Université de Nice Sophia-Antipolis

Master recherche PLMT

Rapport de stage

présenté en Juin 2008

par Mathieu ACHER

Vers une ligne de services pour la grille : application à l'imagerie médicale

Responsables de stage : Philippe LAHIRE, Philippe COLLET
Rapporteur : Manuel SERRANO

Vendredi 13 Juin 2008

Chapitre 1

Introduction

1.1 Contexte

La grille fournit de la puissance de calcul et de l'espace de stockage en quantité via l'agrégation de ressources distribuées pour pouvoir réaliser des tâches de plus en plus complexes [11]. Dans le domaine scientifique ou industriel, il est maintenant courant d'exécuter des algorithmes extrêmement coûteux en temps ou manipulant des masses de données très importantes. La parallélisation des traitements – en utilisant simultanément les différentes ressources de la grille – permet ainsi d'accélérer sensiblement les performances d'une application. Le potentiel de la grille réside également dans sa capacité à partager des algorithmes et des données pour une communauté d'utilisateurs.
Le domaine de l'imagerie médicale illustre remarquablement les bénéfices engendrés par l'utilisation des grilles [10]. En effet, l'exploitation des analyses d'images médicales dans un milieu clinique impose des contraintes en temps – en fonction de l'urgence de la situation – et requiert un temps de calcul prohibitif pour une infrastructure informatique classique. De plus, les grilles permettent la gestion à grande échelle de données médicales volumineuses, qui peuvent atteindre plusieurs téra-octects pour des départements radiologiques, et ainsi fournissent l'accès à ces données à des catégories d'utilisateur variées (praticiens, patients, *etc.*) à travers le monde. Enfin, de nombreux algorithmes de traitement d'images médicales méritent d'être comparés et validés, par exemple en terme d'efficacité. La procédure de validation consiste alors en des études statistiques nécessitant des traitements intensifs sur de grands ensembles de données et elle n'aurait certainement pas été envisageable sans l'utilisation des grilles [16].

Dans une infrastructure de grille, les ressources telles que les noeuds de calcul, les moyens de stockage, les applications ou les bases de données sont fortement hétérogènes, disséminées dans des lieux géographiques différents, et interconnectées par un réseau. Un problème crucial est alors de parvenir à coordonner le partage de ces ressources et la résolution de problèmes dans un environnement dynamique, hétérogène, multi-institutionnel [12]. Comment, en effet, assurer une interopérabilité entre des codes applicatifs programmés dans des langages différents, s'exécutant sur des systèmes d'exploitation spécifiques et appartenant à des organisations et des instituts divers ? De plus, la synchronisation de tâches concurrentes, la tolérance aux fautes ou la gestion de données distantes sont autant d'aspects non triviaux à maîtriser. Incontestablement, il existe un besoin d'abstraction en matière d'architecture logicielle pour fournir à l'utilisateur un accès transparent aux ressources de la grille.

Dans ce contexte, l'approche orientée services (SOA) possède des caractéristiques pertinentes [13]. Les services sont des entités logicielles accessibles via le réseau et encapsulent le code applicatif à travers une interface standardisée. Les détails d'implémentation, comme le langage de programmation, ne sont pas révélés aux clients, ce qui permet à un service d'être indépendant d'une quelconque plateforme et de maîtriser l'hétérogénéité des codes des applications de la grille. Du fait de leur faible couplage avec les autres entités logicielles, les services peuvent être ajoutées, retirées ou remplacées selon les besoins, ce qui illustre les bénéfices de SOA en terme de flexibilité et de capacité d'évolution

pour un environnement distribué aussi dynamique que la grille. De plus, l'approche SOA permet de construire des applications en réutilisant des codes déja existants, par assemblage de briques logicielles, les services, typiquement sous la forme de chaînes de traitement, les *workflows*.

1.2 Problématique

L'approche SOA a pour objectif de cacher complètement l'infrastructure de grille à l'architecte logiciel chargé de composer les services. Cependant, la construction d'applications sur la grille par une approche SOA implique de considérer plusieurs problèmes. En effet, les mainteneurs de code doivent associer aux services une information suffisament fine pour que leurs compositions soient facilement réalisables, tout en maîtrisant leurs déploiements sur la grille. En particulier, les aspects non fonctionnels des services, qui décrivent les *conditions* dans lesquelles les fonctionnalités sont rendues, doivent être pris en compte lors de la composition des services. De nombreuses propriétés non fonctionnelles doivent être exploitées durant le déploiement ou l'exécution des services avec pour objectif d'assurer une qualité de service (QoS) adaptée à l'utilisateur. Ces propriétés de QoS révèlent des caractéristiques diverses et s'expriment sous différentes formes, car elles peuvent être relatives aux services mêmes, notamment en terme de performance (fiabilité, temps d'exécution attendu, *etc.*), au contexte d'exécution (temps de latence, débit, *etc.*) ou aux besoins de l'utilisateur (qualité du résultat, coût économique du calcul, *etc.*). Les applications doivent être en mesure, par exemple, de s'adapter au comportement de la grille, qui peut varier fréquemment lors de l'exécution et le plus souvent de façon non-déterministe. Un temps de latence trop important ou l'indisponibilité d'une ressource peuvent alors être rédhibitoires dans une situation où l'exécution des services est urgente.

Nous partons du constat que les problématiques de composition et de déploiement de services sont la conséquence d'une importante *variabilité*, difficilement maîtrisable. En effet, plusieurs services fournissent des fonctionnalités relativement similaires et sont autant de candidats pour effectuer une tâche donnée. Il est alors plutôt compliqué pour l'expert composant les services d'assurer la cohérence de son assemblage. Nous considérons aussi que la variabilité de QoS est encore plus importante – les propriétés de QoS des services concernent de nombreuses dimensions comme le temps, le côut, la qualité des résultats, *etc.*–. La composition des services par l'expert devient alors extrêmement complexe et suit des cycles d'essais/erreurs fastidieux. Pour résoudre ces problèmes, il s'agit non seulement d'expliciter les propriétés fonctionnelles et non fonctionnelles de ces services, mais aussi de capturer comment ces aspects diffèrent. Maîtriser cette variabilité nécessite de décrire des informations supplémentaires ajoutées aux services afin de mieux les réutiliser en termes de facilité, de flexibilité et de fiabilité. L'utilisation de mécanismes gérant la variabilité des services permettra, par exemple, la sélection de services adaptés au mieux aux contraintes imposées par l'utilisateur.

L'objectif de ce stage est de pouvoir introduire la variabilité fonctionnelle et non fonctionnelle dans les services de la grille, en particulier pour le traitement d'images médicales, qui sera notre cas d'étude. L'imagerie médicale est un domaine applicatif qui exacerbe tous les problèmes de variabilité rencontrés dans les grilles et constitue un élément de validation pour les travaux réalisés.

1.3 Démarche

Notre démarche s'appuie sur le concept de variabilité. En génie logiciel, celui-ci repose sur le besoin de spécifier les éléments communs et leurs différences pour décrire la généralité d'une entité. En particulier, le paradigme *lignes de produits* (SPL) [3] se reporte aux méthodes, techniques et outils logiciels pour la création de système logiciels partageant des artefacts logiciels. L'approche SPL tend d'ailleurs à connaître une attention particulière dans le domaine SOA [7]. Cette ligne de produits permettra aux fournisseurs de services et aux experts de workflows de *i)* capturer les éléments communs et les différences des codes déja existants, *ii)* de construire efficacement le bon service en fonction des éléments communs et des différences, *iii)* d'utiliser la ligne pour sélectionner les services

les plus adaptés en fonction des critères fonctionnels et non fonctionnels. Les techniques d'ingénierie dirigée par les modèles (IDM) seront utilisées pour construire la ligne de produit en tant que modèle des services manipulés, de telle sorte que la description, la sélection et la vérification de compatibilité puissent être réalisées à travers la ligne de produits.

Dans un premier temps, les opérations possibles pour les workflows de la grille sont envisagées et décrites en considérant les propriétés de QoS des services (chapitre 2). Une classification des dimensions de QoS est proposée en s'appuyant sur des travaux sur le sujet, et la variabilité de la QoS est ensuite analysée (chapitre 3). En particulier, le domaine de l'imagerie médicale est étudié pour déterminer l'impact de la variabilité des propriétés de QoS sur les services de la grille (section 3.3). Pour faire face à la variabilité des services, la construction de la ligne de services est décrite et le cas de la segmentation d'images illustre nos propositions (chapitre 4). Le chapitre 5 conclut ce mémoire et présente les perspectives de recherche.

Chapitre 2

QoS pour les services de la grille

2.1 Services et workflows pour la grille

Comme cela a déjà été dit précédemment, les scientifiques et les ingénieurs utilisent la grille pour construire des applications de plus en en plus complexes à maîtriser, manipulant des masses de données très importantes sur des ressources distribuées. L'émergence de l'approche orientée service (SOA) est le résultat d'un long processus pour parvenir à maîtriser des problèmes d'interopérabilité et d'indépendance vis-à-vis des plateformes. Ceci est absolument crucial, car les codes et les programmes partagés sont fortement hétérogènes, tous ayant leurs propres architectures, des langages d'implémentation et des besoins systèmes spécifiques. SOA encourage la réutilisation de codes, en mettant à disposition de l'architecte logiciel de nombreux *services*, dont les caractéristiques (couplage faible, extensibilité, *etc.*) facilitent leurs compositions. L'idée de construire des applications informatiques en composant des entités logicielles réutilisables est une approche naturelle : il s'agit par exemple de chaîner des traitements, de manipuler un flot de données ou d'exécution, *etc.*. Les workflows sont un des paradigmes de composition logicielle [15]. Un ensemble de règles définit alors les interactions entre un ensemble de services dans le but de les composer pour former un workflow. La logique d'une telle application composée est décrite à travers un ensemble de tâches de traitement à réaliser et de contraintes sur l'ordre des traitements. Plus précisément, le workflow représente un flot d'exécution, décomposé en plusieurs tâches inter-dépendantes – une tâche peut avoir besoin du résultat d'une autre avant de pouvoir s'effectuer –.

