UNIVERSITÉ DE TOURS

École Doctorale MIPTIS

LABORATOIRE D' INFORMATIQUE FONDAMENTALE ET APPLIQUÉE DE TOURS

THÈSE présenté par :

Marwa BOULAKBECH

soutenue le : 7 Juillet 2020

pour obtenir le grade de : Docteur de l'université de Tours

Discipline/ Spécialité : Informatique

Mashup de Services par Configuration : Application au domaine Touristique

THÈSE DIRIGÉE PAR :

DEVOGELE Thomas	Pr. Université de Tours

RAPPORTEURS :

GAALOUL Walid	Pr. Télécom SudParis
FACI Noura	MCF HDR. Université Claude Bernard Lyon 1

JURY :

CLARAMUNT Christophe	Pr. Ecole navale
CALABRETTO Sylvie	Pr. INSA Lyon
TRONCY Raphaël	MCF . Eurecom
FACI Noura	MCF HDR. Université Claude Bernard Lyon 1
GAALOUL Walid	Pr. Télécom SudParis
DEVOGELE Thomas	Pr. Université de Tours
MESSAI Nizar	MCF. Université de Tours
SAM Yacine	MCF. Université de Tours

Remerciements

Résumé

Le développement du Web 2.0 et l'émergence du Cloud Computing ont engendré une augmentation importante et continuelle des volumes de données collectées, stockées et partagées. Ces dernières, souvent stockées dans des sources de données diverses et appartenant à des organismes et/ou domaines différents, nécessitent pour être exploitées des efforts d'intégrations souvent important, notamment quand les sources sont distribuées et hétérogènes. Nous nous sommes intéressés à une telle problématique, dans le contexte de services de données, en proposant, dans un environnement ouvert et dynamique, une solution d'intégration de données construite à la volée par les utilisateurs afin de résoudre des demandes d'information liées à leurs vies quotidiennes. Nous avions plus particulièrement la construction de plans touristiques en Centre-Val-de-Loire comme domaine d'application. Nous avons proposé une approche basée sur la théorie de la configuration et des techniques de mashup pour interconnecter un ensemble de services et APIs Web en vue de répondre à une requête de plan touristique. Notre solution est destinée à des utilisateurs novices en offrant un environnement de création de services intuitif.

Les paradigmes visuels cachent en effet la complexité typique de la composition des services, de l'intégration des données ainsi que les applications sensibles au contexte pour aider les utilisateurs finaux à créer des applications de mashup personnalisées.Un type spécial d'application Web, appelé Linked Data Mashup, utilise l'initiative Linked Data pour créer une vue intégrée des différentes sources de données liées et exploiter les données sémantiques. Puisqu'il existe des relations logiques sous-jacentes entre les données, une navigation dans le graphe de données explorant ces relations logiques aiderait à stimuler la personnalisation du mashup.

Mots clés : Mashups, Composition APIs Web, développement orienté utilisateur final, Sensibilité au contexte, Linked Open Data.

Abstract

The fast development of powerful rich Internet applications and the growing availability of information on the Web is stimulating researchers to investigate data overload phenomenon. The potential of mashup techniques to address seamless integration of data and services to provide a unified view of the information is presented as an evident solution. However, the opportunity to access a large amount of information does not always in harmonization with the people knowledge. In fact, the data deluge we are confronting today virtually leds people to continuously discover new information that they do not know how to manage and filter to obtain the most suitable to their current need. The recent development of mashup tools has refueled research on end-user approach allowing users without programming skills to create, consume, and manage added value services throught designing their own application.

This thesis presents a novel mashup approach for end-user development. That is, we introduce a domain-specific approach to mashups that is aware of the concepts and conventions of the domain and where the user is comfortable with. The domain-specific approach mashup approach is based on configuration theory that, in generic point of view, allows users aggregate components by selecting appropriate ones whose respect user constraints. We illustrate the configuration based-mashup approach by implementing a mashup plateform regarding the tourism domain to help visitors in customising their trip planning in Loire Valley region. This region is characterized with a cultural heritage that often does not succeed in being completely enhanced. The natural, artistic and cultural resources symbolizing the Loire Valley, above all the smallest and not very popular ones, many times remain hidden for the tourists. This problem becomes even more important when the visitor has a limited visit time. Where eating ? What seeing ? How moving ? These are the typical questions that such a user makes when he/she isplanning for a trip.Even if, there are available services that can be useful for a tourist who unlikely knows where finding them. Therefore, it is necessary to create a framework that can integrate contents and services to support the user inside a certain territorial context. Thus, exchanged and produced data can be exploited by end-user in the design of a mashup application which is seen as composed of a set of single smart spaces making the system “smart”.

In line with visual programming paradigm, our approach is characterized by the adoption of design environments that take advantages of the use of high-level visual abstractions. Visual paradigms indeed hide the complexity typical of service composition, data integration and also the context-awareness apps to assist end-users in the creation of personalized mashup applications. A special kind of web application, called Linked Data Mashup uses the Linked Data initiative to build an integrated view of different Linked Data sources and take advantage of semantic data. Since there are underlying logical relations between the data, a deep convergence of all the data and full considerations of such logical relations would help to boost mashup personalisation.

Keywords : Mashups, Web APIs, End-User Development, Context-awerness, Linked Open Data.

Table des matières

Liste des tableaux

Table des figures

Introduction Générale

0.1 Contexte et Motivation

Au cours de la dernière décennie, les applications distribuées ont évolué à un rythme effréné, s'appuyant sur l'intégration d'une pléthore de services composables pour offrir une multitude de nouvelles fonctionnalités à valeur ajoutée. Cette capacité à créer des services à valeur ajoutée par extension de fonctionnalités est l'une des contributions centrales de l'architecture orientée services. Ainsi, un écosystème de services et d'utilisateurs est créé faisant passer ces derniers de l'inaction à l'interaction.

0.1.1 Mashups de sercvices

Des approches pour la conception d'applications orientées utilisateurs, avec un cycle d'intégration de ressources réalisé à la volée, sont devenues nécessaires de nos jours. Ces applications, connues sous le nom de *Mashup*, sont développées par agrégation de ressources accessibles au moyen d'APIs. Cette technique de création de services composites par les utilisateurs finaux s'appuie sur les principes de flexibilité et de facilité d'utilisation. Pourtant, à l'instar de ce qui s'est produit dans la composition des services Web, les plateformes de mashup développées ont tendance à exposer trop de fonctionnalités et trop de détails techniques, de sorte qu'elles le sont puissantes et flexibles mais ne conviennent qu'aux programmeurs.

En fait, les utilisateurs finaux, qui ne possèdent pas de compétences en programmation, composent potentiellement les différentes ressources sous forme de processus d'intégration ad-hoc afin de satisfaire leurs besoins. Ils sont plutôt préoccupés par des techniques intuitives de mashup de services conçue en termes d'efficacité cognitive et de simplicité sans courbe d'apprentissage abrupte des technologies sous-jacentes. L'objectif des plateformes de mashup étant de guider les utilisateurs dans le processus d'intégration en fournissant des techniques de combinaison adaptées à travers des paradigmes visuels simplifiant le processus de composition.

Le style de composition graphique représente un premier pas vers l'autonomisation des utilisateurs les plus novices et le processus interactif permet d'assurer une meilleure réponse à leurs besoins. En effet, reconnaissant le potentiel émergent des mashups, diverses initiatives ont été mises en place pour faciliter le développement des mashups en fournissant des environnements graphiques permettant à l'utilisateur de composer par lui-même son application composite. Cependant, malgré ces efforts de simplification dans le domaine des mashups, les utilisateurs finaux rencontrent encore des difficultés lorsqu'ils essaient de donner un sens à cette technologie. Il n'est, en effet, toujours pas évident pour un utilisateur de savoir comment exprimer sa demande.

L'idée, donc, est d'assister les utilisateurs finaux dans l'expression du mashup à travers une progression étape par étape s'appuyant sur des abstractions de haut niveau et des paradigmes d'interaction visuels adéquats. Alors que la simplicité est notre objectif, il est important de se focaliser sur un domaine bien déterminé permettant de mieux adapter le paradigme d'agrégation aux besoins spécifiques du domaine.

0.1.2 Tourisme : un domaine multi-dimensionnel

Les applications de mashup sont des systèmes centrés sur l'utilisateur dont le fonctionnement évolue en fonction des besoins et des contraintes d'un utilisateur spécifique dans des domaines d'applications aussi divers que les villes intelligentes, le tourisme intelligent, l'agriculture intelligente et la prestation de soins de santé intelligents. Nous nous focalisons dans cette thèse sur le domaine touristique. Par exemple, il pourrait s'agir d'une application de planification de séjours touristiques qui permet à l'utilisateur d'intégrer plusieurs services (service de points d'intérêt, service de transport, service de restauration, etc.). Dans ce cas, le choix des services et des objectifs de composition est déterminé par l'environnement de l'utilisateur, ses objectifs, ses préférences, ses contraintes et ses préférences personnelles.

De plus, les services sont souvent étroitement liés au contexte dans lequel ils sont exécutés, par exemple pour le domaine du tourisme, l'exécution d'un service de recherche de points d'intérêt peut dépendre du type de l'activité touristique à pratiquer, des horaires d'ouverture, des conditions météorologiques, etc. D'ailleurs, le contexte tend à être volatil, c'est-à-dire que des événements exogènes peuvent changer l'état du contexte et donc affecter l'exécution de la composition. Par conséquent, pour être en mesure de produire des compositions cohérentes avec le contexte environnant, il est important d'avoir un modèle de service adapté au contexte qui reflète les caractéristiques contextuelles des services. Nous sommes ainsi face à un besoin d'une stratégie de mashup qui soit adaptative étant donné que les environnements avec lesquels les utilisateurs interagissent sont dynamiques par nature (changement de lieu, facteurs environnementaux, exigences des utilisateurs).

L'adaptation se traduit, ainsi par la sensibilité au contexte fournissant une meilleure expérience utilisateur.

D'un autre côté, la croissance explosive et la variété de l'information disponible sur le Web ont souvent submergé les utilisateurs. Cette surcharge d'information peut porter à confusion au lieu de produire un bénéfice. Les systèmes de recommandation se sont révélés être un moyen précieux pour faire face au problème de la surcharge d'information. Ils apparaissent comme une solution naturelle dans un tel environnement afin d'aider les utilisateurs à accéder aux ressources désirées. Les systèmes de recommandation sont devenus de plus en plus populaires et sont aujourd'hui un composant principal de beaucoup d'applications dans différents domaines. Une multitude d'applications offrant aux utilisateurs un grand nombre de fonctionnalités disparates sont disponibles dans le domaine touristique telles que les guides de voyage, les cartes, les sites institutionnels en ligne et les blogs de voyage.

Or, du point de vue de l'usager, exploiter correctement les services disponibles pour organiser des trajets répondant aux attentes personnelles devient une tâche complexe. En effet, découvrir et sélectionner les services appropriés dans un domaine ouvert et en constante expansion est un processus de plus en plus intimidant qui comporte de multiples étapes. Pour n'en citer que quelques-unes, choisir parmi plusieurs options de transport, les hôtels où se loger ou encore rechercher des attractions touristiques aux différentes destinations. Par conséquent, les utilisateurs doivent réaliser le mashup des différents services ayant une couverture partielle afin de concevoir une stratégie intelligente de planification en devinant le temps nécessaire pour visiter chaque attraction et pour se déplacer d'un endroit à un autre. Ceci car ces systèmes de recommandation classiques sont limités à des recommandations d'éléments individuels ou des listes d'éléments qui sont souvent indépendants les uns des autres.

Néanmoins, les utilisateurs sont potentiellement intéressés par des éléments organisés en paquets cohérents plutôt qu'en listes de classement, ce qui constituera une expérience exploratoire améliorée.

De ce fait, les services composites, organisant les attractions, présentent des relations spatio-temporelles complexes intrinsèques. Par exemple, un itinéraire ne doit comprendre que des attractions qui sont spatio-temporellement corrélées. De plus, il peut exister une notion de compatibilité entre les éléments de l'itinéraire, qui peut être modélisée comme des contraintes que l'utilisateur peut spécifier. A titre d'exemple, "pas plus de trois musées ", "la distance totale couverte pour visiter les attractions devrait être inférieure à 20 km", etc.

0.1.3 Personnalisation de mashup

Bien que les systèmes de recommandation soient facilement accessibles et de plus en plus populaires, ces applications sont loin de fournir des recommandations personnalisées. Les suggestions proposées sont basées sur des lieux d'intérêt qui sont jugés populaires et négligent l'aspect de personnalisation. En particulier, ces ressources reflètent le point de vue de leurs auteurs, et ils peuvent ne pas être des sources d'information faisant autorité lorsque les préférences touristiques divergent des préférences les plus populaires. Et les applications qui essayent de s'attaquer à la personnalisation se limitent aux offres préétablies qui ne sont pas flexibles.

Les approches de type « one-size-fit-all » proposant des fonctionnalités hautement standardisées pour servir le plus grand nombre d'utilisateurs possible ne sont plus appropriées. En effet, l'utilisateur est unique, ce qui conduit à la différence des exigences, et ce qui demande de plus en plus d'adaptation aux besoins spécifiques.

Par conséquent, proposer un service composite personnalisé qui combine un ensemble de services respectant les préférences des utilisateurs et tenant compte des contraintes qui s'appliquent aux éléments consécutifs et à l'ensemble de la séquence devient un réel besoin. En outre, les utilisateurs n'ont pas forcément une idée claire de leurs objectifs, en particuliers les visiteurs des villes qui ne savent pas comment organiser leurs séjours. C'est pourquoi il est crucial que le processus de personnalisation puisse guider l'utilisateur afin de garantir sa satisfaction. Pour information, un utilisateur qui ne dispose d'aucune aide a moins d'une chance sur 100 de trouver un service personnalisé satisfaisant [Triki, 2016] ; notamment dans le cas de service complexe comportant un grand nombre de composants.

Ces avantages sont particulièrement observés durant l'approche de configuration qui assiste l'utilisateur dans le processus de personnalisation. En fait, la configuration est en train de devenir une fonctionnalité sous-jacente émergente, en particulier pour la personnalisation dans un environnement dynamique où les utilisateurs finaux peuvent sélectionner leurs exigences pour atteindre un objectif spécifique. Il s'agit d'un processus de navigation dans une liste de choix qui peut aider les utilisateurs à construire leurs propres solutions par le biais d'une procédure interactive. La configuration interactive a pour but d'informer l'utilisateur en temps réel des caractéristiques souhaitables / possibles en fonction de ses choix tout en garantissant le respect des contraintes. En effet, la configuration exécute un raffinement pas à pas des contraintes pour réduire progressivement l'espace de recherche du problème.

Ce qui rend la configuration plus avantageuse comparée à la recommandation est que la première implique l'utilisateur dans la sélection et la composition du service sur mesure. Cette stratégie de personnalisation offre une réelle expérience du sur mesure. A la différence

de la recommandation qui peut ne pas satisfaire l'utilisateur, la configuration lui offre exactement ce qu'il veut car c'est lui-même qui contrôle le processus de personnalisation.

La configuration peut être réalisée au moyen d'interfaces faciles à utiliser où les utilisateurs finaux pourraient appliquer des règles de configuration contextuelles afin de composer un service dans une logique déclarative.

L'idée clé de cette thèse est d'exploiter le potentiel de la configuration pour le mashup de services complexes personnalisables avec comme domaine d'application le tourisme. Ceci constitue un enjeu majeur qui n'est pas encore très exploré par la recherche notamment dans l'industrie du patrimoine culturel. Les services personnalisés sont encore recherchés dans le patrimoine culturel à contrario des guides mobiles et des technologies du web social qui sont courants. Afin de faciliter la planification des séjours touristiques, il serait intéressant de faire le mashup de services complémentaires au lieu de proposer des services qui fournissent des fonctionnalités similaires. Le servie composite ainsi généré peut être configuré par les utilisateurs en faisant correspondre leur exigences aux contraintes du domaine.

0.2 Problématique

L'émergence de l'idée « tout est comme service » (Xaas) [Duan *et al.*, 2015] a particulièrement amplifié l'écosystème des services faisant vivre l'utilisateur dans l'infobésité [Vulbeau, 2015, Ledent *et al.*, 2017]. C'est le sentiment d'être submergé par l'information qui caractérise une saturation chronique d'information lié à l'explosion de la production et de la circulation des données sur le Web. Le phénomène de l'infobésité reconnu souvent par le problème de la surcharge d'information constitue un défi en soi pour plusieurs secteurs économiques. Le tourisme est le secteur le plus confronté à cette problématique. Un visiteur souhaitant organiser un séjour touristique se retrouve à explorer de multiples sites et applications Web à la recherche de l'information appropriée. L'hyper choix présenté par ces sources d'information le conduit vers un labyrinthe informationnel sans sortie. Il sera amené à passer beaucoup de temps dans une tache fastidieuse de sélection, parmi les longues listes retournées des options qui correspondent le plus à son intérêt global. Faute de bouée de sauvetage, l'utilisateur au bord de la noyade aurait aimé disposer d'une paire de lunettes pour s'accommoder avec la surabondance de l'information.

Donc, le défi auquel nous nous attaquons dans cette thèse se traduit par la capacité de gérer et d'intégrer cette diversité afin de fournir une vue unifiée qui s'adapte aux changements. Cela signifie une stratégie flexible avec des mécanismes de contrôle ouverts offrant le sur mesure, c'est à dire des services personnalisés, vu que non seulement les données, mais aussi les utilisateurs s'élargissent de jour en jour.

En effet, de plus en plus de fonctionnalités sont fournies à partir de services ou d'APIs

Web. Cependant, ces services Web ont été pensés et développés indépendamment et présentent leurs informations de manières très hétérogènes. Et cette hétérogénéité et surabondance de l'information ne facilite pas l'élaboration d'un schéma d'ensemble capable de répondre aux besoins complexes des utilisateurs.

Le mashup se propose de produire de nouveaux services composites qui fusionnent ces informations par l'intégration des vues fragmentées exposées par les services élémentaires. Dans le but d'apporter une réelle valeur ajoutée, les applications de mashup recueillent des données multiples à partir de diverses sources et permettent aux utilisateurs novices connus sous le nom de End-User de regrouper, combiner et visualiser de nouveaux types de données. En plus de l'aspect hétérogène et évolutif des données, l'orientation utilisateur rend le processus de mashup complexe du fait que la composition est réalisée par les utilisateurs finaux qui ne disposent pas necessairement des compétences en programmation pour pouvoir invoquer et enchaîner les services. Le guidage peut être bien utile pour que l'abstraction des services devienne intelligible.

En outre, le mashup devrait fournir une expérience riche et satisfaisante pour les utilisateurs. Or, les applications actuelles permettent juste de répondre à une multitude de besoins basiques. Seulement, les individus ont envie de s'échapper de la standardisation proposée jusqu'à maintenant au profit de plus de liberté et de différenciation. Les voyages au forfait et les packages fixes pré-négociés avec une conception du « tout sous le même toit » ne font plus de succès. Et ceci est valable pour tous les secteurs. Les systèmes de recommandations sont apparus comme une réponse à cette demande des utilisateurs en fournissant un service plus près de leurs attentes et de leur désir de se différencier. Néanmoins, ces systèmes ne peuvent pas être appliqués à des systèmes composites tels que les applications de mashup qui agrègent diverses ressources. En fait, ces dernières ne tiennent pas compte des contraintes globales qui les lient. Bien que des possibilités de mashup personnalisé ont commencé à émerger, c'est encore à l'état embryonnaire. La personnalisation des services composites à savoir le voyage sur mesure ou à la carte est notre deuxième enjeu qui est devenu la clé de voûte d'une tendance qui ne cesse d'aller en s'accroissant.

Dans ce cadre, nous avons soulevé la principale question de recherche suivante :

Question principale : Comment offrir à l'utilisateur, en occurrence un visiteur, une vision globale de l'information recherchée et qui soit personnalisable ?

Afin d'apporter les éléments de réponse à cette question, il est nécessaire de lever les verrous scientifiques suivants.

Question 1 : Comment faire le mashup des services ou APIs Web pour unifier la vue ?

Comme mentionné précédemment, le mashup permet de faciliter l'indigestion d'information et d'éviter le burn-out numérique provoqués par la sur-information. Il s'agit de chercher une stratégie de connexion de services pour offrir aux utilisateurs une perspective plus globale leur permettant la planification des visites touristiques. Ces dernières peuvent se présenter sous forme d'itinéraires de points d'intérêt de différents types, par exemple : monument, parc, musée, restaurant, hôtel, etc ; au lieu des listes triées retournées par les systèmes de recommandation classiques qui sont difficile à exploiter.

L'idée est de trouver un moyen de combiner l'exploration des sites de visites, la restauration et l'hébergement, ce qui est relativement complexe au vu des contraintes qui entrent en jeu. En effet, à chaque point d'intérêt visité correspond un temps et un budget consommés. L'utilisateur dispose d'un temps de visite et d'un budget limités pour faire sa visite. Il faut donc que le mashup puisse respecter ces contraintes dans la composition des parcours. Afin d'apporter une réponse à cette question, une vue d'ensemble des différentes stratégies de composition est présentée dans cette thèse.

En plus, un aspect important des mashups est celui d'offrir une expérience transparente et axée sur l'utilisateur, ce qui nous conduit à s'interroger sur la position de ce dernier dans le processus.

Question 2 : Comment les utilisateurs peuvent-ils être impliqués dans le processus de mashup ?

Le défi étant de fournir une porte d'entrée vers un "Web programmable", où les utilisateurs finaux ont la possibilité de construire facilement par eux-mêmes des applications composites qui fusionnent les ressources hétérogènes afin de satisfaire leurs besoins spécifiques. Le scénario qui se dessine autour du développement de mashup s'oriente vers un fort potentiel d'innovation centré sur l'utilisateur en tant que producteur de la valeur significative. Nous parlons de la « culture participative », le désir et la capacité des utilisateurs à développer leurs propres applications, à réaliser leurs propres idées et à exprimer leurs propres créativités. La vision est centrée sur ce que souhaite l'utilisateur et ce dont il a réellement besoin, partant du principe que les utilisateurs finaux sont les mieux placés pour évaluer et qu'ils jouent un rôle décisif dans le développement.

Néanmoins, les approches proposées dans la littérature ne tiennent pas pleinement compte de cette vision. Elles exigent à l'utilisateur de connaître au minimum les entrées/-sorties des services qu'il invoque et les possibles enchaînements entre eux. C'est dans cette conception participative que nous nous interrogeons sur la méthode de composition que l'approche de mashup proposée par cette thèse devra considérer pour rendre le processus d'agrégation plus intuitif.

— Quels outils faut-il adopter pour faciliter aux utilisateurs l'accès et l'intégration de ressources répondant à leurs besoins sans se soucier de ce qui se passe en arrière-

plan ?

— Comment rendre le processus de mashup plus naturel, intuitif et plus proche du modèle mental des utilisateurs ?

Une autre question se rapporte à l'aspect dynamique des mashups est la suivante.

Question 3 : Quel rôle joue le contexte dans le développement d'application de mashup ?

L'environnement de mashup est par essence dynamique et continuellement en évolution. Le fonctionnement de ces systèmes peut ainsi être perturbé par les modifications du contexte. Et, le tourisme, notre domaine applicatif, s'inscrit dans ce cadre d'influence. En conséquence, il nous paraît évident que la conception doit être faite en considérant l'information contextuelle. Il faut prendre en compte l'évolution du contexte pour pouvoir agir de manière à satisfaire l'utilisateur, puisque, le contexte évolue pendant que l'utilisateur se déplace d'une localisation à une autre, ou pendant qu'il obtient de nouvelles ressources, etc. Cette continuité dans l'évolution du contexte doit s'accompagner d'un fonctionnement continu des systèmes malgré les changements perpétuels du contexte. Les actions fournies par ces applications doivent être dotées de mécanismes qui collectent les informations de contexte, stockent et gèrent ces informations et qui réagissent sur la base de ces dernières. Ainsi, la question qui se pose est : Quels mécanismes faut-il adopter pour traiter l'information contextuelle ?

Vu l'hétérogénéité, la diversité et la qualité variable de cette information contextuelle, il est préférable de faire une catégorisation pour faciliter l'opération d'adaptation. Les différentes catégories qui constituent le contexte sont dépendantes des objectifs du système. En fait, les parcours touristiques doivent être contextuellement cohérents en satisfaisant plusieurs contraintes, telles que spatiales (lieux proches ou éloignés), temporelles (lieux pertinents pour un moment de la journée), sociales (le voyage peut être un déplacement familial), etc. Quels seront donc les dimensions nécessaires pour une application de mashup touristique sensible au contexte ?

Motivés par le fait que l'individu cherche de plus en plus une consommation personnalisée lui permettant d'affirmer sa propre identité, nous nous sommes intéressés à une autre question.

Question 4 : comment rendre le processus de mashup plus personnalisable ?

La personnalisation est aujourd'hui une tendance importante dans divers secteurs notamment le secteur touristique où les visiteurs expriment un désir profond d'expérience personnelle. Par conséquent, le processus de mashup devrait être doté de mécanisme adaptés à la production personnalisée. Ceci nous ramène à mettre en lumière l'importance de considérer conjointement la disparité, le dynamisme et la diversité des données dans la conception de l'application composite. Avec la caractéristique ad-hoc, les applications de mashups personnalisées nécessitent une technique flexible et configurable qui couvre l'en-

semble du processus de l'identification à l'intégration des ressources. Ainsi, quels sont ces outils qui permettent d'offrir un degré de flexibilité et de configurabilité pour une utilisation personnalisée dans les plates-formes de composition ?

Bien qu'il existe déjà des systèmes de recommandation fournissant des solutions personnalisées, ces dernières sont généralement fixes et sans moyen d'adaptation. C'est bien les voyages forfaits et les packages fixes pré-négociés proposés par les applications de recommandation de visites. Il serait plus intéressant d'offrir à l'utilisateur la possibilité de construire son package parfait, comme il l'entend, entièrement personnalisable via la même plateforme. A ces fins, comment supporter le mashup personnalisé considérant les descriptions floues telles que "ouvert pendant les vacances" ?

0.3 Contributions

Face aux problématiques exposées ci-dessus, les contributions des travaux de cette thèse s'inscrivent dans le cadre d'une approche de mashup interactive qui aide les utilisateurs finaux novices dans la composition de services. Le spectre de la contribution est présenté par la figure 1. L'optimisation est une facette qui est prise en charge par une autre équipe en collaboration sur le projet SmartLoire.

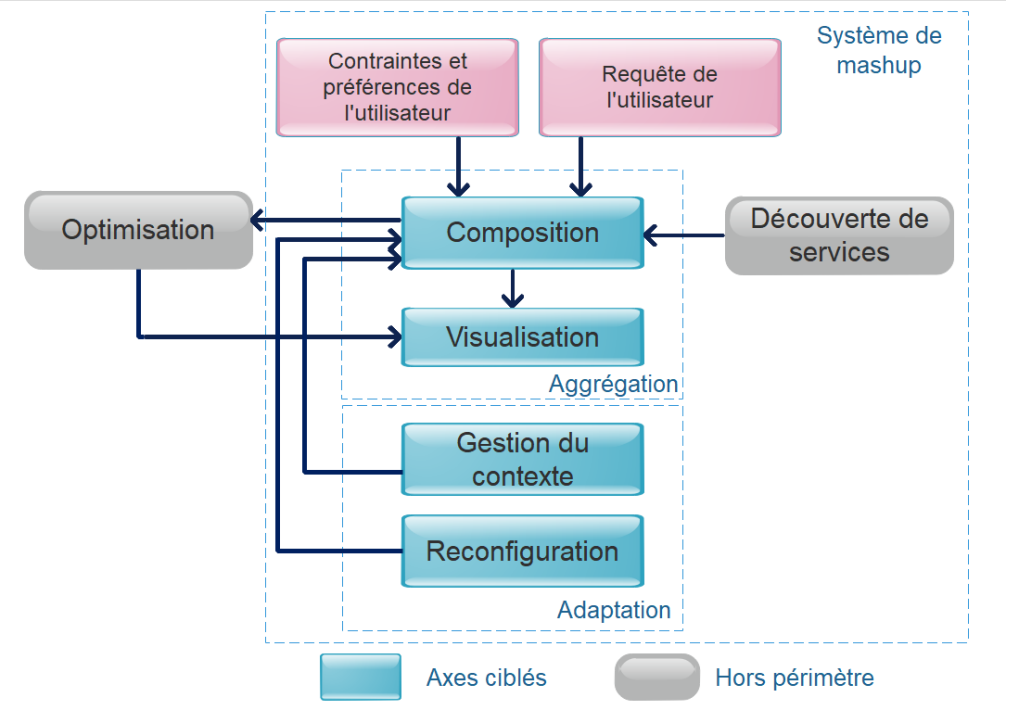


FIGURE 1 – Contributions Principales de la thèse

Les apports de nos travaux relatifs à chacune des problématiques identifiées précédemment comportent les volets suivants.

Le premier volet cible la première problématique présentée. Dans ce cadre, nous proposons un modèle de mashup par configuration orienté utilisateur. De façon interactive, l'algorithme proposé aide à la configuration de service en tenant compte des exigences de l'utilisateur. L'approche consiste à générer de manière incrémentale le mashup sous forme de configurations partielles. Ces dernières constituent des mini mashups établis par des intégrations de données à la volée depuis des sources hétérogènes. Le mashup global est ainsi composé selon une expérience personnalisée qui répond aux demandes et aux attentes des utilisateurs. Le modèle de mashup s'appuie aussi sur des techniques visuelles afin de faciliter le processus de composition pour les utilisateurs novices. Une méthodologie de configuration de mashup spécifique au domaine est présentée dans ce chapitre.

Le second volet consiste à rendre le processus de mashup dynamique par l'intégration de la sensibilité au contexte. Nous avons développé un cadre de conception basé sur la spécification des dimensions contextuelles jugées pertinentes pour notre domaine d'application. Par rapport à d'autres approches de mashup, l'activité de composition et, plus spécifiquement, la sélection des services n'est pas exclusivement déterminée par les caractéristiques fonctionnelles des services disponibles ou par la compatibilité de leurs paramètres d'entrée et de sortie. Au contraire, la spécification initiale des exigences contextuelles permet la sélection dynamique et le filtrage des ressources adéquates, puis l'adaptation des données des services en temps réel. La prise en compte du contexte peut mieux satisfaire les besoins situationnels des utilisateurs. Nous avons adopté le raisonnement à base de règles afin de décrire le comportement contextuel. Une structure arborescente est utilisée pour la modélisation des dimensions contextuelles.

Pour que le système de mashup soit flexible et entièrement personnalisable, nous proposons une stratégie de reconfiguration qui exploite les relations entre les ressources . En effet, nous nous appuyions sur le Web sémantique et les Données Liées pour représenter ces relations via un modèle de graphe de connaissances. Ainsi, une navigation dans ce graphe permet de construire un mashup élastique par ajustement de ressources pour satisfaire la nature multi-objectifs des utilisateurs.

0.4 Structure du document

Le manuscrit est structuré en deux parties. La première partie présente les aspects généraux théoriques du contexte de la thèse. La seconde partie comprend les chapitres qui traitent les contributions de cette thèse.

La Partie 1, *Le contexte en profondeur*, comprend les chapitres 1 et 2.

Le chapitre 1 est consacré à la présentation des concepts de base du développement orienté services et celui orienté utilisateurs à travers le mashup. Nous présentons également dans ce chapitre une introduction à la problématique de personnalisation et les techniques utilisées à cet égard.
Le chapitre 2 présente une revue de l'état de l'art sur les travaux portant sur la composition de services, ceux relatifs aux mashups avec une orientation utilisateurs ainsi que les travaux liés à la personnalisation des services. Nous faisons ensuite une synthèse de cet état de l'art au regard de notre problématique.

La Partie 2, *Le mashup personnalisable*, correspond aux contributions et comprend les chapitres 3, 4, 5 et 6.
Le chapitre 3 présente les différents artefacts de conception, le formalisme et montre également le rôle d'un modèle de domaine, d'un méta-modèle et d'un méta-modèle spécifique au domaine dans le développement du mashup spécifique au domaine. A l'issu de ce chapitre, un modèle fonctionnel de configuration de mashup est proposé.
Le chapitre 4 décrit notre modèle de configuration dirigé par les règles qui porte une solution à la deuxième sous problématique concernant les mashups dynamiques sensibles au contexte. Une couche de contextualisation vient augmenter le modèle fonctionnel de configuration de mashup.
Le chapitre 5 présente le modèle de reconfiguration qui offre aux utilisateurs la possibilité de personnaliser leur mashup de manière aisée. Une couche sémantique via les données liées enrichie, en effet l'approche de configuration afin de fournir un mashup élastique.
Le chapitre 6 est consacré à la mise en œuvre de notre approche de configuration pour le mashup de services personnalisables dans le contexte touristique pour le projet SmartLoire où s'inscrivent en partie les travaux de cette thèse. L'outil CART implémente ainsi notre approche de configuration de mashup. Ensuite, nous présentons l'évaluation par les utilisateur et les résultats obtenus.
Le chapitre 7 constitue la conclusion générale de cette thèse. Nous rapportons les leçons apprises et nous montrons le caractère générique de l'approche qui peut s'appliquer à d'autres domaine. Nous clôturons enfin ce manuscrit par les directions futures qui seraient intéressantes.

Première partie

Le mashup personnalisé

Chapitre 1

Le contexte en profondeur

1.1 Introduction

Nous pressentons dans ce chapitre les notions techniques préliminaires de développement orienté services et son élément clé qui est les services Web. Nous évoquons ensuite le besoin d'intégration de ressources comme réponse à la demande de l'utilisateur. Un accent est mis sur la composition « end-user » avec les applications mashup où le processus de composition de ressources est réalisé par les utilisateurs novices. Nous nous focalisons par la suite sur le besoin de personnalisation et les techniques proposées à cette fin à savoir la customization, la recommandation et la configuration avec les niveaux de granularité de l'item et celui de la séquence.

1.2 Evolution du Web

Au cours des dernières années, le Web a reconnu une importante évolution et un développement continu. Cette avancée technologique considérable a changé notre rapport au monde, aux autres et à l'information. Les nouvelles applications Web émergentes ont envahi notre vie de manière que leur utilisation apparaît non seulement comme acquise mais encore comme indispensable.

Dans sa première génération, le Web était considéré comme un système de cognition. Ce Web 1.0 représentait un lieu permettant de diffuser l'information aux utilisateurs. Au début, le Web offrait un nombre limité d'interactions avec les utilisation où la recherche et la lecture d'information sont les seules possibilités.

Après l'explosion de la bulle Internet, le Web n'avait jamais semblé aussi important et novateur. Les nouveaux sites et applications semblaient avoir quelque chose de commun utilisant des améliorations technologiques, sémantiques, un modèle innovant, reposant sur-

tout sur un renversement de la logique "top-down" du web initial : alors que ce dernier "descendait" vers l'utilisateur pour lui proposer contenus et services, le web 2.0 mettait l'accent sur une nouvelle forme d'interactivité qui place l'utilisateur au centre de l'internet et se veut plus social et collaboratif. Révolution ou simple évolution, ce concept du Web 2.0 se décrit selon les principes suivants [O'reilly, 2009]

— *Le Web en tant que plateforme* : le Web se voit passer d'un service de diffusion d'informations fourni sur Internet à une véritable plateforme pour génération d'applications et de services.

— *Orientation vers les utilisateurs* : dans cette nouvelle vision, l'implication des utilisateurs dans le réseau devient le facteur-clé pour le Web 2.0. En effet, le centre d'intérêt de la technologie se déplace vers les utilisateurs. Il était peu axé sur la place des personnes dans l'image de l'ancienne génération du Web. Comme la technologie vise à améliorer notre vie et la faciliter, les utilisateurs et leurs besoins doivent être l'orientation de toute vision du logiciel. Le Web 2.0 tourne fondamentalement autour de nous. Il cherche à s'assurer que nous nous engageons, participons et collaborons ensemble, que nous nous faisons mutuellement confiance et que nous nous enrichissons mutuellement dans le processus de développement d'applications. Cette intégration des utilisateurs dans l'équation technologique est le facteur majeur du Web collaboratif.

— *Modèle de programmation léger* : le Web 2.0 repose aussi sur une autre idée phare qui est la simplicité d'utilisation côté usager par le biais des techniques intuitives et interactives. Les applications Web actuelles offrent un contenu de plus en plus riche et de meilleures expériences utilisateur. Grâce aux technologies telles que JavaScript et AJAX, la ligne de démarcation entre le client autonome et le serveur se fait brouiller voir complètement disparu. Les données peuvent être récupérées directement à partir des services qui fournissent le contenu et peuvent ensuite être intégrées directement chez le client. Ce principe clé conduit à un nouveau modèle de programmation. Le modèle de programmation doit être léger pour permettre la création de systèmes faiblement couplés et rendre les composantes réutilisables individuellement favorisant l'innovation par l'assemblage où les applications modernes se développent de manière originale et efficace.

1.3 Paradigme orienté services

L'idée de connecter et d'assembler facilement des composants faiblement couplés afin de créer des applications agiles a provoqué une remise en question de certaines pratiques de développement traditionnelles entraînant l'émergence d'un nouveau paradigme. Ce nouveau paradigme, le développement orienté services, est un modèle de développement utilisant

les services Web comme blocs de construction pour la création d'applications composites distribuées [Bouguettaya *et al.*, 2014]. L'écosystème des services Web repose sur une architecture communément appelée Architecture Orientée Services (AOS). Il s'agit d'une architecture logique pour la conception de systèmes logiciels fournissant des services exploitables soit directement par l'utilisateur final, soit par d'autres services distribués dans un réseau, via des interfaces [Papazoglou, 2008]. L'AOS s'articule autours de trois rôles comme il est illustré par la figure 1.1. Le service Web est mis en œuvre et rendu accessible au public par un fournisseur de services. Ce dernier publie un descripteur de service décrivant ses caractéristiques en terme d'identifiant où y accéder (un URI) et comment y accéder (à l'aide de quel protocole de communication). Les services Web sont stockés dans un registre dédié qui regroupe les descripteurs de services de plusieurs fournisseurs. Le client souhaitant utiliser une fonctionnalité peut interroger le registre à la recherche de services Web appropriés obtenant comme réponse un descripteur de service. Afin d'invoquer le service correctement, le client doit suivre les instructions décrites dans le descripteur récupéré.

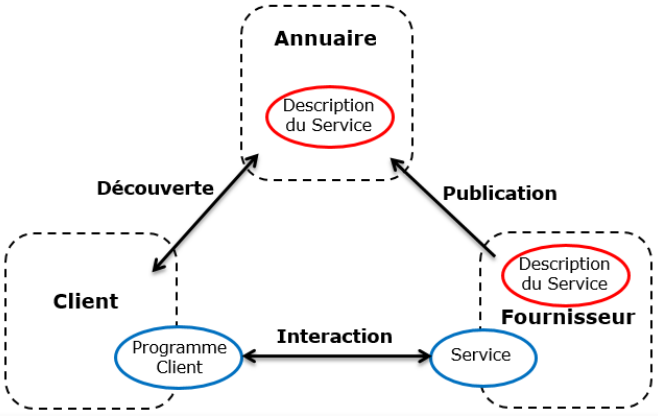


FIGURE 1.1 – Architecture orientée services

L'écosystème de services Web facilement réutilisables fût le concept principal sur lequel le développement orienté services s'appuie. Nous détaillons dans la suite cette notion.

1.3.1 Services Web

Le concept de service Web peut se définir en tant qu'entité logicielle exposant ses propos via une interface qui décrit les fonctionnalités offertes par le service et les informations requises pour assurer son bon fonctionnement. Il s'appuie sur des protocoles Web standards tels que HTTP, SMTP, SOAP, etc, permettant à un client d'invoquer et d'interagir avec

l'application distante. Exposé en tant que boîte noire, le service permettra de représenter ses fonctionnalités à haut-niveau et d'être composé avec d'autres services à travers une couche de médiation.

Selon [Papazoglou, 2008], un service Web peut être une tâche commerciale autonome, un processus métier complet (un processus de commande automatisé), une application complète (une librairie en ligne) ou une ressource (une base de données accessible via un service). Donc le service Web encapsule un module logiciel réalisant une tâche spécifique.

Il existe deux grands types de technologies et de philosophies de services Web : les services de type SOAP, les premiers sur le marché, qui ont eu plus d'influence sur les standards et sur les middlewares orientés services que les services RESTful qui sont une technologie en pleine croissance. Les détails de ces deux types de services sont présentés dans ce qui suit.

1.3.1.1 Services SOAP

Ce type de service repose sur le protocole SOAP (Simple Object Access Protocol) qui est un simple protocole de communication basé sur XML permettant l'échange d'informations via HTTP. Les services Web de type SOAP sont orientés opérations c'est-à-dire qu'ils exposent les fonctionnalités offertes par le service sous forme d'opérations.

L'interaction dans une application manipulant les services SOAP est à base de messages. SOAP définit un format de message standard pour la communication, décrivant comment les informations doivent être regroupées dans un document XML. Les messages SOAP échangés sont encapsulés dans des enveloppes SOAP. L'enveloppe SOAP est composée d'une en-tête et d'un corps. L'en-tête contient des informations facultatives sur les métadonnées (authentification), alors que le corps contient les informations utiles du message destiné à être traité par le destinataire. Par la suite, le client peut rechercher le service dans l'annuaire et récupérer la description WSDL pour invoquer le service Web. En raison des échanges lourds des messages SOAP handicapant les applications volumineuses, une alternative a vu le jour.

1.3.1.2 Services RESTful

Les services Web de type RESTful se présentent comme une alternative légère aux services de type SOAP. Ils sont conçus pour faciliter la consommation, la composition et la mise en place d'applications dans le contexte de l'architecture orientée services. Ce type de services s'appuie sur le style architectural REST (Representational State Transfer) présenté [Fielding et Taylor, 2000].

Ce modèle est largement adopté pour implémenter les services Web [Richardson *et al.*, 2013].

Il s'articule autour de la notion centrale de ressource, qui est une entité abstraite identifiable par un URI (Uniform Ressource Identifier) et manipulable par une interface uniforme permettant d'exposer les données et les fonctionnalités. Les services REST se basent sur les primitives standards de HTTP permettant aux clients de manipuler les ressources où ces méthodes (Post, Get, Put et Delete) sont mappées aux opérations CRUD respectives Créer, Lire, Mettre à jour et Supprimer. Les réponses de service contiennent la représentation de la ressource demandée dans le format JSON, XML ou dans d'autres formats similaires.

1.4 Problème d'intégration

Actuellement, une multitude de sites Web et d'applications sont disponibles pour fournir aux utilisateurs les informations nécessaires. Toutefois, exploiter ces services pour répondre à un besoin de type organisation de trajets devient une tâche complexe. En effet, les utilisateurs sont tenus à effectuer un ensemble d'étapes répétitives souvent désagréables afin d'accéder à l'information. Ils cherchent les sites et les applications appropriés et les explorent itérativement pour accéder à l'information en fonction de leurs besoins. Par exemple, Alice souhaitant planifier un séjour en Val de Loire se retrouve naviguer entre plusieurs pages Web à la recherche d'activités intéressantes. En plus, elle prévoit un voyage long ce qui a conduit à chercher un restaurant et un hébergement pour se reposer et se ressourcer. Vu que les résultats proposés sont fragmentés, Alice sera obligée d'itérer le processus d'exploration pour chaque type d'activité.

Donc, ces services fournissant des fonctionnalités disparates n'aident pas Alice à accomplir son trajet. Elle veut exprimer ses besoins d'une manière globale et avoir les réponses des différents services sur une même page Web. Elle cherche une solution qui intègre les données issues des services disponibles sur le web comme Google Maps, Yelp, Google Places et autres capables de répondre à son besoin facilement.

Dans ce contexte, la problématique fondamentale qui se pose est : ***Comment combiner les données provenant de différentes sources pour obtenir une vue d'ensemble ? et Comment interconnecter et orchestrer les différentes composantes du système dans une logique d'exécution intégrée ?***

Ce concept d'intégration s'articule selon trois axes principaux, nous distinguons alors l'intégration des données, l'intégration d'applications et l'intégration d'interfaces utilisateur.

1.4.1 Intégration de données

L'intégration des données se rapporte à l'extraction de données à partir de sources distribuées et de la combinaison des données hétérogènes pour fournir une vue unifiée.

Le besoin d'intégration des données découle en particulier de l'évolution des technologies Web, qui ont considérablement influencé la façon dont les systèmes d'information sont conçus aujourd'hui. Par exemple, dans le scénario de planification de séjour touristiques, le visiteur non connaisseur de la ville serait intéressé par les avis des autres visiteurs concernant les endroits à visiter. D'après cet exemple, ainsi que dans de nombreuses autres situations, il est évident qu'il est nécessaire de rapprocher des données différentes traitant les mêmes entités afin d'accroître la valeur de certaines par la combinaison avec des sources de données complémentaires. Vu que les données sont représentées par des sources hétérogènes, les méthodes d'accès, les formats et les schémas adoptés sont différents. Par conséquent, l'objectif d'un système d'intégration est de masquer la distribution et l'hétérogénéité des sources sous-jacentes et de fournir aux utilisateurs un point d'accès à une vue homogène de ces données fragmentées.

La question de l'hétérogénéité dans l'intégration peut être résolue avec la notion d'ontologie [Cruz et Xiao, 2005] qui est une spécification formelle et explicite d'une conceptualisation partagée d'un domaine d'intérêt [Gruber, 1993]. Il s'agit de donner un sens aux termes définissant la connaissance d'un domaine particulier selon un format compréhensible par la machine.

Étant compréhensibles par les machines, les ontologies permettent l'intégration des données, puisqu'elles fournissent un support pour résoudre automatiquement les différents problèmes d'hétérogénéité qui se produisent typiquement dans l'intégration de données. Par exemple, l'appariement des schémas, visant à rapprocher les éléments des différents schémas, peut être réalisée automatiquement en s'appuyant sur la description ontologique des schémas de la source de données. La médiation peut être réalisée par le mapping ontologique en exploitant les relations d'équivalence et de similarité identifiées. De même, les techniques de fusion et d'intégration peuvent être exploitées pour créer une ontologie agissant comme schéma global à partir de la réutilisation des différentes ontologies locales avec des concepts ou des définitions qui se chevauchent. Étant très ambitieuses, les approches basées sur l'ontologie pour l'intégration des données nécessitent un effort supplémentaire pour créer des annotations sémantiques et définir des représentations structurées des données par le biais de ce concept. Une nouvelle forme d'intégration de données dite intégration légère [Doan *et al.*, 2012] est caractérisée par les deux aspects suivants :

Le premier aspect concerne les stratégies de combinaison d'ensembles de données. Dans un contexte caractérisé par le dynamisme et l'innovation, les techniques semi-automatiques

d'intégration s'avèrent plus appropriées. En effet, un outil peut suggérer sur cette base des correspondances à l'utilisateur [Di Lorenzo *et al.*, 2009]. Il en ressort une forte implication des utilisateurs dans le processus de mapping. Ceci est à travers l'expression interactive de la fonctionnalité souhaitée et la sélection des alternatives les plus intéressantes parmi celles proposées automatiquement par les outils.

Le deuxième aspect se rapportant aux opérateurs de manipulation et d'intégration de données, les paradigmes de flux de données sont les mieux adaptés pour exprimer des opérations incrémentales permettant de restructurer le schéma des données importées. Les opérateurs de tri et de filtrage traitent les ensembles de données extraites, ceux de fusion et d'agrégation assurent la combinaison des données [Di Lorenzo *et al.*, 2009]. Les techniques telles que les visualisations efficaces des schémas sources et des ensembles de données correspondants, les paradigmes de programmation par l'exemple, les interactions par glisser-déposer pour la spécification des diagrammes de flux de données et des opérations se révèlent comme les ingrédients essentiels pour rendre le processus d'intégration des données plus naturel et accessible à tous les utilisateurs.

1.4.2 Intégration d'applications

En passant de l'intégration des sources de données hétérogènes à l'intégration de systèmes logiciels complets avec des fonctionnalités différentes, on parle d'intégration d'applications. Dans ce type d'intégration, la logique applicative du système composite est développée par la mise en communication et l'orchestration des applications composantes. Cette pratique d'intégration fonctionne bien lorsque les applications de composantes exposent leurs fonctions par le biais d'une interface de programmation. Le problème de l'intégration est qualifié de dynamique quand les composants du système sont combinés au fur et à mesure de leur exécution.

Historiquement, les systèmes intégrés qui fonctionnent sur de multiples plateformes logicielles et matérielles étaient connus sous le nom de systèmes distribués. Un système distribué se définit comme un ensemble d'applications indépendantes mais qui est présenté à ses utilisateurs comme un système cohérent [Tanenbaum et Van Steen, 2007].

Les systèmes distribués sont de nature hétérogènes. Cette hétérogénéité est la raison pour laquelle il y a le besoin d'un ingrédient essentiel pour le développement d'applications distribuées qui est le middleware. Le middleware est un logiciel qui médiatise l'hétérogénéité des éléments composants l'application distribuée. Il fournit une couche système qui masque cette hétérogénéité et génère une vue abstraite sur les capacités du système distribué. Cette vue abstraite est fournie aux applications distribuées s'exécutant sur le dessus dans le but de faciliter leur développement. Le middleware représente ainsi, d'une part, une infrastructure informatique sur laquelle des applications distribuées peuvent être développées et, d'autre

part, une abstraction de programmation qui facilite le développement.

1.4.3 Intégration d'interfaces utilisateur

L'intégration de l'interface utilisateur fait plutôt référence à un ensemble de nouvelles méthodes axées sur la définition de systèmes d'intégration spécifiquement adaptés aux besoins liés à l'interface utilisateur telles que la synchronisation des vues et la communication événementielle. L'intégration des interfaces utilisateur est liée naturellement au développement de l'interface utilisateur (IU) qui est l'une des parties les plus longues du développement, des tests et de la maintenance des applications. Ainsi, il est clair que la réutilisation des composants de l'interface utilisateur est tout aussi importante que la réutilisation de la logique applicative [Daniel *et al.*, 2007].

Le problème de l'intégration de l'interface graphique consiste à l'intégration des composants en combinant leurs faces avant de présentation au lieu de leur logique d'application ou de leurs schémas de données. Ici, la granularité des composants est celle des modules ou des applications autonomes ; l'objectif est de tirer parti des interfaces utilisateur individuelles des composants pour produire des applications composites plus riches. Comme l'intégration d'interfaces utilisateur vise à composer des applications au niveau de la couche présentation, la responsabilité de la gestion des données et de la logique métier est laissée à chaque composant. L'intégration d'interface utilisateur s'applique particulièrement lorsque l'intégration d'applications ou de données n'est tout simplement pas réalisable (par exemple, lorsque les applications n'exposent pas les APIs de niveau métier), ou lorsque le développement d'une nouvelle interface utilisateur est trop coûteux (par exemple, lorsque l'application du composant change fréquemment ou que son interface utilisateur est trop complexe) [Daniel *et al.*, 2007].

Nous présentons par la figure 1.2 un résumé de la problématique d'intégration qui vise à satisfaire le besoin de l'utilisateur par la composition de l'ensemble des systèmes et applications. Ce processus d'intégration peut se focaliser sur les données comme il est montré par la figure 1.2 (a) ou se concentrer sur la composition de la logique métier des applications (Figure 1.2 (b)). L'intégration touche aussi le niveau présentation (Figure 1.2 (c)) en fusionnant les interfaces graphiques exposées des systèmes composants.

1.5 Composition de services Web

La composition de services se rapporte à l'activité d'agrégation de services Web pour créer un nouveau service. Elle se différencie de l'intégration d'applications traditionnelles où les pièces logicielles sont étroitement couplées et physiquement combinées. La composi-

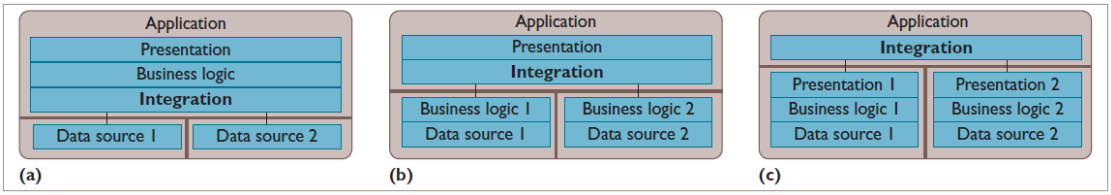


FIGURE 1.2 – Le problème d'intégration à ses trois niveaux

tion des services englobe les rôles et les fonctionnalités permettant de regrouper plusieurs services en un seul service compositepour développer de nouvelles applications plus riches. Ces services sont appelés composants qui à leur tour, peuvent être composés pour former des services composites. Ces types de services, à la fois élémentaires et composites, coexistent avec d'autres systèmes de la couche ressources pour mettre en œuvre la logique de composition décrite par le processus métier.

