Gestion de données et de connaissances appliquée aux bioprocédés

Alexandre Granier

CONSERVATOIRE NATIONAL

DES ARTS ET MÉTIERS

CENTRE RÉGIONAL ASSOCIÉ
DU LANGUEDOC-ROUSSILLON

MEMOIRE
présenté en vue d'obtenir
le DIPLÔME d'INGÉNIEUR CNAM

SPÉCIALITÉ : Informatique

OPTION : Informatique Réseau et Systèmes Multimédia

par

Alexandre Granier

Gestion de données et de connaissances appliquée aux bioprocédés

Soutenu le 6 juillet 2010

JURY
PRÉSIDENT: Mr Yves Lalloum
MEMBRES: Mme Thérèse Libourel, Mr Michel Sala, Mr Éric Latrille, Mr Patrice Buche, Mr Laurent Becsei

Table des matières

Remerciements

Je souhaite remercier :

Pascal Neveu qui m'a proposé le sujet et permis de le traiter dans d'excellentes conditions. Un grand merci pour son accueil et la confiance qu'il m'a accordée.

Thérèse Libourel qui m'a apporté un soutien appuyé durant la rédaction du mémoire.

Christian Picou pour sa bonne humeur et son aide concernant le procédé de fermentation alcoolique.

Stéphane Lapeyre qui a relu ce travail avec attention et dont les conseils m'ont été précieux.

Toute l'équipe du laboratoire MISTEA qui m'a accueilli avec gentillesse ces deux dernières années.

Introduction

L'industrie agroalimentaire fait face depuis quelques années à des changements importants en réponse aux nouvelles exigences des consommateurs en termes de qualité, de sécurité alimentaire et de diversification de la production. Du point de vue du consommateur, cette nouvelle donne est orientée par des motivations sociales ou éthiques, comme par exemple l'impact de la production sur l'environnement ou le caractère « durable » d'une méthode de production. À l'autre bout de la chaine, l'industrie agroalimentaire est en quête permanente de nouveaux marchés et a intégré dans sa stratégie l'importance d'apparaître soucieuse de l'impact environnemental de son activité.

Pour aborder ces nouveaux enjeux, il faut concevoir de nouvelles approches d'ingénierie des procédés agroalimentaires intégrant ces exigences. Il s'agira dans bien des cas de minimiser la consommation d'énergie ou d'eau mais également d'être en mesure d'assurer la traçabilité des **intrants**[1]. Cela peut également concerner l'amélioration des qualités gustatives ou nutritives des aliments.

Tout naturellement, les organismes de recherche en agronomie sont précurseurs dans ce domaine.

L'INRA en tant qu'acteur majeur de la recherche agronomique met en œuvre dans ses différents laboratoires des expérimentations sur les procédés agroalimentaires. Les grands objectifs de cet institut consistent à relever les défis que posent l'alimentation et l'environnement. Dans cette perspective, l'INRA a pris part au projet européen CAFE dont l'ambition est de proposer de nouveaux modèles pour aborder la problématique des procédés alimentaires.

Ce mémoire s'inscrit dans ce cadre général et fait suite à un stage dans l'unité mixte de recherche de **M**athématique, **I**nformatique et **ST**atistique pour l'**E**nvironnement et l'**A**gronomie (**MISTEA**). Le stage a consisté en un travail de réflexion, de modélisation et de mise en œuvre de solutions innovantes pour traiter les données scientifiques issues des expérimentations sur les procédés. Les résultats de ce travail ont notamment permis une gestion plus efficace des informations manipulées par les différents acteurs (opérateurs, experts, techniciens...) ainsi qu'une meilleure valorisation de la donnée expérimentale. Il dépasse donc le simple cadre des procédés agroalimentaires et apporte des solutions sur l'ensemble des bioprocédés, alimentaires ou non.

Quatre bioprocédés ont servi de cas d'étude pour la réalisation de ce mémoire :

1. un procédé de fermentation alcoolique étudié par l'UMR sciences pour l'œnologie de l'INRA,
2. un procédé de filtration de la bière mis en œuvre à l'université de Wageningen,

1. Les mots apparaissant en gras font l'objet d'une entrée dans le glossaire.

3. un procédé de dépollution par méthanisation expérimenté au laboratoire de biotechnologie de l'environnement de l'INRA à Narbonne,
4. un procédé de liophylisation étudié au LGMPA [2] de l'INRA de Grignon.

Une implantation des méthodes et techniques proposées a été réalisée sur le procédé de fermentation alcoolique et est en cours sur les procédés de liophylisation et de dépollution. La plupart des exemples de ce mémoire sont issus du travail sur le procédé de fermentation.

Dans une première partie, nous présenterons le contexte des procédés alimentaires en explicitant les différentes modalités sous lesquelles se présente l'information scientifique. Nous établirons alors la lettre de mission de travail. Dans une deuxième partie, nous présenterons l'état de l'art en la matière et nous détaillerons les différentes architectures de systèmes expérimentaux et de leurs modèles de données. Puis nous présenterons l'analyse du métier indépendamment de l'architecture. Après l'analyse nous présenterons les technologies utilisées et nous détaillerons la mise en œuvre technologique. Dans une sixième et dernière partie, nous en présenterons les résultats en nous appuyant sur l'implantation au laboratoire Sciences pour l'œnologie. Nous conclurons par une discussion sur les apports et les limites de la démarche adoptée.

2. Laboratoire de Génie et de Microbiologie des Procédés Alimentaires

Chapitre 1

Définitions, contexte des procédés de transformation alimentaire

1.1 Domaine à la croisée de l'informatique, de l'automatique et des agrosciences

Les **procédés agroalimentaires** désignent les moyens utilisés par l'industrie agroalimentaire pour transformer la matière première animale et végétale en aliment consommé. La recherche scientifique sur les procédés s'appuie souvent sur des méthodes issues de la tradition et essaie de les observer pour mieux les comprendre et les automatiser.

Exemple:
La vinification, la transformation du raisin en vin, est un procédé de transformation alimentaire comprenant un certain nombre de phases (qui peuvent différer selon le type des vins que l'on souhaite produire) :
- *le foulage, qui consiste à éclater les grains de raisin, sans pour cela écraser les pépins et la rafle, et qui permet d'obtenir le moût,*
- *l'ensemencement, qui consiste à introduire dans le moût des levures,*
- *la fermentation alcoolique, qui consiste en l'action des levures sur les sucres du moût,*
- *la fermentation malolactique au cours de laquelle l'acide malique du vin se transforme en acide lactique.*

La reproduction d'un procédé implique des équipements particuliers, notamment un certain nombre d'actionneurs pour agir sur le procédé et des capteurs pour réaliser les observations.

Actionneur Un actionneur permet d'assurer l'évolution du procédé dans le sens souhaité. *Il peut s'agir d'une vanne, d'une pompe, d'une lampe, etc.*

Capteur un capteur est un équipement permettant d'effectuer une mesure. La qualité de cette dernière dépend de la qualité du capteur. Les capteurs doivent être régulièrement étalonnés (on vérifie que l'information fournie par le capteur correspond bien à la mesure effectuée).

Le schéma 1.1 montre le fonctionnement général des procédés automatisés.

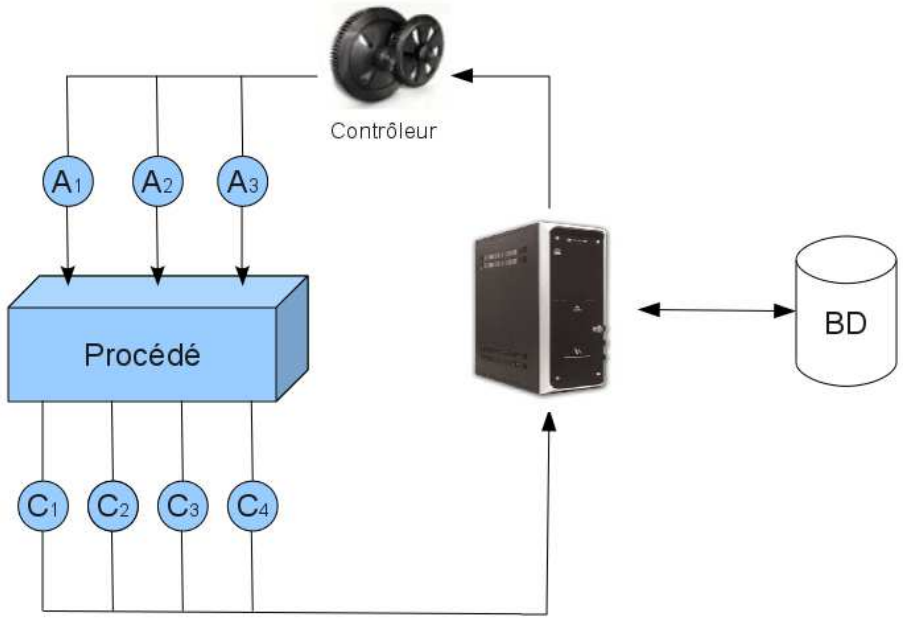


FIGURE 1.1 – Schéma général des procédés. Les C_i sont les capteurs, les A_i sont les actionneurs contrôlés par une loi de commande.

Les capteurs et actionneurs sont reliés à un appareil appelé Automate Programmable Industriel (API). L'opérateur communique avec l'API via l'ordinateur. Il est ainsi renseigné sur l'état du procédé et peut le faire évoluer en décidant de la manœuvre de certains actionneurs.

Lorsque c'est possible, on essaiera de remplacer l'action humaine par celle d'une machine, supposée plus homogène. Ce positionnement implique une démarche d'automatisation du procédé et de son observation.

1.1.1 La démarche d'automatisation

La démarche d'automatisation a pour vocation de substituer la décision de l'homme par un système, est plus riche que la simple mécanisation. L'interaction entre le procédé et l'aliment devient prépondérante. On peut souligner deux finalités[5] de l'automatique dans les procédés :

- l'automatique classique mobilise des outils, majoritairement informatiques, pour rétroagir sur le système et en corriger le fonctionnement conformément à des objectifs connus,
- l'automatique s'applique aussi comme une des composantes des systèmes d'information industriels, dont la finalité est d'aider l'opérateur dans ses tâches et notamment d'améliorer sa réactivité sur le procédé considéré. Il s'agit souvent ici de prendre en compte les situations non prévues (manque d'un capteur, etc.). C'est l'opérateur qui devra assurer les décisions.

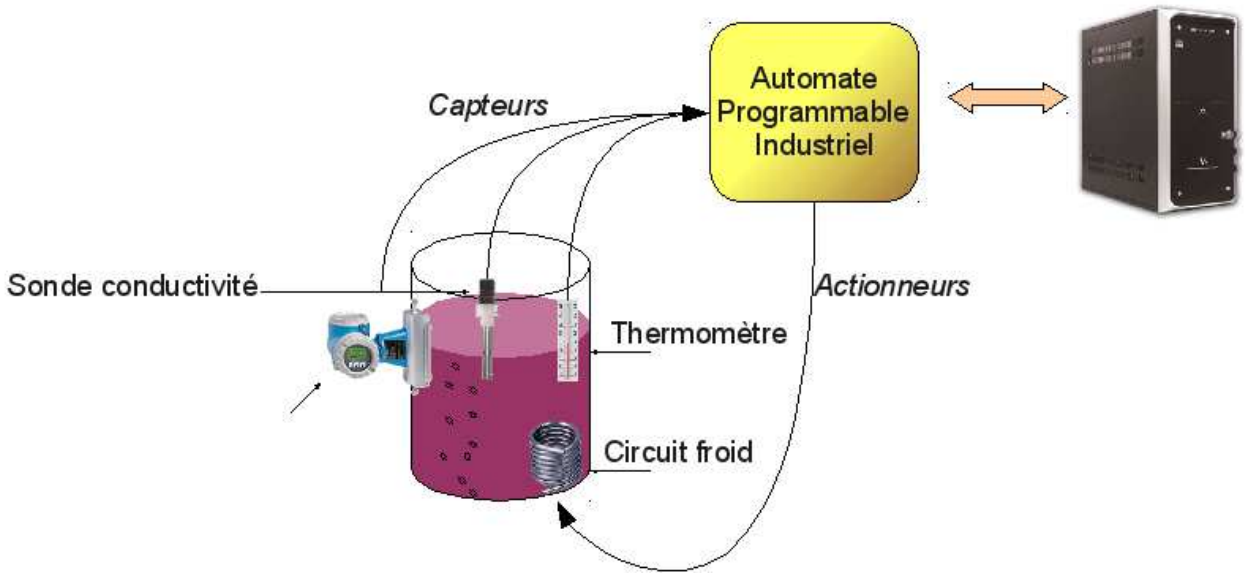


FIGURE 1.2 – Un exemple simple d'automatisation d'un procédé de fermentation alcoolique.

1.1.2 La régulation continue

La boucle de régulation fermée constitue la base de l'automatique continue. Elle est constituée d'un capteur-transmetteur qui mesure en permanence la grandeur réglée. Cette mesure est transmise à un régulateur. Le régulateur compare la mesure à la consigne (valeur optimale de fonctionnement de la grandeur réglée), décide de la meilleure correction à apporter au procédé grâce à une loi de commande et transmet cette correction à la vanne de régulation. La vanne de régulation exécute l'ordre donné par le régulateur, modifie ainsi le débit qu'elle contrôle (la grandeur réglante). Cette modification de l'état du procédé doit contribuer à ramener le plus efficacement possible la grandeur réglée à la valeur de consigne.

Exemple:
Dans notre exemple (figure 1.2), le thermomètre transmet la température du moût en fermentation au régulateur qui enclenchera éventuellement le circuit froid pour corriger un écart trop important avec la consigne.

1.1.3 La spécificité des industries agroalimentaires

Les procédés agroalimentaires, dans un cadre industriel ou de recherche, ont des caractéristiques particulières vis-à-vis de l'automatisation. Les opérations sont parfois instables (fermenteurs, par exemple), souvent non linéaires et caractérisées par beaucoup de grandeurs dont une petite partie est observable.
La nature des grandeurs qui caractérisent les opérations est variée : elle est souvent analogique, ou événementielle, parfois symbolique et le plus souvent multidimensionnelle.

Exemple:
Prenons l'exemple de la fabrication du pain. La phase de fermentation est discontinue du point de vue des flux de matière. La levure est mélangée à la farine et repose un certain temps dans une étuve à 25 °C pour assurer le levage, puis l'étuve est vidée et le cycle reprend. Les informations utilisées sont des évolutions de température, de volume de produit. Ce sont des informations de nature analogique (à variation continue dans le temps). Ce type d'opération est une opération batch, ou discontinue.
Le four qui réalise la cuisson procède d'une opération continue. Les pains défilent sur un tapis, un temps donné dans le four. Les informations nécessaires au contrôle sont de nature analogique : température, teneur en eau, volume, couleur, etc. Ce type d'opération est appelé opération continue. Si l'on détaille les informations à recenser pour le contrôle de l'opération, on trouve des grandeurs analogiques, des données de nature symbolique (par exemple, produit bien cuit, croûte fine, mie moelleuse, etc.), et des données de nature événementielle : démarrage de four, arrêt, changement de production, etc. Il apparaît également que nombre des opérations nécessaires ne sont pas mesurables avec des capteurs.

1.1.4 Les logiciels de supervision

La gestion informatique d'un procédé peut se réaliser au travers de l'utilisation d'un logiciel dit « de supervision ». Ce type de logiciel, souvent assez coûteux, intègre un certain nombre de fonctionnalités de base indispensables pour gérer le procédé.

La plupart du temps, on retrouve trois grands modules :
- une couche pilote de matériel, capable de dialoguer avec le matériel (les capteurs et actionneurs),
- une couche de contrôle qui assure d'une part la régulation du procédé par l'intermédiaire d'une loi de commande et d'autre part la sauvegarde des données,
- une interface utilisateur pour suivre le cours du procédé en observant les mesures.

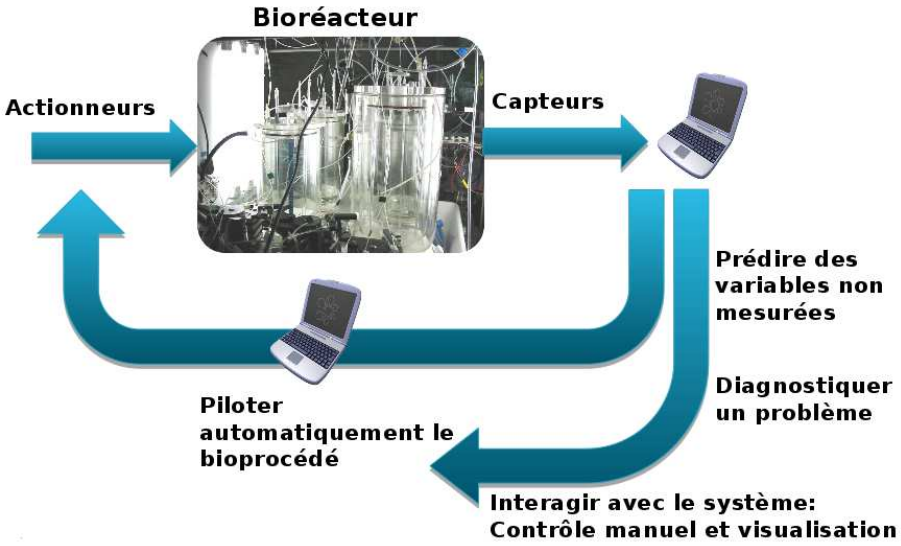


FIGURE 1.3 – Principe de fonctionnement des logiciels de supervision.

Les principaux acteurs du marché

Il nous semble intéressant de présenter quelques logiciels de supervision rencontrés dans les laboratoires ou l'industrie. Ces logiciels sont de fait des incontournables du métier. Nous citons ici trois exemples.

LabVIEW

LabVIEW est un logiciel de développement d'application de la société américaine *National Instrument* basé sur un langage de programmation graphique appelé langage **G**. Les domaines d'application traditionnels de LabVIEW sont la commande et la mesure à partir d'un PC (acquisition de données, contrôle-commande d'instruments de mesure, de dispositifs expérimentaux, de bancs de test).

Pour le développeur, un programme en langage G se présente comme un schéma, le diagramme, réunissant différentes icônes reliées par des fils de couleur. Chaque fil symbolise le passage d'une donnée depuis une source dont elle sort (comme résultat), vers une cible où elle entre (comme paramètre).

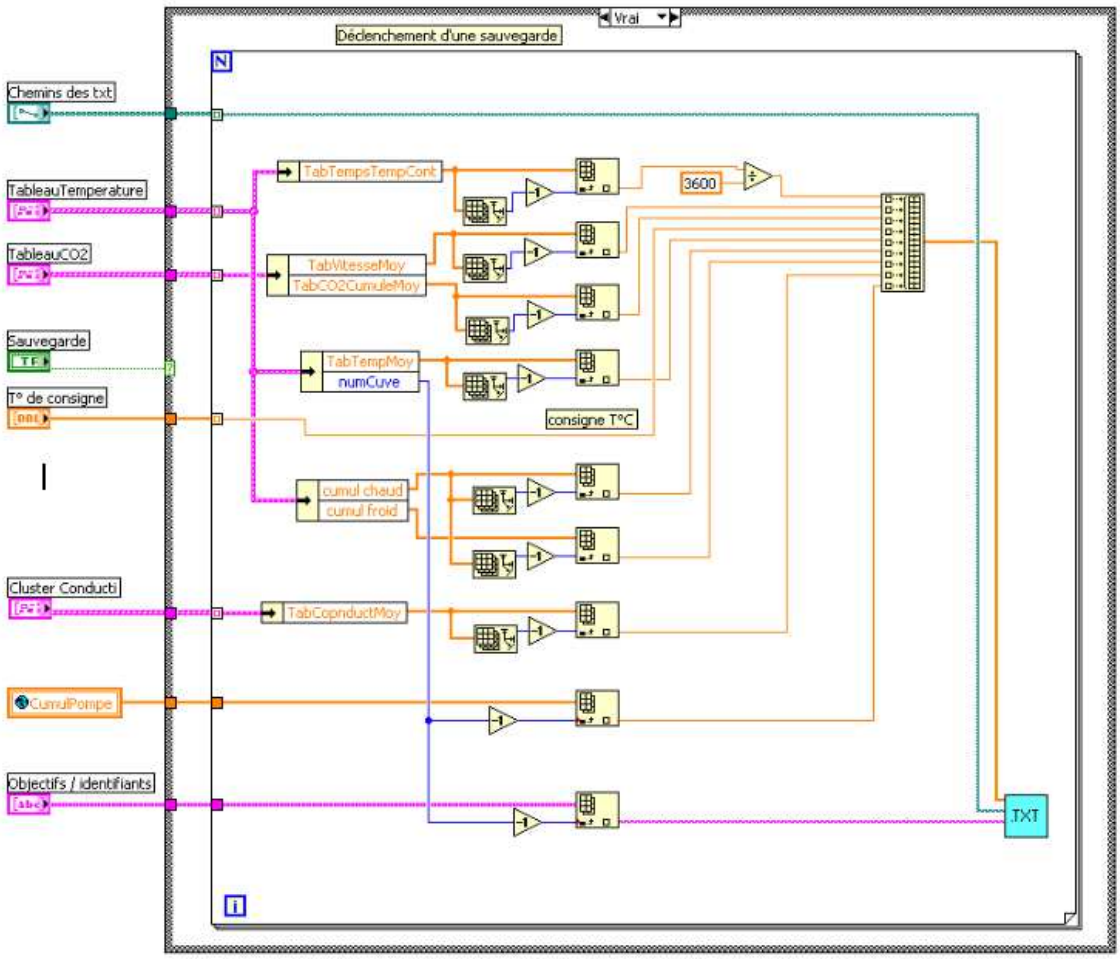


FIGURE 1.4 – Le langage de programmation graphique de LabVIEW (source : Marc Pérez, Laboratoire SPO Montpellier).

Une licence LabVIEW coûte entre 1 250 € et 4 650€.

inTouch

Ce logiciel de supervision de la société *Wonderware*, utilisé dans l'industrie, assure également l'acquisition et la commande et propose un moteur de bases de données intégré, et des fonctionnalités d'analyse de données, de conception graphique et de développement objet.

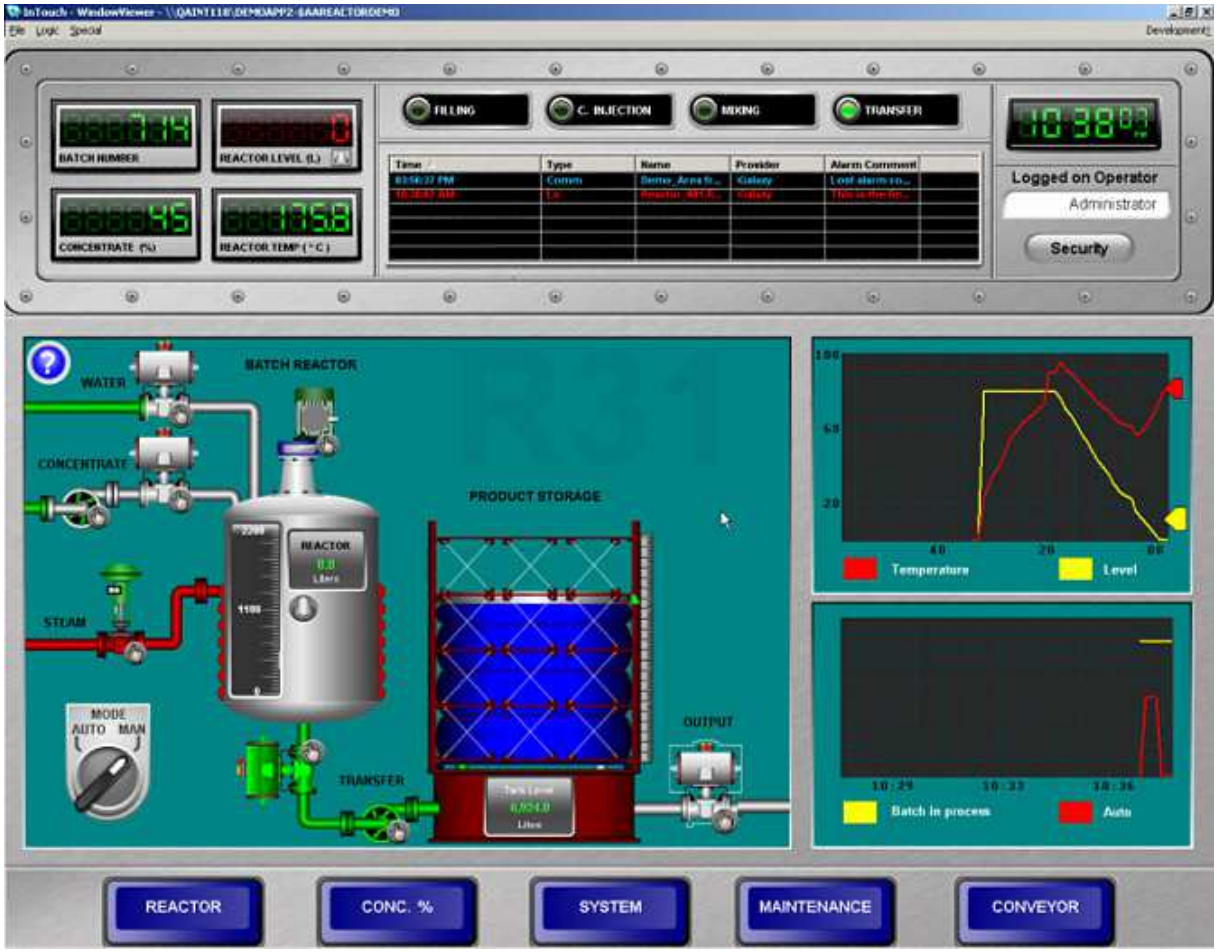


FIGURE 1.5 – Une interface de supervision réalisée avec inTouch.

inTouch est probablement le plus complet des logiciels (peut-être un peu sur-dimensionné pour un laboratoire de recherche) et affiche un prix de 4 450 €.