Le terme workflow a la particularité d'être utilisé à la fois pour dénoter la représentation d'une application dans la communauté *services* et dans le monde *grille de calcul*. Dans les deux cas, un workflow correspond à la description de la logique d'une application indépendamment de l'implémentation de ses composants logiciels et de l'infrastructure cible. Les gestionnaires de workflow ont été construits pour décrire, optimiser et contrôler l'exécution des flux de données. Une caractéristique fondamentale des workflows est la possibilité d'exprimer un parallélisme des tâches à effectuer et ainsi de distribuer efficacement l'exécution aux différentes ressources de la grille pour améliorer sensiblement les performances. De nombreuses applications scientifiques requièrent le traitement de données qui peuvent être manipulées indépendamment les unes des autres : les traitements peuvent alors être partagées entre différents services [28]. De plus, le modèle de pipeline, exploité avec succès pour les processeurs, peut être adapté aux parties séquentielles des services du workflow. En considérant le domaine de l'analyse d'images médicales, les workflows permettent aux différents acteurs d'exprimer les préoccupations de leurs domaines respectifs, c'est-à-dire *i)* au développeur d'applications – le scientifique spécialiste de l'imagerie médicale – de développer les services et les fonctionnalités du workflow *ii)* à l'utilisateur final – le clinicien ou le praticien – d'instancier le worklow sur les données *iii)* à l'expert de la grille de déployer le workflow sur la grille et d'ordonnancer efficacement le workflow sur les ressources de la grille. Les workflows constituent ainsi un cadre logique idéal pour une collaboration multi-disciplinaire entre différents experts.

2.2 Impact de la QoS sur les workflows

Dans ce contexte, les propriétés non fonctionnelles, exprimant les conditions dans lesquels les fonctionnalités sont rendues (consommation des ressources, propriétés systèmes requis, performance fournise, protocole de communication, *etc.*), doivent être impérativement considérées pour faciliter la composition des services dans les workflows, en définissant des règles et des contrats pour leurs déploiements et leurs réutilisations. Fournir des qualités de service non triviales tel que le temps de réponse, le débit ou la disponibilité est d'ailleurs une caractéristique majeure de la grille [11]. Dans un environnement comme la grille, il y a un nombre élevé de ressources similaires ou équivalentes, fournies par différentes organisations virtuelles [13]. Les utilisateurs de la grille peuvent sélectionner les ressources les plus adaptées et les utiliser dans les workflows. Ces ressources peuvent fournir une même fonctionnalité, mais optimiser différentes mesures de QoS, ce qui constitue un critère primordial de différenciation. Les besoins en QoS, comme une date limite ou un budget limité, doivent être gérés à l'exécution par le moteur de workflow. Dans ce contexte, la gestion de la QoS des workflows de la grille a un impact à de nombreux niveaux, incluant la spécification du workflow, la découverte des services et l'ordonancement du workflow.

2.2.1 Sélection de services

Un scénario classique est la spécification par l'utilisateur de ces besoins en qualité de service au niveau du workflow. Par exemple, comme dans la figure 2.1, le praticien veut exécuter son application dans un temps restreint (i.e date limite, durée maximale, *etc.*) avec un coût économique donné (i.e budget limité). Son application est un workflow composé de plusieurs tâches élémentaires qui appliquent un algorithme sur une donnée en entrée et produisent une donnée en sortie. Les tâches sont chaînées, non nécessairement de manière linéaire : les tâches peuvent être exécutées en parallèle (section 2.1). Pour chaque constituant du workflow, plusieurs services implémentent la fonctionnalité attendue, mais certains s'exécutent plus ou moins rapidement et ont un coût économique plus ou moins important. Le gestionnaire de workflow doit alors être capable de sélectionner le *meilleur* service parmi les candidats. C'est pourquoi les mécanismes de découverte des services doivent intégrer les descriptions de QoS des services. La démarche s'apparente alors à une composition de services dirigée par la qualité de service [33]. Plus précisément, les tâches composant le workflow doivent être ordonancées, statiquement ou à l'exécution, en fonction des informations non fonctionnelles des services et des qualités de services requises par l'utilisateur final. Le problème de l'ordonancement de workflow sur la grille se définit comme l'assignation des différents services de la grille aux différentes tâches du workflow. Chaque service a des caractéristiques qualitatives s'appuyant sur différents critères (fiabilité de 50%, temps d'exécution de 8 minutes pour le service S_{41} de la figure 2.1). L'objectif d'optimisation de l'ordonanceur est de trouver une combinaison qui fournisse le meilleur compromis entre les contraintes spécifiées par l'utilisateur et les qualités offertes par les services.

Plusieurs variantes du problème d'ordonancement existent [30] : *i)* l'optimisation peut s'effectuer selon un seul critère (i.e uniquement le temps d'exécution du workflow) ou plusieurs critères ; *ii)* les critères d'optimisation sont définis pour satisfaire l'utilisateur ou pour utiliser au mieux l'infrastructure de la grille (i.e usages limités des ressources) ; *iii)* l'optimisation est une fonction objectif (i.e minimiser le temps d'exécution) ou une restriction (i.e budget limité) ; *iv)* l'optimisation peut être globale (et concernée l'ensemble des tâches du worflow) ou bien définie pour une tâche spécifique du workflow.

2.2.2 Vers l'adaptativité des services

Un service de supervision d'application peut fournir aux utilisateurs et aux administrateurs de la grille des données fiables sur les applications qui s'exécutent sur la grille [27] : les propriétés de QoS des services alimentent alors cet outil de *monitoring*. Ainsi, la supervision est utilisée pour fournir à l'ordonanceur des informations suffisament fines lui permettant d'effectuer un raisonnement

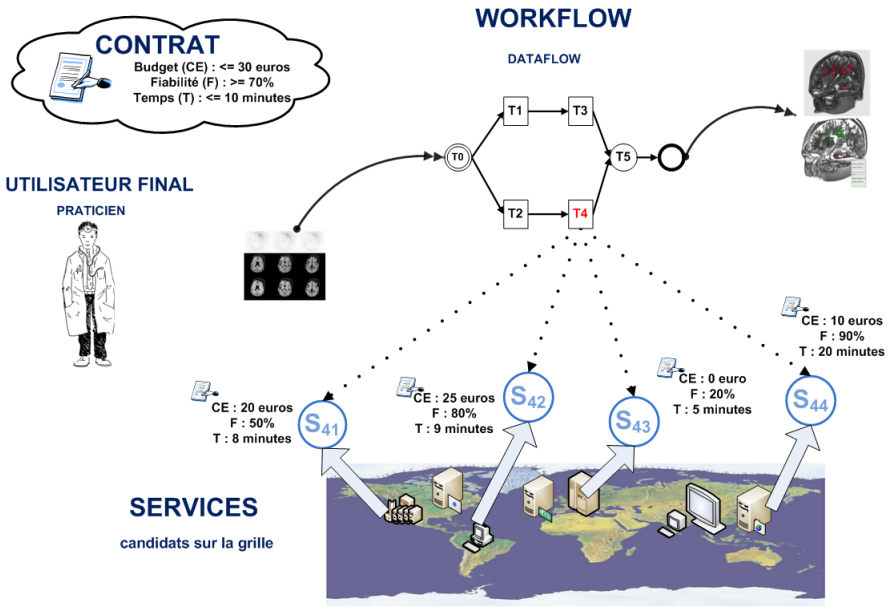

Fig. 2.1 – QoS et workflow : scénario d'utilisation

sur les workflows. Ensuite, le contrôle de la qualité de service au cours de l'exécution permet de vérifier le bon comportement des applications de la grille. Dans le cas d'une détection d'erreur, par exemple un temps de latence trop important, une stratégie adaptative peut être utilisée pour palier les défaillances observées. Les mesures de l'outil de surveillance peuvent éventuellement aider les applications à respecter les qualités de service exigés par l'utilisateur, sous forme de contrats (*Service Level Agreement*). De plus, la collecte d'informations sur le contexte d'exécution peut optimiser le déploiement des services. Enfin, la connaissance *en amont* des qualités de services offertes calculée par des méthodes statistiques ou de prédiction grâce aux informations acquises permet d'adapter au mieux l'exécution des applications de la grille. L'utilisation de mécanismes d'auto-adaptativité, comprenant à la fois des opérations de surveillance, de réparation et d'optimisation, peut dans certains cas s'envisager et participer à la vision d'une informatique *autonome* [21].

2.3 Synthèse

L'étude de la QoS des services de la grille, dans le cadre de leurs compositions sour forme de workflow, a montré des enjeux et bénéfices possibles pour les scénarios envisagés. Aussi nous considérons que la gestion de la QoS dans les workflows doit permettre *i)* la connaissance de la QoS d'une application *avant* de rendre le service disponible aux clients, *ii)* la sélection (statique ou dynamique) des services participant au workflow en s'appuyant sur leurs QoS ; en particulier, l'ordonancement de tâches sur la grille de calcul en considérant les paramètres de QoS des services peut sensiblement améliorer les performances des workflows [4], *iii)* la surveillance de la QoS qui autorise la mise en place des stratégies adaptives quand des valeurs de QoS indésirables sont atteintes. L'objectif à terme est d'équiper les services de la grille d'informations supplémentaires pour permettre de contrôler, manipuler, raisonner sur la qualité de service. Pour parvenir à maîtriser des caractéristiques très diverses et complexes, à confronter les considérations métiers d'un domaine spécifique avec les préoccupations d'une infrastructure distribuée, et finalement parvenir à associer les exigences d'un utilisateur final avec les capacités bas niveaux des systèmes, une analyse plus élaborée des propriétés de qualité de service s'impose, ce qui permettra d'envisager la mise en oeuvre d'une approche logicielle pragmatique.

