

Développement d'une ontologie 101 : Guide pour la création de votre première ontologie

Natalya F. Noy et Deborah L. McGuinness
Université de Stanford, Stanford, CA, 94305
noy@smi.stanford.edu et dln@ksl.stanford.edu

Traduit de l'anglais par Anila Angjeli, BnF, Bureau de normalisation documentaire.

1. Pourquoi développer une ontologie?

Ces dernières années le développement des ontologies - spécifications formelles explicites de termes d'un domaine et de relations entre elles (Gruber 1993) - a quitté les laboratoires d'Intelligence Artificielle pour gagner les postes informatiques des experts de domaines. Les ontologies sont devenues très courantes dans le World-Wide Web. Le champ des ontologies dans le web varie de taxonomies larges servant à catégoriser les sites Web (tels que dans Yahoo!) aux catégorisations de produits destinés à la vente et de leurs caractéristiques (tel que dans Amazon.com). Le Consortium WWW (W3C) est en train de développer le Resource Description Framework (Brickley and Guha 1999), un langage d'encodage de la connaissance sur les pages Web pour rendre cette connaissance compréhensible aux agents électroniques qui effectuent des recherches d'information. Defence Advances Research Projects Agency (DARPA), conjointement avec le W3C, développe le DARPA Agent Markup Language (DAML) en élargissant RDF avec d'autres constructions expressives qui visent à faciliter l'interaction des agents sur le Web (Hendler et McGuinness 2000). Plusieurs disciplines développent actuellement des ontologies normalisées utilisables par les experts de domaines pour partager et commenter l'information dans leurs domaines. La médecine par exemple, a produit de vastes vocabulaires normalisés structurés tels que SNOMED (Price and Spackman 2000) et le réseau sémantique du Unified Medical Language System (Humphreys and Lindberg 1993). De même naissent de larges ontologies universelles : par exemple le Programme des Nations Unies pour le développement et Dun & Bradstreet ont unis leurs efforts pour développer l'ontologie UNSPSC qui fournit une terminologie pour les produits et les services. (www.unspsc.org).

Une ontologie définit un vocabulaire commun pour les chercheurs qui ont besoin de partager l'information dans un domaine. Elle inclue des définitions lisibles en machine des concepts de base de ce domaine et de leurs relations.

Pour quelles raisons développer une ontologie ? En voici quelques-unes :

- Partager la compréhension commune de la structure de l'information entre les personnes ou les fabricants de logiciels.
- Permettre la réutilisation du savoir sur un domaine
- Expliciter ce qui est considéré comme implicite sur un domaine
- Distinguer le savoir sur un domaine du savoir opérationnel

Analyser le savoir sur un domaine

Partager la compréhension commune de la structure de l'information entre les personnes ou les fabricants de logiciels est une des raisons les plus courantes qui conduit à développer des ontologies (Musen 1992 ; Gruber 1993). Supposons, par exemple, qu'un certain nombre de sites Web contiennent de l'information médicale ou fournissent des services de e-commerce en médecine. Si ces sites partagent et publient tous la même ontologie, qui est à la base des termes qu'ils utilisent, alors les agents informatiques peuvent extraire et agréger l'information de ces

différents sites. Les agents peuvent utiliser cette information agrégée pour pouvoir répondre aux interrogations des utilisateurs ou comme données d'entrée pour d'autres applications.

Permettre la réutilisation du savoir sur un domaine était une des raisons majeures qui ont poussé la recherche sur les ontologies ces dernières années. Par exemple, les modèles de plusieurs domaines ont eu besoin de représenter la notion de temps. Cette représentation comprend les notions d'intervalles de temps, de moments précis de temps, de mesures relatives de temps, etc. Lorsqu'un groupe de chercheurs développe une telle ontologie en détail, les autres groupes peuvent simplement réutiliser pour leurs propres domaines l'ontologie développée. Qui plus est, s'il est besoin de construire une ontologie plus large, il serait possible d'intégrer plusieurs ontologies existantes décrivant des portions d'un domaine. On peut, également, réutiliser une ontologie générale tel que le UNSPSC et l'étendre pour permettre de décrire un domaine d'intérêt spécifique.

Rendre explicites les postulats d'un domaine sous-jacents à une implémentation rend possible la modification de ces spécifications, en cas d'évolution du savoir sur le domaine. Les postulats implicites sur un domaine exprimés en langage codé de programmation deviennent non seulement difficiles à deviner et comprendre mais, également difficiles à modifier, en particulier pour des personnes non expertes en programmation. Les spécifications explicites du savoir sur un domaine sont, de surcroît, utiles pour les nouveaux utilisateurs qui doivent apprendre la signification des termes du domaine.

Distinguer le savoir sur un domaine du savoir opérationnel est une autre des finalités courantes des ontologies. Nous pouvons décrire la tâche de configuration d'un produit à partir de ses constituants, en respectant les spécifications requises et implémenter un programme qui réalisera cette configuration indépendamment des produits et de leurs composants (McGuinness and Wright 1998). On peut alors développer une ontologie des parties composantes et des caractéristiques d'un PC et appliquer l'algorithme pour configurer des PC sur mesure. On peut aussi utiliser le même algorithme pour configurer des ascenseurs si cet algorithme est alimenté par une ontologie des parties composantes des ascenseurs (Rothenfluh et al. 1996).

Analyser le savoir sur un domaine est possible dès que la spécification des termes du domaine est faite. L'analyse formelle des termes est extrêmement précieuse aussi bien quand on veut réutiliser les ontologies existantes, que quand on veut les étendre (McGuinness et al. 2000).

Souvent une ontologie de domaine n'est pas toujours un but en soi. Développer une ontologie s'apparente à définir un ensemble de données et leur structure pour qu'elles soient utilisées par d'autres programmes. Les ontologies et les bases de connaissances élaborées à partir des ontologies sont utilisées comme données par les méthodes de solutions de problèmes, les applications indépendantes des domaines et les fabricants de logiciels. Par exemple, dans cet article nous développons une ontologie sur le vin, les mets et les alliances appropriées des vins et des plats. Cette ontologie peut être utilisée comme base pour toute une série d'applications visant le management des restaurants. Une application pourrait réaliser des suggestions de vins pour le menu du jour ou des questions-réponses des serveurs de restaurants et des clients. Une autre application pourrait analyser une liste d'inventaire des vins d'une cave et suggérer quels vins faire consommer et quels vins acheter pour les prochains menus.

Sur ce guide

Nous nous sommes fondés sur notre expérience en utilisant Protégé-2000 (Protege 2000), Ontolingua (Ontolingua 1997), et Chimaera (Chimaera 2000) comme environnements d'éditeurs d'ontologies. Dans ce guide nous utilisons Protégé-2000 pour nos exemples.

L'exemple du vin et des mets, que l'on utilise tout au long de ce guide, est inspiré d'un exemple de base de connaissances présenté dans un article décrivant CLASSIC - un système de

représentation de connaissances basé sur une approche de description-logique (Brachman et al. 1991). Le groupe universitaire de travail CLASSIC (McGuinness et al. 1994) a poursuivi l'élaboration de cet exemple. Protégé-2000 et d'autres systèmes FRL décrivent les ontologies de façon déclarative, indiquant explicitement ce qu'est une hiérarchie des classes et à quelle classe appartiennent les individus.

Certaines idées dans la conception des ontologies dans ce guide proviennent de la littérature sur la conception orientée-objet (Rumbaugh et al. 1991 ; Booch et al. 1997). Toutefois, développer une ontologie n'est pas comme concevoir classes et relations dans la programmation orientée-objet. La programmation orientée-objet est fondamentalement centrée sur les méthodes basées sur les classes - un programmeur réalise les décisions conceptuelles en s'appuyant sur les propriétés *opérationnelles* d'une classe, alors qu'un concepteur d'ontologie le fait en s'appuyant sur les propriétés *structurelles* d'une classe. En résultat, une structure de classe et les relations entre les classes dans une ontologie sont différentes de la structure d'un domaine similaire dans une programmation orientée-objet.

Il est impossible de traiter tous les points qui peuvent concerner un développeur d'ontologies. Dans ce guide nous ne faisons pas non plus l'effort de nous adresser à tous ces développeurs. En fait, nous essayons de fournir un point de départ ; un guide d'initiation qui pourrait aider un nouveau concepteur d'ontologies à développer des ontologies. Enfin, nous suggérons des références à consulter pour des explications sur des structures plus complexes et sur des mécanismes de conception, si les besoins du domaine traité les rendent nécessaires.

Finalement, il n'y a pas qu'une seule méthodologie correcte de conception d'ontologies et nous n'avons pas essayé d'en définir une. Les idées que nous présentons ici sont celles que nous avons jugées utiles dans notre propre expérience de développement d'ontologies. A la fin de ce guide nous suggérons une liste de références pour les méthodologies alternatives.

2. Qu'est une ontologie ?

La littérature d'Intelligence Artificielle contient plusieurs définitions pour une ontologie ; un bon nombre d'entre elles sont même contradictoires. Pour les besoins de ce guide une **ontologie** est une description formelle explicite des concepts dans un domaine du discours (**classes** (appelées parfois **concepts**)), des propriétés de chaque concept décrivant des caractéristiques et attributs du concept (**attributs** (appelés parfois **rôles** ou **propriétés**)) et des restrictions sur les attributs (**facettes** (appelées parfois **restrictions de rôles**)). Une ontologie ainsi que l'ensemble des **instances** individuelles des classes constituent une **base de connaissances**. Il y a en réalité une frontière subtile qui marque la fin d'une ontologie et le début d'une base de connaissances.

Les classes constituent le centre d'intérêt de plusieurs ontologies. Les classes décrivent les concepts dans le domaine. Par exemple une classe de vins représente tous les vins. Les vins spécifiques sont des instances de cette classe. Si en lisant ce document vous avez devant vous une coupe de vin de Bordeaux, le vin de Bordeaux contenu dans votre coupe est une instance de la classe des vins de Bordeaux. Une classe peut avoir des **sous-classes** qui représentent des concepts plus spécifiques que la super-classe (ou classe supérieure). Par exemple, on peut diviser la classe de tous les vins en vins rouges, blancs et rosés. Alternativement, nous pouvons diviser une classe de tous les vins en effervescents et non effervescents.