ODIN

ODIN est un projet *open source* sous licence CeCILL [1] développé par l'INRIA. La première version de ce logiciel est sortie en 2004 et son développement est toujours actif. ODIN se concentre sur les bioprocédés et dispose des trois fonctionnalités classiques des logiciels de supervision. Il intégre le programme SCILAB [2] qui permet de spécifier des lois de commande.

1. La licence CeCILL est une licence pour les logiciels libres issue de la licence GPL et adaptée au droit français.

2. SCILAB est un logiciel libre de calcul numérique fournissant un environnement de calcul pour des applications scientifiques.

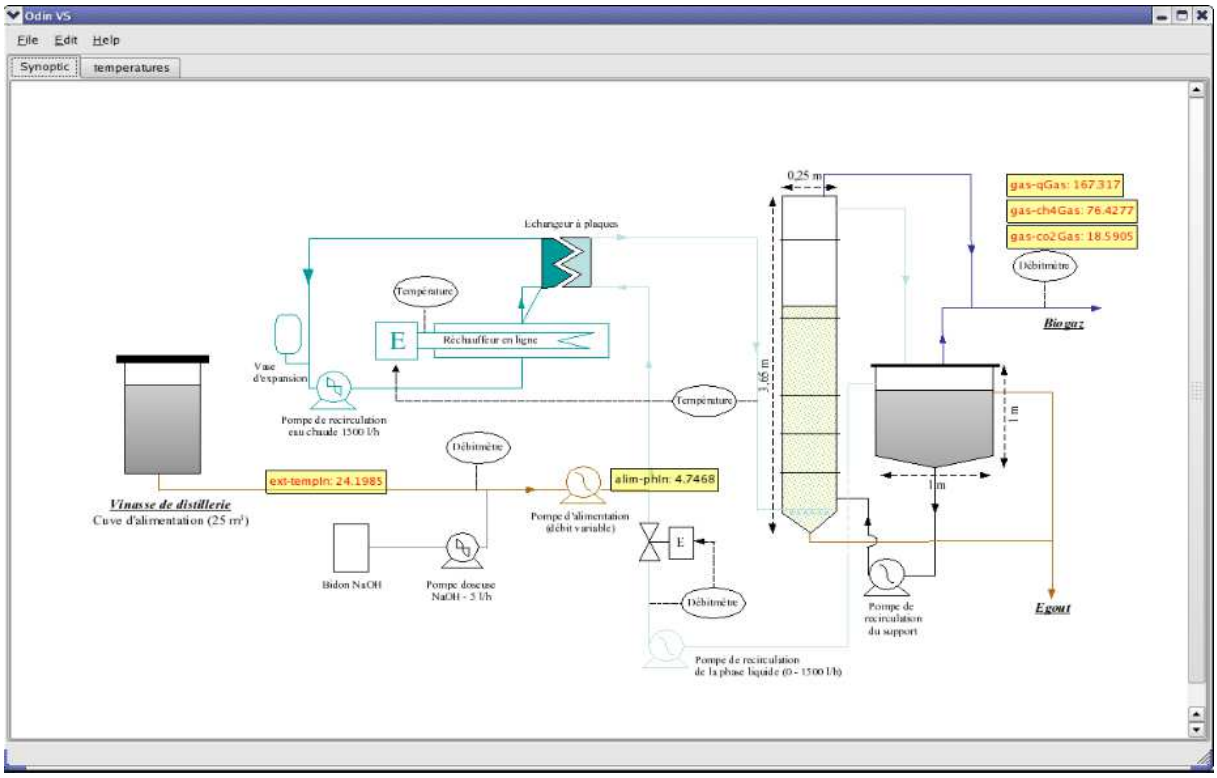


FIGURE 1.6 – L'interface de supervision de ODIN, les encadrés jaunes indiquent les variables mesurées en temps réel.

1.2 Variables d'un procédé et acquisition des données

Les mesures et les consignes sur les procédés doivent être stockées dans le système [3]. Elles deviennent des mesures de variables. Elles peuvent être classées selon leurs usages dans le procédé. Nous présentons ici une typologie de ces variables[7]. Nous traiterons ensuite des différents modes d'acquisition des mesures, c'est-à-dire de leur passage du procédé au système.

1.2.1 Variables d'état et variables objectifs

En automatique, les variables d'état déterminent la transformation, au cours du procédé, des propriétés d'un matériel alimentaire. Ces propriétés peuvent être micro-biologiques et toxicologiques, organoleptiques (texture, couleur, goût, etc.), nutritionnelles (teneur en protéines, valeur nutritionnelle, etc.) et technologiques (densité, teneur en eau, etc.).
Parmi elles se trouvent les variables objectifs, qui correspondent aux objectifs de l'opération, par exemple la teneur en eau d'un produit après un procédé de séchage.

1.2.2 Variables de commande

Les variables de commande, ayant un effet sur le système, englobent toutes les variables que l'on veut maîtriser pour obtenir une certaine évolution des variables d'état du produit.

3. Le système initialement réduit à l'expérimentateur sera ensuite progressivement, dans notre propos, un réel système d'information

L'établissement des trajectoires de variables de commande, au lieu des consignes constantes, est l'option la plus adéquate vis-à-vis de la conduite des procédés alimentaires.

1.2.3 Variables perturbatrices

Les variables perturbatrices sont des conditions qui agissent sur le système en provoquant des changements dans les variables d'état. Elles peuvent être mesurables ou non et agir de façon aléatoire ou systématique. La variabilité de la matière première, les conditions environnementales au moment de la production sont des exemples des ces perturbations. Un des objectifs en automatique est de réduire les effets de celles-ci sur le système [4].

1.3 L'acquisition des données

Les données issues du procédé peuvent être classées selon leur mode d'acquisition. On distinguera les données hors-ligne des données en ligne. Elles peuvent également être envisagées sous l'angle de leur dimension, certaines données peuvent être qualifiées de complexes.

1.3.1 Données hors-ligne

Ces données ne font pas l'objet d'une acquisition automatique. Elles nécessitent la plupart du temps le prélèvement d'un échantillon sur le procédé. Elles peuvent de ce fait introduire un biais dans le cours du procédé. On essaiera autant que possible de limiter ce biais en rendant négligeable la quantité prélevée.
Il s'agit souvent d'analyses sur le substrat mettant en jeu d'autres équipements plus ou moins automatisés.

Il est fréquent d'effectuer des analyses sur la matière première, avant le début du procédé ou bien lorsqu'il est terminé. Les mesures hors-ligne ont de ce point de vue un caractère statique. Notons également que leur quantité est souvent plus faible.

1.3.2 Données en ligne

Les données en ligne sont issues automatiquement du procédé, pendant son déroulement. Elles nécessitent des capteurs sophistiqués capables de transformer une réalité physique en une grandeur numérique. C'est l'abondance de ces mesures qui fait leur intérêt car elles permettent de saisir l'évolution d'un procédé dans sa dynamique temporelle.

Généralement, on capture simultanément un ensemble de valeurs correspondant à un instant donné. Il est cependant possible de rencontrer des installations produisant des groupes de données non synchronisés. Ce cas de figure aura des implications sur le système d'information associé.

1.3.3 Données complexes

Les données complexes sont issues de mesures qui produisent un vecteur ou une matrice de valeurs à un instant donné. Ces données posent un problème de stockage puisqu'il faudra adapter

4. En optimisation, les effets de ces variables peuvent être regroupés sous le terme d'≪ incertitudes ≫. Celles-ci peuvent être prises en compte dans l'optimisation, ceci pour obtenir des points de fonctionnement plus ≪ robustes ≫ afin de rejeter ces perturbations.

la structure de leur format de stockage sous-jacent. [5]

Exemple:
Le comptage de cellules (de levures, de bactéries etc.) est une mesure qui donne la répartition des cellules en fonction de leur taille (diamètre) dans une quantité donnée du substrat et à un instant donné.

1.3.4 Données symboliques

Les données symboliques sont de nature qualitatif. Elles sont courantes dans les procédés alimentaires, notamment parce que l'on s'intéresse aux caractéristiques organoleptiques des produits. Il peut s'agir d'une fiche de dégustation, etc.

1.4 La lettre de mission

Après avoir présenté le contexte général des procédés agroalimentaires, il convient de préciser la problématique de ce mémoire. Les procédés ont une nature complexe, il est en soi un défi pour un laboratoire de mobiliser les moyens pour les mettre en œuvre, ce qui entraine un coût de production de la donnée élevé.
Or, on constate une exploitation relativement limitée de ces données. Limitation due en grande partie à l'absence de réflexion sur le système d'information hôte. La plupart du temps, une série d'expérimentations est lancée par un chercheur ou une équipe, avec un objectif de recherche bien défini. L'accumulation des données commence et se poursuit jusqu'à l'achèvement du programme de recherche. Une production scientifique est faite. Les équipes passent. La donnée retombe dans l'oubli.

Dans la réalité, le tableau que nous dépeignons est parfois moins sombre. Un groupe d'expérimentations fait souvent partie d'un projet implicite (thèse, programme de recherche, partenariat avec un industriel, etc.), les équipes peuvent identifier pendant un certain temps dans quel contexte se situe une donnée. Cependant, même dans ce cadre là, l'absence d'un certain nombre d'informations peut s'avérer bloquante. La figure 1.7 montre la représentation graphique de la vitesse de dégagement de CO_2 lors d'une fermentation alcoolique. La courbe en bleu représente les données présentes dans la base. Les indications en rouge, expliquant les pics de variation dans les mesures, ne sont pratiquement jamais présentes. Elles sont pourtant fondamentales pour pouvoir exploiter à nouveau ce jeu de données. En leur absence, on ne peut plus rien faire des données brutes.

5. Ce qui rendra difficile la définition d'un schéma générique si le support envisagé est une base de donnée

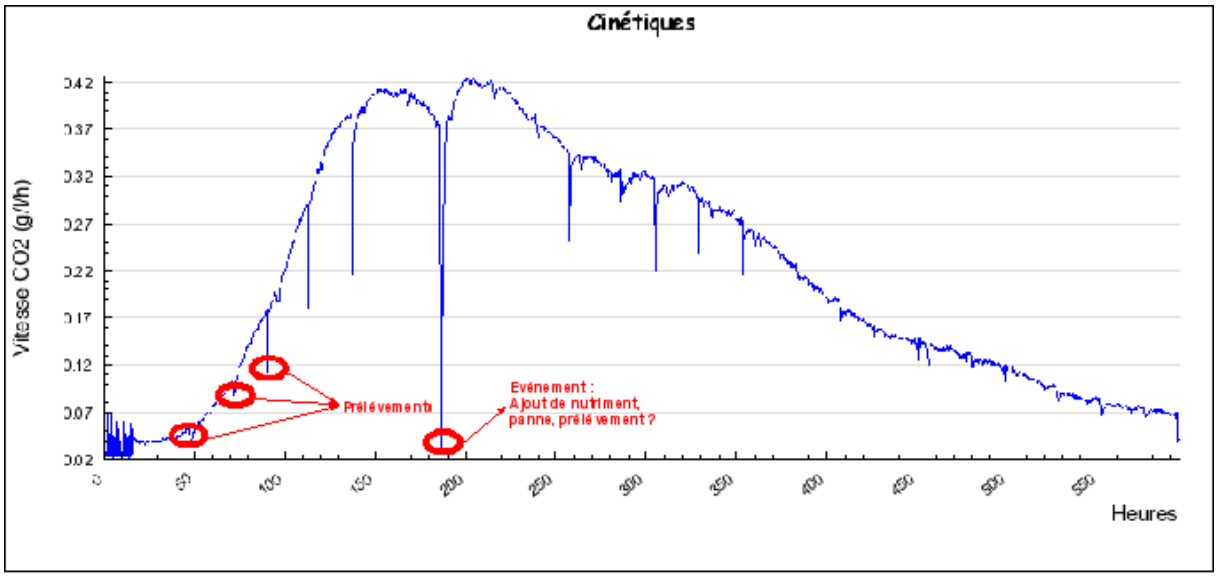


FIGURE 1.7 – L'absence d'information sur les données peut empêcher leur réutilisation.

Nous nous proposons donc d'apporter des solutions à cette problématique de pérennité de la donnée scientifique et de son contexte à des fins de réutilisation. Nous allons pour cela explorer le domaine de l'**ingénierie des connaissances** pour concevoir un réel système d'information accompagnant les efforts de recherche et permettant de mieux les capitaliser.
L'ingénierie des connaissances est une discipline étudiant l'extraction et la formalisation de connaissances venant d'un expert humain. Le terme est souvent associé aux systèmes experts qui permettent de résoudre des problèmes dans un domaine d'application déterminé à l'aide d'une base de connaissances établie à partir de l'expertise humaine. Notre solution n'est pas un système expert mais elle utilise un certain nombre d'idées des systèmes experts, comme les bases de connaissances et l'inférence.

Nous allons essayer de modéliser la connaissance experte des bioprocédés et d'utiliser cette modélisation pour décrire les données (et assurer leur stockage), les expérimentations et leur contexte. Ce faisant, nous découvrirons un certain nombre d'externalités positives sur le système d'information.

1.5 Cas d'utilisation de référence

Le cas d'utilisation [6] de référence nous servira de fil conducteur pour construire une solution adaptée au besoin des utilisateurs. Il décrit la démarche typique d'un chercheur désireux de lancer une ou plusieurs expérimentations et d'en consulter les résultats. Le système d'information que nous nous proposons de mettre en place se devra de répondre à ces exigences minimales.

Première étape, l'utilisateur déclare l'expérimentation, il précise :
- son identité,
- les quantités en jeux,

6. En langage UML, les cas d'utilisation servent à capturer les différentes façons dont les utilisateurs se serviront du système.

- les matières premières,
- les autres produits,
- ce qui va être mesuré,
- la finalité de l'expérience.

Éventuellement, il peut préciser le plan d'expérience et un profil de régulation. La matière première peut faire l'objet d'une analyse préalable.
Pendant le cours du procédé, l'utilisateur :

- suit l'évolution des mesures,
- effectue des prélèvements qu'il analyse,
- réalise des opérations sur le procédé,
- constate des anomalies dans le déroulement.

Lorsqu'il estime que le procédé est terminé, l'opérateur le signale au système qui stoppe l'acquisition des mesures. L'équipement du procédé est disponible pour une nouvelle expérimentation.

Pour compléter ce cas d'utilisation standard, nous ajoutons quelques exigences supplémentaires :

- l'utilisateur doit pouvoir lancer plusieurs procédés en même temps,
- le système doit avoir une bonne utilisabilité en terme d'ergonomie, d'esthétique et de documentation,
- le système doit être performant,
- le système doit avoir une bonne capacité à être maintenu, configuré et étendu.

Chapitre 2

L'état de l'art

2.1 Les systèmes d'information autour des procédés

L'état de l'art s'intéresse à l'existant, c'est-à-dire aux systèmes d'information que l'on retrouve associés aux procédés. Le terme « système d'information » peut avoir de multiples définitions, pour lever toute ambiguïté nous retiendrons la suivante.

Système d'information Un système d'information (SI) est l'ensemble des ressources humaines, logicielles et de données participant à la gestion, au traitement, au transport et à la diffusion de l'information au sein d'une organisation.

Il est hélas fréquent, dans le contexte des recherches expérimentales, de ne pas disposer d'un système d'information informatisé permettant de garantir la pérennité des expériences et des données afin de permettre leur utilisation ultérieure. Nous différencierons à cet effet la notion de système expérimental, qui peut être informatisé, de système d'information qui présente les caractéristiques mentionnées ci-dessus.

Nous recensons ici différentes architectures de systèmes expérimentaux rencontrées dans les laboratoires de recherche. Nous présentons une typologie non exhaustive de l'organisation des systèmes expérimentaux dans les procédés alimentaires. Les solutions retrouvées dépendent grandement des compétences locales des informaticiens responsables des SI.

2.1.1 Solution manuelle

Dans ce cas de figure, de plus en plus rare, il n'y a pas d'informatisation. Les différents acteurs effectuent des prélèvements d'échantillons sur le procédé, réalisent des analyses et notent les résultats sur des cahiers de laboratoire. La régulation est nécessairement manuelle.

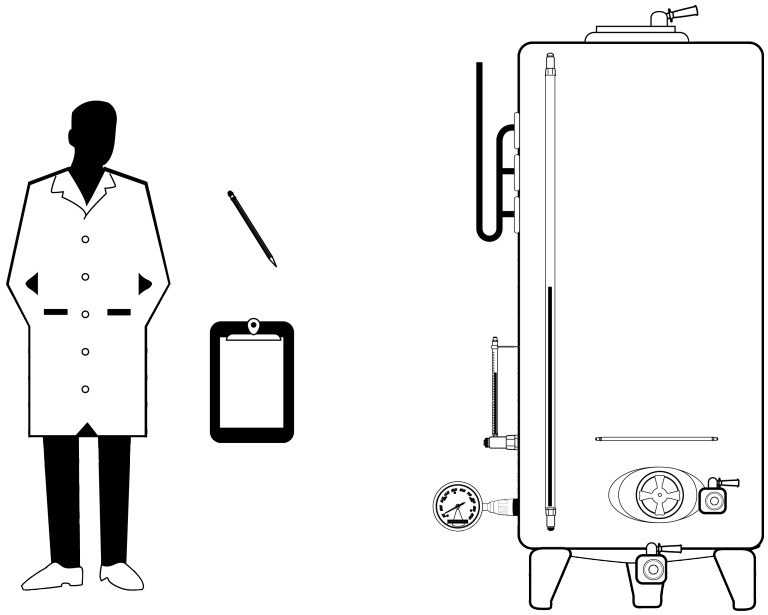


FIGURE 2.1 – L'acquisition manuelle

Nous ne nous étendrons pas d'avantage sur ces méthodes non automatisées.

2.1.2 Solution d'acquisition automatisée

Parfois, la communication entre les capteurs et les ordinateurs est entièrement gérée par des programmes spécifiques mis au point par des ingénieurs de support à la recherche [1].

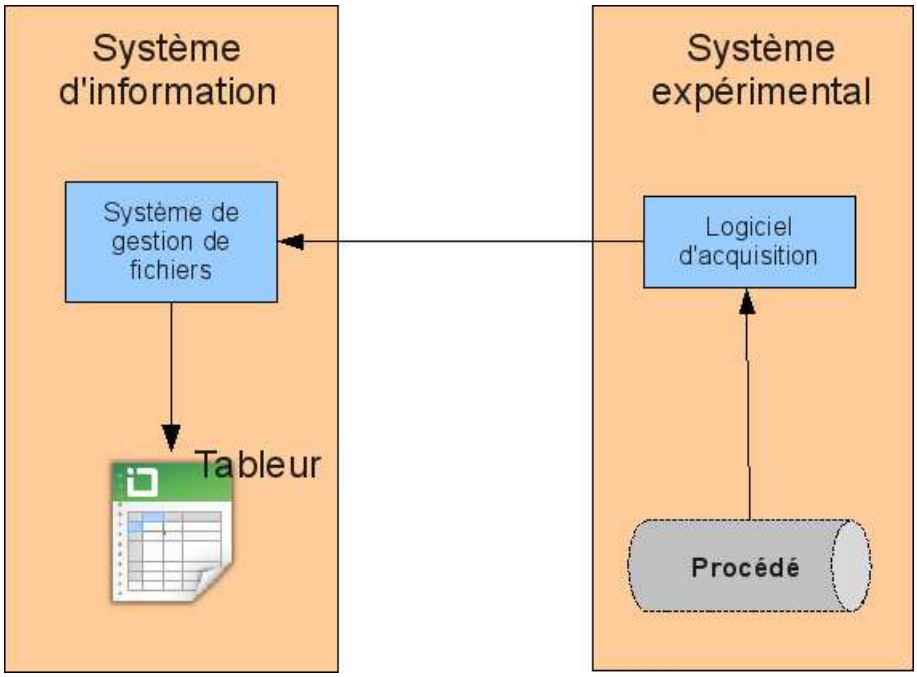


FIGURE 2.2 – Solution ad-hoc d'acquisition automatisée : le système expérimental est informatisé.

1. On retrouve tout type de systèmes (DOS, UNIX, Windows etc.) et langages de programmation (C,C++, TCL/TK, etc.), souvent proche du matériel

Si une régulation est nécessaire, elle sera souvent manuelle. C'est-à-dire qu'un opérateur interviendra sur le procédé pour activer un actionneur.

Le fichier plat (ou tabulé) est le format de stockage le plus fréquemment retrouvé. Le format tableur (excel, lotus, openoffice) est également souvent utilisé puisqu'il permet à la fois de conserver les données et d'en faire une analyse statistique rudimentaire. Les fonctions de production de graphique qui lui sont associées sont appréciées.

Solution avec logiciel de supervision

C'est le cas le plus fréquemment rencontré. Un logiciel commercial de supervision (et d'acquisition) assure la communication avec les capteurs et les actionneurs, et écrit les données dans des fichiers. La flêche « contrôle » indique un flux d'information à destination des actionneurs.

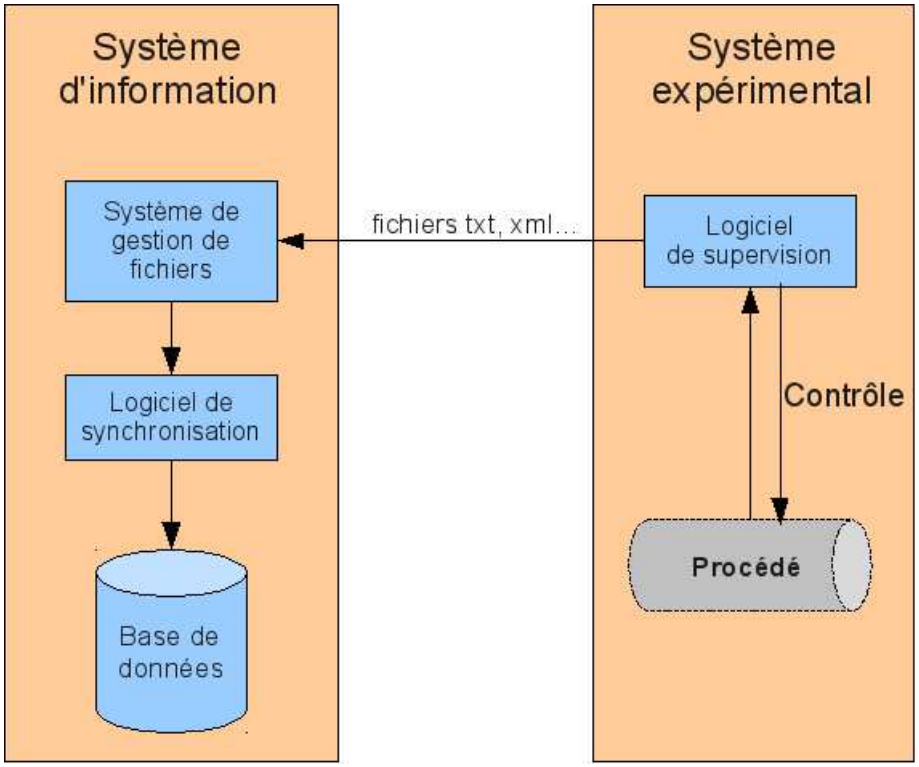


FIGURE 2.3 – Cas classique : utilisation d'un logiciel de supervision

Le format de ces fichiers est généralement simple, soit des fichiers csv [2] (ou tabulé), soit un format XML.

2.1.3 Solution distribuée

Il arrive fréquemment que le procédé et les ordinateurs servant à le superviser soient éloignés physiquement de la base de données. Le transport de l'information est alors nécessaire. Dans ce cas, ou bien on réalise le transport des fichiers csv ou tabulé par l'intermédiaire de protocoles

2. Comma-Separated Values : un format ouvert représentant les données sous forme de valeurs séparées par des virgules

de transfert de fichier **FTP** [3] ou bien on peut mettre en place un service Web, sur le système contenant la base.

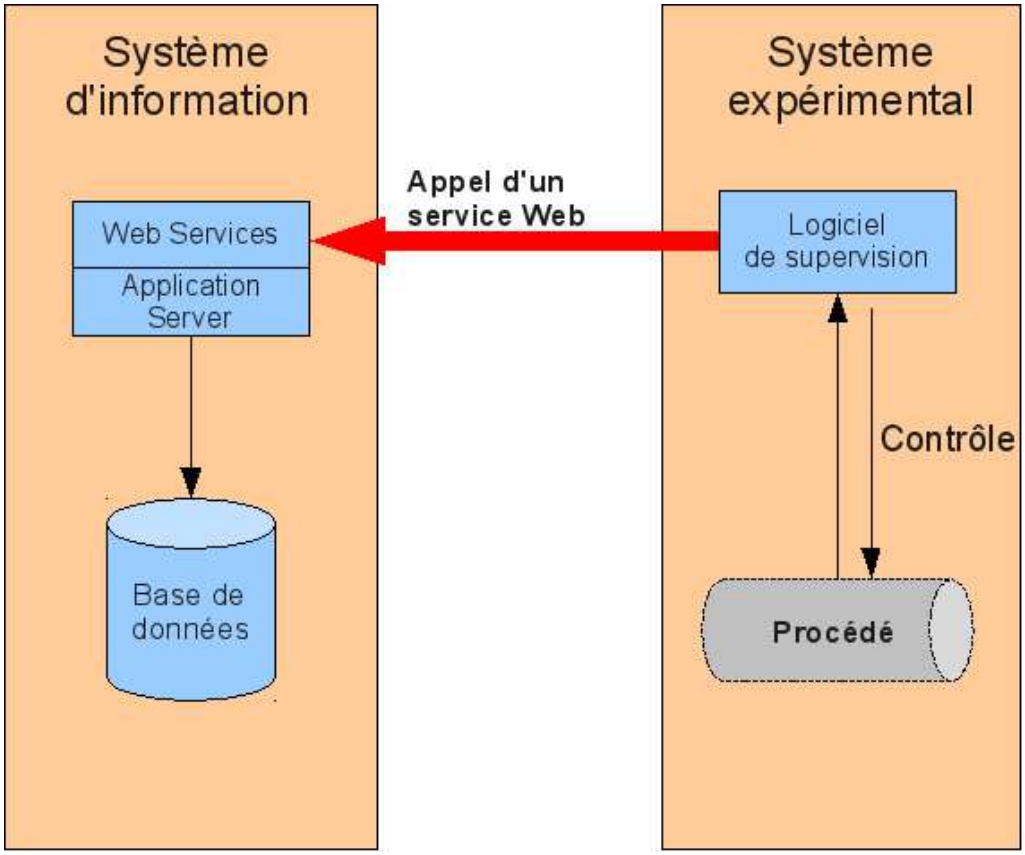


FIGURE 2.4 – Serveur d'application et Web Services

Notons que beaucoup de logiciels de supervision récents permettent de faire des appels à des Web Service. Cette architecture nécessite cependant plus d'investissement initial et de maintenance d'un **serveur d'application**.