Chapitre 3

Variabilité de la QoS

Les propriétés de QoS des workflows représentent les caractéristiques qualitatives et quantitatives d'une application nécessaires pour répondre à des besoins initiaux. Pour pouvoir apprécier quantitativement les propriétés de QoS et faire des observations, celles-ci doivent être mesurées concrètement, de manière statique ou dynamique. Les différentes propriétés de QoS sont décrites à travers différentes *dimensions* [14]. Ainsi, dans la suite, nous relevons les différentes dimensions pertinentes vis-à-vis des workflows pour la grille, en considérant une partie des nombreux travaux du domaine. Ensuite, nous faisons le constat que les propriétés de QoS sont de nature très diverses et ont des caractéristiques particulières, ce qui nous conduit à analyser leur *variabilité*. Le domaine de l'imagerie médicale permet d'illustrer notre étude et de valider notre analyse.

3.1 Classification

Cardoso [6] étudie spécifiquement les propriétés de QoS vis-à-vis des workflows, mais hors contexte de la grille, et propose quatre dimensions. Dans [24], les auteurs proposent sept dimensions pour classifier les propriétés de QoS des *web* services, l'implémentation la plus commune du concept de services, également sans considérer leur utilisation dans une infrastructure de grille. Dans [31], en s'appuyant sur les travaux précédemment décrits [6][24], les auteurs ont proposé cinq concepts d'assez haut niveau pour permettre d'exprimer l'ensemble des caractéristiques de QoS rencontrées dans l'état de l'art.

Aussi, il est pertinent de reprendre les dimensions proposées pour établir une classification comprenant : *i)* le temps, c'est à dire le temps total nécessaire à une instance pour transformer un ensemble d'entrées en un ensemble de sorties, *ii)* le coût, qui peut être un coût matériel, économique, humain, etc *iii)* la fidélité, qui évalue en quoi un service produit un bon résultat, *iv)* la fiabilité (i.e un taux d'échecs), et *v)* la sécurité, qui se réfère à des notions comme la confidentialité ou la confiance. Nous effectuons une analyse plus détaillée de chaque dimension :

Temps. Selon [6], le temps est une mesure commune et universelle pour évaluer la *performance*. Précisément, c'est le temps total s'écoulant entre la soumission de la requête et la réception de la réponse. Le temps nécessaire pour qu'une tâche se termine est alors décomposé comme la somme d'un temps de latence [25] (qui comprend le délai de configuration du service, le temps d'attente dans la queue (timeout), *etc.*), et le temps effectif de la tâche. La dimension temps dans les workflows a fait l'objet de multiples recherches et est un critère trés important. On trouve également cette dimension dans [32] où le temps d'exécution pour chaque tâche du workflow – ou du workflow dans son ensemble – est spécifié par l'utilisateur. Dans [18], une métrique de performance est définie et l'originalité de l'approche est d'utiliser un modèle de QoS probabiliste pour capturer l'incertitude sur la grille (par exemple du temps de latence [22]). Par ailleurs, la mise à l'échelle de ressources de la grille peut conduire à l'augmentation des performances des services. Cependant, certains services ne sont pas associés à des algorithmes parallèles ou n'ont

pas de propriétés de passage à l'échelle, ce qui peut avoir une influence sur l'évaluation des propriétés de temps. Sur la figure 2.1, la dimension de temps est utilisée lorsque le praticien contraint l'ordonanceur du workflow en fixant une durée d'exécution maximale.

Coût. Le coût associé aux tâches du workflow exprime aussi bien le coût des équipements de la grille (coût sur le matériel, charge, monopolisation d'une machine, *etc.*) que le côut de gestion (humain, financier). Le coût est ainsi une notion générale et peut être constitué de l'aggrégation de plusieurs critères. Le coût économique est particulièrement considéré [33][18][4][32] et peut être sujet à l'exigence de l'utilisateur (i.e budget limité) ou à une politique économique de la part de la grille (par exemple la volonté de limiter le coût en bande passante de l'exécution du workflow). Un exemple est lorsque le praticien ne veut pas dépasser un certain budget (voir figure 2.1). Les services peuvent avoir une influence directe sur le coût structurel de la grille : en effet, plus un service nécessite de puissance de calcul, plus le coût sur le matériel sera élevé. Le coût peut alors être utilisé par les ordonanceurs privilégiant des stratégies d'optimisation de l'infrastructure de grille.

Fidélité. La fidélité se définit par la qualité de rendu de l'exécution et reflète en quoi un service produit un *bon* résultat. C'est une mesure de perception et de jugement, donc parfois difficilement mesurable, mais il est important de la prédire dès que possible. Dans [6], la fidélité est un vecteur représentant un ensemble d'attributs de fidélité, chaque attribut représentant une caractéristique ou une propriété du service. L'intérêt de la représentation est de pouvoir pondérer les attributs de fidélité en fonction de leur importance. En outre, la réputation d'un service peut faire intervenir directement l'utilisateur en lui permettant d'évaluer la qualité de service.

Fiabilité. Dans [17], la fiabilité est étudiée spécifiquement dans une architecture orientée service. La fiabilité fournit un taux d'échecs et peut être défini comme le rapport entre le nombre d'exécutions aboutissant à un succès et le nombre d'exécutions total. Dans [6], les auteurs préconisent d'utiliser un modèle de fiabilité adapté aux spécificités du workflow, sachant qu'il existe un grand nombre de modèles de fiabilité dans la littérature. Les notions associées à la dimension fiabilité vont de la tolérance à la faute à la robustesse. C'est une dimension de QoS classique que l'on retrouve dans les domaines du calcul distribué, du réseau ou des grilles. La fiabilité exprime également la capacité d'un service à reproduire une performance en fonction des conditions rencontrées (nature des données en entrée, *etc.*).

Sécurité. La sécurité est une notion générale regroupant d'autres notions comme la confidentialité, le contrôle d'accès, la règlementation, *etc.*Il s'agit pour le moteur du workflow d'assurer de bout en bout la sécurité des données dispersées dans des lieux géographiques et des domaines d'administration différents. Ainsi, la confiance dans les ressources de la grille se réfère aux politiques de sécurité mises en œuvre, aux capacités d'auto-défense, à l'historique des attaques réalisées, à la réputation ou encore à la vulnérabilité d'un site [35].

3.2 Analyse de la variabilité

La taxonomie de dimensions de QoS décrite précédemment (voir figure 3.1) a illustré le large éventail de propriétés qu'on peut trouver dans les workflows de la grille. Ainsi, classifier les propriétés de QoS est une première approche pour capturer leurs similarités et leurs différences. Nous complétons maintenant l'analyse des éléments communs et variables des qualités de service.

En comparant la *fiabilité* d'un algorithme de traitement d'images et d'un service d'authentification, nous pouvons constater une forte divergence de point de vue : le premier exprime des propriétés comme la robustesse ou la précision, tandis que le second se réfère à propriétés du réseau ou de sécurité. C'est-à-dire que selon les services, une dimension de QoS donnée n'aura pas la même signification. C'est pourquoi la caractérisation qualitative des services de la grille par des dimensions très générales de QoS peut être raffinée par des propriétés de QoS plus explicites ou appartenant à un domaine particulier

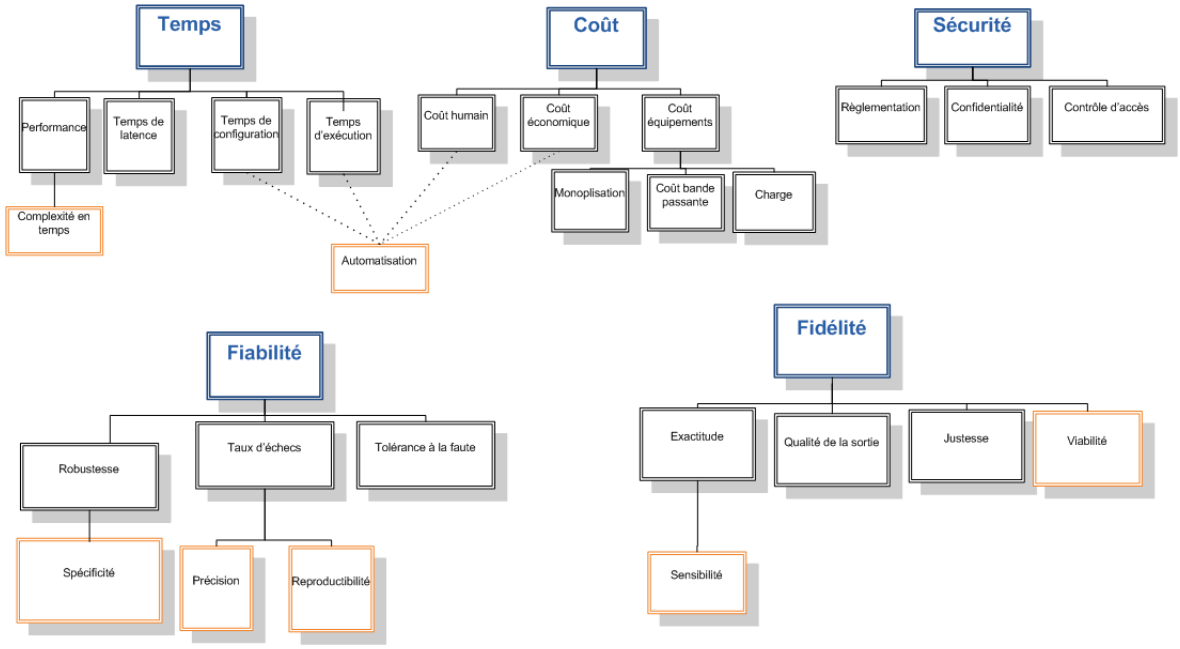

Fig. 3.1 – Aperçu des dimensions de QoS des workflows de la grille et des dimensions spécifiques de l'imagerie médicale comme la sensibilité.