Donc, l'ingrédient de base de toute application composite sont *les composants (services Web)*. Ces derniers encapsulent la fonctionnalité, les données et/ou l'interface utilisateur, qui peut être réutilisée, et fournissent un ensemble d'opérations permettant d'interagir avec la fonctionnalité encapsulée et/ou l'interface utilisateur. De plus, comme les services ressemblent essentiellement à une application logicielle, il n'est pas surprenant que les couches de la pile services imitent de très près les "couches d'intégration" logicielles que nous avons identifiées avant et nous retrouvons les mêmes problèmes.

1.5.1 Représentation de service composite

Une représentation graphique permettant de décrire un processus métier (service composite) est le modèle de workflow, qui représente un certain nombre de tâches ordonnées. Chaque tâche dans un workflow peut représenter un service Web, et les flux dans le workflow définissent la façon dont les services Web sont coordonnés en décrivant les règles de conversation et les protocoles entre les services.

WS-BPEL (Web Services Business Process Execution Language)[Jordan *et al.*, 2007] est le langage standard pour la composition de services Web, un langage XML qui est spécifiquement adapté aux besoins de la composition de services Web SOAP. [Pautasso, 2009] proposent une extension de BPEL pour la composition des services Web RESTful. La Figure 1.3 illustre un service Web composite décrit par un workflow BPEL.

Selon une perspective de développement pour les utilisateurs finaux, une attention particulière doit être accordée à l'adoption de représentations adéquates de service. Les représentations techniques, par exemple, mettant en évidence les paramètres d'entrée et de sortie sont difficiles à maîtriser par l'utilisateur final. En effet, des études portant sur

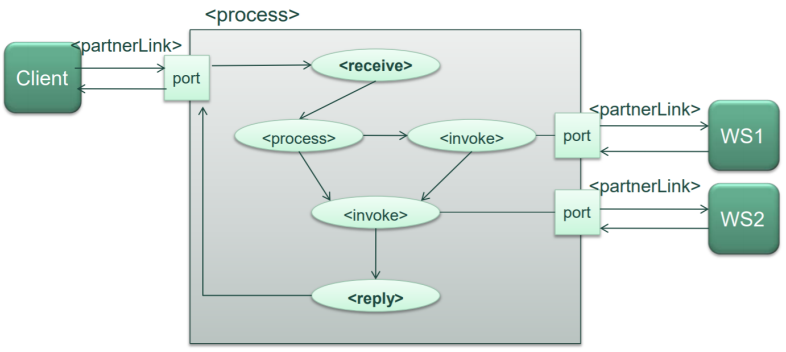


FIGURE 1.3 – Représentation d'un service composite en WS-BPEL

la façon dont les non programmeurs perçoivent les services et la composition des services [Namoun *et al.*, 2010], ont montré que les utilisateurs finaux, non-programmeurs, ne savent pas exactement ce qu'est un service et ne sont pas capables de maîtriser adéquatement les paramètres des services. Ainsi, la première question qui se pose dans ce cadre est de comprendre *quels paradigmes de programmation sont les mieux adaptés à la programmation par l'utilisateur final.*

Cela implique l'introduction de modèles de programmation qui peuvent masquer l'hétérogénéité technologique et n'exposer que les caractéristiques essentielles pour aider les utilisateurs à comprendre comment un composant peut être utilisé et intégré dans le mashup. Par conséquent, pour aider les utilisateurs à comprendre les caractéristiques fournies par les services, le rôle que chaque service peut avoir dans la composition, et la manière dont il peut être intégré avec d'autres services, il est important de proposer des représentations de services qui s'éloignent des détails techniques de programmation, tout en augmentant l'expressivité des rôles que ces éléments jouent dans les activités de composition selon une *représentation visuelle plus accessible aux utilisateurs finaux.*

1.5.2 Méthodes de composition

Deux principales méthodes de composition de services Web règnent dans la littérature [Paik *et al.*, 2017] : l'orchestration de services et la chorégraphie de services. Ces concepts ont été empruntés des domaines artistiques par l'architecture orientée services pour les adapter au contexte des services Web.

En effet, l'orchestration vient des orchestres où les musiciens qui jouent de leurs instruments suivent les instructions du chef d'orchestre qui dirige l'équipe. Les musiciens connaissent bien leur rôle ; ils savent comment jouer de leurs instruments, mais pas nécessairement en interaction directe les uns avec les autres. C'est le chef d'orchestre qui

coordonne et synchronise les interactions des différents composants pour exécuter un morceau de musique cohérent. Concernant la chorégraphie, elle est liée à la danse où sur une scène de danse il n'y a que des danseurs, néanmoins ils sont au courant de ce qu'ils doivent faire et comment communiquer avec les autres danseurs. Chaque danseur est chargé d'être synchronisé avec le reste du groupe. Nous décrivons en détails ces aspects de la composition appliquée à l'environnement des services Web dans la section suivante.

1.5.2.1 Orchestration de services

L'orchestration des services décrit comment les services Web interagissent les uns avec les autres et dans quel ordre ces interactions doivent s'exécuter [Paik *et al.*, 2017]. Elle adopte une perspective centralisée avec un élément principal qui se charge d'intégrer les composants participants dans la composition. Ces différents services invoqués par l'orchestrateur n'ont pas conscience qu'ils font partie d'un processus global de composition, seul l'élément central est au courant de l'objectif général à atteindre. Cependant, cette approche reste statique nécessitant une adaptation du schéma de composition à chaque changement de besoins.

L'orchestration peut être appliquée avec un processus de "médiation" ; dans ce cas, on l'appelle médiation de service au lieu d'orchestration de service. Le médiateur est un service Web placé au milieu des autres services, qui interagit avec les différents protocoles des services Web [Paik *et al.*, 2017].

1.5.2.2 Chorégraphie de services

La chorégraphie est associée à la coordination entre les composants où un accord décrivant les règles d'interaction est établi entre les multiples participant au processus. Ce contrat entre les partenaires coopérants constitue le résultat de la chorégraphie qui devrait être implémenté au niveau de chaque composant individuellement. Du point de vue chorégraphique, la composition est considérée dans une perspective globale où toutes les parties ont la même importance. Il y a une vue d'ensemble des protocoles appliqués et de l'échange de données [Paik *et al.*, 2017]. Cette approche est particulièrement utile dans les situations où plusieurs parties doivent collaborer, mais aucune d'entre elles ne veut prendre la responsabilité de diriger une orchestration centralisée [Lemos *et al.*, 2016].

L'orchestration et la chorégraphie pourraient être utilisées ensemble dans un processus global de collaboration. Mais ceci n'est que théorique, puisque les parties internes des processus ne sont généralement pas partagées [Paik *et al.*, 2017].

1.6 Mashups

La composition de services Web met l'accent uniquement sur la génération de processus métier. A l'opposition, le mashup va plus loin dans la mesure où il permet plus de fonctionnalités et peut composer des ressources hétérogènes telles que les services de données, les services d'interface utilisateur, etc.

A l'origine, cette notion de mashup a été empruntée de la musique pop, où elle désigne la création d'une composition musicale à partir de deux ou plusieurs parties de chansons déjà existantes pour livrer de nouvelles œuvres dérivées. Une chanson de mashup est générée par l'assemblage des parties vocales d'une première chanson avec la partie musicale d'une seconde. Dans l'environnement Web, le mashup consiste à agréger le contenu du Web, des ressources hétérogènes et à l'origine indépendantes, pour fournir de nouveaux résultats et ainsi créer de l'innovation par l'assemblage.

De la sorte, des opportunités particulières en combinant les capacités existantes du Web ont été identifiées afin d'offrir de nouvelles solutions et des inspirations qui à leur tour mènent à de nouvelles opportunités innovantes. Ces applications créées à l'aide de la technique de mashup sont appelées applications de mashups utilisant les services comme "ingrédients" qui peuvent être agrégés et appariés pour générer de nouvelles applications Web composites. Leur prolifération récente montre que la notion de mashup suscite un grand intérêt en tant qu'approche prometteuse pour l'information ad-hoc et l'intégration de services [Yu *et al.*, 2008, Di Lorenzo *et al.*, 2009, Daniel et Matera, 2014].

1.6.1 Concept et définitions

Plusieurs définitions ont été données pour caractériser le concept de mashups.

Un mashup est définit par [Markl *et al.*, 2008] comme une application web qui combine le contenu de deux applications ou plus pour créer une nouvelle application. Ces applications sont qualifiées d'application Web situationnelles construites à la volée pour résoudre un problème métier spécifique. Elles sont souvent développées par des utilisateurs finaux qui ne disposent pas de compétences en programmation logicielle.

[Yee, 2008] présentent une autre définition qualifiant les mashups comme des sites Web ou des applications Web qui " combinent de façon transparente du contenu provenant de plus d'une source dans une vue intégrée ". Cependant, la réduction de son objectif à l'intégration des données seulement, néglige l'une des innovations clés des mashups qui est l'intégration au niveau de la couche de présentation [Yu *et al.*, 2008].

Même si ces définitions ont caractérisé la notion de mashup, elles n'arrivaient pas à couvrir le réel potentiel que cette technique révèle. Nous adoptons la définition suivante

qui convient le mieux à notre propre compréhension et au but de cette thèse. Une application de mashup selon [Daniel et Matera, 2014] est "une application composite rapide et personnalisable, développée à partir de données réutilisables, d'une logique d'application et/ou d'interfaces utilisateur provenant généralement, mais pas obligatoirement, du Web".

Nous distinguons alors, les deux éléments principaux pour la création d'application de mashup : *les composants de mashup* qui représentent les pièces élémentaires réutilisables et *la logique de mashup* qui correspond à la logique d'intégration de ces composants.

Un composant de mashup est tout élément de données, logique d'application et/ou interface utilisateur qui peut être réutilisé, à savoir les flux RSS/Atom, les services Web RESTful et SOAP, les APIs Web ou les widgets d'interface utilisateur. Ils sont accessibles localement ou à distance. La logique de mashup est la logique interne de fonctionnement d'un mashup ; elle spécifie la sélection des composants, le flux de contrôle, le flux de données, les transformations de données et l'interface externe du mashup. Généralement, les interactions avec le mashup se font via l'interface utilisateur de l'application, qui sert d'interface externe vers ses utilisateurs. Toutefois, les mashups de données, fournissent en général une ressource Web, par exemple, une API, pour accéder au mashup.

Le développement d'application de mashup se traduit par la spécification de la logique de mashup. Particulièrement, les outils de mashup visent à fournir un développement de mashup assisté par ordinateur, en s'appuyant généralement sur des paradigmes tels que celui du développement visuel, le développement guidé par le modèle et ainsi que les environnements d'exécution de mashup.

1.6.2 Écosystème de mashup

La représentation graphique modélisant la relation entre les mashups et les APIs affilées est illustrée par la figure 1.4 exprimant un réseau d'APIs en interaction où les liens indiquent quelles APIs sont utilisées dans quels mashups. La distribution de l'utilisation des APIs dans les mashups suit une loi de puissance, ce qui implique que l'écosystème possède un petit nombre d'APIs formant un concentrateur ou keystone qui fournit la fonctionnalité de base pour plusieurs cas d'utilisation [Weiss *et al.*, 2013]. Ce concentrateur instaure le noyau de l'écosystème du mashup autour duquel sont ancrées les nouvelles fonctionnalités.

La figure 1.5 décrit la structure en couche de l'écosystème du mashup avec l'API Google Maps au centre, constituant le premier niveau. Particulièrement, Google Maps API ajoute de la valeur à la multitude d'autres APIs en proposant un moyen puissant pour combiner les données spatiales avec d'autres dans une application Web. Celle-ci est entourée d'un anneau d'API populaires telles que Facebook, Youtube mais moins centrales formant le niveau 2 et d'une constellation d'autres APIs à la périphérie pour dessiner le niveau 3. La

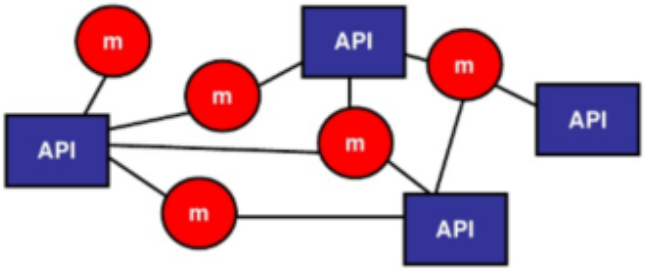


FIGURE 1.4 – Graphe de mashups

plupart des APIs qui fournissent des services de plate-forme se situent au niveau 2, alors que la plupart des fournisseurs de données se situent au niveau 3.

1.6.3 Langages de mashup

Nous présentons deux exemples représentatifs de langages de mashups textuels qui sont à la base des outils de mashups pilotés par le modèle, à savoir, le langage EMML (Enterprise Mashup Markup Language) et le langage OMDL (Open Mashup Description Language).

1.6.3.1 EMML

EMML [Alliance, 2013] est un langage déclaratif basé sur XML pour le développement de mashups de données avec capacités de visualisation des données. EMML est une initiative de l'Open Mashup Alliance EMML qui fournit une syntaxe uniforme pour combiner des composants de mashup hétérogènes, tels que les services RESTful, les services Web SOAP ou des flux RSS/Atom. Il prend en charge une variété de formats de données différents, tels que XML, JSON, objets Java, et similaires, ainsi qu'un ensemble d'opérateurs de transformation de données intégrés, tels que des opérateurs de filtre, tri, jointure, etc. Une logique de traitement complexe peut être intégrée sous forme de scripts. Bien que ce type EMML propose un développement simplifié du mashup, il est intuitif que l'écriture directe de XML n'est pas la meilleure façon de développer et que, étant donné les abstractions déjà intrinsèquement présentes dans le balisage EMML, une notation visuelle du développement peut être injectée dans le language EMML.

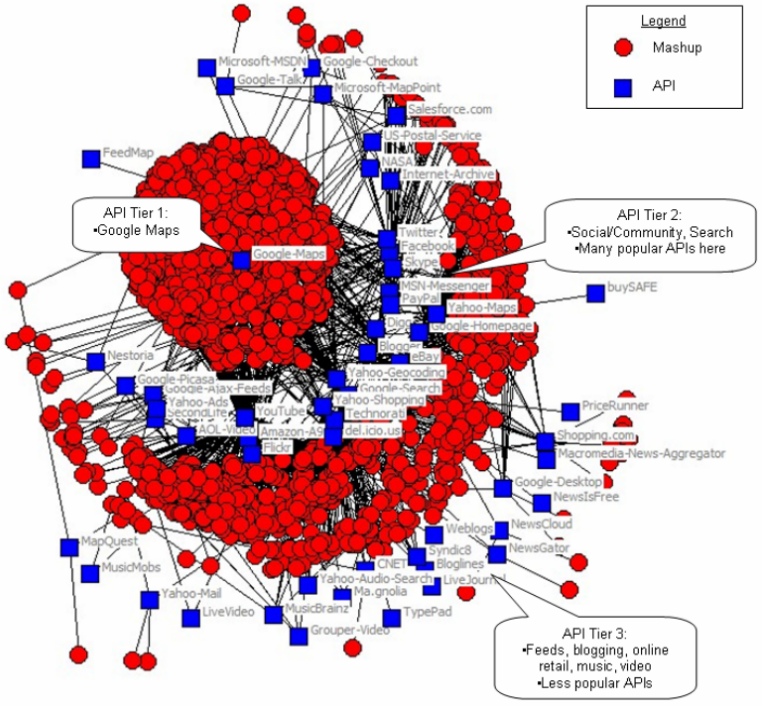


FIGURE 1.5 – Mashup Ecosystème

1.6.3.2 OMDL

OMDL [1] est une autre initiative de langage de mashup qui favorise la portabilité et l'interopérabilité entre les différentes plateformes de mashup. L'initiative a été lancée par le projet de recherche OMELETTE. OMLD s'adresse aux mashups de type interface utilisateur basés sur des widgets. Un mashup OMDL décrit donc un espace de travail de widget et la disposition des widgets dans l'espace de travail en produisant du code XML.

Comme pour EMML, il est clair qu'écrire du code XML n'est pas très efficace du point de vue de l'utilisateur final et qu'un outil de développement graphique peut améliorer la productivité. C'est ce que le Consortium OMELETTE a réalisé avec la mise en place d'Apache Rave qui, étant compatible avec OMDL, fournit un environnement de modélisation "live", dans lequel l'utilisateur place les widgets dans l'espace de travail et Rave les rend immédiatement. Autrement, au lieu d'avoir des constructions de modélisation graphique représentant des widgets de l'interface utilisateur, Rave affiche directement le widget réel.

EMML et OMDL sont des exemples de langages de mashups qui se basent respectivement sur les flux de données et les événements et qui peuvent être réutilisé dans le cadre

1. http ://omdl.org

d'outil de mashup de données ou d'interface utilisateur. Néanmoins, nous avons remarqué que dans de nombreux cas d'outils un langage de mashup avec une notation graphique spécifique au domaine a été développé puisque les langages génériques tels que EMML et OMDL peuvent ne pas fournir les abstractions et les capacités spécifiques. Cette dimension d'adaptation au domaine sera détaillée dans les section suivantes.

En effet, malgré leur grande puissance expressive, ces paradigmes basés sur les scripts ont apporté avec eux toutes les barrières typiques du développement orienté utilisateur : les utilisateurs ont été contraints d'apprendre et d'adopter une logique de programmation proche de la sémantique de la technologie sous-jacente et non de celle du domaine utilisateur. *Ces langages génériques s'adressent, en effet, aux utilisateurs relativement inexpérimentés ce qui a conduit à abandonner cette alternative au profit des abstractions de programmation suffisamment simples pour être compréhensibles par les experts du domaine et en même temps assez complexes pour mettre en œuvre une logique d'application Web.*

1.6.4 Classes de mashup

Conformément aux APIs Web, programmableWeb constitue un référentiel qui liste les mashups en spécifiant les APIs utilisées dans tel mashup. Dans programmableWeb, les mashup sont décrits par des tags qui constituent une meilleure façon de rechercher des mashups dans un domaine donné. Toutefois, nous pensons que le résultat de la classification des mashups en cartographie, recherche, social, etc n'aide pas beaucoup à lui seul à clarifier l'écosystème du mashups pour comprendre à quoi ressemblent et comment fonctionnent les composants du mashup. Les classes comme les mashups Web, les mashups mobiles, les mashups de données, etc. sont plus sémantiques de ce point de vue. Bien qu'il semble y avoir une prolifération arbitraire de combinaisons des préfixe-mashup, sans aucune liaison évidente entre elles, ces types de mashups ne sont pas mutuellement exclusifs où nous pouvons retrouver des mashups de données Web. Par conséquent, l'écosystème de mashup peut se présenter sous forme de cube avec trois perspectives de décomposition [Daniel et Matera, 2014] comme il est illustré par la figure 1.6.

a. **Perspective composition**

   Avec un focus placé sur les composants de mashup, cet axe s'intéresse à la manière dont les composants sont agrégés dans une nouvelle application. Il s'appuie sur le principe de la séparation traditionnelle d'application découpant une application en trois couches, une couche de données, une couche logique d'application et une couche de présentation. Cette séparation n'a pas seulement influencé la façon dont les applications sont structurées mais a également favorisé la croissance d'APIs et de composants afin de faciliter l'interopérabilité et l'intégration des couches. La perspective de composition regroupe donc les mashups en mashups de données, mashups logiques, mashups d'interface utili-

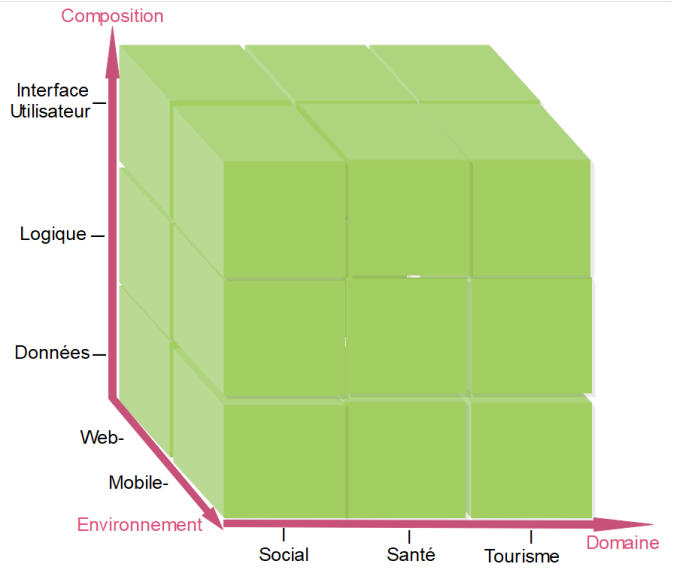


FIGURE 1.6 – Les classes du mashup

sateur et mashups hybrides combinant les autres types.

b. **Perspective domaine**

Cette perspective met l'accent sur l'objectif d'un mashup, c'est-à-dire, la fonctionnalité qu'il offre. Vu que les tags employés par les développeurs décrivent bien les domaines d'application du mashup, ils s'inscrivent alors dans cette perspective. Ainsi, les domaines se rattachent aux cadres d'application des mashups en l'occurrence le domaine social, cartographique, économique, touristique, etc.

c. **Perspective environnement**

L'axe environnement s'intéresse au contexte de d'exécution du mashup. Plus précisément, nous identifions les deux types : les mashups Web et les mashups mobile. Les mashups Web sont typiquement développés en utilisant les standards du Web et les mashups mobiles se basent sur les technologies mobiles natives ainsi ceux du Web ce qui donne lieu à la classe hybride mashup Web-Mobile. Nous ne considérons pas que ce type est une classe à part entière mais une sous-catégorie des mashups mobile

Comme l'indique la structure cubique, un mashup donné peut maintenant être classé, par exemple, comme un mashup de données mobiles pour le domaine médical ou un mashup de l'interface utilisateur Web de cartographie, etc. La classification peut inclure une nouvelle perspective qui est la dimension utilisateur, par exemple les mashups pour les développeurs, ou ceux destinés pour les utilisateurs finaux. Cet axe est considéré partiellement couvert par l'orientation domaine. C'est pour cela que nous avons limité la classification des mashups à une représentation cubique qui reste gérable.

1.6.5 Architecture de mashup

Nous retrouvons deux éléments principaux pour la création d'application de mashup : *les composants de mashup* qui représentent les pièces élémentaires réutilisables et *la logique de mashup* qui correspond à la logique d'intégration de ces composants.

Un composant de mashup est tout élément de données, logique d'application et/ou interface utilisateur qui peut être réutilisé, à savoir les flux RSS/Atom, les services Web RESTful et SOAP, les APIs Web ou les widgets d'interface utilisateur. Ils sont accessibles localement ou à distance.

La logique de mashup est la logique interne de fonctionnement d'un mashup ; elle spécifie la sélection des composants, le flux de contrôle, le flux de données, les transformations de données et l'interface externe du mashup. Généralement, les interactions avec le mashup se font via l'interface utilisateur de l'application, qui sert d'interface externe vers ses utilisateurs. Toutefois, les mashups de données, fournissent en général une ressource Web, par exemple, une API, pour accéder au mashup.

Le schéma conceptuel du modèle de base de mashup est décrit par la figure 1.7 exprimant les ingrédients principaux d'une application de mashup : les composants de mashups qui peuvent jouer le rôle de sources de données brute, servir comme composants applicatifs ou exposé une interface utilisateur et la logique de mashup, le second élément de base, inclut les mécanismes d'invocation, d'intégration et de présentation.

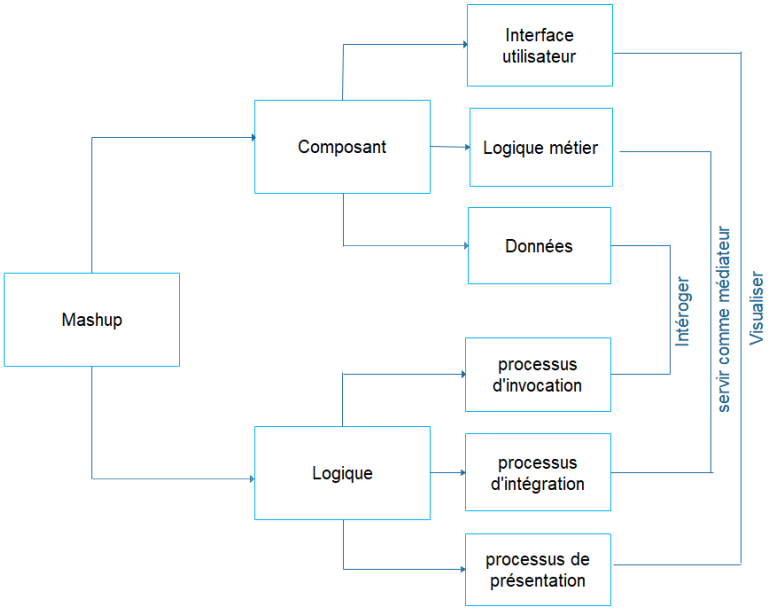


FIGURE 1.7 – Schéma conceptuel du modèle de Mashup

1.6.5.1 Composants de mashup

Les trois principaux composants d'une application mashup sont les composants de données, les composants encapsulant la logique applicative et ceux composants l'interface utilisateur.

a. **Composants de données** permettent d'accéder aux données via des interfaces appropriées. En général, les sources de données comprennent les flux RSS ou Atom qui sont des sources statiques, ainsi que, les services Web SOAP et REST interrogeables dynamiquement à partir des points d'entrée des services qui sont de type services de données.
   Avec l'avènement récent des données ouvertes promouvant l'idée que les données devraient être librement disponibles et réutilisables, la disponibilité des composants de données en ligne augmente à la fois en termes de quantité et de qualité. Ceci constitue une tendance dont les mashups vont fortement bénéficier.

b. **Composants logiques** encapsulent la logique et les fonctionnalités de l'application de mashup. Par exemple, un composant logique peut offrir la fonctionnalité de réservation d'hôtel. D'un point de vue conceptuel, les composants logiques et les composants de données partagent des points en communs. Ces derniers, peuvent être considérés comme des exemples particuliers de composants logiques avec des fonctionnalités limitées. Par exemple, les composants de données sont généralement des ressources passives en lecture seule. Nous faisons références aux services de données qui exposent uniquement les opérations de types GET dans le cas des services RESTful.

c. **Composants de l'interface utilisateur** exposent leur propre interface utilisateur et, éventuellement, la logique d'application et des données complètes derrière. L'API Google Maps constitue bien un représentant de cette classe de composants.
   A la différence des composants logiques et des composants de données, qui disposent d'un ensemble de technologies relativement stables et largement acceptées, les composants d'interface utilisateur ne bénéficient pas du même avantage. En partie, ce manque de standard pour les composants d'interfaces utilisateur est dû à l'état d'enfance de cette pratique de réutilisation et d'intégration d'éléments d'interfaces utilisateur dans les applications Web tierces de type mashups. En effet, l'intégration d'interface utilisateur est essentiellement née avec les mashups. Néanmoins, des modèles de composants réutilisables ont émergé et prennent de plus en plus d'importance.

1.6.5.2 Logique de mashup

Une fois que les composants d'intérêt sont identifiés, leur mise en communication suivant une stratégie de composition constitue la seconde étape dans le développement d'applica-

tion de mashup. Tout comme dans la composition des services Web, il ne suffit pas de pouvoir invoquer les services ; il est important de les orchestrer en alimentant les sorties d'un service par les entrées d'un ou d'autres services, construisant ainsi le service composite permettant de répondre aux besoins. Egalement, les mashups orientés vers les utilisateurs finaux doivent assurer, par exemple, que la carte est centrée sur une adresse spécifique après la sélection d'un lieu d'intérêt de la liste. C'est cette synchronisation qui offre de la valeur ajoutée à l'utilisateur en réduisant l'effort cognitif que l'utilisateur devrait dépenser autrement pour récupérer le même résultat en invoquant les services séparément.

1.6.6 Mashup hybride

L'enjeu des mashups hybrides est d'exploiter la puissance des données, de la logique et des mashups d'interfaces utilisateur au sein d'un seul type de mashup. Nous définissons un mashup hybride comme un mashup qui intègre des composants de mashup à plusieurs couches au niveau la pile d'applications, cela veut dire au niveau des données, de la logique applicative et/ou de la couche de présentation. Les mashups hybrides peuvent se présenter sous forme d'applications Web interactives.

En fait, à l'exception des mashups de données, en pratique, il est difficile de trouver des mashups qui sont purement logiques ou des mashups d'interfaces utilisateur. Lors du développement d'une application de mashup qui intègre des composants hétérogènes, tels que, des widgets de visualisation et des services Web, cela nécessite éventuellement une logique d'intégration à différentes couches, par exemple, une synchronisation de l'interface utilisateur au niveau de la couche de présentation et une orchestration de services avec des transformations de données au niveau de la couche logique. En fonction des composants utilisés et de la manière dont ils sont intégrés, le modèle spécifique d'un mashup hybride donné n'utilisera qu'une sélection des caractéristiques.

1.6.7 Mashup mobile

Les mashups deviennent de plus en plus personnels et se sont généralisés pour fonctionner sur des appareils mobiles. Nous parlons alors de mashups mobiles, c'est-à-dire de mashups qui, en plus d'intégrer des API Web, peuvent utiliser des composants de l'appareil mobile. La particularité des mashup mobile est que les APIs du dispositif offrent aux utilisateurs une expérience enrichie où les capacités des dispositifs élargissent les fonctions produites par le mashup avec des fonctionnalités, par exemple, l'enregistrement d'un événement directement dans l'agenda personnel, ou l'accès au navigateur GPS pour se guider dans la recherche des adresses.

1.6.8 Mashup spécifique au domaine

La portée d'une plateforme de mashup peut aller des tâches de développement génériques à des tâches très spécifiques à un domaine particulier [Sutcliffe *et al.*, 2003] . Jusqu'à présent, la plupart des plateformes de mashup proposées visent à accroître leur applicabilité dans différents domaines. Ils sont donc génériques, indépendantes du domaine. Bien que cela semble être un avantage évident, dans le contexte d'une perspective orientée utilisateur, c'est plus un obstacle qu'un privilège [Casati, 2011].

En fait, les outils génériques ont tendance à être d'avantage axés sur la technologie et non adaptés aux spécificités des domaines d'application individuels. Cependant, les utilisateurs finaux ne sont pas des techniciens et ne savent pas comment adapter des technologies génériques à leur domaine. L'absence de personnalisation de la fonctionnalité de la plateforme et du langage de composition devient un obstacle à l'adoption de telles plateformes. Un certain nombre d'avantages des langages spécifiques au domaine sont résumés dans les travaux de [Chudnovskyy *et al.*, 2012].

Les langages spécifiques à un domaine sont importants pour fournir à l'utilisateur final des concepts, une terminologie et des métaphores familiers permettant d'améliorer la productivité par la simplicité. Ils sont particulièrement utiles car ils sont adaptés aux exigences du domaine, tant en termes de sémantique et de puissance expressive, n'exigent pas aux utilisateurs finaux à monter en compétences, qu'en termes de notation et de syntaxe en fournissant des abstractions appropriées issues du domaine. Ils parlent ainsi le langage des utilisateurs.

Plusieurs exemples de langages spécifiques au domaine existent : langages de configuration pour les systèmes domotiques, langages d'analyse pour les applications financières, langages de modélisation et d'autres. Ces langages peuvent être déclaratifs ou impératifs. La notation adoptée par le langage peut être graphique ou textuelle. La notation graphique, connus sous le nom de langages de modélisation spécifiques au domaine, impliquent des modèles visuels pour les résultats et des éléments graphiques tels que les blocs. La notation textuelle s'appuie sur le langage XML. Toutefois, les diverses expériences en conception d'application spécifique au domaine n'affirment pas des préférences particulières des développeurs pour l'un ou l'autre type de langage. Cependant, selon [Daniel et Matera, 2014], les langages typiquement orientés vers les utilisateurs finaux ont tendance à être plus visuels et déclaratifs, tandis que ceux destinés aux développeurs sont souvent textuels et impératifs.

[Imran *et al.*, 2012] ont proposé les constituants clés pour le développement de mashup spécifique au domaine qui se résument comme suit : A partir de l'analyse du domaine à supporter, un modèle de concept de domaine qui exprime les données et les relations

du domaine ainsi qu'un modèle de processus sont produits. Ensuite, un méta-modèle est crée permettant au développeur de configurer facilement les fonctionnalités de mashup en se basant sur le modèle du mashup unifié (UMM). Ce dernier inclut aussi une syntaxe spécifique au domaine offrant un ensemble de composants du domaine. Lors de la phase de la mise en œuvre, le méta-modèle génère automatiquement la plate-forme de mashup spécifique au domaine, composée d'un langage de mashup spécifique au domaine et d'un environnement d'exécution et de conception adapté. La figure 1.8 illustre ces trois grandes phases du développement spécifique au domaine commençant par l'analyse, la conception et l'implémentation.

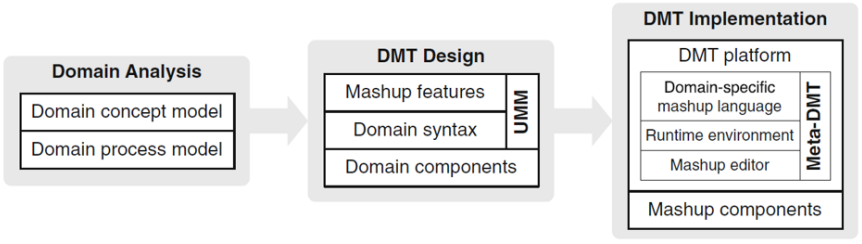


FIGURE 1.8 – Methodologie de development de mashup spécifique au domaine proposée par [Soi *et al.*,]

1.6.9 Mashup appliqué au tourisme

Nous décrivons les mashups appliqués au tourisme puisque ce domaine est le carde d'application de cette thèse. L'objectif global de ce type de mashup est de fournir de l'information à jour, pertinentes qui répond aux besoins du visiteur et tient en compte de son contexte, afin de l'aider dans la planification de son voyage. La disponibilité de la donnée touristique n'est plus la problématique. Elle est partagée en ligne au moyen des carnets de voyage ; des blogues ; des wikis (Wikitravel) ; des référentiels de photos en ligne (Flickr) ; des plateformes géo-sociales (Foursquare) ; des sites de recommandation (Tripadvisor). Bien que chacune de ces ressources soient développées indépendamment des autres, le moyen simple de les connecter facilement est l'orchestration de services afin de satisfaire un besoin particulier d'information.

Un mashup de voyage peut se définir comme la combinaison de l'ensemble des sources en fonction des besoins du visiteur afin de construire des informations pertinentes et les présenter à l'utilisateur final comme des informations directement utilisables. Ce type de mashups fourni aux utilisateurs un moyen pour interagir avec leur environnement, en combinant les informations provenant de différents types de capteurs, à la fois physiques (

GPS) et basés sur le Web (ressources en ligne). Cette intégration fusionne les dimensions spatiales, sociales et temporelles pour fournir des services capables de couvrir un besoin spécifique des visiteurs des nouvelles villes dans un monde en évolution dynamique. Les mashups appliqués au tourisme peuvent être répertoriés en deux groupes en fonction de la cible qui utilisera l'application [Cano *et al.*, 2013] :

— Mashup pour l'utilisateur qui exploite le service afin d'avoir une idée éclairée sur son voyage. Une facette importante de cette catégorie est la personnalisation considérant les exigences individuelles de chaque consommateur. Celle-ci sera détaillée dans la section suivante.

— Mashup pour les développeurs où le service fourni sera employé soit comme un bloc de construction pour concevoir des services plus complexes, soit comme un service d'assemblage qui permet aux développeurs de combiner d'autres applications.

Une architecture typique du mashup spécifique au domaine du tourisme, décrite par la Figure 1.9, est proposée par [Cano *et al.*, 2013]. Il s'agit d'une structure en trois couches qui se compose : *la couche sources d'information* ; *la couche de Mashup* ; et *la couche de présentation.*

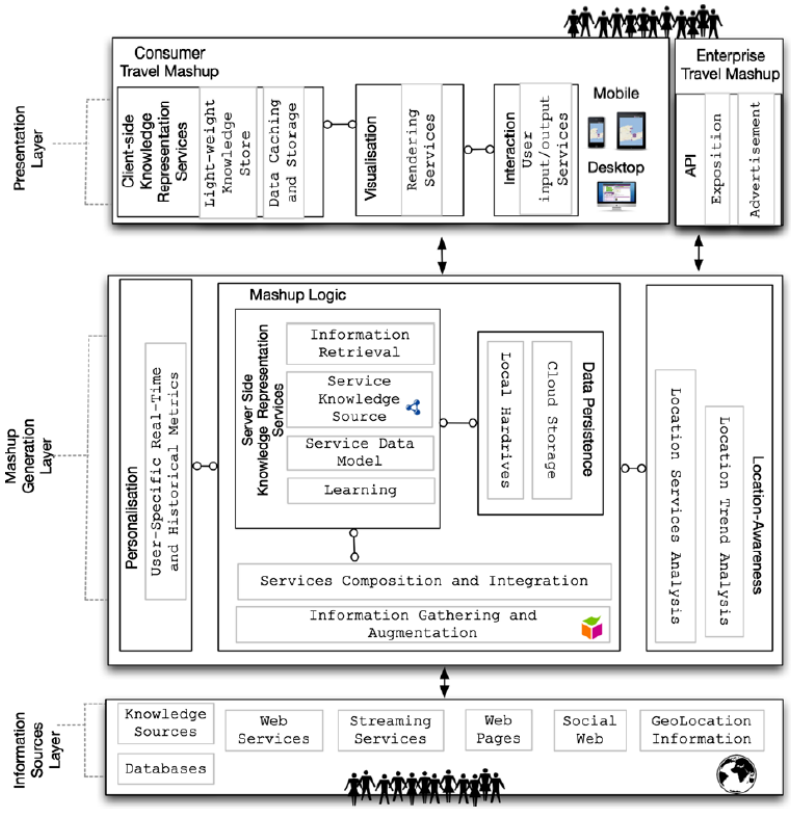


FIGURE 1.9 – Architecture du mashup spécifique au domaine touristique

1.7 Problème de personnalisation

L'évolution des nouvelles technologies transforment les individus en utilisateurs non seulement plus connectés, mais aussi hyper-connectés à travers de multiples appareils qui peuvent aller des ordinateurs portables, aux téléphones mobiles et aux appareils intelligents que les gens utilisent quotidiennement : des vêtements comme Fitbit, l'Apple Watch, la navigation et le divertissement automobile, et même des appareils électroménagers.

La montée en puissance de ces objets intelligents conduit à des nouveaux besoins de faire face à la surcharge d'information générée par ces derniers et offre de grandes opportunités pour de nouvelles formes de personnalisation d'objets et de services. Fournir une expérience personnalisée est devenue une fonctionnalité clé pour les nouveaux systèmes. Il s'agit du processus de collecte d'informations sur l'utilisateur par interaction avec ce dernier, qui sont ensuite exploitées pour offrir une assistance et des services appropriés, adaptés aux besoins des utilisateurs. La personnalisation est motivée par la reconnaissance des besoins de l'utilisateur afin d'établir une relation satisfaisante avec lui.

Nous décrivons dans un premier temps les techniques de personnalisation puis les classes qui traitent le niveau de granularité du sujet à personnaliser.

1.7.1 Techniques de personnalisation

Il existe trois techniques de personnalisation : *la contextualisation*, *la configuration* et *la recommandation.*

1.7.1.1 Contextualisation

La contextualisation est considérée comme la première étape à réaliser dans une solution de personnalisation. Cette connaissance du contexte est particulièrement importante parce qu'elle améliore l'adaptation des services et des applications aux changements de situations. Donc, pour fournir un service adéquat aux utilisateurs, les applications doivent tenir compte du contexte donnant lieu aux applications sensibles au contexte.

Ces applications peuvent adapter leur comportement en fonction du contexte découvert représentant une gestion active du contexte ou présenter le contexte à l'utilisateur à la volée et/ou le stocker pour que l'utilisateur puisse le récupérer plus tard [Silva *et al.*, 2018] permettant une gestion passive du contexte. [Li *et al.*, 2015b, Perera *et al.*, 2013] tentent de définir le contexte, en fonction des nécessités et de l'environnement étudié. La définition la plus adoptée est celle de [Dey et Abowd, 2000] décrivant le contexte en tant que " toute information qui peut être utilisée pour caractériser la situation des entités (une personne, un lieu ou un objet) est considérée comme pertinente pour l'interaction entre un utilisateur

et une application, y compris l'utilisateur et l'application. Le contexte est généralement l'emplacement, l'identité et l'état des personnes, des groupes et des objets informatiques et physiques ".

Il existe trois paradigmes différents pour intégrer l'information contextuelle dans le processus de recommandation : *la réduction ou le pré-filtrage*, *le post-filtrage* et *la modélisation du contexte* [Perera *et al.*, 2013]. Dans les méthodes fondées sur la réduction (pré-filtrage), seules les informations respectent le contexte d'utilisation actuel sont utilisées pour calculer les recommandations. Dans le filtrage contextuel ou le post-filtrage, l'algorithme de recommandation ignore l'information de contexte dans un premier temps. Puis il ajuste pour n'inclure que les recommandations pertinentes dans le contexte cible. Dans la modélisation contextuelle, la plus sophistiquée des trois approches et celle que nous avons adoptée, les données contextuelles sont explicitement utilisées dans le modèle de personnalisation.

Ainsi, le cycle de vie de la gestion du contexte est délimitée par quatre événements significatifs tel qu'il est illustré par la figure 1.10. Le processus commence par l'acquisition de différents types de contextes suivis par la formalisation et l'inférence, et se termine par la visualisation.

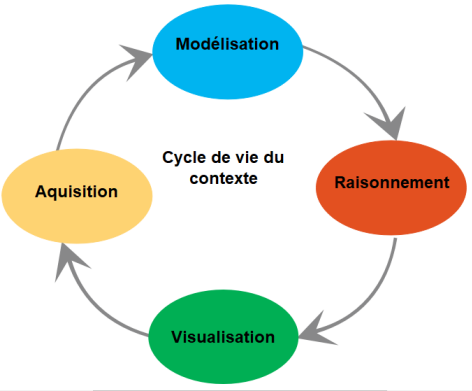


FIGURE 1.10 – Cycle de vie de l'information contextuelle

a. **Phase d'acquisition de l'information contextuelle**

Le but de l'acquisition est de rassembler un maximum de données, de sorte que les applications intelligentes puissent être maximisées grâce à une information de contexte plus riche. Plusieurs classifications de l'information contextuelle ont été proposées par [Li *et al.*, 2015b, Alegre *et al.*, 2016, Urbieta *et al.*, 2017]. Nous nous concentrons sur le concept 5W1H qui propose une catégorisation intuitive selon six questions : Qui ? se réfère à une entité qui traite les rôles, à savoir une personne, une organisation ou un système impliqué dans un contexte, Quand ? est relatif au temps, ou plus exactement

la durée d'un contexte., Où ? c'est la localisation associée, Quoi ? correspond au statut du contexte, Pourquoi ? fait référence à la raison et Comment l'action à réaliser.

b. **Phase de modélisation de l'information contextuelle**

Suite à la phase d'acquisition, une grande quantité de données contextuelles structurées en plusieurs formats est obtenue. Pour les utiliser, les concepts du monde réel sont traduits sous une forme lisible et traitable par la machine, toutes les données doivent être converties dans un format unifié de sorte que le contexte puisse être compris et partagé.Une brève introduction aux techniques les plus couramment utilisées pour la modélisation du contexte [Li *et al.*, 2015b, Perera *et al.*, 2013, Alegre *et al.*, 2016] est présentée dans la suite.

Modélisation basée sur la logique formelle : le contexte est défini comme des faits, des expressions et des règles. Les règles sont principalement utilisées pour exprimer les politiques, les contraintes et les préférences. Il offre donc un formalisme à haut niveau avec une richesse expressive. Diverses applications ont adopté ce modèle.

Modélisation centrée sur l'utilisateur : explore la façon dont l'information contextuelle est perçue par les utilisateurs. Il s'agit simplement d'un compromis entre la complexité de l'expression et la facilité d'utilisation.Les perspectives qui caractérisent les différentes situations contextuelles dans lesquelles les utilisateurs peuvent agir dans un scénario d'application donné sont modélisées à l'aide du *Modèle de Dimensions Contextuelles* [Bolchini *et al.*, 2013, Daniel *et al.*, 2018], qui organise selon une structure hiérarchique les différentes perspectives à travers lesquelles l'utilisateur perçoit le domaine d'application. Elle est composée (i) d'un noeud racine qui définit le contexte à travers (ii) un ensemble de dimensions contextuelles, représentées par des noeud fils de la racine, décrivant des classes contextuelles. Par exemple, les préférences, le temps, l'environnement, etc. A chaque dimension est associée une ou plusieurs (iii) valeur(s) dimensionnelle(s), par exemple culture, matin, transport public, etc. Ces données peuvent être soit des valeurs personnalisées fournies par l'utilisateur lors de l'exécution, soit des données acquises par des capteurs (par exemple, les coordonnées GPS actuelles d'un dispositif donné).

Ce modèle hiérarchique a été initialement introduit pour adapter, au moment de la conception, les parties contextuelles d'une base de données globale, afin de permettre aux utilisateurs d'accéder à des ensembles de données volumineux en fonction du contexte d'exécution.

c. **Phase de raisonnement**

Sur la base des données modélisées, différents types de conclusions peuvent être déduites, lorsque ces données peuvent être considérées comme des preuves à l'appui de la conclusion [Alegre *et al.*, 2016]. De cette façon, de nouvelles connaissances et une

nouvelle compréhension sont obtenues, en fonction du contexte disponible. Il existe principalement trois méthodes de raisonnement : le raisonnement par cas, le raisonnement fondé sur les règles et celui fondé sur l'ontologie.

d. **Phase de visualisation**

La visualisation de l'information contextuelle offre une opportunité aux utilisateurs de manipuler l'information de manière interactive et instinctive. Dans les domaines des maisons intelligentes et du développement mobile, des services et des applications tels que IFTTT (If This Then That) [Ur *et al.*, 2016] permettent aux utilisateurs de créer des chaînes de déclarations conditionnelles simples, déclenchées en fonction des changements apportés aux dispositifs ou services Web à savoir Facebook, Gmail, Calendrier.

1.7.1.2 Configuration

Dans la configuration, aussi appelée la customisation, les utilisateurs jouent un rôle actif dans le processus de personnalisation. Il s'agit du processus à travers lequel les utilisateurs adaptent les offres à leurs propres préférences [Bleier *et al.*, 2018]. Typiquement, l'exemple de prototype de Dell offrant à ses clients la possibilité de choisir eux-mêmes les composants de leurs ordinateurs prouvant sa capacité à construire des ordinateurs en fonction des besoins spécifiques des clients. Ceci offre notamment des possibilités de personnalisation presque illimitées. Le niveau de contrôle que les clients peuvent exercer sur le processus de customisation et le degré d'autonomie qui leur est accordé diffèrent d'une solution à une autre [Bleier *et al.*, 2018].

Les systèmes de configuration, également connus sous le nom de boîtes à outils pour la personnalisation, sont chargés de guider l'utilisateur dans le processus de personnalisation. Différentes variantes sont évaluées, représentées et visualisées suivant un processus d'apprentissage par la pratique. L'avantage de tel système d'interaction est cependant défini non seulement par ses capacités technologiques, mais aussi par son intégration de l'utilisateur dans l'ensemble du processus afin de mieux comprendre ses besoins par l'apprentissage par ce qui fournira de l'expérience et de la satisfaction [Franke et Piller, 2003]. Ainsi, l'idée de cette méthode de personnalisation dépend d'une intégration profonde et efficace des utilisateurs dans la création de valeur à travers un processus de navigation dans les choix proposés.

Le principal défi de la configuration réside dans la capacité d'aider les clients à identifier leurs propres solutions tout en minimisant la complexité et en facilitant la navigation dans les choix, c'est-à-dire en améliorant l'expérience des besoins des utilisateurs et l'interaction dans un processus de personnalisation.

La technique de configuration est appliquée dans divers domaines tels que l'industrie [Hvam, 2006], les services Web [Ten Teije *et al.*, 2004, Mesmoudi *et al.*, 2011] et l'Internet des objets (IoT) [Dalipi *et al.*, 2016, Felfernig *et al.*, 2016] . Pour un aperçu des applications et une discussion sur l'historique des systèmes de configuration, nous nous référons à [Felfernig *et al.*, 2014]. L'environnement de configuration et ces constituants sont décrits par la figure 1.11.

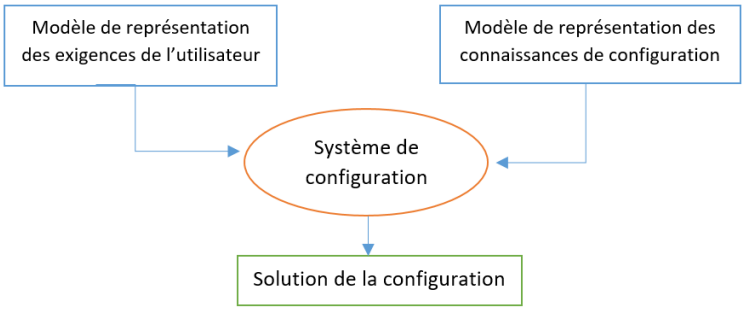


FIGURE 1.11 – Environnement de configuration

L'architecture typique d'un environnement de configuration comprend un modèle utilisateur qui représente les différentes exigences en terme de contraintes et préférences utilisateur. Le second composant décrit l'ensemble des connaissances de configuration englobant la description des composants et leurs fonctionnalités et les connections possibles entre-eux. Le système de configuration ou le configurateur détermine ainsi sur la base de ces modèles une solution de configuration qui tient en compte de l'ensemble des exigences et des contraintes définies. Les approches de modélisation des connaissances de configuration sont de deux types détaillés dans les sous-sections suivantes.

Configuration basée sur la modélisation des fonctionnalités

La modélisation des fonctionnalités ou des caractéristiques du système à configurer est un élément clé pour la gestion de la variabilité du système. Il s'agit de sélectionner des caractéristiques à partir d'un modèle de fonctionnalités basé sur les exigences spécifiques du domaine et les objectifs et les contraintes de l'utilisateur. Les composants de cette approche sont représentés dans la figure 1.12.

Les fonctionnalités représentent les propriétés du système qui servent à saisir les points communs ou à faire de la discrimination entre les systèmes. Ces fonctionnalités ou caractéristiques sont organisées en diagramme. Un diagramme de caractéristiques est une arborescence dont la racine représente un concept et ses nœuds sont les caractéristiques de celui-ci. Les relations entre les fonctionnalités peuvent être de type obligatoire, op-

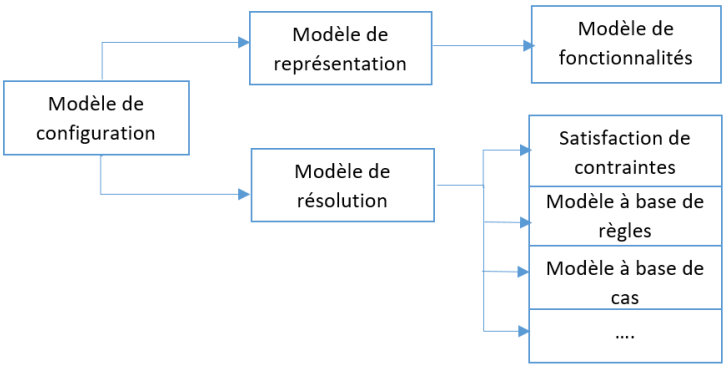


FIGURE 1.12 – Schéma de configuration à base de fonctionnalités

tionnelle, alternative ou une relation de cardinalité[n..m] qui spécifie qu'au moins n et au plus m caractéristiques doivent être présentes dans le système configuré. Dans la gestion de variabilité, il est important de définir les contraintes du système qui peuvent être des contraintes booléennes ou complexes. Le premier type exprime des relations d'inclusion ou d'exclusion alors que le second permet de définir des relations complexes. L'avantage majeur de cette approche est la séparation entre la représentation et la résolution. Par conséquent, le deuxième sous module permet de définir les techniques d'analyse et de résolution du problème de configuration à savoir la technique de satisfaction de contrainte, le modèle à base de règle, le raisonnement à base de cas ou autre.

La configuration à base du modèle de caractéristiques est également un processus de sélection du meilleur ensemble de caractéristiques pour le produit à configurer en fonction des objectifs spécifiques, des exigences et des limitations des parties prenantes, ainsi que les contraintes d'intégrité du modèle des fonctionnalités. Le processus de configuration peut être vue comme la sélection des fonctionnalités désirées et l'élimination de celles indésirables ou non pertinentes pour le produit configuré [Czarnecki *et al.*, 2004].

La relation entre le modèle de fonctionnalités et le modèle de configuration est comparable à celle entre une classe et son instance dans la programmation orientée objet. Le processus de dérivation d'une configuration à partir du modèle de caractéristiques est également appelé configuration [Czarnecki *et al.*, 2004].

