Département

Projet de plateforme « sol virtuel »

Analyse informatique du cahier des charges

Version 2009

Auteur : Nicolas Moitrier[1]

Co-auteur : Nathalie Moitrier[2]

Relecteurs : groupe projet de la plateforme « sol virtuel »[3], Jean-Christophe Fabre[4]

Dernière modification : 4 novembre 2009

1 Chef de projet informatique de la plateforme « sol virtuel », nicolas.moitrier@avignon.inra.fr
2 Équipe projet informatique de la plateforme « sol virtuel », nathalie.moitrier@avignon.inra.fr
3 Alain Mollier (mollier@bordeaux.inra.fr), François Lafolie (lafolie@avignon.inra.fr), Isabelle Cousin (Isabelle.Cousin@orleans.inra.fr) et Valérie Pot (vpot@grignon.inra.fr)
4 Plateforme « paysage virtuel » / *MHYDAS* – OpenFLUID, fabrejc@supagro.inra.fr

Table des matières

Index des illustrations

Index des tables

1 Préambule

Ce document est une interprétation des besoins du projet -dans son contexte scientifique- par un informaticien. Il contient donc naturellement des vulgarisations, des déformations de la discipline scientifique. Il ne remet pas en cause les besoins scientifiques exprimés dans le cahier de charges[5] fonctionnel. L'interprétation des besoins formulés et les choix proposés dans ce document ont été discutés et avalisés par les membres du groupe projet « sol virtuel ».

Le principal public visé est les futurs utilisateurs de la plateforme, qu'ils soit utilisateurs de modèles ou contributeurs, c'est à dire principalement la communauté impliquée scientifique du département EA.

Ce document ne remplace pas une analyse « métier » -c'est à dire scientifique- des besoins. En effet, la phase d'expression du besoin décrit la définition de ce que l'on attend, les fonctions attendues. L'analyse scientifique doit définir le périmètre du projet -ce sur quoi on va l'évaluer- et surtout les arbitrages -ce qui est important et ce qui l'est moins- en tenant compte des moyens disponibles. Ceux-ci permettront l'élaboration d'un planning des tâches à réaliser et surtout à leur ordonnancement.

Pour gagner du temps, le lecteur aguerris à la gestion de projet informatique et impliqués dans les plateformes du département « Environnement et Agronomie » pourra passer le présent préambule, ainsi que le chapitre suivant.

1.1 L'analyse fonctionnelle

L'analyse fonctionnelle -qui est le sujet principal de ce document- consiste à définir et valider le quoi -ce qu'il faut faire-. Tout ce qui est de l'ordre du comment -dans le cas ou il ne représente pas une contrainte- sera exprimé lors de l'étape suivante : la conception.

L'analyse fonctionnelle est basée sur l'étude -par l'informaticien- du cahier des charges fonctionnel en interaction avec les scientifiques du groupe projet pour précisions. L'objectif est de définir -le plus exhaustivement- les spécifications de base d'un service à réaliser. Ainsi les limites et contours de la plateforme seront exprimés dans ce document. L'analyse fonctionnelle sert à formaliser les besoins et à les expliquer aux différents acteurs du projet, pour s'assurer que tout le monde est d'accord. Elle est donc à juste titre considérée comme un référentiel contractuel partagé. Ce qui en fait un outil fondamental de communication pour le chef de projet informatique.

L'analyse fonctionnelle énumère les contraintes techniques avérées. Toutefois, elle ne doit pas confondre préférences et contraintes. D'où la nécessité d'arbitrer et donner des priorités. Toute incompréhension ou remise en cause peut être aussi dramatique qu'elle est tardive.

1.2 La formulation des fonctions

Les fonctions attendues peuvent être de trois ordres : principales/secondaires, contraintes ou complémentaires. Le recensement -le plus complet- de ces fonctions est important. Il permet d'effectuer un dimensionnement correct du projet -le plus tôt possible- sans interdire ou limiter les évolutions futures. La formulation des fonctions doit être à la fois indépendante des solutions susceptibles de la réaliser -sauf si c'est une contrainte exprimée- et la plus claire possible.

5 Document complété par les différents compte-rendus des réunions de recadrages déroulées durant le 1er semestre 2009.

1.2.1 Les fonctions principales et secondaires

Aussi appelée fonction d'usage, c'est la fonction qui satisfait le besoin. Elle assure la prestation du service rendu. Une fonction principale peut être répartie en plusieurs fonctions élémentaires. Une fonction élémentaire correspond à une action attendue pour répondre à un élément du besoin, traduisant la raison d'être d'un sous-système.

1.2.2 Les fonctions contraintes

La contrainte, c'est la limitation à la liberté de choix du concepteur-réalisateur informatique. Il s'agit de recenser les conditions qui doivent être impérativement vérifiées, mais qui ne sont pas sa raison d'être.

1.2.3 Les fonctions complémentaires

La fonction complémentaire facilite, améliore, ou complète le service rendu. Ce type de fonction ne résulte pas de la demande explicite, et n'est pas non plus une contrainte. Il s'agit de proposer des améliorations[6].

Il est nécessaire de faire la distinction entre la partie logicielle qui répond aux besoins (ex : modéliser des phénomènes dans le sol) et de celle qui s'occupe de techniques informatiques (ex : gestion des E/S)

1.3 Remerciements, retours d'expériences et sources d'inspiration

Ce document est le fruit de l'application d'un certain nombre de méthodes informatiques qui guident et facilitent l'analyse, notamment UML[7]. Mais c'est aussi le résultat d'années d'expériences, propres à l'auteur ou à son entourage direct et indirect. Contributions qui donnent du recul, à la fois sur les activités de développement, mais aussi dans une certaine mesure sur celles de gestion de projet informatique.

Ce document est aussi enrichi d'expériences, parfois vécues de l'intérieur au titre de participations temporaires, dans le cadre de plateformes[8] à usage scientifique. Notamment celles à fortes composantes informatiques dont principalement :

- RECORD (« **RE**novation et **CO**o**RD**ination de la modélisation de cultures pour la gestion des agrosystèmes », http://record.toulouse.inra.fr/) ;
- SEVE et son implantation avec PALM (« **P**rojet d'**A**ssimilation par **L**ogiciel Multiméthodes », http://www.cerfacs.fr/globc/PALM_WEB/) ;
- OpenFluid (« Software Environment for Modelling Fluxes in Landscapes », http://www.umr-lisah.fr/openfluid/) ;
- OpenAlea (« Software Environment for Plant Modeling », http://openalea.gforge.inria.fr/).

Il y a aussi la plateforme INRA-ICATA-ACTA et plus récemment le RMT Modélisation (« modélisation et logiciels d'intérêts communs appliqués à l'agriculture »,

6 Dans le cadre d'une prestation fournie par une « Société de Service en Ingénierie Informatique » (SS2I), ces fonctions sont intentionnellement omises. Elles pourront alors faire l'objet d'avenants, très lucratifs.

7 UML = “Unified Modeling Language”. Voir la section32: Analyse à l'aide du langage de modélisation UML.

8 L'auteur signale au passage qu'il préfère au vocable « plateforme » celui de projet ou programme scientifique, éventuellement complété de l'expression « à forte composante informatique ».

http://www.modelia.org/).

Dans un tout autre ordre de grandeur, la somme des expériences acquises auprès de modèles intégrés[9] -dont le département EA possède un nombre important- est une source intéressante. Notamment parce que ces modèles cherchent -a priori comme « sol virtuel »- à assembler un grand nombre de sous-modèles, par un système de couplage qui se veut -par opposition aux coupleurs génériques- simple d'usage et parfaitement adapté aux besoins propres. L'objectif dans ce cas est non seulement de rendre la somme des services des sous-modèles, mais aussi de faire apparaitre des propriétés émergentes, dues au grand nombre des systèmes dynamiques comportant des rétroactions.

Dans le département EA, l'auteur peut citer les modèles :

- PASTIS (« Predicting Agricultural Solute Transport in Soils », http://w3.avignon.inra.fr/pastis/) ;
- STICS (http://www.avignon.inra.fr/AGROCLIM_STICS) ;
- Alexis (http://w3.avignon.inra.fr/alexis/) ;
- RATP (« RAyonnement, Transpiration et Photosynthèse », UMR PIAF, INRA Clermont-Ferrand).

L'analyse informatique de « sol virtuel » est -entre autre- formalisées par le langage de modélisation UML. L'action FORMINFO du « Réseau des Informaticiens du département EA » (RIEA) a récemment mis en place deux sessions de formations autours de ce langage, que l'auteur à suivi :

- en 2007 était organisée une semaine de formation, dans le cadre d'une école informatique intitulée « Fonctionner en équipe dans un projet informatique » ;
- en 2008, s'est déroulé une session de trois jours comportant de nombreux travaux dirigés intitulée « Comment utiliser UML pour modéliser un problème ».

Des remerciements tout particuliers au projet PLUME[10]. Dont le site Web et la première action de formation[11] furent très instructifs pour le début du projet « sol virtuel ».

9 Un modèle intégré est défini ici comme un groupe de sous-modèles qui peuvent communiquer entre eux.

10 Le projet « Promouvoir les Logiciels Utiles Maitrisés et Économiques dans l'Enseignement Supérieur et la Recherche » (http://www.projet-plume.org/) est une initiative hébergée et soutenue par le CNRS.

11 La première édition de ENVOL (« formation pour le dEveloppemeNt et la ValOrisation des Logiciels en environnement de recherche ») s'est déroulée, pendant une semaine, fin 2008.

2 Introduction

2.1 Historique, contexte scientifique

En quelques mots, le programme « sol virtuel » a été initié en avril 2006 par le département EA (document initial réf. EA : LB – JB – FG / 3478 – 0406). Son objectif : favoriser l'émergence de modèles ou modules de simulation numérique du fonctionnement des sols et de l'interface sol-plante. Cela implique d'encourager l'association de modélisateurs et de spécialistes de l'observation ou de mécanismes et de proposer des couplages de processus en interaction dans les sols.

La solution retenue fut de mettre en place une plateforme de modélisation -a priori à forte composante informatique[12]- à vocation transversale aux unités, destinée à supporter les modélisations du programme et d'autres modélisations éventuelles en permettant de mutualiser les outils et savoir-faire. Le première action fut de constituer quatre groupes de travail, chacun sur une thématique scientifique précise.

Pour citer quelques dates importantes :

- Une réunion initiale du programme « sol virtuel », associant la plupart des unités du département EA en rapport avec le sol, s'est déroulée le 9 octobre 2006 (doc de synthèse réf. EA : LB – JB – FG / 3762 – 1106).
- Le 9 novembre 2006, une lettre de mission du département EA initie des groupes de réflexion par champs thématique (réf. EA : LB – JB - FG / 3763 - 1106).
- Le 18 février 2008, le département EA a constitué un groupe de scientifiques -issu des groupes de réflexion et représentatif des différentes disciplines intéressées par le projet- en charge de la rédaction du cahier des charges fonctionnel de la plateforme « sol virtuel » (réf. EA : LB – FG / 4289 – 0208).
- Après une tournée de consultation des unités concernées, ce document fut rendu au département EA le 29 octobre 2008.
- Le 30 janvier 2009, le département EA a mis en route la phase opérationnelle du projet « sol virtuel » et son mode de fonctionnement envisagé, au travers d'un courrier destiné à tous les directeurs d'unités EA.

2.2 Une démarche de gestion de projet souple

Pour un projet de l'ampleur de « sol virtuel », s'inscrire un minimum dans une démarche de gestion de projet est obligatoire. La vocation de ce document est d'y inscrire la partie informatique. Compte tenu de l'environnement de type « recherche », nous retiendrons comme premier critère d'utiliser une méthode qui sache être souple et s'adapter aux spécificités et contraintes du projet. Notamment une échelle de temps prévue de plusieurs années -avec des échéances espacées- et des moyens -notamment humains- modestes, tant en nombre qu'en compétence.

Suivre une démarche de gestion de projet, c'est passer par différentes étapes clairement identifiées. Pour simplifier, elles peuvent être classées en trois catégories, forcément liées et qui se succèdent :

- les cahiers des charges

12 L'auteur pense que la plupart des modèles scientifiques dans le département EA ont ou voudraient une représentation informatique.

- les études informatiques
- la réalisation informatique

2.2.1 Les cahiers des charges

Le premier cahier des charges est fonctionnel. C'est à dire qu'il décrit les besoins, les fonctions scientifiques attendues. Le deuxième est -par opposition au premier- non-fonctionnel. C'est à dire qu'il décrit tout ce qui n'est pas de l'ordre du besoin. On y exprime particulièrement les contraintes techniques. Ces documents sont principalement construits par les utilisateurs et les commanditaires du projet. Il est cependant souvent utile d'y faire participer des experts -dans notre cas informatiques- qui guident la rédaction pour se poser les bonnes questions qui donneront des réponses lors de la phase suivante : les études informatiques.

2.2.2 Les études informatiques

L'analyse informatique -le présent document- est une étude directe du cahier des charges fonctionnel. C'est une interprétation des besoins exprimés, totalement dissociée des aspects techniques, c'est à dire informatiques. La seconde étude -qui fera suite à la présente analyse- est donc liée à la technique. Ce sont des choix -des arbitrages- sur les outils, les langages et les usages informatiques.

2.2.3 La réalisation informatique

La réalisation informatique comprend le codage proprement dit du projet. C'est à dire l'élaboration de code informatique propre à fournir un (ou plusieurs) exécutable -à l'aide des choix techniques- qui réponde aux besoins. Avant la livraison « publique », la réalisation passe par des étapes de validation :

- unitaires, par briques logicielle indépendantes ;
- d'intégration, à chaque fois qu'une brique rentre dans « le tout » ;
- par des utilisateurs désignés-volontaires, qui doivent tester et comparer les résultats avec les attendus.