Les attributs décrivent les propriétés des classes et des instances : le vin Château Lafite Rothschild Pauillac est un vin charpenté ; il est produit par l'établissement vinicole de Château Lafite Rothschild. Nous avons deux attributs décrivant le vin dans cet exemple : l'attribut corps ayant pour valeur charpenté et l'attribut producteur ayant pour valeur établissement vinicole Château Lafite Rothschild. Au niveau de la classe, on peut dire que les instances de la classe Vin auront des attributs décrivant leur odeur, leur corps,

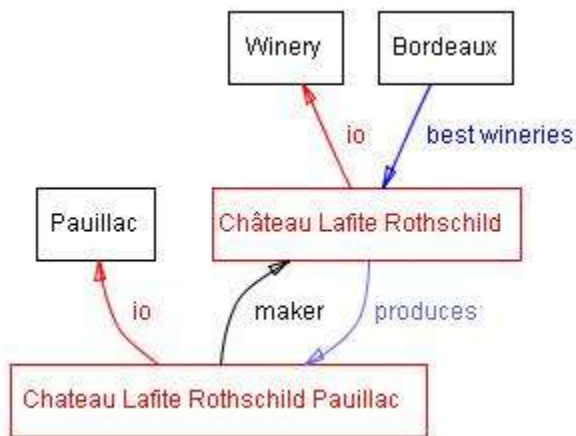
leur niveau de sucre, le producteur du vin et ainsi de suite

Toutes les instances de la classe Vin, et de sa sous-classe Pauillac, ont un attribut producteur dont la valeur est une instance de la classe Etablissement vinicole (Figure 1). Toutes les instances de la classe Etablissement vinicole ont un attribut produit qui se réfère à tous les vins (instances de la classe Vin et de ses sous-classes) produits par cet établissement vinicole.

En termes pratiques, développer une ontologie inclut :

- définir les classes dans l'ontologie,
- arranger les classes en une hiérarchie taxinomique (sous-classe - super-classe),
- définir les attributs et décrire les valeurs autorisées pour ces attributs
- renseigner les valeurs pour les attributs des instances

Nous pouvons créer une base de connaissances en définissant les instances individuelles de ces classes, en précisant les valeurs spécifiques des attributs ainsi que les restrictions des attributs.



3. Une simple méthodologie de génie cognitif

Comme nous l'avons déjà dit, il n'existe pas qu'un seul mode ou qu'une seule méthodologie "correcte" pour développer des ontologies. Ici nous abordons les points généraux qui doivent être pris en considération et nous offrons une des procédures possibles pour développer une ontologie. Nous décrivons une approche itérative dans le développement de l'ontologie : nous commençons par aborder l'ontologie de façon frontale. Ensuite nous revenons sur l'ontologie, que nous considérons en processus d'évolution, en l'affinant et en la complétant par des détails. Tout au long de ce processus nous discutons les décisions de modélisation à prendre par le concepteur, ainsi que les pour, les contre et les implications des différentes solutions.

Tout d'abord, nous voudrions mettre l'accent sur un certain nombre de règles fondamentales dans la conception des ontologies, règles auxquelles nous nous référerons plusieurs fois. Elles peuvent paraître dogmatiques, mais elles permettent de prendre des décisions à plusieurs occasions.

1. *Il n'y a pas qu'une seule façon correcte pour modéliser un domaine - il y a toujours des alternatives viables. La meilleure solution dépend presque toujours de l'application que vous voulez mettre en place et des évolutions que vous anticipez.*
2. *Le développement d'une ontologie est nécessairement un processus itératif.*
3. *Les concepts dans une ontologie doivent être très proches des objets (physiques ou*

logiques) et des relations dans votre domaine d'intérêt. Fort probablement ils sont des noms (objets) ou verbes (relations) dans des phrases qui décrivent votre domaine.

Ce qui veut dire que la définition du but d'utilisation de l'ontologie et de son degré de finesse (détaillée ou générale), guideront la plupart des décisions de modélisation tout au long du processus. Parmi les alternatives viables, nous aurons besoin de déterminer la plus adaptée à la tâche que nous nous sommes fixées, la plus intuitive, la plus extensible et la plus réalisable en termes de maintenance. Il faut également se rappeler qu'une ontologie est un modèle de la réalité du monde et que les concepts dans l'ontologie doivent refléter cette réalité. Après avoir défini une version initiale de l'ontologie, nous pouvons l'évaluer et la mettre au point en l'utilisant dans des applications ou dans des méthodes de solutions de problèmes, ou bien en discutant avec les experts du domaine ou, à la limite, faire les deux. En résultat, nous serons sûrement amenée à réviser l'ontologie initiale. Ce processus de conception itérative continuera vraisemblablement tout au long du cycle de vie de l'ontologie.

Etape 1

Nous suggérons de commencer le développement d'une ontologie en définissant son domaine et sa portée. C'est à dire en répondant à quelques questions de base :

- Quel est le domaine que va couvrir l'ontologie ?
- Dans quel but utiliserons nous l'ontologie ?
- A quels types de questions l'ontologie devra-t-elle fournir des réponses ?
- Qui va utiliser et maintenir l'ontologie ?

Les réponses à ces questions peuvent varier au cours du processus de la conception de l'ontologie, mais à chaque moment donné, elles aident à limiter la portée du modèle.

Naturellement, les concepts qui décrivent les différents types de vins, les types principaux de mets, la notion d'une bonne alliance d'un vin et d'un plat ainsi que celle d'une mauvaise alliance figureront dans notre ontologie. En même temps, il est peu probable que l'ontologie inclue des concepts pour la gestion des stocks dans les établissements vinicoles ou la gestion des employés dans les restaurants, même si ces concepts sont, en quelque sorte, en relation avec les notions de vin et de mets.

Si l'ontologie que nous concevons est utilisée pour assister la production en langage naturel des articles dans les revues spécialisés en oenologie, il peut être important d'y inclure les synonymes des concepts ainsi que toute autre information sur les concepts, exprimée en langage parlé. Si l'ontologie est utilisée pour aider les clients des restaurants à décider quel vin commander, nous aurons besoin d'utiliser des informations sur les prix de vente au détail. Si elle est utilisée pour les grossistes en vin, des informations sur les prix de vente en gros ainsi que sur la disponibilité de la marchandise peuvent être nécessaires. Si les personnes chargées de maintenir l'ontologie décrivent le domaine dans une langue différente de la langue des utilisateurs de l'ontologie, il peut être nécessaire de faire le mapping entre les langues.

Questions de compétence

Une des méthodes pour déterminer la portée d'une ontologie est de rédiger une liste de questions auxquelles une base de connaissances fondée sur une ontologie devrait pouvoir répondre, appelées questions de compétence (Gruninger and Fox 1995). Elles serviront plus tard de test décisif : L'ontologie contient-elle suffisamment d'information pour répondre à ce type de questions ? Les réponses nécessitent-elles un niveau particulier de détail ou la représentation d'un domaine particulier ? Ces questions de compétence sont juste une ébauche qui n'a pas besoin d'être exhaustive.

Voici quelques questions de compétence possibles dans le domaine du vin et des mets :

- Sur quelles caractéristiques dois-je me fonder pour choisir un vin ?
- Le vin de Bordeaux est-il un vin rouge ou un vin blanc ?
- Un Cabernet Sauvignon peut-il accompagner les plats de fruits de mer ou de poissons ?
- Quel serait le meilleur vin pour accompagner des grillades ?
- Quelles sont les caractéristiques du vin qui affectent sur son accord avec un plat ?
- Le bouquet ou le corps d'un vin particulier varient-ils avec le millésime ?
- Quels étaient les bons millésimes pour Napa Zinfandel ?

Partant de cette liste de questions, l'ontologie comprendra l'information sur les différentes caractéristiques et les différents types de vins, les millésimes - bons ou mauvais - les classifications des mets qu'il faut prendre en compte pour faire un choix approprié et les associations recommandées de vins et de plats.

Etape 2. Envisager une éventuelle réutilisation des ontologies existantes

Il est toujours utile de prendre en considération ce que d'autres personnes ont fait et d'examiner si nous pouvons élargir des sources existantes et les affiner pour répondre aux besoins de notre domaine ou de notre tâche particulière. Réutiliser des ontologies existantes peut même constituer une exigence si notre système a besoin d'interagir avec d'autres applications qui utilisent déjà des ontologies spécifiques ou des vocabulaires contrôlés. Plusieurs ontologies sont déjà disponibles sous forme électronique et peuvent être importées dans votre propre environnement, si cet environnement permet le développement des ontologies. Le formalisme d'une ontologie n'a souvent pas d'importance, la plupart des systèmes de représentation de connaissances étant capables d'importer et d'exporter des ontologies. Même si un système de représentation de connaissances ne peut pas travailler directement avec un formalisme particulier, la traduction vers un autre formalisme ne présente généralement, pas de difficultés.

Il existe des bibliothèques d'ontologies réutilisables sur le Web et dans la littérature. Par exemple, nous pouvons utiliser la bibliothèque des ontologies Ontolingua (<http://www.ksl.stanford.edu/software/ontolingua/>) ou bien la bibliothèque des ontologies DAML (<http://www.daml.org/ontologies/>). Un certain nombre d'ontologies commerciales sont également disponibles pour le tout public (ex. UNSPSC (<http://www.unspc.org>), RosettaNet (<http://www.rosettanet.org>), DMOZ (<http://www.dmoz.org>)).

Par exemple, une base de connaissances sur les vins français peut déjà exister. Si l'on peut importer cette base de connaissances ainsi que l'ontologie sur laquelle elle est fondée, nous aurons non seulement la classification des vins français mais aussi une base pour la classification des caractéristiques utilisées pour distinguer et décrire les vins. Des listes de propriétés de vins peuvent être disponibles sur des sites commerciaux tels que www.wines.com, utilisés par les clients pour leurs achats de vins.

Toutefois pour les besoins de ce guide, nous supposons qu'il n'existe pas d'ontologies appropriées et nous commençons le travail de développement de l'ontologie à partir de l'ébauche que nous avons rédigée.

Etape 3. Enumérer les termes importants dans l'ontologie

Il est utile de noter sous forme de liste tous les termes à traiter ou à expliquer à un utilisateur. Sur quels termes souhaiterions-nous discuter ? Quels sont les propriétés de ces termes ? Que veut-on dire sur ces termes ? Par exemple, parmi les termes importants relatifs aux vins il existe : vin, cépage, établissement vinicole, localisation, couleur d'un vin,

corps, odeur et contenance en sucre ; différents types de mets, tels que poisson et viande rouge, sous-types de vin tels que vin blanc, etc. Tout d'abord, il est important d'établir une liste exhaustive de termes et de ne pas se soucier de l'éventuelle chevauchement entre les concepts qu'ils représentent, les relations entre les termes ou tout autre propriété des concepts, ni si ces concepts sont des classes ou des facettes.

Les deux étapes suivantes - développement de la hiérarchie des classes et définition des propriétés des concepts (facettes) - sont étroitement entrelacées. Il est difficile de le faire en ordre linéaire strict. D'habitude, nous créons quelques définitions de concepts dans la hiérarchie, ensuite nous procédons à la description des propriétés de ces concepts et ainsi de suite. Ces deux étapes sont aussi les plus importantes dans le processus de la construction d'une ontologie. Nous en ferons ici une brève description. Dans les deux autres sections nous aborderons les autres points plus complexes nécessitant réflexion, les pièges habituels, les décisions à prendre, etc.