2.2 Modèles pour les bases de données

Nous avions signalé précédemment que le support de stockage des données pouvait être une base de données. Nous présentons ici les différents modèles pour les systèmes d'information utilisant des bases de données relationnelles. Ce sont des modèles logiques, nous discuterons brièvement des avantages et inconvénients de chacun d'eux. L'expérimentation et les mesures associées sont en général représentées par deux **tables** : une pour le le concept d'expérimentation, une présentant les divers type de mesures (censés être connus à l'avance). Le code de l'expérimentation, clé de la table expérimentation, est clé étrangère dans la table des mesures.

Cas 1 : une table pour les mesures

Le modèle est organisé autour d'une table qui enregistre les données des grandeurs mesurées. Un champ est utilisé pour chaque variable mesurée.

3. File Transfert Protocol

Mesures				
ExpId	Date	pH	Temp	Conductivite
M12-01-09F01	12/01/2010 10:12:25	7.3	21	12
M12-01-09F01	12/01/2010 10:22:25	7.1	21.3	11
...				

Avantages Ce modèle est le plus simple à mettre en place et probablement l'un des plus performants concernant l'extraction des données.

Inconvénients La modification de l'instrumentation (ajout de nouveaux capteurs) implique la modification de la structure de la table. Incidemment, la table peut contenir beaucoup de valeurs `NULL` si certaines variables ne sont mesurées que rarement.

Cette solution est souvent retenue pour stocker les mesures en ligne.

Cas 2 : Une table pour les mesures et les variables

La table des mesures enregistre une valeur par n-uplet, un champ faisant référence au type de mesure (identifiant de variable).

Mesures			
ExpId	**IdVariable**	**Date**	**Valeur**
M12-01-09F01	ph	12/01/2010 10:12:25	7.3
M12-01-09F01	temp	12/01/2010 10:12:25	21.0
M12-01-09F01	conductivite	12/01/2010 10:12:25	12.0
M12-01-09F01	ph	12/01/2010 10:22:25	7.1
...			

Variables			
Id	**Nom**	**Unité**	**SymboleUnite**
ph	pH	NULL	NULL
temp	Temperature	Degré	°C
conductivite	Conductivité	Siemens par mètre	$S.m^{-1}$
...			

Avantage Dans ce modèle, il est plus simple d'ajouter de nouvelles variables (nouveaux capteurs), il suffit d'insérer un nouveau n-uplet dans la tables `Variables`. Cette table permet en outre de décrire de manière plus complète la variable.

Inconvénients

- il n'y a pas de contrôle sur le type de la donnée ainsi que sur le domaine,
- chaque jeu de mesure occupe autant de n-uplets que de variables à mesurer sur le procédé, l'espace occupé grandit beaucoup plus vite et l'on peut rencontrer des problèmes de performance assez rapidement [4],

4. Dans la pratique, les administrateurs de base de données sont conduits à partitionner la table des mesures lorsqu'elle devient trop lourde, ce qui induit souvent une complication au niveau de la maintenance des applications.

– la réalisation d'une requête, même simple, implique une jointure supplémentaire.

Cas 3 : une solution mixte, 3 colonnes

Ce modèle est une variante du précédent et a pour objectif de permettre le typage des données. La table est construite avec une colonne pour chaque type de valeur que l'on mesure, type au sens type de donnée des bases de données. Un seul des champs est utilisé pour stocker la valeur de la donnée, les autres restent à `NULL`.

Mesures					
ExpId	**IdVariable**	**Date**	**ValReelle**	**ValEntiere**	**ValTexte**
M12-01-09F01	ph	12/01/2010 10:12:25	7.3	NULL	NULL
M12-01-09F01	nb cellules	12/01/2010 10:12:25	NULL	523	NULL
M12-01-09F01	couleur	12/01/2010 10:12:25	NULL	NULL	'rouge'
M12-01-09F01	ph	12/01/2010 10:22:25	7.1	NULL	NULL
...					

Naturellement, une table recense les variables et précise le type de la donnée.

Cas 4 : Une table pour chaque variable

Cette proposition essaie d'apporter de la souplesse au système tout en conservant les possibilités de typage et de définition de domaine de valeur.

Une table recense les variables et leurs paramètres (nom, type, domaine de valeur, unité) et on associe un déclencheur (trigger) sur l'insertion d'un nouveau n-uplet dans cette table. La déclencheur lit les valeurs du n-uplet inséré et à partir de celles-ci crée une nouvelle table de nom le nom de la variable et avec 3 champs : code, date et valeur de type (le type de l'insertion). spécifié.

temperature		
ExpId	**Date**	**Valeur**
M12-01-09F01	12/01/2010 10:12:25	21.0
...		

Cette solution est intéressante dans la pratique puisque les données issues des procédés sont souvent des séries temporelles. Ainsi une requête pour extraire les données d'une variable pour produire un graphique s'avère extrèment simple. Le lecteur trouvera en annexe A le script SQL (sous PostgreSQL) réalisant cette solution.

Chapitre 3

Analyse et proposition

Nous présentons ici une analyse métier des bioprocédés qui nous conduira à la construction d'une ontologie des procédés agroalimentaires. Une ontologie est l'ensemble structuré des termes et concepts d'un domaine. On peut se représenter une ontologie comme un vocabulaire sur lequel l'ensemble des utilisateurs s'accordent. L'ontologie va expliciter le modèle du procédé, celui des données afférentes, du contexte etc.

3.1 Méthodologie

Les modèles qui suivent ont été construits lors d'entretiens avec les utilisateurs et les experts du domaine selon la méthode suivante :

1. délimitation d'un sous-domaine de connaissance, par exemple les opérations,
2. interview d'expert,
3. fusion dans l'ontologie et vérification de la cohérence,
4. soumission à un regard extérieur, non-expert mais utilisateur, pour identifier les idées les plus simples qui font souvent défaut,
5. vérification finale de la part des experts.

Les modèles présentés utilisent le formalisme UML. Ils sont construits avec le maximum d'abstraction par rapport à ce que seront les choix techniques finaux.

3.2 Noyau du modèle

Nous qualifions de noyau la partie du modèle a priori minimale pour que le système puisse enregistrer des mesures. Il est établi à partir de l'état de l'art. Dans la réalité, on trouve en général tout ou partie de ce modèle « minimal ».

3.2.1 Procédé et procédé unitaire

Dans le vocabulaire « métier », le mot *procédé* est utilisé pour désigner deux concepts différents. D'une part le procédé en tant que principe et d'autre part le procédé en tant qu'appareillage technique sur lequel se déroule le procédé.

Le procédé en tant que principe

Un procédé est une séquence de procédés unitaires et d'opérations. Un procédé unitaire est une composante unique faisant partie d'un procédé de fabrication qui transforme chimiquement ou biochimiquement une matière première en produits finis.

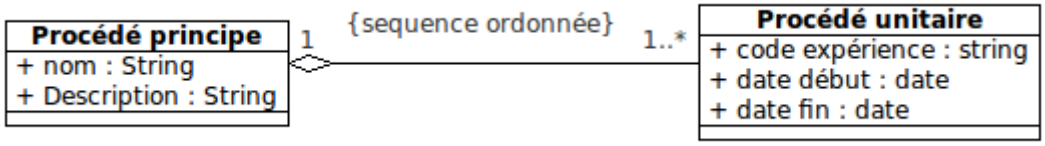


FIGURE 3.1 – Un procédé est une suite d'opérations unitaires.

Exemple:
Le procédé de lyophilisation est constitué de la séquence des procédés unitaires suivants :
- *pré-culture d'un inocula,*
- *fermentation,*
- *concentration et centrifugation,*
- *lyophilisation,*
- *stockage.*

Chaque procédé unitaire peut faire l'objet d'expérimentations.

Le procédé en tant qu'installation technique

D'autre part, on désigne sous le terme de *procédé* l'installation technique sur laquelle se déroule le procédé, par exemple une cuve ou un réacteur. Sur les diagrammes d'analyse, nous utiliserons le mot « appareillage ».

3.2.2 Le modèle

À partir de ces définitions, nous pouvons construire le cœur de notre modèle.

Les expérimentations

Le point de départ est le procédé en tant que principe, dont les instances sont déclenchées par les utilisateurs. Un *appareillage* représente le lieu sur lequel se déroulent les *procédés unitaires*. Dans un cadre scientifique, une instance de procédé unitaire constitue une *expérimentation*. L'expérimentation est un concept central dans le génie des procédés.

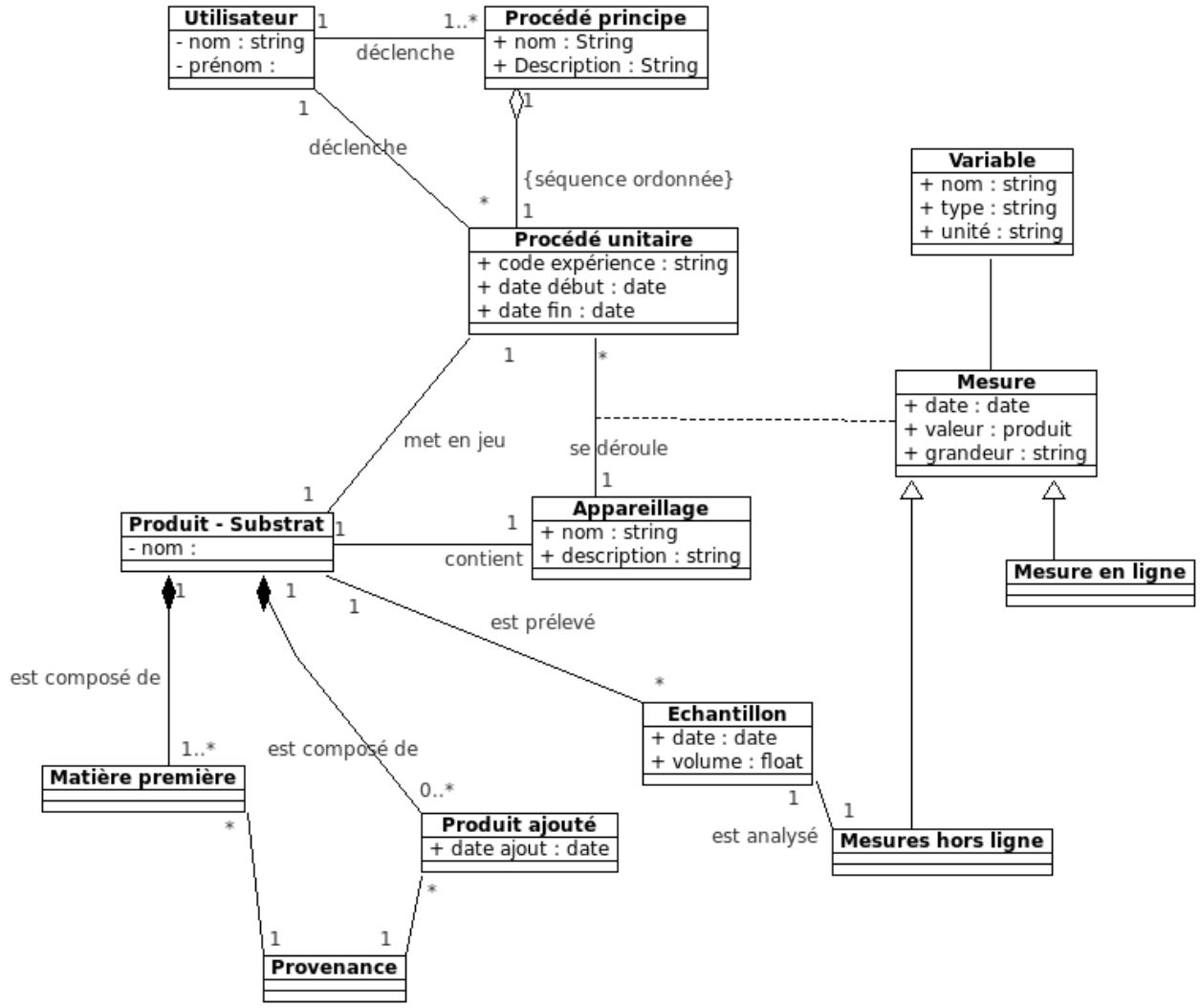


FIGURE 3.2 – Modèle du procédé.

3.2.3 Les produits

Un procédé unitaire met en jeu des produits, leur mélange constitue un substrat. Les produits sont de deux sortes, les matières premières qui sont indispensables au déroulement du procédé et les produits ajoutés, qui ne sont pas indispensables mais qui font souvent l'objet d'études approfondies.

Exemple:
Le procédé de fermentation alcoolique contient au moins un moût (la matière première), mais aussi très souvent des levures, du dioxyde de soufre, etc. que l'on qualifie de produits ajoutés.

3.2.4 Les mesures

Dans un contexte d'expérimentations scientifiques, l'objectif est de stocker un ensemble de mesures. On retrouve naturellement la typologie présentée en 1.3. Une mesure en ligne est donc issue de l'association d'un procédé unitaire avec l'appareillage sur lequel il se déroule, ce dernier comprenant l'ensemble des capteurs indispensables à la réalisation de cette mesure. Les analyses ou mesures hors-ligne sont réalisées sur des échantillons prélevés sur le procédé ou sur les produits (par exemple avant le début de l'expérimentation).

Ce modèle minimal suffit pour stocker les données brutes et permettre leur exploitation. Cependant, pour assurer à l'expérimentation une utilité pérenne, nous proposons d'ajouter des informations d'ordre sémantique.

3.3 Ajouts sémantiques

Partant du constat que la donnée expérimentale sortie de son contexte n'a plus aucun sens et ne peut donc plus être exploitable, nous proposons ici d'adjoindre aux données brutes une couche de données complémentaires, accompagnant les données brutes et leur donnant du sens. Ces données sur les données sont qualifiées de métadonnées.

Basiquement, ces métadonnées pourraient être du texte simple décrivant le contexte dans lequel les données brutes ont été acquises. Cependant, dans un contexte informatisé, le texte brut ne permet que des traitements basiques (**recherche plein texte** par exemple). Nous disposons aujourd'hui de langages de formalisation des métadonnées que nous pouvons utiliser pour enrichir les descriptions d'un petite dose d'automatisation. Les techniques proposées sont largement présentées dans la partie suivante de l'ouvrage. Nous présentons ici les domaines propres aux bioprocédés.

3.3.1 Les thèmes de recherche

Un des premiers objectifs des chercheurs, dans le domaine des procédés, est d'expliquer pourquoi l'expérimentation a été réalisée. C'est cependant un exercice de réflexion complexe et long à mener puisque l'expérimentation dépend du domaine dans lequel elle est menée.

Il a donc été difficile d'établir une arborescence de thèmes *génériques*. Nous nous contenterons donc de proposer une entrée *Thèmes de recherche* dans l'ontologie à partir de laquelle on pourra créer des thèmes adaptés au contexte du procédé.

À titre d'exemple, nous proposons une hiérarchie de thèmes de recherche applicables au procédé de fermentation alcoolique.

1. fermentations à vitesse CO_2 constante
2. fermentations an-isothermes
3. étude d'un paramètre
   (a) azote
      i. azote initial
      ii. azote en cours de fermentation
   (b) oxygène

   (c) température
   (d) clarification des moûts
4. étude d'une levure
   (a) levure sur-produisant un composé
      i. liste de composés
   (b) ...

Cette hiérarchie sera amenée à croître en fonction de l'évolution des thèmes de recherche abordés par les laboratoires.

3.3.2 Les évènements

Une hiérarchie d'évènements se produisant sur le procédé est proposée, elle différencie les opérations volontaires des incidents. On distingue les opérations unitaires et les pannes.
Une opération unitaire concerne la transformation physique d'une matière première, par exemple : filtration, centrifugation, cuisson, etc. Cette hiérarchie doit bien évidemment être spécialisée en fonction du procédé considéré. L'opération de *remontage* propre à la fabrication du vin n'a pas lieu d'être prise en compte dans un procédé de filtration de la bière.
Les pannes sont également spécifiques aux procédés.

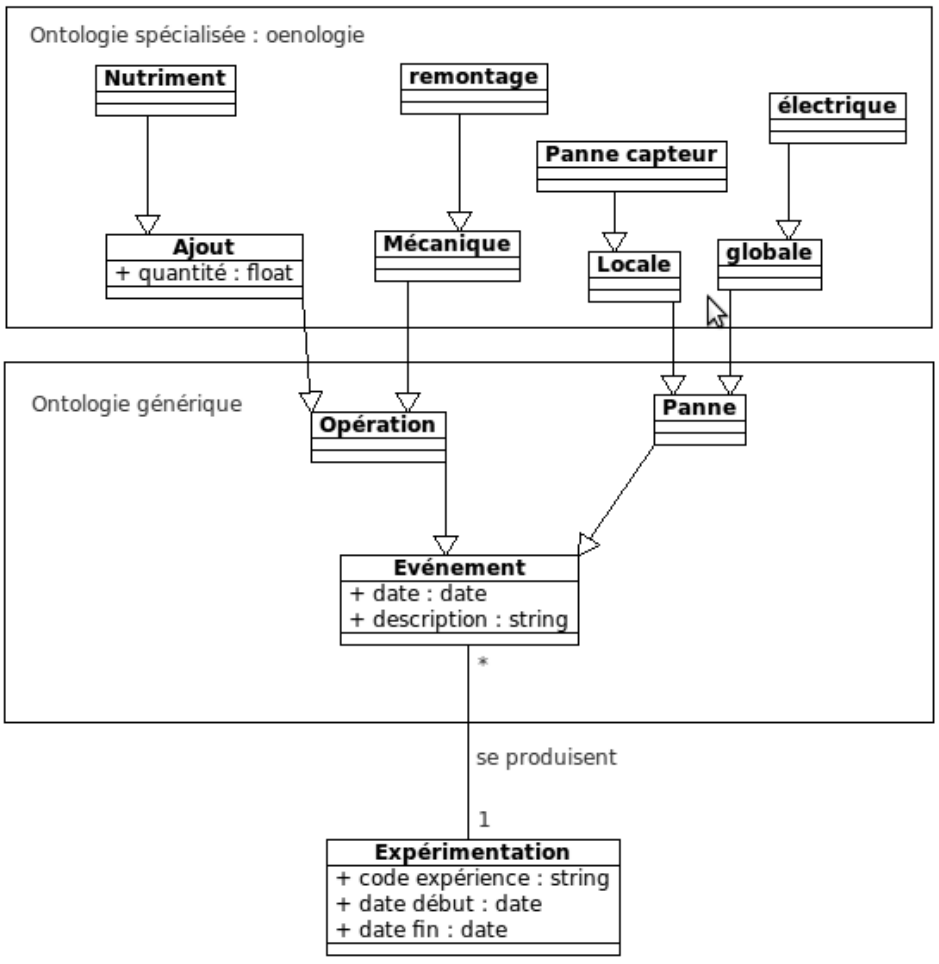


FIGURE 3.3 – Hiérarchie d'événements.

3.3.3 Le contexte d'instrumentation

La notion de contexte d'instrumentation est nouvelle. Nous définissons le contexte d'instrumentation d'un procédé comme l'ensemble des actionneurs et capteurs installés sur un procédé, produisant des *données en ligne* ainsi que les variables et unités associées.
Le contexte d'instrumentation est amené à être modifié chaque fois qu'un capteur ou actionneur est ajouté ou remplacé.

Les unités de mesures

Ce concept fait apparaître le besoin d'un vocabulaire pour traiter des unités de mesure. Une recherche a montré que plusieurs initiatives existent pour gérer les unités de mesures. Citons notamment OBOE[6] (Extensible Observation Ontology) et UnitDim[15] de l'université de Wageningen. Ces deux projets sont assez aboutis mais peut-être un peu complexes par rapport à nos besoins. Nous nous contenterons de nous en inspirer.
Nous retenons les concepts d'unité de base [1], de multiplicateur et de décalage (offset) qui permettent les conversions d'unité.

> ***Exemple:***
> *Nous souhaitons effectuer une conversion de mesures exprimées en degré Celsius en degré Fahrenheit : nous transformons dans l'unité de base, le Kelvin, en additionnant le décalage 273,15 et en multipliant par 1, puis nous retournons vers les degrés Kelvin en soustrayant le décalage 255,402 et en divisant par le multiplicateur 0,556.*

Les informations de décalage et de multiplicateur se trouvent dans l'ontologie.

Explication du diagramme

Le point de départ est le procédé (physique) auquel on associe un contexte d'instrumentation à une date donnée. Un jeu de variables correspond à un ensemble de variables dont les valeurs, issues des capteurs, sont synchronisées. La plupart du temps, il n'y a qu'un seul jeu de variables.

1. Le système internationnal retient pour chaque unité et ses variantes une unité de référence, afin de gérer les conversions.
Il comporte 7 + 2 unités de référence. Voir annexe B.

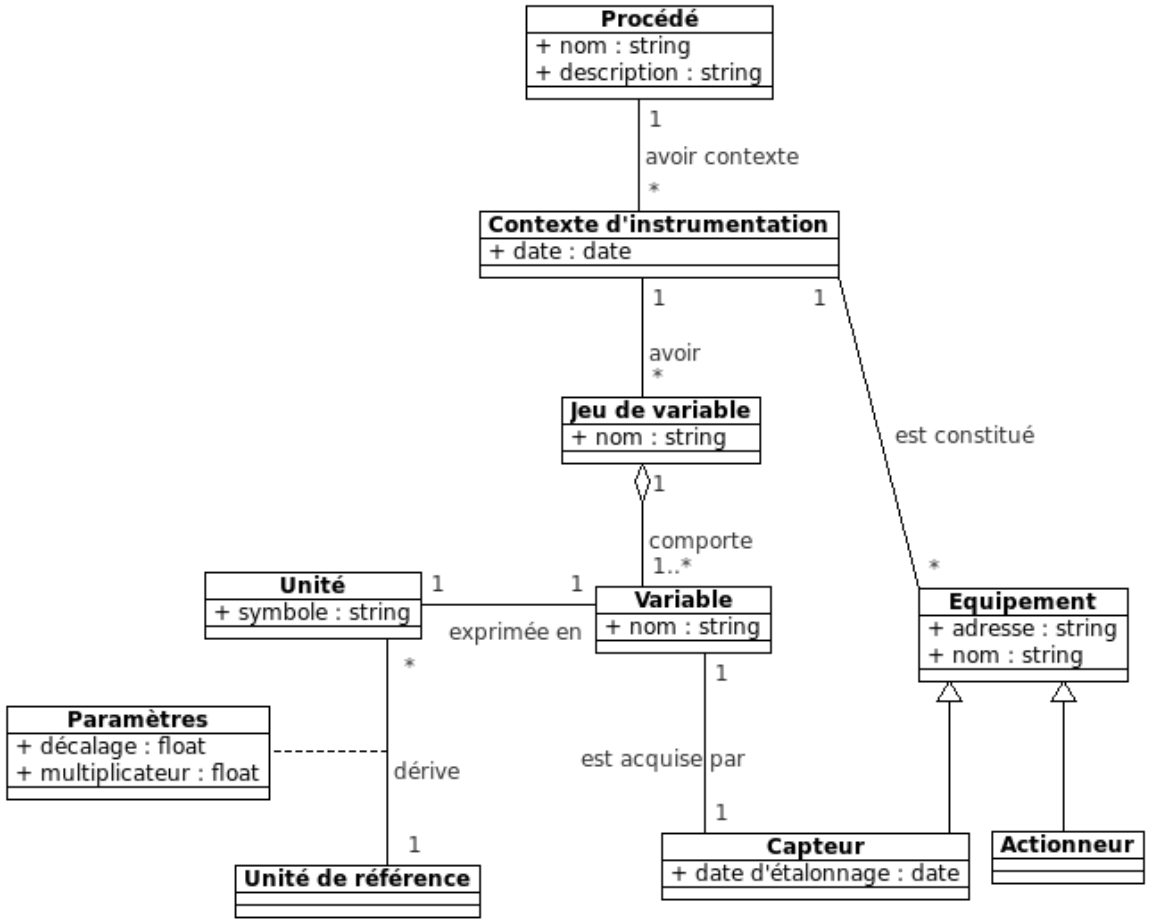


FIGURE 3.4 – Le contexte d'instrumentation.

3.3.4 Description des expérimentations

Décrire une expérimentation est fondamental pour lui donner une utilité future. Il est important d'indiquer :

- le contexte de recherche dans lequel elle s'inscrit,
- les paramètres de base comme les matières premières, les produits ajoutés, le volume, etc.
- les paramètres de régulation.

La régulation (voir page 10) qui s'opère par les actionneurs consiste à fournir des trajectoires cibles pour certaines variables du procédé. Elle s'applique à un procédé unitaire.