2.3 Une démarche souple

Suivre une démarche, c'est être guidé. Cela implique de la rigidité. Les techniques récentes de gestion de projet se montrent plus souples que par le passé. Elles introduisent très explicitement les inconnus et les aléas dus à des facteurs extérieurs -non maitrisables par nature. Le plus important d'entre-eux est le facteur humain : humain donc perfectible. Dès l'expression des besoins, ces méthodes admettent que -quel que soit le temps que l'on y passe- l'expression des besoins n'est jamais exhaustive, ni même sans ambigüités.

Tout d'abord, l'expression des besoins n'est jamais exhaustive parce que les besoins évoluent avec le temps. De plus, il est impossible de tout spécifier complètement -surtout quand il s'agit d'interface utilisateurs- sauf cas triviaux. Ceci est encore plus vrai dans le contexte d'un institut de recherche dont l'objectif n'est pas -par définition- de fournir des solutions finalisée.

Ensuite, l'expression des besoins n'est jamais sans ambigüités parce que -quel que soit le formalisme utilisé- il contient inévitablement des doubles sens, des expressions subjectives. Les auteurs peuvent souvent croire -à tort- que tout le monde met les mêmes matérialités sous une

même expression. Ces malentendus sont parfois des cas difficiles à détecter.

Éviter ces « erreurs » est impossible. Il faut donc accepter la nécessité de progresser petit à petit, par essai/erreur. De ce constat découlent les méthodes de conception itératives, dans lesquelles aucune définition des besoins ou réalisation n'est définitivement acquise. Au lieu de prévoir un plan d'ensemble -où tout est parfaitement organisé- il s'agit de procéder par touches successives, de s'autoriser des corrections, des re-cadrages inévitables. Que ce soit sur les réalisations comme sur les définitions des besoins.

L'esprit général de ces méthodes dites « agiles » -décrites dans la section suivante- consiste donc à produire rapidement un logiciel -aux fonctionnalités partielles- ; à le présenter aux utilisateurs intégrés dans l'équipe projet ; à les faire réagir ; à noter les changement à apporter pour la prochaine version ; et ainsi de suite. Ainsi le logiciel progresse par essais-erreurs[13] successifs. Les besoins sont donc en phase -recalés- avec la production logicielle. Ce qui au final s'avère souvent salvateur, même pour des projets en apparence peu évolutifs.

2.4 Les méthodes agiles

Les méthodes agiles semblent donc plus appropriées -efficaces- pour un projet comme « sol virtuel ». Si elles sont largement utilisées de nos jours, elles ne sont -dans l'esprit- pas si récentes que cela. En effet les méthodes dites « agiles » -comme « eXtreme Programming » (XP/XUP) ou « Rapid Application Development » (RAD)- s'inspirent très largement du développement dit « en spirale », qui fut enseigné et utilisé dans les années 1980-1990, notamment dans le développement de modèles scientifiques.

2.4.1 Les grands principes

Toutes les méthodes agiles se basent sur quatre grands principes :

- Spécifier complètement les besoins avant toute réalisation informatique -c'est à dire dès le départ du projet- est cher et de toute façon utopique (cf. section 2.3 , page 9).
- L'informaticien -comme tout humain- est perfectible dans son interprétation des besoins.
- Le « client » se lasse s'il ne voit rien venir.
- Un grand tout logiciel peut être construit de briques qui viennent petit à petit, par ordre d'importance.[14]

2.4.2 Les quatre grandes étapes

Les méthodes agiles définissent quatre étapes principales, se succédant :

1. Se donner des objectifs et des besoins à différents termes. C'est à dire répondre au présent, sans hypothéquer l'avenir. Une pondération doit donc être mis en place. Elle doit tenir compte de ce qui est plus ou moins réalistes et faisables.
2. Mettre le focus -les moyens- sur les objectifs et les besoins les plus importants, les plus immédiats et les moins risqués.
3. Produire rapidement un résultat -un logiciel- évaluable par les utilisateurs.

13 Les anglo-saxons qui ont mis au point les première méthodes de ce type utilisent l'expression « trial and error ».

14 Ce qui ne veut pas dire que l'objectif global n'est pas spécifié. Il est simplement informe au début et ne sera détaillé qu'au fur et à mesure de l'avancement du projet. Sa forme ne sera définitive que lorsque le projet sera terminé.

4. Faire réagir les utilisateurs par rapport à la réalisation. Énumérer les remarques, les nouveaux besoins, les ambigüités... Comparer aux besoins initialement définis. Noter les nouveaux, ceux en évolution. Apprendre de ses « erreurs » et repartir au point 1.

Dans le cadre de ces méthodes, l'industrie informatique utilise souvent des cycles qui se comptent en semaines. La littérature parle souvent de deux semaines. Mais leurs moyens et leur compétences disponibles sont tout autres, comparées à celles disponibles dans un institut comme le notre. Dans le contexte du projet « sol virtuel », il faut tenir compte de deux facteurs qui augmentent considérablement cette durée :

1. La plupart des acteurs -même ceux officiellement à temps plein sur le projet- partagent leur temps avec d'autres activités, parfois tout aussi prioritaires. En particulier, les scientifiques qui ne sont pas en charge de la plateforme -ou pour qui elle n'est pas prioritaire- pourraient ne donner qu'un temps très limité.
2. L'appel à des moyens humains non permanents -obligatoire dans notre contexte- peut engendrer des délais de mis en œuvre. Le recours à des stagiaires impose de se caler sur la fréquence de disponibilité des écoles et de s'adapter au fait que leur compétence n'est généralement pas au niveau de professionnels aguerris. L'engagement d'ingénieurs en développement peut poser des contraintes administratives -notamment pour la prévision et l'allocation des budgets suffisamment à l'avance- en plus de celle qui consiste à trouver celui qui corresponde bien au profil, les salaires de la fonction publique n'aidant évidemment pas.

Pour toutes ces raisons, une nouvelle version majeure[15] sur une fréquence annuelle -ou semestrielle- semble être raisonnable. S'imposer une fréquence trop courte empêcherait de travailler avec sérénité et surtout de consacrer du temps à d'autres activités -dans « sol virtuel »- tout aussi importantes :

- correction de bogues et améliorations mineures[16] ;
- accompagnement et assistance aux utilisateurs, en particulier des nouveaux ;
- soutenance des sessions de formations.

2.5 Définitions

Le cahier des charges fonctionnel possède un ensemble de définitions indispensables à l'élaboration de cette analyse informatique. Il est nécessaire d'en ajouter de nouvelles. Quelques-unes paraitront évidentes au lecteur, elles éviteront peut-être des incompréhensions et des malentendus.

2.5.1 Scientifique

Parfois appelé « chercheur », le scientifique est -de manière générale- une personne qui se consacre à l'étude d'une science. Un moyen important de mettre en valeur son action est de produire des publications scientifiques.

Dans le cadre de ce projet -sans être réducteur-, cet agent peut souhaiter utiliser un modèle pour résoudre une question scientifique. Il peut aussi -de par ses compétences scientifiques- être

15 Les évolutions majeures apportent de nouvelles fonctionnalités, voire restructurent l'application.
16 Les évolutions mineures apportent des corrections de bogues ou des ajouts de fonctionnalités secondaires

capable d'élaborer et d'agencer des processus, donc des squelettes. Principalement à l'aide de formalismes ordonnés, des équations mathématiques.

S'il a des aptitudes en calcul numérique et en informatique, le scientifique peut aussi participer à l'introduction de modules dans la plateforme. Toutefois, il n'accorde -à juste titre- que peu d'importance aux aspects informatiques, que se soit pour construire ou utiliser un outil informatique. C'est pour lui un moyen -qui doit rester simple- d'arriver à son objectif scientifique.

2.5.2 Informaticien en développement

Ce vocable représente en fait une personne de type « ingénieur en développement d'applications informatiques ». En contraste à la définition du scientifique, c'est une personne très impliquée dans les techniques en informatiques de développement, mais pas seulement. Son travail lui demande aussi d'adopter des démarches et des modes d'organisation du travail informatique appliqués au contexte : recherche, conception de systèmes, production et maintenance.

Il peut avoir des aptitudes en calcul numérique et même des compétences scientifiques suffisantes pour comprendre les formalismes mathématiques et les réalités scientifiques qu'elles modélisent. Son objectif premier reste la réalisation informatique. Ce qui peut inclure, à l'échelle de la mission qui lui est confiée, des compétences en gestion de projet.

La principale motivation de l'informaticien passe par la satisfaction d'une construction technique bien accomplie[17] et qui répond le mieux possible aux besoins fonctionnels initiaux à ceux exprimés mais aussi à ceux pressentis. Si son gout prononcé pour la « mécanique informatique »[18] est un plus, elle ne doit pas lui donner des tendances à quelquefois trop la privilégier[19].

Dans le cadre de ce projet, son métier consiste en particulier à traduire des formalismes ordonnés -expressions mathématiques- en du code informatique exécutable. Ce dernier doit être un logiciel utilisable « simplement » par un non informaticien. Soit directement, au travers d'une interface utilisateur -le plus souvent graphique- soit indirectement, au travers d'un environnement logiciel tiers -un environnement dédié, une plateforme, comme R par exemple.

Outre l'aspect purement technique, la qualité de son travail passe notamment par la traçabilité. Celle-ci est souvent assurée par des algorithmes et des documents connexes (cf définition 2.5.3 , page 12).

2.5.3 Algorithme

De manière générale, c'est un processus systématique de résolution -par le calcul- d'un problème permettant de présenter les étapes vers le résultat. En d'autres termes, un énoncé d'une suite d'opérations pouvant être exécutées en séquence, en parallèles ou réparties/distribuées.

Dans le cadre de ce projet, il s'agit d'obtenir une suite ordonnée d'équations mathématiques, facilement traduisibles en un langage informatique de plus ou moins haut niveau. C'est donc une forme intermédiaire, entre la description scientifique -tout en restant proche de la publication- et le langage informatique. Un algorithme possède une propriété intéressante : il est indépendant de tout code informatique de toute réalisation concrète.

Élaborer des algorithmes est une bonne pratique. Cette forme de description a l'immense avantage d'être à la fois compréhensible par les informaticiens et les scientifiques. C'est en quelques

17 Terme volontairement vague qui recouvre plusieurs notions comme la concision de l'écriture, l'exactitude des résultats, la lisibilité, la stabilité face aux conditions extrêmes/limites/imprévues...

18 Sans vouloir faire du sexisme, ce phénomène est souvent observé du coté du genre masculin.

19 NDRL : sans vouloir faire du sexisme, d'où l'intérêt d'avoir une composante féminine dans une équipe informatique.

sortes un langage pivot.

Pour avoir un réel intérêt, un algorithme doit toutefois être accompagné d'informations complémentaires, réunies dans des documents connexes :

- Le dictionnaire de données -entrées, sorties et paramètres de l'algorithme- avec notamment les bornes min/max, les domaines de validité, les unités...
- Les méthodes de résolution des équations utilisées, avec -si besoin est- les conditions techniques de fonctionnement idéales[20].

2.5.4 Paramètre

Le terme « paramètre » -évoqué à plusieurs reprises dans le cahier des charge fonctionnel- est souvent utilisé pour désigner des sujets différents. Pour éviter toute méprise, il mérite une définition, au sens informatique.

De manière générale, un paramètre est une donnée entrant dans une équation mathématique ou plus largement dans un algorithme. En d'autres termes, un paramètre est donc un élément d'information à prendre en compte pour prendre une décision ou pour effectuer un calcul. Le paramètre est -par opposition à la variable dont la valeur est susceptible d'être modifiée en cours d'exécution du calcul- constant durant tout l'exécution du programme informatique.

De manière moins stricte, on peut considérer qu'un paramètre est simplement constant à l'échelle d'une fonction. C'est donc un paramètre pour la fonction qui l'utilise -par exemple la fonction qui représente un module- mais une variable pour l'appelant de la fonction -qui peut décider de changer sa valeur entre chaque appel au module concerné-.

20 Exemples : haute précision sur les nombres réels (128 bits), processeur 64 bits, environnement multi-thread.

3 Étude du cahier des charges fonctionnel

3.1 Éclaircissements sur les termes employés

3.1.1 Processus et état d'un processus

Un processus est identifié de manière unique par un nom. Exemples : transfert hydrique, transfert de chaleur, décomposition de la matière organique, transferts de solutés, transferts de gaz, etc.

A la définition donnée à ce terme dans cahier des charges fonctionnel (section 1.3) est rajouté la notion d'état. Un état est définit au travers des valeurs des variables d'état du processus. Les variables d'un processus dépendent -entre autres- de celles d'autres processus, en amont. Cela amène à définir -pour chaque processus- les variables d'entrées et de sorties. L'état d'un processus est représenté par ses variables de sorties.

Toute variable est identifiée par un nom unique (exemple : Θ). Elle possède une unité, un type (booléen, entier et réel) et des bornes de validité[21]. Chaque variable d'état dépend du temps et de sa position dans l'espace (1D, 2D ou 3D).

Toute variable de sortie est calculée par une fonction qui dépend à la fois de variables d'entrées -au même temps ou au temps précédent- et d'un ensemble équations algébriques ou différentielles. Ces équations correspondent à une modélisation du processus. Elles utilisent donc les variables d'état du processus et des paramètres qui eux dépendent de la modélisation qui est faite. La résolution des équations requiert :

- au temps précédent, la valeur de la variable de sortie au même point -pour les processus locaux- et en plus aux point adjacents -pour les processus non locaux (par exemple les phénomènes de transfert qui font apparaitre des dérivées en espace).
- au temps courant, la valeur de la variable de sortie au même point -pour les processus locaux- et en plus aux point adjacents -pour les processus non locaux (cas des processus non linéaires).