Configuration basée sur la modélisation des composants-connecteurs

Dans cette perspective, la configuration est représentée sous forme d'un graphe de composants et de connecteurs qui décrit comment ils sont coordonnés l'un à l'autre. Cette interaction est décrite par une interface de configuration. Pour assembler ces éléments de la configuration, deux types de liens existent :

— Attachement : exprimant quel composant donné sera connecté à quel connecteur.
— Liaison : définissant la relation entre un composant global et ses sous-composants.

Donc selon cette approche, les éléments clé de la configuration sont *les composants* et *les connecteurs*.

Les composants représentent des entités qui encapsulent des données et des fonctionnalités du système. Ils sont décrits par une interface qui expose les services qu'ils fournissent et les services dont ils ont besoin. Les bases de données sont un exemple de composants.

Les connecteurs comme leur nom l'indique représentent un moyen de combiner les composants. Un connecteur peut décrire une instruction d'interaction simple comme l'appel de procédure ou une interaction complexe comme les protocoles d'accès à la base de données. Elle est principalement représentée par une interface.

L'interface est la seule partie visible des composants et des connecteurs. Elle fournit l'ensemble des services offerts et l'ensemble des points d'interaction proposés qui sont appelés des ports qui peuvent être fournis ou requis. Pour les connecteurs, les points d'interaction sont appelés rôle qui sont aussi de deux types : les rôles fournis et les rôles requis.

La figure 1.13 décrit le schéma conceptuel ces éléments constitutifs de la configuration [Sadou *et al.*, 2005].

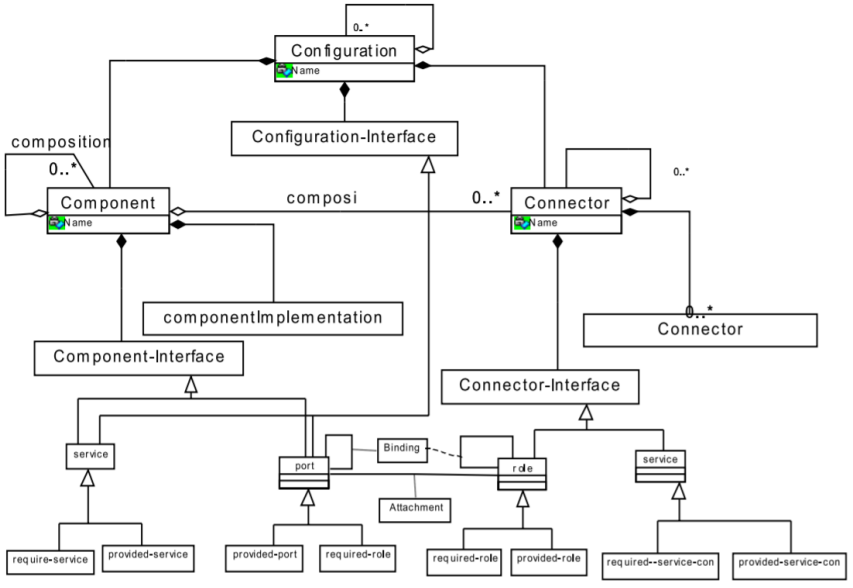


FIGURE 1.13 – Schéma de configuration à base de composant-connecteur

Deux autres concepts sont également utilisés dans cette optique de configuration : les opérations et les règles.

Les opérations s'appliquent à un des éléments de la configuration (les composants et les connecteurs) afin de la faire évoluer. Elles sont les suivantes : l'addition, la modification et la substitution. En général, la configuration du système est réalisée par l'exécution d'une ou plusieurs opérations sur ses éléments. Celle-ci est décrite par le second concept qui est un ensemble de règles. Elles expriment les conditions nécessaires à l'exécution de l'opération ainsi que le mécanisme de propagation aux dépendances. Le formalisme fréquemment utilisé est celui des ECA (Événement-Condition-Action). Une règle est ainsi construite de cette manière :

— L'événement déclenche la règle qui sera interceptée par le système de configuration pour l'analyse ;
— Puis, une ou plusieurs conditions qui doivent être satisfaites pour l'exécution de l'action de la règle qui peut être élémentaire (modification du nom d'un composant) ou complexe (combinant une suite de règle).

1.7.1.3 Recommandation

La recommandation est une autre technique de personnalisation visant à proposer à l'utilisateur le produit ou le service le plus approprié. Ainsi, c'est le système qui détermine pour chaque utilisateur le produit qui lui convient au mieux selon la prédiction de ses préférences alors qu'en configuration le rôle du système est de guider l'utilisateur à façonner son produit personnalisé.

La différence entre la configuration et la recommandation réside donc au niveau d'implication de l'utilisateur dans le processus de personnalisation. La première est perçue comme une activité contrôlée par les utilisateurs, alors que l'individualisation se concentre principalement sur la conception sur mesure de produits mettant l'accent sur la satisfaction des contraintes de l'utilisateur.

Les systèmes de recommandation peuvent être classés en trois grandes catégories comme il est représenté par la figure 1.14 [Bobadilla *et al.*, 2013, Ricci *et al.*, 2011].

Filtrage collaboratif

Le filtrage collaboratif s'appuie sur l'historique pour recommander aux utilisateurs des éléments qui ont déjà fait l'objet d'une évaluation positive par des utilisateurs ayant des préférences similaires avec l'utilisateur actuel. Ces systèmes de recommandation collaboratifs cherchent à prédire l'utilité et la pertinence des éléments pour un utilisateur particulier en se basant sur les éléments précédemment évalués par d'autres utilisateurs [Adomavicius et Tuzhilin, 2005].

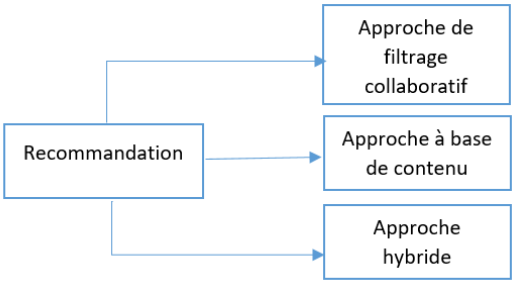


FIGURE 1.14 – Classes des systèmes de recommandation

L'hypothèse sous-jacente de cette technique de recommandation est que les utilisateurs ayant un comportement similaire ont des habitudes similaires. Si deux personnes A et B partagent la même opinion concernant un item i, alors A est plus susceptible d'avoir la même opinion que B sur un autre item j comparant à d'autres personnes choisies au hasard [Gandhi et Gheewala, 2017].

La principale limite qui est présentée par l'approche de recommandation par filtrage collaboratif est le problème du démarrage à froid. En effet, le système doit d'abord connaître les préférences de l'utilisateur en analysant ses activités passées. Avec l'entrée d'un nouvel utilisateur ou un d'un nouvel élément, il est difficile de trouver des similitudes à cause du manque d'informations. Une façon de surmonter le problème de démarrage à froid est de recueillir plus d'informations afin d'améliorer la performance des recommandations et de développer des systèmes de recommandation personnalisés.

Filtrage à base de contenu

Le filtrage basé sur le contenu, également connu sous le nom de Filtrage cognitif, recommande les éléments appropriés en faisant correspondre les attributs d'un profil d'utilisateur, regroupant les préférences et les intérêts, avec les attributs d'un objet (item). Plus l'utilisateur interagit avec le système, plus son profil s'enrichi. L'idée derrière ce système de recommandation est que " si l'utilisateur a aimé l'élément dans le passé, alors il aimera probablement le même type d'élément dans future " [Das *et al.*, 2017].

La principale limite des systèmes de recommandation par filtrage à base de contenu est la sur-spécialisation [Das *et al.*, 2017]. En effet, le système ne recommande que les éléments que l'utilisateur a aimé ou apprécié dans le passé. En se basant sur l'historique, il propose des éléments dont les scores sont élevés lorsqu'ils sont comparés au profil de l'utilisateur. De cette manière, l'utilisateur se voit recommander des éléments similaires à ceux déjà notés comme il n'y a pas de méthode inhérente pour trouver quelque chose d'inattendu.

Ceci devient problématique car l'utilisateur peut s'intéresser à des nouvelles choses et le système est incapable de les proposer à cause du degré de nouveauté limité. Par exemple, si l'utilisateur a aimé les lieux d'intérêt de type musée d'art, le système suggère des musées de ce type que l'utilisateur n'a pas encore vus, mais ce dernier peut aussi aimer les châteaux incluant des musées. Donc, la sur-spécialisation réduit le champ d'application, bien qu'il y ait beaucoup d'autres éléments potentiellement intéressants que l'utilisateur aime.

Recommandation hybrides

Un système hybride, comme son nom l'indique, vise à combiner deux ou plusieurs techniques dans le but d'utiliser les avantages de l'une pour corriger les inconvénients de l'autre [Thorat *et al.*, 2015, Ricci *et al.*, 2011]. Ils peuvent également être complétés par des techniques fondées sur les connaissances, afin d'améliorer les performances de la recommandation et d'éviter les limites des systèmes de recommandation traditionnels. Par exemple, le système de recommandation basé sur les connaissances EntreeC [Shah *et al.*, 2016] utilise certaines connaissances du domaine des restaurants, des cuisines et des aliments (par exemple " fruits de mer " n'est pas " végétarien ") pour recommander des restaurants à ses utilisateurs. L'idée derrière ces systèmes est d'exploiter une connaissance approfondie du domaine [Ricci *et al.*, 2011] pour le calcul des recommandations.

Le principal inconvénient des systèmes basés sur la connaissance est la nécessité d'acquérir des connaissances connues sous le nom de goulot d'étranglement. La représentation des connaissances la plus utilisée est sous forme d'ontologie où l'information est bien structurée et compréhensible par la machine. Par exemple, les systèmes Quickstep et Foxtrot [Middleton *et al.*, 2004] utilisent l'ontologie des sujets de recherche pour recommander des articles aux chercheurs.

1.7.2 Classes de personnalisation

Le problème de personnalisation peut être structuré en deux grandes classes : une première classe qui recommande une liste d'éléments et une deuxième qui s'intéresse à un autre niveau de granularité et propose une séquence d'éléments.

1.7.2.1 Personnalisation d'item

Les systèmes de recommandation traditionnels proposent les meilleurs item d'une collection d'éléments, par exemple des livres, des nouvelles, des documents de recherche ou des lieux d'intérêt [Deng *et al.*, 2013], classés selon une fonction d'utilité, satisfaisant certains critères définis par l'utilisateur [Deng *et al.*, 2015, Quadrana *et al.*, 2018, Baral *et al.*, 2018].

Ces systèmes s'intéressent généralement à la recommandation du prochain élément. Les informations historiques sur les habitudes des utilisateurs sont utilisées pour prédire le prochain item susceptible d'intéresser l'utilisateur [Baral *et al.*, 2018].

Des approches mettent l'hypothèse que pour les utilisateurs ayant des intérêts similaires, les prochains éléments seront les mêmes. Les différents éléments présents dans l'historique des utilisateurs sont modélisés sous forme de graphe orienté et pondérés avec des valeurs de similarité permettant de déterminer l'élément prochain. Une hypothèse peut se dégager affirmant que les utilisateurs s'intéressent souvent aux nouveaux éléments notamment dans le cas de la recommandation des Points d'Intérêt (POI) où les visiteurs cherchent les nouveaux points d'intérêt à explorer. [Laß *et al.*, 2017] suivent cette hypothèse donnent plus d'importance à l'information globale dégagée de l'historique et reposent moins sur la similarité des utilisateurs. Une intégration métrique par les paires est développée pour modéliser les transitions entre les POIs. Les paires de POIs pour lesquelles les transitions sont plus dans les données observées sont mappées plus près dans l'espace d'intégration. Cette incorporation de la proximité géographique entre les points favorise la personnalisation des propositions. Différents facteurs contextuels tels que le contexte temporel, l'influence géographique et les métadonnées sont exploités pour apprendre les représentations des éléments et les utiliser pour fournir un classement des candidats potentiellement intéressants pour l'utilisateur.

La plupart de ces algorithmes de recommandation traditionnels supposent généralement l'indépendance entre les éléments et se basent sur le principe de l'ordonnancement de ces items selon l'intérêt de l'utilisateur. Les centres d'intérêt sont indiqués explicitement dans le profil de l'utilisateur ou récupérés à partir des méthodes d'apprentissage. En l'occurrence, dans le domaine touristique, la position géographique, la fréquence de visite (check-in) ou la popularité sont des facteurs exploités pour l'apprentissage. Un score de pertinence des POIs est calculé ensuite et ces derniers sont ordonnés selon ce score et le top k résultats sera proposé à l'utilisateur [Li *et al.*, 2015a] [Yang et Fang, 2014] [Zhang et Chow, 2015b] [Zhang et Chow, 2015a].

Malgré que ces techniques reposent sur des informations contextuelles et des metadonnées (évolution des goûts de l'utilisateur) pour la personnalisation des recommandations, celles-ci ne tiennent compte ni de l'aspect consécutif des éléments, ni des contraintes de l'utilisateur.

1.7.2.2 Personnalisation de séquence d'items

Les systèmes de recommandation classiques proposent des recommandations portant sur des items individuels, par exemple, des livres, des DVD, des lieux d'intérêt, etc. Cependant, il y a plusieurs applications qui peuvent bénéficier d'un système capable de recomman-

der des ensembles d'items [Xie *et al.*, 2012, Jiang *et al.*, 2016, Eirinaki *et al.*, 2018]. Les exemples de telles applications peuvent inclure la planification de voyage [Xie *et al.*, 2011, **?**] (un séjour touristique), la musique [Hansen et Golbeck, 2009] (par exemple les listes de lecture de chansons), l'éducation [Parameswaran *et al.*, 2011] (une collection de cours) ou la restauration (une liste de repas pour un régime hebdomadaire). En particulier, pour le domaine touristique, un utilisateur est naturellement plus intéressé par des suggestions de lieux à visiter, ou des lieux d'intérêt organisés en paquets plutôt que des listes de classement [Amer-Yahia *et al.*, 2014, Benouaret et Lenne, 2016]. Ceci constituera une expérience exploratoire améliorée pour le visiteur. Le premier système opérationnel de recommandation touristique a été introduit par Delgado et Davidson [Liu *et al.*, 2012].

Le problème de personnalisation de séquence d'items est plus difficile que celui de la personnalisation d'éléments individuels. Puisque, les items doivent non seulement répondre à des critères multiples de sélection d'éléments individuels, mais aussi satisfaire des contraintes de compatibilité et des contraintes globales définies sur l'ensemble des éléments. Comme les caractéristiques des joueurs dans la recommandation d'équipe de jeu ou les prérequis nécessaires dans la recommandation d'une collection de cours [Deng *et al.*, 2013, Deng *et al.*, 2015, Eirinaki *et al.*, 2018] pour personnaliser les programmes d'études. L'interdépendance géographique doit être prise en compte dans la recommandation de parcours touristiques, il n'est pas optimal de proposer des POIs répartis aux quatre coins de la ville pour la même journée. Donc, l'aspect consécutif des éléments est bien considéré par cette catégorie de systèmes de recommandation.

Deux types d'approches pour la recommandation de séquence d'éléments se distinguent. La première approche subdivise le problème en deux sous-étapes : l'estimation d'un score de pertinence personnalisé pour chaque élément puis la résolution du problème d'optimisation [Yu *et al.*, 2016, Schaller *et al.*, 2014]). Généralement la construction de la séquence est calculée à l'aide d'algorithme d'optimisation sous contraintes ou l'algorithme de tournée sélective du domaine de la recherche opérationnelle [Vansteenwegen *et al.*, 2011, Gavalas *et al.*, 2014]. Quant à la deuxième approche, la séquence est déterminée selon un processus incrémental en calculant la probabilité de transition d'un élément à un autre [Sang *et al.*, 2015].

1.8 Conclusion

Nous avons présenté dans ce chapitre les différentes notions sur lesquels nous nous appuyons dans cette thèse à savoir les services et les APIs Web, le développement orienté utilisateurs final et la personnalisation. Ces dernières traitent les problématiques d'intégration et de personnalisation de données. L'objectif de cette thèse vise à tisser des liens

entre ces deux problématiques afin de pouvoir présenter à l'utilisateur une vision unifiée et customisée des données. Avant de présenter notre solution qui permet d'atteindre cet objectif, nous étudions dans le chapitre suivant les travaux de la littérature relatifs à ces directions de recherche.

Chapitre 2

Agrégation et personnalisation en littérature

Aprés avoir défini les concepts de base, nous détaillons au niveau de ce chapitre les approches présentées dans la littérature dans les domaines d'agrégation et de personnalisation en se focalisant sur les critères suivants pour le mashup à savoir, le développement orienté utilisateur final, le développement spécifique au domaine, le lien avec les données liées et l'agrégation interactive. Nous analysons aussi, les travaux traitant la personnalisation avec concentration particulière sur les critères de diversification des éléments personnalisés, le sens du contrôle et la transparence de la personnalisation interactive.

La problématique abordée dans cette thèse vise à croiser ces domaines comme illustré dans la Figure 2.1.

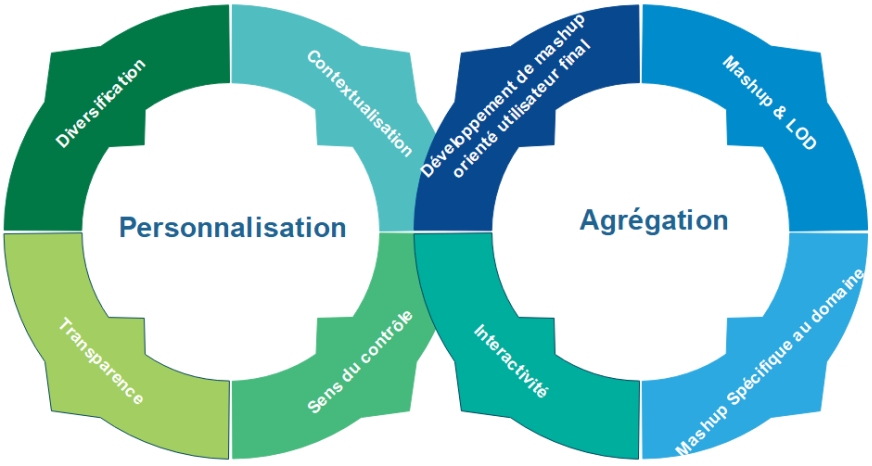


FIGURE 2.1 – Modèle unifié des deux techniques

2.1 Mashup de services

L'une des raisons de l'émergence des mashups était le désir des utilisateurs finaux de construire des applications répondant à leurs besoins situationnels. C'est ce qui est appelé l'innovation axée sur l'utilisateur qui dépend de l'implication de ce dernier dans le processus de mashup et a changé par conséquent la façon dont les mashups sont développés.

Ceci fait appel, en particulier, aux techniques de développement orientées utilisateur dans la conception d'outils de mashup. Ainsi nous mettons le focus dans cette section sur la liaison entre les techniques de mashup et les paradigmes du développement centré sur l'utilisateur. Une discussion des principaux éléments, qui promeuvent le mashup permettant aux utilisateurs finaux de créer leurs propres applications par composition, est présentée par [Soi *et al.*, 2014]. Il s'agit (i) de processus dits « légers » qui se caractérisent par *des métaphores intelligibles* favorisant l'innovation pilotée par l'utilisateur, mais aussi (ii) *des mécanismes d'assistance* capables d'aider les utilisateurs à définir la logique d'intégration et (iii) *une adaptation au domaine* afin d'exprimer les besoins recherchés.

Nous avons analysé les plus importants travaux de [Matera *et al.*, 2013, Roy Chowdhury *et al.*, 2013, Cappiello *et al.*, 2015, Cheng *et al.*, 2017, Ardito *et al.*, 2018, Barricelli *et al.*, 2019] qui abordent les obstacles typiques de l'utilisation des outils de mashups orientés utilisateur par une série d'expériences visant à identifier les difficultés des utilisateurs finaux. En particulier, leur solutions pour maîtriser le langage visuel proposent des alternatives de conception qui peuvent être utilisées pour classer les outils de mashup et fournir une feuille de route pour le développement des outils de mashup de nouvelle génération.

2.1.1 Représentation orientée utilisateur

La modélisation orientée utilisateur d'outils de mashup repose sur deux éléments fondamentaux pour l'agregation de services comme nous l'avons décrit précédemment, la représentation des services eux-même puis l'adoption d'un mécanisme d'interaction adéquats.

2.1.1.1 Représentation des services

La notion de composant unifié qui regroupe différentes technologies dans un modèle abstrait avec une notation visuelle est appropriée au modèle mental de l'utilisateur final. Plus les constructions sont abstraites, plus l'outil de mashup résultant est générique ; plus il est concret, plus il est spécifique au domaine. Dans ce contexte, des paradigmes *WYSIWYG (What You See is What You Get)* ont été proposés dans lesquels les composants sont représentés au moyen de leurs interfaces utilisateur, fournies nativement par le service lui-même ou produites par l'outil de mashup, permettant de générer un retour immédiat

du comportement des composants. De même, l'effet de toute action de composition visant, par exemple, à synchroniser le comportement des composants, est immédiatement représentée, selon un style hautement interactif propre à la "programmation en direct". Ainsi, l'application est automatiquement et continuellement reconstruite et exécutée en réponse à toute modification.

A travers la manipulation des objets visuels et des propriétés visuelles, les utilisateurs seront capables de construire le mashup facilement. Par exemple, l'ajout d'un tel objet dans l'espace de travail de composition avec visualisation de l'interface utilisateur et des données fournies par le composant correspondant permet de réduire les barrières conceptuelles. En effet, l'évaluation de ce type d'approches visuelles d'après [Daniel et Matera, 2014] a montré que la concentration sur la visualisation immédiate des services augmente la perception de l'utilité et de la facilité d'utilisation de l'outil de mashup.

Dans le même esprit que le paradigme WYSIWYG, la technique " glisser-déposer " est récemment employée dans les approches de mashup visuel [Husmann *et al.*, 2014, Radeck *et al.*, 2013].

Par exemple, un paradigme de notation visuelle est fourni par [Aghaee et Pautasso, 2014], NaturalMash, qui se démarque par l'utilisation d'un langage naturel structuré à travers lequel les utilisateurs peuvent exprimer leurs intentions sur les composants. L'avantage d'utiliser le langage naturel réside dans sa compréhensibilité, qui raccourcit et simplifie la courbe d'apprentissage. Il s'agit d'une technique de composition interactive hybride combinant un langage naturel contrôlé pour le développement de mashup avec une interface visuelle et le paradigme glisser-déposer permettant la pré-visualisation et la modification du processus de mashup. NaturalMash adopte une démarche incrémentale axée sur l'utilisateur où ce dernier n'a qu'à taper des requêtes assistées. Il vise à aider les utilisateurs à exprimer en langage naturel quels services ils veulent inclure dans leur mashup et comment les orchestrer. Néanmoins, l'expression des exigences est limitée à un sous-ensemble d'un langage naturel avec un vocabulaire et une grammaire restreints pour des raisons d'amélioration de l'exactitude des demandes des utilisateurs.

[Weber *et al.*, 2013] proposent une approche de mashup basée sur des formulaires, qui permet aux utilisateurs d'exprimer des logiques d'intégration de services via des formulaires. Les services Web sont représentés sous forme de formulaires où ils sont décrits à l'aide de noms significatifs pour l'utilisateur et indépendants des noms techniques. Les auteurs proposent aussi une modélisation des compositions dans un langage générique restreint accompagnée d'un processus de vérification immédiate pour réduire la charge de travail de l'utilisateur et lui permettre de construire une composition correcte.

Nous citons aussi l'outil SpreadMash présenté par [Kongdenfha *et al.*, 2008] qui exploite les blocs de construction d'applications appelés widgets de données décrivant divers modèles

d'importation et de présentation de données dans les feuilles de calcul. Les widgets de données permettent de séparer les tâches des utilisateurs finaux (composition des widgets de données) des tâches des architectes de données (création d'abstractions de données et de widgets de données).

Nous pensons que la caractéristique visuelle est importante dans le développement efficace de mashup centré sur l'utilisateur final. Ainsi, nous optons pour cette représentation dans notre approche de mashup pour sa représentation compacte de la ressource d'un côté et sa capacité d'être un paradigme familier pour les utilisateurs novices de l'autre côté.

2.1.1.2 Représentation de la logique d'intégration

Dans une approche de mashup, les objets visuels sont dotés d'une logique d'intégration appelée par[Daniel *et al.*, 2009] intégration universelle. L'intégration universelle fait référence à l'intégration des données, de la logique applicative et des interfaces utilisateur dans un seul et même environnement de modélisation.

a. **Approches impératives**

La plupart des approches de mashups adoptent la notation câblée comme mécanisme d'interaction où les utilisateurs créent des diagrammes représentant les flux de données et le flux de contrôle pour intégrer les données. Ce choix de notation câblée est justifié par le fait de la similitude entre l'intégration des composants dans les mashups et la composition de services, un domaine où les notations schématiques sont largement adoptées par les développeurs pour représenter l'intégration des services [Daniel et Matera, 2014]. Nous étudions dans ce qui suit les approches de mashup employant cette notation.

Yahoo Pipes [Jones et Churchill, 2009] est l'outil de mashup le plus connu de Yahoo. Il fournit un éditeur visuel de composition qui permet de concevoir des logiques de traitement des données. L'éditeur de composition Yahoo Pipes offre un ensemble de composants visuels fonctionnant dans la logique graphique de glisser-déposer. Les composants admettent l'approche basée sur les données où chaque composant reste en attente jusqu'à ce que les données soient disponibles à son port d'entrée. Il convient que l'approche basée sur les données est plus intuitive pour l'utilisateur final que l'approche basée sur le contrôle tant qu'elle reste triviale [Daniel et Matera, 2014]. Ceci n'est pas tout à fait le cas de Yahoo Pipes où il offre un ensemble très technique de composants (composant boucle, composant, expression régulière, etc) complexifiant le mashup pour les utilisateurs finaux. En effet, ils ont besoin des concepts de base de

la programmation afin de pouvoir connecter les composants entre eux. Les notations de programmation restent, malgré l'encapsulation par une couche de visualisation, difficiles à comprendre par les experts non informaticiens.Le problème est dû à la généricité de l'outil qui se restreint à une terminologie pas forcément compréhensible par les experts du domaine.

En plus de Yahoo Pipes, les outils de mashup industriels ne manquent pas MashMaker [Ennals et Garofalakis, 2007], Mashroom [Wang *et al.*, 2009], MashQL [Jarrar et Dikaiakos, 2008] et la liste est encore longue. Un aperçu de ces outils est donné par [Yu *et al.*, 2008] soulignant les caractéristiques de ces derniers. Tous ces outils sont généralement accompagnés d'un ensemble de ressources facilement réutilisables par exemple, une carte Google, un calendrier, un adaptateur de flux RSS ou des opérateurs de transformation de données. Ils sont représentés sous forme de composants pouvant être configurés et intégrés grâce à des notations visuelles de développement qui visent à faciliter la tâche de mashup d'après [Roy Chowdhury *et al.*, 2014]. Toutefois, ces solutions industrielles sont sophistiquées pour les utilisateurs novices en demandant des connaissances élevées.

Le côté académique s'attaque également à ce challenge où, [Daniel *et al.*, 2009] proposent leur outil de mashup MashArt dans le cadre d'un projet universitaire. MashArt adopte principalement une approche d'intégration universelle pour la connexion de composants centrée sur l'utilisateur final. L'intégration se base sur un modèle de données réalisant un ensemble d'affectations de paramètres pour combiner les composants qui sont livrés en trois types (données, services et interfaces utilisateur). MashArt propose un éditeur Web simple et un environnement d'exécution léger intégré permettant une pré-visualisation instantanée destiné aux utilisateurs finaux non experts en informatique. Ses connecteurs de flux de données permettent la définition d'un mapping personnalisé des paramètres indiquant au moteur de mashup quels paramètres de sortie doivent être affectés à quels paramètres d'entrée de l'opération cible. Bien que MashArt a opté pour une approche orientée utilisateur final, il n'est pas en mesure de le cibler efficacement car il ne fournit que les composants de types génériques.

Dans le contexte mobile, MobiMash proposé par [Cappiello *et al.*, 2012] est un environnement de conception permettant aux utilisateurs de composer des mashups mobiles en complétant des modèles visuels. Les modèles visuels adoptés jouent un double rôle. D'un côté, ils offrent une représentation intuitive de la façon dont les données seront affichées dans l'application finale et de l'autre, leur complément visuel avec des champs de données assure un paradigme d'intégration par exemple. De plus, MashupEditor [Ghiani *et al.*, 2016], un environnement intelligent pour le développement de mashups, se base sur la réutilisation des composants existants de différentes applications pour en créer de nouvelles par combinaison. Le processus de mashup

exploite une métaphore intuitive du copier-coller qui s'inspire du paradigme de la programmation par l'exemple.

Ces approches de mashup reposent fortement sur les connexions entre les composants dans une logique intrinsèquement impérative qui bien qu'elles adoptent des paradigmes visuels, elles exposent trop de détails techniques pour qu'elles soient adaptées au modèle mental de l'utilisateur novice. D'autres se concentrent plutôt sur les composants selon une logique déclarative.

b. **Approches déclaratives**

Dans le cadre des approches déclaratives pilotées par les objectifs, le mécanisme d'interaction le plus adopté est les mots-clés ou les tags qui décrivent de manière intuitive les capabilités des services [ZHOU, 2017].

[Sabatucci *et al.*, 2016] proposent une approche de mashup en suivant le principe de découplage entre ce qu'il faut faire et comment le faire. Les auteurs représentent la logique du mashup sous la forme d'un objectif exprimé en tant que spécification GoalSPEC. Il s'agit d'un langage orienté objectif permettant de décrire le comportement attendu d'un système en termes d'états à traiter.

La plateforme de mashup Spacebrew [1] exploite aussi le paradigme de la programmation visuelle. Il s'agit d'une boîte à outils pour connecter des services web et des objets intelligents au moyen de règles événement-condition-actions. Les règles peuvent être créées dans un espace de travail divisé en deux parties : un panneau pour la configuration des services publiant les événements et un autre panneau pour la configuration des services fournissant des actions en réponse aux événements.

De même, une autre approche de mashup, présentée par [Desolda *et al.*, 2015], consiste en une plateforme appelée EFESTO privilégiant les paradigmes de composition visuelle qui s'adaptent au modèle mental de l'utilisateur. EFESTO offre un modèle de source de données "polymorphe" qui, en exploitant les données liées, permet d'accéder à des informations "mutables" en fonction des besoins situationnels exprimés dans le mashup en construction. L'espace de travail interactif présenté, fournissant une exploration et une composition transparentes de sources hétérogènes, est appelé "live mashup". Les composants graphiques faisant l'objet d'une action sont appelés "actionables" [Desolda *et al.*, 2017a] dans le vocabulaire EFESTO. Une extension de EFESTO [Desolda *et al.*, 2016] qui s'accompagne d'un paradigme de composition basé sur des règles se focalise sur les ressources de types objets intelligents. Le paradigme de composition est caractérisé par des opérateurs pour coupler de multiples événe-

1. http ://docs.spacebrew.cc/

ments et conditions exposés par des objets intelligents, et pour définir des contraintes temporelles et spatiales sur l'activation des règles. Dans sa nouvelle version EFESTO s'adresse aux utilisateurs finaux via une notation visuelle de composition de règles qui permet d'exprimer la composition d'objets intelligents d'une manière déclarative sous la forme de "Quel est le service à interroger pour détecter l'événement déclencheur ?", "Quel événement de service doit être exécuté ?", "Quand et Où l'événement doit se produire ?"

Une synthèse des métaphores et styles de programmation utilisés dans les approches orientées utilisateur final est illustrée par la figure 2.2 présentée par [Paternò et Santoro, 2017].

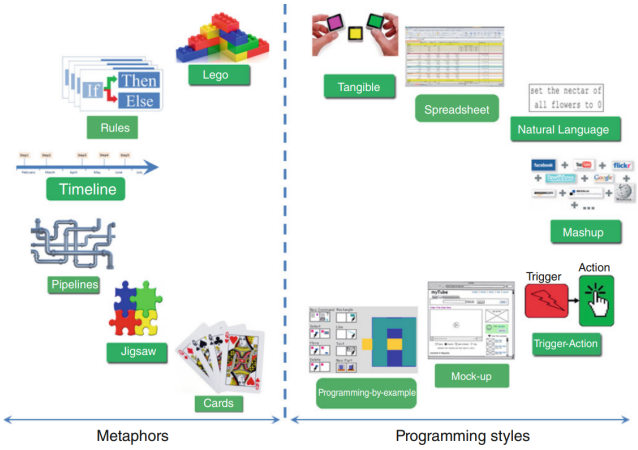


FIGURE 2.2 – Les métaphores et les styles de programmation utilisés en développement orienté utilisateur final

Nous pensons que le raisonnement déclaratif à base de règles constitue un bon compromis entre l'intuitivité et l'expressivité ce qui nous a conduit à l'adopter dans notre approche de mashup.

2.1.2 Assistance au développement de mashup

Partant du fait que la sélection des "bons" composants pour réaliser le mashup peut ne pas être triviale pour les utilisateurs novices, des approches de recommandation visant à assister les utilisateurs dans le processus de mashup ont été proposées par [Roy Chowdhury *et al.*, 2013, Cao *et al.*, 2013, Gao et Wu, 2017]. Ces derniers proposent une étude de compatibilité et de similarité pour vérifier la cohérence des composants dans l'espace de modélisation de mashup. Des recommandations sont ensuite générées en classant les composants disponibles sur la base de leur attitude à augmenter la qualité technique ainsi que la va-

leur ajoutée du mashup. Elles peuvent également tenir compte de l'historique d'interaction et des meilleures pratiques adoptées par les communautés de développeurs de mashup, en exploitant des mécanismes de filtrage collaboratifs telles que celles proposées par [Buqing *et al.*, 2014, Yao *et al.*, 2018, Samanta et Liu, 2017]

En plus de recommander des composants individuels, il est plus utile de suggérer des connaissances plus complexes sur la composition, par exemple sous la forme de modèles de mashups. En particulier [Roy Chowdhury *et al.*, 2014] proposent une technique de composition interactive permettant d'aider le développeur à affiner progressivement son modèle de mashup par la suggestion de la prochaine action de modélisation qui peut être exécutée (par exemple, saisir des données ou connecter des composants). Ils s'appuient sur des modèles de composition réutilisables appelés fragments de modèles représentant des connaissances d'une variété de sources. Une mise en œuvre de cette architecture de recommandation sous forme d'outil de mashup de données appelé Baya dont la description est proposée par [Roy Chowdhury *et al.*, 2012]. Baya est une extension de de la plateforme Yahoo ! Pipes qui guide le développement du mashup en recommandant la connaissance de la composition sous la forme de modèles de mashup réutilisables qui sont extraits d'un référentiel de pipes existantes puis classés.

Toutefois, cette approche s'adresse aux utilisateurs qui sont déjà familiers avec l'environnement de composition, mais qui ont besoin d'aide pour trouver les blocs de construction appropriés. Or, dans le contexte de mashup pour les utilisateurs novices qui ne disposent pas ou très peu d'expertise en développement, ceci reste trop complexe.

MashupAdvisor, un système proposé par [Elmeleegy *et al.*, 2008] , suggère à partir d'un composant placé par l'utilisateur, un ensemble de composants connexes en utilisant les probabilités de co-occurrence conditionnelle et l'appariement sémantique pour classer les résultats du mashup. Le processus se déroule en deux étapes. En premier lieu, le MashupAdvisor génère un classement des sorties recommandées qui peuvent être ajoutées au mashup partiel, ensuite lorsque l'utilisateur sélectionne une sortie recommandée, MashupAdvisor calcule le meilleur plan pour générer la sortie sélectionnée compte tenu du mashup partiel. Un processus qui peut prendre plus de temps ce qui le rend peu efficace pour les systèmes dynamiques.

Dans le même esprit, Marmite [Wong et Hong, 2007] supporte le mashup de données de manière fluide par le biais de connecteurs de type opérateur de jonction et de filtrage, etc. C'est dans un principe similaire aux tubes Unix, que cette plateforme connecte les données par une série d'opérateurs.

[Domínguez et Ko, 2018, Ardito *et al.*, 2018] adoptent le mécanisme événementiel des règles. Ce dernier a été prouvé, dans les travaux de [Wajid *et al.*, 2011, Desolda *et al.*, 2017c],

qu'il est une alternative simple et efficace pour assister les utilisateurs dans leurs activités quotidiennes. Par exemple, [Ur *et al.*, 2014] proposent un modèle de recommandation de mashup de services, basée sur des règles de type événement-condition-action (ECA), pour guider les utilisateurs dans les taches d'automatisation dans le contexte des maisons intelligentes. Cette approche de mashup, également connue sous le nom d'approche if-then-else, est adoptée par plusieurs outils de mashup récents tels que IFTTT, Atooma ou tasker permettant aux utilisateurs de créer des chaînes de règles. Un tel paradigme repose sur une structure compacte et intuitive, qui relie directement les événements et/ou conditions dynamiques avec les réactions attendues sans nécessiter l'utilisation de structures de programmation complexes.

Dans un autre domaine qui est le bien-être électronique, [Barricelli et Valtolina, 2017] proposent une méthodologie pour modéliser graphiquement les règles en utilisant une notation visuelle capable d'aider les membres des sports d'équipe à synchroniser les activités physiques de leurs athlètes. Leur Framework SmartFit vise à collecter et connecter les données provenant des dispositifs, des capteurs et des applications au via un éditeur de règles. Il offre aux experts du domaine le moyen d'exprimer des règles complexes et des contraintes temporelles fournissant une solution de mashup ad-hoc et personnalisée. *Nous adoptons ce paradigme événementiel dans notre approche de mashup.*

La plupart des solutions proposées partagent une caractéristique commune qui est la généricité où elles visent à accroître leur applicabilité dans différents domaines. Bien que cela semble être un avantage évident, dans le contexte du développement orienté utilisateur final, il s'agit plus d'un défaut que d'avantage affirmé par [Casati, 2011]. Ces derniers sont souvent trop génériques pour répondre à des besoins spécifiques. En effet, l'absence de personnalisation de la fonctionnalité de mashup constitue un obstacle pour les experts du domaine. Afin d'accroître l'efficacité des approches de mashup orientées utilisateur final, il est nécessaire de passer des solutions de mashup horizontales polyvalentes à des solutions verticales spécifiques à un domaine, permettant aux experts du domaine de composer de manière autonome leurs applications.

2.1.3 Mashup spécifique au domaine

L'idée de se concentrer sur un domaine particulier pour créer des environnements de développement plus efficaces et plus simples est débattue dans les travaux de [Lédeczi *et al.*, 2001, Mernik *et al.*, 2005, Maximilien *et al.*, 2007, Casati *et al.*, 2012, Soi *et al.*, 2014] . Ces auteurs affirment que le développement de mashup spécifique à un domaine nécessite de renoncer à la généricité des concepts utilisés et de réduire l'objectif de l'outil à un domaine bien défini en adaptant le paradigme de développement, ses modèles et ses composants aux

besoins spécifiques de ce domaine.

FlexMash est un outil de mashup proposé par [Hirmer et Mitschang, 2015] utilisant un modèle de mashup spécifique appelé plan de mashup afin de permettre aux utilisateurs de définir des scénari d'intégration de données. Le plan de mashup, introduit par les auteurs, est un modèle de graphe qui représente les sources de données sous forme d'objets métier modélisant un objet spécifique à un domaine. Il inclut aussi des opérations de traitement de données de type conjonction et filtre propres au domaine. Par une subdivision du mashup de données en quatre niveaux d'abstraction : niveau de modélisation, de transformation, d'exécution et de présentation, une flexibilité du processus d'intégration est offerte. Cependant, l'interaction avec l'utilisateur est limitée à la phase de modélisation. Ceci a conduit la mise en œuvre d'une seconde version FlexMash 2.0 décrite dans [Hirmer et Behringer, 2016] qui se concentre sur l'amélioration de la convivialité parmi les nouvelles fonctionnalités introduites. Le nœud de fusion visuelle permet l'intégration interactive de différentes sources de données sous le contrôle total de l'utilisateur. Toutefois, cette séparation des niveaux d'abstraction n'est pas tout à fait adéquate pour les utilisateurs novices car ils ne sont pas en mesure de distinguer la modélisation de l'exécution, ils ont besoin d'un retour immédiat de toute action de composition.

Suite à une étude de leur outil EFESTO, [Ardito *et al.*, 2017] ont constaté que la plateforme est "trop générique" pour satisfaire les exigences de nombreux domaines. Ils proposent donc la personnalisation des domaines comme solution pour rendre la méta-conception plus efficace pour la création de plateformes qui répondent réellement aux besoins de l'utilisateur final. Cette approche se base sur un méta-modèle de conception structuré en trois niveaux et exploitant la spécificité du domaine. Le premier niveau fait référence à une activité de méta-conception réalisée par des développeurs professionnels qui implémentent des modèles visuels génériques. Ensuite, la couche suivante s'intéresse à personnaliser l'outil général résultant. C'est les experts du domaine de l'archéologie qui interviennent à ce niveau en manipulant un environnement de composition permettant de sélectionner les services et modéliser le mashup par le biais d'opérations de composition de données comme la jointure et l'union. Et la dernière couche se présente comme à un espace interactif manipulé par les utilisateurs finaux leur permettant de sélectionner les services, de les interroger et d'agréger le contenu récupéré. Ainsi, la particularité de cette approche est l'introduction d'activités méthodologiques qui traitent de l'adaptation des systèmes à des domaines spécifiques.

[Casati *et al.*, 2012] proposent également un outil de mashup spécifique au domaine se basant sur la méthodologie de développement de mashup spécifique au domaine décrite dans le chapitre précédent. ResEval Mash [Muhammad *et al.*, 2012] , leur plateforme de mashup traite le domaine de l'évaluation de la recherche scientifique en termes d'agrégation

des données sur les publications scientifiques et les chercheurs, de calcul de métriques et leur visualisation. Il s'agit d'un environnement de mashups doté d'une interface utilisateur graphique intuitive qui fournit à chaque objet de l'environnement une métaphore visuelle que l'expert du domaine connaît. La particularité de cette approche est la restriction du langage de mashup à un seul domaine et les composants de mashup au modèle conceptuel du domaine. Ce faisant permet aux utilisateurs de ne pas traiter le mapping des données qui est un aspect complexe du mashup. Ce dernier est déterminé automatiquement. Il reste aux utilisateurs que la charge du flux de données.

Nous pensons que cette manière simplifie considérablement le processus de mashup pour les utilisateurs novices. C'est pourquoi nous l'adaptons dans notre approche.

2.1.4 Mashups dans le domaine du tourisme

[Govardhan et Feuerlicht, 2007] proposent une application de calcul d'itinéraires qui permet aux utilisateurs de créer un plan des destinations qu'ils prévoient visiter et de les afficher sur une carte. Cependant, le mashup ne prend pas en compte des données dynamiques qui peuvent influencer le calcul d'itinéraire. Motivés par le problème d'accès des touristes à l'informations lorsqu'ils sont en déplacement et par la prolifération des appareils mobile, [Meng et Xu, 2010] proposent un système de guide touristique mobile qui se base sur la technologie des mashups pour combiner les ressources touristiques afin de créer des services à valeur ajoutée. La liste des lieux d'intérêt proposée reste insuffisante pour planifier un itinéraire surtout pour les nouveaux visiteurs.

Aussi, [Kanellopoulos *et al.*, 2010] proposent un framework conceptuel qui intègre des applications Web liées au tourisme. Les auteurs se sont inspirés du processus d'achat regroupant trois grandes étapes (pré-achat, achat et post-achat) pour la conception de leur framework de mashup. Ils exploitent ainsi cette structure dans le domaine touristique pour répondre aux besoins des voyageurs. A chaque phase, l'utilisateur exécute les applications Web appropriées pour satisfaire ses besoins en tant que voyageur. Ce faisant ne permet que lancer des pointeurs aux applications indépendantes sans permettre une agrégation des données pour unifier et construire une vue globale. A priori, il ne s'agit pas d'une approche de type mashup mais plutôt d'une recommandation d'une séquence d'applications.

Smart Travel Planner, un outil de mashup pour faciliter le processus de planification des voyages implémenté par [Jafri *et al.*, 2013], propose d'intégrer les données nécessaires à la planification des voyages dans leur intégralité dans une seule plateforme. Ceci constitue une vraie approche de mashup. La particularité de ce système est la prise en compte des préférences utilisateurs en termes de disponibilité en leur permettant de bloquer certains horaires. [Cano *et al.*, 2011] proposent Topica une approche de mashup qui enrichit

la structure des points d'intérêt en ajoutant des ressources DBpedia extraites des commentaires sur ces derniers. Le mashup extrait les données d'interaction sociale des utilisateurs pour les compléter avec les données liées afin de présenter aux visiteurs les lieux d'intérêt à proximité. Néanmoins cette approche n'aide pas les visiteurs qui cherchent à planifier une journée touristique.

[Ardito *et al.*, 2018]visent à aider les utilisateurs finaux à créer des expériences de visite personnalisables en utilisant la technologie de mashup adaptée au besoin du domaine de l'héritage culturel. Ils ont étendu un paradigme de composition générique, initialement conçu pour le développement orienté utilisateur final des systèmes de l'internet des objets, pour satisfaire les exigences de l'utilisateur lors des visites des sites culturels. Grâce aux éditeurs Web, les utilisateurs peuvent synchroniser le comportement des objets intelligents en définissant des règles de type événement-condition-action. La plateforme EFESTO-SE implémentant cette approche, se base sur les objets intelligents comme des composants constructeurs du mashup qui, en plus de leur fonction primaire de dispositifs capables de contrôler l'environnement, peuvent s'intégrer de manière transparente. Ceci permet aux acteurs de l'héritage culturel de devenir des concepteurs de l'expérience intelligente. Malgré l'efficacité de cette approche dans la configuration des expériences intelligentes, celle-ci peut être compléter par des méthodes de représentation des connaissances, telles que les ontologies et les données liées, afin que les utilisateurs finaux puissent être supporter dans le processus de personnalisation de la composition.

2.1.5 Mashup et données liées

Nous décrivons dans cette section les travaux qui combinent les techniques de mashup avec le concept des données liées.

[Ekaputra *et al.*, 2017] proposent la plateforme Open Data Mashups for Data qui vise à établir un lien entre les éditeurs et les consommateurs de l'information. Elle facilite l'accès, la réutilisation et l'intégration des données ouvertes pour le journalisme des données. Cette plateforme est implémentée en utilisant la «linked widget platform» un environnemment de mashup de données où les utilisateurs finaux peuvent combiner des ensembles de Widgets liés comme un mashup pour répondre à leurs besoins en données agrégées. En plus, elle offre aux journalistes des visualisations interactives des données.

De manière similaire, [Trinh *et al.*, 2016] utilisent les « linked widget » dans leur approche de mashup des données ouvertes. Remarquant qu'il est difficile pour les utilisateurs finaux d'utiliser efficacement les données ouvertes vu qu'elles ne sont pas nécessairement publiées de manière à ce qu'elles puissent être facilement découvertes et comprises ; et qu'il y a un manque d'outils qui permettraient aux non-experts de les intégrer, les auteurs répondent à ce défi par une plateforme combinant les concepts du web sémantique et de

mashup. L'objectif étant d'aider l'utilisateur à explorer les données via un modèle sémantique de composants de mashup. Les tâches d'intégration de données sont vues comme des fonctions réutilisables encapsulées dans des blocs d'interface utilisateur de haut niveau.

Motivés par la représentation simpliste des dispositifs et des services de l'internet des objets, [Corno *et al.*, 2019] présentent leur plateforme EUPont qui permet la création d'applications IoT de haut niveau, capables de s'adapter à différentes situations contextuelles. EUPont intègre un modèle ontologique dans l'architecture d'une plateforme de mashup. Elle offre une représentation ontologique de haut niveau pour le développement orienté utilisateur final à base de règles événementielles. Cependant, la situation contextuelle ici caractérise uniquement une adaptation technique où la solution doit fonctionner sur différents dispositifs.

La plateforme OntoMash proposée par [CHOI, 2019] permet également d'agréger multiple informations contextuelles pour la gestion des environnements d'internet des objets en se basant sur un modèle ontologique. OntoMash vise à établir des relations sémantiques entre des classes de contexte. La philosophie d'OntoMash est une adaptabilité permettant d'ajuster les services contextuels en fonction de diverses sources contextuelles dans un environnement ouvert et dynamique. Même si les techniques de modélisation fondées sur l'ontologie présentent des avantages en ce qui concerne la gestion de données diverses, elles ne sont pas toujours raisonnables lorsqu'il s'agit de trouver un compromis entre l'expressivité et la complexité.

Ainsi, le volume de données et de services disponibles exige non seulement un besoin d'intégration mutuelle mais aussi une adaptation afin d'offrir à l'utilisateur une vision appropriée et améliorer l'efficacité du traitement en fournissant uniquement ce qui est pertinent en fonction du contexte. Une structuration du concept « contexte » qui permet de préciser une situation contextuelle se voit un besoin nécessaire. La section suivante présente un aperçu de l'état de l'art de la composition sensible au contexte.

2.2 Composition sensible au contexte

Les travaux de [Alegre *et al.*, 2016, Abdulkarem *et al.*, 2019] traitant la problématique de l'adaptabilité et de la sensibilité au contexte peuvent être classifiés en trois catégories : Les approches pro-actives ou les approches de Pré-Filtrage qui tentent de prédire les changements futurs possibles pour effectuer l'adaptation. Il s'agit de filtrer les données à l'aide d'un ensemble spécifié d'attributs contextuels avant d'appliquer la logique d'agrégation. La seconde catégorie décrit les approches réactives ou les approches de Post-Filtrage qui appliquent le filtrage des données après la composition. Enfin, la troisième catégorie représente les approches de modélisation contextuelle qui visent à conceptualiser les attributs

décrivant le contexte qui seront injectés dans la logique d'agrégation.

Nous suivons cette structuration dans ce chapitre en mettant particulièrement l'accent sur la troisième catégorie qui représente la classe de notre approche.

2.2.1 Approches pro-actives

La plupart des approches pro-actives considèrent le problème de la composition de services comme un problème de recherche dans un graphe. Un algorithme de recherche dans ce contexte est appliqué sur un graphe de dépendance pour récupérer les résultats de la composition des services.

Par exemple, [Elmaghraoui *et al.*, 2017] formalisent le problème de composition des services Web en tant que problème de recherche dans un graphe de dépendance de service AND/OR, où les nœuds représentent les services disponibles et les arcs représentent les dépendances sémantiques des entrées/sorties entre ces services. Un ensemble de techniques d'optimisation dynamique basées sur l'analyse de redondance et la dominance de service a été inclus pour réduire la taille de ce graphe et ainsi améliorer l'évolutivité et la performance de l'approche. Un pré-calcul de tous les chemins les plus courts entre chaque paire du nœud dans le graphe à l'aide d'un algorithme de recherche est présenté. La construction du graphe et le calcul des chemins les plus courts sont effectués hors ligne pour alléger le processus de recherche de la composition, optimisant ainsi le processus de composition en réduisant l'effort de calcul lors de l'exécution de la requête.

L'adaptation pro-active se concentre sur la détection précoce des exceptions où des méthodes de prédiction sont employées dans le processus d'adaptation. Alors qu'il n'est pas pratique de prévoir tous les éventuels changements, une adaptation en temps réel se voit essentielle.

2.2.2 Approches réactives

Comme les systèmes doivent réagir immédiatement dès qu'une situation indésirable est détectée, le scénario de composition de service a besoin d'un processus d'adaptation réactif capable de gérer la situation actuelle. La plupart des approches réactives ramènent le problème de composition dynamique de service à un problème de satisfaction de contraintes.

Encouragés par la grande mobilité des utilisateurs et de leurs appareils dans le contexte pervasif, [Cervantes *et al.*, 2017] présentent un protocole de composition dynamique pour les systèmes de systèmes dans des réseaux ad hoc. L'approche transforme le problème de composition dynamique en problème de satisfaction des contraintes pour établir les nouvelles exigences du système. Les contraintes traités sont de type spatiales et temporelles. Une heuristique permettant de recomposer les services basée sur l'identification de compo-

sants de service disqualifiés en raison du dynamisme de l'environnement est présentée. Ce qui évitera que la composition soit redéfinie à partir de zéro.

Ces approches réactives exécutent des actions correctives après l'exécution de services, ignorant ainsi l'opportunité de prévenir les comportements erronés à un stade précoce. Elles provoquent une interruption de l'exécution jusqu'à ce que la ré-sélection soit effectuée. Par conséquent, une approche hybride sera intéressante. Elle devra adopter un processus de détection précoce capable de prédire le comportement du système dans le futur et agir avant que la situation ne se produise complété avec une adaptation réactive qui doit prendre une décision en temps réel.