La régulation peut s'opérer manuellement par l'intermédiaire d'un opérateur humain. Lorsqu'elle est automatisée, elle utilise une loi de commande. La loi de commande s'appuie sur un certain nombre de consignes (ou variables de commande) et sur les mesures issues du procédé. Elle effectue des calculs et transmet éventuellement des ordres au module de commande qui déclenchera les actionneurs.

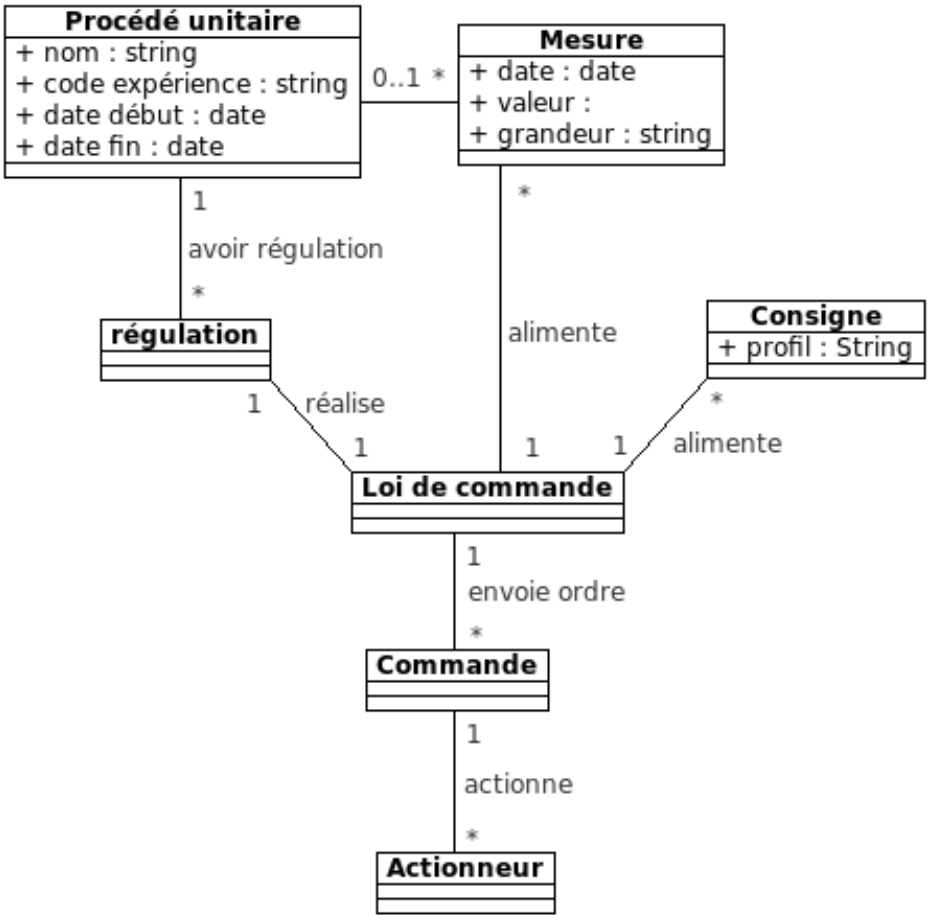


FIGURE 3.5 – La régulation.

Exemple:
On souhaite réguler la température au sein d'une cuve dans un procédé de fermentation au moyen d'un circuit d'eau chaude. On fixe une consigne de température de 20 ° C. La loi de commande reçoit les mesures de température sur la cuve et les compare à la consigne. Si l'écart est trop important, elle signale au module de commande d'actionner le circuit d'eau chaude.

La définition des lois de commande est une activité de recherche en soi et elle met en jeu des algorithmes qui peuvent être très complexes. Elle peut aussi être assez basique. Dans tous les cas, il est important de connaître les consignes.

Chapitre 4

Mise en œuvre

Ce chapitre présente une proposition d'implémentation et d'intégration de notre modélisation. L'idée générale est d'utiliser l'ontologie définie au chapitre précédent en tant que ciment pour le système d'information. Le déploiement est présenté au chapitre suivant, le lecteur peut se reporter à la figure page 51 pour avoir une vue d'ensemble.

Le chapitre est structuré en deux parties. La première présente le modèle relationnel de la future base dérivé d'une partie du modèle du chapitre précédent. La seconde présente les technologies du web sémantique ainsi que leur utilisation dans le cadre des bioprocédés.

4.1 Le modèle pour les données brutes

4.1.1 Le modèle relationnel

Le modèle relationnel proposé par E.F. Codd repose sur des principes mathématiques ensemblistes.

On appelle « schéma relationnel » un ensemble d'attributs définissant une intention, et chaque instance de ce schéma est appelée relation et comporte un ensemble de n-uplet. Les relations sont habituellement représentées sous forme de table. Codd a défini une *algèbre relationnelle* et des opérateurs permettant de construire d'autres relations à partir de relations. Cette théorie est à la base de la plupart des systèmes de gestion de base de données.

Nous ne nous étendrons pas d'avantage sur le modèle relationnel. Ce qui est important pour nous, c'est que les technologies le mettant en œuvre ont été éprouvées, sont fiables et performantes.

4.1.2 La dérivation du modèle d'analyse

Par dérivation, nous entendons « changer de niveau d'abstraction », c'est-à-dire s'approcher du modèle physique qui sera implanté en machine. Ce nouveau modèle est adapté à une base de données relationnelle sans que nous nous soyons arrêtés sur un type particulier de SGBD [1].

1. **S**ystème de **G**estion de **B**ase de **D**onnées.

Nous dériverons essentiellement les données de la figure 3.2 page 27.

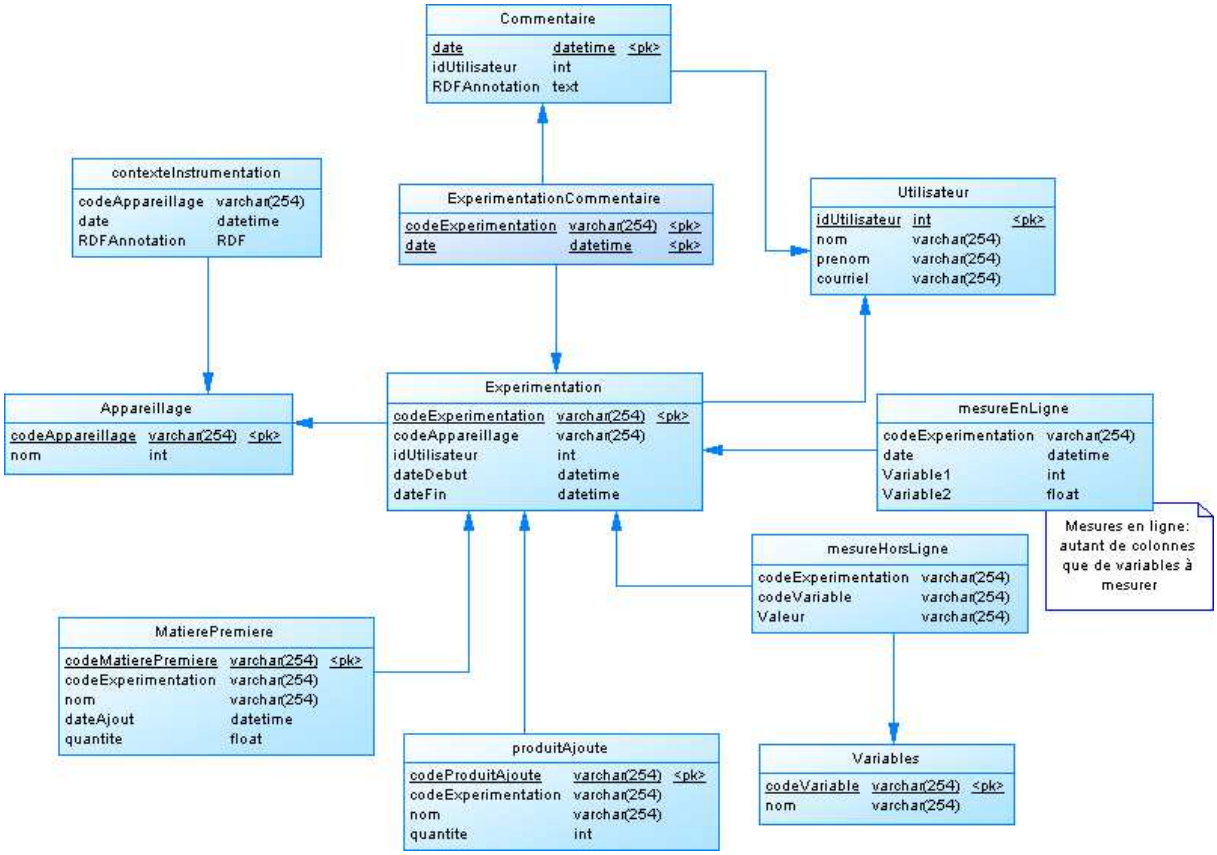


FIGURE 4.1 – Modèle logique de la base de données, les types de données figurent à titre indicatif.

Le concept central est ici l'*expérimentation*. Nous avons éliminé les concepts abstraits de procédé unitaire et de procédé en tant que principe. Cette première simplification s'explique par le fait que notre système sera utilisé sur un seul procédé.

Nous confondons le *substrat* et l'*expérimentation* au sein d'une table *Experimentation*. Cette simplification est due aux cardinalités qui relient ces deux tables et qui montrent que les concepts sont confondus dans la pratique. En effet, une expérimentation met en œuvre un substrat sur un appareillage.

Nous avons retenu une entité *MesureEnLigne* contenant un champ par variable mesurée pour des raisons de performance, comme évoqué page 22. À l'inverse, les *mesures hors ligne* s'utilisent en association avec une table décrivant les variables, ces dernières étant moins nombreuses et plus susceptibles de mesurer des variables différentes.
Les noms de ces variables devront être conformes à ceux établis dans l'ontologie.

L'entité *Commentaire* contiendra les annotations de type *événement* ainsi que les autres commentaires, au format RDF. L'interrogation de cette table nécessitera un traitement plus spécifique qu'une simple requête SQL.

Le *contexte d'instrumentation* est bien associé à un procédé physique et un nouveau n-uplet sera produit chaque fois que les installations changeront sur le procédé. Le contenu sera une annotation RDF conforme à l'ontologie.

Une entité *Utilisateur* permettra d'identifier les auteurs des *commentaires* et les initiateurs d'*expérimentations*.

Remarque S'il n'y a qu'un seul appareillage physique, on pourra faire l'économie de l'entité *Appareillage* et on reliera le *contexte d'instrumentation* avec l'entité *Experimentation*.

4.2 Le modèle sémantique

4.2.1 Les langages du web sémantique

L'implémentation des ontologies présentées dans le 3e chapitre pourrait être réalisée avec le modèle relationnel, mais la représentation de données de type graphe n'y est pas très aisée. Le schéma atteindrait une complexité importante et rendrait le système d'information difficile à maintenir et améliorer. C'est pourquoi nous proposons d'utiliser le langage d'ontologie web (OWL) pour implémenter les composantes « sémantiques » de notre modèle.

OWL est au sommet d'une pile de technologies XML. La figure 4.2 montre l'imbrication des différentes couches. Nous présenterons ces langages en insistant sur leurs particularités intéressantes pour notre étude.

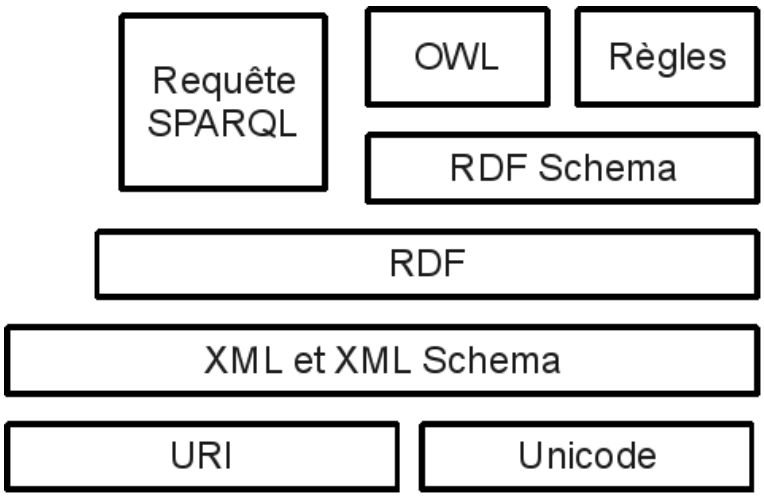


FIGURE 4.2 – Les technologies du web sémantique.

4.2.2 Unicode, URI et espace de noms

Le langage XML est un langage informatique de balisage générique, issu du *Standard Generalized Mark-up Language* (SGML). Il permet de stocker et/ou de transférer de l'information et il est réputé lisible par l'être humain. Il est qualifié d'extensible car il permet à l'utilisateur de définir ses propres balises. Ces applications sont multiples.

```
<?xml version="1.0">
<experimentation code="M10-03-23F10">
```

```
  <matierePremiere code="chardonnay_2010">
    <couleur>Blanc</couleur>
  </matierePremiere>
</experimentation>
```

Listing 4.1 – Le XML est lisible par l'être humain.

Dans cet exemple, les mots « experimentation », « matierePremiere » et « couleur » ont été définis par l'utilisateur lorsqu'il a élaboré son format.

Le codage des caractères utilise la norme Unicode [2] afin d'être aussi universel que possible en acceptant les caractères de la plupart des langues.

Le XML introduit la notion d'espace de noms qui permet de faire cohabiter plusieurs vocabulaires de balises au sein d'un même document. Pour fonctionner, c'est-à-dire pour être sans ambiguïté, les espaces de nom utilisent des identifiant unique de ressource (URI [3]).
Un URI est une chaîne de caractères servant à identifier une ressource sur un réseau sans ambiguïté, cette ressource peut être abstraite ou concrète.

Exemple:
Si nous souhaitons nous référer à une cuve de fermentation du domaine expérimental Pech-Rouge, nous pouvons écrire :
http://www.inra.fr/pechrouge/cave#cuve12
La structure de cet URI nous indique :
– *le domaine, ici* ***inra.fr****,*
– *le chemin vers la ressource* ***pechrouge/cave***
– *le nom local de la ressource, ici* ***cuve12***

L'exemple montre comment identifier une ressource physique, mais il peut s'agir des noms des balises des formats définis par un utilisateur d'XML. Ainsi, nous pourrions réécrire notre premier exemple comme suit :

```
<?xml version="1.0">
<experimentation xmlns:vin="http://www.inra.fr/pechrouge/cave"
                 xmlns:dc="http://purl.org/dc/elements/1.1/"
                 code="M10-03-23F10">
  <vin:matierePremiere code="chardonnay_2010">
    <vin:couleur>Blanc</vin:couleur>
    <dc:description>Etat sanitaire correct</dc:description>
  </vin:matierePremiere>
</vin:experimentation>
```

Listing 4.2 – Plusieurs espaces de noms dans un document XML.

Dans cet exemple, nous utilisons deux vocabulaires, l'un défini par nos soins, l'autre issu du standard Dublin-Core utilisé pour décrire une ressource. Le début du document contient les déclarations des espaces de noms ainsi qu'un préfixe qui sera utilisé pour différencier les balises.

2. Pour cela, il conviendra d'utiliser un éditeur de texte capable d'encoder les caractères dans la norme Unicode.
3. Uniform Resource Identifier

Définition d'un schéma

Le langage XML dispose d'un vocabulaire pour décrire la structure d'un document XML. Le document produit est un schéma XML qui consiste à :

- donner le nom de toutes les balises autorisées dans le document,
- formaliser l'imbrication des balises,
- indiquer les types de données autorisés à l'intérieur de ces balises.

L'intérêt des schémas XML est qu'ils permettent la validation des documents, c'est-à-dire pouvoir dire que tel document est conforme ou non au schéma qui le décrit.
De plus, les schémas XML proposent de nombreux types de données [4].

4.2.3 Le cadre de description de ressources RDF

À partir de ce socle de base qu'est XML, RDF (Resource Description Framework) est destiné à décrire de façon *formelle* les ressources du web. Le RDF utilise un modèle de graphe et est structuré comme un ensemble de triplets *sujet*, *prédicat*, *objet*.
Ces trois éléments sont identifiés par un URI.
Si par exemple nous souhaitons indiquer la transformation d'une vendange en moût, nous pouvons faire la déclaration suivante.

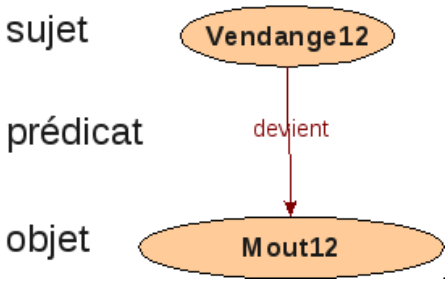


FIGURE 4.3 – Un triplet RDF.

Le RDF peut se sérialiser [5] selon divers formats et notamment sous le format de RDF/XML. Ainsi, le graphe ci-dessus peut s'exprimer comme suit :

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:process="http://www.inra.fr/ontologies/process">

  <rdf:Description
     rdf:resource="http://www.inra.fr/pechrouge/vigne#Vendange12">
    <process:devient>
      <rdf:Description
          rdf:about="http://www.inra.fr/pechrouge/cave#Mout12">
    </process:devient>
  </rdf:Description>
</rdf:RDF>
```

Listing 4.3 – Sérialisation d'une déclaration RDF.

Le terme *devient* est défini dans notre espace de noms dédié aux procédés.

4. Il existe un autre langage définition de fichier XML, les DTD. Mais ce dernier ne permet pas le typage des données.
5. S'écrire sous la forme d'une chaîne de caractères.

L'utilisation des URI conduit à écrire de longues chaînes pour faire référence aux ressources. XML offre la possibilité d'abréger un URI utilisé fréquemment, on définit pour cela une entité dans l'en-tête du document.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY vigne "http://www.inra.fr/pechrouge/vigne#">
  <!ENTITY cave "http://www.inra.fr/pechrouge/cave#">
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:process="http://www.inra.fr/ontologies/process">

  <rdf:Description rdf:resource="&vigne;Vendange12">
    <process:devient>
      <rdf:Description rdf:about="&cave;Mout12">
    </process:devient>
  </rdf:Description>
</rdf:RDF>
```

Listing 4.4 – Utilisation des entités dans une déclaration RDF.

4.2.4 Les Schémas RDF

RDF met à disposition des outils pour faire des déclarations sur des ressources, cependant il ne permet pas de définir de nouveaux vocabulaires. Lorsque nous déclarons par exemple qu'un moût de Chardonnay n°12 *devient* le vin blanc n°15, nous souhaiterions pouvoir spécifier que *moût de Chardonnay n°12* fait partie de la classe abstraite *moût*. C'est dans cet objectif qu'a été proposé le principe des schémas RDF.
Ils permettent de :

- déclarer que certaines ressources sont des classes pour d'autres ressources,
- de spécifier des propriétés de type à ces classes (en utilisant le typage des schémas XML),
- de relier des classes entre elles par des propriétés d'objets,
- de spécialiser les concepts et créer ainsi des arborescences.

Il y a une forte proximité entre le concept de classe RDF et le concept de classe dans la programmation orientée objet. On peut notamment parler d'héritage entre classes.

Nous pouvons par exemple créer des classes pour les ressources de la figure 4.3.

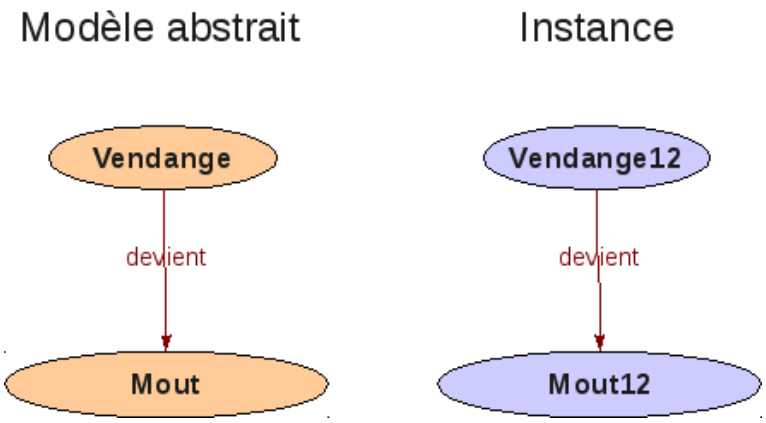


FIGURE 4.4 – Un modèle de classe et une instance.

Dans cet exemple, *Vendange* et *Moût* sont des classes, *devient* est une propriété d'objet reliant les classes *Vendange* et *Moût*. De la même manière que précédemment, on peut sérialiser ce graphe en XML :

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:process="http://www.inra.fr/ontologies/process#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <rdfs:Class rdf:ID="Vendange" />
  <rdfs:Class rdf:ID="Mout" />
  <rdf:Property rdf:ID="devient">
    <rdfs:domain rdf:resource="#Vendange"/>
    <rdfs:range rdf:resource="#Mout"/>
  </rdf:Property>
</rdf:RDF>
```

Listing 4.5 – Définition de classe et propriété d'objet en RDFS.

À cette définition de classes, on peut associer une déclaration qui constitue une instanciation de celle-ci. [6] [7]
La déclaration 4.3 peut s'écrire :

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:process="http://www.inra.fr/ontologies/process#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

   <process:Vendange rdf:about="#Vendange12">
    <process:devient>
      <process:Mout rdf:about="#Mout12" />
    </process:devient>
  </process:Vendange>
</rdf:RDF>
```

Listing 4.6 – Une déclaration conforme au modèle de la figure 4.4.

Les schémas RDF (contrairement à RDF seul) fournissent tous les outils nécessaires à la conception de vocabulaires et nous permettent d'implémenter la modélisation présentée dans le chapitre 3.

4.2.5 Le langage d'ontologie OWL

OWL (Ontology Web Language) propose davantage de raffinements dans la construction d'un vocabulaire que RDFS. L'objectif de ces ajouts est de permettre des mécanismes d'inférence sur les instances ou annotations. Il est fondé sur les logiques de description [8].

6. Ces instances constitueront des annotations pour l'expérimentation à décrire.
7. Les annotations peuvent figurer dans le même fichier que les classes ou bien dans un fichier différent.
8. Les logiques de description, aussi appelées logiques descriptives, (LDs) sont une famille de langages de représentation de connaissance qui peuvent être utilisés pour représenter la connaissance terminologique d'un domaine d'application d'une manière formelle et structurée. Le nom de logique de description se rapporte, d'une part à la description de concepts utilisée pour décrire un domaine et d'autre part à la sémantique basée sur la logique qui peut être donnée par une transcription en logique des prédicats du premier ordre.

L'inférence recouvre plusieurs mécanismes de déduction de connaissances à partir d'un corpus de connaissances pré-existant. L'inférence est rendue possible grâce à la définition d'axiomes de base. Dans notre système, ce mécanisme facilitera l'interrogation des annotations [9].

Premier type d'inférence : la subsomption

La subsomption désigne une relation hiérarchique entre concepts, elle est proche de la relation « est impliqué par » en logique classique. En OWL, c'est la relation `owl:subClassOf` qui nous permet la mise en œuvre de la subsomption.

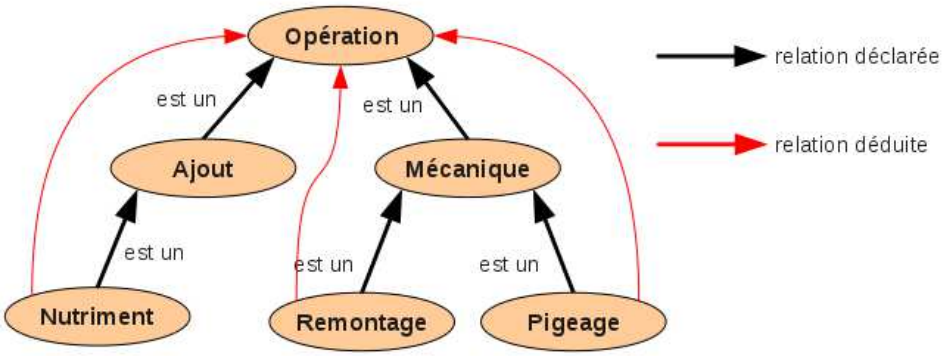


FIGURE 4.5 – *Mécanique* est une sorte d'opération, *Pigeage* est une sorte d'opération *Mécanique*, donc *Pigeage* est une sorte d'*Opération*.

Au chapitre suivant, nous montrerons que l'on peut récupérer dans l'ensemble des résultats d'une requête les instances des sous-classes de la classe spécifiée.

Inférences en utilisant les caractéristiques des propriétés d'objet

RDFS introduit les propriétés d'objets reliant deux classes entre elles (ou leurs instances). OWL permet de préciser certaines caractéristiques de ces propriétés fournissant des améliorations pour les mécanismes de raisonnement. Nous présentons uniquement celles dont nous avons une utilité dans notre système : la transitivité et la propriété inverse.

La transitivité Soit la propriété P et les classes x, y, z. On note $P(x, y)$ la classe x est reliée à la classe y par la propriété d'objet P. Si P est transitive alors $P(x, y)$ et $P(y, z)$ impliquent $P(x, z)$.