(voir plus loin la section 3.3). Il est alors possible de décomposer qualitativement un concept non fonctionnel en sous-aspects non fonctionnels plus précis. La capacité d'un service à pouvoir décrire des propriétés plus ou moins fines, ou au contraire à se limiter à des notions générales, est un autre point de variabilité. Il est également important de permettre à l'utilisateur final de spécifier – idéalement de manière intuitive – des qualités de service via des concepts de haut niveau tout en étant capable d'établir la correspondance avec les propriétés non fonctionnelles des services. La difficulté réside dans le fait qu'il faut lier les contraintes de qualité avec les éléments de l'application. Pour pouvoir manipuler les propriétés de QoS (section 2.2), il est intéressant de caractériser les concepts associées [9]. Il s'avère que la *structure* des propriétés de QoS révèle également une grande variabilité.

Le calcul d'une QoS peut s'effectuer de manière dynamique ou statique (i.e par des méthodes statistiques, empiriques ou prédictives), et dans certains cas, ce calcul est tout simplement impossible : la propriété est non mesurable. Pour être quantifié, une propriété de QoS a parfois besoin d'une connaissance *a priori* sur les données manipulées en entrée. Ainsi, la nature des entrées peut avoir une influence considérable sur les performances d'un algorithme, tout comme la spécification des paramètres pour les opérations d'un service. Le contexte d'exécution, d'environnement ou d'utilisation, en particulier sur la grille, doit également être considéré : un temps de latence trop important peut être dommageable pour le temps d'exécution d'un service. En outre, certaines propriétés de QoS sont parfois déterminées *après* l'exécution du service : c'est par exemple le cas des outils de monitoring qui contrôlent la qualité sur la sortie. Enfin, des notions d'erreurs sont parfois introduites pour fournir en parallèle une appréciation de la validité des mesures. Plus généralement, la qualité du calcul de la QoS est elle-même variable : un compromis doit être fait entre des mesures très fines, mais complexes et coûteuses à obtenir (valeurs fortement changeantes et mesurées périodiquement), et des mesures moins précises mais qui peuvent être plus facilement mesurées et sont plus facilement exploitables (mesures statistiques).

Le début de cette analyse a montré que les modalités de calcul de la QoS présentent plusieurs différences notables. Cette variabilité a une influence considérable sur les opérations que l'on souhaite réaliser : il est par exemple nécessaire d'être capable de contrôler avec *précision* – une erreur d'appréciation la plus basse possible – et de manière *dynamique* – par opposition à statique – une application

temps-réel critique. Ce sont autant de points de variation à prendre en compte et à adapter selon les besoins.

Par ailleurs, une propriété de QoS peut être inter-dépendante avec une autre : une restriction sur l'utilisation de la mémoire d'un système logiciel – correspondant à la dimension de coût – peut faire chuter les performances en temps de ce même système. Ensuite, une QoS possède une métrique[1] dont les unités de mesure, le type de valeurs et la dimension qualitative associés diffèrent. Il est important de pouvoir comparer des QoS ; il est difficile cependant de mettre en relation la fiabilité d'un service météo et d'un service de traitement d'images. Typiquement, les opérateurs d'égalité (resp. de différence), d'infériorité (resp. supériorité), de minimum (resp. maximum) sont utiles pour sélectionner des services en fonction de la QoS. C'est pourquoi les QoS doivent fixer les conditions précises dans lesquelles la comparaison avec d'autres QoS est possible.

3.3 Application à l'imagerie médicale

Les nombreuses modalités d'acquisition d'images médicales [1] ont incontestablement révolutionné la pratique médicale en fournissant au praticien les moyens de visualiser les informations anatomiques du patient. Ainsi, les outils de traitement et d'analyse des images médicales apportent au médecin une aide considérable pour le diagnostic et lui permettent également de simuler, contrôler et valider la thérapeutique mise en place [10]. Sur la grille, la plétore d'algorithmes que l'on trouve dans le domaine médical est accessible de manière transparente aux utilisateurs pour leur permettre construire des applications. Pour pouvoir réutiliser au mieux des codes hétérogènes, pour la plupart non modifiables, et les composer sous forme de workflows (section 2.1), nous analysons la variabilité des propriétés de QoS de ces services.

3.3.1 Les dimensions de QoS

Pour un traitement d'images donné, comme le recalage, la segmentation ou le filtrage, il existe plusieurs algorithmes à même de l'effectuer. La variabilité peut concerner les aspects fonctionnels : par exemple, un algorithme va être plus adapté car il supporte le format de l'image en entrée (DICOM, Nifti, *etc.*), la modalité d'acquisition de l'image (IRM, scanner, *etc.*) ou parce qu'il est capable de fournir un résultat selon un modèle donné (forme matricielle, vectorielle, *etc.*). Cependant, on constate que les variations des algorithmes concernent principalement leur *comportement*, c'est-à-dire les qualités qu'ils présentent en terme de performance d'exécution ou de fidélité vis-à-vis d'un résultat attendu. Les dimensions de QoS identifiées dans la classification de la section 3.1 se retrouvent ainsi dans les algorithmes d'imagerie médicale [19] :

Temps. La complexité fonctionnelle et le temps de calcul des méthodes de traitement d'images sont associées pour exprimer les qualités d'exécution en temps des algorithmes. L'intervention d'un opérateur humain est parfois nécessaire pour guider les algorithmes. Ainsi, le degré d'automatisation peut être intégré dans l'évaluation du temps nécessaire.

Coût. La complexité en espace de l'algorithme est à prendre en compte, tout comme les ressources nécessaires et mises en œuvre pour l'utilisation des algorithmes. De plus, le coût humain et économique peut être variable selon le degré d'automatisation.

Fidélité. L'exactitude est un critère déterminant pour le traitement d'images et dénote le degré de similitudes entre le résultat obtenu et un résultat juste. L'évaluation de l'exactitude peut concerner l'image dans sa globalité ou certaines zones de l'image. La viabilité exprime le degré d'efficacité accordé à un algorithme en fonction d'un objectif clinique à réaliser [29] et de ce fait mesure son aptitude *pratique* à être utilisé dans des conditions précises.

[1] dans le sens *mesure*

Fiabilité. La reproductibilité (ou précision) est la capacité d'un processus à reproduire un même résultat pour une même entrée et mesure une fluctuation aléatoire. La prévisibilité d'une analyse d'images est importante pour l'expert qui a un modèle mental de comment l'algorithme fonctionne et qui peut ainsi corriger un mauvais fonctionnement. Pouvoir détecter automatiquement les erreurs est une qualité intéressante pour évaluer la tolérance aux fautes. La fiabilité d'un programme de traitement d'images se mesure également en terme de robustesse, c'est-à-dire à la performance réalisée en présence de facteurs perturbateurs ou encore sa capacité à pouvoir traiter une variété d'images en entrée.

Sécurité. les images médicales contiennent parfois des informations sur les patients ; il est de ce fait nécessaire d'assurer la confidentialité des données et un contrôle sur l'accès à ces informations de bout en bout de la chaîne de traitements.

La figure 3.1 intègre quelques propriétés de QoS analysées ci-dessus dans la classification des cinq dimensions de QoS. Les considérations qualitatives en terme de fidélité et de fiabilité sont les plus étudiées dans la littérature. Ce sont deux dimensions inter-dépendantes comme le révèlent les notions de sensibilité et de spécificité. La sensibilité représente les tests positifs pour des instances positives d'un problème. Dans l'établissement d'un diagnostic, cela correspondrait à la probabilité d'avoir un résultat positif lorsque le patient est malade, c'est-à-dire l'aptitude à détecter les cas de maladie. D'autre part, la spécificité représente les tests négatifs pour les instances négatives d'un problème : pour reprendre l'exemple précédent, cela évalue l'aptitude à ne détecter que les cas d'une maladie précise, c'est-à-dire la probabilité d'avoir un diagnostic faux lorsque le patient est sain. L'évaluation des deux aspects via une unique métrique ne peut se réaliser sans compromis [26].

3.3.2 Variabilité de la QoS pour la segmentation

La segmentation est une technique de traitement d'images très utilisée et sujette à de nombreux travaux de recherche. Les diverses tentatives pour parvenir à résoudre le problème de la segmentation ont conduit à la conception d'algorithmes très divers, utilisant des techniques différentes. Cette supposée forte variabilité nous a conduit a étudié le problème particulier de l'analyse d'images, notamment en terme de qualité comportementale des algorithmes. La segmentation d'images consiste en la reconnaissance et l'extraction d'objets perceptibles dans une image, de telle manière que l'image soit plus compréhensible et facile à analyser. Dans le domaine médical, le but de la segmentation est de sélectionner des objets artificiels ou correspondant à une partie de l'anatomie d'un patient, et qui ont besoin d'être mesurées ou visualisées par le praticien clinique. Le processus de segmentation est une étape cruciale et souvent préliminaire pour l'analyse d'images médicales. La segmentation automatique est un problème sans solution générale, qui dépend de plusieurs facteurs, comme la modalité d'acquisition de l'image médicale, du bruit contenu dans l'image, des caractéristiques des objets extraits, ou simplement de l'objectif même de la segmentation. Pour une même image, des algorithmes de segmentation peuvent donner des résultats extrêmement différents. Sélectionner une technique de segmentation appropriée, juste et efficace peut ainsi éviter d'éventuels résultats inappropriés. C'est pourquoi le besoin d'une mesure standard de qualité a été soulignée à maintes reprises dans le but d'évaluer et de comparer les algorithmes de segmentation [34].