Par exemple, [Sekkal *et al.*, 2018] proposent un framework pour les services intelligents qui s'appuie sur une approche réactive et proactive pour traiter le contexte et ses aspects temporels. L'aspect temporel est exprimé par une méthode hybride proactive-réactive. L'approche exploite la technologie sémantique et le concept de contexte, pour permettre au service web intelligent de prendre la décision la plus efficace afin de s'adapter en fonction du changement de contexte.

Les travaux de [Machado *et al.*, 2017] visent à déterminer si une approche soutenue par la technologie du Web sémantique permet de combiner des caractéristiques réactives, proactives et extensibles dans le contexte des systèmes sensibles aux situations. Un réseau d'ontologies est utilisé pour soutenir ces caractéristiques et permettre l'intégration d'applications pervasives pour aider et faire face à différentes situations. Un modèle de référence s'appuyant sur raisonnement avec incertitude en utilisant les technologies du Web sémantique est proposé pour les systèmes prédictifs sensibles aux situations.

Cependant, ces travaux abordant l'adaptabilité ignorent la modélisation du concept « contexte ». Or les approches récentes considèrent la sensibilité contextuelle comme une dimension explicite de la conception à traiter avec des artefacts de conception spécifiques. Ainsi, nous nous intéressons, dans la sous-section suivante, à la question de représentation contextuelle avec une orientation utilisateur final.

2.2.3 Approches de modélisation contextuelle

La perspective de modélisation contextuelle couvre la représentation des propriétés du contexte à travers des modèles contextuels formalisés et la définition des abstractions de modélisation de haut niveau pour la conception des comportements adaptatifs.

[Cassani *et al.*, 2016] soulignent le rôle donné au contexte pour l'identification des ressources les plus adéquates qui peuvent satisfaire les besoins situationnels des utilisateurs et l'adaptation en temps réel des données fournies. Ils représentent des abstractions basées sur le contexte pour générer des modèles spécifiant comment les données renvoyées par les

services sélectionnés doivent être fusionnées et visualisées aux moyens de vues intégrées.

[Daniel et Matera, 2008] proposent un framework pour le développement d'applications Web contextuelles qui vise à masquer la complexité de la conception de l'adaptabilité tout en favorisant la réutilisation. L'approche s'appuie sur le développement à base de composants où les applications contextuelles peuvent être construites en combinant des composants gérant la logique de l'application avec des composants contextuels dédiés. Cependant l'adaptation contextuelle reste limitée au paramétrage des composants spatio-temporels. La plateforme MyService, présentée par [Lee et Joo, 2013] , permet aux développeurs de choisir des règles prédéfinies afin de filtrer les services au moment de l'exécution sur la base du contexte identifié qui est seulement limité à l'emplacement de l'utilisateur.

De manière similaire, [Daniel *et al.*, 2018] adoptent le modèle de dimensions contextuelles pour guider l'accès aux ressources hétérogènes. Les auteurs se sont concentrés sur comment fournir une réécriture automatique des requêtes agnostiques au contexte en requêtes contextuelles par une modélisation explicite du contexte dans un scénario d'application donné. Cette approche de mashup considère les requêtes contextuelles comme des requêtes de mashup qui intègrent à la volée des sources de données sélectionnées. [Casillo *et al.*, 2017] ont opté aussi pour le modèle des dimensions contextuelles afin de développer une solution touristique. L'approche vise à piloter la sélection contextuelle des services provenant des sources hétérogènes pour les intégrer et fournir des recommandations appropriées à l'utilisateur permettant d'accompagner les touristes dans leurs visites.

Ces approches visent donc la collecte et l'agrégation des données en fonction du cadre contextuel modélisé dans la logique de la composition par le biais du modèle des dimensions contextuelles sans que cette adaptabilité puisse être contrôlée par l'utilisateur final. [Alegre *et al.*, 2016] affirment que les systèmes sont efficaces pour l'agrégation des données mais une expertise humaine est néanmoins nécessaire pour valider l'action à effectuer suite à la situation détectée. En effet, [Preuveneers et Novais, 2012] soulignent l'importance d'impliquer activement l'utilisateur final dans le processus d'adaptation et la nécessité d'une représentation explicite du contexte au moyen d'abstractions de haut niveau afin de comprendre comment le contexte influence l'interaction avec l'utilisateur.

Conformément aux approches récentes de programmation visuelle des mashups, CAMUS de [Cassani *et al.*, 2016] se caractérise par un environnement de conception qui utilise des abstractions visuelles. Un module dédié de l'environnement de conception, le gestionnaire de mashups, traduit les actions de composition visuelle des concepteurs en schémas XML qui expriment la logique d'intégration des données en fonction du contexte. Cependant, les auteurs n'étudient pas les questions de convivialité de la création de descripteurs de contexte par les utilisateurs finaux. L'idée est d'offrir une couche visuelle qui permet aux utilisateurs de raisonner à un niveau élevé d'abstraction afin de contrôler le comportement

du système.

Alors que l'outil implémentant cette approche n'est utilisable que par les développeurs non spécialistes de ces paradigmes, [Ghiani *et al.*, 2017] vise à impliquer les experts du domaine et les utilisateurs finaux via un environnement intuitif de création des règles. Il s'agit d'une méthode permettant aux utilisateurs finaux sans expérience en programmation de prendre le contrôle et de créer des solutions adaptées à leurs propres besoins et contextes. Une architecture basée sur un gestionnaire de contexte, capable d'activer, d'interpréter et d'appliquer ces règles aux applications considérées, est aussi proposée. Le but étant de supporter la création et l'exécution dynamique de versions d'applications personnalisées mieux adaptées aux besoins des utilisateurs dans des contextes spécifiques d'utilisation. Selon une étude d'utilisation menée par ces auteurs, ils montrent que les utilisateurs finaux trouvent les règles de déclenchement utiles à appliquer dans différents contextes (les exploiter comme point de départ pour créer de nouvelles règles dépendant du contexte ou les partager avec d'autres utilisateurs).

En plus, pour chaque domaine d'applications un modèle de contexte spécifique est défini. A partir d'un modèle générique décrivant des classes d'objets contextuels standards, une instanciation d'un sous-ensemble de ces classes est générée déterminant le modèle de contexte spécifique. La représentation contextuelle combinant un modèle générique et un modèle spécifique conduit à une structure complexe. Des difficultés sont engendrées dans la navigation des attributs contextuels en raison de la hiérarchie profonde du modèle.

Dans ce qui suit nous faisons une projection sur le domaine touristique afin de développer un panorama complet des systèmes de recommandation pour la planification de séjours touristiques qui s'affichent la capacité de s'adapter à l'utilisation de l'information contextuelle offrant un niveau de satisfaction et d'expérience utilisateur plus élevés en étant plus intelligents, adaptatifs et automatisables. Ceci étant valable dans d'autres applications comme la navigation cartographique, la gestion de la flotte, les renseignements météorologiques, l'assistance routière et les services de localisation personnelle et les systèmes de secours.

2.2.4 Systèmes de recommandation sensibles au contexte dans le tourisme

L'intégration de l'information contextuelle sur les préférences utilisateurs, la position, les besoins de l'utilisateur et les caractéristiques environnementales dans le processus de prestation de services permet de fournir à l'utilisateur des services plus pertinents, et mieux adaptés à ses besoins parmi un nombre important de services disponibles. Par exemple, un jardin botanique se verrait offrir par un système de recommandation une valeur plus élevée les jours ensoleillés plutôt que les jours pluvieux d'hiver. L'objectif de ces systèmes

de recommandation n'est plus de recommander des éléments pertinents en fonction du profil ou des préférences de l'utilisateur, mais de fournir des recommandations "adaptées" à une personne "spécifique", au moment "parfait", à l'endroit "approprié" en fonction de son état émotionnel, son activité actuelle et les conditions circonstancielles. Ainsi, les approches respectant le contexte portent généralement sur quatre facteurs, à savoir l'influence géographique, les modèles temporels, les corrélations sociales et les indications du contenu.

2.2.4.1 Sensibilité géographique

L'influence géographique joue le rôle le plus important dans la recommandation des lieux d'intérêt vu la nature de ce domaine où la mobilité humaine dans le monde physique est limitée. Par exemple, les personnes aimeraient visiter des endroits d'intérêt à proximité suivant le principe "Tout est lié, mais les endroits proches sont plus liés que les distants" [Zhao *et al.*, 2016a].

Les services basés sur la localisation, exploitant la position de l'utilisateur comme principale information contextuelle, sont devenus de plus en plus populaires grâce à la disponibilité d'appareils mobiles puissants équipés de systèmes de positionnement comme le GPS. En fait, les systèmes analysent la position GPS des utilisateurs afin de proposer des lieux suffisamment proches de leur position actuelle ou ayant une forte relation avec les trajectoires GPS qui modélisent leurs expériences de voyage [Sassi *et al.*, 2017].

En outre, [Viktoratos *et al.*, 2015] combinent les services d'information de localisation et le Web sémantique afin de proposer aux utilisateurs des offres contextualisées. Leur système, appelé GeoSPLIS , permet de mettre en relation les propriétaires des lieux et les clients potentiels via un modèle d'adaptation flexible des offres proposées. L'approche présentée s'appuie sur un raisonnement à base de règles pour la recommandation des lieux d'intérêt. La première phase de cette approche consiste à collecter les données auprès des sources externes telles que Google Places API et les sites Web des lieux d'intérêt. Le modèle schema.org est adopté pour la représentation des lieux. Afin d'offrir aux utilisateurs la possibilité d'ajouter leurs propres préférences et les combiner avec des offres ciblées des propriétaires des lieux, un éditeur Web de règles est proposé. Ces règles sont traduites dans le langage RuleML [Viktoratos *et al.*, 2014a] puis transformées dans le format Jess pour qu'elles puissent être compréhensibles par la machine. Suite à l'évaluation à la volée de l'ensemble des règles sur les données modélisant les lieux d'intérêt par les triplets RDF, les différents lieux vérifiant ces règles sont affichés sur une carte Google MAP.

2.2.4.2 Sensibilité temporelle

L'information temporelle est un facteur contextuel aussi important à incorporer dans le processus de recommandation touristique. Cette dimension temporelle inclut l'information sur l'heure du jour (matin, soir ou nuit), le jour de la semaine (jour ouvrable, jour de congé ou week-end) et sur la durée de la visite.

Deux modèles temporels importants des activités humaines sont définis : la périodicité et la séquence temporelle. Considérant que les utilisateurs ont tendance à visiter les lieux à des moments différents, [Yuan *et al.*, 2013] ont injecté les informations temporelles dans un algorithme de recommandation de filtrage collaboratif guidé par l'utilisateur. Ils proposent de calibrer les similitudes entre les utilisateurs avec leurs enregistrements dans les 24 intervalles. [Gao *et al.*, 2013] ont étudié les tendances cycliques temporelles en termes de non-uniformité temporelle et de consécutivité temporelle à travers un modèle de factorisation matricielle. Ils se basent sur des fonctions objectives pour les 24 intervalles au lieu d'une fonction objective globale. En plus, [Zhang *et al.*, 2016] ont étudié la différence entre les jours de la semaine pour adapter leurs recommandations.

D'autres approches telles que celles proposées par [He *et al.*, 2016b] ont montré que la fusion du contexte spatial et temporel peut améliorer l'efficacité des recommandations par l'adoption de schémas chronologiques séquentiels de recommandation d'itinéraire.

2.2.4.3 Sensibilité spatio-temporelle

Selon [Shukla *et al.*, 2017], pour générer les recommandations de séquences de lieux d'intérêt, les caractéristiques spatiales et temporelles du comportement humain doivent être prises en compte conjointement. [Cai *et al.*, 2018] proposent d'intégrer l'effet spatio-temporel dans le modèle de factorisation matricielle pondérée combiné avec le modèle de classement pour la recommandation des lieux d'intérêt. De même, [Zhao *et al.*, 2016b] présentent une méthode de classement spatio-temporel (STELLAR) pour modéliser les interactions entre les utilisateurs, les lieux d'intérêt et le temps. Le modèle STELLAR s'appuie sur un cadre de factorisation des tenseurs par paires basé sur le classement avec une modélisation fine des interactions utilisateur-POI, POI-time et POI-POI pour les recommandations de point d'intérêt successifs.

Similairement, [Nurbakova *et al.*, 2016, Nurbakova *et al.*, 2017] ont essayé de répondre à la question "Comment organiser les activités en une séquence qui maximise la satisfaction de l'utilisateur tout en tenant compte des contraintes spatio-temporelles et de la nature séquentielle des activités ?". L'approche appelée ANASTASIA se compose de trois modules : un module d'extraction des motifs comportementaux typiques des utilisateurs en se basant sur l'historique des activités. Afin de modéliser les séquences, les auteurs se

fient à un modèle probabiliste pour construire un graphe de transition entre activités et un autre graphe de transition à l'échelle catégorie. Le module de calcul des scores des activités vise à mesurer la pertinence d'une activité donnée par rapport aux centres d'intérêts d'un utilisateur tout en considérant la dimension spatio-temporelle. Bien que cette approche a étudié l'influence de différentes dimensions pour recommander un itinéraire personnalisé, elle n'a pas considéré la dimension météorologique qui est importante pour les événements distribués à durée limitée. De plus, l'approche ne mets pas l'utilisateur au centre du processus de personnalisation, il est considéré comme un consommateur de l'itinéraire conçu alors qu'il est acteur de ce dernier.

2.2.4.4 Sensibilité au contenu

Cette caractéristique se voit aussi importante pour une recommandation efficace des lieux d'intérêt. Puisque deux personnes demandant la même information au même endroit reçoivent la même réponse malgré leurs préférences qui peuvent être différentes. [He *et al.*, 2017] exploitent la corrélation catégorielle entre les utilisateurs et les points d'intérêt pour évaluer la popularité du lieu dans la catégorie correspondante.

Le problème de recommandation d'itinéraire se fie également aux données collectées auprès des médias sociaux qui reflètent des itinéraires réels dans plusieurs villes du monde. Par exemple, [Lim, 2015] proposent l'algorithme TOURRECINT pour recommander des circuits définis dans une catégorie de lieu donnée. Cet algorithme implémente le problème d'optimisation " Orienteering Problem " avec des contraintes saptio-temporelles et relatives à la catégorie d'intérêt obligatoire. La catégorie obligatoire est définie comme étant la catégorie la plus fréquemment visitée d'après l'historique des visites du touriste. Cependant cette solution algorithmique n'a pas été intégrée dans une application web pratique. Similairement, [Taylor *et al.*, 2018] définissent le problème TourMustSee qui incorpore l'ensemble des lieux incontournables à ne pas manquer dans l'itinéraires recommandé en tant que variante du problème d'optimisation Orienteering Problem en considérant les temps de déplacement entre les lieux et la durée des visites. Toutefois cette approche est limitée à une tournée d'un jour et des catégories de lieux réduites.

De même, [Brilhante *et al.*, 2015] s'adresse au problème de planification de visite sur la base de l'album photo Flickr à travers leur système TripBuilder. Mais les auteurs adoptent une formalisation différente en s'appuyant sur le problème de la couverture maximale généralisée. Le modèle TripCover exploite les catégories de Wikipédia associées aux lieux en tenant compte de la popularité des préférences réelles des utilisateurs relatives à ces catégorie. TripBuilder fournit des visites guidées en composant des trajectoires populaires reflétant le comportement de vrais touristes. Mais Wikipédia seul peut ne pas être une source appropriée pour collecter les points d'intérêt, car des endroits potentiellement in-

téressants peuvent ne pas être trouvés, surtout si ces endroits n'ont pas de noms bien identifiés, tels que des places d'art, ou des endroits qui offrent de magnifiques points de vue sur les lieux.

Dans cette perception, [Gavalas *et al.*, 2017] suggèrent d'ajouter les itinéraires pédestres à caractère architectural, naturel et panoramique dans la recommandation de visite. La matérialisation de cette approche, est réalisée par l'utilisation de la méta-heuristique de recherche locale itérative implémentant une variante du problème Orienteering Problem. Cette approche algorithmique est appliquée dans Scenic Athens, un guide mobile contextuel pour Athènes en Grèce qui fournit des services de planification de circuits touristiques.

Bien que la contrainte financière soit influente dans le domaine touristique, celle-ci n'a pas été considérée par les systèmes TripBuilder et Scenic Athens. Cette contrainte est prise en compte par [Xie *et al.*, 2011] qui ont proposé leur système CompRec de recommandation de séquence de lieux d'intérêt. Le modèle combine à la fois le score associé à un lieu reflétant son intérêt et le coût engendré que l'utilisateur peut le contraindre par un budget maximum.

2.2.4.5 Sensibilité à l'environnement

En plus des caractéristiques des lieux telle que l'effet spatio-temporelle et le contenu, le domaine touristique est particulièrement sensible à l'environnement physique notamment les conditions météorologiques qui influencent beaucoup la décision des visiteurs [Braunhofer *et al.*, 2013]. Cependant, le climat et les conditions météorologiques ont été peu exploités dans les systèmes de recommandation adaptés au contexte.

L'utilité du contexte météorologique a été étudiée par [Trattner *et al.*, 2016]. Leur approche examine dans quelle mesure les caractéristiques météorologiques influencent le comportement des visiteurs et comment ces caractéristiques fonctionnent dans le contexte d'un système de recommandation des points d'intérêt. Les auteurs suggèrent un algorithme de recommandation et démontrent que le contexte météorologique peut augmenter de façon significative l'exactitude de la recommandation d'un POI. Parmi le peu de travaux qui se sont orientés vers cet aspect, [Braunhofer *et al.*, 2014] présentent leur système mobile sensible au contexte STS (South Tyrol Suggests) de recommandation des lieux d'intérêt au Tyrol du Sud en Italie. Ce système incorpore différents facteurs contextuels relatifs aux conditions météorologiques dans le modèle de recommandation à savoir la température, la saison, le temps, etc afin d'affiner les suggestions. Néanmoins, le système souffre du problème de complétude des données contextuelles collectées où 89% des valeurs sont des valeurs inconnues.

Dans un esprit similaire, [Arigi *et al.*, 2018] incluent dans leur modèle ontologique la dimension environnementale avec trois valeurs : " Beau temps ", " Mauvais temps ", " Très

mauvais temps " . Le système est constitué de deux phases. Premièrement, une recherche des destinations touristiques est effectuée en considération des contraintes temporelles de chaque destination individuelle. Puis une évaluation de chaque destination sélectionnée est réalisée sur la base d'une valeur d'utilité qui est calculée en fonction de la distance, du poids de la valeur météorologique autour de la destination et des préférences de l'utilisateur.

2.2.4.6 Sensibilité multiple

Plusieurs systèmes de recommandation suggèrent d'agréger différentes informations contextuelles en raison de la corrélation entre les facteurs contextuels [Braunhofer et Ricci, 2017], par exemple, la catégorie du lieu, la distance et le temps de déplacement peuvent influencer la décision du visiteur. En effet, [Baral et Li, 2016] combinent l'aspect spatio-temporel, catégoriel et social dans un modèle de contexte unifié pour prédire les lieux potentiellement intéressants sur la base des réseaux sociaux géolocalisés. Le système MAPS (Multi Aspect Personalized POI Recommender System) analyse les données des « check-in » en fonction des aspects catégoriels, sociaux, spatiaux et temporels pour fusionner ces aspects en un seul modèle. Le problème de recommandation est modélisé sous la forme d'un graphe de points d'intérêt avec des contraintes limitant les distances selon la catégorie.

[Baltrunas *et al.*, 2012] proposent une estimation de la dépendance des préférences de l'utilisateur à partir d'un ensemble initial de facteurs contextuels. Un outil Web est développé pour demander aux utilisateurs d'évaluer si une condition contextuelle particulière ("à la compagne", "il fait froid", "il pleut") avait une influence positive, négative ou nulle sur l'évaluation d'un type particulier d'élément (restaurant, jardin, musée). En utilisant les résultats de cette évaluation, une sélection des facteurs contextuels les plus importants pour différents types de lieux est déterminée. Ainsi, les facteurs sélectionnés sont utilisés dans le modèle de factorisation matricielle contextuelle pour générer les recommandations.

Ainsi, les systèmes de recommandations à sensibilité multiple cherchent une corrélation entre les différents facteurs contextuels afin d'améliorer les prédictions. Néanmoins, la conception d'un tel système multidimensionnel accumulant de nombreuses données peut se terminer par un algorithme de recommandation complexe. En outre, ce n'est pas parce que le lien entre les éléments contextuels est étroit que la combinaison du plus grand nombre possible d'aspects environnementaux, sociaux, spatio-temporels et autres s'explique. Par exemple, l'influence spatio-temporel a des répercussions plus importantes sur la recommandation touristique que l'influence sociale. L'étude de [Yu et Chen, 2015] a signalé que les approches de recommandation de lieux d'intérêt à sensibilité sociale n'ont pas obtenu d'importantes améliorations du fait que le processus de sélection est influencé par la propriété géographique des lieux. Par conséquent, il est important d'étudier la pertinence des facteurs contextuels, dans un domaine particulier, avant de les injecter dans la procédure

de recommandation.

Ainsi les informations contextuelles récoltées automatiquement (la date, la localisation, ...) ou en interrogeant l'utilisateur (le budget, l'accompagnateur, ...) permettent de définir le modèle de recommandation qui satisfait la demande de l'utilisateur. Cependant, ces données sont multiples, hétérogènes, changeantes, voir contradictoires, et doivent être comprises grâce aux interactions homme-machine adéquates. C'est pourquoi, l'enjeu suivant dans la conception de systèmes de recommandation tenant compte du contexte consiste à intégrer une couche d'interaction assurant la personnalisation des recommandations. La sensibilité au contexte est un premier pas vers des techniques plus avancées visant la personnalisation et l'intelligence du service.

2.3 Composition de service personnalisée

Bien que les approches adaptables offrent une composition de services appropriée en tenant en compte de l'information contextuelle, le nombre de solutions reste large. Cela augmentera la difficulté de la sélection des solutions de service et réduira l'efficacité de la composition des services. Ainsi, une technique de personnalisation, permettant de rendre la composition plus efficace, se voit nécessaire. Comme nous l'avons illustré précédemment, les deux techniques de personnalisation utilisées sont la configuration et la recommandation. Nous adoptons la technique de configuration dans cette thèse.

Cette technique est perçue selon trois vues : la facette produit qui a été exploité par les industries modernes dans le contexte de la personnalisation de masse pour relever les défis d'exigences individuelles croissantes. La deuxième facette concerne le processus où la grande idée était d'étendre la technologie de configuration pour couvrir la gestion de l'ensemble du produit personnalisable. Puis ce concept a été emprunté par les service Web.

2.3.1 Vision produit

D'une manière générale, la tâche de configuration de produit consiste à assembler un ensemble de composants et établir des connexions (si possible) entre ces composants pour satisfaire les exigences individuelles d'un client sans violer les contraintes. [Wang *et al.*, 2017]. Les études sur les systèmes de configuration de produits peuvent remonter au début des années 1980, lorsque Barker et O'Connor ont mis au point en 1989 le premier système de configuration appelé R1, qui utilisait une méthode de configuration basée sur des règles.

Le problème de configuration peut être considéré comme un problème de satisfaction de contraintes où il est décrit en terme de composants et de connexions entre les composants à travers la notion de ports [Fargier *et al.*, 2016]. [Yang et Dong, 2013] caractérisent

le problème de configuration par des règles de configuration basées sur la cardinalité. Des règles d'inclusion sont utilisées pour décrire des cardinalités et des hiérarchies de structure. Ensuite, le problème de configuration est encodé en problème de satisfaction de contraintes pour la résolution. Un encodage des modèles de configuration dans un problème de satisfaction de contraintes dynamique est suggéré par [Yang *et al.*, 2012]. Ils traitent les contraintes de configuration telles que les contraintes de réquisition et d'exclusion (des contraintes de compatibilité), ce qui permet à un composant de bas niveau de se joindre au processus de résolution seulement après que son composant de haut niveau est sélectionné dans la configuration.

Dans les approches présentées ci-dessus, des représentations séparées sont utilisées pour la modélisation et la résolution de processus. La différence avec l'approche que nous proposons, si nous considérons les services comme des objets, c'est que les services et les contraintes de l'utilisateur sont spécifiés et traités de la même manière en utilisant le même modèle. Le processus de configuration consiste cependant à combiner les services existants. Cela signifie que les services peuvent être ajoutés ou supprimés à tout moment, ce qui est utile pour les services Web, qui sont des entités généralement gérées par des environnements dynamiques.

2.3.2 Vision processus métier

La perspective processus de la configuration porte sur les aspects de contrôle de flux. Après avoir été considéré comme un ensemble de composants dans une "vue physique", la configuration s'étend pour couvrir l'ensemble du cycle de production. L'idée des modèles de processus configurables a été développée par [Gottschalk *et al.*, 2009] pour aligner les options de variation de processus largement standardisés avec des variations. Les modèles de processus configurables intègrent plusieurs variantes de processus dans un seul modèle adaptable aux besoins individuels par la désactivation de toutes les parties inutiles du processus. Le défi recherché avec cette perspective est la gestion de la variabilité afin d'offrir un processus paramétrable en fonction des cas proposés. [Hallerbach *et al.*, 2010] présentent l'approche Provop en utilisant des informations contextuelles dans la configuration de processus. Sur la base des faits, diverses options de configuration sont sélectionnées. En plus d'établir les faits directement, Provop offre la possibilité de raisonner sur les faits.

Une synthèse des approches de modélisation de processus configurables est présentée par [Rosa *et al.*, 2017] où un modèle de processus personnalisable est représenté par une famille de variantes de processus de sorte qu'un modèle de chaque variante est généré par dérivation en ajoutant ou en supprimant des fragments selon les options de personnalisation. Les auteurs concluent que toutes les approches prennent comme point de départ un langage de modélisation de processus et y ajoutent une notion de point de variation pour générer

un processus personnalisable. Ces dernières peuvent se regrouper en deux catégories : la personnalisation par restriction où un modèle de processus personnalisable qui contient toutes les variantes de processus sera configuré par la désactivation de certaines branches. Et la personnalisation par extension qui, au contraire représente le comportement le plus courant, le plus partagé par la plupart des variantes de processus et la personnalisation se fait par l'ajout d'activité.

Toutefois, la principale limite de ces approches dans cette perspective est le manque de méthodes et d'outils interactifs pour aider les utilisateurs dans la création du processus personnalisable.

2.3.3 Vision service Web

La technique de configuration de composants réutilisables peut être directement appliquée à la problématique des services Web. [Teije *et al.*, 2004] décrivent une approche de configuration de services où un service Web complexe est représenté comme un modèle fixe qui doit être configuré pour chaque utilisation spécifique. Également [McIlraith et Son, 2002] proposent une approche pour construire des services composites basés sur la notion de procédures génériques et l'adaptation des contraintes de l'utilisateur. Les auteurs affirment qu'une version augmentée du langage de programmation logique Golog fournit un formalisme naturel pour composer automatiquement des services sémantiques. Ils suggèrent une personnalisation des procédures génériques hautement réutilisables mais pas dans une perception utilisateur final. Cette dernière est discutée par [Sheng *et al.*, 2009] à travers leur système de composition configurable et fourniture adaptative de services composites. Les auteurs se basent sur le concept de schéma de processus qui fournit une infrastructure système pour l'approvisionnement distribué, adaptatif et contextuel de services Web composites. Les utilisateurs finaux ont la possibilité de personnaliser ce schéma de processus en assignant les contraintes contextuelles au schéma. Un modèle d'exécution de services événementiel fournit la sémantique d'exécution des services composites adaptatifs. L'avantage de cette approche est l'indépendance des connaissances où le modèle événementiel d'adaptation au contexte, au moyen des règles de type condition-action est séparé de la spécification des services Web composites. Afin d'offrir plus de flexibilité aux développeurs pour spécifier leurs besoins en termes de qualités de service, [Imed et Graiet, 2017] ont mis au point un algorithme de configuration de services composites permettant la construction et l'adaptation de services dans une procédure de calcul récursive.

Les objets intelligents peuvent en effet être accessibles en tant que services, car ils sont souvent fournis avec un URI qui les identifie sur Internet et sont publiés en exploitant des technologies de services Ceci permet la capture des événements et des actions à distance. Ainsi, les nouvelles extensions privilégient principalement la synchronisation événementielle

au moyen de règles de type Evénement-Conditio-Action, une technique de composition expérimentée pour la composition d'API Web par des approches de mashup léger . Une orchestration afin de créer de la valeur ajoutée pour l'utilisateur final conduit donc à de nouveaux défis pour les techniques de personnalisation où il s'agit de les orienter vers les utilisateurs novices en adoptant des représentations intuitives.

En l'occurrence, [Mayer *et al.*, 2016] présentent une approche permettant aux utilisateurs finaux de configurer leur environnement intelligent par le biais d'un éditeur graphique intuitif. À l'aide d'un appareil mobile, les utilisateurs spécifient leurs préférences telles que la chanson ou la station de radio à jouer dans leur environnement et leur température de confort, règlent les alarmes ambiantes dans l'environnement ou ajustent l'éclairage en fonction de leurs niveaux préférés. Cet appareil négocie ensuite avec l'environnement pour ajuster les paramètres spécifiés aux réglages de confort de l'utilisateur en s'appuyant sur un raisonnement sémantique qui crée de manière transparente des mashups de services. Par exemple, pour lire des chansons d'un genre musical relatif à l'emplacement actuel de l'utilisateur, plusieurs services doivent coopérer : un système de localisation est utilisé pour localiser l'utilisateur et découvrir un lecteur multimédia à proximité ; un autre service est chargé de trouver des fichiers audio lisibles de chansons correspondant au genre donné et le lecteur multimédia découvert doit être configuré pour lire ces fichiers.

[Cabitza *et al.*, 2017] proposent une étude comparative des outils les plus utilisés pour la configuration d'environnements intelligents. Cette étude a démontré que Atooma est plus avantageux que IFTTT où il fournit plus d'options notamment la combinaison de conditions et d'actions dans une règle évitant que l'utilisateur fasse des tâches mécaniques inutiles.

2.3.4 Approches de personnalisation interactives

Bien que l'objectif des systèmes de recommandation soit la proposition des suggestions les plus appropriées aux utilisateurs, ceci peut conduire à la sur-adapation et influencer négativement l'experience utilisateur. La diversification se présente comme une solution du problème de sur-adaptation ainsi qu'une technique d'améliorer la qualité de l'experience utilisateur [Kunaver et Požrl, 2017]. En effet, elle exige une implication humaine à travers une approche interactive afin d'unifier la sémantique de la diversité. Cela est dû au fait que la diversité prévue n'est pas directement corrélée à la diversité perçue.

Chaque utilisateur appréhende la diversité différemment. [He *et al.*, 2016a] montrent que la visualisation des recommandations dans des catégories plutôt que dans une liste améliore la perception de la diversité des recommandations par les utilisateurs et a un effet positif sur leur acceptation.

[Ziegler *et al.*, 2005] utilisent la mesure intra-liste dans une approche de diversification des éléments recommandés afin de refléter l'éventail complet des intérêts de l'utilisateur. La liste des recommandations personnalisées suggérées établit un équilibre entre l'exactitude et la diversité en se reposant sur le mapping des éléments avec les taxonomies. Les auteurs montrent qu'il devient indispensable d'aller au-delà de la simple précision, puisque celle-ci ne dit pas tout, pour regarder les autres aspects liés aux systèmes de recommandation personnalisés notamment la diversité qui reflète une réelle experience utilisateur.

Une autre formulation de la diversité sous forme de problème de couverture est aussi adoptée dans la littérature visant à étendre le spectre des recommandations par la réduction de l'homogénéité des occurrences.

Une approche de représentativité de type un à la fois qui résume séparément chaque collection d'éléments homogènes. En l'occurrence, [Jaffe *et al.*, 2006] résument les points d'inétrêt représentatifs d'une ville sur une catre en se basant sur une large collection des photos géo-référencées appelés Tag Map. Cependant cette technique traite des collection d'éléments homogènes alors que, les visiteurs d'une ville seraient intéressés par une collection d'éléments hétérogènes où différents types de points d'intérêt (musée, événement, restaurant ...) sont proposés. La technique d'un seul type à la fois ne garantit pas nécessairement que chaque zone de la carte contiendra des éléments représentatifs de chaque type.

Par conséquent, une approche en deux étapes est proposée par [Brilhante *et al.*, 2013] produisant un plan de visite personnalisé qui satisfait les préférences individuelles des visiteurs selon un processus qui vérifie d'abord la satisfaction des contraintes puis optimise la solution. Ce processus limite la représentativité des recommandations, car il suppose que l'ensemble généré par la première phase est très diversifié et qu'il suffit de l'optimiser pour obtenir l'effet maximal possible.

Ainsi, [Leroy *et al.*, 2015] proposent une approche d'intégration qui offre un compromis entre la validité et la représentativité. L'algorithme de classification flou utilisé intègre la satisfaction des contraintes afin d'optimiser simultanément la validité et la représentativité des recommandations. Il repose sur deux fonctions objectives couramment exploitées : la distance entre les points d'intérêt pour une meilleure couverture et la similitude pour la représentation des suggestions.

Les recherches dans cette direction se concentrent sur divers facteurs humains qui influent sur l'acceptation des recommandations, comme la satisfaction des utilisateurs, la confiance, la transparence et le sens du contrôle. Ils s'investissent dans la combinaison de modèles de recommandation avec des techniques de visualisation pour alimenter le processus de recommandation d'un cadre interactif améliorant le niveau de la personnalisation. Nous nous focalisons, dans une perspective utilisateur, sur les critères de *transparence* et

du *sens du contrôle* au cours de cette revue des systèmes de recommandation personnalisés.

2.3.4.1 Transparence de la personnalisation

Les applications de recommandations s'affichant comme " boite noire ", avec aucune indication sur la logique de recommandation, empêchent les utilisateurs d'interagir avec le processus de recommandation. La transparence traite la nature dissimulée des systèmes de recommandation actuels en expliquant la logique interne du système aux utilisateurs finaux. Cette ouverture permettra la compréhension de la raison d'être des recommandations afin d'offrir aux utilisateurs la possibilité de piloter le processus de recommandation ce qui conduira à mieux satisfaire les demandes de l'utilisateur et accroître sa confiance.

Pour se faire, des visualisations interactives ont été déployées dans les systèmes de recommandation afin d'offrir un degré plus élevé de transparence. Par exemple, [Jin *et al.*, 2016] fournissent aux utilisateurs une mesure d'inspecter graphiquement les recommandations proposées. PARIS, leur système de recommandation, permet de visualiser quelles informations sont utilisées dans le processus. Lorsque les utilisateurs modifient leur profil, un organigramme visualise les facteurs qui influencent la sélection dans le but de montrer l'impact de ces informations sur le processus de recommandation.

D'autres systèmes ne dévoilent pas la logique interne de l'algorithme de recommandation mais donne une vue abstraite du processus. Ils décrivent juste la raison derrière les recommandations proposées sans indiquer comment le système fonctionne ; autrement ils ne font que justifier les recommandations. En l'occurrence, Amazon justifie les suggestions proposées en expliquant qu'il s'appuie sur l'historique des produits achetés dans sa logique de recommandation.

Dans certains cas, se limiter à la justification de recommandation se voit préférable à la transparence totale à cause de la complexité du système avec l'injection de l'information contextuelle dans le processus.

Cependant ne faire que justifier les recommandations ne permet pas d'étayer un bon degré de satisfaction de l'utilisateur. Une approche hybride assurant un bon compromis entre la transparence et la justification et faisant interagir les utilisateurs avec le système est nécessaire. La participation des utilisateurs dans le processus de sélection de services de manière interactive et itérative fournit un service personnalisé satisfaisant. Nous nous plaçons dans ce cas de figure avec notre approche de configuration que nous détaillons dans le chapitre suivant.

2.3.4.2 Sens du contrôle de la personnalisation

La contrôlabilité renforce la participation de l'utilisateur dans le processus de recommandation. Le sens du contrôle de l'application de recommandation peut avoir plusieurs facettes ; l'introduction des préférences et contraintes utilisateur, l'ajustement des données sur les préférences, la révision ou l'exploration des recommandations et l'intégration des commentaires et retours sur les suggestions.

CT-Planner est un système de recommandation, implémenté par [Kurata et Hara, 2013], proposant des plans de visites qui sont affinés au fur et à mesure que l'utilisateur exprime ses préférences et ses contraintes (durée, réticence à marcher, etc.). L'utilisateur est autorisé à modifier les visites guidées dérivées, à supprimer les points d'intérêt indésirables et/ou à ajuster l'heure de visite programmée pour des points d'intérêt particuliers. Cependant ces modifications mettent en cause toute la visite proposée et déclenche un calcul du nouveau plan satisfaisant les nouvelles préférences, ce qui engendre un coût supplémentaire.

[Yahi *et al.*, 2015] présentent leur système de planification de tournés pédestres Aurigo, faisant participer les touristes dans le processus de recommandation via des visualisations interactives. Les auteurs proposent une approche combinant les recommandations guidées par les données, une visualisation interactive et des fonctions de personnalisation hautement paramétrables afin d'aider le visiteur à construire rapidement ses itinéraires individualisés étape par étape. A travers cette technique interactive et incrémentale, l'utilisateur est placé au centre du cercle de décision pour piloter le processus de recommandation personnalisée ce qui entraîne un contrôle total de la recommandation. Contrairement à Aurigo, notre approche supporte en plus la sensibilité au contexte qui n'est pas respectée par ce système.

Dans le même esprit, [Roy *et al.*, 2011] formalisent la recommandation d'itinéraires interactive comme un processus itératif. A chaque étape, l'utilisateur fournit un retour sur les lieux proposés puis le système recommande les meilleurs itinéraires sur la base des rétroactions obtenues et suggère un nouvel ensemble pour solliciter un retour pour la prochaine étape. Alors que Aurigo essaie de construire l'itinéraire point d'intérêt par point d'intérêt, [Roy *et al.*, 2011] proposent de naviguer dans l'ensemble d'itinéraires généré qui commence avec toutes les instances valides possibles ensuite rétrécie itérativement en suggérant les lieux par lots basés sur les scores les plus élevés de l'itinéraire prévu. Dans notre approche, une adaptation de l'itinéraire touristique est réalisée par reconfiguration du plan de visite où l'utilisateur pourra remplacer les sites non désirés.

[Singh *et al.*, 2017] étendent leur notion de "Composite Item" par l'aspect d'interactivité pour construire des itinéraires personnalisés. Les auteurs ont étudié l'avantage de l'interactivité dans la personnalisation de forfaits composites de points d'intérêt. Cette dernière offre une flexibilité pour atteindre un équilibre entre la personnalisation et la

cohésion des lieux d'intérêt. A travers un ensemble d'opérations, les utilisateurs peuvent ajouter, supprimer ou remplacer certains lieux d'intérêt afin de personnaliser le plan de visite. Seulement, cette interactivité est légère, faisant participer le visiteur qu'à la phase de raffinement du plan de visite. En plus, elle est limitée auquel cas le système ne regarde que la proximité géographique sans inclure les autres dimensions contextuelles comme nous proposons dans notre approche de personnalisation.

2.4 Synthèse

Après avoir fait un tour d'horizon des approches dans les thématiques de mashup, de sensibilité au contexte et de personnalisation avec une orientation utilisateur final, nous dessinons dans cette section le bilan suivant. L'analyse de l'échantillon de plateformes de mashup a montré qu'ils présentent tous des limites similaires et qu'aucune des initiatives n'est en mesure de permettre aux utilisateurs finaux de développer leur propre application agrégateur de données personnalisée. Ceci est dû principalement au fait que la plupart des solutions proposées demandent une manipulation du mapping des données qui nécessite des connaissances techniques par les utilisateurs. Ceci est loin du modèle mental de l'utilisateur final qui n'est pas nécessairement un développeur. Ainsi, forcer les utilisateurs à penser en termes de contrôle de flux de données n'est pas naturel [Daniel et Matera, 2014]. Par conséquent, la notion de service en tant qu'entité productrice et consommatrice de données n'est pas évidente pour les novices. Ce qui nous a conduit à éliminer la branche de l'exploration des processus métier de la revue de la littérature puisque les approches de composition dans cette direction se concentrent sur le flux de contrôle. De ce fait, nous proposons dans notre approche de s'abstraire des détails techniques et de représenter les ressources sous forme de composants avec des propriétés à configurer.

En outre, bien que la quasi-totalité des approches de mashup adoptent des métaphores visuelles pour faciliter le développement, les utilisateurs novices ont besoin que ces dernières soient complétées de mécanismes d'exécution immédiate. Le paradigme visuel et interactif WYSIWYG caractérisé par le mélange de la conception et de l'exécution de mashups est un bon exemple de paradigmes supportant une évaluation progressive permettant de les guider dans le processus de composition. En effet, les utilisateurs finaux ne sont pas en mesure de faire la distinction entre la phase de conception et la phase d'exécution. Par conséquent, dès la toute première action de composition, l'utilisateur devrait être en mesure de voir une application en cours d'exécution et d'observer progressivement l'effet des actions de composition.

Même avec l'exploitation des métaphores visuelles, les initiatives de mashup ne sont pas tout à fait adaptées aux besoins de l'utilisateur final vu qu'elles exposent encore des

concepts de programmation qui ont une sémantique non compréhensible par les utilisateurs et en raison de la complexité de leurs paradigmes de composition.Ce problème est essentiellement dû au fait que les plateformes de mashup exposent trop de fonctionnalités et trop de technicités, par conséquent, ces outils ne parlent pas le langage de l'utilisateur. Le caractère générique de ces approches limite les utilisateurs finaux à les adapter aisément. Ainsi, elles doivent être adaptés aux domaines dans lesquels elles seront utilisés. C'est-à-dire limiter le langage au domaine en termes de puissance expressive et fournir une notation spécifique au domaine afin qu'il soit plus facile à utiliser. En particulier, l'utilisateur final connaît mieux son domaine où il est expert alors qu'il n'est pas conscient des aspects de composition et de mapping des données. Par conséquent, nous optons pour une approche spécifique au domaine qui vise à injecter terminologie du domaine au modèle du mashup afin de donner un sens à l'intégration. Nous cherchons en effet à rajouter une couche de spécification au domaine à la couche d'intégration visuelle dans la solution que nous proposons. En outre, nous visons à pousser la simplification encore plus loin par rapport aux approches de mashup actuelles ; qui, bien qu'elles tentent de réduire la complexité par une notation spécifique au domaine, elles demandent aux utilisateurs de tracer le mapping entre les ressources. Ainsi, nous pensons que l'alliance de la puissance de la conception spécifique au domaine et de la représentation visuelle appropriées nous permettra d'élargir encore davantage le spectre des personnes capables à développer des mashups.

D'un autre côté, les approches traitant les domaines à caractère dynamique doivent intégrer la gestion du contexte aux techniques de composition de services. Parmi les approches de sensibilité au contexte, nous nous concentrons sur les approches de modélisation contextuelle. L'objectif étant d'identifier les facteurs contextuels qui influencent le processus de mashup en fonction des spécificités du domaine d'application afin de fournir à l'utilisateur un service adapté à la situation particulière. Il s'agit de donner à l'utilisateur le moyen de personnaliser leur application en fonction de son contexte. En effet, le modèle contextuel repose sur une structure compacte nommée le modèle de dimensions contextuelles formant ainsi un modèle spécifique au domaine qui est configuré à partir d'une classification générique.

Le comportement adaptatif exploite un paradigme déclaratif basé sur un raisonnement événementiel qui a été prouvé adapté au modèle mental de l'utilisateur. En spécifiant leurs règles de contextualisation, les utilisateurs sont en mesure d'obtenir un meilleur support et une plus grande satisfaction dans l'utilisation de leurs applications. Cependant, les approches adoptant ce raisonnement événementiel en automatisant des tâches pour synchroniser le comportement des objets intelligents n'ont pas réussi à mettre en œuvre le compromis entre l'exhaustivité/expression et l'intuitivité/la simplicité. Les outils existants prétendent s'adresser aux utilisateurs non techniques mais ne contribuent réellement qu'à

la création de règles " de base " qui synchronisent un seul événement avec une seule action. Et ceux qui permettent de créer des règles plus expressives, nécessitent des compétences en programmation. A cet égard, nous visons à surpasser les approches actuelles en termes de satisfaction des utilisateurs et leur proposer un moyen de combiner les propriétés contextuelles pour décrire le comportement adaptatif du mashup de manière adéquate à leur modèle mental.

Généralement, les approches de contextualisation sont implémentées en tant que systèmes de recommandation. Ces systèmes de recommandation en tourisme présentés peuvent être organisés selon deux niveaux de granularité. Le premier niveau regroupe les approches de recommandation de Top-k points d'intérêt où l'objectif principale vise à suggérer une liste de lieux d'intérêt classés en fonction de leur pertinence pour un utilisateur. Ces travaux utilisent généralement des approches basées sur diverses adaptations de factorisation matricielle ou d'algorithme de filtrage collaboratif pour la recommandation d'une liste de points d'intérêt. Le deuxième niveau s'intéresse à organiser les différents lieux d'intérêt recommandés sous forme d'itinéraire complet. Plusieurs travaux font appel aux formalismes de la recherche opérationnelle, avec des heuristiques en occurrence la recherche tabu, pour résoudre le problème de recommandation d'itinéraire. Ces approches de type " Filter-First, Tours-Second " cherchent à générer des visites respectant les contraintes et les limitations sur la durée des visites, les budgets et les autres restrictions. Nous nous plaçons dans ce cas de figure sauf que nous allons au-delà de la contextualisation pour permettre une personnalisation de la visite par reconfiguration de composants en exploitant le potentiel des données liées.

De la sorte, notre approche générale orientée utilisateur final vise à réaliser le mashup des bonnes pratiques de ces thématiques pour exploiter les points fort respectifs. Vu qu'aucune solution ne permet aux utilisateurs novices une intégration personnalisée des données. En effet, une solution spécifique au domaine est combinée au mashup visuel en raison de la non efficacité d'une conception générique pour couvrir un large éventail de domaines d'application et suffisamment puissante pour fournir une logique triviale ; en plus de la sensibilité contextuelle et la personnalisation.

Deuxième partie

Approche de configuration de mashup personnalisé

Chapitre 3

Modèle Fonctionnel de Configuration de Mashup

Introduction

Dans le chapitre précédent, nous avons présenté une revue de la littérature dans les domaines de composition de services, de mashup, de développement orienté utilisateur final et de personnalisation. L'analyse des approches les plus connues de ces domaines a montré qu'elles n'ont pas réussi à offrir des solutions pouvant aider les utilisateurs finaux à répondre à leurs besoins complexes d'intégration de données.

D'un coté, nous avons constaté que les approches de composition ne parviennent pas à trouver le juste équilibre entre la facilité d'utilisation et la puissance expressive. Ceci est dû au caractère générique de ces dernières qui limitent aux utilisateurs finaux les possibilités à les adapter aisément. En effet, malgré la définition de certaines abstractions utiles, les outils de mashups restent complexes. Ils exigent aux utilisateurs de réaliser des traitements de modélisation tels que le mapping des données qui ne sont pas simples pour les novices.

De l'autre coté, nous avons remarqué le manque de l'aspect personnalisation au sein des approches de recommandation dans le contexte touristique. Celles-ci ne permettent pas à l'utilisateur de configurer son itinéraire comme il l'entend. La raison réside dans l'écart d'interaction avec l'utilisateur qui réduit son degré d'acceptation. Les utilisateurs qui contribuent à la création des services, plutôt que de simplement suivre les instructions de l'application, ressentiront un sentiment de réussite et continueront à utiliser ce système.

Inspiré par la façon dont les utilisateurs aimeraient être assistés lors du développement de mashup [Daniel *et al.*, 2012], nous présentons dans cette seconde partie du manuscrit notre approche de configuration pour le mashup personnalisé des ressources Web en vue de recommander des séjours touristiques. Elle permet aux utilisateurs de briser les barrières

technologiques d'utilisation d'applications pouvant être complexes en leur permettant en quelques sorte de "parler leurs langues". Cette approche de mashup s'intègre bien dans la représentation cubique des classes de mashup (figure 1.6) où elle se retrouve au coordonnées (Tourisme, Hybride, Web-Mobile). Le premier chapitre de cette partie décrira la méthodologie pour le développement de mashup dérivée de la classe des langages spécifiques au domaine. Le modèle fonctionnel de configuration de mashup implémentant cette méthodologie est ensuite introduit. Il illustre une construction incrémentale du service de mashup par l'agrégation des fonctionnalités proposées par les services élémentaires.

3.1 Méthodologie de configuration de mashup orientée utilisateur

Notre méthodologie de conception de mashup est inspirée de l'approche de [Imran *et al.*, 2012]. Elle reflète la philosophie du *live development* en fournissant des recommandations à la volée à partir d'un raffinement visuel qui guide les utilisateurs dans la procédure de mashup. La définition des objectifs fonctionnels ainsi que la procédure de mashup suit une orientation composant. Les composants, qui peuvent être considérés comme des unités fonctionnelles de base dans la conception architecturale, représentent une encapsulation des services à travers des éléments visuels pour permettant un haut niveau d'abstraction.

Ces derniers doivent être suffisamment génériques pour fonctionner dans une variété de contextes et couvrir plusieurs domaines applicatifs. Ils doivent également être suffisamment puissants pour une utilisation simple et intuitive afin d'être réellement accessibles aux utilisateurs finaux. Concevoir de tels composants n'est pas trivial. Ceci nécessite de sacrifier la généricité au profit de la pratique, c'est-à-dire borner l'objectif du mashup pour l'adapter aux besoins spécifiques au domaine. C'est bien le point de vue que nous partageons avec [Imran *et al.*, 2012]. Notre approche stipule qu'il faudrait gérer les contraintes, sur les services atomiques ainsi que sur la composition, exprimées par l'utilisateur. Ceci est très important car les contraintes globales définies sur le mashup sont efficaces pour la sélection des services. Par exemple, pour le mashup de planification de séjours touristiques qui comprend la réservation d'un hôtel, la réservation d'un restaurant et les différentes activités, une limitation du budget total devrait éliminer la composition des services coûteux. Par contre, si un seul service est coûteux, il peut être inclus dans le mashup de service, à condition que le budget total soit dans les limites exprimées.

Un mashup spécifique au domaine se définit comme un processus composite manipulant les concepts du domaine à travers des activités et des processus respectant les règles du domaine. Il est représenté par le biais d'une notation graphique spécifique au domaine.

Le mashup est décrit donc uniquement avec des composants conformes au modèle de

processus du domaine et la collaboration de ces composants suit le modèle conceptuel. Les composants sont organisés en classes pouvant être par exemple des composants encapsulant des services de données, des composants de filtrage et d'agrégation ou encore de visualisation. Il s'agit en d'autre termes d'une boite à outils de composants. Puisque les composants s'intègrent dans des classes et interagissent sur la base d'un modèle de données commun, il devient plus facile de les combiner pour permettre ainsi de définir des mashups.

Le problème de mashup spécifique au domaine revient donc à fournir les concepts nécessaires à l'élaboration de modèles de mashup. Le problème n'est ni simple ni trivial, nous devons comprendre précisément quelles propriétés de domaine sont nécessaires pour couvrir de manière exhaustive tous les aspects de domaine qui sont nécessaires pour adapter une plate-forme de mashup à un domaine spécifique, quelle propriété entre en jeu, comment les aspects de domaine sont matérialisés (par exemple, visualisés) dans la plateforme de mashup tout en entraînant une charge cognitive minimale.

Nous cherchons à changer l'orientation des outils de mashups, construits autour d'une métaphore informatique, vers des artefacts cognitifs adaptés au langage des utilisateurs. L'interaction réussie et intuitive réside dans la familiarité avec l'interface, son style d'interaction et la métaphore à laquelle elle se conforme.

Le développement de telle plateforme de mashup s'articule autour de trois aspects : *les concepts du domaine*, *les processus et l'implémentation*. Les étapes décrivant la démarche pour le développement de plateformes de mashup spécifique au domaine sont :

1. Définition *d'un modèle des concepts du domaine (MCD)* pour exprimer les données et les relations du domaine. Les concepts sont au cœur de chaque domaine, ils permettent de le caractériser. Il est ainsi important de délimiter les concepts représentatifs du domaine, afin de développer des composants qui les implémentent. La spécification des concepts du domaine permet à la plateforme de mashup de comprendre quel type d'objets elle doit supporter, ce qui n'est pas le cas des mashups génériques. Ces derniers considèrent des formats de données génériques et non des objets spécifiques.

2. Identification *d'un méta-modèle de mashup (MM)* capable de s'adapter aux besoins de composition du domaine. Il s'agit de spécifier la classe de mashup parmi les types de mashup existants qui couvre les trois nivaux d'abstraction d'une application.