9. Nous n'utiliserons pas l'ensemble des possibilités offertes par OWL et donc nous ne présentons que celles utilisées.

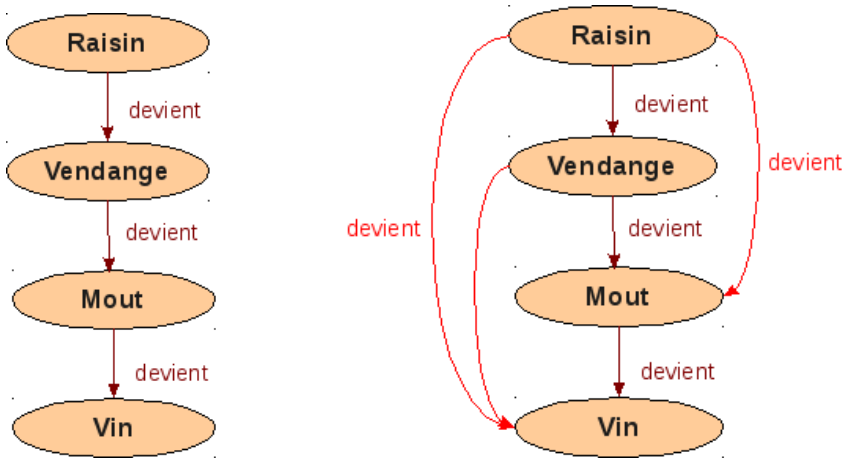


FIGURE 4.6 – Un exemple de transitivité.

L'inverse Si la propriété P_1 est inverse de P_2, alors $P_1(x, y)$ implique $P_2(y, x)$ et vice versa.

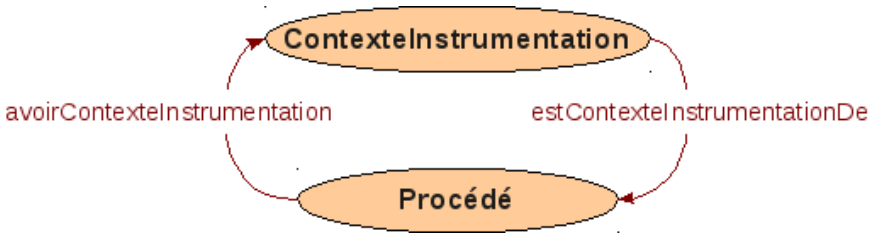


FIGURE 4.7 – La propriété d'objet *avoirContexteInstrumentation* est l'inverse de la propriété d'objet *estContexteInstrumentationDe*.

Cette caractéristique est très pratique, en effet nous pouvons définir *avoirContexteInstrumentation* comme une propriété reliant des objets de la classe *Procédé* à des objets de la classe *ContexteInstrumentation*. Puis nous déclarons *estContexteInstrumentationDe* inverse de *avoirContexteInstrumentation* et un moteur d'inférence déduira le domaine et la portée de la propriété inverse.

```
<owl:ObjectProperty rdf:about="#hasInstrumentationContext">
  <rdfs:range rdf:resource="#InstrumentationContext"/>
  <rdfs:domain rdf:resource="#Process"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#isInstrumentationContextFor">
  <owl:inverseOf rdf:resource="#hasInstrumentationContext"/>
</owl:ObjectProperty>
```

Listing 4.7 – Une déclaration de propriété inverse.

En outre les propriétés inverses facilitent le parcours dans les graphes RDF.

4.2.6 L'extensibilité de OWL

OWL dispose d'un mécanisme d'importation permettant à une ontologie d'être construite à partir d'autres ontologies. Cette propriété est très intéressante car elle va nous permettre de séparer notre ontologie générique des procédés, des ontologies spécialisées sur un procédé

particulier. Le listing suivant montre un fragment de l'ontologie « générique » des bioprocédés (présentée à la section suivante) [10]. Il déclare une partie de la hiérarchie des événements, à savoir qu'un *Ajout* est une *Opération*, elle-même un *Événement*.

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
        [...]
]>

<rdf:RDF xmlns="http://www.inra.fr/cafe/2009/process#"
     xml:base="http://www.inra.fr/cafe/2009/process"
     xmlns:process="http://www.inra.fr/cafe/2009/process#"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     [...]

    <owl:Ontology rdf:about="">
        <rdfs:label>Ontologie des éébioprocds</rdfs:label>
        <dc:creator>INRA</dc:creator>
        <dc:date>2009-08-21</dc:date>
        <owl:versionInfo>1.0</owl:versionInfo>
    </owl:Ontology>

    [...]
    <owl:Class rdf:about="#Event">
        <rdfs:label>éEvnement</rdfs:label>
    </owl:Class>

        <owl:Class rdf:about="#Operation">
        <rdfs:label>Operation</rdfs:label>
        <rdfs:subClassOf rdf:resource="#Event"/>
    </owl:Class>

    <owl:Class rdf:about="#Addition">
        <rdfs:label>Ajout</rdfs:label>
        <rdfs:subClassOf rdf:resource="#Operation"/>
    </owl:Class>

        [...]
</rdf:RDF>
```

Listing 4.8 – Fragment de l'ontologie des bioprocédés générique.

Nous pouvons spécialiser cette ontologie dans le domaine de la fermentation alcoolique :

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
        [...]
```

10. Pour faciliter la lecture, nous avons abrégé certaines lignes.

```
]>

<rdf:RDF xmlns="http://www.inra.fr/cafe/2009/process#"
   xml:base="http://www.inra.fr/cafe/2009/process"
   xmlns:process="http://www.inra.fr/cafe/2009/process#"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   [...]

   <owl:Ontology rdf:about="">
     <owl:imports rdf:resource="http://www.inra.fr/cafe/process.owl"/>
   </owl:Ontology>

   [...]
   <process:Addition rdf:about="#SO2Adding">
     <rdf:type rdf:resource="&owl;Thing"/>
     <rdfs:label>Ajout de dioxyde de souffre</rdfs:label>
   </process:Addition>

   [...]
</rdf:RDF>
```

Listing 4.9 – Fragment de l'ontologie du procédé de fermentation.

Ce listing déclare la ressource *#SO2Adding* comme un individu de la classe *Addition* définie dans l'ontologie des procédés. La référence à l'ontologie des procédés est une URL (http ://www.inra.fr/cafe/2009/process.owl), ce qui signifie que son fichier de déclaration est accessible en ligne.

Il faut distinguer le mécanisme d'importation et celui de l'espace de nommage. Les déclarations d'espace de nommage représentent un moyen *commode* d'appeler des noms définis dans d'autres ontologies OWL. L'importation *inclut* toutes les assertions de l'ontologie cible.

4.2.7 Le langage d'interrogation SPARQL

Parcourir l'ontologie avec les outils classiques [11] que l'on rencontre pour XML ne permettrait pas de profiter des possibilités offertes par l'inférence. Il faut donc se tourner vers des programmes spécifiques. Il peut s'agir d'API comme Jena ou Corese[3], mais RDF possède un outil dédié faisant l'objet d'une recommandation du W3C : le langage SPARQL.

SPARQL [12] est un langage de requête pour RDF. Il s'appuie sur le modèle de graphe de RDF et exprime les requêtes sous forme de triplets. Sa syntaxe et ses fonctionnalités ressemblent au SQL.
L'exemple ci-dessous renvoie toutes les *Operations* de la base de déclaration RDF ainsi que leur traduction en français si elle est disponible.

```
PREFIX process: <http://www.inra.fr/ontologies/process#">
SELECT ?operation ?label
WHERE {
  ?operation rdf:type process:Operation.
```

11. On peut citer le langage XPath ou les analyseurs syntaxiques XML présents dans la plupart des langages
12. SPARQL est l'acronyme de **S**PARQL **P**rotocol and **R**DF **Q**uery **L**anguage.

```
  OPTIONAL {?operation rdfs:label ?label}
  FILTER ( lang(?label) = 'fr' )
}
```

Listing 4.10 – Une requête SPARQL.

La requête est composée de :

- une déclaration d'espace de nommage avec un préfixe pour raccourcir les requêtes,
- une clause SELECT qui permet de sélectionner l'ensemble des n-uplets correspondant aux contraintes de la clause WHERE,
- une clause WHERE indiquant les projections à réaliser sur l'ontologie.

Le format des résultats dépend du programme utilisé pour réaliser la requête, le W3C propose cependant une recommandation de format, le SPARQL Query Results XML Format.

4.2.8 Construction de l'ontologie

Dôtés de cet arsenal technologique, nous pouvons construire l'ontologie des procédés. Comme évoqué plus haut, nous la diviserons en deux parties :

- une ontologie générique avec l'espace de noms suivant :
  `http://www.inra.fr/cafe/2009/process#`
  elle contiendra la définition de toutes les classes, les propriétés d'objets et de données,
- une ontologie spécifique destinée à correspondre à un bioprocédé particulier. Elle contiendra des individus « membres » de l'ontologie des procédés ainsi que des définitions de concepts spécifiques au procédé considéré. Par exemple pour le procédé de fermentation alcoolique tel qu'étudié au laboratoire SPO [13], nous pouvons déclarer un nouvel espace de noms :
  `http://www.inra.fr/cafe/2010/spo#`

À ces deux ontologies correspondront deux fichiers de définition : `process.owl` et `spo.owl`. Ces deux fichiers devront être accessibles par les applications du système d'information.

La syntaxe du langage OWL étant particulièrement pointue, nous pouvons nous aider d'un logiciel de conception d'ontologie. L'université de Standford a développé « protégé », un logiciel de conception d'ontologie OWL sous licence libre [14].

13. Sciences pour l'œnologie.
14. `http://protege.stanford.edu/`

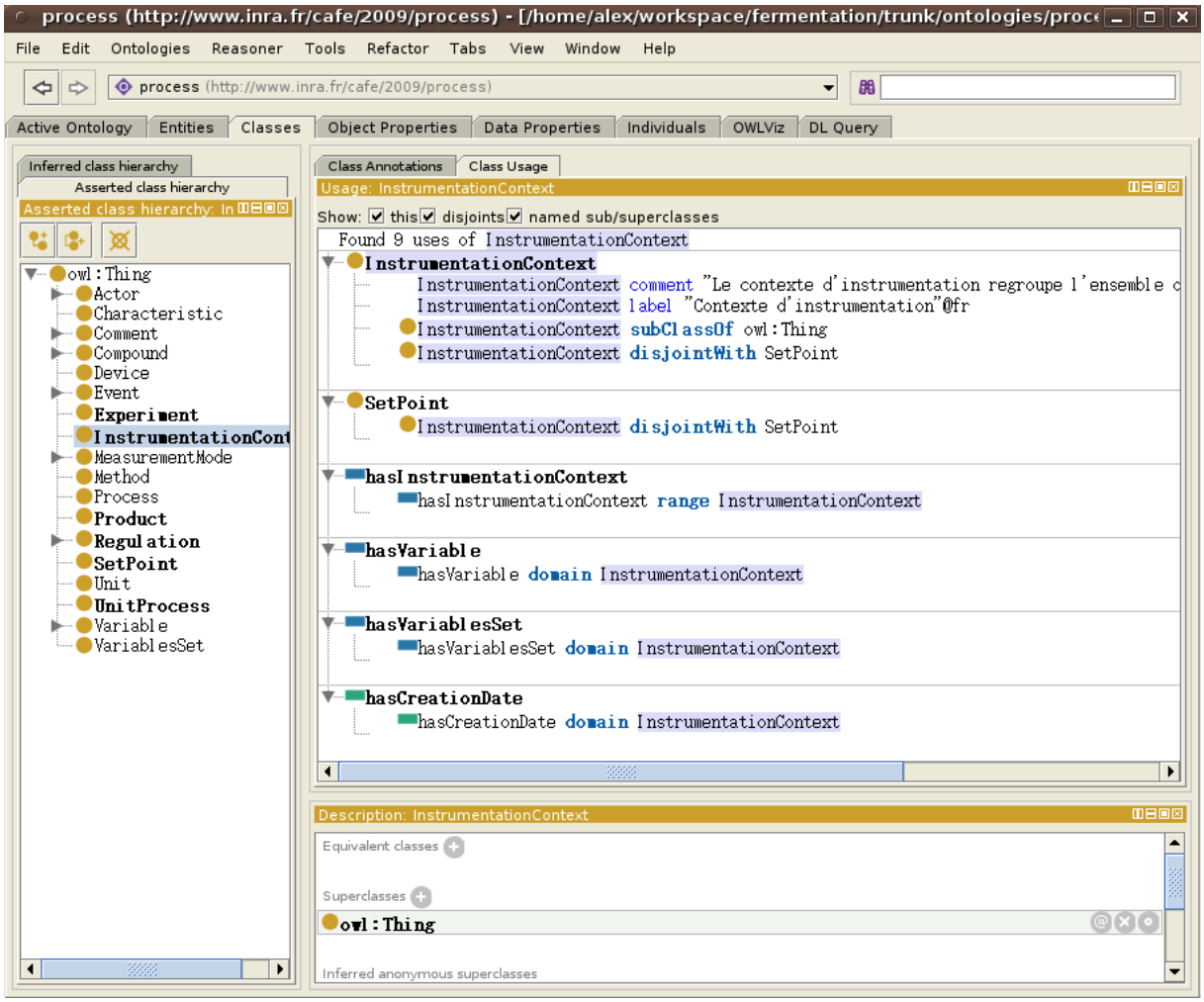


FIGURE 4.8 – La modélisation sous « protégé ».

L'ontologie des bioprocédés

Cette ontologie reprend en définitive l'ensemble des concepts présentés dans les « ajouts sémantiques » du chapitre 3. La représentation en OWL diffère légèrement de la conception UML. À titre d'exemple, nous reproduisons le graphe décrivant la relation entre les variables, les unités de mesure et les capteurs.
Les classes sont représentées par des ovales orange, les propriétés d'objet par des flèches rouges et les propriétés de données par des rectangles verts. Les relations de spécialisation/généralisation sont représentées par des flèches noires.

Note : Le diagramme est en anglais, la langue retenue par les équipes d'experts avec lesquelles l'ontologie a été construite. La traduction pour ce diagramme ne pose pas de problèmes.

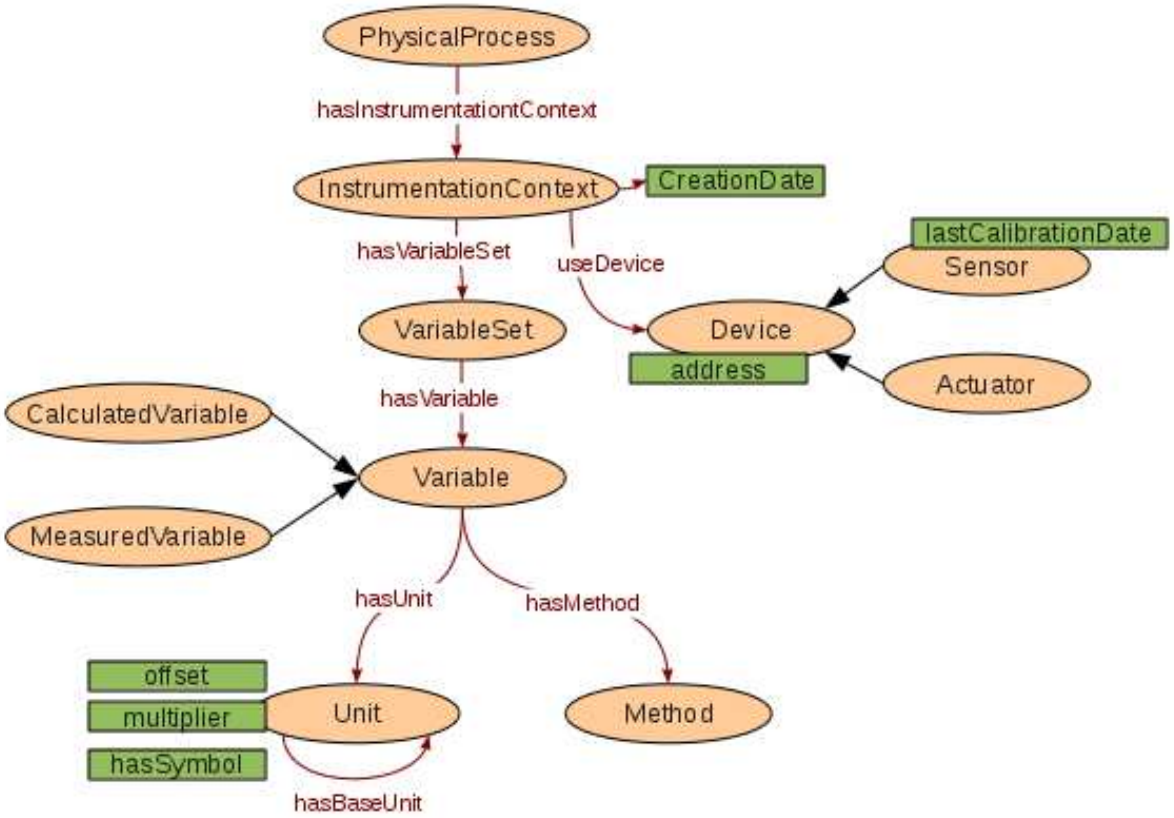


FIGURE 4.9 – Les classes OWL associées au contexte d'instrumentation.

L'ontologie de la fermentation alcoolique

On peut voir cette ontologie comme une instance de l'ontologie des procédés. Elle contient des déclarations d'individus mais également des classes qui étendent l'ontologie des procédés. Dans sa définition elle inclut l'ontologie des procédés. On y trouvera en tant qu'individu au sens OWL :

- l'ensemble des variables mesurables sur le procédé de fermentation,
- l'ensemble des unités de mesure associées (avec leur symbole),
- l'ensemble des méthodes d'acquisition,
- une liste des composés mesurables sur le procédé,

ainsi que des classes spécifiques :

- une hiérarchie des évènements spécifiques (pannes, opérations),
- une classification des commentaires,
- une classification des thèmes de recherche.

L'exemple suivant montre la variable *vitesse de dégagement de* CO_2 et les unités associées.

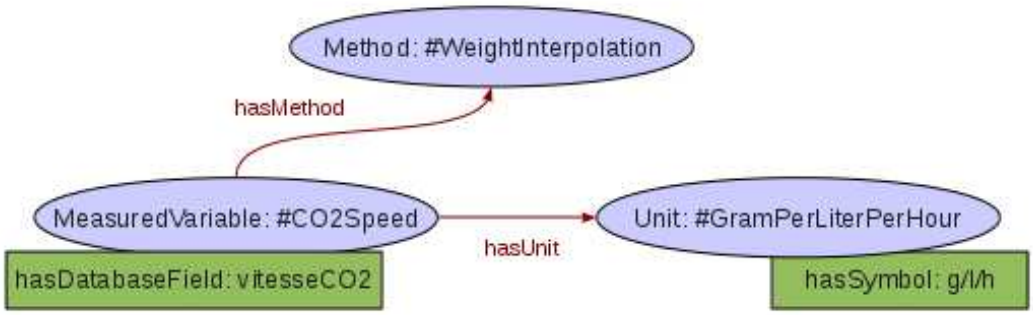


FIGURE 4.10 – La variable vitesse de dégagement de CO_2 a pour unité le g/l/h et utilise une méthode d'acquisition par calcul depuis le poids.

Chapitre 5

Implantation

Dans cette partie, nous présentons une proposition d'implémentation des idées exposées jusqu'à maintenant.

5.1 Les choix technologiques

5.1.1 Une interface web

Dans le développement d'application, la tendance actuelle est de se tourner vers la construction d'applications dans un contexte web, c'est-à-dire par l'intermédiaire d'un navigateur internet.
Ce choix repose sur les arguments suivants :

- une indépendance vis-à-vis de la plateforme,
- un mélange de technologies dont certaines sont facilement appréhendables (comme le HTML) par un non-spécialiste,
- la combinaison des technologies du web, HTML, CSS, Javascript permet de réaliser des interfaces riches [1],
- un environnement adapté à une architecture orientée service,
- les langages de script exécutables côté serveur ont atteint une maturité suffisante pour produire des applications robustes.

5.2 Architecture

L'architecture de notre système d'information peut être divisé en trois sous-systèmes :

- *sous-système synchronisation* : système de synchronisation ou d'acquisition des données, en charge d'assurer la migration des informations depuis le procédé vers la base relationnelle,
- *sous-système consultation* : une interface utilisateur pour consulter les données,
- *sous-système interrogation d'ontologie* : un service pour réaliser des requêtes SPARQL sur les ontologies.

1. Le terme Rich Internet Application a été introduit dans une publication de Macromedia en mars 2002. Il désigne une application web qui offre des caractéristiques similaires aux logiciels traditionnels, notamment dans leur dimension interactive et leur vitesse d'exécution.

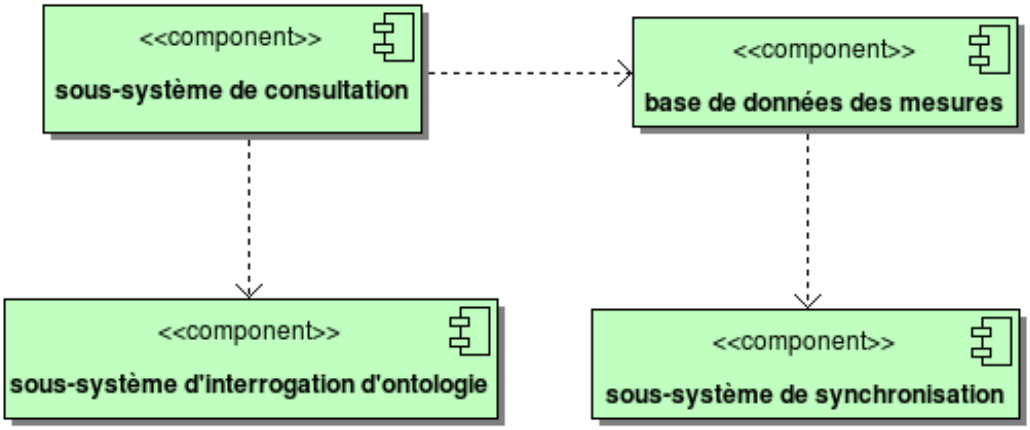


FIGURE 5.1 – Vue globale de l'architecture.

5.2.1 Sous-système de synchronisation

Cette partie du SI est la plus hétérogène, nous avons détaillé dans l'état de l'art les différentes méthodes de synchronisation. Il n'est pas dans les objectifs de ce mémoire de proposer des solutions à ce problème. Remarquons que sur le terrain, différentes méthodes cohabitent et dépendent en définitive des capteurs/transmetteurs utilisés.

Lorsque les données sont disponibles sous un format texte, on pourra se tourner vers un logiciel d'extraction (ETL [2]) qui facilitent la mise en place d'une synchronisation souvent laborieuse. Citons par exemple Talend [3], un ETL libre.

5.2.2 Sous-système de consultation

C'est le sous-système le plus complexe et qui demande le temps de développement le plus grand. Une application web peut être implémentée sous différentes plateformes.

Environnement de script

L'environnement de script le plus répandu sur l'internet est PHP. Plusieurs autres raisons concourent à ce choix :

- PHP est un logiciel libre, donc gratuit,
- PHP est multiplateforme Microsoft Windows, MacOS X, UNIX,
- une forte dynamique entoure son activité, de multiples communautés d'utilisateurs, etc.
- PHP est un langage simple à appréhender et ce paramètre est très important dans un contexte de recherche car il est extrêmement fréquent que les développements informatiques soient dévolus à des stagiaires faute, de ressources humaines suffisantes,
- PHP est aujourd'hui assez abouti et permet un style de programmation orienté objet,
- il s'interface avec la plupart des SGBD du marché, libres ou commerciaux.

2. **E**xtract **T**ransform **L**oad, une technologie informatique permettant d'effectuer des synchronisations d'information d'une base de données vers une autre.

3. `http://www.talend.com/`

Serveurs Web et SGBD

Le choix du serveur Web dépend surtout du système d'exploitation hôte, les cas d'étude observés lors de la phase de recherche montrent que le serveur Web *Apache* est quasiment présent partout, mais rien n'empêche d'utiliser un serveur IIS [4] de Microsoft.

Le SGBD a peu d'importance, il devra néanmoins disposer des caractéristiques suivantes :
- être accessible par TCP/IP, c'est-à-dire qu'on exclut les programmes qui n'ont pas de fonction « serveur » comme Microsoft Access ou Filemaker Pro (en version de base),
- respecter autant que faire se peut le standard SQL, à ce propos les programmes s'efforceront de ne pas utiliser les dialectes SQL non standards.

L'implantation sur le procédé de fermentation utilise le serveur MySQL.

Fonctionnement Les pages HTML seront générées par des scripts PHP. La description de la solution technique de développement d'une application web sort du cadre de ce mémoire. Cependant le lecteur intéressé pourra se référer à l'annexe E pour une description du système déployé au laboratoire SPO.

Le schéma suivant montre l'interaction entre le sous-système de consultation, la base de données et le sous-système d'interrogation d'ontologies.