3.1.2 Module

Un module est obligatoirement rattaché à un et un seul processus. Il définit en plus :

- une conceptualisation/modélisation. Ex : transfert hydrique avec la modélisation par « réservoirs », « Richards », « écoulements préférentiels »...
- une technique de résolution numérique. Ex : transfert hydrique avec la modélisation par « Richards » et une résolution utilisant les « différences finis »
- une réalisation informatique, cad le nom donné au code informatique qui le réalise. Ex : processus hydrique avec la modélisation par « Richards », une résolution utilisant les « différences finies » est inscrit dans le modèle PASTIS. Il se peut que la réalisation informatique soit vide, c'est à dire inexistante
- Des méta-informations : échelles de temps et d'espaces, sensibilité aux entrées, temps de

21 Ce peut-être des bornes strictes -la simulation est en erreur lorsqu'elles sont dépassées- ou des valeurs « normales » -la simulation alerte l'utilisateur d'un comportement potentiellement anormal lorsqu'elles sont dépassées.

calculs, précision...

3.1.3 Typologie des modèles informatiques

Dans le cadre des unités de recherche des départements, le développement de modèles va souvent de pair avec celui de développements logiciels associés. Sauf exceptions, ces implantations informatiques sont l'objet d'acteurs qui peuvent être de deux types : soit un scientifique, possédant des aptitudes en développement informatique (cf. 3.1.5), soit un informaticien en développement. Toutes modalités intermédiaires étant possibles.

Lors d'une de ses présentations -aux « Journées des Informaticiens du département EA » 3ème édition, à Orléans en 2006-, Guy Fayet[22] dressait une typologie des implantations logicielles des modèles produits dans les unités des départements de recherche. Cette classification prend toute son importance avec la montée en charge des programmes inter-unités et des plateformes lancées par le département EA. On peut donc classer en quatre catégories, de « qualité informatique » croissante, les implantation informatiques des modèles du département EA. Chaque catégorie étant le préalable à la suivante :

1. modèle « prototype de chercheur »
2. modèle « logiciel d'unité »
3. modèle « logiciel inter-unité »
4. modèle « logiciel commercial »

La première catégorie de ces « modèles informatiques » est le prototype. Il est directement issu de formalismes scientifico-mathématiques. Il est généralement le produit du scientifique, qui veut valider rapidement et facilement ses hypothèses scientifiques et produire une publication. Il le produit donc pour son usage direct. Dans ce contexte, la flexibilité du code et sa maitrise -par le scientifique concepteur- sont des points essentiels. Le développement logiciel est une vue à court terme et la qualité informatique de la production -qui n'est pas le sujet- est jugée comme secondaire. Le code informatique n'a pas vocation à être diffusé et doit donc être vu comme jetable.

La seconde catégorie est le modèle d'unité. Il est issu de l'agglomérat de modèles produits par l'unité. C'est donc généralement le résultat d'un assemblage des prototypes informatiques. Les deux grands objectifs sont d'une part de fédérer l'unité autour de ce modèle et d'autre part de diffuser auprès des scientifiques de l'unité et des unités partenaires. Les questions de couplages scientifique puis informatique sont au centre des réflexions, mais dans un cadre scientifique maitrisé. L'assemblage est co-construit en interne et doit « résister » scientifiquement et techniquement aux mises à jour et évolutions futures : modification d'un prototype, ajout de nouveau prototype.... Dans ce contexte unité, la difficulté technique de couplage n'est pas liée à la diversité des langages informatiques utilisés pour l'implémentation informatique des prototypes mais à la robustesse du couplage aux modifications. En fait, il est souvent indispensable de réaliser des modifications -autant scientifiques qu'informatiques- aux prototypes pour être intégrés/couplés avec d'autres.

La troisième catégorie est le modèle inter-unité. Les modèles doivent être assemblés, adaptés pour répondre à des contextes d'utilisation plus larges. Par exemple pour répondre à des questions d'un champ thématique du département EA. La construction et la diffusion à l'échelle de la communauté scientifique sont donc la priorité. L'esprit de concertation et de prise de décision collective sont la règle dès que l'on touche aux inter-actions entre modèles. L'outil informatique

22 Responsable et figure emblématique de la partie développement de ce qui fut longtemps appelé la « Mission Informatique » à l'INRA.

produit doit être à la fois souple -pour accepter les évolutions futures- mais aussi rigide -pour éviter un ébranlement de l'ensemble. Les plateformes scientifico-informatiques du département EA -donc « sol virtuel »- en sont une forme possible.

La dernière catégorie est hors sujet -ou du moins prématurée- par rapport au sujet de cette analyse informatique de la plateforme « sol virtuel ». C'est le niveau « grand public ». La qualité informatique doit être de type commercial.

L'idée de cette classification est de montrer que le niveau de qualité informatique demandé aux produits finaux est très différent, et donc les compétences des acteurs intervenant dans chacune de ces catégories aussi.

La troisième catégorie est issue de l'émergence de prototypes de première et/ou seconde catégories. La troisième catégorie n'est pas forcément une finalité. En effet, un prototype intégré dans un modèle unité et/ou une plateforme peut encore exister dans sa forme initiale. Il est plus aisé -et motivant- d'apporter des modifications à son prototype seul et de les valider que dans un contexte scientifique et technique plus complexe. Garder cette « double vie » des modèles -à la fois dans les unités et dans la plateforme- ne doit donc pas être vu comme un gaspillage d'énergie. C'est plutôt un incitateur à apporter des améliorations. S'il y trouve de l'intérêt, le scientifique pourra alors transférer régulièrement dans la plateforme les nouveautés produites localement. La plateforme joue alors pleinement sont rôle de point de partage et de bénéfice pour toute la communauté qu'elle rassemble. Les scientifiques non producteurs de modèles prototypes profitent des modèles prototypes et les valorisent dans leurs productions -publications- scientifiques.

3.1.4 Modèle de plateforme

La gestion des erreurs -qui n'est pas le propos d'un prototype informatique- prend tout son sens dès qu'a lieu une diffusion plus large que celle de son seul concepteur. Un modèle issu d'une unité de recherche -lorsqu'il s'insère dans une plateforme comme « sol virtuel »- doit obligatoirement répondre à des critères de qualité. En plus de ceux scientifiques -hors de propos dans cette analyse informatiques- les exigences informatiques ne doivent pas être prises à la légère.

En effet, l'assemblage de nombreux « bouts de code » -venant de sources très différentes et conçus dans des conditions plus ou moins favorables- dans un tel projet, exige un cadre et des règles à même de garantir une exécution fiable -d'un point de vue informatique- pour l'utilisateur final. Il n'est pas envisageable que l'utilisateur puisse croire que les résultats sont corrects alors même que certaines modules sont en erreur ou ont levés des alarmes. La plateforme doit exiger de ces modules qu'ils déclarent les cas d'erreurs sur leurs entrées/sorties. Un plus serait qu'ils détectent et fassent remonter leurs problèmes internes. Ceci entraine à la fois un travail dans le code proprement dit et un travail de documentation, sur les entrées/sorties et les cas d'erreurs.

Le travail de documentation informatique -en plus de celui scientifique- doit permettre à un développeur informatique de « rentrer dans le code » simplement. Il ne doit pas être obligé de faire de la « rétro-ingénierie[23] » pour en comprendre déduire -potentiellement mal- le fonctionnement général du modèle. Cela passe par trois types de documentation intimement liées : l'algorithme, la documentation de conception et la documentation dans le code.

23 Plus connu sous l'anglicisme « reverse engineering », ce terme désigne l'activité qui consiste à étudier un objet -ici un code informatique- pour en déterminer le fonctionnement interne ou sa méthode de fabrication.

3.1.4.1 L'algorithme

L'algorithme est la description du processus systématique de résolution, par le calcul, du problème permettant de présenter les étapes vers le résultat. En d'autres termes, un algorithme est un énoncé d'une suite d'opérations permettant de donner la réponse à un problème. Dans notre cas, cela inclut les informations liées aux entrées/sorties : unités, bornes de validité...

3.1.4.2 La documentation de conception

La documentation de conception ou documentation technique -les deux sont en général confondues- consiste à décrire les choix techniques mis en œuvre et les instructions techniques d'utilisation. En général, la documentation générale est suffisante, à condition que la documentation du code soit bien réalisée. Sinon, il faut une documentation de conception détaillée.

3.1.4.3 La documentation dans le code

La documentation dans le code comprend tous les commentaires qui accompagnent le code source. Outre les commentaires -toujours utiles- dans le corps des fonctions, ce sont les commentaires associés aux fonctions -et aux classes dans le cas de programmation objet- qui sont les plus indispensables. C'est ce que l'on appelle l'API[24]. C'est à dire l'ensemble des fonctions, procédures ou classes mises à disposition par un programme informatique, une bibliothèque logicielle, un système d'exploitation ou un service. La connaissance de l'API est indispensable à l'interopérabilité entre les composants logiciels, donc les modules et la plateforme.

L'API doit donc être accompagnée d'une documentation -directement dans le code- qui décrit pour chaque fonction :

- son rôle, au minimum par une brève description ;
- ces entrées et sorties, ainsi que leur rôle ;
- la valeur de retour
- les cas d'erreur, ainsi que les exceptions possibles ;

Dans le cas de « sol virtuel », les fonctions concernées sont principalement celles des modules. Ce peut aussi être le code source liés aux modèles. De manière générale, tout le code de la plateforme devra être commenté de la sorte.

De nombreux outils permettent de générer une documentation lisible à partir du code source documenté. Les outils logiciels « Javadoc » et « Doxygen » sont des références en la matière. Ils permettent de générer de nombreux formats de documents riches -comportant des graphiques- automatiquement.

3.1.5 Développement informatique

Dans le cadre de ce document d'analyse informatique, le terme développement se rattache naturellement au domaine informatique. De manière générale, le développement logiciel comprend l'ensemble des étapes et mécanismes qui permettent de passer de l'expression d'un besoin informatique à un logiciel fonctionnel et fiable. Dans notre cas, il faut distinguer le développement au niveau module et au niveau plateforme (infrastructure proprement dite).

Le niveau plateforme ou infrastructure est le projet qui permettra l'articulation et le couplage

24 « Application Programming Interface » ou en français « Interface de programmation »

des modules. Les étapes amont -analyse informatique des besoins et conception générale- et aval -tests d'intégration et validation/livraison- seront réalisées dans le cadre de ce projet.

Le niveau module est la mise en place des modules dans la plateforme. C'est principalement un travail d'insertion de code informatique : soit d'adaptation à partir de codes existants ou d'un code crée « ex nihilo ». Ce niveau est une activité technique forte. Il met donc le focus sur les étapes de conception détaillée, (re-)codage dans un langage informatique et les tests unitaires, pour le développement des modules.

3.2 Les acteurs et rôles du projet

Le terme acteur englobe les intervenants extérieurs au système informatique étudié et qui interagissent avec lui. Dans notre cas, ce sont des personnes physiques[25]. Un individu peut-être successivement plusieurs acteurs, mais jamais simultanément. Un acteur joue un rôle, il est donc dans un contexte -un certain état d'esprit- et tente de résoudre ses objectifs en conséquence.

Dans la majorité des cas, une classification des acteurs par métier est naturelle et pertinente. Elle est donc utilisée dans cette analyse, pour faire la distinction entre la famille des acteurs scientifiques et celle des informaticiens de développement. Mais cette classification n'est pas suffisante puisque, dans chaque famille, différents rôles sont clairement identifiables.

3.2.1 Informaticien de plateforme

Acteur de type « ingénieur en informatique de développement », il travaille sur l'infrastructure -informatique- de la plateforme. Son objectif premier est de réaliser puis d'assurer que la plateforme est utilisable, par les autres acteurs, dans les conditions qui ont été fixées avec les responsables de la plateforme. Il peut être amené à aider les autres acteurs à la création ou au transfert de modules dans la plateforme. Son préoccupation est de garantir la cohérence et le bon fonctionnement, au moins informatique, de la plateforme, y compris sur le long terme. Cette garantie est inversement proportionnelle à l'autonomie qu'il peut laisser aux acteurs de type « informaticien de modèle d'unité » dans la plateforme.

3.2.2 Informaticien de module

De type « ingénieur en informatique de développement » ou scientifique possédant une bonne aptitude à la programmation informatique, cet acteur a comme objectif principal la création ou le transfert de réalisations informatiques -modules- dans la plateforme. Au départ, ce sont la plupart du temps des modèles d'unités qui deviendront alors modules de « sol virtuel », avec un rattachement à un processus identifié. Son autonomie sur cet objectif est directement proportionnelle à sa compétence en informatique de développement. Elle peut-être totale, s'il est suffisamment autonome d'un point de vue informatique.

3.2.3 Scientifique de plateforme

Ce type d'acteur agit en concertation avec la communauté et le groupe projet de la plateforme. Son rôle se situe à deux niveaux intimement liés : processus et squelettes (voir cahier des charges fonctionnel, chapitre 1.3). Il est responsable de la bonne cohérence scientifique d'une partie de la plateforme, suivant ses compétences scientifiques.

25 De manière générale, ce peut aussi être -en plus des personnes physiques- des systèmes informatiques.

3.2.3.1 Intégrer des processus

Le « scientifique de plateforme » peut introduire des processus nouveaux, c'est à dire des phénomènes, par le biais d'entrées/sorties clairement identifiées et minimales. Les sorties d'un processus lui sont spécifiques/exclusives. C'est à dire que ces sorties ne peuvent venir que du processus et de lui seul. Après avoir reçu l'aval des autres « scientifique de plateforme » du même domaine de compétence, l'emplacement de ce nouveau processus -avec ces entrées/sorties- est créé. A partir de là, des modules peuvent lui être rattachés. Le travail de cet acteur s'arrête une fois qu'il a réalisé le cadre « processus ». La réalisation informatique pour représenter le processus qu'il veut modéliser, sera réalisé par l'acteur « informaticien de module »

3.2.3.2 Construire/modifier des squelettes

Le « scientifique de plateforme » est chargé de la construction de squelettes. Donc toute introduction ou changement dans les processus -dans son domaine de compétences- doit recevoir son aval. De par ses compétences fortes au regard de certains mécanismes du sol, ce type d'acteur est capable d'assembler de manière cohérente des processus. Il a donc la capacité de faire un squelette -parfois partiel- représentant au mieux certains fonctionnements du sol, ou une partie de ceux-ci. Cet assemblage peut tenir compte de contraintes déterminées : échelles de temps/d'espace particuliers, temps de calcul limités, précision des résultats, résultats non divergents...