3. Étant donné le méta-modèle générique, l'étape suivante consiste à y injecter le domaine afin que toutes les caractéristiques de celui-ci puissent être considérées. On parle de méta-modèle de mashup spécifique. La démarche de spécification du méta-modèle générique comprend :

— La définition d'un modèle de processus qui représente des classes d'activités de domaine et, éventuellement, des processus. Les activités et les processus du domaine expriment l'aspect dynamique du domaine en s'appuyant sur les concepts du domaine. Cette étape consiste à étendre le méta-modèle générique pour considérer la nature des activités du domaine en terme de composants.

— La description de la syntaxe du domaine qui fournit un symbole propre à chaque concept dans le méta-modèle de mashup spécifique au domaine. Il s'agit de garnir les activités du domaine de métaphores visuelles connues indiquant les fonctionnalités exposées.

— Une instanciation des composants fournissant un ensemble de composants concrets conformes au méta-modèle.

4. La dernière étape est la mise en œuvre de l'outil de mashup en tant que plateforme qui exprime le pouvoir du méta-modèle spécifique au domaine et respecte le modèle des concepts.

Comme les domaines évoluent généralement au fil du temps, et pour s'aligner avec des exigences pouvant être changeantes, nous proposons la mise en place de nouveaux composants sans repasser par l'ensemble des étapes de la méthodologie. Nous détaillons dans les sections qui suivent chacune de ces étapes.

3.2 Modèle de concepts du domaine

Afin de développer les composants nécessaires pour le développement du mashup, nous avons besoin de délimiter les concepts qui caractérisent le domaine. Nous représentons cette connaissance du domaine à travers le modèle des concepts. Pour modéliser ce type d'information, il faut comprendre les éléments d'information fondamentaux et les relations qui les structurent. La compréhension de ces éléments nous aide à définir comment le domaine doit être représenté.

Le modèle des concepts comporte deux types d'acteurs :

— Les utilisateurs finaux qui développeront différents mashups à partir de composants existants. Pour eux, il est important que le modèle conceptuel soit facile à comprendre.
— Les développeurs de composants, qui sont des programmeurs. Ces derniers doivent connaître le format des données dans lequel les entités et les relations peuvent être représentées, par exemple en termes de schémas XML.

Nous représentons le modèle des concepts de domaine comme un diagramme de classes. Il inclut également une représentation des entités sous forme de schémas XML. Par exemple, dans la figure 3.1 nous n'avons présenté que les principaux concepts pouvant être identifiés dans notre domaine applicatif, détaillant les entités, les attributs et les relations. L'élément principal dans la planification de séjours touristiques est le lieu d'intérêt qui est localisé à une position géographique. Ainsi, un point d'intérêt est caractérisé par des thèmes selon la catégorie à laquelle est rattachée le lieu par exemple, site culturel, hébergement, restauration,etc. Il est ouvert pour les visiteurs à des horaires spécifiques et soumis à une tarification. Une visite est composée d'un ensemble de lieux d'intérêt. Elle est déterminée par une date de début et une date de fin du séjour ainsi qu'un budget à respecter. Un schéma XML est fourni en annexe (A.1) pour une description plus détaillée des concepts.

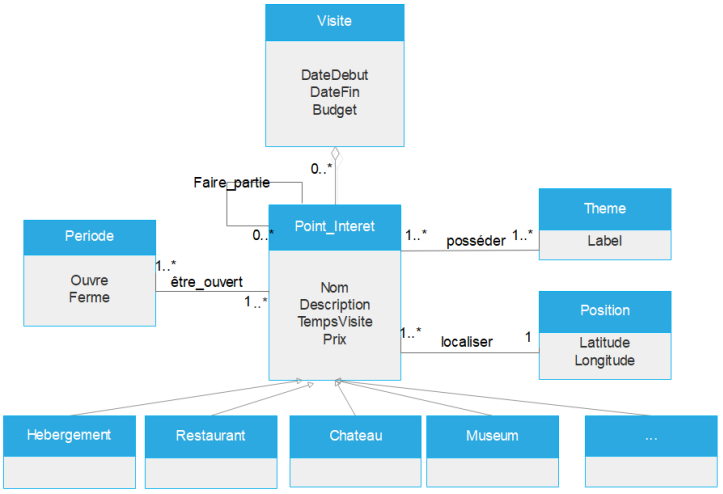


FIGURE 3.1 – Modèle de concepts du domaine planification touristique

3.3 Méta-modèle de mashup

Le modèle des concepts décrit précédemment est représenté par un langage générique de modélisation à savoir le diagramme de classe UML ou le schéma XML. Cependant, pour le développement de mashups pour les utilisateurs finaux, le choix du formalisme de modélisation n'est pas évident. Si nous essayons d'injecter des types de données spécifiques dans les modèles de mashup existants, nous ne parviendrons probablement pas à réduire le niveau de complexité. C'est pourquoi nous avons choisi de définir une approche spécifique au domaine.

Ainsi, pour définir le formalisme de modélisation et le type de mashup, il est nécessaire

de modéliser les fonctionnalités (en termes de capacités logicielles) que les mashups doivent pouvoir supporter. Les mashups sont basés sur les composants et intègrent une variété de services, de sources de données et d'interfaces utilisateur. Ils peuvent avoir besoin d'une disposition propre pour placer les composants, avoir besoin de flux de contrôle ou de flux de données, demander la synchronisation des interfaces utilisateur et l'orchestration des services, etc. L'ensemble de ces caractéristiques de mashup est décrit par son méta-modèle. Le méta-modèle spécifie donc quel type de mashup est décrit et comment le processus de mashup est réalisé.

Nous commençons par la description du méta-modèle générique qui est capable de s'appliquer à différents domaines. Nous présentons ensuite le méta-modèle spécifique au domaine, qui nous permettra d'élaborer des modèles de mashup spécifiques au domaine touristique.

3.3.1 Méta-modèle générique de mashup

Le méta-modèle générique de mashup définit la classe ou le type de mashup, c'est-à-dire les composants et les paradigmes de composition que la plateforme doit fournir pour le développement de ce type de mashup. Les constituants de ce modèle de mashup sont décrits par la Figure 3.2. Il s'agit de spécifier le pouvoir d'expressivité de la plateforme de mashup par le biais des composants combinables. Le modèle de mashup est structuré en deux classes de composants :

— La classe de composants d'intégration où le modèle de mashup doit supporter la capacité d'intégrer de multiples sources de données afin d'unifier les vues partielles retournées par ces ressources d'information.
— La classe de composants connecteurs décrivant la structure abstraite du mashup.

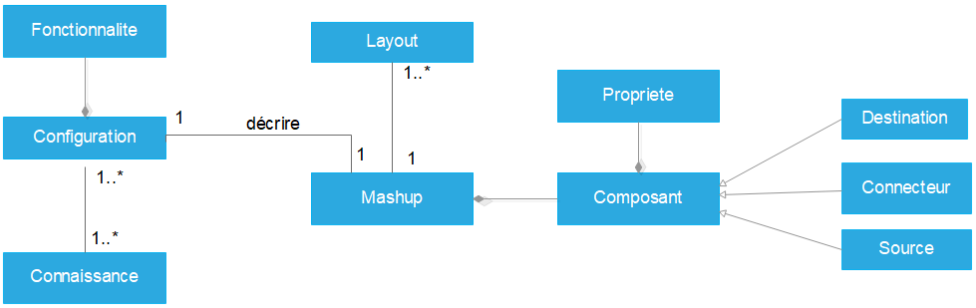


FIGURE 3.2 – Méta-modèle générique de mashup

La dérivation du mashup concret est réalisée par le système de configuration. Un problème de configuration ou un système de configuration est généralement considéré comme le processus de sélection et de mise en connexion d'instances de composants formant le mashup

en tenant compte des contraintes. Un système de configuration doit apporter ainsi des réponses aux questions suivantes :

— Existe-t-il une configuration qui répond aux exigences de l'utilisateur ?

— Quelles sont les instances qui traduisent le mashup abstrait en mashup concret ?

Par conséquent, il est clair que le système de configuration est spécifique au domaine pour une satisfaction des utilisateurs, où ces derniers spécifient le mashup suivant une logique de composants. Il s'agit d'une passerelle adéquate au dessus des caractéristiques techniques.

Cette logique de composants pour la configuration de mashup n'est pas seulement plus proche des particularités du domaine mais fournit également une vision plus réaliste de la problématique de mashup. C'est la raison qui nous a conduit à choisir la configuration comme technique de construction du mashup.

Définition 1. (Problème de configuration) : le problème de configuration de mashup CP se définit formellement comme un tuple $CP = (m,\ F,\ CK)$, où :

— m : représente le mashup à configurer,

— F : correspond aux fonctionnalitésà réaliser,

— CK : regroupe les connaissances de configuration (Configuration Knowledge) permettant l'agrégation des composants du mashup.

Définition 2. (Mashup) :

le mashup m est représenté par le tuple $m = (id,\ layout,\ C,\ DF)$ où :

— id : permet d'identifier le mashup,

— $layout$: spécifie l'arrangement graphique des composants,

— C : représentent l'ensemble de composants,

— DF : exprime le flux de données.

Définition 3. (Composant) : un composant c est défini par le tuple $c = (name,\ icon,\ type,\ P)$ représentant une encapsulation d'un service Web où :

— $name$: correspond au point d'entrée du service (URI),

— $icon$: exprime une identité graphique du composant permettant d'informer les utilisateurs sur les données manipulées,

— $type$: indique le type du composant à savoir un composant de données ou un composant d'opérateur,

— P : représente l'ensemble de propriétés exposées.

Une propriété $p \in P$ est appelée *propriété de configuration (Configuration Property)* permettant de configurer les composants. Typiquement, il s'agit de filtre pour l'élagage des instances qui ne respectent pas le prédicat.

Pour chaque propriété $p \in P$ nous associons une valeur $v_p \in V$ où $V = \{v_p \mid p \in P\}$. Ainsi le couple $f = (p, v_p)$ où $p \in P$ et $v_p \in V$ est appelé la fonctionnalité f du composant. L'ensemble des fonctionnalités $F = \{f_c \mid c \in C\}$ d'un composant c permet de spécifier les valeurs rattachées aux propriétés. Chaque propriété p apparaît au plus une fois dans l'ensemble F.

Le flux de données $df \in DF$ du mashup, qui implémente la logique de propagation des données entre deux composants c_i et c_j, utilise les propriétés de ces derniers. Il se définit ainsi, $df \in P_i \times P_j$. L'ensemble DF assure la compatibilité des composants et la dépendance des types de données.

Par ailleurs, les connaissances de configuration se basent sur les fonctionnalités des composants pour la configuration du mashup. Elles utilisent les connexions fonctionnelles, c'est-à-dire que le modèle de mashup est fondé sur les propriétés fonctionnelles des composants à configurer. Les relations entre les composants sont définies par les fonctionnalités locales des composants qui dérivent des dépendances compositionnelles.

Par conséquent, les connaissances de configuration ne spécifient pas des relations explicites figées entre les composants mais exploitent des dépendances plus profondes du domaine. De cette manière, les connaissances en configuration peuvent être acquises et maintenues facilement. La modification et l'ajout de composants n'affectent pas les autres composants de la configuration. Toutefois, un plus grand espace de recherche doit être traité puisqu'il n'y a aucun guidage prédéfini du processus de configuration. Ce processus est implicitement limité par les descriptions des composants locaux qui doivent former un modèle global fonctionnant correctement à l'assemblage.

Nous représentons ces *connaissances* par un ensemble *d'opérateurs de configuration* qui agissent sur les propriétés des composants. Il s'agit d'opérateurs d'addition qui assurent la combinaison de composants pour construire le mashup.

Définition 4. (Opérateur de configuration) : pour chaque propriété p nous associons un opérateur d'addition, op_p, qui est une fonction d'ordre partiel $op_p : \{V \times V \to V$ $(v_p, v_p) \to v_p\}$ $OP = \{op_p \mid p \in P\}$ comprend tous les opérateurs d'addition.

Un opérateur d'addition spécifie comment deux valeurs d'une propriété seront composées à une nouvelle valeur, si un nouveau composant est ajouté à une collection donnée, qui eux-mêmes décrivent une partie du système de mashup à configurer.

Un opérateur ne spécifie pas obligatoirement un opérateur d'addition entre deux nombres, mais tout type d'opération est réalisable. De manière générale, les opérateurs de configuration expriment des fonctions d'agrégation sur les propriétés des composants. Par le biais de ces connaissances de configuration, nous pouvons représenter les exigences internes et externes du mashup à configurer. Les exigences internes sont modélisées par le flux de

données qui exprime le mapping des données échangées entre composants. Et les exigences externes spécifient les besoins qui sont fournis par l'utilisateur final.

Définition 5. (Requête de mashup) : la requête de mashup se définit comme le tuple $MCQ = \{C, CONST\}$ où :
— C : représente l'ensemble de composants qui forme le mashup
— $CONST$: regroupe l'ensemble de contraintes utilisateur à appliquer sur le mashup.

Définition 6. (Contrainte de configuration) : une contrainte $const \in CONST$ exprime une relation globale entre les composants constituants le mashup. Elle se décrit sous la forme suivante : $const :< fonctionnalite >< operateur >< valeur >$ où :
— $< fonctionnalite >$ désigne une propriété globale du mashup, par exemple budget total, durée totale, nombre d'activités, etc.
— $< operateur >$ désigne des opérateurs (=,<,>) pour les fonctionnalités globales et des opérateurs de choix opérant sur les composants
— $< valeur >$ désigne la valeur associée à la fonctionnalité.

Après avoir défini le modèle de mashup comme un problème de configuration qui modélise des composants via des propriétés, nous présentons la solution de ce problème. Pour cela, nous avons d'abord besoin de définir comment composer les fonctionnalités.

Soient les fonctionnalités suivantes f et g telles que $(f, x) \in F$ et $(g, y) \in F$ ainsi qu'un opérateur $op(x, y) \in OP$. Une composition des fonctionnalités f et g se définit comme suit :

$$\phi((f,x),(g,y)) = \begin{cases} (f, a_f(x,y)) & \text{si } f = g \\ (f,x),(g,y) & \text{sinon.} \end{cases}$$

D'après cette équation, le calcul de fonctionnalités composites se base sur les opérateurs de connaissances de configuration. Si les fonctionnalités sont égales alors l'opérateur peut être appliqué pour calculer la nouvelle valeur de la fonctionnalité. Sinon, l'ensemble de fonctionnalités composites contiendra les deux fonctionnalités élémentaires.

Définition 7. (Solution de configuration) : une solution de configuration $CS \in CSS$ L'ensemble de solution de configuration (Configuration Solution Set) à un problème de configuration CP se décrit par le couple $CS = (I, Q)$ où :
— I : correspond l'ensemble d'items sous la forme (n, c) désignant que le composant $c \in C$ apparaît n fois dans le mashup à configurer.
— Q : représente l'ensemble de fonctionnalités du mashup *(appelée qualité)* sous forme de couple (f, x) qui définit la valeur de la fonctionnalité f.

Une solution de configuration CS se définit aussi par récurrence comme suit :

1. $CS = (\emptyset, \emptyset)$ est une solution de configuration.

2. Si $CS = (I, Q)$ est une solution configuration et $c \in C$, alors $CS' = (I', Q')$ est une solution de configuration si et seulement si :

   (i) Pour chaque $(f, x) \in Q$ et Pour chaque $(g, y) \in Q$, la fonction de composition $\phi((f, x), (g, y))$ est définie ou $Q = \emptyset$.

   (ii)

$$I' = \begin{cases} I \backslash (k, c) \cup (k+1, c) & \text{si } \exists (k, o) \in I \\ I \cup (1, c) & \text{sinon.} \end{cases}$$

   (iii)

$$Q' = \begin{cases} Q \cup (\phi((f, x), (g, y)) | (f, x) \in p_c, (g, y) \in Q) & \text{si } Q \neq \emptyset \\ p_c & \text{sinon.} \end{cases}$$

La condition (i) garantit que seuls les composants $c \in C$ seront ajoutés à la configuration CS, si toutes les propriétés du composant p_c peuvent être combinées avec toutes les qualités de CS. La condition (ii) spécifie comment un nouveau composant peut être ajouté à l'ensemble d'items I. La condition (iii) spécifie comment un nouvel ensemble de qualités sera construit, si un nouveau composant est ajouté à la configuration.

Bien que les qualités et les fonctionnalités soient syntaxiquement égales, nous distinguons entre elles puisqu'une fonctionnalité est la caractéristique d'un composant, tandis qu'une qualité (f, x) est le résultat de la combinaison de plusieurs composants ayant la fonctionnalité f dans leurs ensembles de fonctionnalités F.

Par conséquent, une solution de configuration CS est une solution de configuration qui répond à la requête de mashup MCQ si et seulement s'il existe une qualité (fonctionnalité composite) $(f, x) \in Q$ pour chaque contraintes $const = (g, op, y) \in CONST$ telle que $f = g$. L'ensemble de solutions de configuration CSS (Configuration Solution Set) où $CS \in CSS$ représente l'espace de variantes qui satisfait la requête.

Nous partons de ce modèle simple, tant en termes de notation que de sémantique d'exécution, qui permet aux utilisateurs finaux de modéliser leurs propres mashups facilement.

Cette image représente une spécification exécutable qui n'est pas si loin de la complexité de tracer un diagramme. Cependant, comparée à la spécification BPMN ou le processus BPEL nous sommes à un niveau d'abstraction plus élevé. En effet, c'est au niveau du mapping de données que réside toute la complexité, ce n'est pas pratique pour les utilisateurs finaux qui n'ont pas les connaissances nécessaires de réaliser cette tâche. Le mapping est à la charge du système de configuration qui va résoudre la requête de mashup définie par l'utilisateur de manière déclarative à travers des composants graphiques sans se soucier des détails de liaison entre les composants. L'identité graphique du composant aidera l'utilisateur à comprendre la fonctionnalité offerte. Ainsi, chaque composant dans le modèle

de mashup spécifique au domaine implémente une passerelle cognitive qui manipule les données décrites dans le modèle de concepts. Ceci signifie que tous les composants sont conformes au modèle de concepts.

3.3.1.1 Syntaxe générique de mashup

La définition graphique de mashup nécessite une spécification de la syntaxe descriptive des concepts du méta-modèle. La figure 3.3 illustre les symboles de la syntaxe où les composants sont représentés par des pièces de puzzle dotées d'une icône décrivant l'identité graphique.

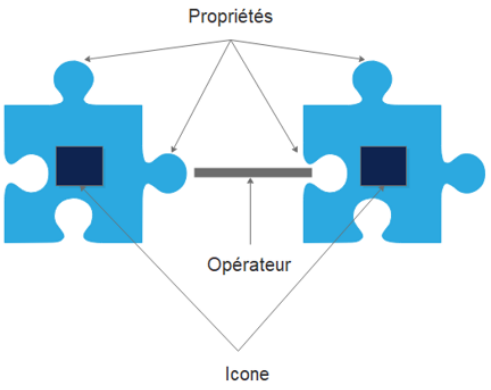


FIGURE 3.3 – Syntaxe basique du méta-modèle de mashup

La métaphore du puzzle s'aligne avec notre approche puisqu'elle considère que le mashup est constitué de pièces qui s'emboîtent selon une configuration donnée ce qui favorise la flexibilité du système. En outre, cette métaphore est assez familière aux utilisateurs finaux qui, par analogie, transfèrent au monde du développement les connaissances que le concept de métaphore possède dans le monde réel. Ainsi, avec un niveau cognitif facilement compréhensible, cette syntaxe facilite la création de mashup en réduisant l'effort d'apprentissage nécessaire à un utilisateur pour manipuler les concepts.

3.3.2 Méta-modèle spécifique de mashup

Dans cette section, nous présentons l'injection d'un domaine particulier dans le méta-modèle générique, notamment le domaine touristique. Cette injection permet la génération de mashup spécifique au domaine. En fait, fournir aux utilisateur un modèle restreint aux concepts du domaine n'est pas suffisant pour assurer la simplicité d'utilisation.

Bien qu'il apparaisse relativement simple, l'utilisateur sera toujours en face d'un nombre assez important de composants à manipuler. Ainsi, une structuration intuitive des com-

posants se voit nécessaire. Ceci passe par une compréhension des concepts par les experts du domaine afin de les transformer en composants "parlant le langage de l'utilisateur". Nous décrivons cette spécialisation dans un modèle de mashup spécifique au domaine qui est une instance du modèle générique introduit précédemment. Cette instanciation consiste à étendre le modèle générique avec des types spécifiques et une terminologie propre afin d'obtenir des activités spécifiques au domaine.

3.3.2.1 Classes de composants du domaine

Nous décrivons dans cette section les classes de composants que l'utilisateur manipulera dans la réalisation du mashup propre au domaine. L'ensemble de ces classes représente le modèle de composants spécifiques. Dans le cas de la planification touristique, il s'agit d'identifier les classes de composants suivantes :

a. **Composants sources**

   Ces composants permettent l'extraction de données depuis les sources d'information. Ils peuvent avoir une ou plusieurs propriétés qui indiquent les contraintes locales à respecter. Il s'agit, par exemple, de services de données de type Google Places API qui retourne une liste de points d'intérêt selon la catégorie de l'activité touristique.

b. **Composants de visualisation**

   Ces composants assurent la visualisation du mashup (le séjour touristique) selon un format spécifié par l'utilisateur via les propriétés de configuration.

c. **Composants opérateurs**

   Ces composants offrent aux utilisateurs la possibilité d'exprimer des contraintes globales sur le mashup en construction. Il s'agit d'opérations de choix conformes à l'ensemble $CONST$ définit dans le modèle générique. Nous répertorions ces composants en deux catégories : les composants opérateurs binaires et les composants opérateurs unaires.

1. **Composants opérateurs unaires** : correspondent à contraindre le mashup en définissant un seuil sur une des propriétés du composant. Par exemple, dans notre scénario de planification de séjours touristiques, un composant opérateur unaire peut être l'opérateur max exprimant une contrainte telle que ne pas inclure dans la solution plus de trois musées.
2. **Composants opérateurs binaires** : permettent aux utilisateurs d'exprimer leurs choix.
   — **Opérateur OU BIEN** : le visiteur a la possibilité d'indiquer qu'il souhaite que son plan inclut soit les musées soit les châteaux.

— **Opérateur APRES** : le visiteur exprime via cet opérateur une séquence d'activités qu'il souhaite réaliser lors de son séjour les une à la suite des autres.

Inspirés par les approches de composition visuelle, nous identifions des mécanismes d'abstraction pour combiner les composants à l'aide d'expressions logiques de sorte que ces derniers puissent être adéquats au modèle mental de l'utilisateur final. En effet, l'opérateur "OU BIEN" et "APRES" font référence respectivement à la sémantique de l'opérateur logique "OU" et l'opérateur logique "ET".

Cette métaphore d'expressions logiques a été définie suite à une la réalisation d'une étude orientée utilisateur qui vise à accroître l'attitude des utilisateurs non techniques à l'égard des expressions logiques.

En pratique, la définition des classes de domaine nécessite d'envisager plusieurs scénarii afin de couvrir toutes les classes possibles de composants.

3.3.2.2 Syntaxe spécifique au domaine

Après avoir défini les classes de composants spécifiques, nous spécifions une syntaxe propre au domaine dérivée de la syntaxe générique du modèle de mashup.

L'utilisateur manipule les instances de composants spécifiques, qui consomment et produisent des concepts du domaine, afin de mettre en œuvre son mashup. Un exemple de scénario de mashup illustrant l'utilisation des composants spécifiques au domaine est décrit par la Figure 3.4.

FIGURE 3.4 – Exemple de mashup exprimé par la syntaxe spécifique au domaine

Les composants POIs sont des instances de composants sources permettant l'extraction des points d'intérêt dont le type est spécifié par l'icône associée. Les propriétés de configuration de ces composants permettent à l'utilisateur d'indiquer les contraintes locales. Par exemple, le scénario décrit par la Figure 3.4 expose trois composants sources représentant une visite de (s) château(x), un restaurant et une participation à (aux) événement(s). Afin d'exprimer les contraintes globales qui traduisent un choix entre les composants, l'utilisateur a interconnecté les deux premiers compsants avec l'opérateur *APRES*, ce qui signifie

que le mashup résultat doit comporter obligatoirement des instances de ces derniers dans la mesure de satisfaire les autres contraintes (budgétaire, temporelles, ...). Par l'opérateur *OU BIEN*, l'utilisateur déclare qu'il souhaite éventuellement participer aux événements. Le dernier composant dans la requête symbolise un composant de visualisation du résultat du mashup sur une carte. Un code couleur a été associé à chaque catégorie de composant : le rouge pour les composants de visualisation, le vert pour les composants opérateurs et le bleu pour les composants source.

Cette restriction de la flexibilité du modèle générique et la réduction aux classes spécifiques nous permettra de gagner en terme d'intuitivité vu que la focalisation est orientée plus vers les besoins spécifiques. En effet, face aux nombreux modules, les utilisateurs n'arrivent pas à retrouver ceux adaptés à leurs besoins situationnels. Ajoutant à cela la complexité de connecter ces modules pour former le mashup.

Ainsi nous présentons une plateforme de mashup qui implémente le méta-modèle spécifique pour exposer une boite à outils extensible de composants spécifique au domaine.

Via cet ensemble de composants, l'utilisateur peut construire son mashup simplement sans se soucier des étapes de composition (trouver le composant le plus adapté à son besoin, lire ses spécifications et déterminer sa compatibilité avec les autres composants) qui sont considérées compliquées et décourageants surtout pour les utilisateurs non familiers avec cette catégorie d'outils.

Après avoir défini la requête de mashup, le framework de configuration traduit cette dernière en système de configuration afin de satisfaire la demande de l'utilisateur. Le processus de cette transformation fera l'objet de la section suivante.

3.4 Scénario de configuration pour le mashup touristique

Avant de décrire le framework de configuration de mashup, nous illustrons par notre scénario de référence comment est configuré le mashup afin de planifier une visite touristique. L'utilisateur commence à construire sa requête de mashup interactivement via la sélection de composants qui représente une structure abstraite du mashup souhaité. En occurrence, la requête (Figure 3.4) décrit une visite de site touristique avec une spécification de la propriété de configuration catégorie qui prend les valeurs château, parc et jardin et musée. Elle comprend aussi deux autres composants source qui désignent respectivement un restaurant de type local et la participation à un événement musical. L'utilisateur termine la description de sa requête par spécifier le mode de visualisation du résultat qui sera au format carte. Les contraintes globales qui figurent dans ce mashup exposent une limite financière et temporelle.

La requête est traitée ainsi par le framework de configuration pour générer le résultat

(les plans de visite). Les services ou APIs correspondants aux composants de la requête sont invoqués et l'ensemble des données est récupéré, les instances de services sont représentées par des symboles différents. Ces instances sont filtrées ensuite pour respecter les contraintes. A titre d'exemple, les sites touristiques qui ne sont pas ouvert pendant la durée de visite souhaités sont élagués des instances candidates. Le processus de configuration repose sur les connaissances configuration représentées par les opérateurs d'addition pour connecter les données. Pour mieux illustrer cette phase de configuration, nous définissons les opérateurs suivants :

$$OP = \{op_{duration}, op_{price}\}$$

$$op_{price}(x, y) = \begin{cases} x.price + y.price & \text{si } x.price + y.price < B \\ \perp & \text{sinon.} \end{cases}$$

$$op_{duration}(x, y) = \begin{cases} x.d + y.d & \text{si } x.d + y.d + \delta t < D, x \in C, y \in C \\ \perp & \text{sinon.} \end{cases}$$

L'opérateur op_{price} permet de déterminer le coût de la visite par l'accumulation progressive des tarifs des instances de service limitée par le budget global B. La définition de l'opérateur $op_{duration}$ est similaire, il additionne les temps de visite et de déplacement entre les points d'intérêt pour ne retenir que ceux qui sont faisables pendant la durée de visite D.

Une solution de configuration comporte les candidats qui s'appliquent à l'ensemble d'opérateurs. Ainsi, nous représentons par la Figure 3.5, l'ensemble des solutions possibles.

Château de Jallange	La Fourchette	Château d'Amboise	Château Clos-Lucé	Festival Européen de la musique de Renaissance
Château de Tours	Le Saint Honoré	Tours su Loire	Fête de la musique	
Château de Villandry	La doulce terrasse	Château Azay le rideau	Château de Langeais	Tortay Gilles
Château de Chenonceau	L'orangerie	Château de Loches		

FIGURE 3.5 – Les solutions de configuration

Dans l'exemple, la première solution se définit comme suit

$$CS_1 = \{(I_1, Q_1)\}$$

$$I_1 = \{(3, POI)\ ;(1, Restaurant)\ ;(1, Evenement)\}$$

$$Q_1 = \{(cout, 60)\ ;(duree, 1)\}$$

Contrairement à la technique de recommandation, les points d'intérêt proposés par la configuration ne sont pas ordonnés. Cela signifie que le visiteur peut choisir de les explorer

dans n'importe quel ordre, cependant ils forment un intérêt commun. Nous parlons de configuration de package qui vérifie certaines caractéristiques.

— Une configuration est dite *valide* si elle respecte les contraintes de l'utilisateur par exemple la contrainte budgétaire qui se réfère au montant à ne pas dépasser pour une activité souhaitée ou le nombre maximal d'activités à pratiquer. Étant donnée un ensemble V de valeurs non négatives, un opérateur op : $2^V \rightarrow \mathbf{R}_+$ et un seuil de budget B alors $\forall CS_i \in CSS; i, j \in \mathbf{N}$ $op(CS_i.I_j, CS_i.I_{j+1}) \leq B$

— Par ailleurs, l'ensemble des solutions de configuration garantit une seconde caractéristique, *la représentativité* qui vise à offrir une expérience plus complète c'est-à-dire fournir une couverture d'un large éventail de recommandations intéressantes. En effet, il a été montré que la diversification a un effet positif sur la satisfaction des utilisateurs [Ziegler *et al.*, 2005].Nous exprimons la représentativité par la couverture spatiale. L'intuition est que chaque solution de configuration représente des activités à pratiquer dans une zone donnée d'une ville ce qui conduit à avoir des activités géographiquement proches les unes des autres. Elle peut être vue comme synonyme d'attractions dans différents quartiers de la ville ou différentes communes de la région. Cette couverture est basée sur la distance de configuration. Elle est introduite dans ce qui suit.

3.5 Le framework de configuration de mashup

Partant de la requête de mashup définie par l'utilisateur de manière interactive, le framework de configuration de mashup assure sa transformation en système de configuration capable d'invoquer les services Web correspondants et de connecter les données retournées pour répondre à cette requête. La figure 3.6 illustre le processus de configuration afin d'exécuter la requête de mashup. Il s'agit d'un processus incrémental qui génère une configuration permettant de fournir la fonctionnalité globale par application des opérateurs de configuration définis par les connaissances de configuration. Le framework de configuration est structuré en deux grandes phases : une étape de filtrage et sélection qui génère à partir de la requête de mashup des services candidats qui seront les entrées de la seconde étape de configuration assurant la construction de la composition en plusieurs itérations.

3.5.1 Phase de Filtrage et de Sélection

La phase de sélection du framework de configuration est déclenchée à l'arrivée d'une requête de mashup décrivant la sémantique du service composite souhaité et les contraintes globales à remplir.

A partir de cette structure générale du mashup à construire, l'ensemble de composants est analysé afin d'invoquer les APIs Web correspondantes. Cette invocation nécessite l'ap-

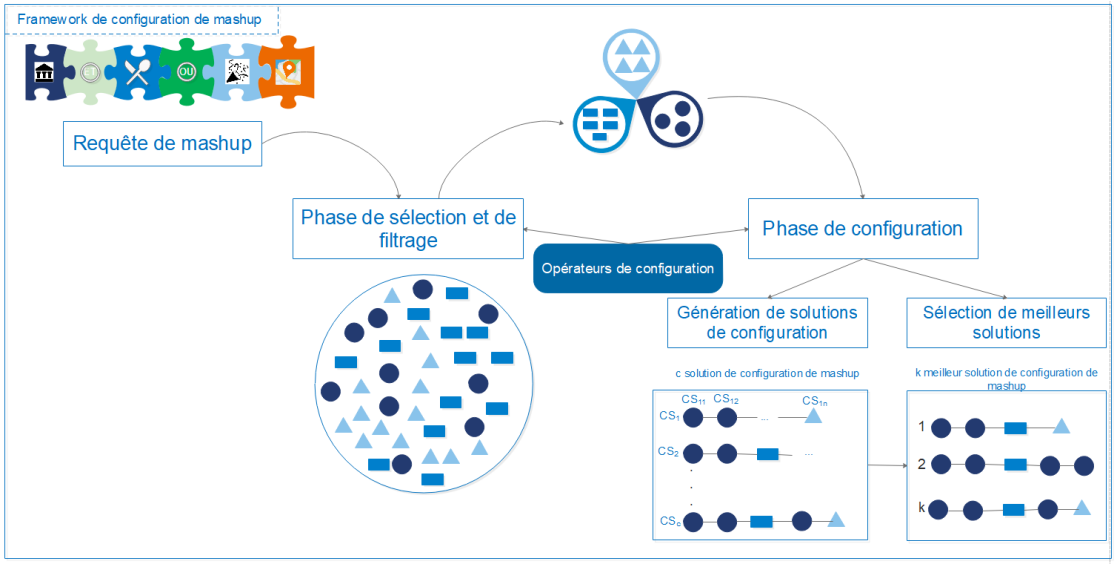


FIGURE 3.6 – Framework de configuration de mashup

plication des opérateurs de filtre sur l'ensemble de services candidats pour la composition en vue de ne garder que les données respectant les contraintes locales indiquées par le biais des propriétés des composants. L'ensemble des opérateurs de filtre (FILT) est une sous classe des opérateurs de configuration permettant de conditionner les services sur la base de certaines propriétés. Il se définit ainsi :

$$FILT = \{filt_p(x) \ = x.p \ [=, <, >] \ val \mid p \in P, val \in V\} \ .$$

Bien que ces opérateurs soient syntaxiquement similaires aux contraintes, ces dernières s'appliquent sur les fonctionnalités locales des composants alors que les contraintes s'adressent aux fonctionnalités globales appelées qualités qui reflètent un conditionnement sur le mashup.

Pour chaque composant source qui participe à la définition du mashup, une requête d'interrogation de l'API Web correspondante est produite et les opérateurs de filtre sont exécutés pour toute propriété de configuration spécifiée par l'utilisateur afin de raffiner la sélection et générer les services candidats.

3.5.2 Phase de configuration

Une fois que l'ensemble des services candidats pour la composition est prêt, l'étape de configuration vise à agréger ces données afin de générer le mashup qui satisfait la demande de l'utilisateur et respecte les contraintes. Cette génération suit un modèle progressif croissant permettant de construire le mashup en plusieurs itérations. L'approche incrémentale consiste à créer des compositions par des patterns de paires en appliquant les opérateurs de

configuration, ce qui signifie une composition de deux à la fois. Une configuration initiale CS_1 est étendue pour incorporer d'autres instances de services candidats par exécution des opérateurs de configuration formant une nouvelle configuration CS_2. Celle-ci est dite configuration partielle puisqu'elle ne vérifie pas encore toutes les contraintes globales du mashup. Par conséquent, pour atteindre cet objectif, la configuration partielle doit être complétée par l'addition d'autres instances jusqu'à l'obtention d'une configuration finale CS_n. La construction des configurations suit alors une conception incrémentale, descendante, c'est-à-dire qu'elle est initiée avec l'union des services atomiques pour former des compositions intermédiaires (partielles), jusqu'à arriver aux compositions complètes désirées par l'utilisateur.

Cette technique de composition itérative est inspirée de la résolution des fonctions arithmétiques composées où $h = fog$ avec f et g des fonctions de base (somme, soustraction, multiplication et division). Celle-ci suit le schéma suivant Par exemple, dans notre scénario,

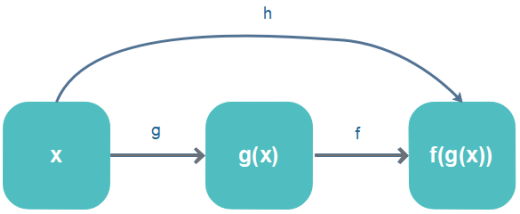


FIGURE 3.7 – Fonction composée

la configuration de la liste des attractions consiste à invoquer premièrement le service de géolocalisation *(getLocation)* permettant de récupérer la position de l'utilisateur, ensuite cette données sera utilisées par le service d'attraction *(getAttractionList)* pour retourner les attractions proches de la zone de l'utilisateur. Puis, afin de calculer le temps de déplacement entre les POIs, nous invoquons le service *getRoute* qui sera paramétré avec les coordonnées des lieux. Ce processus est illustré par la figure 3.8

Il s'agit d'un processus de composition qui agrège les services de manière incrémentale et contrôlable, ce qui évite d'avoir recours à un langage de flux supplémentaire pour décrire des interactions complexes. L'ensembles de services et leurs interconnexions est codé par le même système de configuration offrant un mashup satisfaisant qui respecte la limite fournie par l'utilisateur (appelée niveau d'acceptation *acc*).

3.6 Algorithme de configuration de mashup

Nous nous focalisons dans cette section sur la phase de configuration et nous présentons l'algorithme de configuration de mashup qui s'appuie sur le paradigme " **Générer**

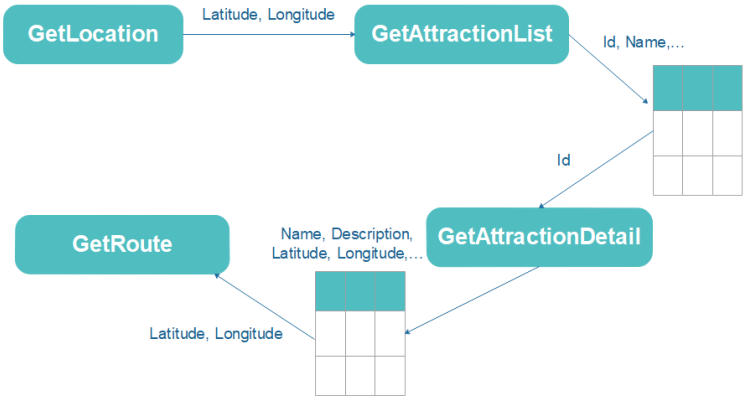


FIGURE 3.8 – Le processus incrémental de configuration

puis Sélectionner ". Il s'agit de produire un nombre assez important de solutions de configuration pour sélectionner ultérieurement les meilleures. De ce fait, la construction des configurations est faite en deux étapes :

— Premièrement, un ensemble de configurations valides et cohérentes, respectant toutes les contraintes spécifiées par l'utilisateur, est produit avec une cardinalité c telle que $c \gg k$. c représente le nombre de configurations à créer et k le nombre de configurations à exposer à l'utilisateur.

— Deuxièmement, un ensemble de k configurations est sélectionné permettant de garantir une représentativité. En effet, lors la visualisation des configurations, l'expérience de l'utilisateur est souvent affectée par la diversité des éléments. Nous parlons ainsi de l'effet visuel qui guide la représentation efficace des configurations. Nous détaillons cette notion dans ce qui suit.

La démarche générale décrivant cette approche est donné par l'algorithme 1. Il prend en entrée l'ensemble d'instances de services (SERVICES) générées lors de la phase de sélection, la requête de mashup (MQ) définie par l'utilisateur afin d'exploiter l'ensemble de contraintes dans la génération de configurations, ainsi que la taille k de l'ensemble des solutions de configuration qui sera le résultat fourni.

Algorithm 1 Generate_and_Select

Data: k, SERVICES, MQ

Result: a set of k configuration solutions

```
1 Cand = COCO(SERVICES, MQ.CONST) ;
2 G = build_config_graph(Cand) ;
3 Return select_config(k,G) ;
```

La phase de génération est assurée par la procédure *COCO* (ligne 1) qui produit un ensemble de configurations candidates valides et cohérentes. La procédure $build\_config\_graph$ (ligne 2) permet de construire le graphe à partir de cet ensemble de candidats. Enfin, le résultat est fourni par la procédure $select\_config$ qui réalise la phase de sélection. Nous détaillons dans ce qui suit chacune de ces procédures.

3.6.1 Générer les configurations candidates

Nous nous concentrons dans cette sous section à la phase de génération de configurations candidates. La méthode de génération est décrite par l'algorithme 2, appelé COCO (Configuration One Component by One Component), qui applique une stratégie de composition incrémentale.

Partant de l'ensemble d'instances de services généré à partir de la phase de sélection du framework de configuration, à chaque itération un élément pivot est choisi et la configuration est construite progressivement autour de ce pivot.

Algorithm 2 COCO

Data: SERVICES, MQ.CONST, c

Result: a set of candidate configuration solutions

```
2  Cand = ∅ ;
3  Pivots = sort(SERVICES) ;
4  while Pivots ≠ ∅ and |Cand| ≤ c do
5  |   p = Pivots [0] ;
6  |   Conf = create_config(SERVICES, p, MQ.CONST) ;
7  |   Pivots = Pivots \ Conf ;
8  |   Cand = Cand ⋃ Conf ;
9  end
10 Return Cand ;
```

L'algorithme COCO commence par initialiser un ensemble vide de candidats et ordonner les éléments pivots pour en sélectionner un. A chaque itération, un élément est prélevé dans les Pivots définis. La stratégie de sélection repose sur une heuristique spatio-temporelle qui limite l'espace de recherche à un rayon spécifique et choisit l'élément le plus proche comme pivot à chaque itération par rapport au pivot précédent. Nous avons opté pour cette heuristique vu l'importance de ce facteur pour notre approche spécifique au domaine.

Une fois le pivot est sélectionné, nous construisons la configuration autour de celui-ci en faisant appel à la procédure $create\_config$ qui se charge d'appliquer les connaissances

de configuration pour la produire. L'algorithme 3 décrit cette méthode. Il s'agit de tisser le mashup en connectant les instances de services par le biais des opérateurs de configuration tout en respectant les contraintes de l'utilisateur. Si la configuration partielle en cours de production ne satisfait pas les contraintes, nous appliquons la technique de Backtrack afin de rechercher une autre alternative. Si nous revenons à la boucle principale de COCO, la configuration retournée par $create\_config$ est rajoutée à l'ensemble des candidats et la liste des pivots est mise à jour.

L'ensemble des configurations générées partagent un élément en commun (appelé S) qui initie chacune d'entre elles. Pour notre domaine spécifique, ça peut être la position actuelle du visiteur ou un point de départ différent choisi par ce dernier.

Algorithm 3 create_config

Data: SERVICES, MQ.CONST, p

Result: a candidate configurations solution

```
2  config = s ;
3  concern = p ;
4  oldCovered = ∅
5  while not satisfy(MQ.CONST) do
6      covered = cover (SERVICE, concern) ;
7      finish = false ;
8      i =0 ;
9      while notfinish or i ≤ |covered| do
10         check= apply_CK(MQ.CONST, config ∪covered[i]);
11         if check then
12             config = config ∪ covered[i] ;
13             finish = true ;
14             concern = covered[i] ;
15             oldcovered = covered ;
16             old = i ;
17         else
18             i= i+1 ;
19         end
20     end
21     if not finish and not satisfy (MQ.CONST) then
22         config = config \{concern};
23         concern = oldcovrerd[old+1] ;
24     end
25 end
```

3.6.2 Construction du graphe des configurations

Après avoir produit l'ensemble des configurations candidates, nous traçons le graphe complet G= (V,E). Chaque configuration candidate $CS \in CSS$ est associée à deux sommets $u \in V$, $v \in V$ et l'arête (uv) reliant les deux sommets u et v associés respectivement aux configurations CS_i et CS_j. Elle est dotée d'un poids w exprimant la distance angulaire entre ces deux configurations.

La distance de configuration est représentée par l'angle (θ) formé à partir du triplet (S, Mcs_i, Mcs_j) où Mcs_i est la médiane traçant les items de la configuration i réduits à la propriété coordonnées géographiques, de même pour CS_j. La Figure 3.9 illustre la représentation de n ensembles de configurations candidates partageant un élément S.

De manière générale, la distance de configuration traduit un mapping des items en fonction de certaines propriétés afin de déterminer une représentativité relative. Par exemple, dans le domaine de la recommandation de films la propriété genre peut être utilisée dans le calcul de distance, ce qui permet de fournir une représentativité thématique des suggestions proposées.

L'élément S constitue un élément central autour duquel gravite un ensemble de satellites. Typiquement, l'utilisateur aura un point de départ et un ensemble d'itinéraires dont chacun contient un ensemble de POIs proches. Ceci permet une exploration efficace. En effet, pour de nombreux utilisateurs, une fois que les configurations satisfont les contraintes d'autres facteurs entrent en jeu pour la décision. Un de ces facteurs est la diversité, ainsi nous nous focalisons sur la diversité géographique pour calculer l'ensemble k de configurations représentatif. Nous introduisons donc *l'effet visuel*, un principe qui guide la façon dont un ensemble de configurations devrait être classé. Formellement le principe de l'effet visuel exploite la distance de configuration afin d'éviter de présenter des configurations trop semblables.

3.6.3 Sélection des meilleurs solutions de configuration

Une fois que l'ensemble des n candidats de configuration est généré, nous cherchons l'ensemble de k solutions parmi les configurations candidates qui assure la représentativité de ces configurations.

Le processus de sélection des meilleurs solutions s'appuie sur une heuristique inspirée de [Asahiro *et al.*, 2000] . Nous avons choisi cette heuristique pour sa simplicité et son applicabilité aux graphes pondérés.

L'algorithme 4 fournit le pseudo-code de cette phase de sélection. Il reçoit en entrée un ensemble de solutions de configurations valides représenté sous la forme d'un graphe pondéré comme expliqué précédemment. Compte tenu de ce graphe et du paramètre k, il s'agit

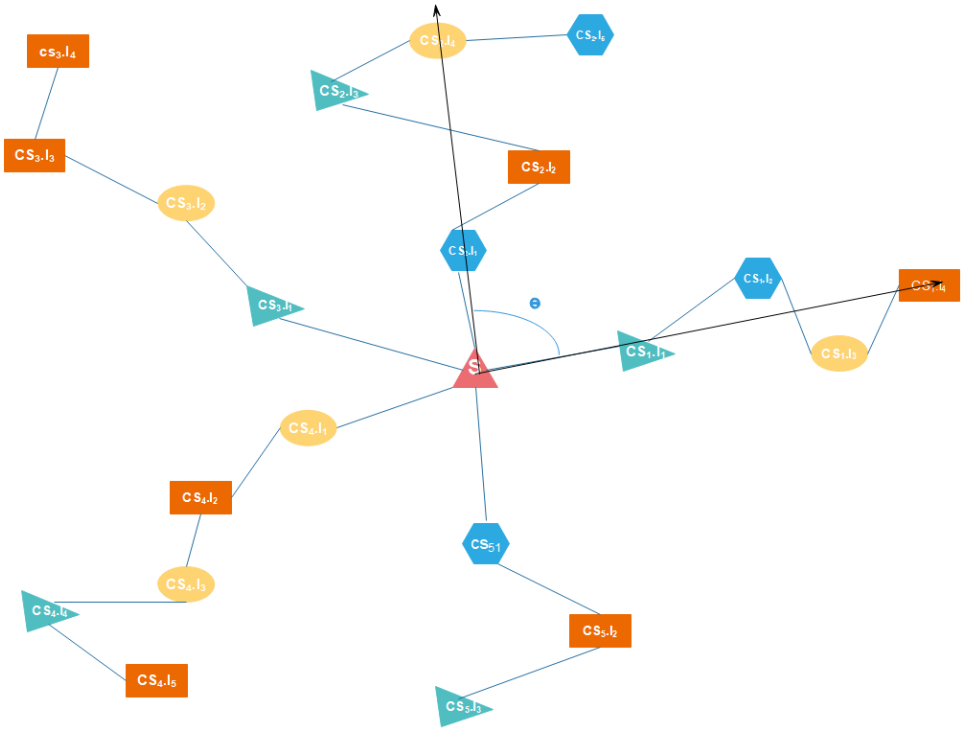


FIGURE 3.9 – Représentativité des solutions de configuration

de supprimer à chaque itération l'arrête ayant le poids minimum, ce qui découpe le graphe en deux sous graphes complets d'ordre $n-1$ et en retenir que celui qui possède le poids total maximal, jusqu'à ce que k sommets soient exactement présents dans le sous-graphe. Ainsi, les sommets de ce graphe obtenu représenteront les solutions de configurations finales qui couvriront les caractéristiques présentées précédemment.

Algorithm 4 select_config

Data: k, the complete and weighted configuration Graph G =(V,E)

Result: a set CSS of k configurations solution

2 CSS = V ;

3 **while** $|CSS| > k$ **do**

4 (u,v) = arg min$\sum_{u \in CSS, v \in CSS} \omega(u, v)$;

5 sub=select_subGraph(G, u, v) ;

6 CSS=sub ;

7 **end**

8 **Return** sub ;

3.7 Etude de la complexité du problème

Nous présentons dans cette section une étude de la complexité de notre problème de configuration de mashup. Nous discutons dans un premier temps la complexité de la recherche d'une solution de configuration, puis nous montrons que le problème de configuration de mashup appartient à la classe de problèmes NP-complet.

3.7.1 Complexité de la recherche d'une configuration

Nous nous référons dans notre approche de mashup au modèle de composant de configuration à base de propriétés. Il s'agit d'une représentation des ressources qui suit la logique de l'offre et la demande de composants. Même si ceci n'est pas explicitement exprimé par les composants où ils exposent des propriétés de configuration sans distinction de la ressource demandée de celle fournie, ce fait n'est pas restrictif de notre modèle. En effet, cette sémantique de l'offre et de la demande est réalisée implicitement par les opérateurs de configuration. Pour démontrer ceci, nous nous référons à la notion de structures algébriques. Nous définissons le monoïde $Monoide = (V_f, op_f)$ où le premier constituant du monoïde représente l'ensemble des valeurs $v_f \in V_f$ de la fonctionnalité $f \in F$ qui sont principalement des valeurs réelles ou des chaînes de caractères. Quant au second constituant, il exprime l'opérateur de configuration op_f assurant l'addition des deux valeurs $x \in V_f, y \in V_f$ de la fonctionnalité f (aussi appelé a_f. Par conséquent, op_f désigne une loi de composition interne. Nous faisons remarquer que cette définition du monoïde est valable pour les opérateurs de configuration binaires. Toutefois, les opérateurs uniaires de filtrage, tels que l'opérateur max, s'associent à l'ensemble des fonctions qui accordent à chaque paire de valeurs de fonctionnalité une valeur booléenne. Autrement dit, le nouveau monoïde se détermine comme suit : $Monoide = (E : v_f \times v_f \rightarrow \{true, false\}, op_f)$

Ainsi les opérateurs de configuration établissent un réseau de dépendances des composants qui représente le modèle fonctionnel du domaine. Le pouvoir de ces opérateurs nous permet de se positionner dans le cadre d'une approche de flux. Ils assurent à la fois le flux de contrôle en dictant comment connecter les composants et pour le flux de données, la transmission des données est décrite par les propriétés aux quelles s'appliquent les opérateurs. Ceci est réalisé de manière transparente pour l'utilisateur, car les utilisateurs finaux ne pensent pas forcement à connecter les composants [Desolda *et al.*, 2017c].

De ce fait, la configuration de mashup peut être vue comme une planification qui décrit la séquence de composants (instances de service) et d'opérateurs à enchaîner. La solution de configuration consiste ainsi à chercher un plan qui réalise la fonctionnalité de mashup en connectant les composants. Avant d'étudier la complexité de la recherche d'un tel plan, nous définissons le plan linéaire.

Définition 8. (Configuration linéaire) : un plan de configuration linéaire ou une configuration linéaire est une permutation $perm = (f_1, ..., f_n)$ où $f \in F, n = |F|$ dite sérialisable. Elle est générée de la manière suivante : pour chaque fonctionnalité f_i du mashup recherché, il existe un ensemble de composants C qui exposent exclusivement des fonctionnalités f_j où $j > i$. Autrement dit, il s'agit de définir un ordre total sur les fonctionnalités f du mashup de sorte que la satisfaction d'une fonctionnalité f_i ne conduit pas à générer une autre fonctionnalité f_k avec $k < i$.

Ainsi, un plan de configuration linéaire ne peut exister que si le graphe de dépendances de fonctionnalités F ne contient aucun cycle. Ce graphe de dépendances $G = (V, E)$ se définit de cette manière : V désigne l'ensemble de composants disponibles et E représente l'ensemble des fonctionnalités échangées entre les composants. Nous distinguons entre les fonctionnalités fournies par un composant et celles demandées donnant ainsi un graphe orienté. Afin de calculer le plan de configuration linéaire, nous allons déterminer les composantes fortement connexes. Pour chaque composante, une liste d'adjacence est calculée par fusion des listes d'adjacence des sommets le composant.