Etape 4. Définir les classes et la hiérarchie des classes

Il existe un certain nombre d'approches possibles pour développer une hiérarchie de classes (Ushold and Gruninger 1996) :

- Un procédé de développement **de haut en bas** commence par une définition des concepts les plus généraux du domaine et se poursuit par la spécialisation des concepts. Par exemple nous pouvons commencer en créant des classes pour les concepts généraux *Vin* et *Mets*. Puis nous spécialisons la classe *Vin* en créant quelques unes de ses sous-classes : *Vin blanc*, *Vin rouge*, *Vin rosé*. Nous pouvons en outre catégoriser la classe *Vin rouge*, par exemple, en *Syrah*, *Bourgogne rouge*, *Cabernet Sauvignon*, et ainsi de suite.
- Un procédé de développement **de bas en haut** commence par la définition des classes les plus spécifiques, les feuilles d'une hiérarchie, et se poursuit avec le regroupement de ces classes en concepts plus généraux. Par exemple, nous pouvons commencer en définissant des classes pour les vins *Pauillac* et *Margaux*. Nous pouvons ensuite créer une super-classe commune - *Medoc* - qui à son tour est une sous-classe de *Bordeaux*.
- Une procédé **combiné** de développement est une combinaison des deux approches, de haut en bas et de bas en haut. Au tout début, les concepts les plus saillants sont définis, ensuite ils sont généralisés ou spécialisés, suivant le cas. Nous pourrions commencer par quelques concepts du haut niveau tels que *Vin* et quelques concepts spécifiques, tels que *Margaux*. Puis, on peut les mettre en relation avec d'autres concepts de niveau intermédiaire, tels que *Medoc*. Ensuite, on peut poursuivre en créant toutes les classes de vins régionaux de France, en créant également de ce fait, tout un ensemble de concepts de niveau intermédiaire.

La figure 2 montre une possibilité d'articulation entre les différents niveaux de généralité.

Aucune de ces trois méthodes n'est fondamentalement meilleure que les autres. L'approche à adopter dépend fortement du point de vue personnel sur le domaine. Si un développeur a un point de vue systématique de-haut-en-bas du domaine, il peut lui être plus commode d'utiliser l'approche de-haut-en-bas. L'approche combinée est souvent, la plus facile à utiliser pour la plupart des développeurs d'ontologies, étant donné que les concepts " du milieu " ont tendance à être les concepts les plus descriptifs du domaine (Rosch 1978).

Si vous avez tendance à penser les vins en distinguant avant tout la classification la plus généraliste, alors l'approche de-haut-en-bas peut être la meilleure pour vous. Si en revanche, vous préférez commencer en vous appuyant sur des exemples précis, l'approche de-bas-en-haut

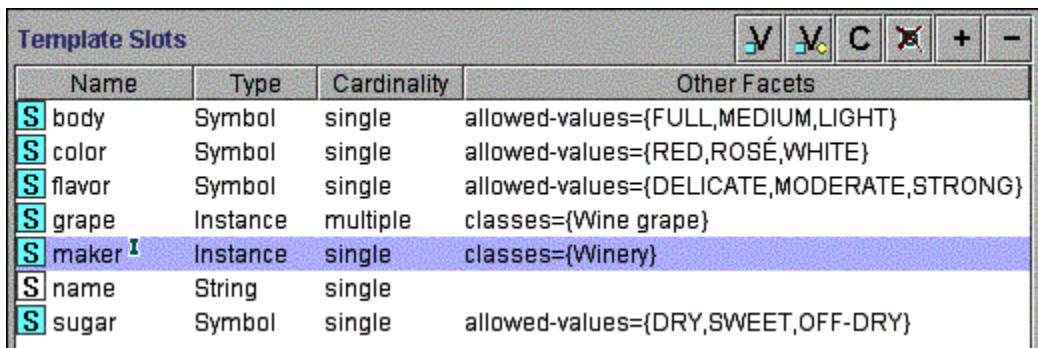
peut être plus appropriée.

Quelle que soit l'approche choisie, nous commençons habituellement par définir les classes. Dans la liste créée pendant la troisième étape, nous sélectionnons les termes qui décrivent des objets ayant une existence indépendante plutôt que les termes qui décrivent ces objets. Ces termes constitueront les classes dans l'ontologie et deviendront des points d'ancrage dans la hiérarchie des classes. Nous organisons les classes dans une taxonomie hiérarchique en nous demandant, si en étant instance d'une classe, un objet sera nécessairement (c'est à dire par définition) une instance d'une autre classe.

Si une classe A est super-classe d'une classe B, alors toute instance de B est également, une instance de A.

En d'autres termes, la classe B représente un concept qui est "une sorte" de A.

Par exemple, chaque vin Pinot Noir est obligatoirement un vin rouge. Par conséquent la classe Pinot Noir est une sous-classe de la classe Vin Rouge.



Name	Type	Cardinality	Other Facets
S body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
S color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
S flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
S grape	Instance	multiple	classes={Wine grape}
S maker ^I	Instance	single	classes={Winery}
S name	String	single	
S sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

La figure 2 montre une partie d'une hiérarchie de classes pour l'ontologie des vins. La section 4 contient une discussion détaillée de ce qu'on doit observer lorsqu'on définit une hiérarchie de classes.

Figure 3. Les attributs pour la classe Vin et les facettes pour ces attributs. L'icône " I " à côté de l'attribut producteur indique que l'attribut possède un inverse (Section 5.1)

Étape 5. Définir les propriétés des classes - attributs

Les classes seules ne fourniront pas assez d'information pour répondre aux questions de compétence de l'Étape 1. Après avoir défini quelques classes, nous devons décrire la structure interne des concepts.

Nous avons déjà sélectionné des classes à partir de la liste des termes que nous avons créée pendant l'Étape 3. La plupart des termes restants ont de fortes chances d'être des propriétés de ces classes. Ces termes comprennent, par exemple, la couleur d'un vin, son corps, son odeur et sa teneur en sucre ainsi que la localisation de l'établissement vinicole. Pour chaque propriété dans la liste nous devons déterminer la classe qu'elle décrit. Ces propriétés deviennent des attributs rattachés aux classes. Ainsi, la classe Vin aura les attributs suivants : couleur, corps, odeur, et sucre. Et la classe Etablissement vinicole aura l'attribut localisation.

En général, certains types de propriétés d'objets peuvent devenir des attributs dans une ontologie.

- propriétés " intrinsèques " telle que l'odeur d'un vin ;

- propriétés “ extrinsèques ” telles que le nom d’un vin et son terroir;
- parties, si l’objet est structuré ; elles peuvent être des “ parties ” physiques ou abstraites (ex : les plats d’un repas)
- relations avec d’autres individus ; ce sont les relations entre les membres individuels d’une classe et les autres entités (par exemple, le producteur d’un vin représentant une relation entre un vin et un établissement vinicole, et le cépage d’origine.)

Par conséquent, outre les propriétés déjà identifiées, il nous faut ajouter les attributs suivants pour la classe `Vin` : `nom`, `terroir`, `producteur`, `cépage`. La figure 3 indique les attributs pour la classe `Vin`.

Toutes les sous-classes d’une classe **héritent** les attributs de cette classe. Par exemple, tous les attributs de la classe `Vin` seront hérités par toutes les sous-classes de la classe `Vin`, y compris `Vin Rouge` et `Vin Blanc`. Nous ajouterons l’attribut supplémentaire `niveau de tannin` (bas, modéré, élevé) à la classe `Vin Rouge`. L’attribut `niveau de tannin` sera hérité par toutes les classes représentant des vins rouges (telles que `Bordeaux` et `Beaujolais`).

Un attribut doit être rattaché à la classe la plus générale pouvant avoir cette propriété. Par exemple, `corps` et `couleur` d’un vin doivent être rattachés à la classe `Vin`, puisque c’est la classe la plus générale dont les instances auront un corps et une couleur.

Étape 6. Définir les facettes des attributs

Les attributs peuvent avoir plusieurs facettes décrivant la valeur du type, les valeurs autorisées, le nombre de valeurs (cardinalité), et d’autres caractéristiques de valeurs que les attributs peuvent avoir. Par exemple, la valeur de l’attribut `nom` (comme dans "le nom d'un vin") est une chaîne de caractères. C'est à dire, `nom` est un attribut ayant le type de valeur : Chaîne de caractères. L’attribut `produit` (comme dans "un établissement vinicole produit tels vins") peut avoir de multiples valeurs et ces valeurs sont des instances de la classe `Vin`. C'est à dire, `produit` est un attribut ayant pour type de valeur `Instance` et pour classe autorisée `Vin`.

Nous allons, maintenant, décrire quelques facettes communes.

La cardinalité des attributs

La cardinalité des attributs définit le nombre de valeurs qu’un attribut peut avoir. Certains systèmes distinguent entre cardinalité unique (n'autorisant qu'une seule valeur) et cardinalité multiple (autorisant une ou plusieurs valeurs). Le `corps` d'un vin sera un attribut ayant une cardinalité unique (un vin ne peut avoir qu'un seul corps). Les vins produits par un établissement vinicole particulier complètent l’attribut `produit` qui a de multiples cardinalités pour la classe `Établissement vinicole`.

Certains systèmes permettent de spécifier une cardinalité minimale et maximale pour décrire plus précisément le nombre de valeurs d’un attribut. Cardinalité minimale N veut dire qu'un attribut doit avoir au moins N valeurs. Par exemple, l’attribut `cépage` pour un vin doit avoir une cardinalité minimale de 1: chaque vin est fabriqué à partir d'au moins une variété de raisin. Cardinalité maximale M veut dire qu'un attribut peut avoir au maximum M valeurs. La cardinalité maximale pour l’attribut `cépage` pour les vins d'une seule variété de raisin est 1 : ces vins sont produits à partir d'une seule variété de raisin. Il est parfois utile de fixer la cardinalité maximale à 0. Cette définition voudrait dire que l’attribut ne peut pas avoir de valeurs pour une sous-classe particulière.

Le type de valeur des attributs

La facette type de valeur décrit les types de valeurs pouvant être affectés à l'attribut. Voici une liste des types de valeurs les plus typiques :

- **Chaîne de caractères** est le type de valeur le plus simple utilisé pour des attributs tels que `nom` : la valeur est une simple chaîne de caractères
- **Nombre** (quelquefois des types de valeurs `Enveloppe` et `Entier` plus spécifiques sont utilisés) décrit des attributs ayant des valeurs numériques. Par exemple, le prix d'un vin peut avoir le type de valeur `Enveloppe`.
- Les attributs **Booléens** sont des icônes simples de type oui - non. Par exemple, si nous choisissons de ne pas représenter les vins pétillants comme une classe distincte, que le vin soit pétillant ou non, cela peut être représenté comme une valeur d'un attribut Booléen : si la valeur est "vrai" ("oui") le vin est pétillant et si la valeur est "faux" ("non") le vin ne l'est pas.
- Les attributs **Énumérés** précisent une liste de valeurs spécifiques autorisées pour l'attribut. Par exemple, nous pouvons spécifier que l'attribut `odeur` peut avoir une valeur sur trois valeurs possibles : `forte`, `modérée` ou `délicate`. Dans `Protégé-2000` les attributs énumérés sont de type `Symbol`.
- Les attributs de type **Instance** permettent la définition des relations entre individus. Les attributs ayant pour type de valeur `Instance`, imposent aussi la définition d'une liste de classes autorisées desquelles les instances sont issues. Par exemple l'attribut `produit` pour la classe `Établissement vinicole` peut avoir comme valeurs les instances de la classe `Vin`.