3.2.4 Scientifique de module

Son action se situe dans le cadre « module ». Il possède et maitrise un code informatique[26] qui réalise un processus de la plateforme. Quel que soit sa forme, le code devra être retravaillé pour être introduit dans la plateforme : documentation, revue du code scientifique proprement dit, code d'interfaçage avec la plateforme... Cet acteur collaborera avec l'acteur « informaticien de module ».

3.2.5 Scientifique de modèle

Cet acteur veut générer une forme informatique -un modèle- qui réponde à une famille de questions scientifiques sur le sol. En fonction de ses compétences -notamment avec certains mécanismes du sol- il est capable de choisir le squelette le plus adapté à ses questions. Pour l'aider dans sa sélection, il peut se baser sur des critères globaux[27], préciser des sorties, des processus et des modules en fonction des besoins. Un fois le squelette déterminé et ses modules tous choisis[28], un modèle utilisable peut être généré et ce pour une architecture informatique donnée. Le résultat peut ainsi être un exécutable autonome, en ligne de commande -pour faire des batchs- ou avec une interface graphique -plus ou moins riche-. Ce peut aussi être un composant dédié à l'inclusion dans un autre logiciel -une autre plateforme- plus « globale ».

3.2.6 Scientifique utilisateur de modèle

Ce type d'acteur -dont le nombre de représentants est directement corrélé à l'intérêt et donc à la réussite de la plateforme- a pour objectif de répondre à une question scientifique -a priori sur le sol- à l'aide d'un modèle informatique. Il souhaite donc utiliser un assemblage cohérent de modules -un modèle- déjà établi par le type d'acteur « scientifique de modèle ». Ce sont les sorties que fournit ce modèle qui lui permettront de répondre à sa question. Pour cela, il doit donner les entrées

26 Ou bien tout autre forme de type publication ou formalisme mathématique

27 Spécifier par exemple : échelles de temps et d'espace, temps de calcul limités, précision des résultats, résultats non divergents...

28 Au cas ou il ne saurait pas choisir les modules de certains processus, le système peut en proposer par défaut.

attendues par les modules qui composent le modèle.

3.3 Études des différents cas d'utilisation

3.3.1 Cas d'utilisation 1 : mise à disposition de modules et de modèles « clé en main »

L'utilisateur recherche ce dont il a besoin -parmi ce qui est disponible- et se sert. Dans ce cas, l'acteur est un consommateur dans le sens où il n'apporte pas de composants à la plateforme. Mais il la légitime. En effet l'utilisation de la plateforme - éventuellement des modèles qu'elle propose- peut apporter des retours intéressants comme l'identification de nouveaux besoins, la remontée d'anomalies ou des jeux de données. La réponse à cette demande est -dans le cahier des charges fonctionnel- imprécise. Elle est précisée dans le compte-rendu de la réunion du groupe projet du 24 mars 2009. Deux scénarios répondent donc à ce cas d'utilisation :

1. La plateforme propose -via son site Web- un annuaire des modèles déjà disponibles « tout prêt » dans le département EA. Au delà de la simple possibilité de télécharger ces modèles, leurs mise à disposition implique un minimum de travail à réaliser par les personnes déposant leurs modèles :
   - documentations scientifiques et techniques ;
   - interface utilisateur, avec si possible -et surtout pour les modèles gourmands en paramètres et entrées- une interface utilisateur graphique ;
   - une aide au choix du modèle par rapport aux autres ;
   - une aide à l'utilisation.
2. La plateforme propose les modules qu'elle utilise pour ses besoins propres. Ils représentent une collection sous la forme de « bout de codes », de fonctions commentées et documentées. L'idée est donc que les scientifiques concepteurs de modules mettent à disposition des autres scientifiques rapidement et facilement un ensemble de fonctions jugées utiles à la communauté. Ces fonctions -qui représentent des phénomènes relativement élémentaires dans le sol- doivent être réutilisables dans des contextes hors plate-forme avec un minimum d'effort. La fonction -au sens informatique du terme- permet cela. La mise en place des fonctions impose que les codes informatiques soient disjoints de leurs modèles originaux[29]. Ce travail de « découpage » par fonctions simples et indépendantes -du contexte d'utilisation- est toutefois nécessaire[30] quel que soit la forme finale que la plateforme « sol virtuel » voudra utiliser. Des compétences significatives en informatique sont donc requises. Le risque majeur lié de ce découpage est la non garantie que le code extrait fonctionne correctement. Cela demande donc de mettre en place des tests de non régression.

3.3.2 Cas d'utilisation 2 : importation d'un module et couplage avec un modèle

L'importation implique que le module est déjà existant hors contexte plateforme. C'est un formalisme mathématique -au moins sous forme de publications scientifiques- auquel est associé

29 D'après mon expérience, un modèle dans une unité est généralement un assemblage de plusieurs phénomènes, donc potentiellement un enchevêtrement de codes entremêlés. Cet effet « plat de spaghettis » est souvent la bête noire des informaticiens lorsqu'ils reprennent un code tiers.

30 Par expérience, ce travail est fastidieux, notamment à cause de l'usage fréquent d'un grand nombre de variables globales.

une forme informatique.

Le module est un « bout de code » ou « algorithme » -potentiellement issu d'un modèle d'unité- à l'usage de quelques scientifiques. Dans le cas où le code informatique original est proposé, un travail d'adaptation sera nécessaire pour obtenir une forme informatique « plateformisée », c'est à dire compatible avec le formalisme des fonctions informatiques imposé par la plateforme. Les modifications/ajouts porteront d'une part sur les aspects informatiques (changement de langage, gestion des entrées/sorties, adaptation de l'apparence du code, commentaires) et d'autre part sur les aspects scientifiques (documentation sur la méthode de résolution, les entrées/sorties, les paramètres..)

Ce travail d'import demande une implication forte du scientifique concepteur. Cette disponibilité doit aussi se retrouver au niveau informatique. En effet, c'est un travail qui peut être conséquent -très intrusif dans le code- et qui peut aller jusqu'à la réécriture complète du module pour la plateforme.

La question de l'évolution d'un même module dans la plateforme se pose lorsque la conceptualisation, la technique de résolution numérique ou le code informatique -ou les trois- évoluent. Ce peut-être pour corriger des bogues informatique ou optimiser l'algorithme. La plateforme doit distinguer deux cas : la correction d'anomalie d'une part et une évolution de concept d'autre part.

Le premier cas est une évolution considérée comme mineure qui ne remet pas en cause les entrées/sorties du module. Dans ce cas, la plateforme rend la précédente version obsolète et ne propose que de garder la nouvelle version, amélioration de l'ancienne. Par rapport à ce premier cas, la plateforme considère le deuxième comme la création d'un nouveau module.

3.3.3 Cas d'utilisation 3 : développement de modèles à partir de modules disponibles

Ce cas exprime la nécessité de mettre en place un système de couplage entre modules. La particularité réside dans le besoin exprimé d'avoir une assistance, un guide, pour mener à bien les opérations demandées.

En première approche, le couplage peut se faire en fonction de dépendances entre modules. C'est à dire en se servant des entrées et sorties des modules pour savoir quel module peut aller avec un autre. Un module ne peut être placé dans la plateforme que si les entrées dont il dépend sont disponibles, que se soit à travers un module amont ou bien des données d'entrées de la plateforme. Ce système permet de garder un maximum de généricité, en ne supposant pas de l'enchainement des modules a priori. L'ordonnancement ne se décide que par les dépendances entre les entrées d'un module aval par rapport aux sorties des modules amonts. On peut d'ailleurs imaginer sans problèmes que certains modules puissent fournir des sorties qui « recouvrent » plusieurs processus. Exemple : un module fournit à la fois l'évolution de la matière organique en même temps que certains solutés.

A noter que le nombre d'alternatives possibles grandissant avec le nombre de modules, cette liberté de couplage pourrait se révéler complexe pour l'utilisateur. Afin de limiter les modèles possibles, il serait souhaitable de le guider par des critères de filtrage[31]. En limitant le nombre de modules y répondant, cela limiterait les assemblages possibles et favoriserait une convergence rapide vers les modèles les plus pertinents.

31 échelles de temps/espace spécifiques, temps de calculs limités, sorties spécifiques, type de processus, 1D/2D/3D...

Même si ce n'est pas très explicite dans le cahier des charges fonctionnel, on peut comprendre qu'un module est la représentation informatique d'un phénomène. Il est donc une réalisation possible d'un -et a priori d'un seul- processus[32]. Si l'on introduit la notion de squelettes -assemblage cohérent de processus- alors le couplage de modules est dirigé par celui des processus. Les contraintes sur les entrées/sorties sont donc en très grande partie « résolues » à l'étape de construction des squelettes, et très peu à celle des modèles.

La réflexion de la représentation des phénomènes dans le sol est laissée au concepteur de squelettes. Le concepteur travaille sans faire d'hypothèses sur les modules qui seront utilisés pour réaliser l'assemblage. A partir de la représentation -du squelette- qu'il a choisi, le concepteur de modèles va alors pouvoir choisir, pour chaque processus, le module qui répond le mieux à ses critères.

A ce stade et pour la suite du projet, les caractéristiques de la plateforme sont prévues pour être non-limitantes par rapport aux représentations stochastiques ou pour l'assimilation. Si elles ne sont pas explicitement décrites, elles sont toujours dans le champs des possibilités que la plateforme veut offrir.

3.3.4 Cas d'utilisation 4 : développement de modules

Pour que la plateforme puisse aider au développement de modules, l'utilisateur doit d'abord choisir quel processus son nouveau module doit représenter. La logique qui impose qu'un module soit une fonction informatique permet donc à la plateforme de proposer à l'utilisateur un prototype -c'est à dire un entête de fonction informatique- avec les entrées et les sorties liées au processus choisi.

32 C'est ce qui est proposé dans la section 3.1.2 , page 14.

4 Caractéristiques de la plateforme

A partir des besoins exprimés -via le cahier des charges fonctionnel et les apports des comptes-rendus de réunion avec les scientifiques du groupe projet-, ce chapitre explicite les caractéristiques et fonctionnalités principales de la plateforme « sol virtuel ». Il reprend, reclasse et reformule les besoins exprimés et planifie leur réalisation, en prenant en compte les moyens mis à disposition (cf annexes, chapitre « forces en présence »). Il est utopique de penser que la totalité des besoins exprimés sont dans ce document. Et ce par delà même l'image d'ouverture que définit la plateforme. Ouverture qui indique des attentes pour l'intégration de futures évolutions prévues et non prévisibles à cet instant.

4.1 Squelettes puis modèles

Un des caractères les plus innovants de la plateforme -décrit dans le cahier des charges fonctionnel- est la notion de squelette. Le squelette est un ensemble ordonné -qui peut être vu comme un graphe orienté- de processus. Au premier abord, un squelette est l'assemblage de tous les processus d'une dimension donnée : 1D/2D/3D. S'il peut y avoir deux processus qui proposent des sorties identiques dans le squelette, alors on peut aussi introduire la notion de squelette avec ou sans conflits.

A titre d'illustration, supposons trois processus :

- P1 a pour sortie 'A'
- P2 a pour entrée 'A' et pour sortie 'B'
- P3 a pour entrées 'A' et 'B'

Il y a donc un ordonnancement a respecter entre ces processus. Cela donne un graphe ordonné qui représente le squelette possible (nommé ici S1) avec la construction suivante :

- P2 est le suivant de P1 (par 'A')
- P3 est le suivant de P1 (par 'A') et de P2 (par 'B')

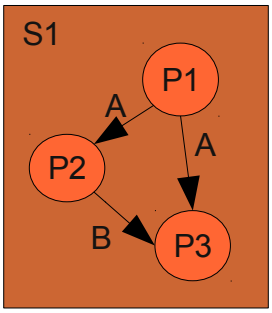

Illustration 1: exemple de squelette simple

Le squelette est déconnecté du modèle -c'est à dire d'un ensemble de modules- dans le sens où il ne dépend d'aucune représentation informatique, d'aucune technique de résolution numérique, ni même d'un choix de conceptualisation ou de modélisation particulier. A partir de là, on peut distinguer trois grandes étapes dans la plateforme « sol virtuel » :

1. La représentation abstraite du fonctionnement sol. Elle consiste à définir des processus, puis fabriquer des squelettes à partir de l'assemblage ordonnée et cohérent des processus.
2. La représentation concrète -informatique- du sol. Elle consiste -à partir d'un squelette donné- :
   - d'abord à choisir les modules qui vont représenter chacun des processus du squelette ;
   - puis de générer un modèle pour une plateforme informatique cible.
3. L'utilisation du modèle. Elle consiste à fournir les paramètres et les entrées nécessaires à l'utilisation du modèle choisi, enfin d'obtenir les sorties désirées.

Chacune de ces étapes successives peut être vue comme une construction quasi-disjointe des autres. Ceci permet d'obtenir trois réalisations logicielles indépendantes lesquelles seront exploitées par des compétences bien identifiées. Elles se baseront sur un ou plusieurs des quatre concepts : processus, squelettes, modules et modèles. Le graphique ci-dessous illustre ces trois grandes composantes.