4. Internet Information Service.

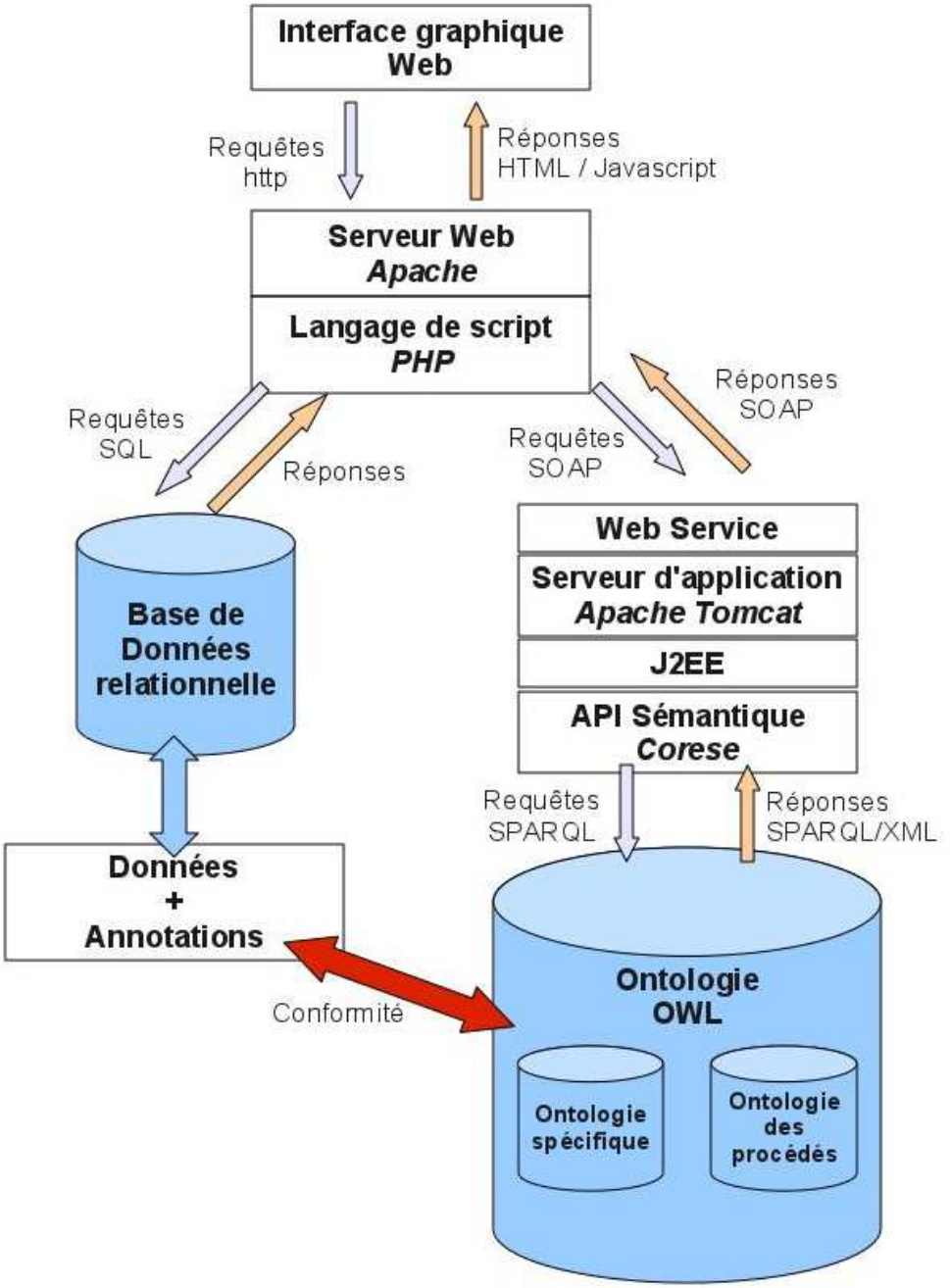


FIGURE 5.2 – Architecture globale du système, les flèches indiquent les points de communication entre les sous-systèmes.

5.2.3 Sous-système d'interrogation d'ontologies

Pour la mise en œuvre des ontologies, plusieurs outils sont disponibles. Nous avons étudié plusieurs solutions mettant en jeu les outils « libres » les plus répandus : Pellet [5], l'API Corese et l'API RDF.

5. `http://clarkparsia.com/pellet`

Solution avec Pellet

Pellet est un programme autonome permettant de réaliser des requêtes SPARQL sur une ontolologie OWL. Il permet aussi de vérifier la consistance d'une ontologie et des annotations associées [6].
Pellet s'utilise en ligne de commande. Les résultats sont au format SPARQL/XML, texte ou JSON [7].
Ci-dessous, un exemple de requête sur notre ontologie des procédés.

Exemple:
La requête essaie d'obtenir toutes les instances de type Variable, défini dans l'espace de noms `http://www.inra.fr/cafe/2009/process\#` *ainsi que les unités associées et le champ de la base de données correspondant. Optionnellement, c'est-à-dire sans faire échouer la requête en cas de non réponse, on demande aussi le symbole de l'unité et la traduction en français de la variable.*

```
PREFIX process: <http://www.inra.fr/cafe/2009/process#>
PREFIX spo: <http://www.inra.fr/cafe/2010/spo#>

SELECT ?var ?unit ?symb  ?label ?champ
WHERE
{
 ?var rdf:type process:Variable.
 ?var   process:hasUnit ?unit.
 ?var   process:hasDatabaseField ?champ.
 OPTIONAL { ?unit  process:hasSymbol ?symb}
 OPTIONAL {?var rdfs:label ?label}
 FILTER ( lang(?label) = 'fr')
}
```

Listing 5.1 – La requête SPARQL, fichier query.sparql.

Le format de sortie (–output-format) choisi est `tabular` pour faciliter la lecture.

```
./pellet.sh query -q query.sparql --output-format tabular spo.owl
Query Results (9 answers):
var            | unit                | symb    | label
=================================================================
Weight         | Gram                | "g"     | "Poids"@fr
CO2Speed5      | GramPerLitrePerHour | "g/l/h" | "Vitesse CO2 (5)"@fr
CO2Speed11     | GramPerLitrePerHour | "g/l/h" | "Vitesse CO2 (11)"@fr
HeatDuration   | Minute              | "mn"    | "Durée chaud"@fr
Time           | Hour                | "H"     | "Temps"@fr
ColdDuration   | Minute              | "mn"    | "Durée froid"@fr
CO2Speed       | GramPerLitrePerHour | "g/l/h" | "vitesse CO2"@fr
Temperature    | DegreeCelsius       | "°C"    | "Température"@fr
AccumulatedCO2 | GramPerLitre        | "g/l"   | "Cumul de CO2"@fr
```

Listing 5.2 – La commande Pellet et son résultat.

6. Vérification de la syntaxe et de la cohérence sémantique.
7. Javascript Object Notation, est un format de données textuel, générique, dérivé de la notation des objets du langage ECMAScript. Il permet de représenter de l'information structurée.

On notera que Pellet gère la clause `owl:imports`, ce qui n'est pas forcément le cas de tous les moteurs d'inférence.

Dans un environnement PHP, on peut utiliser la commande `exec` pour faire cet appel.

```
$command = 'sh pellet.sh query -q ontologies/query.sparql'
                    .' --output-format XML ontologies/spo.owl';

// exec(...) réalise l'appel système et renvoie les résultats dans un tableau
// une ligne par entrée dans le tableau
exec($command, $output);

// On transforme le tableau en chaîne
$sparqlXMLResult = implode('', $output);
```

Listing 5.3 – Appel à Pellet depuis PHP.

Cette méthode fait vite apparaître un problème de performance puisque le temps de chargement de Pellet prend 2 secondes à lui seul et la résolution de la requête 1 à 2 secondes supplémentaires.

Solution avec Corese

La solution avec Corese fonctionne de la même manière qu'avec Pellet. Elle est légèrement plus performante, 1 seconde de mieux, mais tout de même insuffisant pour un environnement web.

Solution avec un service Web

Le problème des performances pour effectuer une requête SPARQL a deux origines :
- le temps de chargement du programme puis des fichiers requête, ontologies et annotations (Pellet et Corese sont écrits en Java, ainsi leur appel nécessite un chargement plus important),
- l'exécution de la requête elle-même.

Cette dernière n'a pas beaucoup d'impact sur des graphes RDF relativement modestes. Pour résoudre le problème du temps de chargement du raisonneur, il suffit que le programme et les ontologies soient déjà en mémoire lors de l'appel.
Ceci peut être résolu par l'écriture d'un programme exécuté en tant que service, c'est-à-dire en tant que processus du système en écoute (par exemple **socket** sur un port TCP). Mettre en place un tel programme est assez coûteux en temps de développement. Nous montrons comment on peut élégamment résoudre cette difficulté par la mise en place d'un Web Service.

Mise en place du Web Service

Ainsi, nous proposons de construire un service Web utilisant une API pour réaliser des requêtes SPARQL. Dans un environnement Java, nous disposons d'une implémentation libre des spécifications des Web Services avec Tomcat et Axis, le lecteur trouvera en annexe C la procédure d'installation de ces composants. Le Web Service utilisera le protocole SOAP [8] (**S**imple **O**bject

8. `http://www.w3.org/standards/techs/soap`

Access **P**rotocol).
Ce dernier sera relativement simple, il comportera une méthode principale prenant en paramètre une requête SPARQL et une URL vers une ontologie, et renverra le résultat au format SPARQL/XML.

Dans un environnement Tomcat/Axis, on écrit le service en Java, et on construit une archive `.aar` que l'on déploie sur le serveur.
Toutes les commandes intégrées dans le constructeur du Web Service seront exécutées lors du déploiement.

Le fragment de code suivant montre une version simplifiée du Web Service utilisant l'API Corese.

```
package fr.inra.spo;

import fr.inria.acacia.corese.api.EngineFactory;
import fr.inria.acacia.corese.api.IEngine;
import fr.inria.acacia.corese.api.IResults;
import fr.inria.acacia.corese.exceptions.EngineException;

import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * Version du service avec Axis2 pour Tomcat6
 *
 * @copyright 2010 - INRA UMR MISTEA,SPO - IFV
 * @author Alexandre Granier
 */
public class CoreseWs {

  // URIs chargés par défaut.
  protected static final String DEFAULT_URI =
                        "http:www.inra.fr/cafe/2009/process.owl";

  // Séparateur d'URIs par défaut. Recommandé: ';'
  protected static final String DEFAULT_SEPARATOR = ";";

  // Classes de CORESE
  protected EngineFactory ef;
  protected IEngine engine;

  /**
   * Constructeur, appelé au moment du déploiement. Initialise les
   * objets en mémoire, active le cache.
   */
  public CoreseWs() {
      ef = new EngineFactory();
      // Création du moteur CORESE
      engine = ef.newInstance();
      System.out.println("CoreseWs initialized");
      try {
```

```
        System.out.println("loading uri: ["+ CoreseWs.DEFAULT_URI +"]");
        engine.load(CoreseWs.DEFAULT_URI);
      } catch (EngineException ex) {
         Logger.getLogger(CoreseWs.class.getName()).
                                log(Level.SEVERE, null, ex);
      }
  }

/**
   * query : réalise une requête SPARQL sur l'ontologie spécifiée en
   *         paramètre
   * @param q: la requête SPARQL a exécuter
   * @param uris: la liste des URIs depuis lesquels charger les
   *              fichiers d'ontologies
   *
   * @return un message d'erreur si la requête a échoué, sinon le
   *         résultat au format XML
   */
  public String query(String q, String uris) {

    // URIs par défaut si non spécifiés
    if (uris == null) return "CORESE missing URI";

    String result = null;

    // exécution de la requête avec CORESE
    // Découpage des URIs selon le séparateur par défaut
    String[] uriArray = uris.split(CoreseWs.DEFAULT_SEPARATOR);

    // Construction du graphe
    try {
      // Chargement des URIs un par un
      for (int i=0; i< uriArray.length; ++i) {
        System.out.println("loading uri: [" + uriArray[i] + "]");
        engine.load(uriArray[i]);
      }
    } catch (EngineException ex) {
       Logger.getLogger(CoreseWs.class.getName()).
                              log(Level.SEVERE, null, ex);
    }

    // Exécution de la requête Sparql
    try {
      IResults res = engine.SPARQLQuery(q);
      if (res != null) {
         result = res.toSPARQLResult();
      }
    } catch (EngineException ex) {
       Logger.getLogger(CoreseWs.class.getName())
                              .log(Level.SEVERE, null, ex);
       result = "CORESE query failed (see exceptions)";
    }
    // Renvoi du résultat
    return result;
```

```
  }
}
```

Listing 5.4 – Code source simplifié du web service

Consommation du Web Service

Le protocole SOAP étant un standard ouvert, nous pouvons utiliser le Web Service avec n'importe quel langage proposant une API SOAP. Nous proposons ci-dessous un exemple en PHP.

```
<?php

// Instanciation d'un objet SoapClient
// http://fr2.php.net/manual/fr/book.soap.php
try {
  $client = new SoapClient("http://localhost:8080/serv/CoreseWs?wsdl");

} catch (SoapFault $sf) {
  echo $sf->getMessage();
}

// La requête SPARQL
$sparqlQuery =
'PREFIX process: <http://www.inra.fr/cafe/2009/process#>
PREFIX spo: <http://www.inra.fr/cafe/2010/spo#>
SELECT ?var ?unit ?symb ?label
WHERE
{
  ?var rdf:type process:Variable .
  ?var   process:hasUnit ?unit .
  OPTIONAL { ?unit process:hasSymbol ?symb }
  OPTIONAL {?var rdfs:label ?label }
  FILTER ( lang(?label ) = "fr")
}';


try {
  $sparqlXMLResult = $client->query($sparqlQuery,
                'http://localhost/ontologies/spo.owl');
} catch (SoapFault $sf) {
echo $sf->getMessage();
}
// Affichage du résultat XML
var_dump ($sparqlXMLResult);
?>
```

La solution avec un Web Service permet effectivement de résoudre le problème de performance puisque les requêtes SPARQL sont traitées en moins de 0,02 secondes. Il conviendra de noter cependant que l'accroissement de la taille des graphes RDF (ou des annotations) risque de reposer tôt ou tard des problèmes de performances.

5.3 Fonctionnement

Cette section décrit l'articulation entre la base de données relationnelle, la base de connaissances et l'interface de consultation. Nous avons vu précédemment que l'appel à la base de connaissance pouvait se réaliser par l'intermédiaire d'un service Web. Nous n'avons cependant pas indiqué comment *nourrir* le service avec les annotations sur lesquelles les requêtes auront lieu.

Le schéma 5.2 indique les flux de données entre la base de données, la base de connaissances et l'interface utilisateur.

Les annotations, selon leur volume, sont stockées dans la base de données de différentes manières. Ou bien elles sont nombreuses et constituent un ensemble d'enregistrement interrogeable à part entière, à l'instar des commentaires ou des descriptions d'expérimentation, ou bien elles sont peu nombreuses et se rapprochent davantage d'informations de configuration du procédé comme les contextes d'instrumentation. Nous détaillons ici ces deux cas de figure.

5.3.1 Annotations nombreuses

Dans le premier cas, l'utilisation de ces annotations impliquera une construction d'une annotation globale, concaténation de l'ensemble des annotations, avant soumission au Web Service. Dans notre modèle, les annotations concernées sont :

- les événements,
- les commentaires,
- les descriptions d'expérimentations.

Nous indiquons ici la méthode avec les commentaires. Conformément au modèle 4.1, la table recensant les annotations se présente sous cette forme :

commentaires		
date	id_utilisateur	RDFAnnotation

2007-09-20	Christian	`<process:Sampling rdf:about="#prelevement2007-09-20 09:04:17">`
		`  <process:occursIn rdf:resource="#M07-09-05F24" />`
		`  <dc:description>`
		`  La recroisssance à 160h est due à une contamination`
		`  </dc:description>`
		`  <dc:author>Christian</dc:author>`
		`  <dc:date>2007-09-20 09:04:17</dc:date>`
		`</process:Sampling>`

2007-04-06	Brigitte	`<process:Operation rdf:about="#maintenance2007-04-06">`
		`  <process:occursIn rdf:resource="#M07-04-05F23" />`
		`  <dc:description>`
		`  Problème de température sur les 5 postes F11 à F15 au`
		`  même moment pendant 16 points x(20 minutes)`
		`  inscription de inf sur fichier source.`
		`  </dc:description>`
		`  <dc:author>Brigitte</dc:author>`
		`  <dc:date>2007-04-06</dc:date>`
		`</process:Operation>`

...	...	...

Dans cette proposition, les annotations sont stockées sans les en-têtes XML indispensables à tout analyseur syntaxique XML. Ce sera à la charge du programme de reconstituer ces en-têtes. Le diagramme d'activité suivant montre les différentes étapes de cette reconstruction.

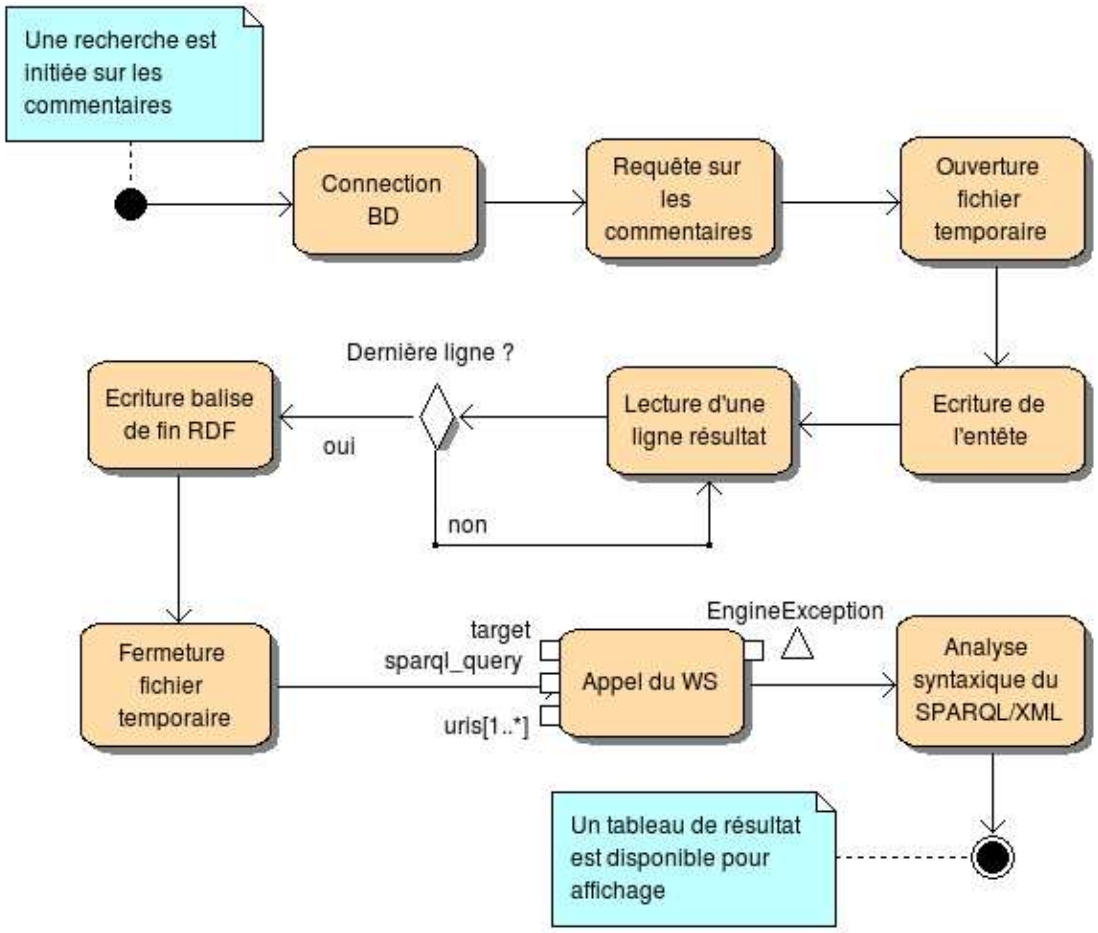


FIGURE 5.3 – Diagramme d'activité d'une recherche de commentaires.

Lorsque la base de faits (les annotations) deviendra importante, il deviendra coûteux d'utiliser cette méthode, notamment à cause de la phase d'écriture du fichier. On pourra se tourner alors vers un système de cache où le fichier sera écrit lors de l'ajout d'une nouvelle annotation, ou bien on pourra maintenir les annotations en mémoire centrale.

5.3.2 Annotations peu nombreuses

Dans ce cas de figure, il n'est pas besoin de concaténer les annotations, une requête SQL simple suffira. L'annotation est dans ce cas-là complète, avec tous les en-têtes nécessaires et notamment la directive `owl:import`. Le modèle logique de la base de données 4.1 nous précise où stocker les annotations.

Contexte d'intrumentation

La table *contexteInstrumentation* contient l'annotation. Dans l'exemple ci-dessous, on déclare un contexte d'instrumentation pour le procédé physique `F01`, il contient un jeu de variables nommé `onlineMeasurement` et regroupant cinq variables. L'attribut `rdf:resource` indique que ces variables sont définies par ailleurs, en l'occurence dans le fichier `spo.owl`.

contexteInstrumentation		
date	codeProcede	RDFAnnotation
2010-01-15	F01	<?xml version="1.0"?> <!DOCTYPE rdf:RDF [<!ENTITY spo "http://www.inra.fr/cafe/2010/spo#" > <!ENTITY process "http://www.inra.fr/cafe/2009/process#" > <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" > <!ENTITY owl "http://www.w3.org/2002/07/owl#" >]> <rdf:RDF xmlns="http://www.inra.fr/cafe/2010/spo#" xml:base="http://www.inra.fr/cafe/2010/spo" xmlns:process="http://www.inra.fr/cafe/2009/process#" xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:spo="http://www.inra.fr/cafe/2010/spo#"> <owl:Ontology rdf:about=""> <owl:imports rdf:resource="http://localhost/ontologies/spo.owl"/> </owl:Ontology> <process:InstrumentationContext rdf:about="#05-01-01F01"> <process:isInstrumentationContextFor rdf:resource="&spo;F01" /> <process:hasVariablesSet> <process:VariablesSet rdf:about="#onlineMeasurement"> <process:hasVariable rdf:resource="#CO2Speed5" /> <process:hasVariable rdf:resource="#CO2Speed11" /> <process:hasVariable rdf:resource="#AccumulatedCO2" /> <process:hasVariable rdf:resource="#Temperature" /> <process:hasVariable rdf:resource="#Weight" /> </process:VariablesSet> </process:hasVariablesSet> </process:InstrumentationContext> </rdf:RDF>

L'utilisation du Web Service se fait au travers des étapes suivantes :

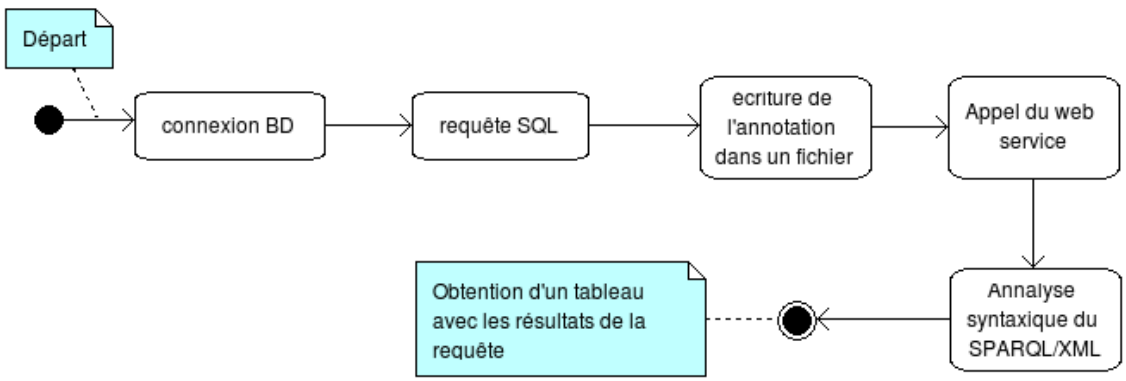


FIGURE 5.4 – Récupération des informations d'un contexte d'instrumentation.

Nous utiliserons les informations de contexte d'instrumentation pour d'une part conserver la mémoire des installations techniques et d'autre part pour générer les interfaces pour visualiser les données (voir le dernier chapitre).

Chapitre 6

Résultats

Une fois les grands principes de notre système établis, il faut concevoir l'interface de consultation et d'annotation des expérimentations. Ce chapitre, plus court, propose une façon de réaliser les interfaces ainsi que les résultats obtenus sur le procédé de fermentation alcoolique.

6.1 Ergonomie générale

L'interface devra *a minima* proposer les fonctionnalités évoquées dans le cas d'utilisation de référence. En outre, elle s'efforcera de remplir les critères d'ergonomie de Bastien et Scapin[1]. On retrouvera tous les critères en annexe D.

La figure ci-dessous montre la page d'accueil pour consulter les expérimentations.

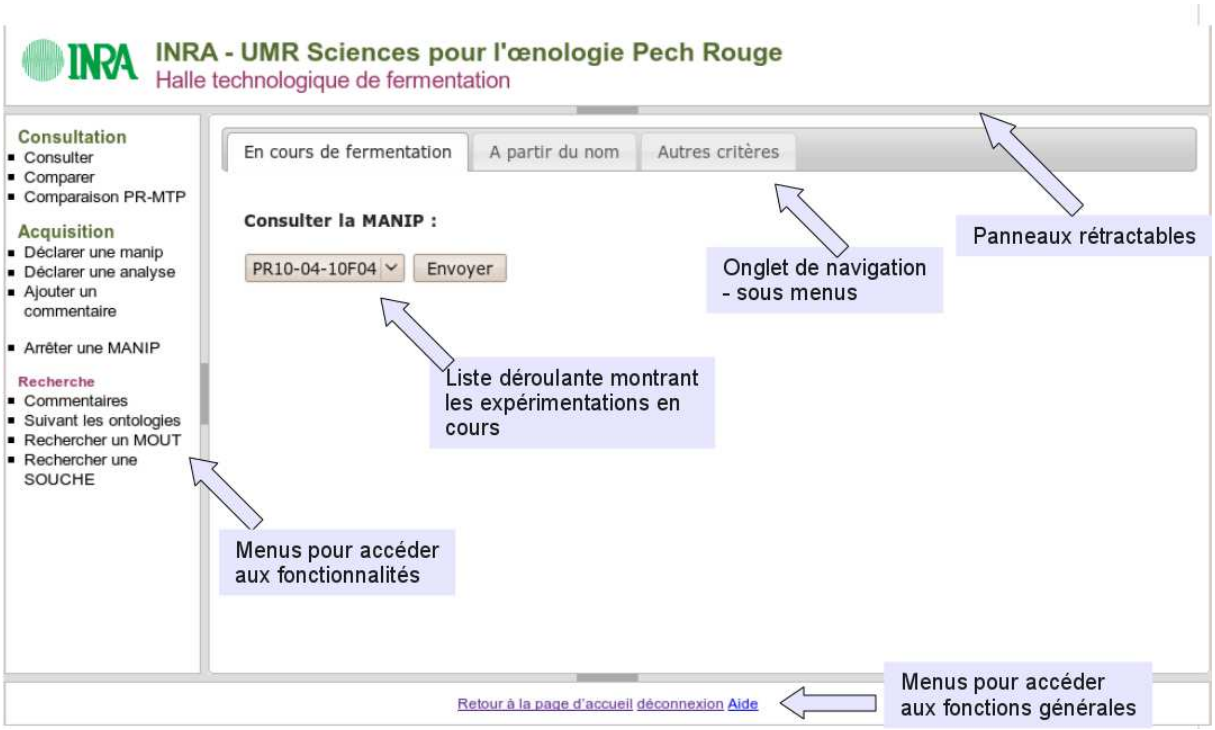


FIGURE 6.1 – Page d'accueil de l'interface.

