :** projet visant à développer un nouveau gestionnaire de fichiers pour l'environnement de bureau Xfce à partir de la version 4.4. Thunar utilise des technologies assistives, de sorte qu'il soit le plus facile d'emploi possible. Il respecte les standards, tout en ayant de hautes performances et un code facilement maintenable.

- **UML** : Unified Modeling Langage : Langage de modélisation des données et des traitements. Ce formalisme est très abouti et non-propriétaire (il est utilisable par tous, soumis à aucune licence). Il permettant de représenter les systèmes orientés Objet. Ce langage propose notamment plusieurs diagrammes, dont les plus connus sont le diagramme de classe, le diagramme de séquence, le diagramme de cas d'utilisations.

- **Xfce :** environnement de bureau léger, destiné initialement aux systèmes d'exploitation Unix.

Annexes

Annexe 1 : Le toboggan

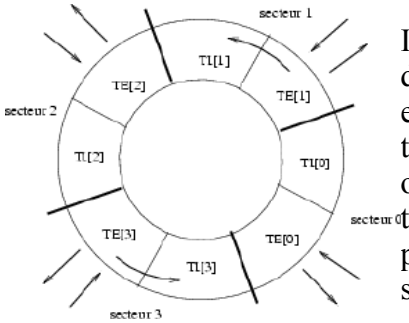


l'objectif est de simuler la trajectoire d'une voiture sur une voie ayant le profil ci-contre, entièrement dans un plan vertical. A tout instant, la position de la voiture est déterminée par :

- une fonction X(t+1) = f(X(t), V(t+1)),
- une fonction V(t+1) = f(V(t), X(t), U(t)), où U(t) est une action de contrôle (1 ou -1 selon que le pilote de la voiture accélère ou freine pendant l'intervalle de temps]t, t+1]). Cette action de contrôle est déterminée par une stratégie U(t) = f(V(t), X(t)). La vitesse est positive quand la voiture va de gauche à droite, et négative en sens contraire,
- la position initiale X(0) dans [-1.2, 0.5] et la vitesse initiale V(0) dans [-0.07, 0.07].

Lorsque la voiture atteint la limite gauche (X(t) = -1.2), elle ne va pas plus loin et sa vitesse est remise à 0. La vitesse est limitée à 0.07, dans les deux sens. Lorsque la voiture sort du secteur par la droite (X(t) > 0.5), la simulation est arrêtée. On mesure le temps que met la voiture pour atteindre ce but.

Annexe 2 : Le carrefour



Il s'agit de simuler le trafic dans un carrefour giratoire à *N* voies d'accès. Le carrefour comporte *N* tronçons d'échange (TE[i]) par où entrent et sortent des voitures, et *N* tronçons internes (TI[i]) par où transitent les voitures. D'un tronçon interne, une voiture passe obligatoirement au tronçon d'échange du secteur suivant. D'un tronçon d'échange, une voiture peut soit sortir du carrefour, soit passer au tronçon interne du même secteur. Une voiture qui désire sortir par sa voie d'entrée doit faire un tour complet. Afin d'assurer le bon fonctionnement du carrefour, au plus *maxEchange* voitures peuvent se trouver dans un même tronçon d'échange, et au plus *maxInterne* voitures peuvent se trouver dans un même tronçon interne. Les dépassements ne sont donc pas autorisés si *maxEchange* = 1 ou *maxInterne* = 1.

Priorité est donnée aux voitures déjà dans le carrefour. Une voiture désirant entrer dans le carrefour par la voie *i* ne peut le faire que si le tronçon d'échange TE[i] et le tronçon interne TI[i-1] sont libres.

Lorsqu'une condition pour entrer dans le carrefour ou pour y avancer n'est pas satisfaite, cette action est différée. La precondition ne sera examinée à nouveau qu'après un certain *délai*.