La figure 4 illustre une définition de l'attribut `produit` dans la classe `Établissement vinicole`.

Le domaine et le rang d'un attribut

Les classes autorisées pour les attributs de type `Instance` sont souvent appelées **étendue** d'un attribut. Dans l'exemple de la figure 4 la classe `Vin` est l'étendue de l'attribut `produit`. Certains systèmes permettent la limitation de l'étendue d'un attribut quand l'attribut est rattaché à une classe particulière.

Les classes auxquelles un attribut est rattaché ou les classes dont l'attribut décrit les propriétés, sont appelées le **domaine** d'un attribut. La classe `Établissement vinicole` est le domaine de l'attribut `produit`. Dans les systèmes où les attributs sont *rattachés* aux classes, les classes auxquelles l'attribut est rattaché constituent le domaine de cet attribut. Il n'y a pas besoin de spécifier le domaine séparément.

Les règles de base pour déterminer le domaine et l'étendue d'un attribut sont similaires :

Pour définir le domaine ou l'étendue d'un attribut, chercher la ou les classes les plus générales qui peuvent être respectivement le domaine ou l'étendue pour les attributs.

*En revanche, ne pas définir un domaine ou une étendue trop généraux : toutes les classes du domaine d'un attribut doivent être décrites par l'attribut et les instances de toutes les classes de l'étendue d'un attribut doivent être des valeurs potentielles de remplissage pour l'attribut. Ne pas choisir une classe trop générale pour l'étendue (c'est à dire, il est inutile de créer une étendue **CHOSE**) mais il est possible de choisir une classe qui couvre toutes les valeurs de remplissage.*

Au lieu de lister toutes les sous-classes possibles de la classe `Vin` pour l'étendue de l'attribut `produit`, on listera tout simplement `Vin`. En même temps, nous ne souhaitons pas, non plus,

spécifier l'étendue de l'attribut en tant que CHOSE - la classe la plus générale dans une ontologie.

En termes clairs :

Si une liste de classes définissant une étendue ou un domaine d'un attribut, comprend une classe et sa sous-classe, retirer la sous-classe.

Si l'étendue de l'attribut contient à la fois la classe `Vin` et la classe `Vin Rouge`, nous pouvons en retirer le `Vin Rouge`, car cela n'apporte pas de nouvelle information : le `Vin Rouge` est une sous-classe de `Vin` et par conséquent est implicitement compris dans l'étendue de l'attribut, de la même manière que les autres sous-classes de la classe `Vin`.

Si une liste de classes définissant une étendue ou un domaine d'un attribut contient toutes les sous-classes d'une classe A, mais pas la classe A elle-même, l'étendue devrait contenir uniquement la classe A mais non les sous-classes.

Au lieu de définir l'étendue d'un attribut pour inclure `Vin Rouge`, `Vin Blanc`, et `Vin Rosé` (par énumération de toutes les sous-classes directes de `Vin`), nous pouvons limiter l'étendue à la classe `Vin` elle-même.

Si une liste de classes définissant une étendue ou un domaine d'un attribut contient seulement quelques sous-classes de la classe A, il faut reconsidérer la nécessité d'une définition plus appropriée pour l'étendue de la classe A.

Dans les systèmes où le rattachement d'un attribut à une classe revient à ajouter la classe au domaine de l'attribut, les mêmes règles s'appliquent au rattachement de l'attribut : d'une part il faut essayer de le rendre aussi général que possible et d'autre part il faut s'assurer que chacune des classes à laquelle un attribut est rattaché peut réellement avoir la propriété représentée par l'attribut. L'attribut `niveau de tanin` peut être rattaché à toute classe qui représente des vins rouges (ex. `Bordeau`, `Merlot`, `Beaujolais`, etc). Cependant, étant donné que tous les vins rouges ont la propriété `niveau de tanin`, nous devons rattacher cet attribut à la classe plus générale des `Vins Rouges`. En revanche, il ne serait pas correct de généraliser davantage le domaine de l'attribut `niveau de tanin` (en le rattachant plutôt à la classe `Vin`), puisque le `niveau de tanin` n'est pas utilisé pour décrire les vins blancs, par exemple.

Étape 7. Créer les instances

La dernière étape consiste à créer les instances des classes dans la hiérarchie. Définir une instance individuelle d'une classe exige (1) choisir une classe, (2) créer une instance individuelle de cette classe, et (3) la renseigner avec les valeurs des attributs. Par exemple, nous pouvons créer une instance individuelle `Château-Morgon-Beaujolais` pour représenter un type spécifique des vins `Beaujolais`. `Château-Morgon-Beaujolais` est une instance de la classe `Beaujolais` qui, à son tour, représente tous les vins `Beaujolais`. Cette instance a les valeurs d'attributs suivantes (Figure 5) :

- Corps : Léger
- Couleur : Rouge
- Odeur : Délicate
- Niveau de tanin : Bas
- Cépage : Gamay (instance de la classe `Raisin (wine grape)`)
- Producteur : Château-Morgon (instance de la classe `Établissement viticole`)
- Région : Beaujolais (instance de la classe `Région viticole`)
- Sucre : Sec



4. Définir les classes et la hiérarchie des classes

Cette partie attire l'attention sur les points qu'il faut observer et les erreurs les plus fréquentes lors de la définition des classes et de la hiérarchie des classes. (Etape 4 de la Section 3). Comme nous l'avons mentionné précédemment, il n'existe pas qu'une seule hiérarchie correcte de classes pour un domaine donné. La hiérarchie dépend des utilisations possibles de l'ontologie, du niveau de détail nécessaire pour l'application, des préférences personnelles et parfois des exigences de compatibilité avec d'autres modèles. Toutefois, nous avancerons quelques recommandations qu'il faut observer lorsqu'on développe une hiérarchie de classes. Après avoir défini un nombre considérable de classes, il est utile de prendre du recul et de vérifier si la hiérarchie émergente est conforme à ces recommandations.

4.1 S'assurer que la hiérarchie des classes est correcte

Une relation de type " est-un "

La hiérarchie des classes représente une relation de type " est-un " : une classe A est une sous-classe de B si chaque instance de A est également une instance de B. Par exemple, Chardonnay est une sous-classe de Vin blanc. Une autre manière de voir la relation taxinomique est de la voir comme relation du genre "une-sortre-de" : Chardonnay est une sorte de Vin blanc. Un avion de ligne est une sorte d'avion. La viande est une sorte d'aliment.

Une sous-classe d'une classe représente un concept qui est " une-sortre-du " concept représenté par la super-classe.

Un seul vin n'est pas une sous-classe de tous les vins

Une erreur très fréquente de modélisation est d'inclure les deux versions, au singulier et au pluriel, du même concept dans la hiérarchie, faisant ainsi de la première une sous-classe de la deuxième. Par exemple, il est faux de définir une classe Vins et une classe Vin comme sous-classe de Vins. L'erreur de modélisation devient claire dès qu'on pense à la hiérarchie comme un outil représentant la relation " une-sortre-de " : un Vin particulier n'est pas **une sorte de** Vins. Le meilleur moyen pour éviter une telle erreur est d'utiliser toujours et exclusivement, soit le singulier, soit le pluriel dans la nomination des classes (voir la Partie 6 pour la discussion

sur la nomination des concepts).

Transitivité des relations hiérarchiques

Une relation de sous-classe est transitive :

Si B est une sous-classe de A et C est une sous-classe de B, alors C est une sous-classe de A.

Par exemple, nous pouvons définir une classe `Vin` et, par la suite, définir une classe `Vin blanc`. Transitivité dans la relation de sous-classe veut dire que la classe `Chardonnay` est également une sous-classe de `Vin`. Quelque fois nous distinguons entre les sous-classes directes et les sous-classes indirectes. Une **sous-classe directe** est la sous-classe la plus “ proche ” de la classe : il n’y a pas de classe entre une classe et sa sous-classe directe dans une hiérarchie. C’est à dire, il n’y a pas d’autre classe dans la hiérarchie entre une classe et sa super-classe directe. Dans notre exemple, `Chardonnay` est une sous-classe directe de `Vin blanc` et n’est pas une sous-classe directe de `Vin`.

Évolution d’une hiérarchie de classes

Maintenir une hiérarchie cohérente de classes peut devenir un vrai défi, en raison de l’évolution des domaines. Par exemple, pendant de nombreuses années, tous les vins `Zinfandel` ont été des vins rouges. Par conséquent, nous définissons une classe de vins `Zinfandel` comme sous-classe de la classe `Vin rouge`. Il arrive parfois que les producteurs pressent le raisin et en extraient immédiatement les éléments colorants modifiant ainsi la couleur du vin obtenu. Par ce procédé, ils arrivent à produire le "Zinfandel blanc" dont la couleur est rose. Par conséquent nous devons diviser la classe `Zinfandel` en deux classes - `Zinfandel blanc` et `Zinfandel Rouge` - en les classifiant respectivement comme sous-classe de `Vin rosé` et sous-classe de `Vin rouge`.

Les classes et leurs noms

Il est important de distinguer entre une classe et son nom :

Les classes représentent des concepts dans le domaine et non pas des mots désignant ces concepts.

Le nom d'une classe peut varier suivant la terminologie choisie, mais le terme lui-même représente la réalité objective du monde. Par exemple, nous pouvons créer une classe `Salicoques` et le rebaptiser ensuite `Crevettes` - la classe représente toujours le même concept. Les associations appropriées de vin et de plats de salicoques devraient se référer aux plats de crevettes. Plus concrètement, la règle suivante devrait toujours être suivie :

Les synonymes pour le même concept ne représentent pas de classes différentes.

Les synonymes sont juste des noms différents pour un concept ou un terme. Donc, nous **ne devrions pas** avoir une classe appelée `Crevette` et une classe appelée `Salicoque`. Il y aura une seule classe, nommée soit `Crevette` soit `Salicoque`. Beaucoup de systèmes permettent d'associer à une classe une liste de synonymes, de traductions, ou de noms de présentation. Si un système ne permet pas ces associations, les synonymes peuvent toujours être listés dans la documentation sur la classe.

Éviter les boucles de classes

Nous devons éviter des **boucles** dans la hiérarchie d’une classe. On dit qu’il y a une boucle dans une hiérarchie quand une classe A a une sous-classe B et qu’en même temps B est une super-

classe de A. Créer une telle boucle dans une hiérarchie revient à déclarer que les classes A et B sont équivalentes : toutes les instances de A sont des instances de B et toutes les instances de B sont aussi des instances de A. En fait, puisque B est une sous-classe de A, toutes les instances de B doivent être des instances de la classe A. Comme A est une sous-classe de B, toutes les instances de A doivent aussi être des instances de la classe B.

4.2 Analyse des fratries (entités soeurs) dans la hiérarchie des classes

Fratries (entités soeurs) dans une hiérarchie de classes

Les fratries dans la hiérarchie sont des classes qui sont les sous-classes directes de la même classe (voir la Partie 4.1).

Toutes les filles de la même mère dans la hiérarchie (sauf celles à la racine) doivent être au même niveau de généralité.