6.1.1 La consultation des expérimentations et des mesures

La consultation des résultats des mesures est probablement la fonctionnalité la plus utilisée de l'interface. Elle doit être particulièrement soignée. La figure 6.2 montre la visualisation des données en ligne sous forme graphique.

Les données en ligne

Des champs de formulaires permettent de contrôler les variables à afficher. Ces champs sont générés avec l'annotation de contexte d'instrumentation contemporain de l'expérimentation.

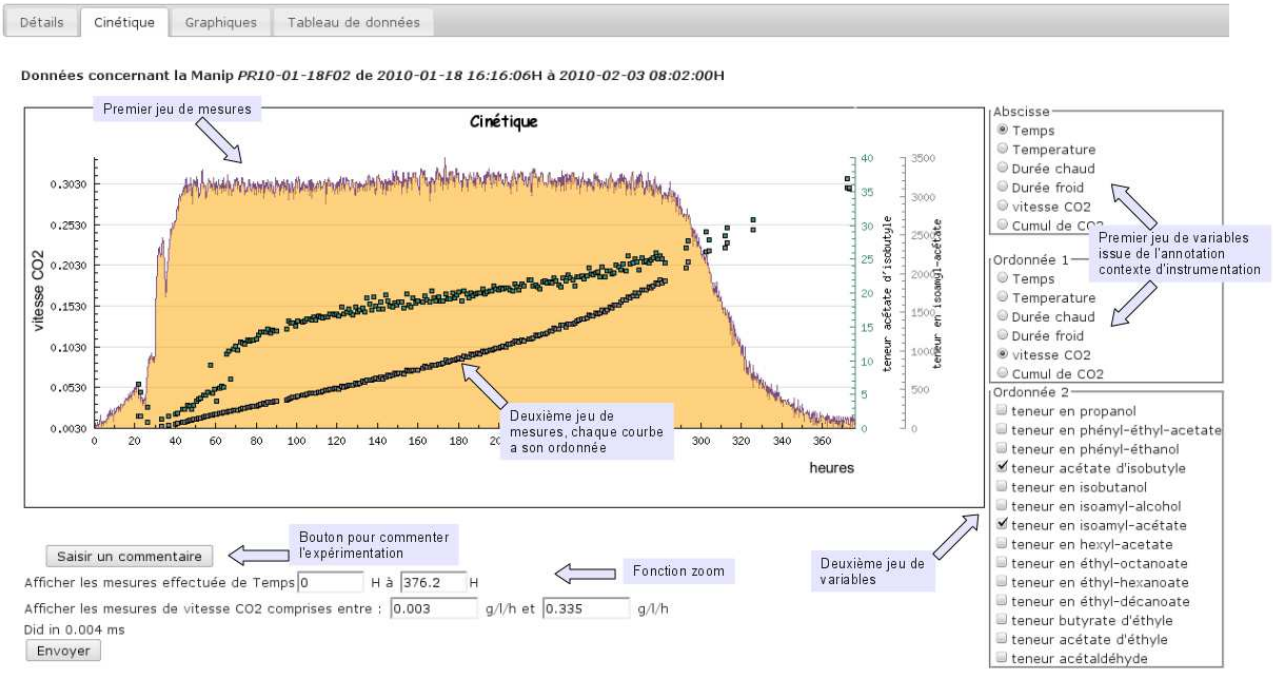


FIGURE 6.2 – Visualisation des données.

Le graphique utilise l'API *jpgraph*[1] qui permet notamment de superposer plusieurs ordonnées sur un même graphique. Le graphique est modifiable de façon interactive en cliquant sur les formulaires. Les deux jeux de variables n'étant pas synchronisés, nous représentons par défaut les données en fonction du temps, laissant à la librairie graphique la tâche de faire les ajustements nécessaires.

Données non synchronisées

Le concept de *jeu de variables* permet le traitement de variables en ligne non synchronisées, c'est-à-dire que la mesure n'a pas lieu au même moment.

Exemple:
La figure 6.2 montre un graphique superposant la vitesse de dégagement de CO_2 et la teneur en acétate d'isobutyle, deux variables en ligne mais dont les mesures sont désynchronisées. Elles sont toutes les deux affichées en fonction du temps.

1. `http://www.aditus.nu/jpgraph/`

Si l'on souhaite afficher une variable d'un jeu en fonction d'une variable d'un autre jeu, il faudra pratiquer une interpolation [2] qui nous permettra d'associer les valeurs de la première variable avec celles de la seconde en se basant sur le temps.
Plusieurs méthodes statistiques permettent de réaliser cette interpolation. Nous avons implanté la méthode des *spline* à l'aide du logiciel de statistique R [3].

```
interpolation <- function (codeManip, champs_var1, taille_echantillon_2)
{
# Initialisation des librairies d'accès à la base de données
   library(DBI)
   library(RMySQL)

# Connexion à la base
  m=dbDriver("MySQL")
  con = dbConnect(m, user="utilisateur", password="mot_de passe",
                     host="localhost", dbname="BD")
  requete = paste("SELECT unix_timestamp(date), var1 from mesures
                where codeManip='",codeManip,"'
                order by date ASC",sep="")

# Exécution de la requête
  res = dbSendQuery(con, statement=requete)

# Récupération des résultats dans une matrice
  A=as.matrix(fetch(res,n=-1))

# Appel de la fonction
  return ( spline(A[,1], A[,2],taille_echantillon_var2) )
}
```

Listing 6.1 – Interpolation d'un échantillon à l'aide de la fonction *spline* de R.

Notons que le programme R assure lui-même la lecture des données dans la base, c'est une solution préférable car plus performante. L'appel à cette fonction R impliquera un appel système pour réaliser les calculs comme déjà indiqué au listing 5.3.

6.1.2 Déclaration d'expérimentation

La déclaration d'une expérimentation est la première étape permettant la synchronisation des données par le sous-système de synchronisation.

Nous montrons l'exemple d'une déclaration de fermentation. Elle comprend une matière première, le moût, et un produit ajouté, une souche de levure. Dans cet exemple, l'interface a été spécialisée pour le cas de la fermentation alcoolique. Il donne cependant l'idée générale d'une interface de déclaration d'expérimentation.

2. L'interpolation est une opération mathématique permettant de construire une courbe à partir de la donnée d'un nombre fini de points

3. Nous partons de l'hypothèse que les mesures sont effectuées à intervalle régulier. Nous ramenons l'échantillon le plus grand à la taille de l'échantillon le plus petit.

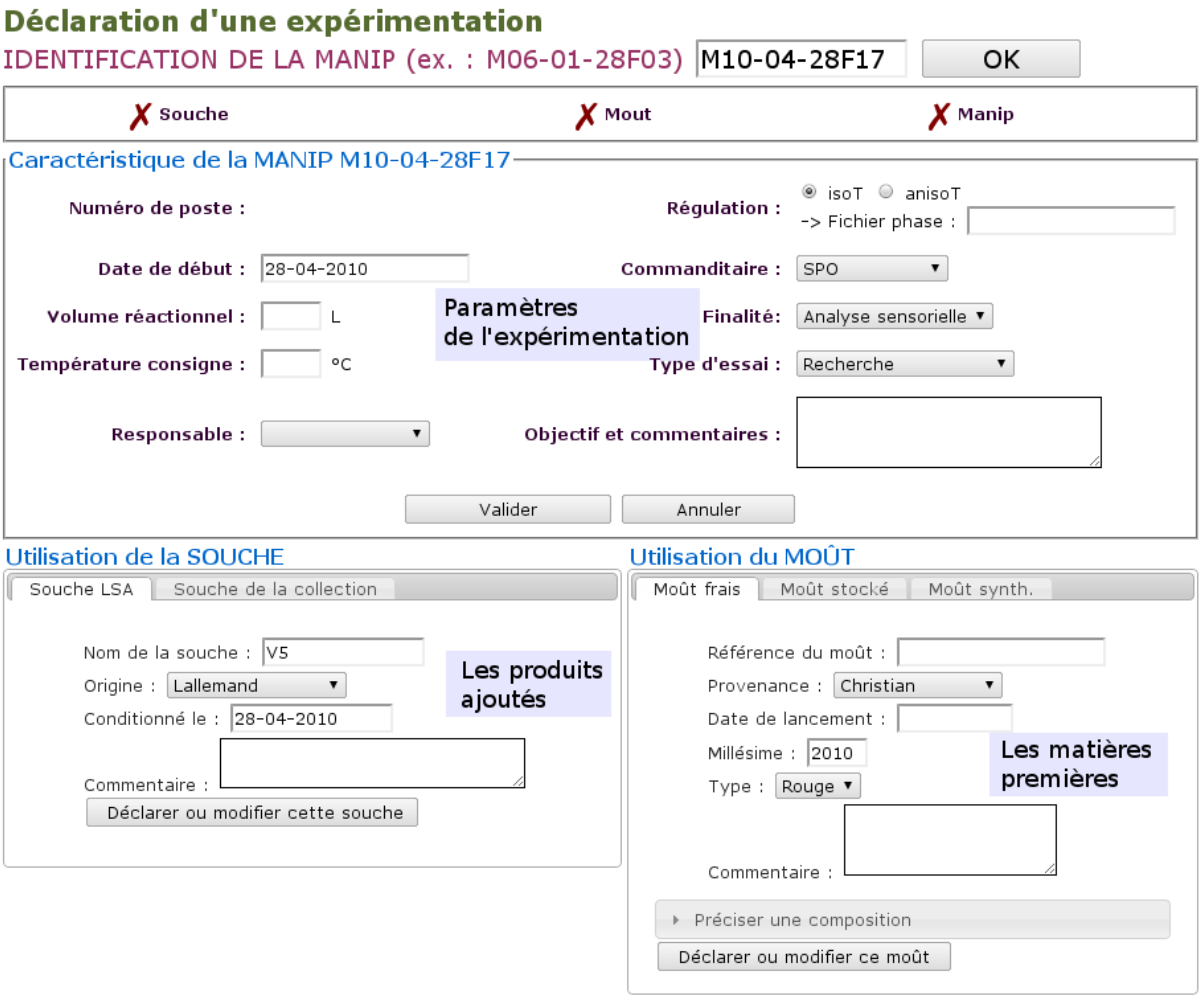


FIGURE 6.3 – Déclaration d'expérimentation.

6.1.3 Déclaration d'analyse

La déclaration d'une analyse hors-ligne pose un problème récurrent dans les laboratoires. Il s'agit d'un travail fastidieux et peu informatisé. Une approche consiste à utiliser un fichier (par exemple un fichier de tableur) préformaté que l'utilisateur n'aura qu'à remplir et à soumettre au système.

Dans le cadre du stage, nous avons développé une interface proposant ces fonctionnalités. Les étapes sont les suivantes :

1. l'utilisateur se rend sur l'interface et sélectionne l'entrée *déclarer une analyse* et commence par télécharger un fichier préformaté et pré-rempli,

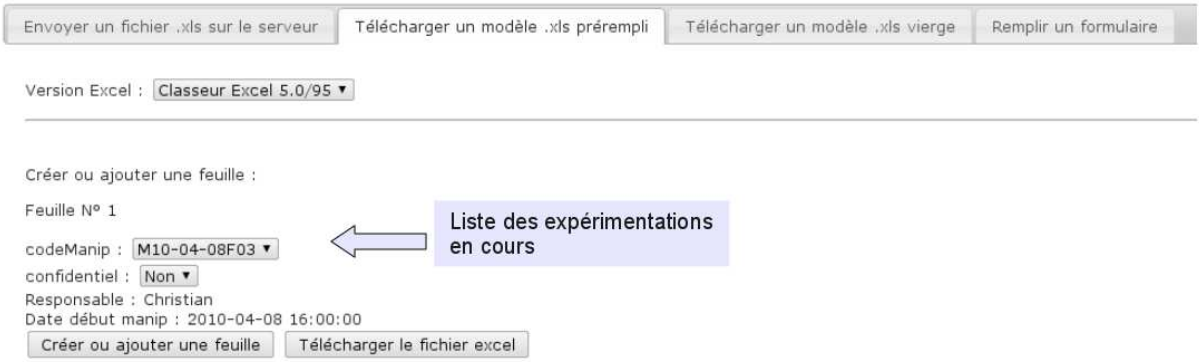
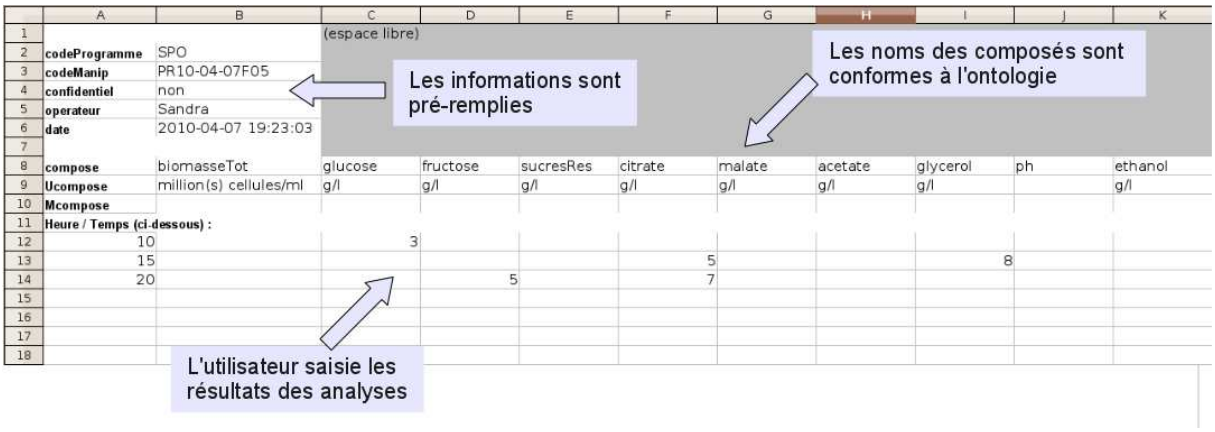


FIGURE 6.4 – Déclaration d'analyse : étape 1.

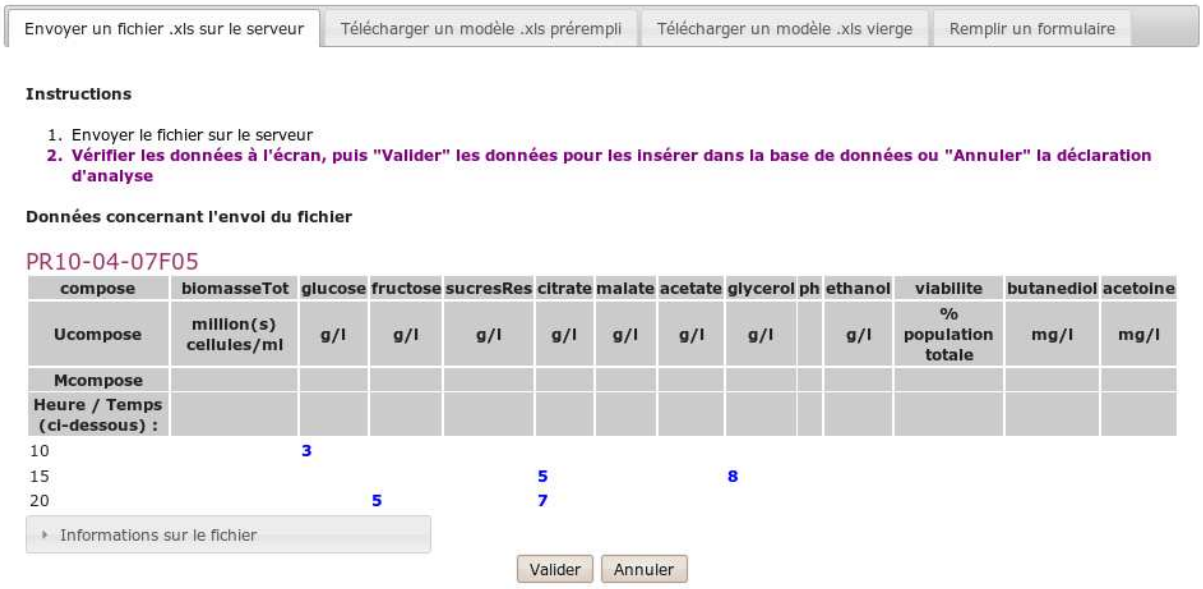
2. il effectue la saisie de ses résultats dans le fichier à l'aide d'un tableur,



	A	B	C	D	E	F	G	H	I	J	K
1			(espace libre)								
2	codeProgramme	SPO									
3	codeManip	PR10-04-07F05									
4	confidentiel	non									
5	operateur	Sandra									
6	date	2010-04-07 19:23:03									
7											
8	compose	biomasseTot	glucose	fructose	sucresRes	citrate	malate	acetate	glycerol	ph	ethanol
9	Ucompose	million(s) cellules/ml	g/l	g/l	g/l	g/l	g/l	g/l	g/l		g/l
10	Mcompose										
11	Heure / Temps (ci-dessous) :										
12	10		3								
13	15					5			8		
14	20			5		7					
15											
16											
17											
18											

FIGURE 6.5 – Déclaration d'analyse : étape 2.

3. il le soumet au système,



compose	biomasseTot	glucose	fructose	sucresRes	citrate	malate	acetate	glycerol	ph	ethanol	viabilite	butanediol	acetoine
Ucompose	million(s) cellules/ml	g/l	g/l	g/l	g/l	g/l	g/l	g/l		g/l	% population totale	mg/l	mg/l
Mcompose													
Heure / Temps (ci-dessous) :													
10		3											
15					5			8					
20			5		7								

FIGURE 6.6 – Déclaration d'analyse : étape 3.

4. le système lui renvoie les résultats en précisant par un code couleur les données nouvelles (en bleu), les données modifiées (en rouge) et les données supprimées (en rouge et barré).

Le fichier soumis est également stocké sur le serveur et pourra être réutilisé et re-soumis ultérieurement.

6.1.4 Les annotations : événements, commentaires

Les fonctionnalités de saisie de commentaires et d'événements proposés au chapitre précédent permettent de rechercher les expérimentations en fonction des annotations qui les décrivent. La figure suivante propose une interface de recherche.

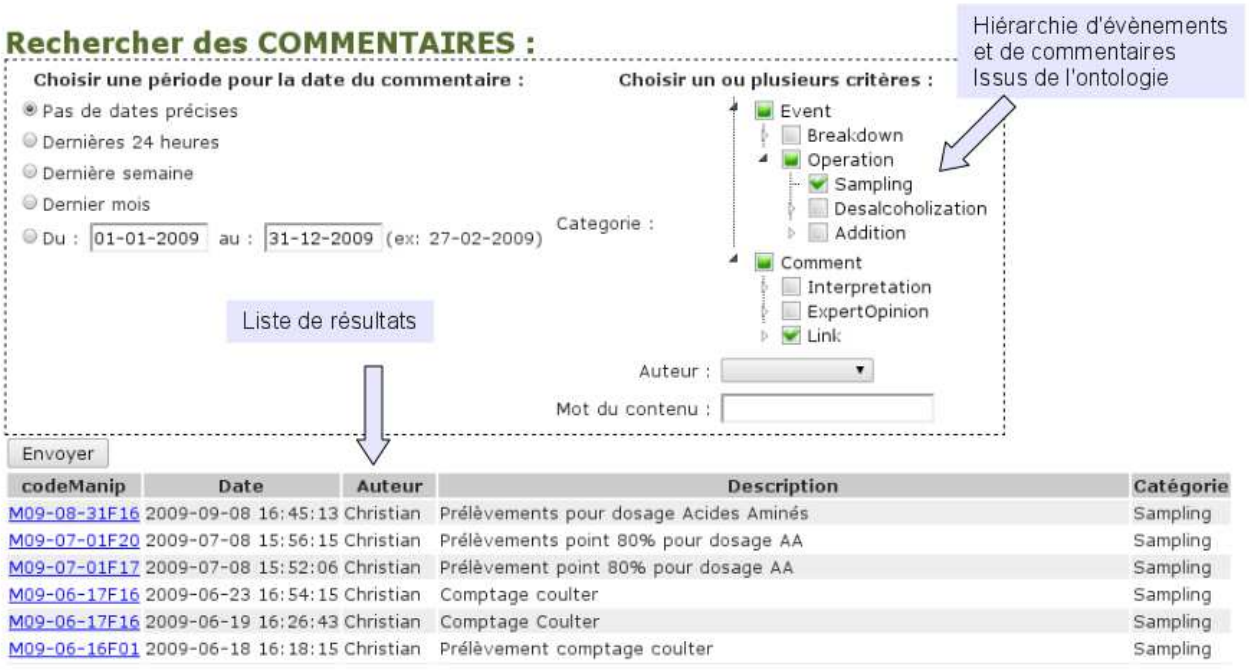


FIGURE 6.7 – Interface de recherche de commentaires.

Le choix de la période pour la recherche (partie de gauche de l'interface) est motivée par des raisons de performance. Rappelons qu'une première requête SQL, réputée rapide, permettra de filtrer les résultats en fonction de la période d'investigation et de réduire l'ensemble sur lequel la requête SPARQL interviendra.
Dans l'exemple, nous demandons à voir les expérimentations sur lesquelles un prélèvement a été effectué et pour lesquelles un fichier est disponible.

Rappelons que le moteur a une capacité de subsomption, et une requête sur les *Opérations* renverrait toutes les opérations (prélèvement, désalcoolisation, ajouts et les opérations pour lequel le sujet n'est pas précisé).

Conclusion

Ce mémoire aborde un sujet finalement assez vaste et se propose de guider les techniciens ayant en charge la mise en place de systèmes d'information scientifique. L'approche retenue se concentre sur l'organisation des connaissances scientifiques, autour de la donnée. La problématique de l'acquisition de la donnée elle-même est assurée par les logiciels de supervision très performants présentés au premier chapitre. L'ingénierie des connaissances en tant que support à la production scientifique qui commence *après* cette acquisition souffrait d'une lacune méthodologique. Nous espérons avoir apporté un certain nombre de réponses.

Le concept central de notre approche est la définition d'un *vocabulaire* commun dont l'objectif n'est finalement que de permettre aux utilisateurs de se comprendre en utilisant le même langage. Les ontologies présentées sont des sortes de dictionnaires auxquels les utilisateurs futurs du système d'information pourront se référer pour comprendre les expérimentations de leurs prédécesseurs. Le reste du document est une activité d'ingénierie autour de cette idée phare.

Ainsi notre travail apporte un cadre méthodologique — utiliser l'ingénierie des connaissances — et une solution technique — une base de données relationnelle juxtaposée à une base de faits conforme à l'ontologie — pour pérenniser les résultats des expérimentations. Cette démarche a été facilitée par l'émergence des technologies du web sémantique. Ces dernières s'intéressent à décrire formellement l'internet pour mieux exploiter ses ressources et faire des recherches plus intelligentes. C'est en définitive la même idée qui anime ce mémoire. L'ensemble des technologies autour de OWL est aujourd'hui suffisamment mature pour permettre leur implantation sur des sites en production.

Cependant, ces technologies restent relativement récentes et il a été difficile de s'appuyer sur des travaux antérieurs pour concevoir notre architecture. Une recherche a pourtant bien été effectuée, mais sans résultats probants, les seuls programmes disponibles sont encore à l'état de prototypes et bien souvent impossibles à tester. Les principales difficultés rencontrées ont concerné :

- la compréhension des technologies du web sémantique, des mécanismes d'inférence, des subtilités différenciant OWL, RDF et RDFS et de leur utilisation *pertinente*. On pourra contre-argumenter que les réalisations de cet ouvrage pourraient être implémentées en utilisant des bases de données,
- la prise en main des outils tels que les raisonneurs qui sont d'une part très sensibles aux erreurs de syntaxe et d'autre part sujets à des problèmes de performance. Par exemple, il a été ardu de réaliser les premières requêtes SPARQL,
- l'articulation entre base de données et base de faits. Où positionner les annotations RDF ? Nous ne pouvons prédire si les solutions retenues résisteront correctement à une montée en charge.

Une réponse a été apportée pour chacun des points et il sera nécessaire d'accumuler des retours sur expérience pour valider ou améliorer la méthode et le système.

Ce travail est une première pierre dans l'édifice de l'informatisation des bioprocédés et il souffre encore de quelques limites :

- l'implication des utilisateurs est fondamentale et sans elle, il est impossible d'annoter correctement les expérimentations. Nous l'avons vu dans la section concernant les thématiques de recherche, il est extrêmement difficile de faire décrire à un chercheur pourquoi il lance telle série d'expériences (probablement parce que les objectifs ne sont pas tout le temps clairs). Ce problème nous rappelle que l'homme est au cœur des systèmes d'information et que la solution est davantage dans le management (formation, sensibilisation) des équipes que dans un dispositif technique omnipotent.
- l'intégration de notre système avec les logiciels de supervision, notamment pour récupérer les informations déjà saisies lors de l'installation des capteurs et actionneurs. Il y a ici un manque de standards de description de plan d'expérience, de paramètres de régulation et de configuration qui permettraient aux deux systèmes de communiquer. La solution se trouve certainement dans l'utilisation de logiciels de supervision ouverts comme ODIN, où il est possible de communiquer avec les équipes de développement.