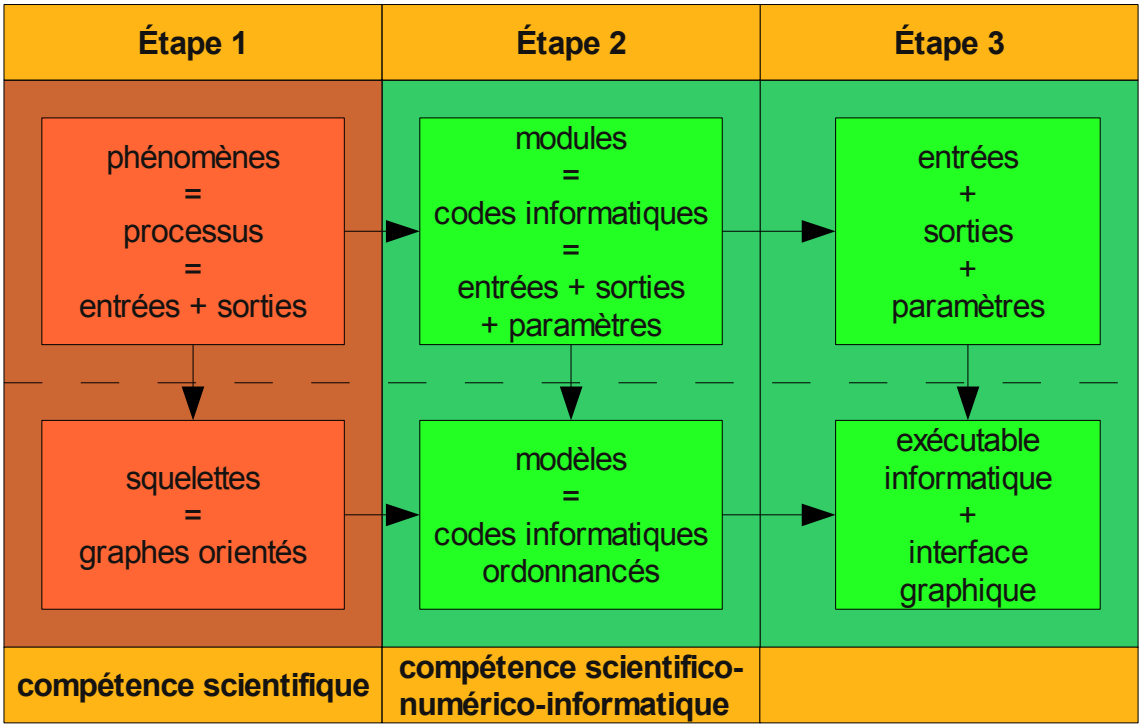

Illustration 2: les trois étapes de la plateforme

4.2 Les contrats de processus et de module

Un modèle peut-être considéré comme une « vue de l'esprit » scientifico-mathématico-algorithmique représentant des phénomènes et leurs relations. Le squelette -qui est la représentation hors informatique d'un modèle- doit donc être un assemblage qui respecte des règles. Un ordonnancement quelconque, sans contrôle de cohérence, entrainerait des anomalies de fonctionnement et/ou de résultats de sorties pouvant être difficiles à identifier...

Imaginer à l'avance tous les squelettes possibles -avec tous les processus possibles et leurs interactions- supposerait une certaine omniscience scientifique dans tous les domaines que couvre la plateforme. Du point de vue des modules, cela reviendrait à décrire -pour chaque processus- toutes les entrées et sorties possibles : irréalisable.

Il faut donc un système qui puisse à la fois proposer des squelettes et des modèles cohérents -donc à la fois du point de vue des processus et des modules -, tout en garantissant une certaine liberté quant à des évolutions non prévisibles. Concrètement, lorsqu'il existera des nouveaux processus ou de nouveaux modules à insérer dans la plateforme, ceux-ci ne manqueront pas de nécessiter de nouvelles entrées et de fournir de nouvelles sorties.

Nous pouvons pour cela imposer à chaque processus qui rentre dans la plateforme un contrat. Ce contrat lie le processus à la plateforme dans les termes suivants :

- tout processus a un nom (P), des entrées (E_P) et des sorties (S_P) ;
- le nom d'un processus est unique ;
- toutes les entrées déclarées par le processus sont fournies par la plateforme, que ce soit des sorties provenant d'autres processus ou des données externes au squelette (exemples : constantes, fichiers) ;
- les sorties d'un processus ne lui sont pas exclusives ;
- s'il est employé dans un squelette, les sorties d'un processus sont exclusives[33].

Chaque module rentrant dans la plateforme a aussi un contrat. Ce contrat lie le module d'un processus à la plateforme dans les termes suivants :

- tout module a un nom, des entrées et des sorties ;
- le nom d'un module est unique ;
- un module est rattaché à un seul processus ;
- le module doit fournir au moins une des sorties déclarées par son processus, c'est à dire un sous-ensemble de S_P ;
- le module peut vouloir utiliser n'importe quelles entrées déclarées par son processus, c'est à dire un sous-ensemble de E_P.

Pour un processus, on parle alors d'entrées maximales et de sorties possibles. Pour un module, on parle d'entrées nécessaires et de sorties obligatoires. Si un processus déclare plusieurs sorties possibles alors selon le choix du module qui le représentera, les modules en aval pourront être différents.

Le diagramme ci-dessous illustre :

- Sur la partie gauche, un processus qui demande deux entrées ('a' et 'b') et peut fournir deux sorties ('A' et 'B').
- Sur la partie droite, deux modules représentant ce processus. Chacun utilise une des deux entrées du processus ('a' ou 'b') et fournie au moins une des deux sorties ('A' ou 'B') déclarées par ce même processus.

33 Contrairement à la section 3.2.3.1 qui précise que « les sortie d'un processus lui sont exclusives ». Cette affirmation reste valable, mais uniquement hors du contexte des squelettes.

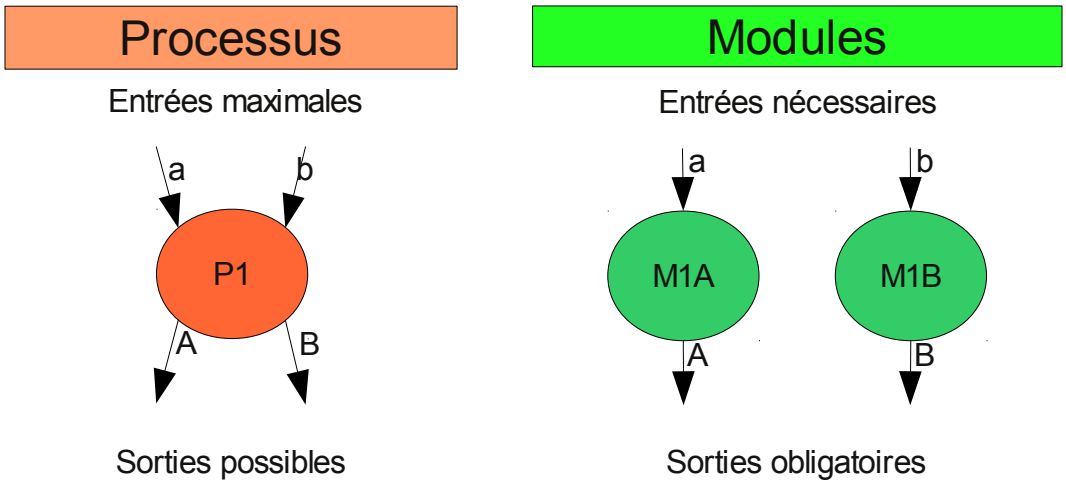

Illustration 3: exemple de contrat de processus/module

Une description des entrées maximales et sorties possibles des processus actuellement identifiés se trouve dans l'annexe de ce document. Les entrées nécessaires et sorties obligatoire des modules seront élaborés au fur et à mesure de leur insertion dans la plateforme.

Avec les contrats des processus, nous pouvons donc définir le squelette comme le sur-ensemble de tous les processus possibles, sans conflits (c'est à dire des sorties identiques). Les squelettes possibles pourront alors être générés automatiquement, à partir -des entrées/sorties- des processus.

4.3 Les piscines

Pour représenter les quatre concepts du cahier des charges fonctionnel -processus, squelettes, modules et modèles- la plateforme « sol virtuel » fournira quatre dépôts. Ceci afin de permettre la fabrication, le stockage et la validation de ces quatre types d'éléments. Au lieu de parler de dépôt, nous préfèrerons le terme piscine qui fait plutôt référence à une mise en commun, un groupement d'objets qui fournissent des services ou remplissent des tâches identiques[34].

Ces piscines n'ont pas vocation à être éditées « à la main ». Il sera cependant utile que leur format permette un accès direct « human readable », c'est à dire du texte. ». Ce format a -de surcroit- l'intérêt d'être plus facilement portable -entre différents systèmes informatiques- que l'autre type de format possible : binaire.

Ces piscines vont contenir la richesse scientifique et informatique de « sol virtuel ». Chaque acteur amenant ses connaissances. En fonction du mode de fonctionnement adopté, le partage de cette richesse sera différent, voire difficile. Il y a deux cas qui s'opposent :

- Cas de la plateforme fonctionnant en complète autonomie sur un poste de travail. L'accès aux piscines des autres n'est pas nativement possible.
- Cas de la plateforme avec des dépôts hébergés sur un serveur et accessibles par le réseau. L'accès est nativement possible à toutes les piscines.

34 Les termes entrepôt ou magasin auraient étés plus appropriés du point de vue de cette définition. Le terme garde l'avantage de rappeler son équivalent anglais : « pool ».

Sans être incompatibles, ces deux solutions sont radicalement différentes. Il faut donc en choisir une. Mais à terme une solution hybride est envisageable. Un consensus possible consiste à diffuser une version « officielle », tout en laissant la liberté à chacun de faire évoluer ses propres définitions. Les modifications de chacun pourrons rentrer dans la nouvelles version officielle, si elles sont retenues commes pertinentes et utiles par le groupe projet de la plateforme.

4.3.1 Piscine de processus

Contenu	Tous les processus disponibles pour fabriquer des squelettes.
Accessibilité	En lecture aux scientifiques de plateforme, afin de leur permettre d'y sélectionner des processus, en vue d'un assemblage en squelette. En écriture, pour que : •les scientifiques de processus y déclarent les phénomènes dans leur domaine de compétence ; •les scientifiques de la plateforme valident les processus qui garantissent une cohérence d'ensemble.
Dépendance	Aucune

Tableau 1: piscine de processus

4.3.2 Piscine de squelettes

Contenu	Tous les squelettes possibles à partir des processus disponibles, en utilisant les contraintes des entrées/sorties de ces derniers.
Accessibilité	En lecture aux scientifiques de modèle, pour leur permettre de choisir un squelette adapté à leur besoin. En écriture aux scientifiques de plateforme, pour fabrication de squelettes.
Dépendance	La piscine de processus

Tableau 2: piscine de squelettes

4.3.3 Piscine de modules

Contenu	Tous les modules possibles. C'est à dire des représentations réalisables -informatiques- des processus.
Accessibilité	En lecture aux scientifiques de modèle, pour leur permettre de choisir les modules qui seront dans le squelette de leur choix. En écriture aux scientifiques de module, pour leur permettre d'y introduire leurs modules.
Dépendance	La piscine de processus

Tableau 3: piscine de modules

La piscine de modules est alimentée soit par des codes informatiques existant, soit par des formalismes -a priori mathématiques- mais dont aucune représentation informatique n'existe.

Dans le premier cas, il faut distinguer le code directement intégrable à la plateforme -sous réserve de qualité[35] suffisante du code- de celui qui nécessite un recodage -en cas de langages informatiques incompatibles par exemple. Le recodage n'est pas la meilleure des solutions, car on repart du code qui est une représentation contextualisée de formalismes. Il est plus judicieux de repartir des formalismes mathématiques, lorsqu'ils sont suffisamment définis et lisibles[36]. Cela signifie que les équations publiées soient clairement organisée -idéalement sous forme d'algorithmes- et que les paramètres soient définis.

Dans tous les cas, l'accent devra être mis sur la documentation, autant scientifique qu'informatique, et plus particulièrement sur les informations liées aux entrées et sorties des modules.

L'alimentation de modules se fera sans doute le plus fréquemment -au moins les premiers temps- à partir de modèles d'unités qu'il faudra rendre modulaires. Dans ce cas, ce sont les scientifiques possesseurs de tels modèles qui devront participer à la plateforme. Potentiellement aidés d'informaticiens de module, ils pourront insérer leurs modules -provenant de leur modèles d'unités- qui réalisent les processus de la plateforme.

4.3.4 Piscine de modèles

Contenu	Des modèles qui réalisent les squelettes possibles.
Accessibilité	En lecture aux scientifiques utilisateurs de modèle, pour leur permettre de choisir les modèles les plus adaptés à leur besoins. En écriture aux scientifiques de modèle, pour la création de modèles à partir d'un squelette et de modules.
Dépendance	La piscines de squelettes et la piscine de modules

Tableau 4: piscine de modèles

4.3.5 Résumé des dépendances entre piscines

A titre d'illustration, le graphique ci-dessous reprend les quatre piscine de la plateforme. Les flèches représentent des dépendances. Les parties en rouge l'étape 1, celles en vert l'étape 2. Le contenu des piscines est donné à titre d'exemple et n'a qu'un but pédagogique.

35 Terme très subjectif. En gros, cela définit la difficulté -ou non- à comprendre le code pour un tiers, qui n'a pas participé à sa création. Rentrent donc en jeu des besoins de documentation et de modularisation du code.

36 La modularisation de gros modèle monolithiques est souvent éclairant sur le sujet. Il nécessite généralement un effort d'ingénierie informatiques très lourd. C'est malheureusement la seule solution lorsque les formalismes nécessaires à son codage « de zéro » sont insuffisants.