Par exemple, `Vin blanc` et `Chardonnay` ne doivent pas être les sous-classes de la même classe (disons de la classe `Vin`). `Vin blanc` est un concept plus général que `Chardonnay`. Les enfants de mêmes parents doivent être des concepts "appartenant à la même génération" de même que les chapitres de même niveau dans un livre représentent le même niveau de généralité. Dans ce sens, les conditions requises pour une hiérarchie de classes sont semblables aux conditions requises pour la structuration d'un livre.

Cependant, les concepts à la racine de la hiérarchie (et qui sont souvent représentés comme les sous-classes directes d'une classe très générale, comme par exemple `Chose`) représentent les divisions principales du domaine et ne doivent pas être des concepts semblables.

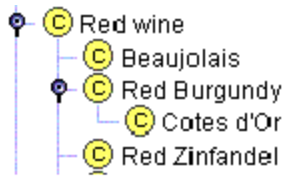
Comment se fait-il que beaucoup soit trop et peu pas assez ?

Il n'y a pas de règle rigide concernant le nombre de sous-classes directes qu'une classe doit avoir. Néanmoins, la plupart des ontologies bien structurées comportent entre deux et une douzaine de sous-classes directes. C'est pourquoi nous observons les deux lignes directrices suivantes :

Si une classe a seulement une sous-classe directe il est possible qu'il y ait un problème de modélisation ou bien l'ontologie n'est pas complète.

S'il y a plus d'une douzaine de sous-classes pour une classe donnée, alors des catégories intermédiaires, supplémentaires peuvent être nécessaires.

La première de ces deux règles est semblable à la règle de typographie qui dit qu'une liste à puces ne doit jamais n'avoir qu'une seule puce. Par exemple, la plupart des vins de Bourgogne rouges sont des vins Côte-d'Or. Supposons que nous ayons voulu représenter seulement ce type majoritaire de vins de Bourgogne : nous pourrions créer une classe `Bourgogne rouge` et ensuite une sous-classe simple `Côte-d'Or` (Figure 6a). Cependant, si dans notre représentation le `Bourgogne rouge` et le `Côte-d'Or` sont des vins fondamentalement équivalents (tous les vins rouges de Bourgogne sont des Côte-d'Or et tous les vins Côte-d'Or sont des vins rouges de Bourgogne), il est inutile de créer la classe `Côte-d'Or` car cela n'ajoute pas de nouvelle information à la représentation. Si nous devons inclure les vins Côte-Chalonnaise, qui sont des vins de Bourgogne de la région juste au sud de la Côte-d'Or mais meilleur marché que les précédents, nous créerions deux sous-classes de la classe de Bourgogne : `Côte-d'Or` et `Cote-Chalonnaise` (Figure 6b).



Supposons maintenant que nous inscrivions tous les types de vins comme des sous-classes directes de la classe `Vin`. Cette liste inclurait alors des types généraux de vins comme Beaujolais et Bordeaux, aussi bien que des types plus spécifiques comme Pauillac et Margaux (Figure 7a). La classe `Vin` comporte trop de sous-classes directes et il paraît raisonnable que, pour refléter les types différents de vins de façon plus organisée pour l'ontologie, `Médoc` soit une sous-classe de `Bordeaux` et `Côtes-d'Or` une sous-classe de `Bourgogne`. Avoir de la sorte des catégories intermédiaires comme `Vin rouge` et `Vin blanc` refléterait le modèle conceptuel qu'a la majorité des personnes lambda du domaine des vins (Figure 7b). Toutefois, s'il n'existe aucune classe naturelle pour grouper les concepts de la longue liste des entités filles, il n'y a nul besoin de créer des classes artificielles - il faut donc laisser les classes telles qu'elles sont. Après tout, l'ontologie est un reflet du monde réel et si aucune catégorisation n'existe dans le monde réel, alors l'ontologie devrait le refléter.

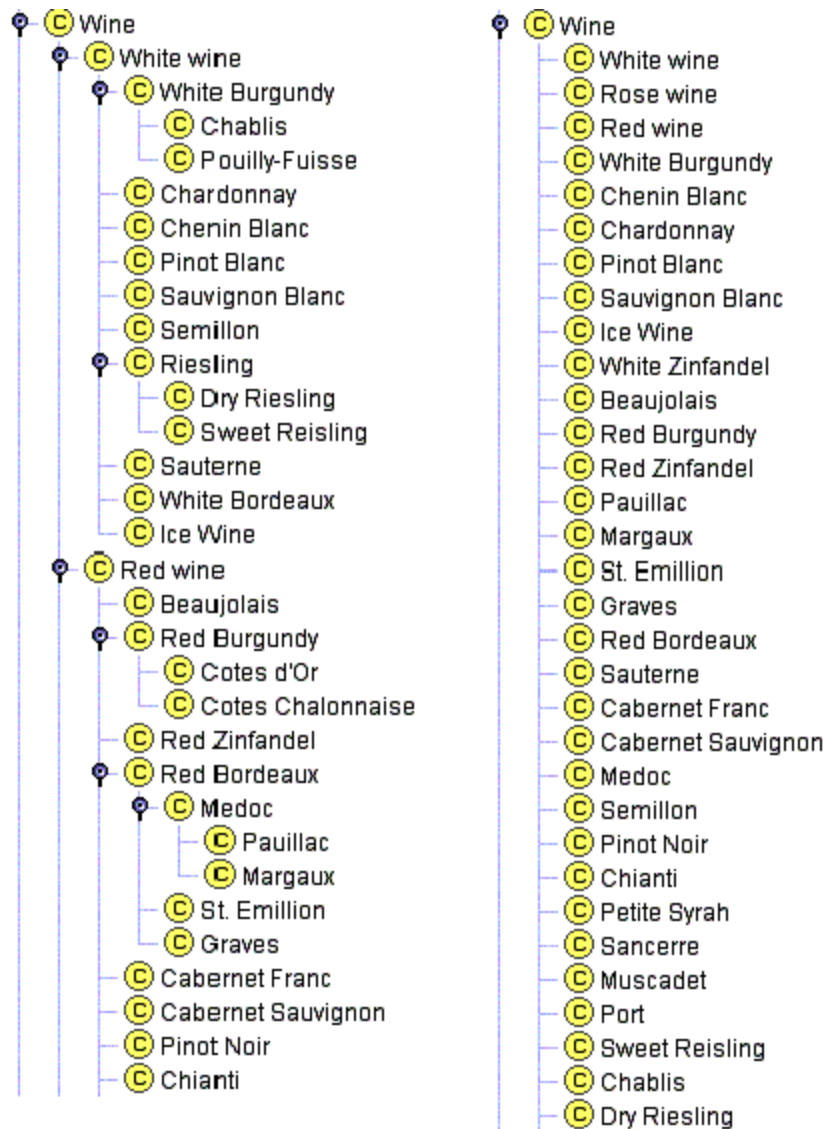


Figure 7. Catégorisation des vins. Avoir tous les vins et tous les types de vins contre Avoir plusieurs niveaux de catégorisation.

4.3 Héritages multiples

La plupart des systèmes de représentation des connaissances permettent l'héritage multiple dans la hiérarchie des classes : une classe peut être une sous-classe de plusieurs classes. Supposons que nous voulons créer une classe distincte pour les vins de dessert, la classe `Vin doux`. Le vin de Porto est à la fois un vin rouge et un vin doux. Par conséquent, nous définissons une classe `Porto` pour avoir deux super-classes : `Vin rouge` et `Vin doux`. Toutes les instances de la classe `Porto` seront aussi bien des instances de la classe `Vin rouge` que de la classe `Vin doux`. La classe `Porto` héritera les attributs et les facettes des attributs de ses deux parents. Ainsi, elle héritera la valeur `DOUX` pour l'attribut de la classe `Vin doux` et l'attribut `Niveau de tanin` et la valeur de son attribut `couleur` de la classe `Vin rouge`.

4.4 A quel moment faut-il introduire (ou non) une nouvelle classe

Une des décisions les plus difficiles à prendre pendant la modélisation est la suivante : quand faut-il introduire une nouvelle classe, ou bien : quand faut-il représenter une distinction par des valeurs différentes de propriété. Il est aussi difficile de naviguer dans une hiérarchie comportant plusieurs niveaux d'emboîtements de classes superflues que dans une hiérarchie très plate avec un nombre trop réduit de classes et comportant trop d'information codée dans les attributs. Trouver le juste milieu n'est pas une affaire simple.

Quelques principes de base permettent de décider à quel moment il faut introduire de nouvelles classes dans une hiérarchie.

Les sous-classes d'une classe (1) possèdent habituellement des propriétés complémentaires que ne possède pas la super-classe, ou (2) des restrictions différentes de celles de la super-classe, ou (3) entretiennent des relations différentes de celles que les super-classes peuvent entretenir.

Les vins rouges peuvent avoir des niveaux différents de tanin, alors que cette propriété n'est pas utilisée pour décrire les vins en général. La valeur pour l'attribut sucre du Vin de dessert est DOUX, alors qu'on ne peut pas en dire autant de la super-classe de la classe Vin doux. Les vins Pinot Noir peuvent très bien accompagner des plats de fruits de mer, tandis que d'autres vins rouges ne le peuvent pas. En d'autres termes, nous introduisons une nouvelle classe dans la hiérarchie, seulement quand il y a quelque chose à dire de cette classe qu'on ne peut pas dire de sa super-classe. Concrètement, chaque sous-classe doit : soit avoir de nouveaux attributs rattachés, soit de nouvelles valeurs d'attributs définies, soit outrepasser certaines facettes concernant les attributs hérités.

Parfois, il peut être pourtant utile de créer de nouvelles classes, même si elles ne présentent pas de nouvelles propriétés

Dans les hiérarchies terminologiques les classes n'ont pas besoin d'introduire de nouvelles propriétés.

Par exemple, quelques ontologies incluent de grandes hiérarchies de référence concernant les termes communs employés dans un domaine. Par exemple, une ontologie qui constitue le socle d'un système électronique de fichiers médicaux peut inclure une classification de maladies diverses. Cette classification peut être juste une hiérarchie de termes, sans propriétés (ou avec le même jeu de propriétés). Dans ce cas, il est toujours utile d'organiser les termes en une hiérarchie plutôt qu'en une liste plate car cela (1) facilitera l'exploration et la navigation (2) permettra au médecin de choisir le niveau de généralité du terme approprié à la situation.

Une autre raison pour introduire de nouvelles classes sans nouvelles propriétés est de pouvoir modéliser certains concepts que les experts de domaine ont l'habitude de distinguer, sans qu'il soit jugé utile de modéliser la distinction elle-même. Étant donné que les ontologies sont employées pour faciliter la communication entre les experts de domaines d'une part, et d'autre part entre les experts de domaines et les systèmes basés sur les connaissances, il est souhaitable d'y refléter l'avis de l'expert du domaine.