Nous réalisons ainsi $O(|E|)$ comparaisons pour la fusion des listes d'adjacence avec $|E| = \sum_{c \in C} |f_c|$ puisqu'une exploration de tous les arcs est nécessaire. Ainsi, le calcul s'effectue en une complexité polynomiale afin de rechercher le plan linéaire. Toutefois, si le nombre de composantes connexes dépasse une composante, ceci signifie qu'un cycle de fonctionnalités existe, ce qui conduit à avoir un ordre qui n'est pas total sur l'ensemble F et par conséquent le plan linéaire n'existe pas.

3.7.2 Complexité du problème de configuration de mashup

Il n'est pas surprenant que notre problème de configuration de mashup appartienne à la classe de problèmes NP-Complet. En effet, il est facile de voir la relation directe entre notre approche et le problème de sous-graphe maximum ayant une propriété p (MAXIMUM EDGE SUBGRAPH) qui est bien connu NP-Complet. Le problème de SOUS-GRAPHE MAXIMAL consiste à trouver un ensemble de k nœuds, de telle sorte que le sous-graphe induit ait la somme maximale des poids. Formellement, soit $G = (V, E)$, une fonction de poids $w : E \to N$ et un entier $k > 0$, un sous-graphe maximal de G est $G' = (V', E')$ où
— $V' \subseteq V$ et $|V'| = k$
— $\sum_{(u,v) \in V \cap (V' \times V')} w(u, v)$

Notre approche, s'appuyant sur le principe de Générer Puis Sélectionner, suggère que la configuration de mashup peut être résolue en créant d'abord des configurations candidates et en sélectionnant ensuite le meilleur sous-ensemble. En fait, nous considérons chaque

configuration comme un nœud d'un graphe où les arrêtes sont pondérées avec les distances de configuration. Ainsi, trouver k configurations qui maximisent la diversité revient à chercher un sous-graphe maximal.

Étant donné que notre problème est NP-difficile, nous intéressons aux heuristiques et aux algorithmes d'approximation. Cependant le problème de sous-graphe maximal ne peut pas être approximé avec des facteurs constants [Bhaskara *et al.*, 2010]. De plus, on croit qu'il est difficile d'approcher dans un facteur polylogarithmique. Nous allons cependant diriger nos efforts vers les heuristiques qui tentent d'utiliser certains algorithmes connus de non-approximité garantie constante. Parmi les heuristiques développées pour le problème de sous-graphe maximal, nous nous sommes référés a celles proposées par [Asahiro *et al.*, 2000, Bhaskara *et al.*, 2010, Feige *et al.*, 2001]. La première d'entre elles peut être appliquée à la version pondérée du problème, tandis que les deux autres ont été développées pour une version non pondérée et peuvent être étendues à la version pondérée en ajoutant un facteur supplémentaire dans le rapport d'approximation de $O(log\ n)$ [Feige *et al.*, 2001]. L'heuristique de [Asahiro *et al.*, 2000] consiste à retirer itérative ment un sommet ayant le degré pondéré minimum dans le graphe restant, jusqu'à ce qu'il reste exactement k sommets. Le rapport d'approximation est de $O(n/k)$. [Bhaskara *et al.*, 2010] et [Feige *et al.*, 2001] proposent d'améliorer le rapport d'approximation jusqu'à $O(n^{1/4+\epsilon})$. Nous avons choisi, comme nous l'avons mentionné précédemment, l'heuristique de [Asahiro *et al.*, 2000] en raison de son applicabilité au cas pondéré. Nous indiquons aussi que dans le cas où la cardinalité des configurations est limitée par une constante le problème peut être résolu en temps polynomial.

Conclusion

Une méthodologie de développement de mashup spécifique au contexte a fait l'objectif de ce chapitre. Nous avons présenté une approche de configuration de mashup qui s'appuie sur un méta modèle de mashup pour injecter les spécificités du domaine. C'est ainsi que nous permettons à une plateforme générique d'être adaptée à un domaine spécifique. Notre approche de configuration de mashup se caractérise par l'adoption d'environnements de conception qui, en accord avec les approches récentes de programmation visuelle des mashups, fait un usage intensif d'abstractions visuelles de haut niveau. Ces paradigmes visuels représentent des composants qui exposent des fonctionnalités permettant de guider l'utilisateur dans le processus de mashup par le biais d'une syntaxe spécifique au domaine.

En effet, nous adoptons un paradigme de composition incrémentale centré sur l'utilisateur non technique qui agit directement sur une interface où il manipule des composants de configuration pour définir le mashup dans une logique de programmation en direct. Il s'agit

d'une intégration à la volée qui, pour chaque action de composition associe une opération d'intégration, générant un retour visuel. Cette intégration des données est fondée sur les opérateurs de configuration qui connectent les instances de services via leurs fonctionnalités dans le cadre d'un framework de configuration de mashup qui masque la complexité typique liée à l'intégration de données. De ce fait, les utilisateurs ne se préoccupent pas de la synchronisation des entrées/ sorties, comme ce qui est proposé par plusieurs outil de mashup, mais plutôt se focalisent sur la définition dans une logique déclarative de la fonctionnalité composite à laquelle le mashup doit répondre. Autrement dit, le mapping des données, qui est considéré complexes pour les utilisateurs finaux, est réalisé par le framework de configuration. Ce dernier s'articule en deux grandes phases afin de générer le mashup. Une première phase de filtrage qui consiste à extraire depuis la requête de mashup visuelle les ressources à intégrer en appliquant un mapping visuel puis invoquer ces ressources afin de sélectionner les instances de services qui vérifient les contraintes utilisateur. Ensuite une application des opérateurs de configuration est effectuée lors de la phase de configuration afin de combiner ces instances et générer un jeu de solutions de configuration.

Les solutions de configurations ainsi obtenues vérifient principalement les caractéristiques de validité et de représentativité. La validité désigne le respect de la configuration des contraintes utilisateurs où le framework applique un traitement de balance qui établit un équilibre entre les fonctionnalités composées du mashup et les exigences initiales de l'utilisateur. La représentativité exprime une notion de couverture de l'ensemble des solutions de configuration rattachée au domaine spécifique. Dans le cadre touristique, nous nous référons à la couverture géographique qui assure une représentativité de la région à visiter. Ceci étant une première formulation de la sensibilité du mashup au contexte de l'utilisateur. Ainsi, il serait avantageux que le modèle de configuration fonctionnel de mashup inclut d'autres dimensions contextuelles de l'intégration afin de mieux répondre au besoins situationnels. Cette adaptation contextuelle fera l'objet du chapitre suivant.

Chapitre 4

Modèle Fonctionnel et Contextuel de Configuration de Mashup

Introduction

Dans le chapitre précédent, nous avons présenté la méthodologie pour le développement d'une approche de mashup spécifique à un domaine qui adopte le paradigme centré sur l'utilisateur final pour l'intégration des données. Le méta-modèle de mashup proposé fournit une plateforme générique qui s'adapte à un domaine particulier par injection des spécificités du domaine. Le principe d'intégration qui caractérise le concept de mashup est assuré par le biais de composants capables de s'intégrer pour former une application plus complexe. En accord avec le paradigme de programmation visuelle, nous avons adopté des abstractions visuelles de haut niveau facilitant l'utilisation de la plateforme de mashup pour les utilisateurs finaux. Ces abstractions masquent, en effet, la complexité typique de la composition et de l'intégration des données. De plus, l'introduction de la syntaxe spécifique au domaine permet d'assurer une manipulation intuitive de la plateforme. Ceci fournit ainsi un environnement de développement efficace pour l'utilisateur final.

Le modèle fonctionnel de mashup proposé dans le chapitre précédent se base sur la technique de configuration qui permet l'agrégation des données en réponse à la requête de l'utilisateur. Il s'articule selon deux grandes phases. Une phase de sélection et de filtrage permettant de déterminer les services à composer en fonction de la requête de mashup. Ensuite, arrive la phase de configuration qui assure la connexion des instances de services par application des opérateurs de configuration. Ces derniers permettent la construction, de manière incrémentale, de la solution de configuration de mashup.

Nous proposons dans ce chapitre d'enrichir notre modèle fonctionnel de configuration de mashup par *la sensibilité au contexte* qui est considérée ici comme une dimension de concep-

tion importante permettant d'identifier les ressources les plus adéquates pour satisfaire les besoins d'information actuels des utilisateurs. Nous présentons donc comment fournir un support pour la prise en connaissance du contexte dans le développement d'applications de mashup afin d'adapter le contenu intégré aux besoins situationnels des utilisateurs.

Le modèle fonctionnel et contextuel que nous proposons vise à masquer la complexité de la conception de l'adaptabilité. Plus spécifiquement, notre modèle permet la transformation de requêtes agnostique de contexte en requête contextuelle par la modélisation explicite des dimensions contextuelles jugées pertinentes dans un domaine spécifique. Il couvre en effet la représentation des propriétés du contexte à travers un modèle contextuel et la définition des abstractions de modélisation de haut niveau pour la conception des comportements adaptatifs.

4.1 Le rôle du contexte dans les applications de mashup

Cette section décrit le rôle donné au contexte, en tant que dimension de modélisation de première classe, pour le développement de mashup. L'injection de cette dimension permet d'identifier les ressources les plus adaptées aux besoins courants des utilisateurs avant de les fusionner pour générer une vue intégrée. Cette adaptabilité assure une évolution dynamique de l'application qui s'appuie sur des données contextuelles génériques non limitées aux données du profil de l'utilisateur.

Particulièrement, notre domaine de référence est influencé par le concept de contexte qui joue un rôle important dans les applications de recommandation de lieux d'intérêt ou une séquence de points d'intérêt[Cassani *et al.*, 2016]. Plusieurs facteurs contextuels, à savoir la dimension spatiale, temporelle ou environnementale, peuvent affecter la recommandation. Par exemple, un utilisateur à la recherche d'un lieu touristique peut être disposé à visiter un lieu proche de son emplacement actuel. En plus de l'information spatiale, l'information temporelle peut aider à déterminer la popularité d'un lieu où les utilisateurs ont tendance à visiter un type de lieu la semaine et d'autres le week-end. En termes de contexte météorologique, il est intuitivement clair que les conditions météorologiques peuvent influencer l'endroit que nous souhaitons visiter. Par exemple, un jour ensoleillé, nous pouvons préférer visiter un parc, alors qu'un jour pluvieux, nous pouvons préférer visiter un musée. Par ailleurs, ces considérations générales sont relatives à la sensibilité individuelle de l'utilisateur à des conditions contextuelles spécifiques [Braunhofer et Ricci, 2016]. Ces données contextuelles peuvent être collectées auprès des capteurs des smartphones des utilisateurs ou par appel aux services Web. Elles sont explicitement spécifiées par l'utilisateur afin de reconnaître sa perception des facteurs contextuels.

Avec l'orientation utilisateur final des applications de mashup, la prise en considération

de la dimension contextuelle doit conserver les mécanismes d'interaction *naturels* favorisant l'intuitivité et correspondant au modèle mental des utilisateurs. Notre objectif ainsi, est de faire évoluer notre approche de configuration de mashup pour supporter la sensibilité au contexte par la conception d'un style d'interaction compatible avec le concept de composition par configuration de composants spécifiques au domaine capables d'établir un équilibre entre la simplicité et l'expressivité.

Le paradigme guidé par les règles assure, en effet, une facilité d'utilisation par les utilisateurs finaux qui peuvent les spécifier, les personnaliser et les maintenir simplement grâce à leur caractère déclaratif. En outre, ce méchanisme offre une expressivité avancée dans l'agrégation de données provenant de diverses sources dans le cadre de mashup [Desolda *et al.*, 2017b].

4.2 Modèle fonctionnel et contextuel de configuration de mashup

Nous décrivons dans cette section comment enrichir notre méta-modèle de mashup afin de supporter la sensibilité au contexte. Pour se faire, nous introduisons un type particulier de composants que nous appelons *composant de contexte* en plus des composants génériques afin de gérer un comportement adaptatif. Cette abstraction de haut niveau, offrant une détection implicite des dimensions contextuelles, est complétée par un mécanisme permettant la spécification explicite de la situation contextuelle, qui exploite le paradigme guidé par les règles. La sous section suivante détaille ces abstractions de gestion des comportements adaptatifs. Quant à la modélisation des dimensions contextuelles, nous exploitons le modèle d'arbre des dimensions du contexte (CDT : Context Dimension Tree) permettant la représentation des perspectives possibles qui caractérisent le contexte au moyen du concept générique de dimension de contexte.

4.2.1 Définition des comportements adaptatifs

En ligne avec notre méta-modèle de mashup, qui s'articule autour du concept de composant offrant une description à base d'abstractions de haut niveau, nous introduisons les composants de contexte. Chaque composant de contexte est responsable d'une dimension contextuelle particulière et par l'agrégation de composants dans le processus de mashup nous obtenons la vue globale du contexte.

Donc au lieu d'avoir un modèle centralisé qui capture les données contextuelles, nous utilisons des composants dédiés qui se chargent d'un élément de contexte. Ces composants de contexte sont caractérisés par des propriétés contextuelles qui seront mapées aux pro-

priétés des autres composants génériques afin d'adapter le comportement du mashup. Un exemple de composant de contexte peut être un composant de localisation capable de détecter la position en termes de longitude et de latitude, ou encore en termes d'adresse. Un composant météo est un autre exemple de composant de contexte qui fournit des prévisions météorologiques par appel à une API de météo.

Comme les composants génériques, les composants de contexte sont spécifiques au domaine. Ils représentent les dimensions contextuelles pertinentes pour le domaine en question. Avec les composants de contexte, nous pouvons fournir un support implicite à la sensibilité au contexte, c'est-à-dire que les événements du contexte sont capturés par ces derniers lors de leur sélection dans la configuration de mashup.

Pour la définition de la situation contextuelle qui permettra de filtrer les services lors de la phase de sélection et de filtrage de notre approche de configuration de mashup, nous optons comme dit précédemment pour le paradigme guidé par les règles. Il s'agit de définir des règles par les utilisateurs finaux afin d'indiquer le comportement adaptatif global par la configuration des propriétés des composants de contexte. Ce paradigme est une technique largement utilisée dans le cadre des systèmes actifs et des tâches d'automatisation [Wajid *et al.*, 2011]. La spécification des règles par les utilisateurs s'appuie sur des notations visuelles adéquates pour les utilisateurs finaux qui ont développé une première expérience avec ces abstractions en exploitant plusieurs outils admettant cette technique de synchronisation événementielle. Offrir une gestion événementielle du contexte pour la configuration de mashup, via des règles de type événement-condition-action, peut réduire la complexité du développement des mashups contextuels en évitant le traitement du flux de données. Ceci est assuré à travers la création de règles dans lesquelles différents types d'événements peuvent activer, en fonction des conditions, des actions indiquant comment le mashup doit s'adapter au contexte détecté.

Ainsi, la Figure 4.1 illustre notre méta-modèle de configuration de mashups sensibles au contexte. L'ensemble des composants est enrichi par le nouveau type qui est le composant de contexte. Il lui est associé un ensemble d'événements qu'il est capable de détecter et un ensemble d'actions à exécuter. La logique déclarative de gestion des comportements adaptatifs du mashup que nous utilisons repose sur les règles. Par conséquent, le mashup est caractérisé par un ensemble de règles décrites au moyen des concepts d'événements, conditions et actions.

De cette manière, nous apportons une extension à la définition du mashup donnée dans le chapitre précédent (Définition 2) afin de tenir en compte du contexte.

Définition 9. (Mashup) : la notion de mashup se voit ici enrichie par un ensemble de règles qui fournissent une gestion événementielle du contexte en appliquant les actions exposées par les composants. Par ailleurs, un mashup m est déterminé par le tuple $m =$

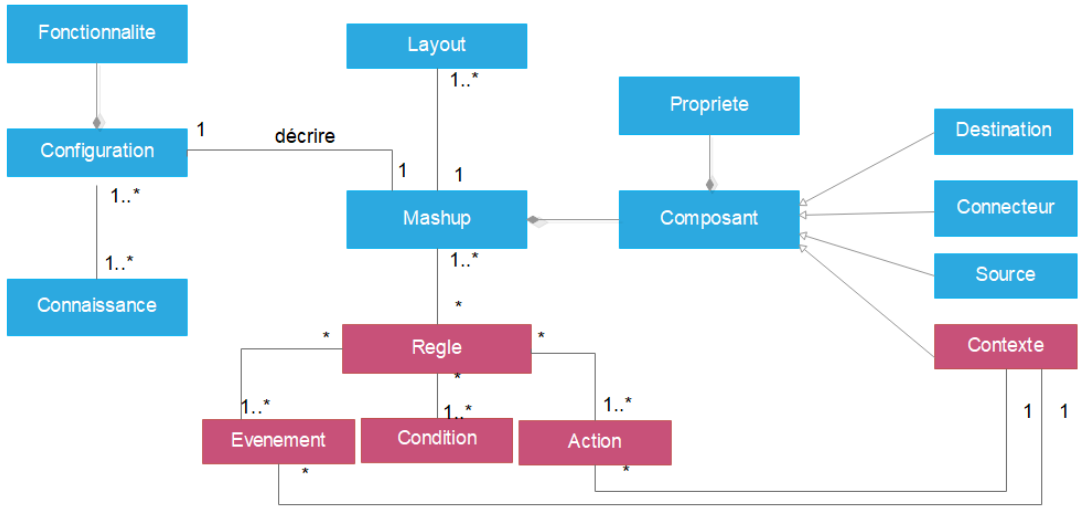


FIGURE 4.1 – Méta modèle de configuration de mashup sensible au contexte

(id, $layout$, C, DF, R) où :

— id : permet d'identifier le mashup,

— $layout$: spécifie l'arrangement graphique des composants,

— C : représentent l'ensemble de composants,

— DF : exprime le flux de données.

— R : désigne l'ensemble des règles qui synchronise l'ensemble des composants C. Chaque règle peut inclure un ou plusieurs composants qui s'interconnectent par le biais des relations de type *Évènement-Condition-Action*.

Définition 10. (Événement) : un événement $e \in \mathcal{E}$, l'ensemble des événements E, représente un changement d'état qu'un composant est capable de détecter. Les valeurs détectées sont sauvegardées dans les propriétés P du composant c auxquelles l'événement se réfère, et peuvent être exploitées pour exprimer des conditions sur l'activation des règles.

Par exemple, le composant *localisation* expose un événement *GetPosition()* qui est détecté par le smartphone du visiteur. La position recueillie par l'appareil est enregistrée dans la propriété position du composant localisation.

Définition 11. (Action) : en réaction aux événements détectés, une action $a \in \mathcal{A}$ fait référence à une fonctionnalité f du problème de configuration de mashup. Les actions engendrent donc un changement d'état au niveau de la configuration du mashup où elles touchent aux valeurs des fonctionnalités globales du mashup.

Par exemple, le composant météo expose l'action filterByWeather(condition) qui permet de filtrer la liste des points d'intérêt en fonction des conditions météorologiques.

Définition 12. (Règle) : une règle $r \in R$ est la combinaison d'événements et d'actions avec les conditions qui jouent un rôle d'activateur. Généralement, une règle exprime que si les conditions spécifiées sont remplies, lors de la détection d'événements, alors les actions sont exécutées. Il s'agit d'un mécanisme de synchronisation entre différents composants que les utilisateurs définissent selon une logique d'intégration pilotée par les événements. Nous représentons une règle r sous la forme d'un tuple $r = (E, CO, A)$ où :
— $r : \mathcal{P}(\mathcal{E}) \times CO \rightarrow \mathcal{P}(\mathcal{A})$
— $E \subseteq \mathcal{E} \rightarrow A \subseteq \mathcal{A}$
— $E = \{e_1, e_2, ..., e_n\} \subseteq \mathcal{E}$
— $A = \{a_1, a_2, ..., a_n\} \subseteq \mathcal{A}$

Un exemple de règle dans notre domaine de référence peut être $r = (E : Weather.Change(), CO : Weather.condition = rainy\ AND\ Situation.type = family, A : GetIndoorPOI())$.

L'introduction de l'aspect contextuel dans le développement du mashup entraîne également une modification dans la formulation du problème de configuration où nous rajoutons une condition dans la définition de la solution de configuration CS.

Définition 13. (Solution de configuration) : la solution $CS = (I, Q)$ est considéré comme une solution de configuration qui respecte le contexte si et seulement si CS remplit les conditions de configuration précisées précédemment et satisfait en plus toute règle $r \in R$.

Ainsi, une nouvelle caractéristique vient compléter l'ensemble des spécificités héritées du modèle fonctionnel de mashup qui est **la cohérence**. Une configuration est dite cohérente si et si seulement elle prend en compte l'aspect contextuel.

4.2.2 Modélisation des dimensions contextuelles

Dans ce qui précède, nous avons décrit notre modèle de configuration de mashup sensible au contexte qui est fondé sur la spécification des règles par les utilisateurs afin de décrire le comportement adaptatif du mashup. Dans le but de guider les utilisateurs dans la définition de règles contextuelles, nous avons besoin de stimuler le modèle mental de l'utilisateur final. Ainsi, il nous est nécessaire de comprendre la perception de la situation contextuelle par l'utilisateur afin de la modéliser, c'est ce que nous allons illustrer dans la suite.

L'être humain a toujours utilisé le concept de contexte, qui fait partie des concepts connus par la majorité de personnes, mais qui sont difficiles à décrire [Casillo *et al.*, 2016]. Il s'articule autour de trois principaux aspects : "où vous êtes ", " avec qui vous êtes " et " quelles sont les ressources à proximité ". Par ailleurs, le contexte peut être désigné par certaines caractéristiques observables. Partant de ceci, la technique 5W-1H [Jang *et al.*, 2005] élargit ces caractéristiques pour représenter toutes les dimensions contextuelles dans un

domaine. Nous nous sommes inspirés, pour la modélisation du contexte, de cette technique qui est employée dans le datajournalisme pour analyser l'histoire complète d'un fait, et en général dans d'autres domaine de la résolution de problème. La technique 5W-1H caractérise le contexte en répondant aux questions suivantes : Qui a fait ça ? (**Who**) ; Que s'est-il passé ? (**What**) ; Quand cela s'est-il passé ? (**When**) ; Où cela s'est-il passé ? (**Where**) ; Pourquoi c'est arrivé ? (**Why**) et Comment s'est –il passé ? (**How**) ; d'où l'appelation 5W-1H.

Le Qui (Who) fait référence au service Web ou au capteur qui détecte l'événement dans quel cas il peut être remplacé par le (Which). Le Quoi (What) spécifie l'action à exécuter. Le Quand (When) et le Où (Where) indiquent respectivement les conditions temporelles et spatialles. Le Pourquoi (Why) permet de donner une explication du comportement. Enfin, le Comment (How) permet d'enrichir les conditions temporelles et spatiales pour inclure des informations sur l'utilisateur, à savoir son état émotionnel, sa situation ou son environnement social.

En général, le concept de règle suit le schéma relationnel cause à effet ($Cause \Rightarrow Effet$) où les causes sont les événements déclenchés par les composants et les effets sont les actions à effectuer sur le mashup afin de supporter le contexte [Casillo *et al.*, 2016]. Nous représentons cette relation par la Figure 4.2 qui décrit la compatibilité de ce schéma avec la méthode 5W-1H, où ces questions se retrouvent des deux côtés de l'équation [Desolda *et al.*, 2017c].

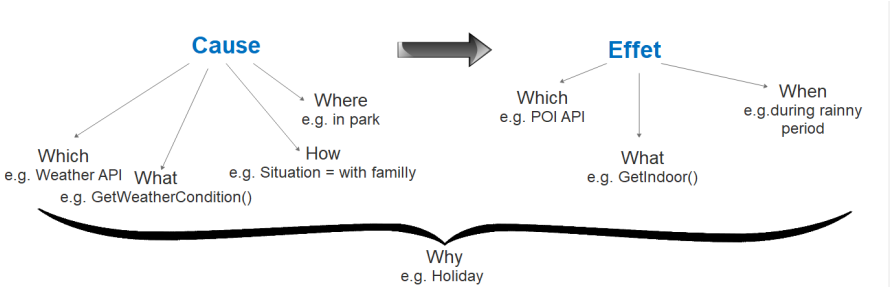


FIGURE 4.2 – Schéma cause à effet d'une règle

De telles questions peuvent guider l'utilisateur à formuler des règles avec des conditions riches permettant l'expression des éléments temporels et spatiaux. C'est pourquoi nous avons opté pour cette méthode afin d'assister les utilisateurs dans la spécification des règles contextuelles qui garantissent le compromis entre la simplicité et l'expressivité. En se basant sur la méthode 5W-1H, nous utilisons le modèle d'arbre de dimensions contextuelles CDT (Context Dimension Tree) afin de représenter le contexte de manière extensible. En effet, nous partons du modèle, que nous appelons Meta CDT, pour modéliser les perspectives qui caractérisent les différentes situations contextuelles dans lesquelles les utilisateurs peuvent se retrouver. Il s'agit d'une structure arborescente composée d'une

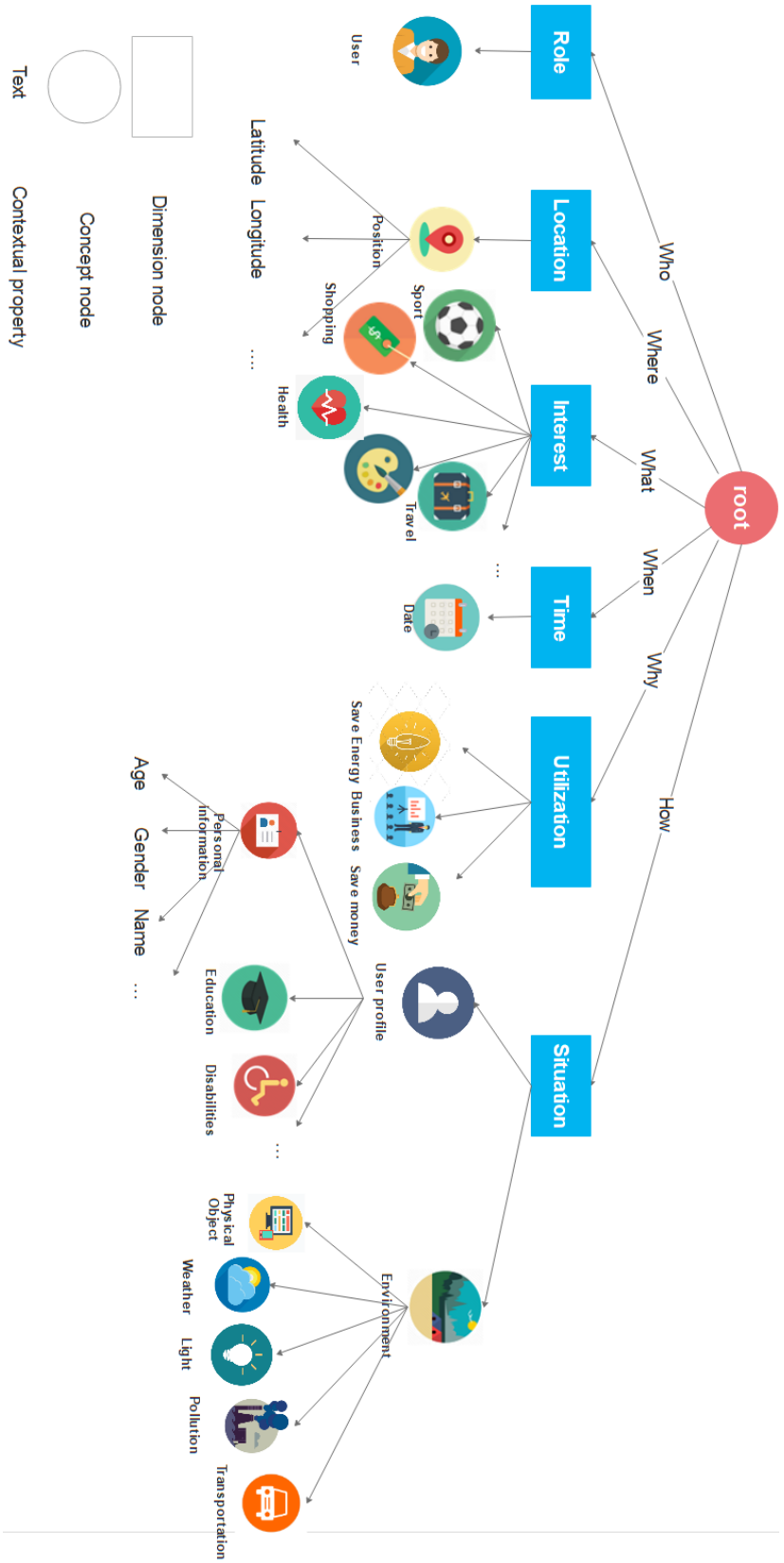


FIGURE 4.3 – Méta modèle d'arbre de dimensions contextuelles

racine qui décrit le contexte et N nœuds qui sont répertoriés en deux catégories, à savoir *les nœuds de dimensions* et *les nœuds concepts*. Les nœuds de dimension, qui sont représentés graphiquement par la couleur bleue (Figure 4.3), décrivent une dimension possible en référence avec les questions 5W-1H . Les nœuds de concepts permettent, eux, d'associer une des valeurs possibles à la dimension. Les nœuds enfants du noeud racine sont tous des nœuds de dimension, ils sont appelés *top dimensions* qui peuvent avoir un sous-arbre représentant les sous dimensions. En lien avec la méthode 5W-1H, les tops dimensions sont : Rôle (Who), Localisation (Where), Intérêt (What), Situation (How), Utilisation (Why) et Temps (When). Les nœuds de concepts, au contraire, doivent être des nœuds intermédiaires. Ils représentent l'espace multidimensionnel du contexte. Les nœuds feuilles expriment des propriétés contextuelles qui incluent des données qui sont soit définies par l'utilisateur soit acquises par les capteurs telles que les coordonnées GPS ou encore retournées par des services Web à savoir les informations météorologiques. Ainsi, nous utilisons différents niveaux d'abstraction pour identifier le contexte de l'utilisateur.

Après l'analyse du domaine, nous pouvons instancier le méta-modèle d'arbre de dimensions contextuelles afin de définir le contexte spécifique au domaine. Il s'agit d'identifier les caractéristiques et les valeurs qu'elles peuvent prendre, respectivement, les nœuds de dimensions et les nœuds de concepts afin de fournir un mashup de services approprié au contexte identifié. Ainsi, nous définissons le modèle d'arbre de dimensions contextuelles spécifique au domaine. La Figure 4.4 décrit une instance de ce modèle pour notre domaine de référence, le tourisme, où la dimension rôle peut prendre comme valeur de concept visiteur ou propriétaire du lieu offrant ainsi à ces derniers la possibilité d'indiquer leurs promotions. La dimension temps divise la journée en quatres intervalles représentés par les concepts matin, midi, après-midi et soir. Nous interprétons la question "Why" par l'objectif visé par le visiteur qui peut exprimer un voyage de travail ou un voyage d'affaire, un voyage de loisir ou encore pour rendre visite à un ami. La dimension situation regroupe principalement des informations sur l'utilisateur, ce qui caractérise son profil et des informations sur son environnement. Le profil de l'utilisateur intègre certaines données personnelles telles que l'âge et d'autres données qui permettent de mieux adapter le mashup des données touristiques au visiteur à savoir avec qui il voyage, ses connaissances de la région ou encore ses disponibilités.

Un élément de contexte est défini comme l'affectation d'une valeur à une propriété contextuelle ($Prop_{context} = val$). Par l'agrégation de ces affectations, nous obtenons la représentation globale du contexte. La structure complexe issue de la composition de ces éléments de contexte se retrouvant au niveau des feuilles de l'arbre des dimensions contextuelles s'appelle une configuration de contexte.

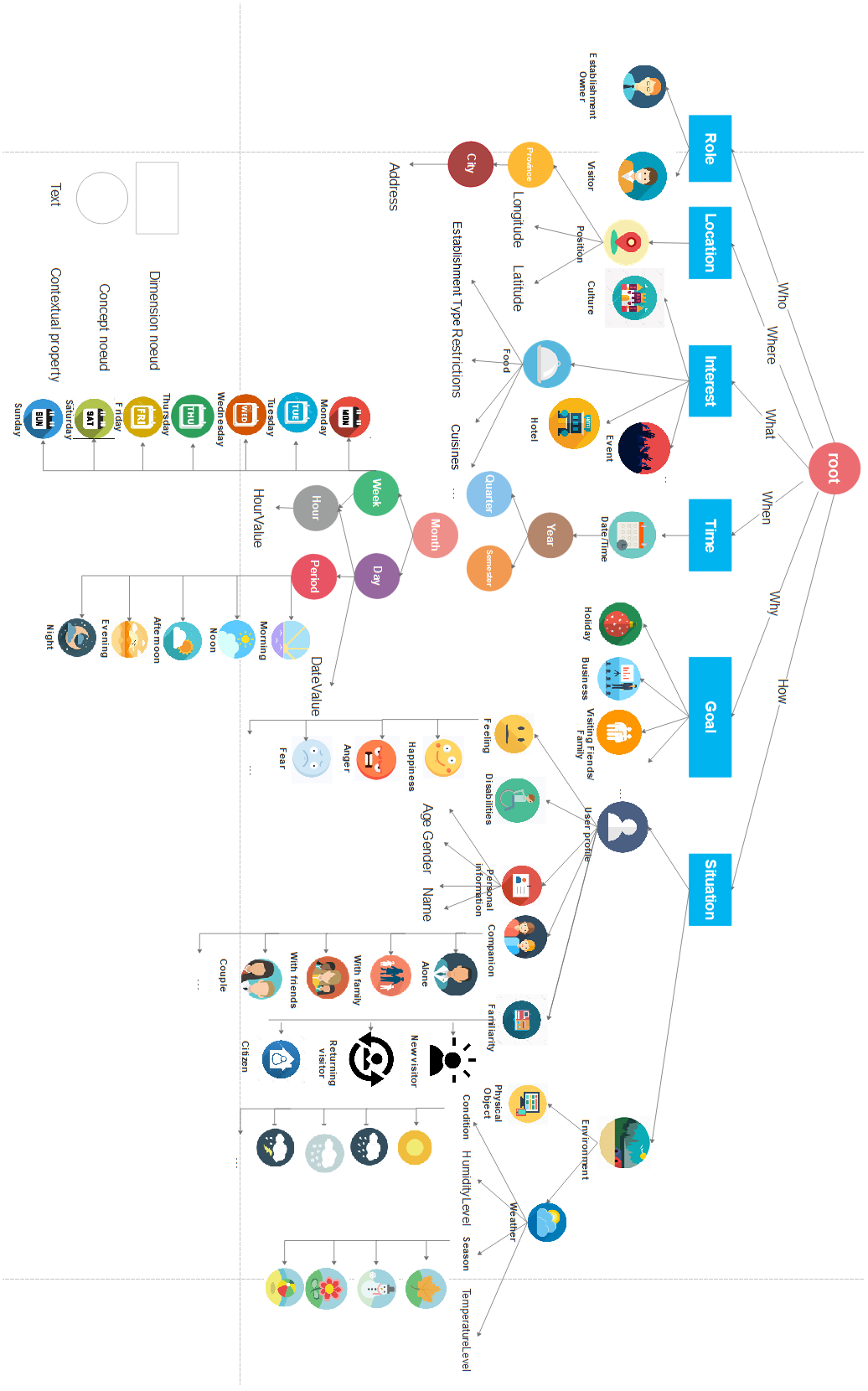


FIGURE 4.4 – Modèle d'arbre de dimensions contextuelles spécifique au domaine

4.3 Framework de configuration de mashup sensible au contexte

Afin de rendre notre framework de configuration de mashup, présenté dans le chapitre précédent, sensible au contexte, nous proposons de l'augmenter par la structure d'arbre de dimensions contextuelles. Plus spécifiquement, cette sensibilité au contexte est gérée par la phase de Pré-sélection qui vise à transformer la requête agnostique au contexte en requête contextuelle permettant d'offrir aux utilisateurs des données et des services personnalisés.

Le processus de configuration de mashup respectant le contexte est illustré par la Figure 4.5. Il commence par interpréter la requête de mashup définie par l'utilisateur, qui exprime la fonctionnalité de mashup recherchée. Une extraction des ressources à composer est réalisée permettant d'établir un schéma agrégateur qui liste les sources de données, en particulier les APIs, à invoquer. Ce schéma est appelé schéma de mashup et est automatiquement généré par le biais du mapping composant-ressource.

Accompagné de ce schéma, une gestion des règles utilisateur, décrivant la situation contextuelle, est effectuée. Il s'agit de transformer ces dernières en format compris par les machines tel que RuleML afin d'extraire les propriétés contextuelles et d'instancier l'arbre des dimensions contextuelles. Chaque nœud de l'arbre est mappé au schéma de mashup afin de procéder à la réécriture de requêtes permettant de dériver des requêtes contextuelles. Étant donné que le schéma de mashup se focalise sur les données indépendamment de leur utilisation, les requêtes sont agnostiques au contexte. Elles doivent être enrichies convenablement d'informations contextuelles via le mapping contexte-ressource.

Nous détaillons dans ce qui suit la phase de pré-sélection et nous conservons le rôle des autres phases du framework.

4.3.1 Pré-sélection et requête contextuelle

La phase de pré-sélection vise à extraire les informations contextuelles provenant des capteurs de contexte qui peuvent être des APIs Web, des capteurs des smartphones des utilisateurs ou encore des données utilisateurs pour les injecter dans la requête de mashup. Ceci passe par deux étapes : (i) la gestion des règles contextuelles pour en extraire les éléments contextuels puis (ii) l'établissement d'un mapping de ressources pour produire un un schéma appelé schéma de mashup regroupant les ressources sélectionnées et masque les problèmes d'accès aux données. Le schéma de mashup sera le point d'entrée à la phase de sélection qui collecte les données.

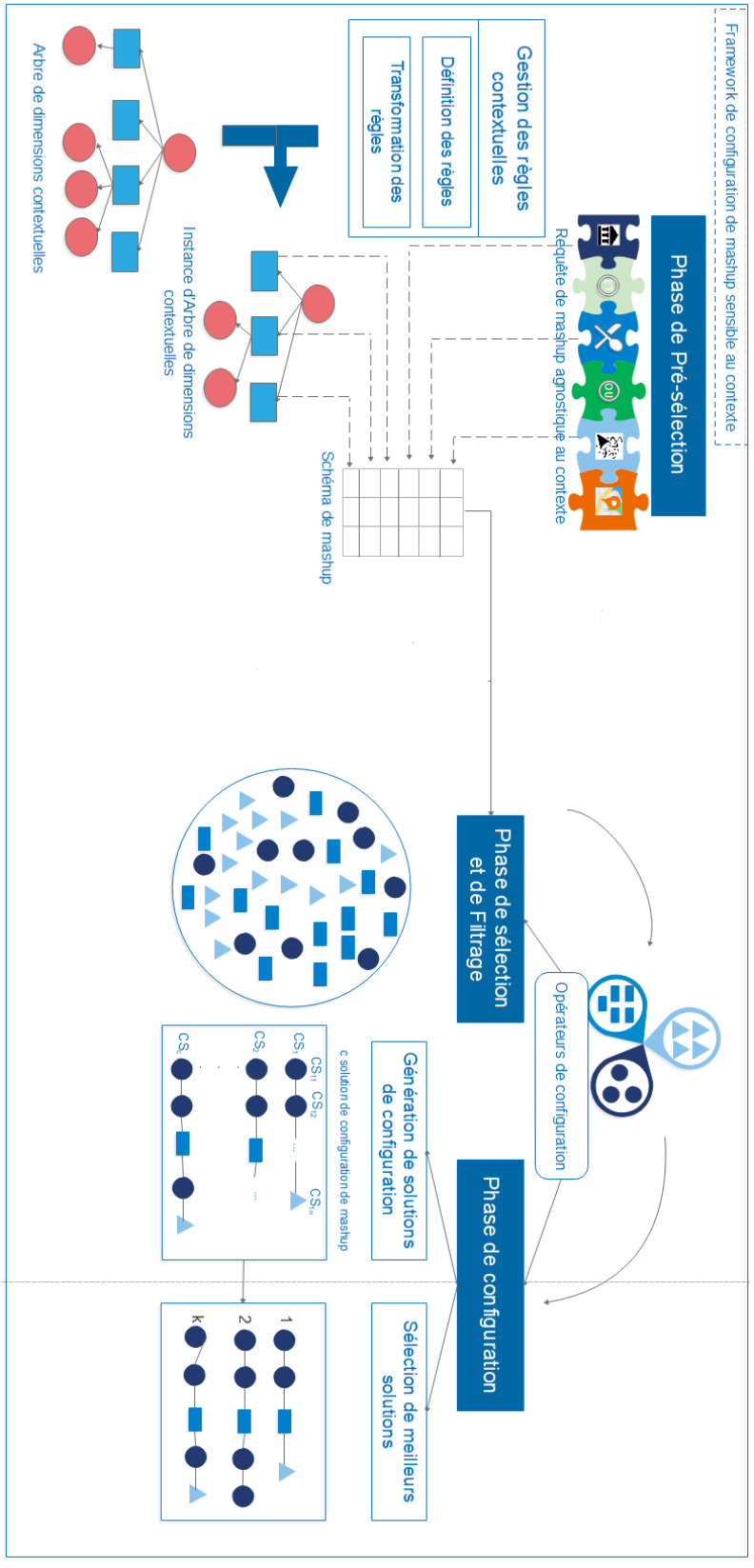


FIGURE 4.5 – Framework de configuration de mashup sensible au contexte

4.3.1.1 Gestion des règles contextuelles

La gestion des règles contextuelles est constituée de deux sous étapes : la spécification des règles par l'utilisateur puis la transformation des règles en format compréhensible par la machine.

Spécification des règles

La situation contextuelle est décrite à travers un ensemble de règles spécifiées par les utilisateurs en utilisant des notations graphiques. En effet, nous offrons des métaphores de contrôle permettant aux utilisateurs de personnaliser leurs mashups en fonction de leurs besoins situationnels.

Afin de se rapprocher du modèle mental de l'utilisateur et de le guider dans la création des règles, nous proposons une conception visuelle qui ne suit pas exactement la syntaxe des règles événement-condition-action. La particularité de notre syntaxe visuelle est que nous structurons la règle contextuelle en deux parties. Une première partie est constituée de conditions représentant une concaténation logique de sous-conditions qui se focalisent sur l'occurrence des événements et sur les situations permettant l'activation des conditions. La seconde partie se rapporte aux actions à exécuter qui ne peuvent être activées que si toutes les conditions de la partie précédente de la règle sont remplies. Autrement, nous nous référons au schéma relationnel cause à effet et nous regroupons les conditions et les événements qui retracent la cause. Ceci s'explique par le fait que les utilisateurs finaux ne sont pas sensibles à la différence entre l'expression des conditions et la spécification des événement déclencheurs. D'ailleurs, les plateformes, qui ciblent les utilisateurs finaux telles que Attoma [1] et IFTTT [2] admettent cette syntaxe visuelle.

Ainsi, la spécification des règles inclut des opérateurs logiques pour l'agrégation de conditions et d'actions. Toutefois, il est reconnu que les relations logiques utilisant des opérateurs de conjonction, de disjonction et de négation représentent un formalisme adapté au modèle mental des experts [Desolda *et al.*, 2017c], tandis que les utilisateurs finaux pourraient ne pas être en mesure de les exploiter correctement. Généralement, ils ont tendance à interpréter l'opérateur « OU » comme un « OU exclusif ». Afin de remédier à ce problème, nous proposons un système qui masque cette complexité où l'opérateur « ET » de conjonction est exprimé par le label « TOUTES LES CONDITIONS », alors que le label « AU MOINS UNE CONDITION » désigne la sémantique de l'opérateur « OU » en langage naturel.

De cette manière, les utilisateurs seront capables de définir des règles contextuelles en

1. https ://atooma.com/
2. https ://ifttt.com/

regroupant les conditions et les actions par le biais des opérateurs logiques. La figure 4.6 expose la maquette de spécification de règles.

FIGURE 4.6 – Maquette de définition de règles

Transformation des règles

Une fois que les règles contextuelles sont spécifiées, un processus de transformation est effectué permettant de traduire la syntaxe visuelle en format interprétable par la machine. A cette fin, un langage de représentation des règles est nécessaire. Nous avons choisi le langage RuleML et plus spécifiquement, Reaction RuleML qui est considéré comme la représentation quasi standard des règles de réaction [Paschke *et al.*, 2012, Boley *et al.*, 2010]. Il s'agit d'un sous langage sérialisé XML convivial de RuleML implémentant les règles événementielles de type raisonnement dérivé IF/THEN/ELSE, règles de production IF/DO, ou de type événement-condition-action ON/IF/DO, ou encore une variation de ON/IF/DO et IF/THEN qui décrit une représentation des connaissances. Les raisons qui nous ont motivé à choisir ce langage de règles est sa capacité à supporter différents types de règles déductives, réactives et normatives ce qui le rend proche du langage naturel.

En effet, RuleML est suffisamment structuré pour empêcher l'utilisateur de définir des expressions absurdes, mais aussi assez expressif pour assurer de la flexibilité. En outre, il fournit une interopérabilité entre différents systèmes sur le Web en offrant une représentation générique des règles. Il convient de mentionner aussi que le langage SWRL pourrait également être utilisé, sauf que puisqu'il est basé sur l'hypothèse du monde ouvert, qui n'inclut pas la négation par défaut, nous préférons un langage de règles plus explicite. L'objectif

principal de la représentation des règles dans ce langage est de simplifier le processus de traduction des règles dans un format lisible par la machine.

Afin que les règles puissent être interprétables par une machine, une traduction de la représentation RuleML en règles Jess est réalisée. Jess a été choisi à cet effet car il s'agit d'un moteur de règles léger qui s'adapte bien avec les technologies Web [Viktoratos *et al.*, 2014b]. Il est basé sur une version augmentée de l'algorithme Rete qui utilise des techniques d'indexation adaptative pour améliorer les performances. [Viktoratos *et al.*, 2015] ont montré également que Jess est l'un des meilleurs langage de règles réactives. Nous effectuons cette transformation des règles représentées dans la syntaxe RuleML au format Jess en utilisant le langage de transformation XSLT.

Le but de cette conversion est d'extraire les propriétés contextuelles définies par les règles, mais avant cela, nous traitons les problèmes reliés aux règles, à savoir le conflit de règles et la terminaison. Nous distinguons principalement deux catégories de règles : celles définies par les visiteurs afin de décrire la situation contextuelle qu'ils souhaitent que le mashup considère, et celles spécifiées par les professionnels du domaine qui exposent par exemple leurs offres promotionnelles.

En ce qui concerne le problème de conflit, les règles des professionnels (fournisseurs de services) sont éventuellement plus susceptibles d'être contradictoires par rapport aux règles des visiteurs. Par exemple, une règle décrivant une réduction le mardi pour les étudiants est en contradiction avec celle qui expose une notification de fermeture exceptionnelle le mardi. Pour résoudre les conflits, nous proposons une stratégie utilisant le mécanisme des priorités qui adapte la technique « salience » de Jess. En effet, contrairement au fonctionnement habituel de Jess qui propose d'exécuter la règle ayant la valeur d'importance la plus élevée en premier, nous laissons celle-ci en dernier ce qui nous permet de persister son effet. C'est-à-dire que les règles contradictoires traitées en premier seront annulées par l'effet de la dernière règle. Alors que les contradictions au niveau des règles visiteurs qui décrivent des préférences sont rares. Nous pensons qu'une approche souple suivant une logique de recommandation plutôt qu'une optique de décision est plus appropriée. Ainsi, nous ne traitons pas dans la phase actuelle de notre framework ces cas rares.

Quant au problème de terminaison, il ne peut pas y avoir un cycle d'activation de règles vu qu'elles sont conditionnées par les propriétés contextuelles des composants et agissent sur les fonctionnalités globales du mashup.

Nous présentons par la Figure 4.7 un exemple de spécification de règle qui décrit une situation contextuelle exprimant « Si je suis avec ma famille et qu'il risque de pleuvoir alors je préfère faire des activités à l'intérieur ». La Figure 4.8 illustre une représentation RuleML de cette règle, tandis que la Figure 4.9 interprète une représentation Jess.

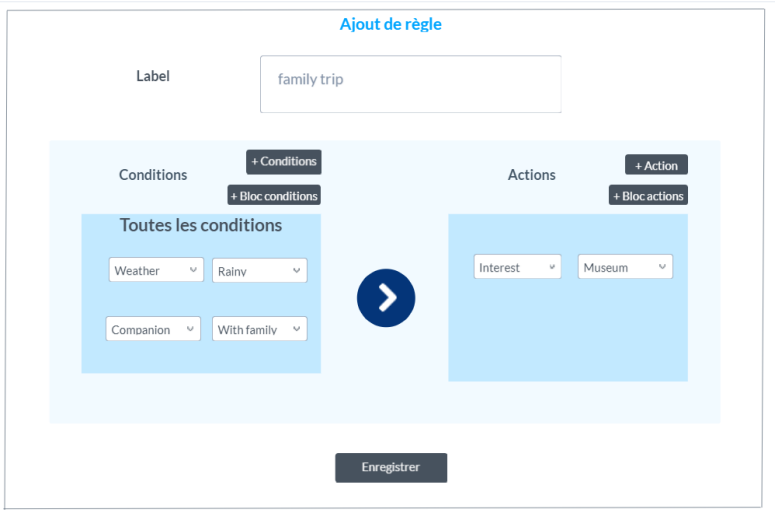


FIGURE 4.7 – Exemple de spécification de règle

RuleML Assert Rule explanation

```
<?xml version="1.0" encoding="UTF-8"?>
<RuleML xsi:schemaLocation="http://www.ruleml.org/0.91/xsd"   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Assert>
        <Rule style="active">
            <label>family trip</label>
            <explanation>I would like to visit museum if it is a rainy day and I am with family </explanation>
            <if>
                <and>
                    <Atom>
                        <Rel>place</Rel>
                        <slot>
                            <Ind>type</Ind>
                            <Ind>Museum</Ind>
                        </slot>
                        <slot>
                            <Ind>uri</Ind>
                            <Var>id</Var>
                        </slot>
                    </Atom>
                    <Atom>
                        <Rel>situation</Rel>
                        <slot>
                            <Ind>companion</Ind>
                            <Ind>withFamily</Ind>
                        </slot>
                        <slot>
                            <Ind>weather</Ind>
                            <Ind>rainy</Ind>
                        </slot>
                    </Atom>
                </and>
            </if>
            <then>
                <Assert>
                    <Atom>
                        <Rel>Recommendation</Rel>
                        <slot>
                            <Ind>id</Ind>
                            <Var>id</Var>
                        </slot>
                    </Atom>
                </Assert>
            </then>
        </Rule>
    </Assert>
</RuleML>
```

FIGURE 4.8 – Représentation RuleML

```
(defrule FamilyTrip
    (place(type Musuem) (uri ?id))
    (situation (weather rainy)(companion withFamily))=>
    (assert(recommendation(id ?id)))
)
```

FIGURE 4.9 – Représentation Jess

4.3.1.2 Mapping de ressources et Schéma de Mashup

Après avoir converti les règles en syntaxe compréhensible par la machine, l'étape suivante consiste à extraire les informations contextuelles afin de les injecter ensuite dans la requête de mashup.

Pour cette fin, nous nous basons sur le modèle d'arbre de dimensions contextuelles qui organise selon une structure arborescente les données contextuelles. Il s'agit d'instancier ce modèle avec des valeurs concrètes qui proviennent des règles définies par l'utilisateur ce qui permet de se concentrer uniquement sur les aspects contextuels qui concernent le plus l'utilisateur. Il convient à noter aussi que certaines données, notamment les conditions météorologiques, sont issues de l'invocation des APIs Web. Ainsi, nous avons besoin de recenser la provenance de toute information contextuelle. A cet effet, nous construisons un schéma de ressources que nous appelons schéma de mashup. Ce dernier vise à unifier la représentation des ressources utilisées pour le mashup. Il s'agit d'un répertoire de métadonnées qui garde trace de la provenance des ressources. Ainsi, chaque attribut du schéma contient l'identifiant de la ressource et les paramètres d'accès à celle-ci. Ces derniers comprennent le point d'entrée à l'API, l'opération demandée, les paramètres d'entrée ainsi que la sérialisation des données de sortie (XML, JSON).

Ce mapping des ressources concerne à la fois les ressources contextuelles et les ressources basiques. Il est appelé respectivement mapping de base pour le premier type de ressources et mapping contextuel pour le second.

Le mapping contextuel permet d'indiquer pour les éléments contextuels dynamiques (provenant des APIs Web) comment récupérer ces données. Quant aux éléments statiques qui sont exprimés par les utilisateurs sous forme de constantes, nous gardons la valeur constante spécifiée à la place des paramètres d'accès. Par conséquent, chaque nœud représentant une propriété contextuelle du sous-arbre obtenu par instanciation est associé à une entrée du schéma de mashup.