Ce travail ouvre également des perspectives d'évolution. Nous l'avons brièvement évoqué dans le dernier chapitre, les données font souvent l'objet d'un traitement statistique avec des langages appropriés (comme R). Il nous semblerait intéressant de permettre que ses fonctionnalités d'analyse soient directement accessibles depuis l'interface web, le mécanisme d'interpolation proposé n'est qu'un aperçu de ce qui pourrait être réalisé. La détection des pannes ou des erreurs de mesure est une autre fonctionnalité qui peut s'envisager à l'aide d'outils statistiques d'analyse de courbes. Quoiqu'il en soit, la démarche de modélisation experte est déjà un pas en avant dans la gestion des données scientifiques.

Bibliographie

[1] J.M.C. Bastien, C. Leulier, and D.L. Scapin. L'ergonomie des sites web. *Créer et maintenir un service Web*, pages 111–173, 1998.

[2] JJ Bimbenet, A. Duquenoy, and G. Trystram. *Génie des procédés alimentaires*. 2007.

[3] O. Corby, R. Dieng-Kuntz, and C. Faron-Zucker. Querying the semantic web with corese search engine. In *ECAI*, volume 16, page 705. Citeseer, 2004.

[4] E.R. Harold and W.S. Means. XML en concentré :[manuel de référence rapide]. 2005.

[5] Lionel Boillereaux Jean-Marie Flaus. *Modélisation est estimation - les procédés agroalimentaires 1*. Lavoisier - Hermes, 2003.

[6] J. Madin, S. Bowers, M. Schildhauer, S. Krivov, D. Pennington, and F. Villa. An ontology for describing and synthesizing ecological observation data. *Ecological Informatics*, 2(3) :279–296, 2007.

[7] Alejandra Olmos Perez. *Contribution à l'optimisation de la conduite des procédés alimentaires*. PhD thesis, Ecole nationale supérieure des industries agricoles et alimentaires, 2003.

[8] W3C Recommandation. Rdf primer. `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`, 2004.

[9] W3C Recommandation. Rdf vocabulary description language 1.0 : Rdf schema. `http://www.w3.org/TR/rdf-schema/`, 2004.

[10] W3C Recommandation. Xml schema part 0 : Primer second edition. `http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/`, 2004.

[11] W3C Recommandation. Extensible Markup Language. `http://www.w3.org/TR/2006/REC-xml11-20060816/`, 2006.

[12] W3C Recommandation. Namespaces in xml 1.1. `http://www.w3.org/TR/2006/REC-xml-names11-20060816/`, 2006.

[13] W3C Recommandation. Sparql query language for rdf. `http://www.w3.org/TR/rdf-sparql-query/`, 2008.

[14] W3C Recommandation. Owl 2 web ontology language primer. `http://www.w3.org/TR/2009/REC-owl2-primer-20091027/`, 2009.

[15] H. Rijgersberg and J. Top123. UnitDim : an ontology of physical units and quantities.

[16] J.D. Ullman. *Database and knowledge-base systems*. Computer Science Press, 1988.

[17] E. Van der Vlist. *XML schema*. O'Reilly Media, Inc., 2002.

[18] Éric Sarrion. *Développement Web avec J2EE*. Édition O'Reilly Paris, 2005.

Table des figures

Table des codes sources

Glossaire

Actionneur Un actionneur permet d'assurer l'évolution du procédé dans le sens souhaité. *Il peut s'agir d'une vanne,d'une pompe, d'une lampe, etc.*

API Application Programming Interface ou interface de programmation est un ensemble de fonctions, classes ou procédures mis à la disposition des programmes informatiques par une bibliothèque logicielle, un système d'exploitation ou un service.

FTP File Transfert Protocol ou protocole de transfert de fichier est un protocole de communication destiné à l'échange informatique de fichiers sur un réseau TCP/IP. Il permet, depuis un ordinateur, de copier des fichiers vers un autre ordinateur du réseau ou encore de supprimer ou de modifier des fichiers sur cet ordinateur.

Ingénierie des connaissances L'ingénierie des connaissances est une discipline étudiant l'extraction et la formalisation de connaissances venant d'un expert humain.

Intrants En économie, un intrant est une donnée qui entre dans le cadre de la production d'un bien.

Recherche plein texte La recherche plein texte est une technique de recherche textuelle dans un document électronique ou une base de données, qui consiste pour le moteur de recherche à examiner tous les mots de chaque document enregistré et à essayer de les faire correspondre à ceux fournis par l'utilisateur.

Serveur d'applications Un serveur d'applications est un logiciel d'infrastructure offrant un contexte d'exécution pour des composants applicatifs. Le terme est apparu dans le domaine des applications web.

Socket Une socket (qui en français signifie prise) est une interface logicielle avec les services du système d'exploitation qui permet de recevoir et d'expédier des données. Il s'agit d'un modèle permettant la communication inter processus afin de permettre à divers processus de communiquer aussi bien sur une même machine qu'à travers un réseau TCP/IP.

Système d'information Un système d'information (SI) est l'ensemble des ressources humaines, logicielles et de données participant à la gestion, au traitement, au transport et à la diffusion de l'information au sein d'une organisation. Un SI n'est pas forcément informatisé.

Table Sous-entendu de base de données, une table est le conteneur fondamental qui regroupe les informations (les données), dans le modèle relationnel de Codd.

Unicode Unicode est une norme informatique, développée par le Consortium Unicode, qui vise à donner à tout caractère de n'importe quel système d'écriture un nom et un identifiant numérique, et ce de manière unifiée, quelle que soit la plateforme informatique ou le logiciel.

Annexe A

Script SQL du trigger présenté page 24

Ce script a été testé sous PostgreSQL et est facilement adaptable à d'autres SGBD.

A.1 La fonction de création de la table

```
CREATE FUNCTION fonc_creationtable() RETURNS trigger AS fonc_creationtable
 DECLARE
 --id de la table à créer
 nomtable text := NEW.idvariable;

-- type des valeurs à stocker
 type text := NEW.type;

BEGIN
 -- Vérification de l'identifiant de la table idTable
 IF nomtable IS NULL THEN
 RAISE NOTICE 'vartablecreate() error: nomtable ne peut etre null';
 RETURN NEW;
 END IF;

 SELECT 'idVariable' INTO nomtable
        FROM 'Variables' WHERE 'id'=idTable;

--Creation de la table

 CREATE TABLE "nameTable" (
   id integer NOT NULL,
   "idVariableInfo" integer,
   value type DEFAULT 0::double precision NOT NULL,
 );

END;
fonc_creationtable LANGUAGE plpgsql;
```

A.2 Mise en place du trigger

```
CREATE TRIGGER trigger_creationtable  AFTER  INSERT
    ON 'Variables'  FOR  EACH  ROW
    EXECUTE PROCEDURE fonc_creationtable ( arguments )
```

Annexe B

Les unités de référence du système internationnal

Il y a sept unités de base du SI :

mètre	m
Le mètre est la longueur du trajet parcouru dans le vide par la lumière pendant une durée de 1/299 792 458 de seconde.	
kilogramme	kg
Le kilogramme est l'unité de masse ; il est égal à la masse du prototype international du kilogramme.	
seconde	s
La seconde est la durée de 9 192 631 770 périodes de la radiation correspondant à la transition entre les deux niveaux hyperfins de l'état fondamental de l'atome de césium 133.	
ampère	A
L'ampère est l'intensité d'un courant constant qui, maintenu dans deux conducteurs parallèles, rectilignes, de longueur infinie, de section circulaire négligeable et placés à une distance de 1 mètre l'un de l'autre dans le vide, produirait entre ces conducteurs une force égale à 2 × 10–7 newton par mètre de longueur.	
kelvin	K
Le kelvin, unité de température thermodynamique, est la fraction 1/273,16 de la température thermodynamique du point triple de l'eau.	
mole	mol
La mole est la quantité de matière d'un système contenant autant d'entités élémentaires qu'il y a d'atomes dans 0,012 kilogramme de carbone 12.	
candela	cd
La candela est l'intensité lumineuse, dans une direction donnée, d'une source qui émet un rayonnement monochromatique de fréquence 540 x 1012 hertz et dont l'intensité énergétique dans cette direction est 1/683 watt par stéradian.	

$http : //www.bipm.org/fr/si/base_units/$

Annexe C

Installation de tomcat et axis sur Linux

C.1 téléchargement du programme

`http://tomcat.apache.org/`

C.2 Installation

se placer dans `/opt` ou tout autre répertoire respectant les conventions de l'administrateur.

```
cd /opt
# Creation d'un repertoire
mkdir tomcat-install

# Telechargement
wget url_tomcat_mirroir

# Decompression
unzip apache-tomcat-version.zip

# Deplacement vers le repertoire de destination
mv apache-tomcat-version /opt

# Modification des droits pour autoriser l'execution du programme
chmod a+x apache-tomcat-version/bin/catalina.sh
chmod a+x apache-tomcat-version/bin/setclasspath.sh
```

C.3 Ajout d'un service tomcat sur le système

```
#création d'une entree dans init.d qui recence les services

vi /etc/init.d/tomcat

# Tomcat auto-start
# description: Auto-starts tomcat
# processname: tomcat6
```

```
# pidfile: /var/run/tomcat.pid

case \$1 in
 start)
sh /opt/tomcat/bin/startup.sh
;;
stop)
sh /opt/tomcat/bin/shutdown.sh
;;
restart)
sh /opt/tomcat/bin/shutdown.sh
sh /opt/tomcat/bin/startup.sh
;;
esac
exit 0

# Sortie de vi

# droits d'execution
chmod +x tomcat
```

C.4 Lancement automatique au démarrage du système

Création d'une entrée dans le runlevel 5, pour assurer un démarrage automatique au démarage de la machine.

```
cd /etc/rc5.d
ln -s /etc/init.d/tomcat S93tomcat

# et pour l'extinction
cd /etc/rc0.d
ln -s /etc/init.d/tomcat K08tomcat

# et le reboot
cd /etc/rc1.d
ln -s /etc/init.d/tomcat K08tomcat
```

C.5 Création d'un utilisateur administrateur

Editer `/opt/apache-tomcat-version/conf/tomcat-users.xml` et ajouter :

```
<role rolename="manager" />
<user username="admin" password="adminadmin" roles="manager" />
```

C.6 Démarrage de tomcat

Vérifier l'ouverture du port 8080 utilisé par défaut par Tomcat.
`/etc/init.d/tomcat start`
Tester avec un navigateur l'adresse :
`http://localhost:8080/`

C.7 installation d'AXIS

Axis est une extension pour les web service, une implémentation de SOAP. A télécharger sur `http://ws.apache.org/axis2/`.
Puis :

```
unzip axis-version.zip
#on obtient un axis2.war à deplacer
mv axis2.war /opt/apache-tomcat-version/webapps/
```

Relancer tomcat :
`/etc/init.d/tomcat restart`
Tester :
`http://localhost:8080/axis2/`

C.8 Déploiement d'un Web Service

Après le développement du Web Service, pour lequel il sera plus simple d'utiliser un IDE comme *eclipse* ou *Netbeans*, lancer la commande `Build` qui génére un fichier `.aar` et déplacer ce dernier dans :
`/opt/apache-tomcat-version/webapps/axis2/WEB-INF/services/`
Ajouter les dépendances dans :
`/opt/apache-tomcat-version/webapps/axis2/WEB-INF/lib/`
Redémarrer tomcat

C.9 Liste des librairies

corese.jar RDFAParser.jar arp.jar arq.jar icu4j.jar jena.jar jrdf.jar lucene-core.jar notio.jar openrdf-model.jar openrdf-util.jar rio.jar slf4j-api.jar slf4j-log4j.jar stax-api.jar wstx-asl.jar xercesImpl.jar

C.10 Fichiers journaux de Tomcat

Pour suivre les journaux de Tomcat :
`/opt/apache-tomcat-version/logs/` et notamment le fichier :
`catalina.out`

Annexe D

Les critères d'ergonomie de Bastien et Scapin

Les "critères ergonomiques" correspondent à une liste d'éléments à observer afin d'évaluer l'ergonomie d'un produit, d'un site, d'une interface... Il existe une multitude de critères, mais les plus célèbres sont ceux recensés par Bastien et Scapin, et les "heuristiques de Nielsen".

Voici un résumé de ces critères en une liste de conseils à prendre en compte dès le début de la conception :

D.1 Favoriser la simplicité

Eviter les informations superflues, les fonctionnalités supplémentaires qui ne font pas partie de l'objectif premier du produit... Allégorie de l'intérêt de la simplicité comparant deux couteaux suisses très différents Parler le langage de l'utilisateur

Essayer d'utiliser des termes que votre cible comprendra à coup sûr. Simplifier le vocabulaire s'il y a un risque de mauvaise compréhension.

D.2 Alléger les pages

Eviter de surcharger les pages d'information :
- Aérer l'information,
- Eviter les discours inutiles,
- Limiter le bruit visuel,
- et ne pas surcharger la charge mentale.

D.3 Être cohérent

Essayer de conserver une cohérence sur l'ensemble du site : une navigation identique sur toutes les pages, une apparence graphique constante, des textes qui gardent le même ton...

D.4 Informer l'utilisateur

S'assuree qu'à chacune des actions de l'utilisateur, il a le moyen de savoir que son action et prise en compte et les conséquences qu'elle a eu.

D.5 Indiquer clairement les actions

Etre explicite dans les intitulés de boutons et de liens.

D.6 Prévenir les erreurs

Afficher un message avant les opérations risquées (fermeture de fichier, suppression...) ou longues. Si une "erreur" survient, essayer de les traiter simplement en indiquant la cause de l'erreur et le moyen de la corriger. Eviter le vocabulaire trop dramatique comme "erreur fatale !".

D.7 Utiliser les conventions

Si l'application n'est pas complètement originale, il existe sans doute déjà des standards qui peuvent lui être appliqué. Il vaut alors mieux les respecter. Exemple : logo cliquable qui ramène à l'accueil.

Annexe E

L'architecture du sous-système de consultation

L'application de gestion des fermentations alcooliques est structurée selon le motif architectural modèle - vue - contrôleur. Il consiste à séparer les données (le modèle), l'interface homme-machine (la vue) et la logique de contrôle (le contrôleur).

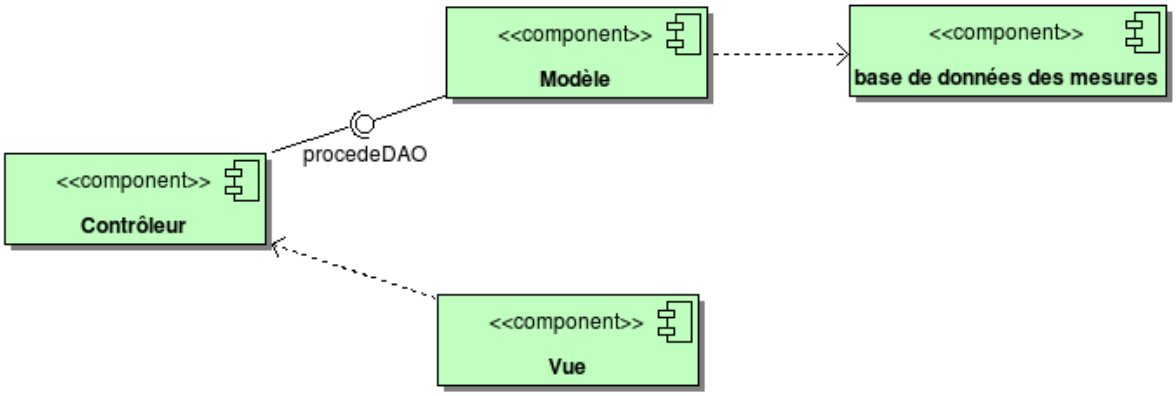


FIGURE E.1 – Architecture de l'application

- *Le modèle* représente les données de l'application. DAO signifie Data Access Object, les classes héritant de cette classe abstraite permettent d'accéder au données de la base. Elles comprennent des méthodes effectuant des requêtes SQL.
- *La vue* représente l'interface utilisateur, ce avec quoi il interagit. Elle n'effectue aucun traitement. Dans un contexte Web, les vues sont la plupart du temps des fichiers HTML.
- *Le contrôleur* gère l'interface entre le modèle et le client. Il va interpréter la requête de ce dernier pour lui envoyer la vue correspondante. Il effectue la synchronisation entre le modèle et les vues.

Ainsi notre application doit obéir à deux règles :
- pas de SQL ailleurs que dans les classes du modèle,
- pas de HTML ailleurs que dans les vues

D'autre part, les vues ne prennent pas la forme de classes, mais plutôt de squelettes HTML (*template* en anglais).

E.1 Les classes du modèle

A minima, on doit retrouver une classe par table de la base de données. Ces dernières implémenterons des méthodes CRUD (**C**reate, **R**ead, **U**pdate et **D**elete). Nous proposons également d'utiliser une couche d'abstraction de base de données. L'objectif étant de pouvoir changer de SGBD sans changer le code source du modèle. [1].
PHP possède une classe **PDO** (PHP Data Object) réalisant cette abstraction. L'utilisation de PDO est triviale :

```
<?php
// connection à la base de donnée
$dbh = new PDO('mysql:host=localhost;dbname=procede', $user, $pass);

// Exécution d'une requête
$sql =  'SELECT RDFAnnotation FROM commentaires';
$statement = $dbh->query($sql);

// Exploitation des résultats
while ($ligne = $statement->fetchObject()) {
  $result[] = $ligne->RDFAnnotation;
}
?>
```

Ainsi les classes du modèle utiliserons l'objet **PDO** créé par le contrôleur en tant que variable globale.

E.2 Le contrôleur

Le contrôleur doit assurer un certain nombres de fonctions du programme, notamment :
- assurer l'identification des utilisateurs, en utilisant des variables de sessions ou une classe d'authentification,
- initialiser la connexion vers la base de données
- construire l'interface de l'application par l'appel aux vues adéquates (voir après),
- appeler l'action demandé par l'utilisateur. Il peut s'agir de l'appel à un service asynchrone.

Les actions sont appelés par l'intermédiaire des menus de l'interface qui renverrons vers une URL contenant en paramètre le nom de l'action. Ce nom correspondant à un fichier PHP.
Par exemple : `http://localhost/accueil.php?action=declareAnalyse` correspondra à la fonction pour déclarer une analyse.

Le point d'entrée (l'URL) de l'application est toujours le même, à savoir un fichier PHP particulier, dans notre exemple `accueil.php`. Par contre les paramètres de l'URL seront différents selon l'action demandée.

E.3 Les vues

Les vues sont dans la plupart des cas des fichiers squelettes en HTML et contenant des balises PHP pour l'affichage des éléments dynamique. A une vue correspond un fichier HTML, lorsque

1. Etant entendu que le programmeur essaiera autant que possible d'utiliser du SQL le plus standard possible

la vue est complexe, on découpera en sous-vue.
Une vue générale contient les entêtes du document HTML. Nous donnons ci-dessous un extrait du squelette général de notre application.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
  <head>
    <meta http-equiv='Content-Type' content='text/html; charset=UTF-8'/>
    <title><?php echo $titre; ?></title>
    <!--    Feuille de style -->
     <link rel="stylesheet" type="text/css" href="style.css" />
    <!--    Base de jquery-->
    <script type="text/javascript" src="js/jquery.js"/>
    <!-- Autres entetes -->
  </head>
  <body>
   <div id="header" class="ui-layout-north">
    <h1><a href='accueil.php'>
    <img src="photos/logo.png" alt="logo INRA" /></a>
      INRA  -  UMR  Sciences pour l'&#156;nologie</h1>
    </div>
    <div id="sidebar" class="ui-layout-west">
      <h1>Consultation</h1>
        <?php echo $menuConsultation; ?>
      <h1>Acquisition</h1>
        <?php echo $menuAcquisition; ?>
    </div>

    <div id="footer" class="ui-layout-south">
      <!-- contenu pied de page -->
    </div>
    <div class="ui-layout-center" id="contenuPricipal">
      <?php echo $rendu; ?>
    </div>
  </body>
</htm>
```

Listing E.1 – Extrait du fichier `general.tpl.html`

Dans ce listing, on retrouve la structure visuelle de la page évoqué page 60 dans les divisions `<div>` de la page.

- *header* contient l'entête de la page,
- *sidebar* contient les menus de l'application, avec deux niveaux, « acquisition » et « consultation » ,
- *footer* propose des liens généraux : aide, déconnexion et retour à l'accueil,
- *contenuPrincipal* est la partie principale de l'application et son contenu dépendra de l'action courante. la variable `$rendu` est une chaîne de caractère contenant du HTML et affectée par les actions.

E.4 Les actions

A chaque item du menu général correspond un fichier de code source réalisant l'action demandée. Le fichier d'action fait partie du contrôleur et aura pour rôle de communiquer avec le modèle et d'appeler les vues.
A chaque fichier d'action est associé un squelette HTML, portant le même nom mais avec une extension `.tpl.html`.

E.4.1 L'appel de l'action par le contrôleur général

Le principe est d'utiliser les inclusions de fichier en PHP et la mise en *buffer*[2] des sorties. Le listing suivant montre ce mécanisme :

```
<?php
// Traitement des variables GET que l'on trouve dans l'URL
if (isset($_GET['action']) && $_GET['action'] != '') {
  if (file_exists(REPERTOIRE_ACTIONS.$_GET['action'].'.php')) {
    // Mise en tampon des sorties
    ob_start();

        // inclusion du fichier
        include_once REPERTOIRE_ACTIONS.$_GET['action'].'.php';

        // récupération des sorties
        $rendu .= ob_get_contents();
        ob_end_clean();
  } else {
        $rendu .= '<p class="alerte">l\'action '.
                        $_GET['action'].' n\'existe pas.</p>';
  }
} else {
  // L'action par défaut si rien n'est précisé dans l'URL
  ob_start();
  include_once REPERTOIRE_ACTIONS.'resume.php';
  $rendu .= ob_get_contents();
  ob_end_clean();
}
?>
```

Listing E.2 – Extrait du fichier `accueil.php`

E.4.2 L'appel des squelettes par les contrôleurs

De la même manière, on incluera le fichier squelette. On notera que lors de cet appel, il est nécessaire que les variables présentes dans le fichier squelette aient été initialisée, notamment la variable `$rendu`.

```
<?php
ob_start();
include_once REPERTOIRE_TEMPLATES.'general.tpl.html';
```

2. Un buffer est une mémoire tampon, c'est à dire temporaire et en mémoire vive. Dans notre cas, les sorties sont réalisées par des commandes `echo` ou `print`, dont l'appel entraîne l'envoie des entêtes HTTP, sans que le script soit terminé, ce que nous souhaitons éviter.

```
$rendu = ob_get_contents();
ob_end_clean();

echo $rendu;

?>
```

Listing E.3 – Extrait du fichier `accueil.php` : appel du squelette général

On répétera cette procédure à l'intérieur de chaque fichier d'action.

E.5 Les services

On peut utiliser le même principe pour l'appel de service. Par service nous entendons ici une fonction PHP accédée via HTTP, c'est-à-dire reprenant les principes des service web REST. Ils peuvent renvoyer n'importe quel type de données, il s'agira souvent de HTML, XML, JSON ou bien d'images (PNG, JPG...). Les services vont nous servir entre autres à la réalisation d'appels Javascript asynchrones.

L'appel à un service se fait par l'intermédiaire du contrôleur général avec un paramètre `service`.
exemple : `http://localhost/accueil.php?service=graphique` appelle le service pour générer un fichier graphique. Noter bien la function `exit();`, indispensable pour que le script ne continue pas son exécution.

```
<?php
if (isset($_GET['service']) && $_GET['service'] != '') {

  if (file_exists(REPERTOIRE_SERVICES.$_GET['service'].'.php')) {
    include_once REPERTOIRE_SERVICES.$_GET['service'].'.php';
    exit();
  } else {
    echo 'Le service n\'est pas accessible';
  }
}
?>
```

Listing E.4 – Extrait du fichier `accueil.php` : appel des services

Les services bénéficient de toutes paramètres généraux de l'application comme la connexion à la base de données ou l'authentification.

Résumé

Ce travail aborde le thème de l'informatisation des procédés sous l'angle de la pérennisation des données scientifiques, données brutes et contexte scientifique qui est à leur origine. Il s'intéresse à améliorer leur utilisabilité dans le temps en partant du constat que les expérimentations, acte fondamental de la production de données, sont souvent mal documentées et ne permettent que très rarement d'être réutilisées.

L'approche présentée met en avant l'analyse métier du domaine des bioprocédés en détaillant les concepts de matières premières, produits ajoutés et des différents type de mesures. Elle propose une hiérarchie de commentaires (événements, opérations, analyses...) ainsi que des thèmes de recherche et associe un contexte d' instrumentation, mémoire de l'installation technique du procédé. Cette analyse débouche sur la division du système d'information en deux parties, un modèle pour les données brutes qui prendra la forme classique d'une base de données relationnelle, un modèle sémantique sous forme d'une ontologie qui utilisera les technologies du Web sémantique. Ce système permettra d'≪ annoter ≫ les expérimentations.

Pour terminer, le mémoire propose une approche pour l'implémentation en machine en justifiant les différents choix technologiques. L'architecture client / serveur est retenue, avec une interface Internet ≪ riche ≫associée à une application Web écrite en PHP et un mécanisme de service Web pour l'interrogation des ontologies.

Abstract

This work deals with the problem of computing biological process with the aim of make durable scientific data, raw data and context data. Aware that experiments, the former act before producing data, are not enough documented and are rarely reused, it tries to improve future use of data.

This approach use business modelization of biological process, pointing out the concept of raw material, adding product and different kind of measurement. A hierarchy of comments, research thema is provided (event, operation, analysis...) and the instrumentation context concept is introduce. This analysis lead us to divide the information system in two part, a model for raw data which be implemented in a classical relational DB way and a semantic model build as an ontology which will use the semantic web technology. This system allow to annotate the experiments.

At last, the study provide a way of implementing the system and provide sone possible technical choices. The client / server architecture is choosen with a rich Internet interface associated with a server side PHP application and a web service for querying ontologies.