Finalement, nous ne devrions pas créer des sous-classes d'une classe pour toute restriction supplémentaire. Par exemple, nous avons introduit les classes Vin rouge, Vin blanc et Vin rosé parce que cette distinction est naturelle dans le monde des vins. Nous n'avons pas introduit de classes pour le vin délicat, le vin moyen, etc. En définissant une hiérarchie de classes, notre but est de trouver le juste milieu entre : créer de nouvelles classes utiles pour l'organisation des classes et créer trop de classes.

4.5 Une nouvelle classe ou une valeur de propriété ?

Lors de la modélisation d'un domaine, nous devons souvent décider si la modélisation d'une distinction spécifique (comme vin blanc, rouge, ou rosé), en tant que valeur de propriété ou en tant qu'ensemble de classes, dépend de nouveau des contours du domaine et de la tâche fixée.

Devons-nous créer une classe `Vin blanc` ou bien simplement une classe `Vin` et renseigner les différentes valeurs de l'attribut `couleur` ? La réponse se trouve, en général, dans les contours que nous avons définis pour notre ontologie. Quel importance le concept `Vin blanc` a-t-il dans notre domaine ? Si les vins n'ont qu'une importance marginale dans le domaine, et si le fait d'être blanc ou non n'a pas d'implications particulières pour les relations d'un vin avec d'autres objets, alors nous ne devons pas introduire de classe distincte pour les vins blancs. Pour un modèle de domaine utilisé dans une usine produisant des étiquettes de vins, les règles concernant la fabrication des étiquettes de vins de toute couleur sont toujours les mêmes et la distinction des vins par la couleur n'a pas de grande importance. D'autre part, pour la représentation du vin, des mets et de leurs alliances appropriées, un vin rouge diffère considérablement d'un vin blanc : il est associé avec d'autres plats, possède d'autres propriétés, et ainsi de suite. De la même façon, la couleur du vin est importante pour une base de connaissances sur les vins et qui peut être employé pour déterminer un ordre d'éléments à respecter dans la procédure de dégustation de vin. Pour cette raison, nous créons une classe distincte pour le `Vin blanc`.

Si des concepts ayant des valeurs distinctes d'attributs deviennent des restrictions pour des attributs distincts dans d'autres classes, alors nous devons créer une nouvelle classe pour représenter la distinction. Sinon, il faut représenter la distinction dans une valeur d'attribut.

De même, notre ontologie sur le vin a des classes telles que `Merlot Rouge` et `Merlot Blanc`, plutôt qu'une classe unique pour tous les Merlots : Merlots rouge et Merlots blancs sont des vins réellement différents (bien qu'issus du même cépage) et si nous développons une ontologie détaillée sur le vin, cette distinction est importante.

Si une distinction est importante dans le domaine et que nous traitons les objets ayant des valeurs différentes pour cette distinction comme des objets de types différents, alors nous devons créer une nouvelle classe pour la distinction.

La prise en compte des cas individuels potentiels d'une classe peut aussi être utile pour décider s'il faut introduire ou non une nouvelle classe.

Une classe qui comporte une instance individuelle ne devrait pas changer souvent.

D'ordinaire, quand nous utilisons les propriétés extrinsèques des concepts plutôt que les propriétés intrinsèques pour différencier les classes, les instances de ces classes doivent souvent migrer d'une classe à une autre. Par exemple, `Vin rafraîchi` ne devrait pas être une classe dans une ontologie décrivant des bouteilles de vin dans un restaurant. La propriété `rafraîchi` devrait simplement être un attribut du vin en bouteille, étant donné qu'une instance de `Vin rafraîchi` peut, alternativement et facilement, cesser d'être une instance de cette classe et le redevenir.

D'habitude les numéros, les couleurs, les emplacements sont des valeurs d'attributs et n'induisent pas la création de nouvelles classes. Toutefois, le vin est une notable exception, tant la couleur du vin est primordiale pour sa description.

A titre d'autre exemple, considérons l'ontologie de l'anatomie humaine. Pour représenter les côtes, faut-il créer une classe particulière pour chacune des " 1^{ère} côte gauche", " 2^{ème} côte gauche", etc. ? Ou vaut-il mieux avoir une classe `Côte` avec des attributs pour l'ordre et la position latérale (droite - gauche) ? Si l'information sur chacune des côtes que nous représentons

dans l'ontologie diffère de façon significative, alors nous devons, en effet, créer une classe pour chacune des côtes. C'est-à-dire que, si nous voulons représenter la contiguïté des détails et l'information sur l'emplacement (qui est différent pour chaque côte) aussi bien que des fonctions spécifiques de chaque côte et les organes qu'elle protège, nous aurons besoin de classes. Si nous modélisons l'anatomie à un niveau de généralité légèrement inférieure et que dans nos applications potentielles toutes les côtes soient à peu-près semblables (il s'agit juste de voir aux rayons X quelle côte serait cassé, sans implication pour les autres parties du corps) nous pourrions simplifier notre hiérarchie et n'avoir que la classe Côte et deux attributs : `position latérale` et `ordre`.

4.6 Une instance ou une classe

Décider si un concept particulier est une classe ou une instance individuelle dans une ontologie dépend des applications potentielles de l'ontologie. Trancher sur : où finissent les classes et où commencent les instances individuelles, commence par la définition du niveau le plus bas de granularité dans la représentation. Le niveau de granularité est à son tour défini par l'application potentielle de l'ontologie. Autrement dit, quelles sont les entités les plus spécifiques qui seront représentées dans la base de connaissances ? Si l'on fait appel aux questions de compétence telles que nous les avons identifiées dans l'Étape 1 de la troisième Partie, les concepts les plus spécifiques qui constitueront des réponses à ces questions sont de très bons candidats pour devenir des individus dans la base de connaissances.

Les instances individuelles sont les concepts les plus spécifiques représentés dans une base de connaissance.

Par exemple, si nous devons parler seulement d'accord des vins avec des mets, nous ne serons pas intéressés par les bouteilles physiques particulières de vin. Donc, des termes tels que Merlot des Vignobles de Sterling seront probablement les termes les plus spécifiques que nous utiliserons. En d'autres termes, la classe Vin rassemble non pas des bouteilles individuelles de vins mais des vins particulières produits par des établissements viticoles particuliers. Donc, le Merlot des Vignobles de Sterling serait une instance dans la base de connaissances.

Par ailleurs, si outre la base de connaissances sur l'accord des vins avec des mets, nous souhaitons maintenir un inventaire des vins dans le restaurant, alors les bouteilles individuelles de chaque vin peuvent devenir des instances individuelles dans notre base de connaissances. De même, si nous souhaitons enregistrer les propriétés différentes de chaque millésime spécifique du Merlot des Vignobles de Sterling, alors tout millésime spécifique de ce vin sera une instance dans la base de connaissances et le Merlot des Vignobles de Sterling sera une classe contenant des instances pour toutes ses millésimes.

Une autre règle peut "déplacer" quelques instances individuelles dans l'ensemble des classes :

Si les concepts forment une hiérarchie naturelle, alors ils doivent être représentés comme des classes.

Considérons maintenant les régions viticoles. Au commencement, nous pouvons définir comme classes les principales régions viticoles, telles que : la France, les États-Unis, l'Allemagne, etc., et comme instances, les régions viticoles spécifiques à l'intérieur de ces grandes régions. Par exemple, la région Bourgogne est une instance de la classe Région France. Cependant, nous aimerions aussi dire que la région Côte-d'Or est une région de la Région Bourgogne. Par conséquent, la Région Bourgogne doit être une classe (de façon à ce qu'elle puisse comporter des sous-classes ou des instances). Pourtant, il paraît arbitraire de traiter la Région Bourgogne comme classe et la Région Côte-d'Or comme instance de la Région Bourgogne : il est très difficile de distinguer clairement

quelles régions sont des classes et lesquelles sont des instances. C'est pourquoi nous définissons toutes les régions viticoles comme des classes. Protégé-2000 permet aux utilisateurs de spécifier quelques classes comme Abstraites, signifiant que la classe ne peut pas avoir d'instances directes. Dans notre cas, toutes les classes représentant des régions sont abstraites (Figure 8).

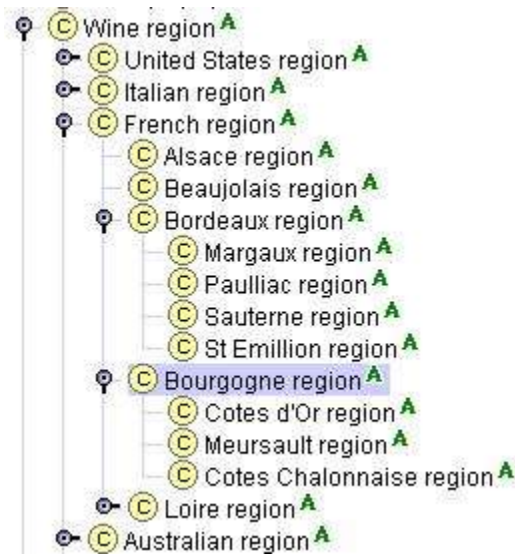


Figure 8. Hiérarchie des régions viticoles. Les icônes "A" à droite des noms des classes indiquent que les classes sont abstraites et ne peuvent pas avoir d'instances directes

La même hiérarchie de classes serait incorrecte si nous avions omis le mot "région" des noms de classes. Nous ne pouvons pas dire que la classe *Alsace* est une sous-classe de la classe *France* : l'Alsace n'est pas une sorte de France. Et pourtant, la région d'Alsace est une sorte de région française.

Dans une hiérarchie, seules les classes peuvent être agencées - les systèmes de représentation de connaissances n'ont pas de notion de sous-instance. Pour cette raison, s'il y a une hiérarchie naturelle parmi les termes, comme dans les hiérarchies terminologiques de la Partie 4.2, nous devons définir ces termes en tant que classes bien qu'ils ne puissent pas comporter leur propres instances.

4.7 Limiter la portée

A titre de dernière remarque dans la définition d'une hiérarchie de classes, les règles suivantes sont toujours utiles pour conclure si la définition d'une ontologie est complète :

L'ontologie ne doit pas contenir toute l'information possible sur le domaine : vous ne devez pas spécialiser (ou généraliser) plus que de besoin pour votre application (au maximum un niveau supplémentaire de chaque côté).

Pour notre exemple de vin et de mets, nous n'avons pas besoin de savoir quel type de papier est utilisé pour les étiquettes ou comment cuisiner les crevettes.

De façon similaire :

L'ontologie ne doit pas contenir toutes les propriétés possibles des classes et toutes les distinctions entre les classes dans la hiérarchie.

Dans notre ontologie, nous n'incluons certainement pas toutes les propriétés d'un vin ou d'un aliment. Nous y avons représenté les propriétés les plus saillantes des classes d'entités. Bien que les livres sur les vins mentionnent la taille du raisin, nous n'avons pas inclus cette connaissance dans notre ontologie. De même, nous n'avons pas ajouté toutes les relations que l'on pourrait

imaginer entre tous les termes dans notre système. Par exemple, nous n'incluons pas de relations comme `vin préféré` et `plat préféré` dans l'ontologie uniquement pour satisfaire une représentation plus complète de toutes les intercommunications possibles entre les termes que nous avons définis.