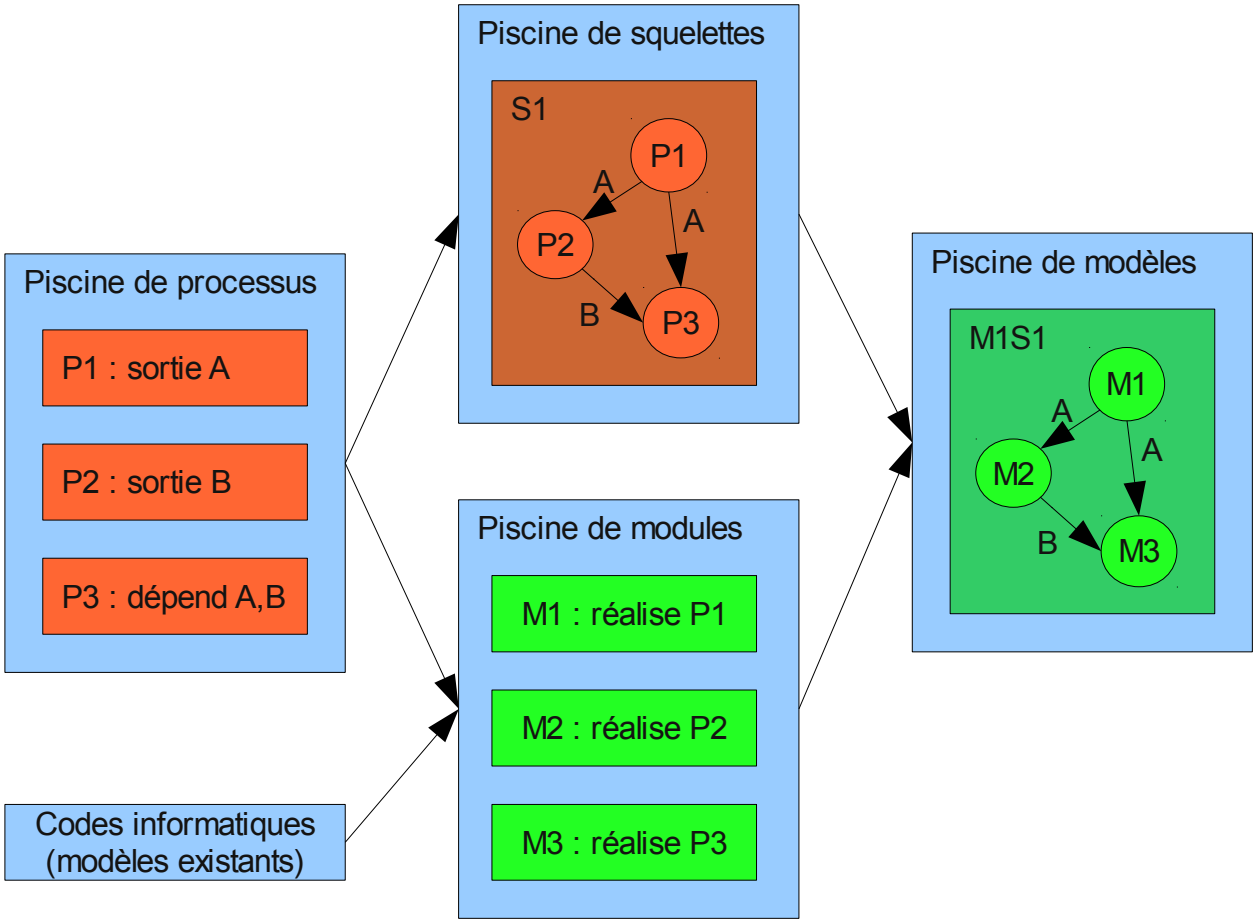

Illustration 4: dépendances entre piscines

4.4 Ateliers informatiques

Pour alimenter et gérer ces piscines, une représentation informatique et des outils informatiques sont nécessaires. Certaines de ces parties ne seront pas visibles aux utilisateurs « finaux ». Il n'est donc pas indispensable de proposer une carrosserie évoluée et séduisante dès les premières versions pour l'ensemble de ces ateliers.

La conception de ces ateliers représente une charge en travail informatique importante. Il sera donc précisé -pour chaque atelier- ses fonctions, son niveau d'utilisation, les pré-requis pour son utilisation et sa forme informatique -à court et moyen terme.

4.4.1 Atelier des processus

Fonctions	Déclaration des processus et de toutes les informations qui s'y rattachent. Créer/modifier/supprimer des processus
Niveau d'utilisation	Uniquement lors de la création ou de la mise à jour d'un processus. Utilisation importante au démarrage, plus sporadique par la suite.
Pré-requis	Aucun
Forme à court terme	Description des processus, formalisées dans des fichiers. Création et modification via des éditeurs. Outils informatiques annexes capables de détecter les erreurs et de valider les fichiers concernés.
Forme à moyen terme	Interface graphique qui permet de créer un processus.

Tableau 5: atelier des processus

4.4.2 Atelier des squelettes

Fonctions	Propose l'ensemble des squelettes -pour la dimension souhaitée- à partir des entrées/sorties de chaque processus issue de la piscine de processus.
Niveau d'utilisation	Vérifier et valider la connexion des processus entre eux dans les différents squelettes possibles.
Pré-requis	Plusieurs processus déclarés dans la dimension souhaitée.
Forme à court terme	Dans les premiers temps, le nombre de combinaisons possibles sera limité donc une description dans un fichier accompagné de simples outils de contrôle syntaxique permettront de valider les squelettes.
Forme à moyen terme	Interface graphique qui permet une vue graphique des combinaisons possibles pour une dimension donnée.

Tableau 6: atelier des squelettes

4.4.3 Atelier des modules

Fonctions	Intégrer facilement les modules à la plateforme.
Niveau d'utilisation	Pour les scientifiques concepteur de modules.
Pré-requis	Les processus que veulent représenter les modules existent dans la piscine de processus.
Forme à court terme	Interface graphique permettant l'insertion de modules sous forme de code informatique compatible.
Forme à moyen terme	Insertion d'un module sous forme de formalismes de plus haut niveau. C'est à dire indépendant de toute représentation, de tout code informatique.

Tableau 7: atelier des modules

Cet atelier va permettre à chaque scientifique concepteur de module de rajouter son travail dans la plateforme, il représente donc une partie essentielle et critique. L'ergonomie et la facilité

d'utilisation sont des points importants.

4.4.4 Atelier des modèles

Fonctions	Mise à disposition des modèles pour leur utilisation. Quatre actions successives sont identifiées : 1.La sélection. En fonction des besoins définis par l'utilisateur, proposer le modèle disponible le plus adapté. L'utilisateur se sert de ses connaissances sur les processus et les modules qui l'intéressent. Si aucun modèle satisfaisant n'existe, l'utilisateur pourra générer un nouveau modèle. Dans ce cas, il devra choisir un squelette et des modules. 2.La configuration. C'est à dire fournir les paramètres demandés par tous les modules du modèle sélectionné, ainsi que les entrées « externes » (exemple : les données climatiques). 3.L'exécution. C'est à dire gérer les différentes erreurs possibles, notamment celles issues des modules. 4.La visualisation. Uniquement si l'exécution s'est « correctement » réalisée. Diverses formes de sorties sont imaginables (exemples : fichier texte, graphiques, source pour un langage aval).
Niveau d'utilisation	les scientifiques utilisateurs et créateurs de modèles
Pré-requis	Modules et squelettes définis et disponibles.
Forme à court terme	Interface graphique uniquement pour la première et la troisième action. Action 2 : paramétrage par édition de fichiers. Action 4 : visualisation par outils annexes.
Forme à moyen terme	Interface graphique pour l'action 2.

Tableau 8: atelier des modèles

Cet outil est le résultat finalisé et appliqué des concepts précédents. Son utilisation sera donc un indicateur de succès -ou d'échec- de la plateforme. Comme l'outil précédent il représente une partie essentielle de la plateforme. L'ergonomie et la facilité d'utilisation sont des points majeurs de cet outil.

4.5 Résumé des rôles et acteurs sur les piscines

En résumé :

- La plateforme permettra de récupérer des modèles prêts à l'emploi. Ces modèles pourront avoir été développés au sein de la plateforme et seront disponible dans la piscine de modèles.
- La plateforme permettra de récupérer des modules pour un usage externe.
- La plateforme permettra de créer des modèles à partir de squelettes (squelette = modélisation des interactions entre des processus) à choisir dans la piscine de squelettes, et de modules qui offriront des représentations informatiques de ces processus (modules pris dans la piscine de modules). Ceci sera possible grâce à un atelier de création de modèles.
- L'utilisateur pourra alimenter sa piscine de modules (atelier création de modules) avec ses propres modules -moyennant le respect de certaines règles- et les utiliser pour faire des modèles.

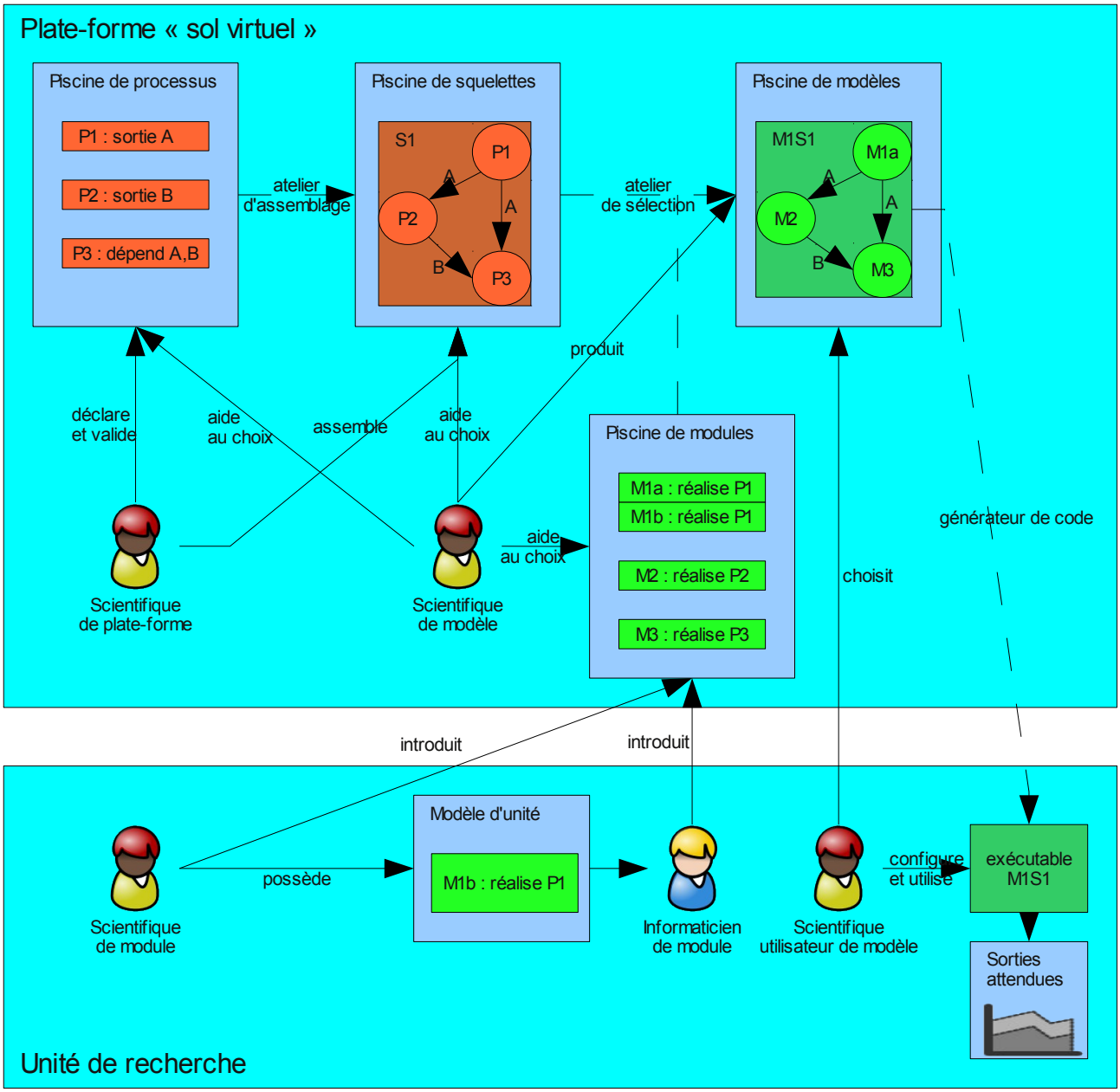

Illustration 5: rôles et acteurs sur les piscines

4.6 Analyse à l'aide du langage de modélisation UML

UML est un langage standardisé par l'OMG (« Object Management Group »). C'est un langage graphique de modélisation des données et des traitements.

Longtemps resté dans le domaine de la conception informatique, il est pourtant bien adapté pour analyser et concevoir tout type de système informatique. Notamment parce que le modèle réalisé constitue un outil majeur de communication entre les différents intervenants du projet dès les

activités d'expression des besoins et de spécification. La démarche d'analyse et de modélisation des exigences se veut itérative et incrémentale, pilotée par les cas d'utilisation. Elle couvre à la fois l'analyse statique et dynamique d'un système informatique. UML peut donc assurer un continuum : de l'analyse générale, le présent document, jusqu'à la génération de code.

Dans cette phase d'analyse informatique des besoins, les diagrammes UML utilisés délivrent volontairement une vue générale et incomplète. L'objectif est de privilégier la confrontation, la comparaison entre l'expression des besoins des scientifiques et leur interprétation, leur analyse par les informaticiens. Dénuée de détails, la discussion reste focalisée sur l'essentiel. Le détail peut venir dans une phase d'analyse détaillée -si le besoin s'en fait sentir- ou dans l'étape de validation du document de conception informatique. C'est donc ici principalement un support de discussion, pas une réalisation définitive.

4.6.1 Diagramme de classes

Dans UML, le diagramme de classes est un schéma conceptuel pour présenter les classes[37] et les différentes relations entre celles-ci. Ce diagramme fait partie de la partie statique d'UML car il fait abstraction des aspects temporels et dynamiques. Premier pas dans l'analyse informatique, son intérêt reste limité s'il reste seul.

L'illustration ci-dessous est une interprétation graphique des définitions données dans le cahier des charges fonctionnel. On y distingue notamment le découpage en deux paquetages : scientifique d'un côté et informatique de l'autre.

37 En programmation informatique, la classe est un des concepts de base de la programmation orientée objet. On appelle classe un ensemble d'objets partageant des propriétés : attributs (état) et comportement.