Différentes méthodes d'évaluation ont ainsi été proposées. Ces méthodes d'évaluation constituent un premier degré de variabilité. Premièrement, les méthodes analytiques considèrent directement les principes et les propriétés (comme les éléments requis, la complexité, *etc.*) des algorithmes. Les méthodes analytiques ont reçu peu d'attention, en regard du fait qu'elles ne peuvent pas obtenir de propriétés à granularité fines, qu'elles ne fonctionnent que dans un contexte particulier et qu'il est de cette manière difficile de comparer les algorithmes. Deuxièmement, les méthodes *goodness* calculent les propriétés spécifiques de l'objet segmenté dans l'image comme l'uniformité intra régions, le contraste entre les régions, l'entropie, *etc.* Comme remarqué dans [5], cette approche implique une certaine subjectivité dans la propriété considérée. Par contre, avec ces méthodes, l'évaluation ne requiert pas

la connaissance d'une référence explicite et peut donner un résultat rapide dans certains cas. Enfin, les méthodes *discrepancy* se réfèrent à une vérité terrain (ou étalon-or). La mise à disposition d'une segmentation de référence, supposée être l'image segmentée idéale, permet de mesurer la relation entre le résultat de la segmentation et les références. Des mesures de similarité ou de différence entre l'image segmenté et la référence sont ainsi calculées, un s'appuyant sur la notion de distance.

Nous avons aussi identifié plusieurs autres points de variabilité de la QoS pour la segmentation. Premièrement, les méthodes d'évaluation doivent spécifier le domaine d'application considéré, ce qui est déterminé, selon [29], par trois entités : le but de la segmentation (la tâche), la région du corps et le protocole de l'image. Typiquement, une méthode de segmentation particulière peut avoir une excellente performance en déterminant le volume d'une tumeur dans le cerveau sur une image IRM, mais peut avoir de piètres performances en segmentant la masse cancéreuse d'une mammographie du sein. Il est ainsi nécessaire d'introduire une information spécifique sur les images et sur les structures anatomiques que le praticien veut identifier pour pouvoir calculer les qualités comportementales d'un algorithme. Plus généralement, l'étude des méthodes d'évaluation montre que la QoS est dépendante d'un contexte : l'absence d'une image de référence interdit l'utilisation des méthodes *discrepancy*, tandis que la connaissance du contexte clinique et des objectifs médicaux peut largement améliorer les mesures de QoS. Ensuite, les analyses statistiques, qui comprennent principalement les méthodes *discrepancy* décrites ci-dessus, doivent proposer des critères communs (*métriques*) pour comparer les algorithmes de segmentation. Mais ces métriques n'ont de sens que si elles sont calculées dans un contexte particulier. Un autre aspect important est que la qualité de l'évaluation elle-même doit être considérée dans le processus d'évaluation [34], et de ce fait, présente des caractéristiques variables. Certaines méthodes d'évaluation se concentrent sur des mesures spécifiques : les métriques de validation en médecine, comme la sensibilité et la spécificité, ont des influences non seulement sur le traitement de l'image médicale mais aussi sur les évaluations des tests médicaux. De plus, ces métriques sont subjectives ou objectives. La complexité des méthodes d'évaluation est également importante, si on veut les utiliser dans un processus de surveillance et de contrôle de la QoS. Finalement, il faut noter les interdépendances complexes entre les différentes dimensions de QoS : une tentative pour augmenter l'exactitude peut impliquer une baisse de la précision.

3.3.3 Bilan

Sur la figure 3.2, un workflow traite quatre séquences d'IRM d'un même patient : ces quatre IRM ont été réalisées selon des paramètres d'acquisition différents (temps de relaxation T1, temps de relaxation T2 avec densité protonique, DEFSE (Dual Echo Fast Spin Echo) et temps de relaxation T2 FLAIR). La première étape consiste à recaler chaque image en fonction d'une image de référence faisant partie des quatre, c'est-à-dire à transformer géométriquement les images de façon à les aligner entre elles. Ensuite, des étapes de prétraitement se succèdent : modification des axes d'orientation, isolation des parties du cerveau en ne considérant pas les tissus n'appartenant pas au cerveau, puis correction d'intensité. Enfin, l'étape de segmentation conclut l'exécution du workflow.

Pour cette dernière étape, plusieurs algorithmes de segmentation existent, sous forme de services. Ils sont autant de prétendants pour effectuer la tâche. Cependant, nous sommes dans un contexte bien particulier : les images médicales sont des IRM, d'un format particulier ; la partie de l'anatomie étudiée est le cerveau, et suite aux étapes de prétraitement, on peut supposer le bruit négligeable. Le contexte tel qu'il est décrit est ainsi l'explicitation de caractéristiques potentiellement très diverses : l'image médicale aurait pu être un scanner ou une écographie et représentée le foie ou le genou d'un patient. Dans le même temps, les capacités fonctionelles des services varient considérablement en fonction de ce contexte. Par exemple, certains algorithmes sont plus adaptés pour traiter des IRM que des scanners. Aussi, en identifiant chaque élément variable du domaine, les services de segmentation caractérisent leurs propriétés fonctionnelles. De même, certains services vont mieux correspondre de part les caractéristiques de leurs propriétés de QoS.

De ce fait, le workflow, via un moteur de sélection, peut choisir le service de segmentation en

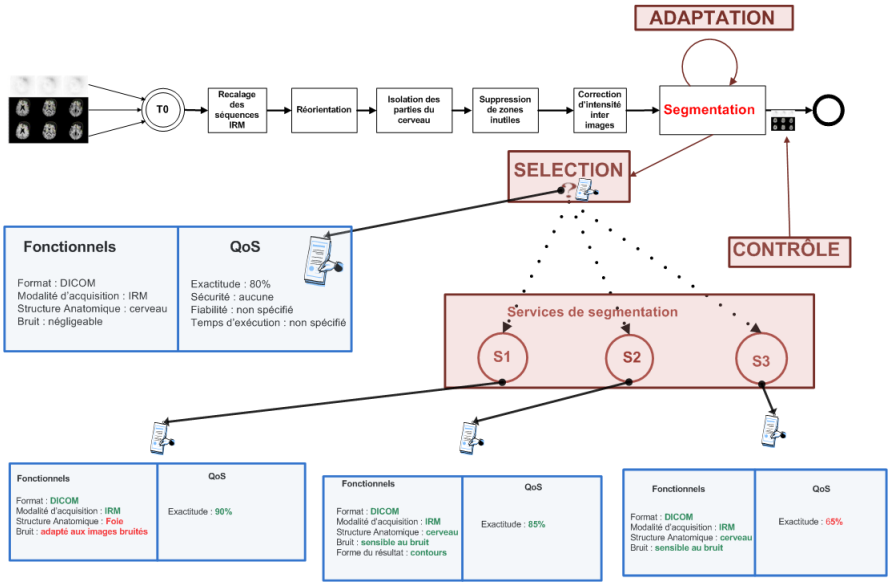

Fig. 3.2 – Variabilité de QoS dans un workflow pour le traitement d'images médicales

utilisant les informations non fonctionnelles. Dans le cadre d'un choix entre plusieurs services, il est nécessaire de pouvoir les comparer, mais encore faut-il que la métrique de la QoS le permette. Pour le scénario envisagé figure 3.2, la fidélité est privilégiée au profit de la fiabilité qui n'est pas spécifiée. Le contôle s'effectue alors de manière dynamique, avec un calcul le plus précis possible, sur la sortie – l'image obtenue après exécution du service de segmentation –. Dans le cas d'un résultat inapproprié vis-à-vis des exigences, une stratégie adaptative est utilisée, par exemple en sélectionnant un autre service. L'échec du service de segmentation pourra être pris en compte, de manière à avoir une connaissance *a posteriori* la prochaine fois que le même traitement est demandé. Le service sera alors en mesure d'évaluer statiquement la QoS.

Chapitre 4

Vers une ligne de services

Afin de mettre en oeuvre la maîtrise de la variabilité, notre démarche s'appuie sur les lignes de produits logiciels (SPL). Capturer la similarité des services pour pouvoir réutiliser des éléments communs est un premier objectif. Il s'agit ensuite de modéliser les différences structurelles et comportementales des services. L'utilisation des techniques de SPL permet alors de développer une *ligne de services*. Après avoir présenté la notion de variabilité et le paradigme ligne de produits, les principes de la ligne de service sont expliqués, tout comme le métamodèle décrivant la ligne.

4.1 Variabilité

La variabilité est basée sur le besoin de spécifier les éléments communs et les différences dans le but de décrire la généralité d'une entité [3]. Considérons un processus de workflow dans l'imagerie médicale. Pour un même type de traitement d'images, il y a la plupart du temps plusieurs manières de le réaliser. Avec pour objectif de rendre la description d'un tel processus suffisament réutilisable, une approche appropriée est d'introduire la variabilité dans la description des services utilisés. Cette approche permet à l'architecte logiciel de *i)* décrire la structure et le comportement d'une seule entité avec les possibles parties communes qui peuvent exister entre différentes entités similaires ; *ii)* proposer des variantes de l'entité et ainsi spécifier les variations possibles de la structure ou bien des fonctionnalités fournises ; *iii)* définir les parties optionelles à la fois pour les aspects structurels et comportementaux ; *iv)* définir les contraintes d'assemblage et d'exécution entre les entités (les principaux types de contraintes sont l'exclusion mutuelle et les contraintes de dépendances) ; *v)* définir des variations comportementales qui sont déduites de la spécification des variantes de l'entité.