La dernière règle s'applique également pour établir des relations entre les concepts que nous avons déjà inclus dans l'ontologie. Prenons le cas d'une ontologie décrivant les expériences en biologie. L'ontologie comportera probablement, un concept `Organismes biologiques`. Elle comportera aussi un concept `Expérimentateur` pour la personne qui conduit une expérience (avec son nom, affiliation, etc.). Il se trouve qu'un expérimentateur, comme personne, est également un organisme biologique. Pourtant, nous ne devons probablement pas incorporer cette distinction dans l'ontologie : pour les besoins de cette représentation un expérimentateur n'est pas un organisme biologique et vraisemblablement, nous ne pratiquerons jamais des expériences sur les expérimentateurs eux-mêmes. Si dans l'ontologie nous représentons tout ce que nous pouvons dire sur les classes, un `Expérimentateur` deviendra une sous-classe de `Organisme biologique`. Néanmoins, nous n'avons pas besoin d'inclure cette connaissance pour les applications prévisibles de l'ontologie. En fait, comprendre ce type de classification complémentaire pour les classes existantes ne se fait pas sans provoquer de nuisance. Notamment, une instance d'`Expérimentateur` aura les attributs poids, âge, espèce ainsi que d'autres informations appartenant à un organisme biologique, mais qui ne sont absolument pas pertinentes dans le contexte de la description d'une expérience. Et pourtant, nous devons enregistrer ce type de décision de conception dans la documentation, au profit des autres utilisateurs potentiels qui pourraient examiner l'ontologie et qui autrement, ne seraient pas avertis de l'application que nous pensions en faire. Sans quoi, toute personne qui envisagerait la réutilisation de l'ontologie pour d'autres applications pourrait être tentée d'utiliser l'`Expérimentateur` en tant que sous-classe d'une personne, sans pour autant savoir que ce fait n'était pas inclus dans la modélisation originale.

4.7 Sous-classes disjointes

Plusieurs systèmes permettent de spécifier de façon explicite qu'un certain nombre de classes sont **disjointes**. Les classes sont disjointes lorsqu'elles ne peuvent pas avoir d'instances en commun. Par exemple, le `Vin de dessert` et le `Vin blanc` ne sont pas disjointes dans notre ontologie : beaucoup de vins sont à la fois instances de l'une et de l'autre. Le `Rothermel Trochenbierenauslese Riesling`, instance de `Riesling doux` en est un exemple. En même temps, les classes `Vin rouge` et `Vin blanc` sont des classes disjointes : il n'y a pas un vin qui puisse être à la fois rouge et blanc. Spécifier que les classes sont disjointes permet au système de mieux valider l'ontologie. Si nous déclarons les classes `Vin rouge` et `Vin blanc` comme étant disjointes, et qu'ensuite nous créons une classe à la fois sous-classe de `Riesling` (sous-classe de `Vin blanc`) et de `Porto` (une sous-classe de `Vin rouge`), le système signalera qu'il y a là une erreur de modélisation.

5. Définir les propriétés - plus de détails

Dans cette partie nous traiterons de quelques détails supplémentaires à observer lorsqu'on définit les attributs dans l'ontologie (Étape 5 et Étape 6 dans la Partie 3). Nous aborderons les attributs inverses et les valeurs par défaut pour un attribut.

5.1 Les attributs inverses

La valeur d'un attribut peut dépendre de la valeur d'un autre attribut. Par exemple, si un `vin` a été produit par un établissement vinicole, alors on peut dire que

l'établissement vinicole produit ce vin. Ces deux relations, producteur et produit, sont appelés **relations inverses**. Il est en effet redondant de stocker l'information dans " les deux sens ". Quand nous savons qu'un vin a été produit par un établissement vinicole, une application utilisant la base de connaissances peut toujours déduire la valeur pour la relation inverse : cet établissement vinicole produit ce vin. Toutefois, du point de vue de l'acquisition des connaissances, il est plus commode que les deux termes de l'information restent explicitement accessibles. Cette approche permet aux utilisateurs de renseigner le vin dans un cas et l'établissement vinicole dans un autre. Le système d'acquisition des connaissances pourrait alors renseigner automatiquement la valeur pour la relation inverse, assurant ainsi la cohérence de la base de connaissances.

Notre exemple contient une paire d'attributs inverses : l'attribut `producteur` de la classe `Vin` et l'attribut `produit` de la classe `Établissement vinicole`. Si un utilisateur crée une instance de la classe `Vin` et renseigne une valeur pour l'attribut `producteur`, le système ajoute automatiquement l'instance nouvellement créée à l'attribut `produit` de l'instance `Établissement vinicole` correspondante. Par exemple, quand on dit que le Merlot Sterling est produit par l'établissement vinicole des Vignobles Sterling, le système ajoute automatiquement Merlot Sterling à la liste des vins que produit l'établissement vinicole des Vignobles Sterling. (Figure9)

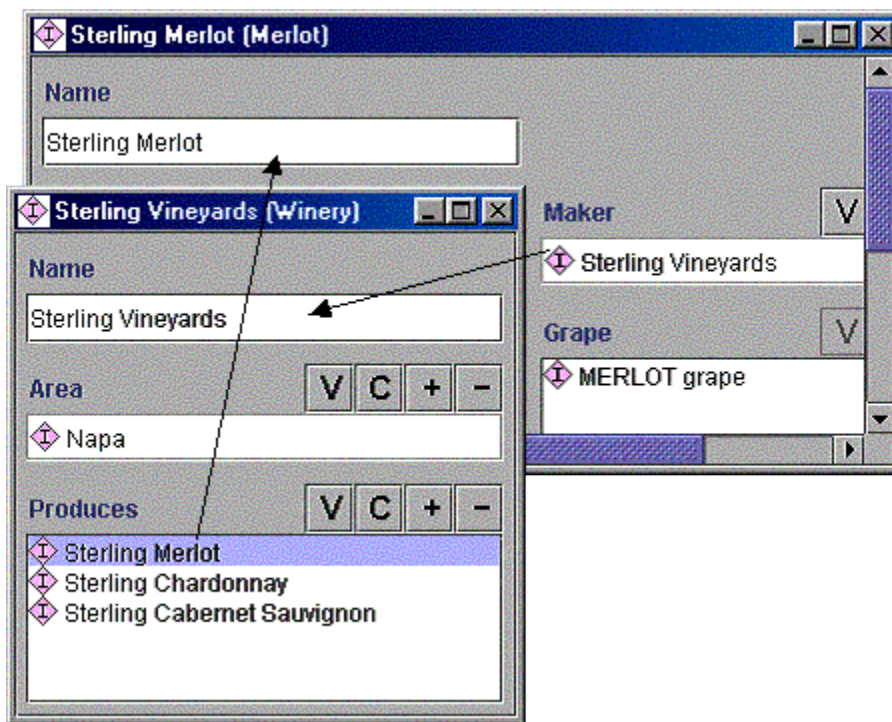


Figure 9. Instances avec attributs inverses. L'attribut `produit` de la classe `Établissement vinicole` est un inverse de l'attribut `producteur` de la classe `Vin`. Le renseignement de l'un de ces attributs déclenche la mise à jour de l'autre.

5.2 Valeurs par défaut

Plusieurs systèmes de type FRL permettent la spécification des valeurs par défaut pour les attributs. Lorsqu'une valeur particulière d'un attribut est la même pour la plupart des instances d'une classe, nous pouvons désigner cette valeur comme étant la **valeur par défaut** pour

l'attribut. en question. Par conséquent, à chaque création d'une nouvelle instance appartenant à une classe et comportant donc cet attribut, le système renseigne automatiquement la valeur par défaut. Nous pouvons aussi transformer cette valeur en toute autre valeur autorisée par les facettes. Cela signifie que les valeurs par défaut sont là par commodité : elles n'apportent ni de nouvelles restrictions dans le modèle, ni de changements quelconques du modèle.

Par exemple, si la majorité des vins que nous traitons sont des vins charpentés, nous pouvons avoir " charpenté " comme valeur par défaut pour le corps du vin. Cela implique que, sauf indication contraire, tous les vins que nous définirons seront des vins charpentés.

Notons la différence par rapport aux **valeurs des attributs**. Les valeurs des attributs ne peuvent pas être modifiées. Par exemple nous pouvons dire que l'attribut `sucré` à la valeur `DOUX` pour la classe `Vin de dessert`. Dans ce cas, toutes les sous-classes et instances de la classe `Vin de dessert` auront la valeur `DOUX` pour l'attribut `sucré`. Cette valeur ne peut être modifiée dans aucune des sous-classes ou instances de cette classe.

6. Qu'y a-t-il dans un nom ?

Définir des conventions à suivre lorsqu'on nomme les concepts dans une ontologie et y adhérer, non seulement rend l'ontologie plus compréhensible, mais aide également à éviter les quelques erreurs les plus fréquentes de modélisation. Plusieurs alternatives existent pour nommer les concepts. Souvent, il n'y a pas de raison particulière pour privilégier l'une ou l'autre de ces alternatives. Néanmoins nous avons besoin de :

Définir une convention de nomination pour les classes et les attributs et y adhérer.

Les caractéristiques suivantes des systèmes de représentation de connaissances affectent le choix de la convention de nomination :

- Le système a-t-il le même espace de nomination pour les classes, attributs et instances ? C'est-à-dire, permet-il d'avoir une classe et un attribut ayant le même nom (tels qu'une classe `établissement vinicole` et un attribut `établissement vinicole`) ?
- Le système est-il sensible à la casse ? C'est-à-dire, traite-t-il de la même façon les noms selon qu'ils sont entrés en majuscules ou en minuscules (tels que `Établissement vinicole` et `établissement vinicole`) ?
- Quels délimiteurs le système autorise-t-il pour les noms ? C'est-à-dire, les noms peuvent-ils contenir des espaces, des virgules, des astérisques, etc. ?

Protégé-2000, par exemple, maintient un espace de nommage unique pour tous ces cadres. Il est sensible à la casse. Ainsi, nous ne pouvons pas avoir une classe `établissement vinicole` et un attribut `établissement vinicole`. Par contre, nous pouvons avoir une classe `Établissement vinicole` (notez la présence de la majuscule) et un attribut `établissement vinicole`. CLASSIC, en revanche, n'est pas sensible à la casse et maintient des espaces de nommage différents pour les classes, attributs et individus. Ainsi, du point de vue du système, il n'y a aucun problème si l'on nomme à la fois une classe et un attribut `Établissement vinicole`.

6.1 Capitalisation et délimiteurs

Tout d'abord, nous pouvons améliorer considérablement la lisibilité d'une ontologie si nous utilisons systématiquement des lettres initiales majuscules pour les noms des concepts. Par exemple, il est d'usage d'utiliser des lettres initiales majuscules pour les noms des classes et d'utiliser des minuscules pour les noms des attributs (en supposant que le système est sensible à la casse).