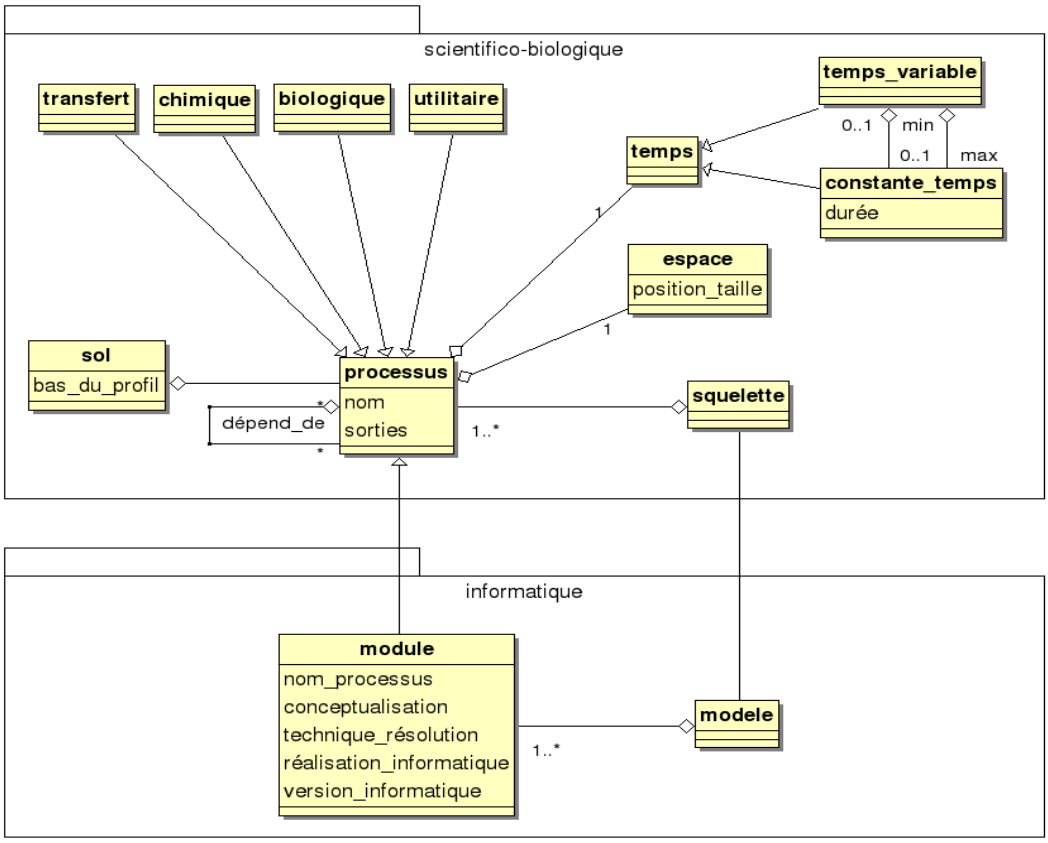

Illustration 6: diagramme de classes

4.6.2 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation correspond à une « vue » par les acteurs du système. Il correspond aux besoins attendus par chaque acteur : c'est le QUOI et le QUI.

L'illustration ci-dessous utilise la définition des acteurs telles qu'ils sont décrits en 3.2. Chacun d'eux est impliqué dans un ou plusieurs cas d'utilisation, soit comme acteur principal soit comme acteur secondaire. Certains cas demandent à être précédés par d'autres pour pouvoir se réaliser.

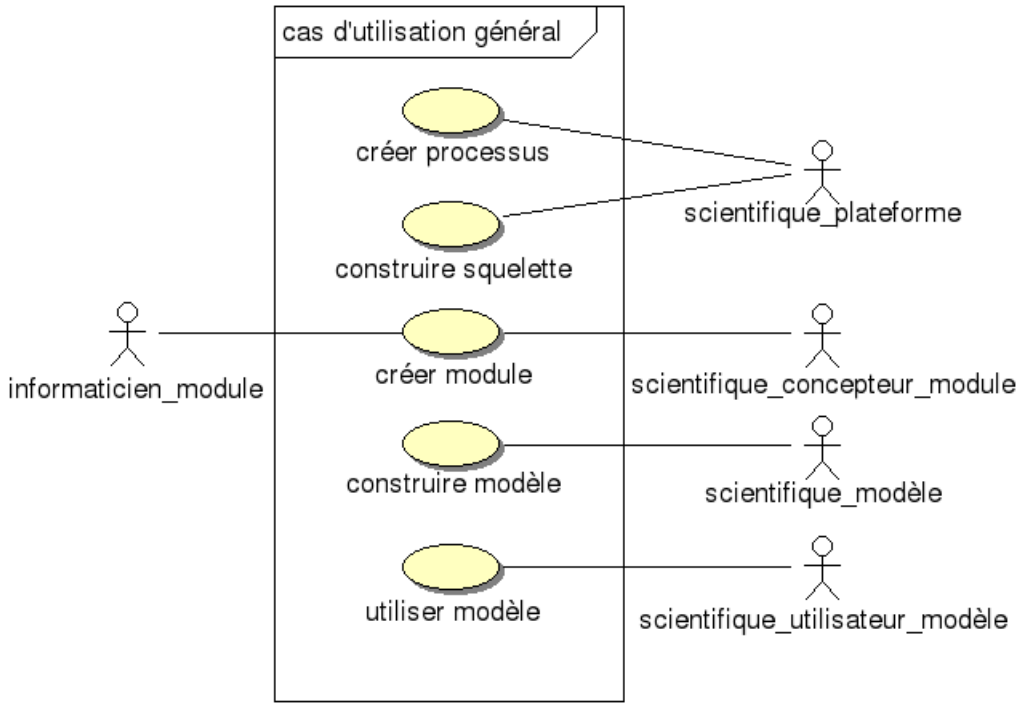

Illustration 7: diagramme de cas d'utilisation

4.6.3 Cas d'utilisation

Les cinq cas d'utilisation de la section précédente couvrent les besoins des différents acteurs. Tout en s'inscrivant dans un continuum logique d'actions, ils décomposent les différentes grandes tâches de la plateforme en trois grandes phases :

1. conception de processus et squelettes
2. conception de modules et modèles
3. utilisation de modèles

Ainsi segmentée, l'activité de la plateforme met en exergue trois types d'activités quasi-indépendantes :

- Les scientifiques en charge des processus et squelettes peuvent réaliser des représentations du sol sans supposer de contraintes techniques, notamment celles liées aux modules : méthodes de résolutions numériques ou codes informatiques.
- Les scientifiques en possession de moyens de réalisation techniques de modules sont guidés dans leur tâche par les contraintes imposées par les définitions des processus. Les scientifiques qui possèdent des compétences suffisantes pour choisir certains modules plutôt que d'autres peuvent assembler des modèles pour en faire des assemblages informatiques intégrés.
- Les utilisateurs « finaux » utilisent les modèles disponibles, dont la cohérence est assurée dans la phase précédente. Ils fournissent les paramètres, les entrées et déclarent les sorties qui les intéressent.

4.7 La communication

Pour éviter la démobilisation, il faut des objectifs forts -à court terme- qui restent inscrits dans des objectifs stables -à long terme. Il y a donc nécessité d'une réalisation rapide, montrable dès la première année de fonctionnement. Des choix « réalisables » en terme à la fois scientifiques, techniques et d'organisationnels doivent donc être décidés. Dans cette optique, les points suivant sont des priorités :

- Donner priorité aux squelettes 1D. Les phénomènes dans cette dimension sont scientifiquement les plus aboutis, même si quelques processus manquent. L'objectif est de se focaliser sur la technique informatique et montrer la faisabilité du scénario entrepris (en particulier du système de couplage).
- Privilégier les ressources locales, facilement disponibles et mobilisables.
- Continuer le travail scientifique en amont. En particulier la formalisation -avec les autres acteurs- de la représentation des processus et des squelettes 2D et 3D.

La mobilisation passe par la communication. Le moyen technique utilisé est un portail Web. Ce point d'entrée permettra de communiquer auprès des communautés -à la fois scientifiques et informatiques- désireuses de s'informer et de participer au projet. Elles y trouveront en particulier les documentations scientifiques et techniques, un point sur l'avancement du projet, des annuaires sur les acteurs et les modèles.

5 Besoins non fonctionnels

Les besoins -exprimés à la fois dans le cahier des charges fonctionnel et dans les comptes-rendus des réunions avec les scientifiques du groupe projet- qui expriment des attentes non fonctionnelles seront analysés dans ce chapitre pour déterminer s'ils sont contraignant ou pas.

Ce chapitre fait volontairement l'impasse sur la question -non fonctionnelle- du choix des licences logicielle. Une long chapitre y est consacré dans les annexes au présent document d'analyse.

5.1 Accessibilité

L'idéal serait une plateforme -ateliers comme modèles- qui permette à la fois un usage centralisé -pour calculs sur un serveur- et un usage localisé -pour utiliser un modèle de manière autonome (cf. cahier des charges fonctionnel, chapitre 4). Concilier ces deux modes de fonctionnement est réalisable mais couteux à mettre en œuvre. Un choix -au moins pour le court terme- est nécessaire. De fait, arbitrer une priorité c'est -au moins temporairement- rendre l'autre besoin non opérationnel. De plus, les choix techniques qui en découlent peuvent avoir une grosse influence sur des fonctions de la plateforme. En effet, certaines possibilités offertes pour un usage « en local » ne sont pas les mêmes que celles en mode « à distance ».

Sur ce point, le groupe projet souhaite privilégier le choix d'une application autonome -sur le poste de travail de l'utilisateur- avec à terme la possibilité d'effectuer facilement des mises-à-jour de modules.

5.2 Autonomie souhaitée

La notion de squelette permet d'offrir un cadre assez sécurisé pour réaliser l'insertion d'un module et de garantir une cohérence structurelle de la plateforme pour son inclusion. Par contre l'insertion de modules -pour un informaticien de module qui sera assez souvent un scientifique- est une opération pouvant avoir un niveau d'autonomie très varié et variable. Le niveau d'autonomie souhaité influe sur la stabilité de la plateforme, son utilisation et aussi sur la forme finale de celle-ci. Trop d'autonomie sur la forme de modules à intégrer engendre des développements au niveau infrastructure très important. A l'inverse si tout est intégré par l'équipe plateforme, cela ferme la plateforme et rend les scientifiques utilisateurs dépendant de cette équipe.

Dans le cadre de l'insertion de modules dans la plateforme, le groupe projet souhaite mettre la priorité pour rendre les scientifiques autonomes. Pour répondre à cette nécessité, la plateforme ne doit pas imposer l'apprentissage d'un nouveau langage informatique d'implémentation[38] pour les modules. Il faut partir des langages les plus utilisés dans la communauté scientifique qu'intéresse la plateforme. A court et moyen terme, la liste des langages énoncés dans le cahier des charges fonctionnels pour le codage des modules est donc réaffirmé : C, C++, Fortran. A plus long terme, des langages complémentaires pourrons nativement -c'est à dire sans couches logicielles intermédiaires- intégrer la plateforme, notamment le langage « R ».

5.3 Intégration de modules

Le chapitre 4 évoque un accès restreint pour les modules en phase développement. Cela impose-t'il que les formalismes scientifiques -dont est issu le module- doivent être publiés avant la

38 comme la plateforme CAPSIS

rentrée du module dans la plateforme ? En tous cas cela impose au moins une sorte de contrôle sur les modules mis à la disposition de la communauté et bien entendu l'accord du ou des concepteurs.

Le besoin de l'intégration de modules non INRA implique sur la structure de la plateforme des contraintes importantes. Notamment en terme de licences logicielles. Ces modules doivent-ils être des outils « à part » (par exemple des exécutables disjoints du reste de la plateforme) ou des modules comme les autres ?

Le choix du groupe projet est de considérer ces outils comme « à part » de la plateforme. Leur usage est donc en dehors des ateliers de la plateforme. Ils seront référencés sur le site Web du projet mais ne pourrons participer aux couplages comme les autres modules intégrés. C'est à dire que les questions de couplage sont à la charge de l'utilisateur, a priori à partir du code source des modèles produits par la plateforme. L'intégration de ce type de modules -jugés importants pour la plateforme- pourra s'effectuer dans le cadre d'un accord spécifique avec les concepteurs-propriétaires du code original.

5.4 Systèmes d'exploitation supportés

Les aspects multiplateformes -en tant que système d'exploitation- demandent à être clairement validés (cf. cahier des charges fonctionnel, chapitre 4). Notamment parce que -d'un point de vue informatique- une bonne compatibilité pour plusieurs systèmes rend les tests et le support technique beaucoup plus lourd et complexe. De plus, le choix de librairies et d'outils est limité à ceux disponibles pour les plateformes cibles. Les modules déposés devront être testés et fonctionnels pour chaque plateforme. Une attitude prudente consiste donc à délimiter les systèmes ciblés aux plus utilisés.

Dans un premier temps, le groupe projet souhaite se focaliser sur les systèmes qui semblent les plus utilisés par les scientifiques à l'INRA : Windows et Linux. Se pose alors la question des versions de ces systèmes que la plateforme doit supporter. La politique de la société Microsoft vis à vis de ces systèmes d'exploitations est relativement claire : cohabitent en permanence un système « courant » -actuellement Windows Vista »- et un système ancien -actuellement Windows XP ». Du coté de Linux, quasiment toutes les distributions se valent en terme de compatibilité et d'offre logicielle. Toutefois, il est raisonnable de ne garder que celle actuellement encouragée par la DISI -Debian- et ses grandes variantes connues -comme Ubuntu.

A noter qu'une solution « client-serveur » est plus adaptée et plus simple à supporter qu'une version « tout sur le poste de l'utilisateur ». En effet, une partie de la plateforme est sur le(s) serveur(s)[39] de la plateforme, partie mieux maitrisable -en terme de système d'exploitation- par l'équipe projet.