4.2 Ligne de produits

Les industriels ont depuis longtemps employé des techniques d'ingénierie pour créer une ligne (ou famille) de produits similaires en utilisant une fabrique commune qui assemble et configure les parties destinées à être réutilisées à travers la ligne de produits. Par exemple, les constructeurs automobiles peuvent créer des centaines de variation d'un modèle de voiture en utilisant un *pool* de parties soigneusement construites et une usine spécialement conçue pour configurer et assembler les parties. De manière similaire, l'approche ligne de produits logiciels se reporte aux méthodes, techniques et outils logiciels pour la création de système logiciels partageant des artefacts logiciels. Dans [23], il est montré comment Nokia a opté pour l'approche ligne de produits pour gérer la diversité des logiciels de ses téléphones mobiles (support d'une multitude de langues, des différents standards de communication, des interfaces utilisateurs, *etc.*). L'idée est alors de déduire de la ligne *le* produit le plus à même de satisfaire un besoin. Une SPL doit supporter les concepts de variabilité décrits au-dessus pour être capable de décrire une famille d'entités et pas seulement une entité spécifique. Cela signifie que la

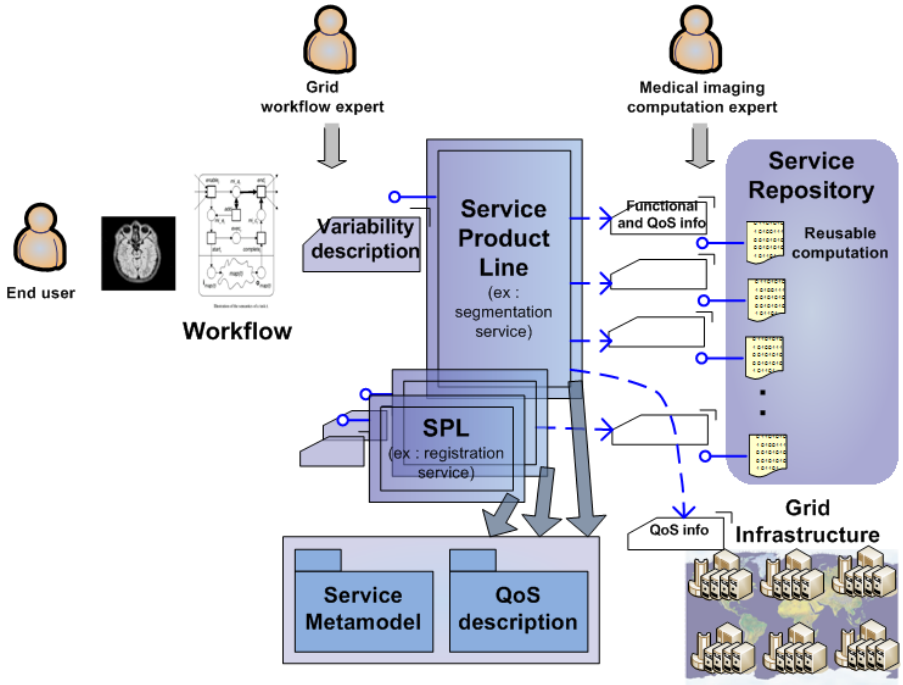

Fig. 4.1 – Aperçu de la ligne de services

SPL doit être en mesure de choisir une entité parmi plusieurs variantes possibles et de sélectionner une partie optionelle. Cette étape est appelée processus de dérivation.

Si nous comparons avec le concept de classe générique dans les langages orientés objets, la description d'une famille d'entités correspond à la description d'une classe générique et le processus de dérivation correspond à l'instanciation de cette classe. A la manière du processus d'instantiation qui a besoin de vérifier la compatibilité de type des paramètres génériques, le processus de dérivation doit s'assurer que les contraintes qui existent entre les différentes parties optionelles ou variantes (typiquement une exclusion mutuelle ou une dépendance) sont conformes aux choix de l'utilisateur. Récemment, la maîtrise de la variabilité dans les premières phases de développement logiciel a été étudié [36].

Pour pouvoir modéliser la variabilité des services et les capacités de la ligne de produits, nous utilisons l'approche ingénierie dirigée par les modèles (IDM) [20], ce qui nous permettra de proposer un métamodèle. IDM est une évolution récente des techniques logicielles, qui place les modèles au centre du processus de développement. Ainsi, les modèles ne sont plus uniquement une source de documentation contemplative et ils permettent de générer automatiquement du code via la transformation de modèles. Les modèles sont indépendants des plateformes d'exécution et des technologies de services, ce qui, on l'a vu tout au long du document, est crucial sur la grille.

4.3 Principes de la ligne de services

Dans notre contexte, nous voulons fournir aux développeurs logiciels les capacités pour manipuler un service d'un domaine particulier (i.e imagerie médicale) non plus comme un seul service mais comme une ligne de produits de services, ce qui autorise de construire facilement des services dérivés incluant des fonctionnalités et des propriétés de QoS s'adaptant aux besoins. Cette conception s'apparente à l'idée qu'un *web service* est une entité informatique, qui est capable (par invocation) de réaliser les objectifs de l'utilisateur, tandis qu'un service est, par opposition, la valeur effective résultante de cette invocation [2]. Ainsi, une ligne de services peut être vu comme un web service qui est en mesure de

fournir l'accès à de multiples services, membres de la ligne. De plus, un web service peut être décrit à travers de multiples interfaces, mais a une et une seule capacité. En spécifiant les éléments communs et les différences dans le but de décrire la généralité d'une entité, cela conduit à la construction d'une interface générique, décrivant indirectement de multiples interfaces.

La figure 4.1 illustre les principes de notre approche. Des services patrimoniaux, – dont les codes ne sont plus modifiables – sont disponibles dans un entrepôt. Supposons que ces services forment une collection d'algorithmes pour le traitement d'images. Dans cet entrepôt, on peut trouver un sous ensemble de services dédiés à la segmentation d'images (voir section 3.3.2). Intuitivement, ces services peuvent être manipulés, via leurs interfaces, comme des services de type segmentation et ainsi être regroupées au sein d'une même ligne de produits "Segmentation". Et de ce fait, un service de l'entrepôt peut être considéré comme le résultat du processus de dérivation de la ligne de services. Comme analysé précédemment (voir section 3.3), la dérivation consistera principalement à manipuler la variabilité de QoS des services. Un aspect à maîtriser est par exemple de gérer les inter-dépendances entre les propriétés de QoS. Aussi est-il nécessaire d'avoir une vue d'ensemble des dépendances complexes, qui peuvent affecter un grand nombre de points variables. La représentation en tant qu'entité de première classe des dépendances, et des relations entre ces dépendances, est un besoin important qui doit être intégré dans le modèle.

La variabilité des services de la grille est capturée dans un métamodèle. Celui-ci permet de raisonner sur les services et ainsi d'effectuer les opérations comme la sélection, l'adaptation, la surveillance, *etc.* Comme le métamodèle décrit toutes les lignes de produits d'un domaine d'application donné, ici l'imagerie médicale, il doit être capable de représenter toutes les propriétés fonctionelles et non fonctionelles communes ainsi que les variations des services de l'entrepôt. Un sous ensemble du métamodèle proposé est dédié à la variabilité de la QoS. De telles informations sont alors utilisables pour comparer des membres de la ligne de produit. L'influence du contexte d'exécution et l'impact de la grille sur la ligne de produits pourront être pris en compte grâce à un second métamodèle qui manipulerait les informations de l'infrastructure.

4.4 Métamodélisation avec les feature diagrams

Notre choix de technologie s'est orienté vers les *feature diagrams* [8]. Avec cette technique, les *features* se réfèrent aussi bien à des caractéristiques fonctionnelles que non fonctionnelles. De plus, les features sont hiérarchiquement organisées dans une structure d'arbre, ce qui permet de facilement exprimer une taxonomie, comme on a pu le faire en classifiant des dimensions de QoS ou en caractérisant les concepts spécifiques aux propriétés de QoS.

Chaque feature peut avoir un attribut qui représente une propriété numérique ou textuelle. La variabilité s'exprime avec les cardinalités de feature, les groupes de feature et les attributs de feature. La cardinalité d'un feature signifie que dans l'arbre de feature, on peut spécifier combien de fois une feature apparaît : une cardinalité [0..1] représente le choix qu'une feature apparaîsse une seule fois, ou pas du tout. Un groupe de features exprime un choix entre plusieurs sous-features. Les attributs de feature peuvent aussi représenter des choix étant donné le fait que la valeur des attributs de feature peut rester non spécifié. Enfin, des contraintes entre les features peuvent être exprimées dans un langage spécifique (par exemple XPath dans [8]). La ligne de produits est ainsi représenté par une structure d'arbre. L'instanciation de la ligne se réalise en spécifiant la cardinalité des features, en sélectionnant les features d'un groupe de features ou en spécifiant la valeur des attributs.

4.5 Instanciation de la ligne de services de segmentation

Le scénario d'utilisation est le même que pour la figure 3.2 : l'expert médical exige les mêmes qualités de service. On considère deux services de segmentation d'images médicales $S1$ et $S3$. Sur la figure 4.2, le contexte fonctionnel du service $S1$ est représenté par un arbre de features, déja instancié.