Lorsque le nom d'un concept contient plus d'un mot (tel que `Ordonnance du repas`) nous avons besoin de délimiter les mots. Voici quelques alternatives de choix :

- Utiliser l'espace : `Ordonnance du repas` (plusieurs systèmes, y compris Protégé, autorisent les espaces dans les noms des concepts)
- Coller les mots les uns aux autres et employer une lettre initiale capitale pour chacun des mots : `Ordonnance Du Repas`
- Utiliser un underscore ou tiret, ou tout autre délimiteur dans le nom : `Ordonnance_Du_Repas`, `Ordonnance_du_repas`, `Ordonnance-Du-Repas`, `Ordonnance-du-repas`. (Si l'on utilise des délimiteurs, il faut également décider si la lettre initiale de chaque mot sera une capitale ou non).

Si le système de représentation de connaissances autorise les espaces dans les noms, les utiliser est vraisemblablement la solution la plus intuitive pour la plupart des développeurs d'ontologies. Il est, néanmoins important de prendre en compte les caractéristiques d'autres systèmes avec lesquels votre système peut être amené à interagir. Si ces systèmes n'utilisent pas d'espaces, ou si votre outil de présentation ne gère pas bien les espaces, il peut être utile d'utiliser une autre méthode.

6.2 Singulier ou pluriel

Un nom de classe représente une collection d'objets. Par exemple, une classe `Vin` représente en effet tous les vins. Pour cette raison quelques concepteurs choisiraient plus naturellement d'appeler la classe `Vins` plutôt que `Vin`. Aucun des termes de l'alternative n'est ni meilleur ni pire que l'autre. Cependant, quel que soit le choix, il persistera à travers toute l'ontologie. Certains systèmes exigent même de leurs utilisateurs de déclarer préalablement s'ils utiliseront le singulier ou le pluriel pour les noms des concepts et ensuite, ne leur permettent pas de dévier de leur choix initial.

L'utilisation systématique de la même forme empêche le concepteur de faire des erreurs de modélisation, telles que créer une classe `Vins` et créer par la suite une classe `Vin` en tant que sous-classe de la première (voir Partie 4.1)

6.3 Conventions sur l'emploi des préfixes et suffixes

Certaines méthodologies de bases de connaissances suggèrent la mise en place des conventions d'emploi de préfixes et suffixes rattachés aux noms pour distinguer les classes des attributs. Il est d'usage de faire précéder le nom de l'attribut par `a-pour` ou de le faire suivre par le suffixe `-de`. Ainsi, si l'on choisit la convention `a-pour`, nos attributs deviennent `a-pour-producteur` et `a-pour-établissement vinicole`. Si par contre, l'on choisit la convention `-de`, les attributs deviennent `producteur-de` et `établissement vinicole-de`. Cette approche permet à toute personne de déterminer immédiatement si le terme est une classe ou un attribut. L'inconvénient est que les noms des termes deviennent légèrement plus longs.

6.4 D'autres considérations sur les nommages

Quelques autres remarques à retenir lorsqu'on définit les conventions de nomination :

- Ne pas ajouter de chaînes de caractères telles que " classe ", " propriété ", " attribut ", etc aux noms des concepts.

C'est le contexte qui clarifie toujours si le concept est une classe, un attribut, etc. De plus, lorsque pour nommer les classes vous utilisez des conventions différentes de celles que vous utilisez pour les attributs (par exemple, l'emploi ou non des lettres initiales capitales), c'est le

nom lui-même qu'indiquera ce qu'est le concept.

Il est toujours avisé d'éviter les abréviations dans les noms des concepts (c'est à dire, utiliser Cabernet Sauvignon plutôt que Cab)

Les noms des sous-classes directes d'une classe doivent tous, sans exception, soit inclure, soit ne pas inclure le nom de la super-classe. Par exemple, si nous créons deux sous-classes de la classe Vin pour représenter les vins rouges et blancs, les deux noms des deux sous-classes doivent être soit Vin Rouge et Vin Blanc soit Rouge et Blanc, mais en aucune manière Vin Rouge et Blanc.

7. Autres ressources

Pour nos exemples, nous avons utilisé Protégé 2000 comme environnement de développement d'ontologies. Duineveld et ses collègues (Duineveld et al. 2000) décrivent et comparent un certain nombre d'autres environnements de développement d'ontologies.

Nous avons essayé d'aborder juste l'essentiel de ce qu'il faut savoir pour le développement des ontologies et ne sommes pas lancés dans la discussion de tous les autres sujets avancés ou bien des éventuelles méthodologies alternatives pour le développement des ontologies. Gomez-Pérez (Gomez-Pérez 1998) et Uschold (Uschold et Gruninger 1996) présentent des méthodologies alternatives pour le développement des ontologies. Le tutorial Ontolingua (Farquhar 1997) débat de certains aspects formels de la modélisation des ontologies.

Actuellement, les chercheurs mettent l'accent non seulement sur le développement des ontologies, mais aussi sur l'analyse des ontologies. Étant donné le nombre croissant d'ontologies qui vont être générées et réutilisées, l'offre des outils d'analyse augmentera proportionnellement. Par exemple, Chimaera (McGuinness et al. 2000) fournit des outils de diagnostic pour analyser les ontologies. L'analyse effectuée par Chimaera comprend aussi bien une vérification de la rigueur logique d'une ontologie que le diagnostic des erreurs habituelles dans sa conception. Un concepteur d'ontologies pourrait souhaiter faire tourner Chimaera sur une ontologie en évolution de façon à pouvoir évaluer sa conformité aux pratiques d'usage commun dans le domaine de la modélisation des ontologies.

8. Conclusions

Dans ce guide, nous avons décrit une méthodologie de développement d'ontologie pour les systèmes déclaratifs de type FRL. Nous avons listé les étapes dans le processus de développement d'une ontologie et abordé les problèmes complexes de définition d'une hiérarchie de classes, des propriétés des classes et des instances. Toutefois, après avoir suivi toutes les règles et suggestions, la remarque la plus importante à retenir est : *il n'y a pas qu'une seule ontologie correcte de référence pour un domaine précis*. La conception des ontologies est un processus créatif et il ne peut pas y avoir d'ontologies identiques faites par des personnes différentes. Les applications potentielles d'une ontologie et la compréhension du concepteur, ainsi que le point de vue qu'il a du domaine traité, affecteront indubitablement les choix de conception de l'ontologie. "C'est à l'usage que l'on juge" - nous pouvons tester la qualité de notre ontologie uniquement en l'utilisant dans les applications pour lesquelles elle a été conçue.

Remerciements

Protégé 2000 (<http://protege.stanford.edu/>) a été développé par le groupe de Mark Musen à Stanford Médical Informatics. Nous avons généré quelques unes des figures avec le Onto Viz plugin du Protégé 2000. Nous avons importé la version initiale de l'ontologie des vins de la bibliothèque Ontolingua des ontologies (<http://www.ksl.stanford.edu/software/ontolingua/>)

qui elle a utilisé une version publiée par Brachman et ses collègues (Brachman et al. 1991) et diffusée avec le système de représentation de connaissances CLASSIC. Nous avons ensuite modifié l'ontologie pour pouvoir présenter les principes de la modélisation conceptuelle pour les ontologies déclaratives de type FRL. Les commentaires de Ray Fergeson et Mor Peleg sur les versions antérieures ont contribué à améliorer considérablement cet article.

Références

- Booch, G., Rumbaugh, J. and Jacobson, I. (1997). *The Unified Modeling Language user guide*: Addison-Wesley.
- Brachman, R.J., McGuinness, D.L., Patel-Schneider, P.F., Resnick, L.A. and Borgida, A. (1991). Living with CLASSIC: When and how to use KL-ONE-like language. *Principles of Semantic Networks*. J. F. Sowa, editor, Morgan Kaufmann: 401-456.
- Brickley, D. and Guha, R.V. (1999). Resource Description Framework (RDF) Schema Specification. Proposed Recommendation, World Wide Web Consortium: <http://www.w3.org/TR/PR-rdf-schema>.
- Chimaera (2000). Chimaera Ontology Environment. www.ksl.stanford.edu/software/chimaera
- Duineveld, A.J., Stoter, R., Weiden, M.R., Kenepa, B. and Benjamins, V.R. (2000). WonderTools? A comparative study of ontological engineering tools. *International Journal of Human-Computer Studies* **52**(6): 1111-1133.
- Farquhar, A. (1997). Ontolingua tutorial. <http://ksl-web.stanford.edu/people/axf/tutorial.pdf>
- Gómez-Pérez, A. (1998). Knowledge sharing and reuse. *Handbook of Applied Expert Systems*. Liebowitz, editor, CRC Press.
- Gruber, T.R. (1993). A Translation Approach to Portable Ontology Specification. *Knowledge Acquisition* **5**: 199-220.
- Gruninger, M. and Fox, M.S. (1995). Methodology for the Design and Evaluation of Ontologies. In: *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95*, Montreal.
- Hendler, J. and McGuinness, D.L. (2000). The DARPA Agent Markup Language. *IEEE Intelligent Systems* **16**(6): 67-73.
- Humphreys, B.L. and Lindberg, D.A.B. (1993). The UMLS project: making the conceptual connection between users and the information they need. *Bulletin of the Medical Library Association* **81**(2): 170.
- McGuinness, D.L., Abrahams, M.K., Resnick, L.A., Patel-Schneider, P.F., Thomason, R.H., Cavalli-Sforza, V. and Conati, C. (1994). Classic Knowledge Representation System Tutorial. <http://www.bell-labs.com/project/classic/papers/ClassTut/ClassTut.html>
- McGuinness, D.L., Fikes, R., Rice, J. and Wilder, S. (2000). An Environment for Merging and Testing Large Ontologies. *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*. A. G. Cohn, F. Giunchiglia and B. Selman, editors. San Francisco, CA, Morgan Kaufmann Publishers.

McGuinness, D.L. and Wright, J. (1998). Conceptual Modeling for Configuration: A Description Logic-based Approach. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing - special issue on Configuration*.

Musen, M.A. (1992). Dimensions of knowledge sharing and reuse. *Computers and Biomedical Research* **25**: 435-467.

Ontolingua (1997). Ontolingua System Reference Manual. <http://www-ksl-svc.stanford.edu:5915/doc/frame-editor/index.html>

Price, C. and Spackman, K. (2000). SNOMED clinical terms. *BJHC&IM-British Journal of Healthcare Computing & Information Management* **17**(3): 27-31.

Protege (2000). The Protege Project. <http://protege.stanford.edu>

Rosch, E. (1978). Principles of Categorization. *Cognition and Categorization*. R. E. and B. B. Lloyd, editors. Hillsdale, NJ, Lawrence Erlbaum Publishers: 27-48.

Rothenfluh, T.R., Gennari, J.H., Eriksson, H., Puerta, A.R., Tu, S.W. and Musen, M.A. (1996). Reusable ontologies, knowledge-acquisition tools, and performance systems: PROTÉGÉ-II solutions to Sisyphus-2. *International Journal of Human-Computer Studies* **44**: 303-332.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorenzen, W. (1991). *Object-oriented modeling and design*. Englewood Cliffs, New Jersey: Prentice Hall.

Uschold, M. and Gruninger, M. (1996). Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review* **11**(2).