5.5 Portée linguistique de la plateforme

Un besoin non fonctionnel -qui n'est au départ pas d'ordre technique- est la volonté de s'ouvrir à l'international. Ce souhait implique donc :

1. d'écrire les codes informatiques et l'ensemble des documentations en langue anglaise ;
2. de choisir des outils et librairies -pour construire la plateforme- documentés en langue anglaise ;

39 Exemple : le modèle Alexis (UR PSH) a sa partie modèle sur un serveur « de calcul » Linux, alors que son interface graphique est exécutée dans le navigateur Web de l'utilisateur, donc quel que soit son système d'exploitation.

3. de préférer un type de licence à portée internationale ;

5.6 Accès à des bases de données

L'accès à des bases de données pose en premier chef les questions de droits d'accès et de licences concédées, en fonction des utilisateurs. C'est une thématique qui dépasse largement celle de cette plateforme. Des intérêts et des rapprochements avec les structures en charge de ces questions -et donc des solutions pour « sol virtuel »- sont inévitables. Le CATI ISIE du département EA est un candidat à privilégier.

Le groupe projet souhaite que les données externes soient à la charge des utilisateurs. La plateforme peut donner des liens vers les données mais ne le héberge pas. La plateforme peut toutefois héberger des données libres, à des fins de tests notamment.

5.7 Cohabitations difficiles

De manière générale, les informaticiens n'aiment pas le préfixe « multi » : multilangage, multicompilateur, multiplateforme[40]... Répondre à ce critère coute la plupart du temps cher en développement informatique. Notamment parce qu'il impose de faire cohabiter des choix souvent peu compatibles. Lorsque c'est possible, il est donc intéressant de réduire la « multitude ». En tout état de cause, il faut savoir mettre des priorités pour satisfaire d'abord le plus grand nombre, le plus important. L'objectif n'est pas exclure le reste, mais simplement de lui donner un temps proportionnel à l'intérêt qu'il suscite.

5.8 Document de conception informatique

Le choix de solutions informatiques - outils de développement, langages - devra être basé sur une analyse du contexte et fondés sur des arguments informatiques mesurables et pondérés : pérennité, interopérabilité, réutilisation, diffusion, etc. C'est donc dans le document de conception informatique -qui est la suite logique du présent document d'analyse- que seront développées les questions de cet ordre.

40 En informatique, se dit de logiciels conçus pour fonctionner sur plusieurs couples ordinateur/système d'exploitation

6 Les autres plateformes

D'autres plateformes scientifiques existent, notamment à l'INRA et dans le département EA. A ce stade de réflexion, la comparaison entre la plateforme « sol virtuel » et d'autres sera réalisée uniquement par rapports aux attentes scientifiques. Les questions portent sur la réutilisation et l'interopérabilité avec les autres plateformes. L'opportunité de l'utilisation des technologies déjà utilisées dans celles-ci sera étudié dans le document de conception informatique.

De manière générale, quel que soit le coupleur utilisé, il faudra l'alimenter en briques élémentaires -processus- « désossées » de la boucle temporelle. C'est à dire que chaque processus fonctionne sur un pas de temps élémentaire, interagissant avec les autres processus sur ce même pas de temps. C'est au coupleur de gérer le temps et de synchroniser les processus entre-eux. C'est dans cet esprit que la plateforme « sol virtuel » est vue par le groupe projet comme une brique élémentaire utilisable par les autres plateformes, notamment celles du département EA : Record, « plante virtuelle » et « paysage Virtuel ». Notre plateforme fournirai alors un exécutable -ou du code source- qui serai un module pour une plateforme tierce.

Les plateformes étudiées sont : RECORD-DEVS-VLE, OpenAlea, SEVE-PALM, SEAMLESS-APES/ModCom[41], CAPSIS[42] et APSIM[43]. L'objectif est d'étudier l'opportunité d'utiliser leur acquis, leurs d'expériences et -à moyen terme- les composants logiciels. Ceci pour éviter de ré-inventer la roue, technologiquement et ou scientifiquement parlant. L'analyse de ces plateformes a été réalisé par rapport aux besoins et demandes scientifico-informatique de « sol virtuel ». L'analyse des démarches et solutions techniques adoptées sera effectuée dans l'étape suivante : la phase de conception.

6.1 Représentation scientifique versus représentation informatique

Le cahier des charges fonctionnel définit les concepts de processus et de squelettes, en les distinguant bien des modules et des modèles. Cette séparation entre la réflexion scientifique d'une part et la réalisation numérico-informatique d'autre part est une caractéristique forte de « sol virtuel ».

Les plateformes étudiées mettent en avant leur coté « plateforme générique de couplage ». Cette généricité permet -dans un contexte de complexité des formalismes équivalent- d'utiliser la plateforme quel que soit le sujet d'étude scientifique. Cependant une structure générique est conçue pour permettre de construire des applications d'un certain type. Ainsi cette structure facilite le développement d'une application qui entre bien dans le cadre considéré, mais perd son intérêt par rapport à une application dont elle ne couvrirait pas l'ensemble des exigences. Ce qui est difficile à juger tant que l'on n'y est pas confronté concrètement.

Les plateformes étudiées peuvent être appréhendées comme des plateformes techniques qui se focalisent sur les besoins de couplage et la solution technique apportée par l'outil informatique mis en œuvre. Elles sont, au moins au premier abord, non typées. C'est à dire qu'elles ne sont pas scientifiquement spécifiques. Hors, il est important pour « sol virtuel » de se focaliser sur les réflexions scientifiques en amont. Notamment celles sur la représentation scientifique du domaine, en dehors de tout contexte numérico-informatique.

Le projet « sol virtuel » se veut être d'abord une plateforme propice à la réflexion et la

41 System for Environmental and Agricultural Modelling Linking European Science and Society (http://www.seamless-ip.org/)
42 Computer-Aided Projection of Strategies In Silviculture (http://capsis.cirad.fr/)
43 Agricultural Production Systems sIMulator (http://www.apsim.info/apsim/)

représentation scientifique, avant le coté informatique. La composante numérico-informatique est importante mais elle est vue comme un moyen. A titre d'exemple, les squelettes définissent les couplages attendus et possibles. Il convient que le système de couplage -et plus largement les outils proposés par la plateforme- se serve de cela pour guider dans une représentation du sol spécifique et validée par les scientifiques qui y ont des responsabilités.

6.2 Atelier logiciel

Le cahier des charges fonctionnel définit le besoin d'un atelier logiciel (section 3.3). Celui-ci doit être un lieu qui permette de répondre aux objectifs suivants :

- de développements de modules ;
- de tests de modules dans un modèle :
- de construction des modèles à partir de squelettes donnés ;
- d'aide au choix des modules ;
- de contrôle des compatibilités.

Chacun de ces besoins n'est pas inédit en soit. Parmi les plateformes citées, certaines offrent d'ailleurs un ou plusieurs de ces services au travers d'une interface graphique, notamment PALM, OpenAlea et CAPSIS. Ces services sont appréhendés différemment par les plateformes et avec des niveaux de complexité différents. Hors, il est important pour « sol virtuel » de réaliser en priorité tous ces objectifs.

Le projet « sol virtuel » ne peut pas faire l'impasse sur ces ateliers qui donnent leur autonomie aux différents acteurs -notamment scientifiques- de la plateforme. Dès la première diffusion, il s'agira de proposer -en plus d'un système de couplage adapté- des productions logiciels graphiques viables et aux fonctionnalités suffisantes pour assurer les utilisations jugées les plus prioritaires. Les ateliers graphiques sont donc des pré-requis indispensables à une bonne adhésion de « sol virtuel » auprès de la communauté concernée, ce dès les premiers utilisateurs.

6.3 Formalismes scientifiques

Un modèle est une abstraction subjective d'une réalité physique. Avant de choisir une solution informatique pour le réaliser, c'est donc le choix d'un formalisme scientifico-mathématico-numérique : équations différentielles (du 1er ou du 2nd ordre), équations aux différences, évènements discrets, etc. De ce point de vue, la plupart des plateformes étudiées font des arbitrages radicaux qui laissent peu de choix au mélange des différents formalismes possibles. La seule plateforme qui permet potentiellement[44] des combinaisons -par nature difficilement conciliables- est « RECORD ». Ce cas isolé s'explique par le fait que cette plateforme devra -à terme- permettre le couplage de modèles réalisés dans ces différents formalismes.

En dehors de toute considérations sur la réutilisation et la capitalisation technique, « sol virtuel » n'a -pour l'instant- pas besoin de multiformalisme, comme pour la plateforme RECORD. En effet, les phénomènes évoqués -jusqu'à maintenant dans le cahier des charges fonctionnel- n'utilisent que des formalismes « simples », de type équations différentielles du 1er et du 2nd ordre. La résolution technique du couplage de ces équations -s'il n'est pas simple- n'est donc pas un problème particulièrement préoccupant en soit. Une solution de couplage simple serait alors un gage de simplicité de mise en œuvre comparé à l'effort d'adaptation des modules à un système de

44 Ce qui est actuellement en cours d'essai.

couplage -comme ceux des plateformes citées- qui lui est pressentit comme une tâche plus complexe et plus longue. Dans l'hypothèse où le besoin de multiformalisme se ferait sentir -au travers de modules innovants- un système de couplage plus adapté pourrait remplacer -avec peu d'efforts- le coupleur existant. Il faudrait alors revoir tous les modules existants mais une grosse partie du travail d'adaptation des modules à un système de couplage serait déjà réalisée.

6.4 Échelles de temps

La section 5 du cahier des charges fonctionnel donne comme priorité la gestion d'échelles de temps allant du phénomène bref -de l'ordre de la minute- à des temps très long -le siècle. Hors, certains modèles répertoriés fonctionnent sur des pas de temps très court : de l'ordre de la milliseconde pour certains phénomènes. Si la plateforme doit rendre des résultats sur des échelles longues, cela impose -au moins en attendant d'avoir des modules adaptés- que les lenteurs imposées par la technique de couplage soient négligeables par rapport au temps d'exécutions des modules. En d'autres termes, le temps de transfert des données entre les modules doit être négligeable par rapport à celui pris par les calculs propres -internes- des modules.

Dans la plateforme « sol virtuel », si l'on considère que les modules représentent des phénomènes élémentaires -c'est à dire bien circonscrits- leur temps de calcul sur un seul pas de temps devrait être potentiellement courts. Il n'est donc pas envisageable -pour avoir des temps de simulation totaux raisonnables- que la solution de couplage prennent du temps à traduire les données -les variables- entres les modules.

Ceci impose donc une contrainte pour que les modules utilisent des formalismes naturellement compatibles entre eux. Il y a alors une contrainte non fonctionnelle qui apparait : utiliser des codes informatiques compatibles, c'est à dire de la même famille binaire. Ces deux obligations pour « sol virtuel » ne sont pas un problème puisque :

1. les modules actuellement répertoriés possèdent des formalismes du même type ;
2. la contrainte non fonctionnelle sur les langages permet de compiler facilement ensemble les codes informatiques en C, C++ et Fortran.

Ce problème de performances est fortement amplifié par les échelles d'espace décrites dans le cahier des charges fonctionnel. Sans qu'il soit une priorité immédiate, ce besoin devra aussi être pris en compte.

7 Synthèse et perspectives

Globalement, la plateforme est un ensemble d'outils et de services permettant de répondre aux besoins de modélisation du sol et de susciter de nouvelles réflexions-modélisations. Les attentes exprimées par les différents acteurs sollicités sont des éléments décisifs dans la construction technique de la plateforme. Ces grandes ambitions s'inscriront inévitablement dans des évolutions des besoins qui peuvent remettre en cause certains choix techniques réalisés. D'où la nécessité de trouver un équilibre qui permette de répondre aux exigences du présent, sans hypothéquer l'avenir.

De plus, le besoin évident de forme informatique ne doit pas cacher les fondations scientifiques. La réflexion scientifique, même si elle doit s'inscrire dans un principe de réalité informatique, devra dès à présent essayer de rester la plus indépendante possible de la technique. L'élément -usine à fabriquer du code informatique- de la plateforme n'est qu'une réalisation technique, qui elle ne capitalise pas le savoir scientifique produit en amont. Le cahier des charges donne d'excellentes bases qui vont dans ce sens.

La réflexion informatique -qui démarre formellement avec le présent document- prolonge le cahier des charges en donnant des rôles -donc des acteurs- pour chaque grande fonction de la plateforme « sol virtuel ». Les ateliers -réalisation informatiques potentiellement disjointes- et surtout les piscines -si elles ne sont pas indépendantes- participent à ce fractionnement en trois étapes.

Dans la suite immédiate du projet, les documents de conception -général puis détaillé- seront une vision possible de l'analyse en termes informatiques concrets. En dehors des choix techniques proprement dits, ces documents seront l'occasion de préciser des besoins et une analyse qui ne peut être qu'incomplète à ce stade du projet. Contexte de recherche oblige. C'est pour cela que leur construction sera étayée par des logiciels informatiques. Ceux-ci serviront des tests mais seront bien réel. L'objectif de ces logiciels -potentiellement non pérennes- sera de valider la pertinence des choix techniques qui semblent adaptées au projet. Au delà de cet intérêt, ces logiciels serviront à confronter les concepts avec la réalité des besoins. Ces réalisations serviront donc autant du coté utilisateur (acteurs), que développeur informatique.

Concrètement, l'idée sera de construire deux logiciels avec interfaces graphiques. L'un pour valider les concepts liés aux processus et à leur assemblage en squelettes. Cela permettra aussi à la communauté concernée de réagir sur l'existant et de proposer de nouveaux processus (voir l'inventaire des processus dans les annexes). L'autre pour illustrer l'utilisation de modèles -assemblages de modules. Notamment le choix des paramètres et des entrées/sorties liées aux modules. Tout ceci de manière dynamique et complètement générique. Les interfaces graphiques étant générées "à la volée" et se construisent en fonction des modèles -et donc des modules- choisis.