Une voiture arrive sur le carrefour et le quitte sur des voies d'entrée et de sortie aléatoires et indépendantes. L'écart entre deux arrivées varie uniformément entre 0 et *intensité* secondes. Chaque voiture possède sa propre vitesse, spécifiée par la durée du passage dans un tronçon (d'échange ou interne). Cette durée est choisie aléatoirement (et uniformément) entre 15 et 30 secondes.

Annexe 3 : Extrait de l'analyse de la modularité dans l'environnement DIESE

Analyse de la mise en place de la modularité de l'environnement Diese

Laureline Vidalenc (stagiaire INRA /UBIAT, sous la responsabilité de Jean-Pierre Rellier), Mai 2008

Ce document a pour but d'établir une définition et une règle de mise en place exhaustive de la modularité dans l'environnement de modélisation DIESE.

DIESE est une bibliothèque de modélisation de systèmes d'intérêt agronomique, qui est utilisée à l'INRA par plusieurs équipes, afin de simuler divers environnements et mécanismes, comme la culture de tomates sous serre.

Cette bibliothèque de classes permet de représenter un système selon un paradigme défini.

Chaque élément peut être représenté soit sous la forme d'une entité (par exemple la tomate), soit sous la forme d'un évènement (le lever du jour), soit sous la forme d'un processus (la pousse de la tomate).

A ces trois classes principales sont associées des classes complémentaires. Celles-ci sont détaillées dans la documentation de DIESE.

Afin de faciliter le développement d'un simulateur par ses concepteurs, une interface nommée SOLFEGE a été créée. Elle aide à la conception en facilitant la représentation du simulateur.

Pour cela, elle propose des fenêtres ergonomiques permettant d'éditer, de visualiser, de créer, de supprimer, de mettre en relation des classes, créées sur le modèle des classes de la bibliothèque DIESE.

Enfin, elle produit un simulateur livrable et capable de fonctionner dans les environnements Unix et Microsoft Windows.

Objectifs de la modularisation

Les buts principaux de cette modularisation sont :

- Tout d'abord, un meilleur regroupement, plus sémantique, des connaissances développées dans une application sous DIESE.
- C'est ensuite permettre un gain de temps à un développeur, qui peut réutiliser un module qu'il a lui-même précédemment développé, dans d'autres applications.
- C'est également une notion de partage de connaissance avec d'autres développeurs qui pourraient avoir besoin du module développé par un collègue, dans un domaine qui leur serait peu connu, et de l'adapter à leur propre simulation.

Un module regroupe un ensemble de classes dont le concepteur pense qu'elles forment un tout, un élément important et réutilisable dans un autre environnement. Cet ensemble est appelé le corps du module. Un module ne peut cependant se limiter à cette approche sémantique : pour être réutilisable et répondre à sa sémantique, il se doit de définir également ce qu'il n'est pas, mais dont il a besoin. Cette partie sera appelée le support du module, par opposition au corps du module, précédemment défini.

Utilisation et besoins

Deux types d'utilisateurs seront amenés à utiliser, via l'interface SOLFEGE, cette notion de modularité.

Les premiers, que nous appellerons "fournisseurs" dans la suite de ce document, sont ceux qui **créent** des modules, pour leur propre usage ou dans le but de les partager.

Les seconds sont ceux qui utilisent des modules pour les intégrer à leurs travaux. Nous les appellerons "collecteurs". Un collecteur de module peut être amené à le **modifier** et le reproposer, avec une sémantique légèrement différente, ou bien encore à **créer** ses propres modules pour les **emboîter** avec des modules téléchargés.

Il est nécessaire de comprendre qu'au fil du temps, les développeurs peuvent passer par les deux statuts.

Le collecteur, tout comme le fournisseur, développe un simulateur et a donc des connaissances poussées sur le fonctionnement de la bibliothèque DIESE. Par opposition, nous appellerons "clients", ceux qui utilisent le simulateur une fois terminé, qui est presque opaque concernant son fonctionnement interne (il n'a qu'une vision boite noire de l'application). La modularisation ne changera donc que peu la vision du client du simulateur qui lui est livré (il sera peut-être amené à voir quelques différences lorsqu'il aura le choix entre des modules alternatifs).