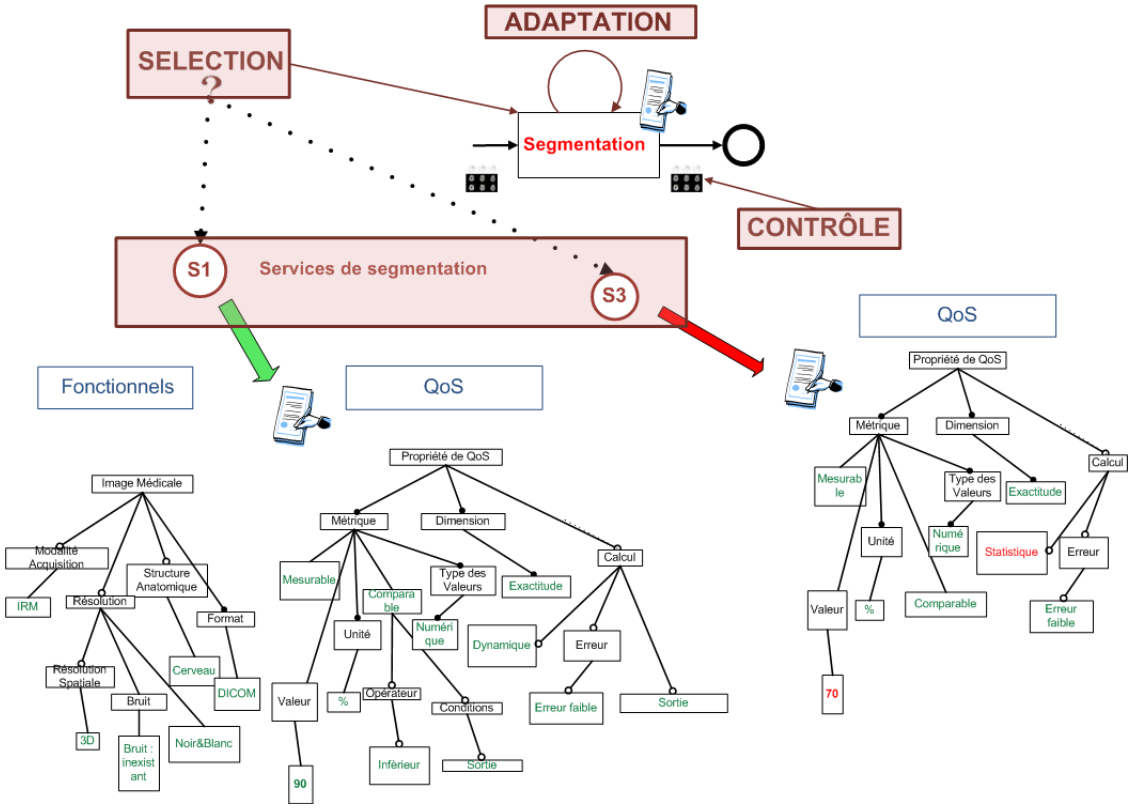

Fig. 4.2 – Variabilité de QoS dans un workflow pour le traitement d'images médicales

Les valeurs des attributs de feature, en vert, ont été affectées[1]. Le service est ainsi à même de manipuler l'entrée de ce workflow. Avec ces informations, il est maintenant capable d'exhiber une qualité de service *fidélité*. On considère que le service $S3$ est lui aussi capable de répondre aux contraintes fonctionnelles. On remarque que le service $S3$ n'est pas capable de calculer sa QoS dynamiquement, en utilisant l'image traitée en sortie ; il ne peut que fournir une mesure statistique. La dynamicité et l'utilisation de la sortie pour calculer la QoS étaient deux propriétés optionelles, ce qui est exprimé dans le métamodèle. Par opposition, le service $S1$ est quant à lui à même de répondre au besoin. En conclusion les services $S1$ et $S3$ sont deux membres d'une ligne de services, s'appuyant sur un même métamodèle de feature diagram. Les instanciations de la ligne permettront d'effectuer les différentes opérations déterminées prédédemment (sélection, adaptation, contrôle, *etc.*.).

[1]Les premiers feature diagrams exprimant la variabilité de la QoS et celle d'une image médicale sont donnés en annexe

Chapitre 5

Conclusion et perspectives

Notre travail s'inscrit dans le domaine des grilles scientifiques utilisées sous forme d'architecture orientée services. Notre objectif était de proposer des mécanismes architecturaux afin de faciliter la manipulation des services sur la grille, tout en validant l'approche dans le contexte de l'analyse d'images médicales. L'étude de la problématique et du domaine ciblé nous a permis d'établir précisément le cahier des charges d'une solution fondée sur les lignes de produits et l'ingénierie des modèles. Cette étude a ainsi permis de préciser la problématique de l'imagerie médicale et l'intérêt d'utiliser la grille pour mettre en œuvre ses applications sous forme de services. Elle a aussi mis en évidence l'impact de la QoS sur le choix et l'exécution des services. Notre analyse a ensuite établi la définition du degré de variabilité des aspects fonctionnels et non fonctionnels (plus particulièrement la QoS) de ces services. Puis, nous avons caractérisé la variabilité dans l'imagerie médicale, ce qui a constitué un cas d'étude idéal, puisque les algorithmes de traitement d'images ont révélé une complexité et une diversité qualitative extrêment importante.

A partir de cela, nous avons proposé *i)* un premier métamodèle permettant de décrire la variabilité des services dans l'imagerie médicale et plus particulièrement pour la segmentation d'images, *ii)* une architecture pour l'utilisation des lignes de services et en particulier la sélection de services adaptés aux besoins de l'utilisateur, spécifié sous forme de contraintes ou de choix.

Ce travail ouvre de nombreuses perspectives. Celles-ci concernent à la fois une amélioration du métamodèle, mais surtout l'extension de celui-ci pour ne plus concerner seulement des services élémentaires mais pour prendre en compte les contraintes de chaines de traitement complètes. Leur description pourra reposer sur des services composites (orchestration de services), et il s'agira alors de déterminer, par exemple, en quoi un service de la chaîne est *optionel* ou *adaptable*. La prise en compte de la complexité de la grille, qui a été évoquée dans notre étude, est une autre piste à explorer car notre objectif à long terme est de proposer une approche complète qui permette aux membres du corps médical de bénéficier d'un support dans leur travail quotidien. Pour permettre une collaboration entre les différents acteurs de la grille, dont les objectifs divergent et les expertises se complètent, la spécification de propriétés non fonctionnelles à travers diverses perspectives nous apparaît incontournable. Enfin, les opérations de sélection, d'adaptation ou de contrôle doivent être implémentées dans la ligne de services, ce qui implique d'expliciter l'architecture décrite et de maîtriser la complexité de problèmes comme l'ordonancement de tâches multi-critères.

La participation de l'équipe dans plusieurs projets et collaborations dans les domaines de l'imagerie médicale et des lignes de produits permettront de valider les avancées et résultats vis-à-vis des perspectives envisagées. La validation de ce travail est d'ailleurs une de nos premières préoccupations. Pour réaliser la mise en oeuvre de lignes de produits nous nous appuierons sur le savoir faire de l'équipe et sur ses collaborations académiques ou industrielles ; nous exploiterons plus particulièrement les résultats de projets comme OpenEmbeDD basés sur la plateforme Eclipse, et plus précisement sur EMF – une mise en oeuvre du MOF (Meta Object Facility) de l'OMG. Il sera aussi particulièrement intéressant de confronter nos résultats avec ceux de la COLOR COSIMO (avec le projet PULSAR de l'INRIA), dans laquelle la variabilité est étudiée dans le domaine de la supervision vidéo notamment pour l'aide aux personnes.

Bibliographie

[1] R. Acharya, R. Wasserman, J. Sevens, and C. Hinojosa. Biomedical Imaging Modalities : a Tutorial. *Computerized Medical Imaging and Graphics (CMIG)*, 19(1) :3–25, 1995.

[2] Ziv Baida, Jaap Gordijn, and Borys Omelayenko. A shared service terminology for online service provisioning. In *ICEC '04 : Proceedings of the 6th international conference on Electronic commerce*, pages 1–10, New York, NY, USA, 2004. ACM.

[3] Jan Bosch, Gert Florijn, Danny Greefhorst, Juha Kuusela, J. Henk Obbink, and Klaus Pohl. Variability issues in software product lines. In Frank van der Linden, editor, *PFE*, volume 2290 of *Lecture Notes in Computer Science*, pages 13–21. Springer, 2001.

[4] Ivona Brandic, Rainer Schmidt, Gerhard Engelbrecht, and Siegfried Benkner. Towards quality of service support for grid workflows. In *Proceedings of the European Grid Conference 2005 (EGC2005)*, Amsterdam, The Netherlands, 2 2005. Springer Verlag.

[5] J. S. Cardoso and L. Corte-Real. Toward a generic evaluation of image segmentation. *Image Processing, IEEE Transactions on*, 14(11) :1773–1782, 2005.

[6] Jorge Cardoso, Amit P. Sheth, John A. Miller, Jonathan Arnold, and Krys Kochut. Quality of service for workflows and web service processes. *J. Web Sem.*, 1(3) :281–308, 2004.

[7] Sholom Cohen, editor. *Service Oriented Architectures and Product Lines - What is the Connection ? SOAPL'07, Workshop at SPLC'07, 10 September 2007, Kyoto*, 2007.

[8] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. Formalizing Cardinality-based Feature Models and their Specialization. In *Software Process Improvement and Practice*, pages 7–29, 2005.

[9] Glen Dobson, Russell Lock, and Ian Sommerville. Qosont : a qos ontology for service-centric systems. In *EUROMICRO '05 : Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 80–87, Washington, DC, USA, 2005. IEEE Computer Society.

[10] James C. Duncan and Nicholas Ayache. Medical image analysis : Progress over two decades and the challenges ahead. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1) :85–106, 2000.

[11] Ian Foster. What is the grid ? - a three point checklist. *GRIDtoday*, 1(6), July 2002.