Toutefois, pour certaines propriétés contextuelles telles que la période, nous utilisons les termes annotés, ce qui permet aux utilisateurs de comprendre la sémantique des données. Par ailleurs, le mapping doit tenir compte de ces termes symboliques afin de pouvoir invoquer correctement les ressources. C'est-à-dire que nous transformons ces derniers en valeurs réelles numériques qui seront utilisées ensuite dans l'interrogation des services lors de la

configuration du mashup. La Figure 4.10 présente un exemple d'extrait de transformation de l'encapsulation des termes symboliques décrivant la période en intervalle de temps.

FIGURE 4.10 – Mapping des propriétés contextuelles exprimées en termes symboliques

En plus du mapping contextuel, le schéma de mashup inclut les ressources non contextuelles issues du mapping des composants visuels de la requête de mashup avec les services ou les APIs Web concrètes. C'est le mapping de base qui permet de retrouver les services de base fournissant les principales données qui constituent la fonctionnalité de mashup recherchée. Ainsi chaque composant visuel est associé à une entrée dans le schéma de mashup qui indique comment accéder aux services.

Ici, nous précisons que notre approche de mashup ne s'appuie pas exclusivement sur les caractéristiques fonctionnelles des services ou sur la compatibilité des paramètres. Au contraire, la spécification des exigences contextuelles permet d'abord le filtrage progressif des services, puis la sélection des données pour tenir en compte des besoins fonctionnels et situationnels. Ceci est réalisé grâce au fait que le schéma de mashup, qui peut être exprimé en langage XML, se concentre sur l'intégration des données indépendemment de leurs utilisations.

Une fois que le schéma est établi, il est possible de dériver la requête agnostique au contexte en requête contextuelle. Cette dernière sera interprétée durant la phase de sélection afin de déterminer l'ensemble d'instances de services candidates au processus de configuration. Lors de la phase de sélection, les opérateurs de configuration de type Filter et Rank sont appliqués. C'est principalement l'opérateur Filter qui est exécuté afin de filtrer les données en fonction du contexte déterminé. Pour certaines propriétés contextuelles, telles que la position de l'utilisateur, nous appliquons plutôt l'opérateur Rank qui permet de trier les données, ce qui nous semble plus pertinent que de filtrer les données et passer à côté de celles potentiellement intéressantes.

Il convient aussi de faire remarquer que le modèle fonctionnel de configuration de mashup applique aussi des opérateurs de filtre pour tenir compte des contraintes utilisateur. Même si elles paraissent proches des règles contextuelles, les contraintes de configuration

se concentrent sur les exigences globales de mashup. A titre d'exemple, « maximum 3 châteaux » concerne le nombre d'instances de châteaux dans tout le mashup.

4.4 Conclusion

Dans ce chapitre, nous avons montré comment notre modèle fonctionnel de configuration de mashup est enrichi d'une couche de sensibilité au contexte. Le modèle fonctionnel et contextuel de mashup ainsi obtenu fait usage d'abstraction visuelle de haut niveau qui, en masquant la complexité de la tâche de développement, guide les utilisateurs finaux dans la description de leurs besoins situaltionnels.

En effet, la spécification de la situation contextuelle suit le paradigme événementiel utilisant les règles de type événement-condition-action. Il offre un compromis entre l'expressivité et la simplicité puisqu'il correspond au modèle mental de la plupart des utilisateurs. Afin de pouvoir adapter le mashup au besoin contextuel des utilisateurs, nous transformons cette syntaxe visuelle en langage compréhensible par la machine. Ainsi, nous réalisons une conversion des règles contextuelles abstraites en règles définies en format RuleML puis en syntaxe Jess à des fins de résolution des conflits potentiels. Ensuite, l'information contextuelle extraite est exploitée dans le modèle d'arbre de dimensions contextuelles qui permet la représentation de toutes les perspectives possibles caractérisant le contexte au moyen du concept générique de dimension de contexte. Ce modèle conceptuel, dérivé de la technique 5W-1H, est employé pour construire le schéma de mashup. Ce dernier, décrivant un répertoire de ressources, permet la transformation de la requête initialement agnostique au contexte en requête contextuelle.

Par conséquent, notre approche de configuration de mashup, pilotée par les règles contextuelles, assure une représentation concise et lisible à la fois par l'homme et la machine des dimensions et propriétés contextuelles pertinentes pour le domaine d'application du mashup afin d'exposer une composition de services personnalisée. Néanmoins, cette perspective de mashup personnalisé, où la sensibilité au contexte est considérée comme un premier pallier dans une approche de personnalisation, peut être complétée par un modèle de connaissances. En se basant sur des ontologies fournissant plus de flexibilité dans l'adaptation, il serait possible d'envisager d'offrir aux utilisateurs une certaine reconfiguration leur permettant d'ajuster le mashup construit. Nous proposons dans le chapitre suivant notre modèle de reconfiguration qui assure cet objectif.

Chapitre 5

Modèle de Reconfiguration à base de Données Liées pour le Mashup Personnalisé

Introduction

Après avoir présenté le modèle de configuration contextuelle de mashup, nous continuons à tracer, dans ce chapitre, la suite de la problèmatique de personnalisation de mashup. En effet, conformément à l'innovation axée sur l'utilisateur, le désir de ces derniers de créer leurs propres applications pourrait être fourni par des approches de mashup. De plus, un besoin croissant de remplacer les applications fixes par des environnements ***élastiques*** qui peuvent être conçus de manière flexible afin de répondre à différents besoins situationnels. Si l'activité de composition par le mashup s'avère d'apporter une nouvelle valeur ajoutée intéressante, un autre avantage pour les utilisateurs est qu'ils peuvent permettre de contrôler les applications afin de répondre exactement à leurs propres besoins.

La possibilité pour l'utilisateur final de sélectionner les services pertinents, de les interroger et d'agréger le contenu récupéré, et surtout la possibilité de définir et de personnaliser des modèles visuels rend l'approche globale élastique. Jusqu'à présent, les systèmes ont été conçus comme des ensembles pré-emballés de données, de fonctionnalités et de visualisations que les développeurs ont implémenté pour les utilisateurs finaux. Les systèmes élastiques s'écartent de cette idée et tentent de promouvoir des paradigmes de flexibilité et de modularité où les contenus et les fonctionnalités peuvent être étendus facilement au moment de l'utilisation. En d'autre termes, l'élasticité est une opportunité d'accommoder des besoins de personnalisation multiples et variables, déplaçant la responsabilité aux utilisateurs finaux de créer leurs propres applications.

Ceci conduit à la conjecture suivante *l'humain reste la clé de la personnalisation.* Cette optique de personnalisation est réalisable par un modèle de reconfiguration fournissant un mashup que l'on pourrait qualifier d'extensible. Le scénario de reconfiguration implique ainsi une certaine modification du mashup existant en le complétant avec des composants, en supprimant des composants, en modifiant les propriétés d'un composant ou les propriétés globales du mashup. Il s'agit d'un modèle de mashup avec des composants actualisables permettant de satisfaire les attentes des utilisateurs. Cette perspective est proche des techniques de gestion de variabilité qui visent une solution de reconfiguration corrective des services dans le but de la tolérance aux pannes. Le processus réactif cherche à construire un pool d'alternatives possibles pour reconfigurer le système en dysfonctionnement. En vue de s'adapter, les systèmes activent ou désactivent certaines caractéristiques dans le cadre de scénarios prédits et bien définis. Contrairement à cela, nous traitons la variabilité dans une optique plus large où elle est corrélée à la capacité de personnaliser un système en envisageant certains changements. Nous utilisons ainsi les termes d'élasticité et d'extensibilité plutôt que le terme de variabilité.

Les solutions pour le domaine de tourisme nécessitent d'intégrer une grande adaptabilité afin de fournir aux utilisateurs une personnalisation flexible de l'offre proposée qui améliore l'expérience numérique. Cependant, aucune des approches de recommandation existantes n'offre une solution qui couvre l'ensemble du processus de composition, depuis l'identification du besoin de mashup jusqu'à l'intégration du contenu dans une plateforme d'agrégation avec un degré de flexibilité et de configurabilité à but de personnaliser le service composite. Ainsi, nous visons à augmenter notre approche de mashup sensible au contexte par des technique d'élasticité fournissant des mécanismes de reconfiguration. Cette extension permettra de lier l'adaptabilité contextuelle et l'adaptabilité des fonctions pour fournir des services hautement personnalisables.

A cet égard, des méthodes de représentation des connaissances basées sur des ontologies peuvent permettre cette adaptabilité dans une logique d'utilisation plus engageante, stimulante et attractive de l'information. Par conséquent, nous émettons l'hypothèse que l'exploitation des normes du Web sémantique pour permettre une combinaison automatique de données LOD (Linked Open Data) et d'API Web pourrait déclencher de nouveaux scénarii de fertilisation de mashups personnalisés par reconfiguration.

5.1 Mashup et Données Liées

La puissance des mashups réside dans la capacité de combiner plusieurs ressources afin de répondre à des besoins complexes. Cette capacité d'agrégation pour l'unification de l'information a mis en évidence plusieurs avantages en faveur des utilisateurs finaux,

n'ayant pas des connaissances en programmation, grâce à des représentations abstraites. Ceci, couplé aux données liées, permet de cerner des relations plus complexes entre les objets à un niveau sémantique plus profond créant des fonctionnalités et des modèles plus informatifs.

En effet, les données ouvertes liées (LOD) représentent un important entrepôt de données structurées de sorte qu'elles puissent être liées entre elles en fonction de leur signification sémantique. Depuis son apparition par [Berners-Lee, 2006], LOD est en constante évolution pour devenir une gigantesque source de connaissances couvrant différents domaines. Les données LOD sont généralement présentées sous forme de ressources identifiées par des URI, où chaque ressource est décrite, par des informations sémantiquement structurées, sous la forme de triplets RDF (Resource Description Framework). Ces triplets sont stockés dans une base RDF accessible via des requêtes SPARQL dédiées. Comme les URIs peuvent être utilisées pour définir des ressources uniques dans le Web, il est facile de relier les ressources LOD les unes aux autres. La base de connaissances qui en résulte est structurée et interconnectée, ce qui permet aux systèmes d'information d'analyser et d'exploiter leurs ressources à un niveau sémantique.

Cette richesse est utile pour le processus de mashup où cette connexion vise à augmenter considérablement la valeur de ces ensembles de données, fournissant ainsi un moyen puissant pour la mise en œuvre de solutions personnalisées. Grâce à l'initiative Linked Open Data, de plus en plus de données sémantiques sont publiées selon les principes du Linked Data, qui permettent d'interconnecter les données dans différentes sources d'information créant, ainsi un seul espace global de données, appelé le Web des Données[Bizer *et al.*, 2011]. En considérant la possibilité de parcourir le graphe des données, le mashup peut ainsi être enrichi, ce qui permet de participer à son efficacité.

Nous retrouvons diverses utilisations des technologies LOD dans plusieurs systèmes expérimentaux, conçus dans le but d'explorer le potentiel du Web sémantique, mais aussi dans des applications pratiques de type moteur de recherche, navigateur ou application spécifiques [Sansonetti, 2019]. En particulier, dans le domaine du tourisme ou plus largement le domaine de l'héritage culturel, les technologies sémantiques, ont exploité différentes taxonomies et catalogues dans le but de gérer efficacement des données.

Par exemple, Foursquare [1] fournit une hiérarchie de catégories pour organiser les lieux touristiques dans un réseau social. Cette hiérarchie a été exploitée par de nombreux chercheurs [Bao *et al.*, 2015, Chen *et al.*, 2014, Lim *et al.*, 2015]. Cependant, la catégorisation n'est pas assez granulaire. Par exemple, la catégorie site culturel n'est pas détaillée, aucune sous-catégories ne lui est associée, il faut réaliser plusieurs requêtes sur différentes catégories pour collecter les données relatives aux sites culturels à cause de la dispersion

1. https ://developer.foursquare.com/docs/data

de l'information. [Lu *et al.*, 2016, Sansonetti *et al.*, 2019] exploitent les données du *GeoNames*[2] pour affiner les recommandations de services par des données spatiales. Une autre alternative, *Europeana*[3], est utilisée pour améliorer les recommandations des lieux de visite en naviguant dans les ensembles de données pour retourner les concepts ayant le plus haut niveau de similarité sémantique avec l'URI d'entrée. Il s'agit de plus de 54 millions d'objets du patrimoine culturel. Le vocabulaire *Schema.org*[4] a été aussi utilisé dans plusieurs travaux, notamment [Viktoratos *et al.*, 2017], permettant la description d'entités de différents types avec des sous-classes, ainsi que des attributs et des relations entre entités, suivant les principes du Web des données. Dans un esprit similaire, plusieurs travaux tirent partie plutôt de la base de connaissances LOD de *DBpedia* [Auer *et al.*, 2007] qui est l'ensemble de données le plus populaire sur Linked Open Data Cloud. A partir d'une liste d'URI de ressources artistiques et culturelles sur DBpedia, une recommandation propose des lieux partageant les mêmes catégories sémantiques que les ressources en entrée visant à améliorer l'expérience de l'utilisateur. Toutefois, nous pensons que l'applicabilité inter-domaines de ces vocabulaires et ontologies soulève la nécessité d'accompagner son adoption de spécifications claires afin qu'ils puissent répondre aux exigences spécifiques du domaine touristique.

A l'échelle de la France, les acteurs du tourisme ont créé en premier la norme *TourInFrance* (TIF)[5] afin de faciliter l'échange des données touristiques. Les principaux systèmes d'information touristique français tels que Raccourci Interactive[6], TourinSoft[7] et Sitra[8] ont adopté le standard TIF se basant sur les technologies XML pour faciliter la publication d'informations sur le Web et l'échange d'informations entre systèmes. Avec la création des technologies Web et la démocratisation des données ouvertes, cette norme est devenue désuète et elle a perdu ses inter-compatibilités. Par conséquent, *DATAtourisme*[9] a vu le jour en tant que plateforme nationale d'agrégation des données touristiques. Mise en ligne en 2018, ce projet national, piloté par la Direction générale des Entreprises du Ministère de l'Économie et des Finances en partenariat avec Tourisme et Territoires, est alimenté par les professionnels du tourisme. Dans une logique de *guichet unique*, DATAtourisme [Soualah-Alila *et al.*, 2016] vise à fédérer et rende compatibles les données entre elles afin de les diffuser en données ouvertes pour faciliter la création d'applications innovantes.

Ainsi, nous exploitons cette ontologie du tourisme qui a été spécifiquement développée pour s'adapter au mieux à la réalité du secteur, dans notre modèle de reconfiguration de

2. https ://www.geonames.org/
3. https ://pro.europeana.eu/page/linked-open-data
4. https ://schema.org/docs/documents.html
5. https ://www.tourisme-territoires.net/tourinfrance/
6. http ://www.raccourci.fr/
7. http ://www.tourinsoft.com/
8. http ://www.sitra-tourisme.com/
9. https ://info.datatourisme.gouv.fr/

mashup. L'idée ici est de tirer partie des données structurées qui s'alignent bien avec les technologies des APIs Web. Il nous semble donc naturel de combiner leurs avantages afin de fournir des mashups personnalisés.

5.2 Modèle de reconfiguration de mashup à base de données liées

Nous proposons d'augmenter notre approche de configuration de mashup avec un modèle de reconfiguration qui s'appuie sur une couche sémantique utilisant les données ouvertes et liées afin d'offrir aux utilisateurs la possibilité de personnaliser leur mashup.

Partant de l'initiative DATAtourisme, nous constatons que ce modèle ontologique reste limité, bien qu'il promet un potentiel significatif. En effet, certains concepts et relations nécessaires pour la planification touristique sont manquants, ce qui s'explique par le fait que DataTourisme est encore à ses débuts. Par exemple, le concept « *VisitDuration*», qui permet de représenter la durée de visite d'un lieu d'intérêt, n'est pas modélisé dans l'ontologie DATAtourisme. A cet égard, nous envisageons de l'enrichir par une combinaison avec les données provenant des services et des APIs Web. Ceci assure une couche sémantique plus riche sur laquelle s'applique une mesure de similarité capable de calculer des alternatives de composants du mashup dans le cadre d'un processus de reconfiguration. Nous profitons, en effet, de la corrélation étroite entre le mashup et les connaissances sémantiques à travers les données liées afin de booster la personnalisation du mashup par les utilisateurs où ils reconfigurent par eux-même leurs compositions.

L'idée ici est de soutenir les utilisateurs par des services avancés et hautement personnalisables. Étant donné que l'information est directement accessible, notre modèle de reconfiguration de mashup assure au-delà de l'exploitation des services liés une solution contrôlable permettant de répondre aux besoins de personnalisation très spécifiques de l'utilisateur. Par exemple, les touristes qui cherchent à réaliser une visite des sites culturels au fil du temps ou ceux qui souhaitent explorer les lieux traçant la vie d'un roi,etc.

Concrètement, le modèle de reconfiguration de mashup est structuré en deux grandes phases, comme il est illustré dans la Figure 5.1. D'abord, la phase d'enrichissement qui a pour objectif de créer un ensemble riche de données liées via un processus d'intégration léger qui permet de compléter les données décrites dans l'ontologie DATAtourisme. Ce processus collecte les données des sources non sémantiques à savoir les APIs Web où des mécanismes d'homogénéisation des données est effectué. Ensuite, cet ensemble est complété pas des données provenant des sources dans le LOD cloud afin d'affiner l'ensemble résultant. Les données enrichies seront, ensuite stockées dans un référentiel de données sémantiques.

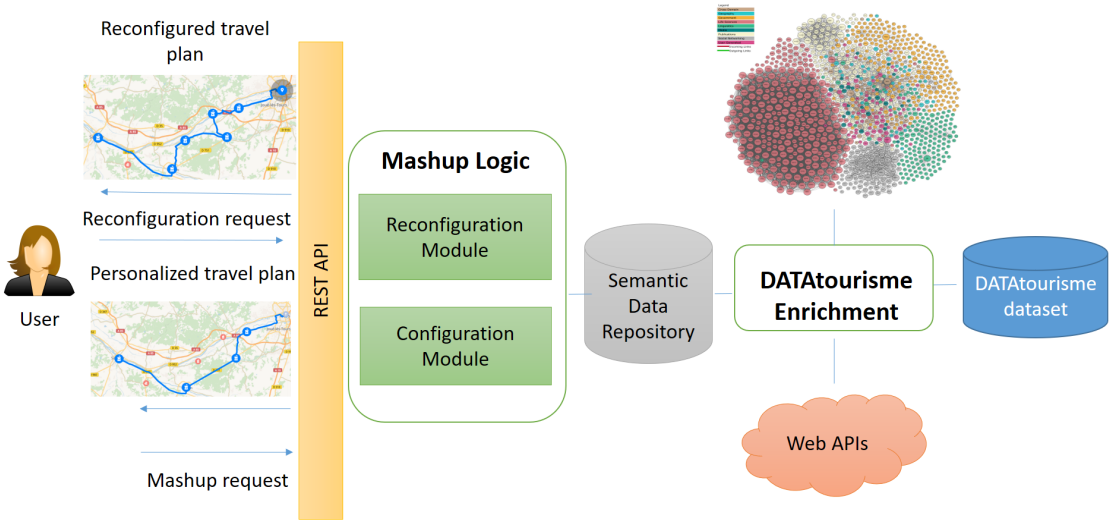


FIGURE 5.1 – Modèle général de reconfiguration de mashup

La seconde phase consiste en un processus de reconfiguration qui permet aux utilisateurs de réajuster les composants du mashup. En s'appuyant sur une mesure de similarité LOD, des alternatives aux composants sont générées pour la personnalisation. En effet, ceci suppose de partir du mashup déjà construit pour le modifier en fonction des besoins de l'utilisateur, tout en tenant compte des restrictions du domaine. Particulièrement, les systèmes de recommandation touristique personnalisés se basent sur des approches de simple filtrage de trajectoires rarement interactifs. Généralement, ces approches visent à fournir une rétroaction sur l'ensemble des lieux suggérés afin de l'utiliser pour les prochaines recommandations de sorte qu'elles soient plus utiles. Dans notre cas, le processus de reconfiguration vise au contraire à personnaliser le mashup des points d'intérêt lui-même à travers un ensemble d'opérations pour manipuler la customisation de la composition dans une logique interactive. Cette interactivité reflète les préférences des utilisateurs qui ne peuvent pas être exprimées dès les premières interactions avec l'application et ceci pour deux raisons : premièrement, demander aux utilisateurs d'indiquer toutes les variantes est très contraignant, et deuxièmement, les utilisateurs ne pourront peut-être exprimer leurs préférences qu'une fois ils auront découvert les alternatives disponibles. Il s'agit d'une *reconfiguration à la demande* en appliquant l'utilisation intelligente des données ouvertes pour l'intégration. Le détail de ce modèle de configuration fera l'objectif des sous-sections suivantes.

5.2.1 Enrichissement de DATAtourisme

La phase de construction d'un ensemble de données sémantiques s'appuie sur le noyau DATAtourisme qui est enrichi par des données prvenant de différentes sources. Avant de présenter le processus d'enrichissement, nous décrivant le modèle ontologique de DATAtourisme et nous soulignant les manques que nous complétons.

5.2.1.1 Ontologie DATAtourisme

DATAtourisme s'appuie sur un ensemble d'ontologies et vocabulaires standards afin d'assurer l'interopérabilité entre plusieurs sources de données. Il s'agit par exemple de «*Schema.org*» qui est utilisée pour la modélisation de la localisation et des tarifs des lieux de visite, ou encore «*ACCO*» Accommodation ontology qui est exploitée pour la représentation des accomodations. Le vocabulaire TIF est aussi interconnecté à DATAtourisme afin de pouvoir intégrer les données des systèmes TourinFrance.

La Figure 5.2, représentant le schéma global du modèle ontologique DATAtourisme, est centrée sur le concept **PointOfInterst** (POI) qui décrit un élément touristique. Autour de ce concept, gravitent quatre sous-classes à savoir le concept **Product** représentant le produit ou le service consommé par le point d'intérêt, le concept **Tour** encapsule un ensemble de points d'intérêt structurant un itinéraire. Les deux autres sous-classes décrivent deux types de points d'intérêt, le premier type **EntertainmentAndEvent** représente un point d'intérêt à caractère temporel et le second type **PlaceOfInterest** modélise le point d'intérêt à caractère spatial.

Les informations sur les points d'intérêt conçus en termes de relations sémantiques se répertorient en deux catégories : réflexives et classiques. Les relations réflexives telles que **hasPart** et **isAPartOf** représente des POIs composés. Quant aux relations classiques, nous retrouvons la relation **isLocatedAt** décrivant la localisation d'un lieu, la relation **hasTheme** pour repésenter les thèmes associés au POI, la relation **hasShortOrLongDescription** exposant une description sur le lieu, ainsi que la relation **provides** qui à travers le concept **Offer** agrège les informations de prix, mode de payement et horraires d'ouverture.

Bien que ce schéma apparaît complet, nous proposons d'affiner plus la modélisation d'un point d'intérêt par l'ajout des deux concepts suivants. Le concept **PopularTime** permettant de décrire les horaires d'affluence qui seront utilisés dans la planification et le concept **Duration** qui expose la durée de visite moyenne du point d'intérêt. Le concept central décrit est entouré en rouge et les nouveaux concepts rajoutés sont représentés en vert sur la figure 5.2.

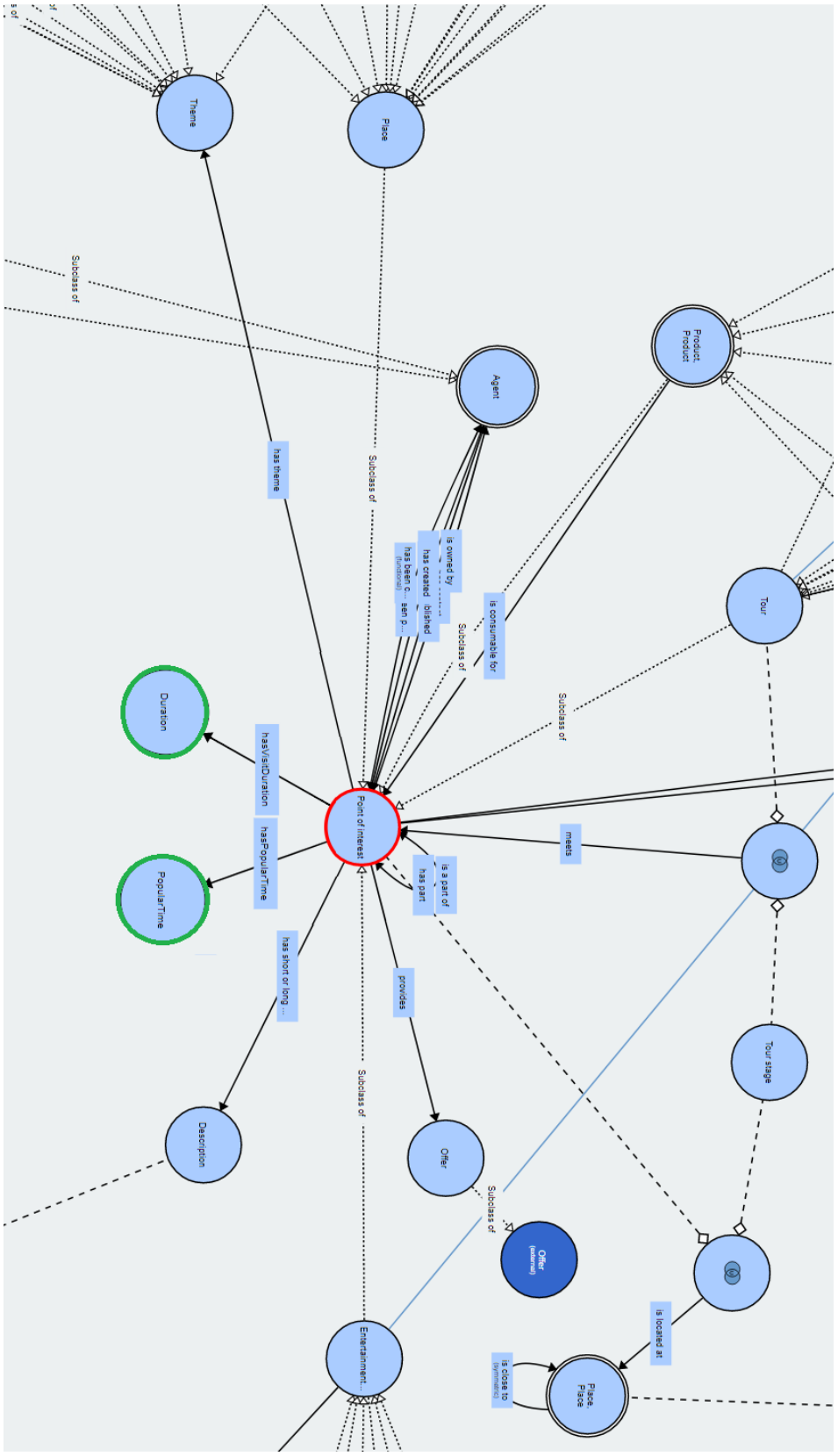


FIGURE 5.2 – Schéma ontologique global de DATAtourisme

Nous détaillons dans la suite les trois concepts **Place**, **EntertainmentAndEvent** et **Tour**.

La modélisation d'un point d'intérêt géo-localisé est représentée par le concept **Place**. La Figure 5.3 décrit un schéma global de ce concept qui est dérivé en différentes sous-classes à savoir, la classe **BusinessPlace** modélisant les lieux à titre commercial, la classe **MedicalPlace** qui représente les lieux médicaux, etc. Dans une logique de raffinement de la description de l'héritage culturel, nous proposons de rajouter les relations **isResidenceOf**, **isDeath** et **isBirth** qui permettent de modéliser les occupants des lieux d'intérêt. Nous avons choisi de détailler les classes **Transport**, **Accomodation**, **CulturalSite**, **NaturalHeritageLandform** et **FoodEstablishment** car ce sont celles que nous utiliserons le plus dans notre modèle de mashup pour le tourisme culturel.

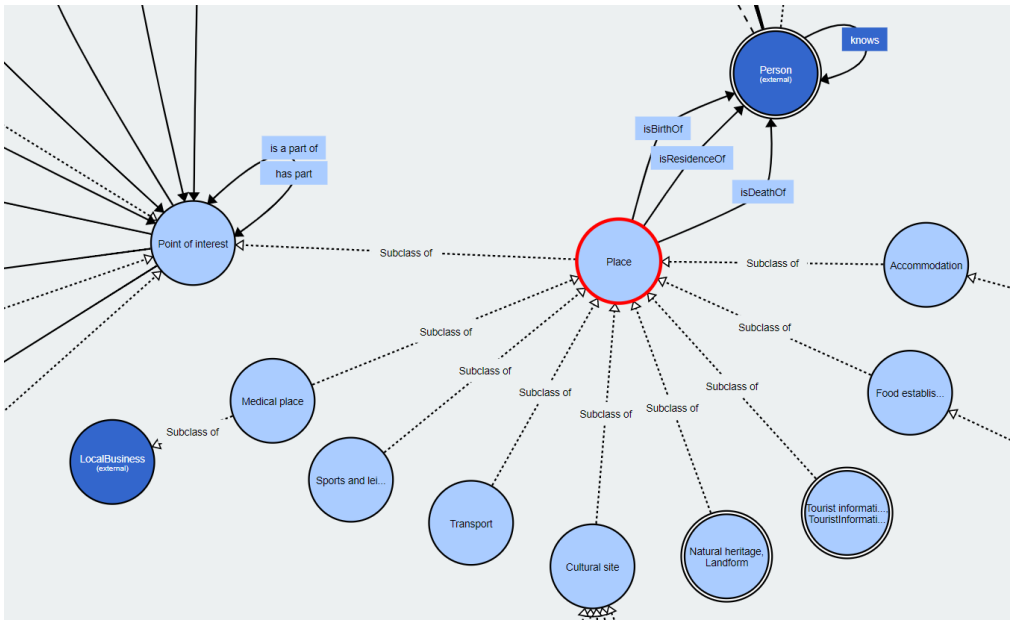


FIGURE 5.3 – Modèle du concept Place

Le concept **Transport**, représenté par la Figure 5.4, modélise tous les types de transport utilisé pour le déplacement. Cependant, nous remarquons l'absence de la prise en compte du transport personnel. Ainsi, nous proposons d'enrichir la liste des sous-classes du concept **Transport** par un nouveau concept **PersonalTransport**.

Le concept **Accomodation** regroupe aussi différents types d'hébergement : l'hébergement en plein air, l'hébergement collectif, l'hébergement locatif, etc, comme montre la Figure 5.5. De même, nous remarquons l'absence de l'hébergement personnel et celui du type "airbnb" qui est devenu une solution très recherchée ces dernières années ceci nous a conduit à rajouter ces deux concepts : **PersonalAccommodation** sous-classe de **Ac-**

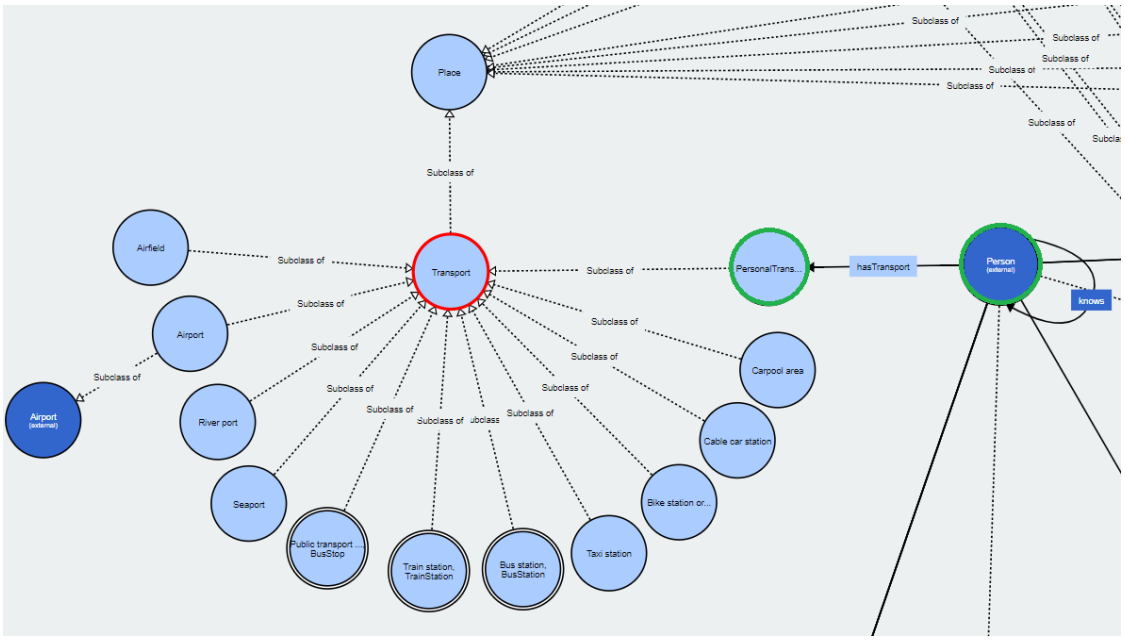


FIGURE 5.4 – Modèle du concept Transport

commodation et le concept **Airbnb** en tant que sous-classe de **RentalAccomodation**.

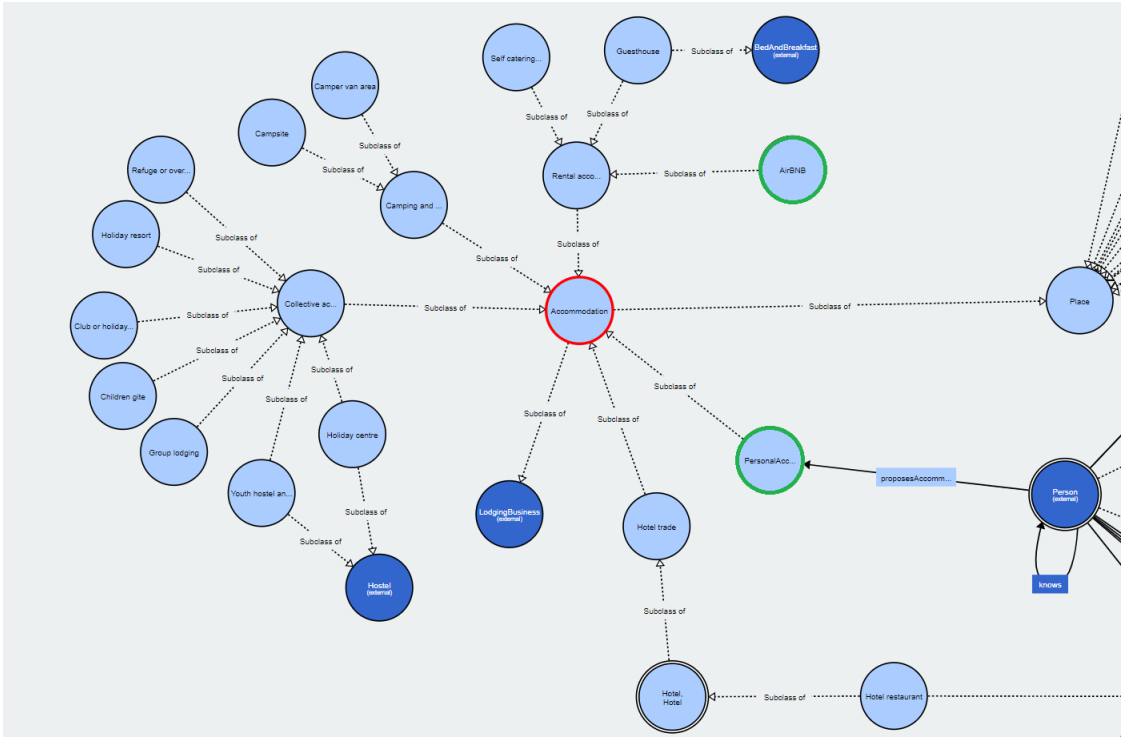


FIGURE 5.5 – Modèle du concept Accommodation

Les lieux de type site culturel sont modélisés par le concept **CulturalSite**, où nous retrouvons les musées, les sites archéologiques, les lieux de mémoire, les sites religieux et d'autres types comme illustré dans la Figure 5.6. Le style architectural d'un site culturel est représenté par la relation **hasArchitecturalStyle** permettant d'offrir une description fine sur les lieux culturels. Quant aux lieux d'intérêt à caractère naturel, ils sont modélisés par le concept **NaturalSite**, décrit dans la Figure 5.7, qui illustre une panoplie de sites naturels.

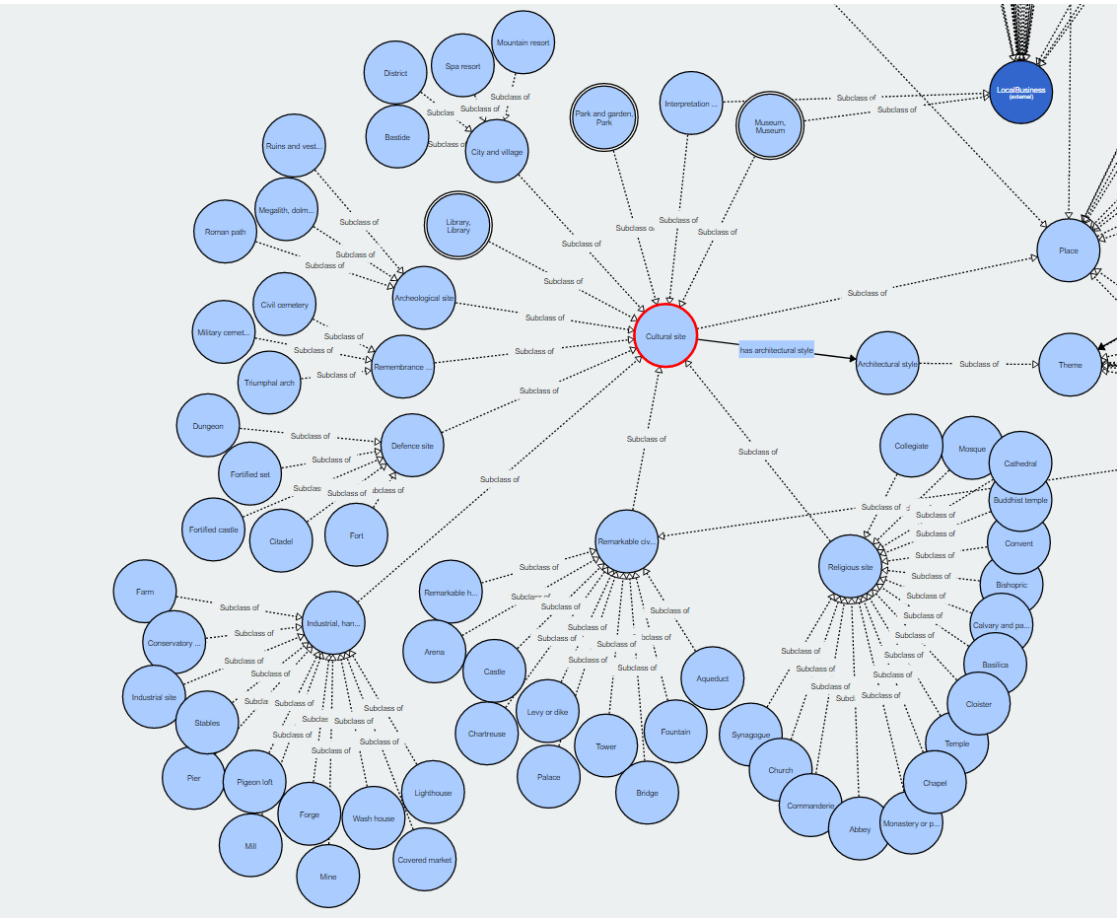


FIGURE 5.6 – Modèle du concept CulturalSite

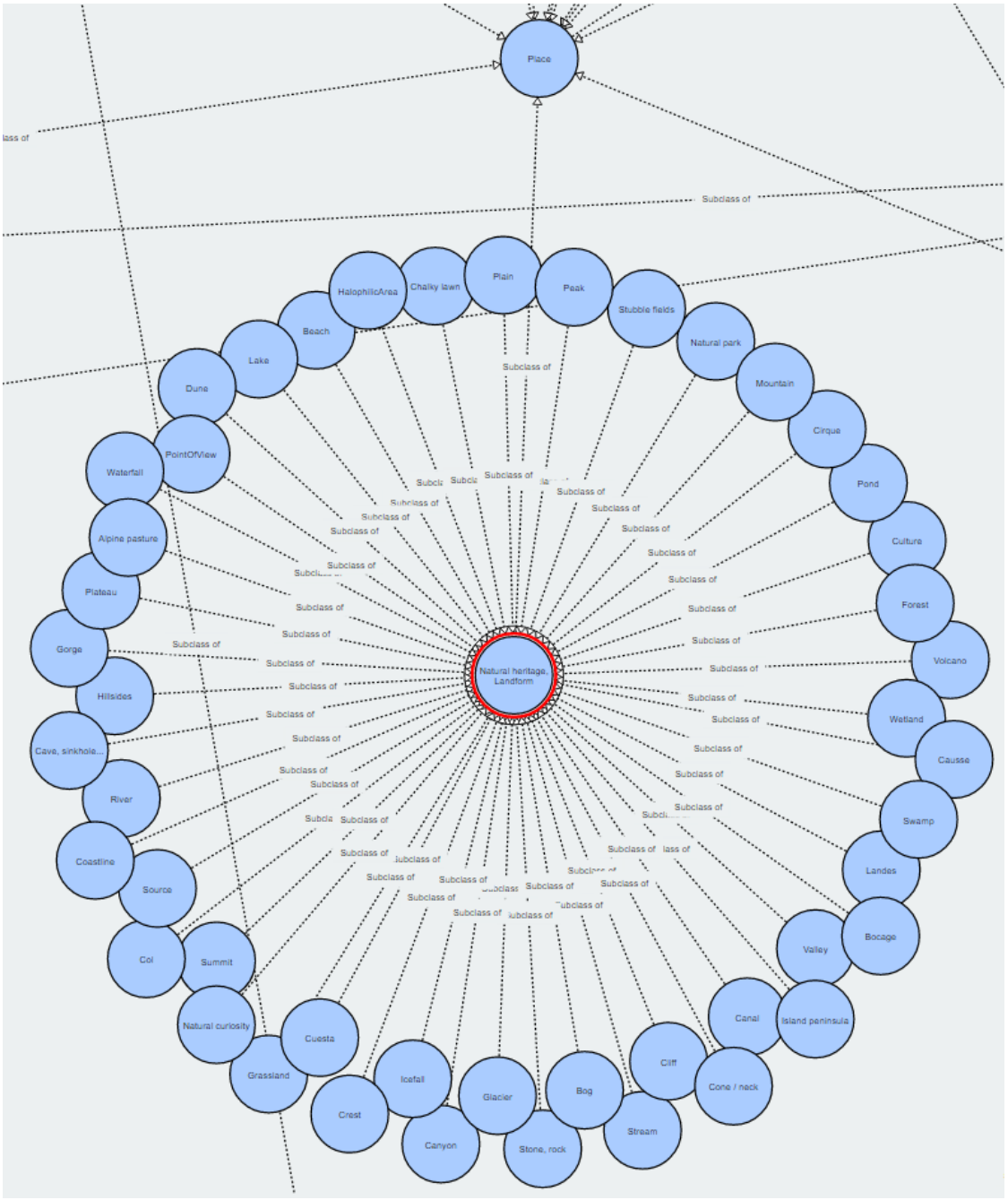


FIGURE 5.7 – Modèle du concept NaturalSite

Le concept **EntertainmentAndEvent**, décrit par la Figure 5.8, représente les événements de type sportif, commercial, social, culturel et d'autres. Ce concept dérivé de la classe **PointOfInterest**, est caractérisé par la relation **takesPlaceAt** qui illustre la période de déroulement de l'événement. De plus, nous proposons de rajouter la relation **takesPlaceIn** afin de représenter des événements géo-localisés.

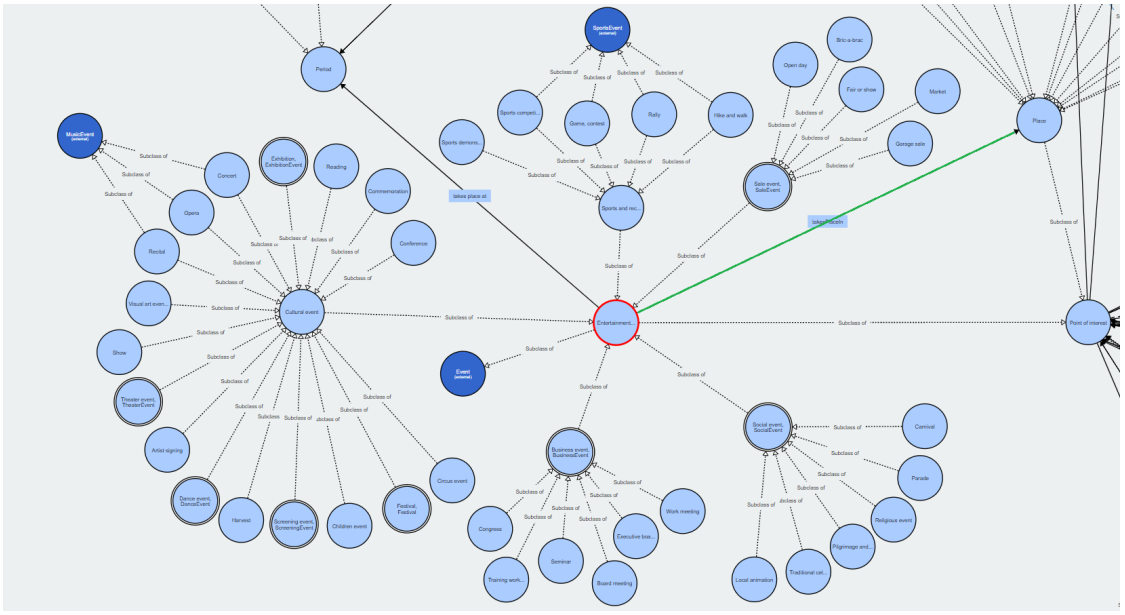


FIGURE 5.8 – Modèle du concept EntertainmentAndEvent

La Figure 5.9 suivante décrit le concept **Tour** représentant un itinéraire touristique sous forme d'étapes de POIs modélisées par le concept **TourStage**. La notion d'itinéraire touristique est conçue en tant que liste ordonnée de POIs géo-localisés. Nous proposons de rajouter la dimension temporelle à l'itinéraire via le concept **Period**.

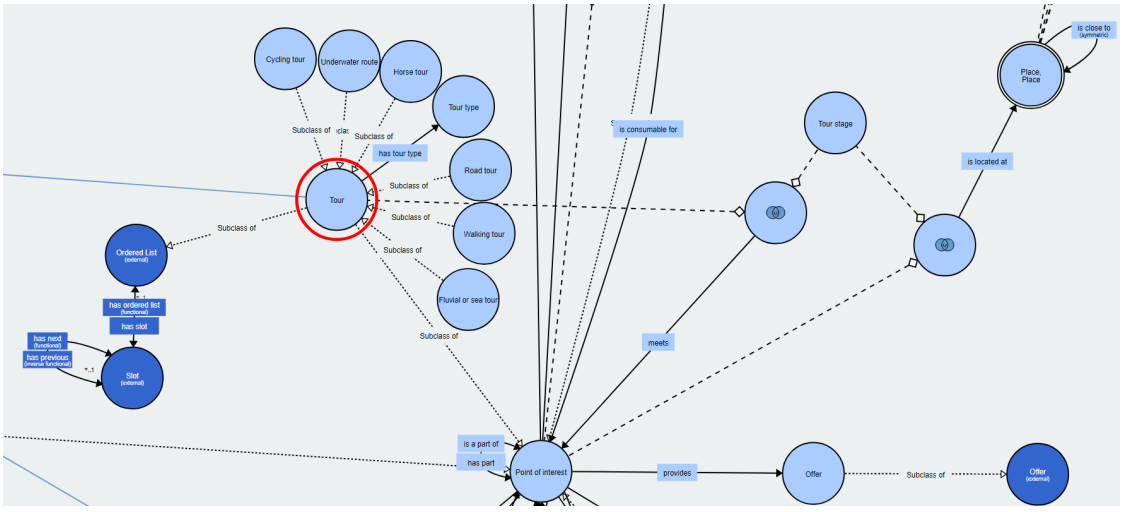


FIGURE 5.9 – Modèle du concept Tour

Après avoir augmenté le modèle ontologique de DATAtourisme au niveau schéma, nous proposons de l'enrichir au niveau instance par un processus de Mapping d'APIs et un processus de Mapping LOD afin de construire un ensemble riche et complet de données

touristiques.

5.2.1.2 Enrichissement par un Mapping d'APIs

L'enrichissement du dataset DATAtourisme par le processus de mapping d'APIs Web vise à compléter cet ensemble de données par des données provenant d'autres APIs. Ce processus comprend deux étapes : l'extraction et l'uniformisation des données en raison de leur hétérogénéité, et leur intégration ensuite au dataset.

Pour la première étape, appelée aussi la sémantisation des données, nous utilisons le modèle de données RDF (Resource Description Framework) qui sert de "*lingua franca*" pour l'interopérabilité sémantique et l'intégration de données aux formats hétérogènes. L'objectif est de générer des triplets RDF à partir de sources de données issues des APIs qui sont au format XML ou JSON. Le langage SPARQL-Generate, proposé par [Lefrançois *et al.*, 2017], répond à ce besoin d'exploitation des données hétérogènes. Il s'agit d'une extension du langage SPARQL, permettant l'interrogation d'un ensemble de documents, qui bénéficie de l'expressivité et de la flexibilité de ce langage. En guise d'illustration, nous présentons dans la Figure 5.10 un exemple de transformation des données RDF en se basant sur SPARQL-Generate.

Les différentes représentations générées par SPARQL-Generate nécessitent une intégration pour construire une représentation unifiée. Pour ce faire, nous utilisons l'outil LDIF (Linked Data Integration Framework), proposé par [Schultz *et al.*, 2012], permettant de recueillir des données liées à partir du Web et de traduire les données recueillies en une représentation locale propre. LDIF inclut un composant d'accès aux ressources sémantiques du Web et un composant de résolution d'identité qui découvre les alias URI dans les données d'entrée et les remplace par un URI cible unique basé sur une heuristique d'appariement flexible. LDIF contient aussi un module d'évaluation de la qualité des données et un module de fusion de données qui permettent de filtrer les données Web selon différentes politiques d'évaluation de la qualité des données et de fusionner les données Web en utilisant différentes méthodes de résolution de conflits.

Dans notre cas le mapping de vocabulaires est une tâche relativement simple en raison de l'utilisation d'un seul schéma qui est celui de DATAtourisme. Par contre, la tâche d'agrégation des données est plus complexe. Pour cette étape, LDIF s'appuie sur deux modules permettant l'évaluation de la qualité des données et la fusion des données fournies de l'outil Sieve. Ce dernier, proposé par [Mendes *et al.*, 2012], est en mesure d'évaluer la qualité des données en attribuant à chaque graphe nommé dans les données traitées un ou plusieurs scores de qualité basés sur des politiques d'évaluation de la qualité configurables par l'utilisateur. Ces scores forment la fonction d'évaluation qui sera utilisée par le module de fusion pour décider comment fusionner les valeurs de propriétés conflictuelles en une

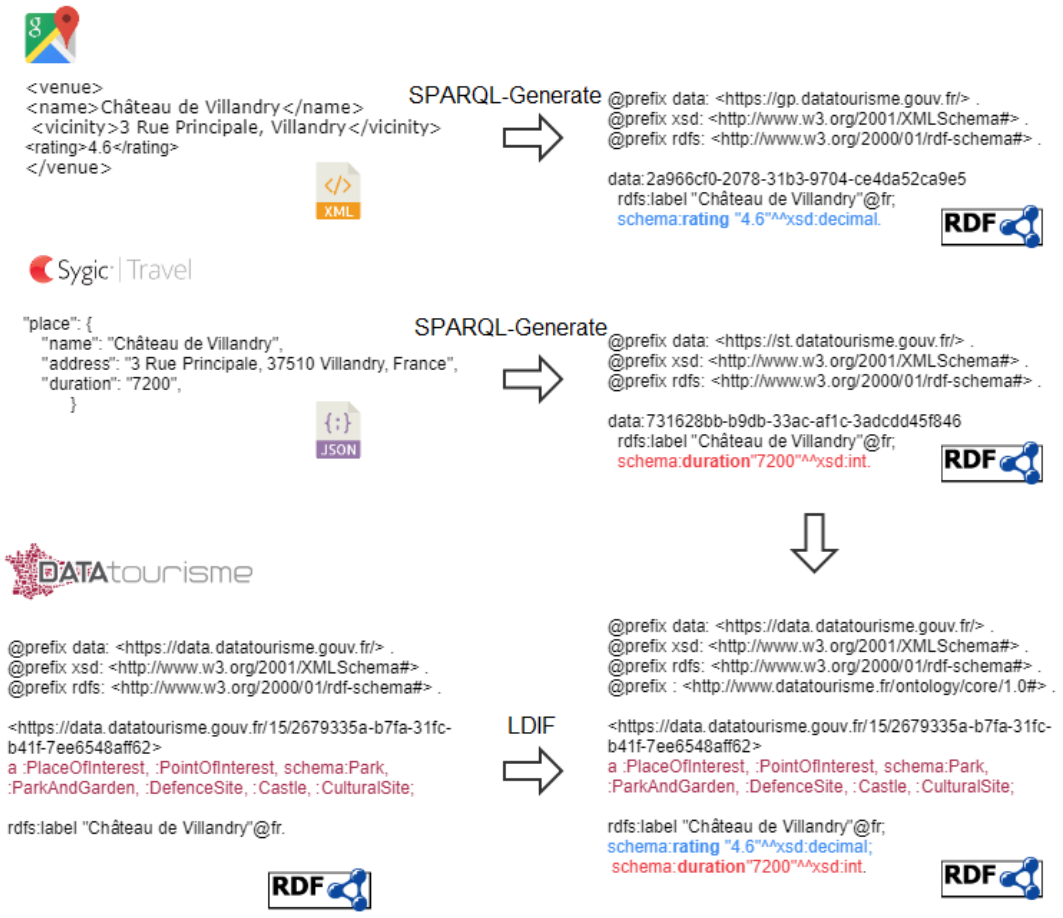


FIGURE 5.10 – Exemple de transformation des données provenant des APIs pour l'intégration au dataset DATAtourisme

représentation claire sur la base des scores attribués. Sieve propose des fonctions de base pour l'évaluation de la qualité et des fonctions de fusion, ainsi qu'une interface pour ajouter des fonctions spécifiques à l'aide d'un fichier de configuration. Les métriques à appliquer incluent les fonctions d'agrégation de type moyenne, somme, maximum, minimum, seuil, etc que nous avons utilisées pour filtrer les données.