- Identification des besoins

Les fonctionnalités qui existent actuellement sont bien sûr toujours présentes dans cette évolution de SOLFEGE, mais devront être adaptées à la modularisation. De nouveaux besoins apparaissent. Ces nouveaux besoins peuvent être en relation avec le rôle de fournisseur, de colleteur, voire des deux.

- Besoins existants à adapter

Le fournisseur doit pouvoir **développer des classes** comme il le faisait précédemment, et pouvoir, s'il le souhaite -et seulement s'il le souhaite- **regrouper ces classes** dans différents paquets. Il est donc à comprendre que la modularisation n'est pas une étape obligatoire, et que c'est à la charge de chaque développeur d'étudier la mise en place de la modularisation dans son simulateur.

Un fournisseur doit être capable **d'enregistrer son simulateur**, de la même façon qu'il l'a toujours fait, afin qu'il puisse le **récupérer** lorsqu'il le souhaite.

Il doit également pouvoir effectuer la **livraison de son simulateur** auprès de son client. Cette livraison peut se faire de deux manières : en donnant le code source au client, de sorte qu'il puisse éventuellement le modifier, ou en ne livrant que le code compilé ainsi que la documentation nécessaire à la compréhension du simulateur.

Cette seconde livraison se fait actuellement à l'aide d'un script nommé makeRelease, qu'il faudra donc l'adapter à la modularisation. Pour la première livraison, où l'on inclut les codes sources, cette manipulation devra pouvoir s'effectuer dans un environnement modularisé : le code source des modules du simulateur devra donc être inclus le cas échéant.

- Nouveaux besoins

Les modules doivent vivre indépendamment de tout simulateur : ils existent en parallèle, et peuvent se raccrocher à un ou plusieurs simulateurs. Ces modules doivent être modifiables, renommables, supprimables.

Le fournisseur doit être en mesure de **donner le contenu** -et bien le contenu, non le code compilé- d'un simple module à un collecteur. Le collecteur, lui, veut pouvoir **récupérer un module** venant d'un fournisseur et l'**intégrer** à son propre système de fichiers.

Lorsqu'il souhaite utiliser un module, qu'il s'agisse du sien ou d'un qu'il a précédemment importé, il faut que le collecteur puisse attacher et choisir le module concerné pour l'intégrer dans le simulateur. Il doit donc être possible **d'attacher et de détacher un module d'un simulateur.**

L'importation d'un module dans un ensemble de classes et d'autres modules implique de nouveaux problèmes :

- incomplétude : par définition un module est incomplet, du fait que ses références ne sont pas satisfaites "en interne". Ce n'est qu'en intégrant le module dans un environnement qu'on peut faire en sorte que les références trouvent leur équivalent. De cette manière, il n'est pas envisageable de pouvoir compiler un module. Le problème est donc la détection automatique des références non satisfaites.
- incohérence : Lorsque l'on souhaite adapter l'environnement au module, plusieurs propositions peuvent exister pour satisfaire une référence. Il est nécessaire que ses propositions correspondent totalement aux attentes de la référence. Il faut donc réfléchir à la manière dont les exigences d'une référence vont être définies, à la façon dont on pourra ou non modifier l'environnement ou le module en conséquence, ce afin que ce travail ne soit pas laborieux pour le modélisateur.
- redondance : une personne intégrant un module à son environnement n'a pas forcément besoin de chacune des fonctionnalités proposées. Il faut alors pourvoir choisir les méthodes que l'on souhaite garder / adapter.

Cette analyse et les propositions qui en découleront devront être retranscrites dans l'interface.

La façon dont les échanges ont lieu entre fournisseur et collecteur ne sera pas étudiée dans ce document : il est considéré que les moyens actuels (internet, bouche à bouche) suffisent pour promouvoir et échanger un module avec un collègue. Cependant, et ce pourra être l'effet d'un autre stage, on pourra par la suite envisager un espace de partage entre chercheurs de ces modules, une communauté, pour garantir un plus grand partage de connaissances.