[12] Ian Foster, Carl Kesselman, J. Nick, and Stevens Tuecke. The Physiology of the Grid : An Open Grid Services Architecture for Distributed Systems Integration. Technical report, Open Grid Service Infrastructure WG, GGF, June 2002.

[13] Ian Foster, Carl Kesselman, and Stevens Tuecke. The Anatomy of the Grid : Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, 5(3), 2001.

[14] S. Frlund and J. Koistinen. Quality-of-service specification in distributed object systems, 1998.

[15] Dennis Gannon. *Component Architectures and Services : From Application Construction to Scientific Workflows*, chapter 12, pages 174–189. In [28], 2007.

[16] Tristan Glatard, Xavier Pennec, and Johan Montagnat. Performance evaluation of grid-enabled registration algorithms using bronze-standards. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI'06) AR=40*, LNCS 4191, pages 152–160, Copenhagen, Denmark, October 2006. Springer.

[17] Vincenzo Grassi and Simone Patella. Reliability prediction for service-oriented computing environments. *IEEE Internet Computing*, 10(3) :43–49, 2006.

[18] San-Yih Hwang, Haojun Wang, Jian Tang, and Jaideep Srivastava. A probabilistic approach to modeling and estimating the qos of web-services-based workflows. *Inf. Sci.*, 177(23) :5484–5503, 2007.

[19] P. Jannin, J.M. Fitzpatrick, D.J. Hawkes, Xavier Pennec, R. Shahidi, and M.W. Vannier. Validation of Medical Image Processing in Image-guided Therapy. *IEEE Transactions on Medical Imaging (TMI)*, 21(12) :1445–1449, December 2002.

[20] Stuart Kent. Model driven engineering. In *IFM '02 : Proceedings of the Third International Conference on Integrated Formal Methods*, pages 286–298, London, UK, 2002. Springer-Verlag.

[21] Jeffrey O. Kephart. Research challenges of autonomic computing. In *ICSE '05 : Proceedings of the 27th international conference on Software engineering*, pages 15–22, New York, NY, USA, 2005. ACM.

[22] Diane Lingrand, Johan Montagnat, and Tristan Glatard. Estimation of latency on production grid over several weeks. In *ICT4Health*, Manila, Philippines, February 2008. Oncomedia.

[23] Alessandro Maccari and Antti-Pekka Tuovinen. System family architectures : Current challenges at nokia. In *IW-SAPF-3 : Proceedings of the International Workshop on Software Architectures for Product Families*, pages 107–115, London, UK, 2000. Springer-Verlag.

[24] Anbazhagan Mani and Arun Nagarajan. Understanding quality of service for web services, January 2002.

[25] Chintan Patel, Kaustubh Supekar, and Yugyung Lee. A qos oriented framework for adaptive management of web service based workflows. In *DEXA*, pages 826–835, 2003.

[26] Aleksandra Popovic, Matas de la Fuente, Martin Engelhardt, and Klaus Radermacher. Statistical validation metric for accuracy assessment in medical image segmentation. *International Journal of Computer Assisted Radiology and Surgery*, 2(3-4) :169–181, December 2007.

[27] Thomas Ropars, Emmanuel Jeanvoine, and Christine Morin. Providing qos in a grid application monitoring service. Research Report RR-6070, IRISA/Paris Research group, Université de Rennes 1, EDF R&D, INRIA, IRISA, Rennes, France, 2006.

[28] Ian Taylor, Ewa Deelman, Dennis Gannon, and Matthew Shields. *Workflows for e-Science*. Springer-Verlag, 2007.

[29] Jayaram K. Udupa, Vicki R. Leblanc, Ying Zhuge, Celina Imielinska, Hilary Schmidt, Leanne M. Currie, Bruce E. Hirsch, and James Woodburn. A framework for evaluating image segmentation algorithms. *Computerized Medical Imaging and Graphics*, 30(2) :75–87, March 2006.

[30] Marek Wieczorek, Andreas Hoheisel, and Radu Prodan. Taxonomy of the multi-criteria grid workflow scheduling problem. In *CoreGrid Workshop*, 2007.

[31] Jia Yu and Rajkumar Buyya. A Taxonomy of Workflow Management Systems for Grid Computing. *Journal of Grid Computing (JGC)*, 3(3-4) :171 – 200, September 2005.

[32] Jia Yu, Rajkumar Buyya, and Chen-Khong Tham. Cost-based scheduling of scientific workflow application on utility grids. In *e-Science*, pages 140–147. IEEE Computer Society, 2005.

[33] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng. Quality driven web services composition. In *WWW '03 : Proceedings of the 12th international conference on World Wide Web*, pages 411–421, New York, NY, USA, 2003. ACM.

[34] Yu Jin Zhang. A review of recent evaluation methods for image segmentation. In *Signal Processing and its Applications, Sixth International, Symposium on. 2001*, volume 1, pages 148–151, Kuala Lumpur, 2001.

[35] Shanyu Zhao, Virginia Lo, and Chris GauthierDickey. Result verification and trust-based scheduling in peer-to-peer grids. In *P2P '05 : Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing*, pages 31–38, Washington, DC, USA, 2005. IEEE Computer Society.

[36] Tewfik Ziadi and Jean-Marc Jézéquel. *Software Product Lines*, chapter Product Line Engineering with the UML : Deriving Products, pages 557–586. Number ISBN : 978-3-540-33252-7. Springer Verlag, 2006.

Annexe A

Feature Diagrams

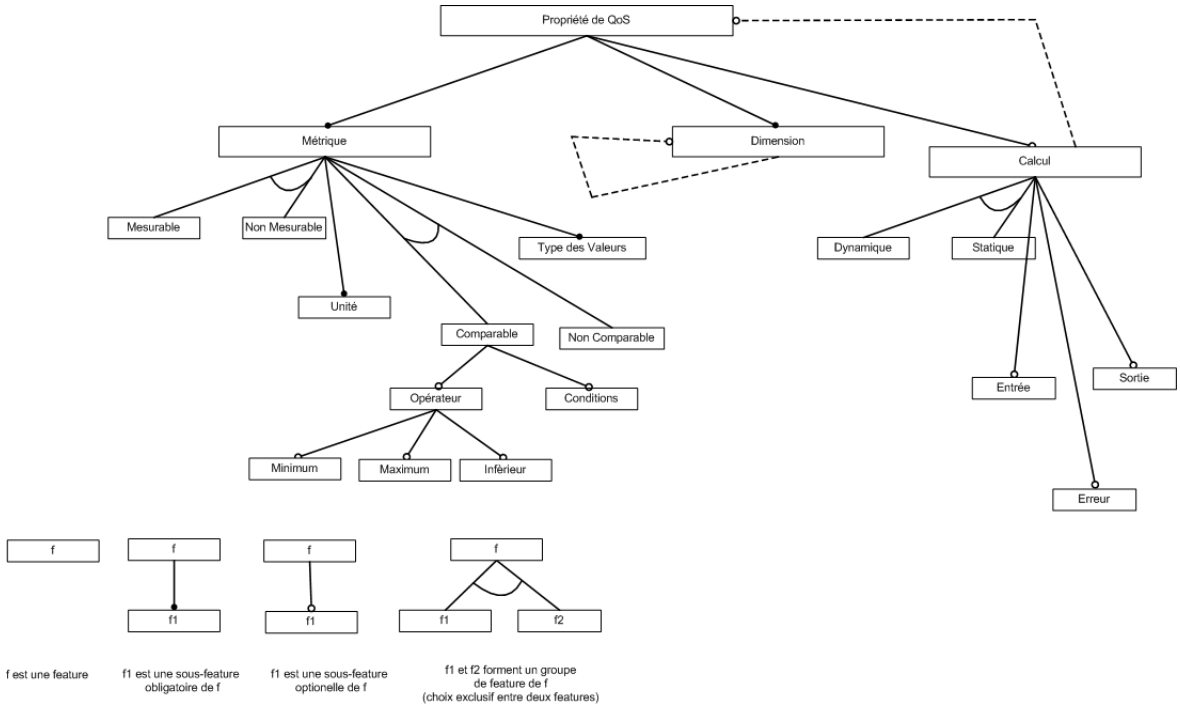

Fig. A.1 – Feature diagram pour exprimer la variabilité de la QoS

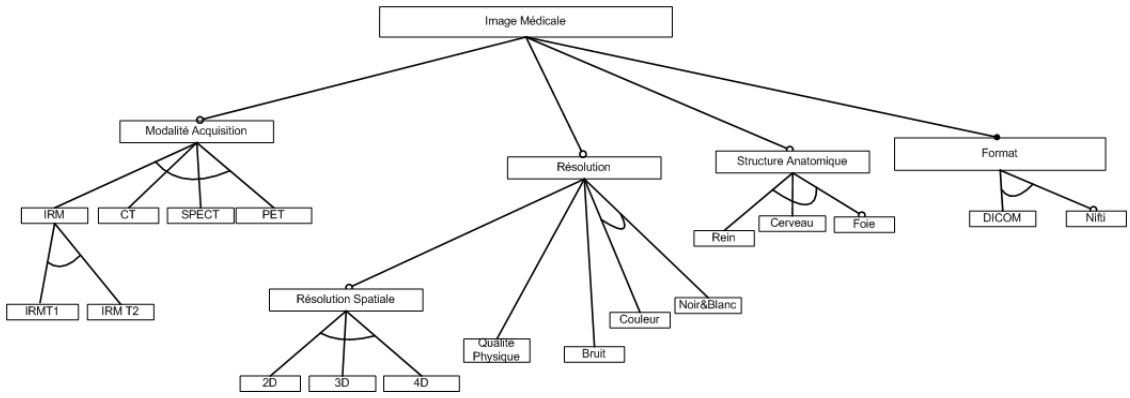

Fig. A.2 – Feature diagram pour exprimer la variabilité d'une image médicale