5.2.1.3 Enrichissement par le LOD

Afin de compléter l'ensemble des données issues de DATAtourisme, nous continuons le processus d'enrichissement par la connexion de l'ensemble des donnés sémantiques au cloud des données liées. Ainsi, ce processus permet dans un premier temps de renseigner au maximum les valeurs manquantes qui n'ont pas été complétées après le processus d'enrichissement par les APIs Web. Ceci passe par la formulation de requêtes SPARQL via

le point d'accès DBpedia afin de rechercher des ressources LOD. L'enrichissement se base sur deux modules : le module *Resource Finder* et le module *Resource Linker* comme illustré dans la Figure 5.11. Le module *Resource Finder* est responsable de la recherche de la ressource DBpedia en utilisant l'API de DBpedia Spotlight. DBpedia Spotlight est un système d'annotation automatique des documents texte, conçu par [Mendes *et al.*, 2011], permettant l'extraction des entités ou des ressources DBpedia à partir d'une description textuelle. Ainsi, le module *Resource Finder* récupère en entrée la valeur de la propriété *rdfs :label* pour rechercher l'identifiant de la ressource DBpedia équivalente. Ensuite, le module *Resource Linker* assure le lien entre les deux ressources (la ressource de DBpedia et celle de DATAtourisme) permettant ainsi d'avoir un graphe de ressources plus complet et plus riche.

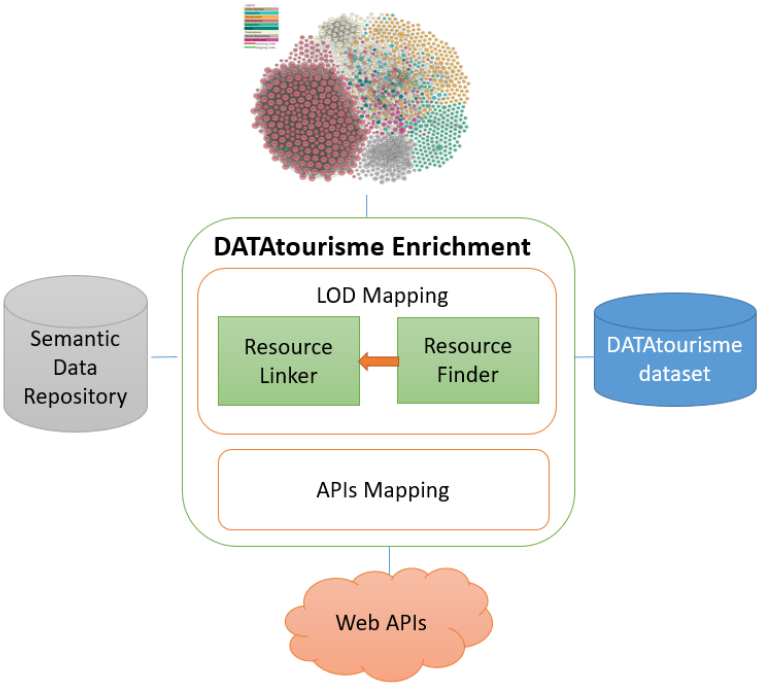
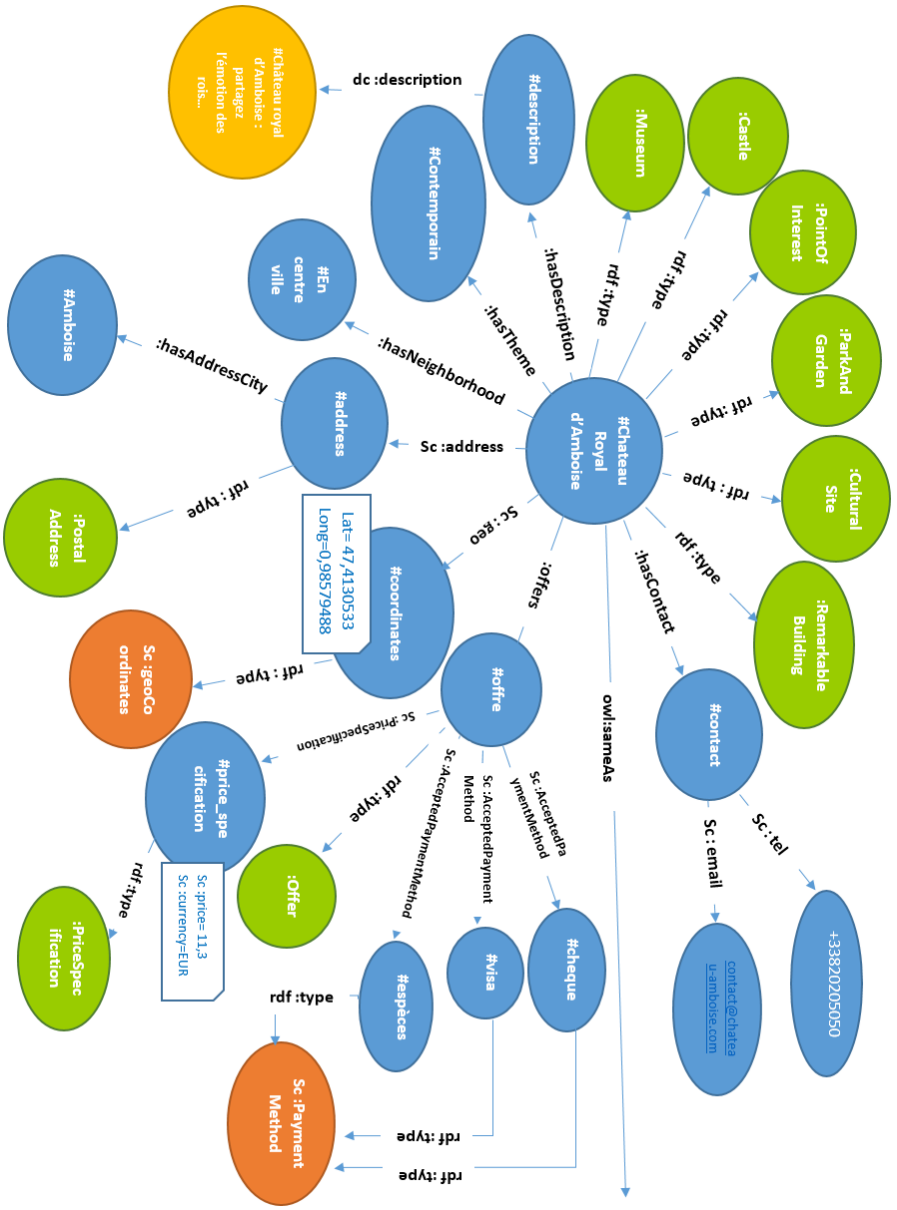


FIGURE 5.11 – Enrichissement de DATAtourisme via LOD

La Figure 5.12 représente le graphe résultant du processus d'enrichissement décrivant la ressource *Château Royal d'Amboise*.

5.2.2 Reconfiguration de Mashup

La phase de reconfiguration de mashup offre à l'utilisateur la possibilité d'adapter son mashup comme il le préfère. Le processus de reconfiguration se base sur le mashup déjà configuré et la base de données sémantiques fournie par la phase d'enrichissement. La Fi-


#Château royal d'Amboise : partagez l'émotion des rois...
dc :description
#description
:Museum
:Castle
:PointOf Interest
:ParkAnd Garden
:Cultural Site
:Remarkable Building
#Contemporain
#En centre ville
#Amboise
:hasTheme
:hasDescription
rdf :type
rdf :type
rdf :type
rdf :type
rdf :type
rdf :type
:hasNeighborhood
:hasAddressCity
#address
Sc :address
#Chateau Royal d'Amboise
:Postal Address
rdf : type
Lat= 47,4130533
Long=0,98579488
Sc : geo
#coordinates
:offers
:hasContact
owl:sameAs
Sc:geoCo ordinates
rdf : type
#offre
#contact
Sc :priceSpecification
#price_spe cification
rdf :type
Sc :AcceptedPaymentMethod
Sc :AcceptedPaymentMethod
Sc :AcceptedPaymentMethod
Sc : email
Sc : tel
:Offer
:PriceSpec ification
rdf :type
Sc :price= 11,3
Sc :currency=EUR
#espèces
#visa
#cheque
contact@chateau-amboise.com
+33820205050
rdf :type
Sc :Payment Method
rdf :type
rdf :type

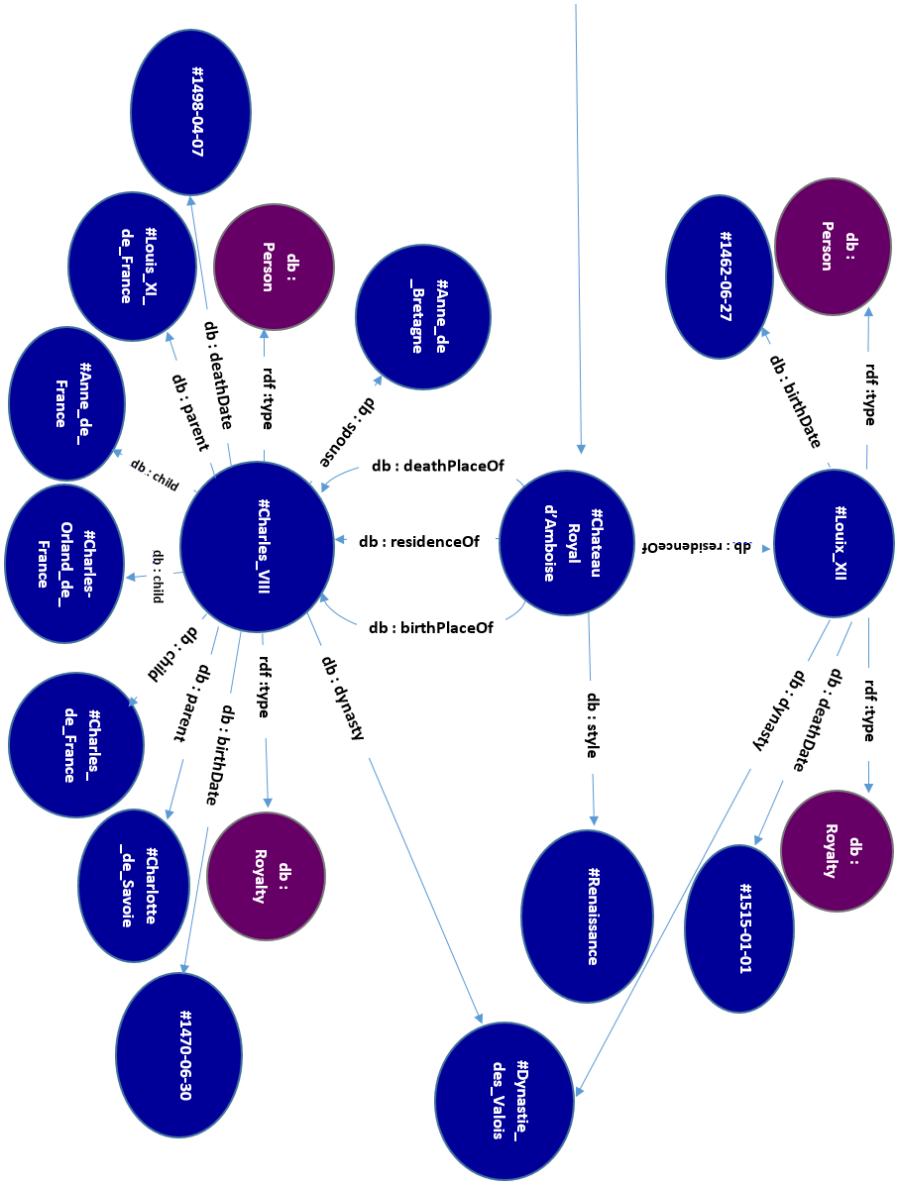


FIGURE 5.12 – Graphe de données de la ressource après enrichissement

gure 5.13 décrit ce processus qui se compose de deux étapes. La première étape est la sémantisation du mashup, c'est-à-dire la transformation des données brutes formant le mashup en données sémantiques. Ceci permet, en effet, de travailler avec des données homogènes afin de profiter de l'interconnectivé des données et d'appliquer, par conséquent, une mesure de similarité pour la génération d'alternatives. Grâce à un ensemble d'opérations de reconfiguration, l'utilisateur peut personnaliser son mashup en se guidant de ces alternatives calculées durant la seconde étape de reconfiguration.

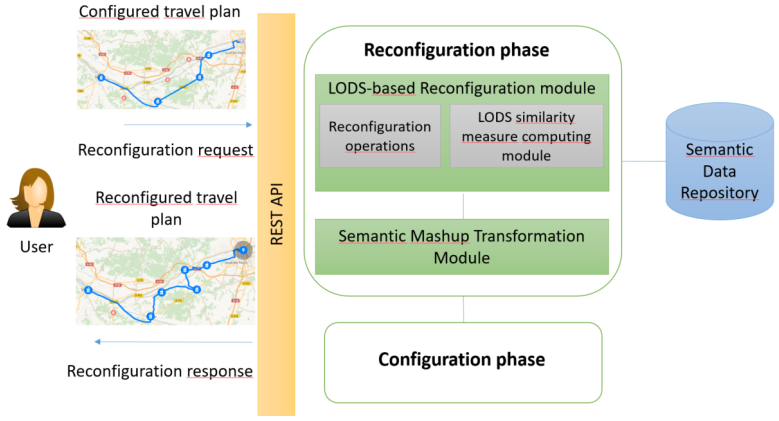


FIGURE 5.13 – Processus de reconfiguration de mashup

5.2.2.1 Sémantisation de mashup et Mesure de similarité LODS

A partir du mashup configuré, nous procédons à la sémantisation de ses composants. Cette étape consiste à rechercher les ressources correspondantes aux composants de données du mashup dans la base sémantique. Il s'agit d'interroger l'ensemble des données enrichies via des requêtes SPARQL. Cette recherche applique un filtre sur la propriété *dc : identifier*, issue de l'ontologie DublinCore, de la ressource où nous avons remarqué une correspondance avec la propriété *identifiant* des composants du mashup. Ainsi, la transformation devient relativement simple grâce à cette observation. La Figure 5.14 représente un exemple de sémantisation de mashup.

```
<TravelMashup name="LoirePlanVisit" >
<steps>
    <step star="2018-06-17T10:00" end="2018-06-17T11:30">
        <poi id="PCU41AASOR100135">
                <Name>Château Royal de Blois</Name>
                <Rating>0.035915155</Rating>
                <Position>
                    <Latitude>47.58</Latitude>
                    <Longitude>1.33</Longitude>
                    <Adresse> Place du Château,41000,Blois</Adresse>
                </Position>
                <Contact>
                    <Telephone>02 54 90 33 33</Telephone>
                    <Mail>contact@chateaudeblois.fr</Mail>
                    <Web>http://www.chateaudeblois.fr</Web>
                </Contact>
                <Description>Résidence favorite des Rois de France à la Renaissance.
                <Types>
                    <Type>CulturalSite</Type>
                    <Type>Museum</Type>
                    <Type>DefenceSite</Type>
                    <Type>Castle</Type>
                    <Type>LocalBusiness</Type>
                <Horaires>
                <Periode debut="2018-01-01T00:00" fin="2018-12-31T23:59:59">
                    <Jour>
                        <Nom>Lundi</Nom>
                        <PlageHoraire debut="12:00" fin="14:00"/>
                        <PlageHoraire debut="19:30" fin ="21:30"/>
                        </Jour>
                    .....
```

```
{
  "@context": {
    "@vocab": "https://www.datatourisme.gouv.fr/ontology/core#", "schema": "http://schema.org/", "bd
    "data": "https://data.datatourisme.gouv.fr/",  "owl": "http://www.w3.org/2002/07/owl#", "xsd": "
    "fn": "http://www.w3.org/2005/xpath-functions#",  "skos": "http://www.w3.org/2004/02/skos/core#"
    "align": "https://www.datatourisme.gouv.fr/ontology/alignment#",  "dct": "http://purl.org/dc/ter
    "kb": "https://www.datatourisme.gouv.fr/resource/core#",  "foaf": "http://xmlns.com/foaf/0.1/",
    "olo": "http://purl.org/ontology/olo/core#", "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#
    "hint": "http://www.bigdata.com/queryHints#",  "ebucore": "http://www.ebu.ch/metadata/ontologies
    "sesame": "http://www.openrdf.org/schema/sesame#",    "dc": "http://purl.org/dc/elements/1.1/"
  },
  "@graph": [{
      "@id": "https://data.datatourisme.gouv.fr/15/9a322fdf-bac2-3547-ad01-3a8b9cd10ff7",
      "dc:date": [{
          "@value": "2010-01-01",
          "@type": "xsd:date"
        },{
          "@value": "2019-01-02",
          "@type": "xsd:date"
        }],
      "dc:identifier": "PCU41AASOR100135",
      "schema:offers": {
        "@id": "data:0e466943-6abd-3454-acae-0ddc67c1c719",
        "schema:acceptedPaymentMethod": [{"@id":"kb:BlueCard"},{"@id":"kb:Cash"},{"@id":"kb:Check"},
        "schema:priceSpecification": [{
            "@id": "data:0807b4be-c598-3e84-99f6-1ee5617ca0c0",
            "schema:minPrice": {
              "@value": "6.5",
              "@type": "xsd:decimal"
            },
            "schema:price": {
```

FIGURE 5.14 – Sémantisation du mashup

Cette transformation est utilisée par le module de reconfiguration qui se base sur la mesure de similarité LODS (Linked Open Data Based Similarity Measure). La mesure de similarité LODS, proposée par [Cheniki *et al.*, 2016], évalue le degré d'appariement entre les paires de termes représentées par des ressources de données ouvertes liées. Elle exploite la structure taxonomique des concepts ontologiques et des schémas de classification en plus

des propriétés sémantiques qui décrivent sémantiquement les ressources LOD.

Il s'agit, en effet, d'une composition de trois sous-mesures : *SimP* exploite la structure taxonomique des concepts ontologiques utilisés pour instancier les ressources, *SimC* utilise les schémas de classification pour catégoriser les ressources et *SimI* exploite les propriétés des ressources liées.

SimP se focalise sur l'espace de propriétés caractérisant deux ressources a et b qui sont instanciées avec des concepts issus des ontologies partagées augmentées $o_i \in O_{a,b}$ et classées en catégories à partir de schémas de classification partagés augmentés $C_{a,b}$. Elle se définit comme suit :

$$SimP_{\forall P_i \in P_{a,b}}(a,b) = \frac{\sum_{P_i \in P_{a,b}} SimP_{P_i}(a,b)}{|P_{a,b}|}$$

Quant à la similarité de classification *SimC*, elle se définit comme suit :

$$SimC_{\forall t_i \in C_{a,b}}(a,b) = \frac{\sum_{t_i \in C_{a,b}} SimC_{t_i}(a,b)}{|C_{a,b}|}$$

La similarité d'instanciation *SimI* de deux ressources décrites avec les concepts d'une ontologie $o_i \in O_{a,b}$ est calculée sur la base des cardinalités des caractéristiques taxonomiques différentielles et communes des ressources comparées. Elle se définit comme suit :

$$SimI_{\forall O_i \in O_{a,b}}(a,b) = \frac{\sum_{O_i \in O_{a,b}} SimI_{O_i}(a,b)}{|O_{a,b}|}$$

Le score de similarité final est obtenu en calculant la moyenne des trois sous-mesures :

$$LODS(a,b) = AVG(SimI_{\forall O_i \in O_{a,b}}(a,b), SimC(a,b)_{\forall t_i \in C_{a,b}}, SimP_{\forall P_i \in P}(a,b))$$

Avec :
— a et b deux ressources
— $C_{a,b}$ représente l'ensemble des concepts décrivant les ressources a et b
— $P_{a,b}$ désigne l'ensemble des propriétés caractérisants les ressources a et b.

La raison derrière notre choix de cette mesure est l'adaptabilité où nous pouvons paramétrer le niveau de granularité pour le calcul de l'appariemment entre ressources. Ainsi, nous fournissons à l'utilisateur le moyen de définir la notion de similarité à travers une mesure configurable permettant d'opérer sur un espace de recherche spécifique pour le

calcul.

5.2.2.2 Module de Reconfiguration

Sur la base de l'ensemble de données riche, le module de reconfiguration construit un graphe de connaissances spécifique au domaine qui interconnecte des types d'attraction à granularité fine exprimant des relations sémantiques.

Définition 14. (Graphe de connaissances) : un graphe de connaissances (Knowledge Graph KG) est un graphe orienté reliant des entités, E en utilisant la relation qui existe entre ces entités, R. Les nœuds sont les entités et les arêtes sont les relations. Ceux-ci sont communément appelées triplets. Pour chaque triplet, l'arête et les 2 nœuds qu'elle relie peuvent être exprimés comme :

$$(h, r, t) \mid h,\ t \in E,\ r \in R$$

Ici, h et t sont respectivement les entités head et tail et r est la relation qui les relie. Dans notre graphe de connaissances, l'ensemble d'entités composé de POI et l'ensemble de relations se compose de toutes les relations qui existent entre les entités qui seront extraites de l'ensemble de données enrichi.

Il est bien connu que les entités sont cruciales et jouent un rôle important dans le graphe. Dans notre travail, il y a de nombreux candidats d'entités. Cependant, nous identifions les entités clés à savoir *l'entité POI*, *l'entité Place*, *l'entité Person* et *l'entité Period* dans un graphe orienté domaine.

Définition 15. (entité POI) : un POI impliqué dans le mashup de voyage est une entité POI. Il est extrait du concept POI et possède certains attributs tels que le nom, la description, le prix, etc.

Définition 16. (Entité Place) : une entité Place est extraite du concept de lieu décrivant un emplacement de lieu et caractérisée avec certains attributs comme la latitude, la longitude, la ville, etc.

Définition 17. (Entité Person) : une Entité Person est extraite du concept Personne décrivant une personne célèbre comme un roi, une Reine ou un Prince qui a occupé une place.

Définition 18. (Entité de period) : une entité Period est extraite du concept de période qui décrit une période historique et caractérisée avec certains attributs comme startDate, EndDate, etc.

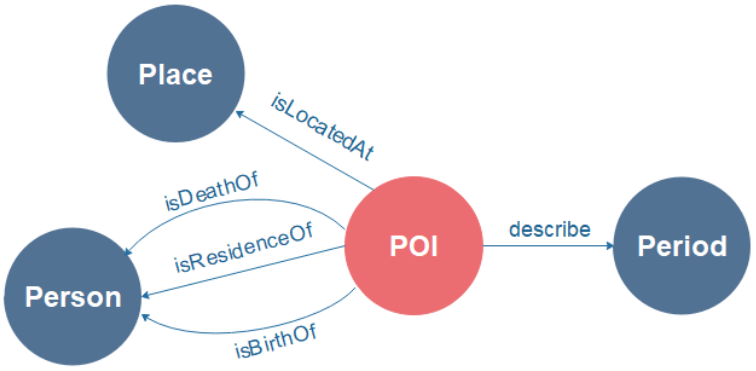


FIGURE 5.15 – Méta graphe de connaissance

Dans le graphe des connaissances, il peut exister plusieurs types de relations entre chaque paire des quatre types d'entités. La figure 5.15 décrit le schéma du graphe des connaissances sur les voyages et montre les différentes relations.

Afin de répondre aux besoins de personnalisation, nous proposons la méthode de reconfiguration qui applique la mesure de similarité LODS sur les relations logiques sous-jacentes dans le graphe de connaissances. Elle vise à calculer une liste de services alternatifs en réponse à une demande de reconfiguration. Avant de présenter cette méthode, nous définissons la demande de reconfiguration.

Définition 19. (Interface de reconfiguration) : une interface de reconfiguration (Reconfiguration Interface RI) fournit un ensemble d'opérations permettant d'adapter le mashup.

— Remove (e) : supprime l'entité e du mashup M.
— Add () : étend le mashup M avec de nouveaux éléments qui seront proposés parmi les alternatives basées sur la mesure LODS où l'utilisateur peut appliquer des filtres sur les propriétés pour contrôler les propositions.
— Replace (e) : remplace l'entité e dans le mashup M par d'autres en fonction des alternatives proposées.

Définition 20. (Demande de reconfiguration) : une demande de reconfiguration (Reconfiguration Request RR) peut être modélisée comme suit $RR = (op, RF)$ Où :

— $op\ in RI$ fait référence à l'opération de reconfiguration demandée.
— RF désigne les fonctionnalités de reconfiguration exprimées en $RF = (FN, FV)$ qui représente un ensemble de noms de fonctionnalités de reconfiguration FN et un ensemble de valeurs de fonctionnalités de reconfiguration FV. Chaque caractéristique de reconfiguration $f = (n, v) \in RF$ est définie comme un couple où $n \in FN$ décrit une

caractéristique spécifique concernant une relation r et v la valeur correspondante. Ainsi, $FN \subseteq R$.

Algorithm 5 Reconfiguration

Data: k, KG, RR, M

Result: un ensemble de k alternatives de services

2 re $\leftarrow$ *find e in KG / * chercher l'entite a reconfigurer dans le graphe * /*

3 services = $\emptyset$

4 **for** f $\in RR.FN$ **do**

5 **for** e $\in KG.E$ **do**

6 **if** e $\notin M.S$ **do**

7 LODS (re, e) $\leftarrow$ *calculer la LODS similarite LODS de re et e*

8 S = $\{s_1, s_2, ...s_n\}$ */ * ensemble de services bass sur la relation f.n * /*

9 **for** $s_i \in S$ **do**

10 services.add (s_i)

11 service _list $\leftarrow$ *get top k alternatives in services*

12 **Return** service _list

L'algorithme de reconfiguration Algorithme 5 prend en entrée la taille des alternatives K, le graphe de connaissances KG, la demande de reconfiguration RR et le mashup M pour générer les k alternatives de services. Premièrement, l'algorithme localise l'entité reconfigurée dans le graphe. Ensuite, pour chaque caractéristique de reconfiguration, l'algorithme calcule la similarité LODS entre l'entité reconfigurée et les entités dans le graphe non présentes dans le mashup. Si la valeur du score LODS dépasse le seuil, le service est ajouté dans l'ensemble S qui représente une similitude des services en termes de relation f.n. Puis (ligne 9 et ligne 10), le vecteur S est ajouté au vecteur de service et un tri sur l'ensemble de services est appliqué lors du calcul des k meilleurs services alternatifs.

Ainsi, nous complétons la liste des caractéristiques de la configuration présentée dans les chapitres précédents par une nouvelle propriété qui est la *complémentarité*. De ce fait, une configuration est dite *complémentaire* si les éléments de la configuration $CS.I$ ($CS.I \subseteq KG.E$) sont connectés moyennent une relation r.

Cette complémentarité permet d'offrir un mashup sur mesure où les ressources sont composées dans une logique sémantique construisant par exemple des visites thématiques qui remontent l'histoire. En l'occurrence, un plan de visite traçant la vie d'un roi peut être généré en exlpoitant les relations *isResidenceOf*, *isBirthOf* et *isDeathOf*.

Conclusion

Dans ce chapitre, nous avons présenté le modèle de reconfiguration qui permet aux utilisateurs de personnaliser leurs mashups. Ce modèle s'appuie sur la technique de reconfiguration couplée avec l'exploitation des données liées afin de fournir un système de personnalisation élastique qui répond de manière flexible aux besoins situalitionnels des utilisateurs. La reconfiguration se base sur la navigation dans un graphe de ressources afin de générer les alternatives utilisées pour l'ajustement du mashup.

Après avoir présenté notre approche de configuration de mashup spécifique au domaine avec ses trois modèles(fonctionnel, contextuel et de personnalisation), nous décrivons son implémentation et son évaluation dans le prochain chapitre.

Chapitre 6

CART : Outil de Mashup Personnalisé

Introduction

Dans ce chapitre, nous présentons comment mettre en application l'approche de configuration de mashup décrite dans les chapitres précédents. Le modèle de mashup spécifique au domaine fournit une base consolidée pour le développement d'outil de mashup pour notre domaine de référence. CART (a Configured mAshup Recommender application for personalized Trip planning) est un outil de mashup adapté au domaine du tourisme, permettant d'agréger les données touristiques afin de suggérer les plans de visite satisfaisant les préférences et les contraintes de l'utilisateur. Il fournit une visualisation interactive et des fonctions de personnalisation pour guider l'utilisateur à affiner son propre mashup.

Dans ce qui suit, nous introduisons d'abord l'architecture générale de notre outil qui se base sur le modèle de domaine. Ensuite, nous détaillons chaque module dans l'architecture et nous décrivons à travers des scénarios le fonctionnement de CART. Enfin, nous présentons l'évaluation via une étude utilisateur.

6.1 Architecture générale de CART

L'architecture de CART repose sur les principes que nous avons présentés dans la partie 1 de ce document qui nous semblent essentiels pour le développement d'une telle plateforme de mashup. Il s'agit principalement de la programmation visuelle des mashup en s'appuyant sur des interfaces utilisateur intuitives qui fournissent une logique simple de mapping.

L'étude de la métaphore de développement d'interface graphique appropriée vise à pro-

poser aux utilisateurs des constructions intelligibles. Il est crucial d'abstraire les différents types d'actions et d'interactions que l'utilisateur peut avoir avec un environnement de développement (par exemple, choisir un composant, connecter deux composants, ...), afin d'identifier la meilleure combinaison d'interactions. A cette fin, nous avons construit une interface simple et pratique de l'outil, qui respecte le modèle de syntaxe de domaine, présenté dans le chapitre 3, en vue qu'elle soit facilement compréhensible par les utilisateurs du domaine.

En ce qui concerne la tâche du mapping, nous proposons que les utilisateurs ne définissent pas ce type de traitement. Les composants de mashup sont capables de "comprendre" et de manipuler les données correctement puisqu'ils découlent du modèle des concepts introduit dans le chapitre 3, qui définit les entités du domaine et leurs relations. Comme tous les composants parlent le même langage, la composition peut se passer du mapping explicite de données. Il suffit en d'autres termes d'indiquer quels composants participent au mashup.

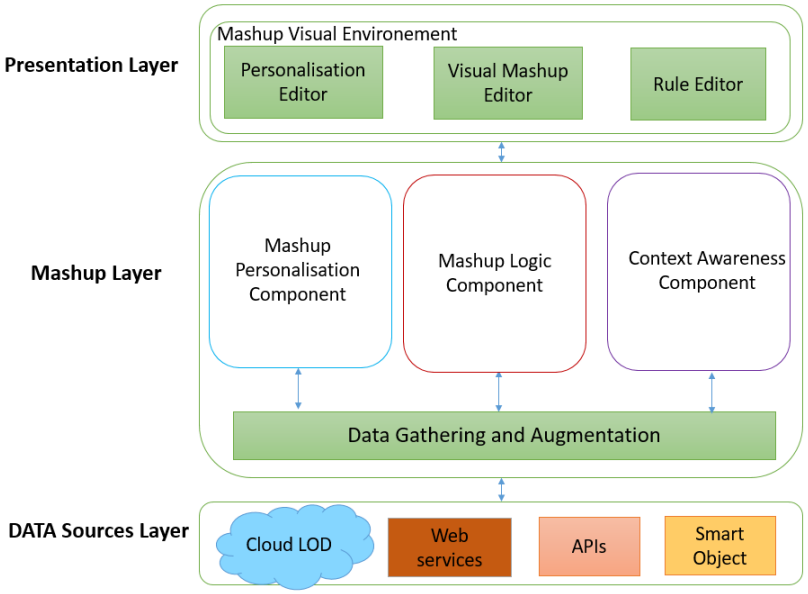


FIGURE 6.1 – Architecture générale de CART

La Figure 6.1 illustre l'architecture globale de CART qui tient compte des principes mentionnés afin de faciliter le développement de mashup personnalisable par des utilisateurs du domaine (touristes et professionnels du tourisme). L'architecture est organisée en trois couches principales, chacune gérant un aspect distinct.

La couche de présentation (Presentation Layer) est composée de trois éditeurs : l'éditeur de mashup visuel met l'accent sur la conception de mashup via un système interactif, l'éditeur de règle permet la contextualisation du mashup en définissant des règles ECA dont

les événements et les actions sont décrits en terme du modèle 5W-1H et l'éditeur de personnalisation fournit une boite à outils interactive pour l'ajustement et la personnalisation du mashup.

La couche de mashup (Mashup Layer) implique la génération et l'adaptation du mashup de services pour la planification touristique avec la gestion de la sensibilité au contexte. Cette couche implémente les trois modèles présentés dans les chapitres précédents (le chapitre 3, le chapitre 4 et le chapitre 5). Ainsi, elle regroupe le module de logique de mashup responsable de la génération du mashup et la composition de service, le module de sensibilité au contexte permettant la gestion des règles contextuelles et enfin le module de personnalisation implémentant le modèle de reconfiguration. Tous ces modules s'appuient sur une brique de base de collecte de données depuis les APIs et l'augmentation du modèle ontologique.

La couche inférieure de sources de données (Data sources Layer) représente les ressources et les services Web,interrogées à la volée, qui sont à la base d'un mashup. Ces services Web sont généralement des APIs REST. Cette couche regroupe aussi les données liées issues du Cloud. Dans ce qui suit nous détaillerons chacune des couches.

6.1.1 Couche de présentation

La couche de présentation est considérée comme le point d'entrée à la plateforme CART avec laquelle les utilisateurs du domaine, qui sont principalement des touristes et des professionnels du tourisme, interagissent. Elle joue le rôle de l'environnement de mashup à travers lequel l'utilisateur construit sa requête, la contextualise et la personnalise également de manière interactive. Par ailleurs cette couche permet de visualiser le résultat du mashup qui est, pour notre domaine, un itinéraire touristique. Il s'agit d'une couche Web-Mobile se basant sur les technologies du Web, à savoir les langages HTML, CSS et JavaScript. L'interaction est étroitement liée, dans la phase de visualisation, à l'utilisation des données stockées côté client. Dans ce contexte, nous utilisons AJAX, qui permet de réduire le temps de réponse, afin d'améliorer l'interactivité. L'interaction tactile et gestuelle est également utilisée. Ainsi, l'environnement de mashup est structuré en trois interfaces : l'éditeur de mashup visuel, l'éditeur de règles et l'éditeur de personnalisation. Le découplage de la couche de présentation favorise le développement léger d'interfaces utilisateur mettant en œuvre des métaphores visuelles plus adaptées à des domaines spécifiques

6.1.1.1 Éditeur de mashup

L'éditeur de mashup fournit aux utilisateurs un canvas pour construire le mashup. Cet éditeur de composition implémente le méta-modèle de mashup spécifique au domaine pré-

senté dans le chapitre 3 et l'expose via la syntaxe du domaine. Il affiche, en effet, une liste de composants à partir de laquelle les utilisateurs constituent la requête de mashup en connectant les composants. La Figure 6.2 illustre les différentes interfaces formant l'éditeur. L'interface de Définition de Mashup est l'interface principale qui inclut une liste de composants où à la sélection d'un composant une interface permettant à l'utilisateur de configurer ce dernier est proposée.

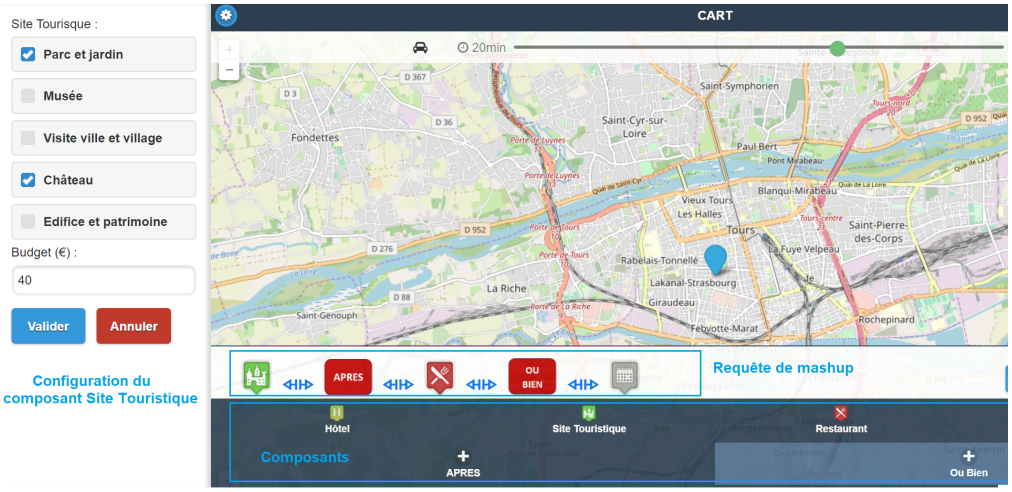


FIGURE 6.2 – Editeur de mashup

6.1.1.2 Éditeur de règles contextuelles

L'objectif principal de l'éditeur de règles est de permettre aux utilisateur de contextualiser leur mashup via la définition d'un ensemble de règles dans une métaphore d'interaction adaptée. En effet, au moyen de conditions, d'actions et d'opérateurs, les utilisateurs puissent construire les règles. En lien avec l'innovation guidée par l'utilisateur, nous avons adopté un paradigme visuel, s'adaptant au modèle mental de ce dernier, qui se base sur des termes symboliques associées à des icônes. Les actions visuelles de l'utilisateur pour la création de règles sont traduites en une spécification XML via le langage RuleML pour qu'elles puissent être évaluées côté serveur par le module gestion du contexte de la couche de mashup. La Figure 6.3 décrit l'éditeur de règles avec un exemple de règle exprimant *"Si le touriste est en famille et les conditions météorologiques prévoient des précipitations alors le plan est réduite à des POIs de la catégorie culture qui proposent une visite à l'intérieur"*.

6.1.1.3 Éditeur de personnalisation

L'éditeur de personnalisation permet aux touristes de raffiner de manière interactive le plan de visite généré par la couche de mashup. A partir d'un itinéraire affiché sur une carte,

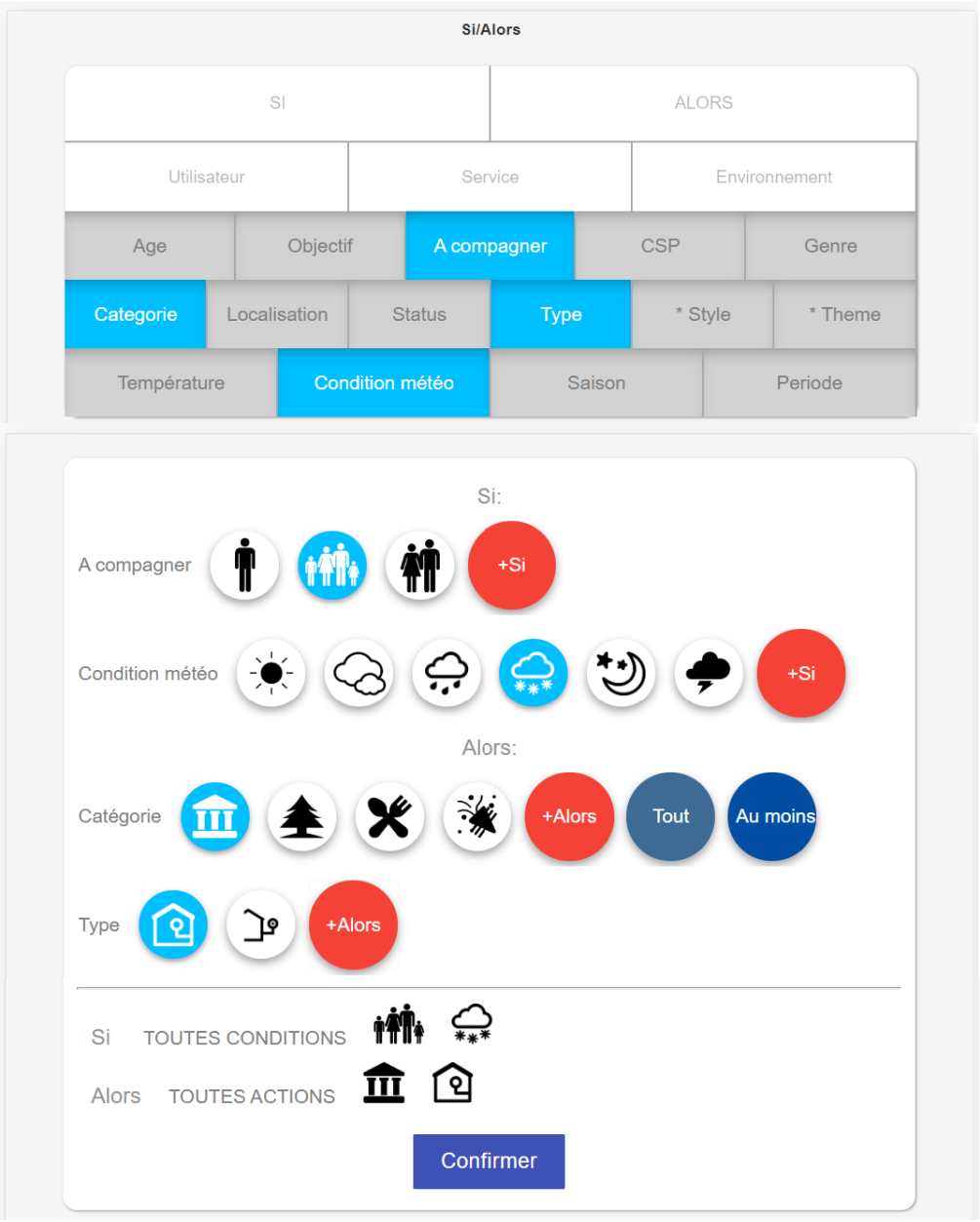


FIGURE 6.3 – Editeur de règles

issue de l'API Google Map, ou visualisé avec un objet Web graphique de type « TimeLine », le touriste peut reconfigurer les POIs constituant le plan (le mashup) via des actions visuelles. Il s'agit de remplacer, d'ajouter ou de supprimer un POI. Ces actions visuelles de reconfiguration sont envoyées à la couche de mashup et plus spécifiquement au module de reconfiguration qui analysera ce traitement d'ajustement.

6.1.2 Couche de mashup

C'est la couche principale de l'architecture qui implémente la logique de mashup de notre approche de configuration dans ses trois modèles. Ces derniers reposent sur le module de collecte de données depuis les sources d'information et la sémantisation des données assurant la transformation des données brutes en données sémantiques tel que nous avons présenté dans le chapitre précédent. Il s'agit d'une application Web coté serveur développée en utilisant le framework Spring Boot qui est une nouvelle façon de créer facilement des applications Web se basant sur le langage Java. Spring Boot offre une auto-configuration de l'application ce qui a permis d'atténuer la complexité de cette tache marquante des applications Spring. [Gutierrez, 2019] présente un aperçu des caractéristiques de Spring Boot. La figure 6.4 décrit les composants de cette couche.

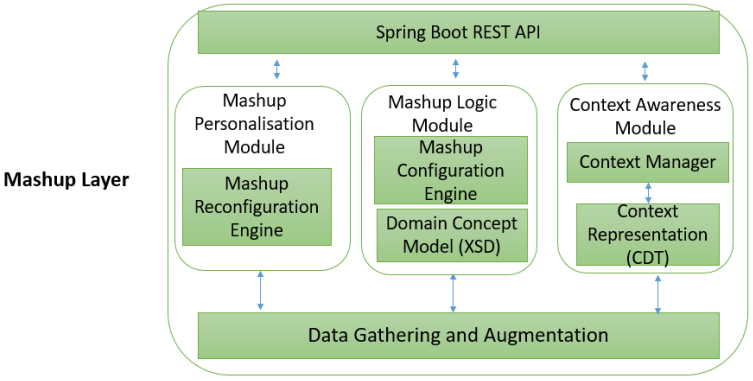


FIGURE 6.4 – Composition de la couche mashup

6.1.2.1 Module de Configuration de Mashup

La requête de mashup issue de l'éditeur de mashup sera analysée par le moteur de configuration de mashup. Il s'agit d'un sous module de l'application Spring Boot qui se charge de la génération de l'ensemble de solutions de configuration qui sont les plans de visite satisfaisant les contraintes de l'utilisateur. Ce résultat est encapsulé dans un objet JSON pour être envoyé à la couche de présentation pour la visualisation. Avant de générer les solutions de configuration, le moteur de mashup doit résoudre le mapping visuel afin de retrouver comment invoquer les ressources. C'est-à-dire établir le schéma de mashup qui indique pour chaque composant de la requête de configuration le service réel correspondant. Dans cette phase, certains paramètres de configuration peuvent être liés à des wrappers qui effectuent des transformations à partir des termes symboliques et leurs associent une valeur spécifique respectant le contrat du service.

6.1.2.2 Moteur de règles contextuelles

Côté serveur, le moteur de règles interprète le fichier RuleML décrivant les règles définies par l'utilisateur au moyen de l'éditeur de règles. Il s'agit d'un module Spring qui instantie l'objet règle en respectant la classification de la représentation du contexte (CDT). Pour l'évaluation des règles, nous utilisons Jess, le moteur d'inférence, compatible avec le langage Java. Jess est un moteur de règles léger, basé sur une version améliorée de l'algorithme Rete, qui se connecte bien avec les technologies Web. La transformation de la représentation RuleML en représentation Jess est réalisée par le langage XSLT. Les modifications sur le mashup sont apportées et le résultat est rendu à l'utilisateur.

6.1.2.3 Module de personnalisation de mashup

Le module de personnalisation implémente le modèle de reconfiguration offrant à l'utilisateur la possibilité d'adapter le mashup construit. Il s'agit d'un module Spring qui implémente la mesure de similarité LODS permettant de générer l'ensemble d'alternatives de reconfiguration suite à la requête de l'utilisateur. Cet ensemble est retourné, en réponse à la requête, à la couche de présentation pour actualiser l'interface de reconfiguration. Suite à la sélection d'une alternative, le module invoque l'opération CRUD sur le mashup afin de l'adapter et renvoyer le mashup modifié à la couche de présentation pour la visualisation.

6.1.3 Couche Sources de données

La couche inférieure, Sources d'information, inclut les ressources et les services Web qui sont à la base de toute application de mashup. Un mécanisme d'exploration a été mis en place pour identifier les ressources potentiellement intéressantes aux touristes. Il s'agit de l'APIs Google Maps permettant d'afficher une carte sur laquelle les plans de visites sont tracés, l'API Directions calcule la distance entre deux POIs ainsi que le temps de déplacement en fonction d'un moyen de transport. Les données sur les POIs sont retournées par une API interne exposant les données des systèmes Tourinsoft de l'information touristique, nous avons bénéficié de ces dernières dans le cadre de la collaboration pour le projet SmartLoire. Bien qu'il existe une panoplie d'APIs exposant les données décrivant les POIs, nous avons choisi l'API Tourinsoft qui nous retourne un ensemble de données plus riche et contrôlé par les professionnels du domaine. D'autres APIs sont aussi utilisées afin de collecter des données contextuelles, par exemple, l'API OpenWeatherMap qui retourne des données sur les conditions météorologiques d'une ville donnée. La position de l'utilisateur est déterminée par le capteur GPS de son appareil mobile. Les données ouvertes issues du cloud tel que le DataSet DBpedia sont exploitées pour la personnalisation du mashup. Pour respecter les données du domaine, nous utilisons l'API JAXB (Java Architecture for XML

Binding) qui permet d'analyser le modèle de concepts de domaine décrit par le schéma XML (XSD) et de générer les classes POJO annotées du domaine.

6.2 Fonctionnement de CART

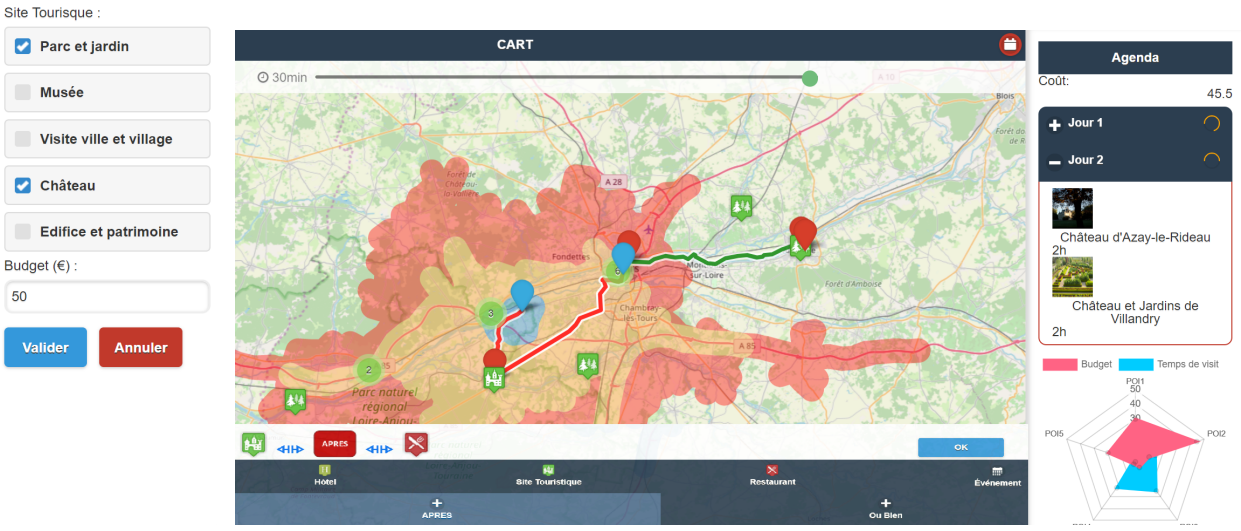
CART est un outil de mashup personnalisé permettant aux visiteurs de planifier leurs séjours touristiques en région Centre Val de Loire. En effet, pour une personne qui se retrouve dans la région Centre, riche d'un héritage culturel exceptionnel inscrit sur la liste du patrimoine mondial, il est important d'avoir la possibilité de trouver un itinéraire rejoignant les points d'intérêt offerts et construire une vision globale des attractions culturelles présentes. Doté de nombreux lieux culturels emblématiques et diversifiés avec plus de 300 monuments, plus de 30 jardins remarquables, 6 villes s'art et d'histoire qui sillonnent le Val de Loire, le visiteur a besoin de découvrir cette richesse historique. Outre le patrimoine des châteaux de la Loire, avec ses nombreuses architectures drainant les visiteurs, la région Centre-Val de Loire organise de nombreux festivals et événements. Confronter à la sélection difficile des attractions à visiter et leurs planification en itinéraires, la plateforme CART, résout ce problème en proposant des itinéraires dynamiques qui respectent les contraintes et les préférences des utilisateurs. En guise de guide moderne, CART fait participer les utilisateurs activement dans la construction de l'écosystème qui consultent des contenus décrivant le patrimoine touristique et culturel afin de créer une expérience personnalisée. CART offre également la possibilité aux utilisateurs experts (particuliers ou membres d'autorités Touristiques et/ou d'autorités Culturelles accréditées) de construire une "proposition" qui inclut une offre particulière. Afin de décrire comment CART fonctionne, nous illustrons les scénarios suivants.

6.2.1 Scénario 1 : Alice invite son amie Kate et veut préparer sa visite

Alice une étudiante en droit s'apprête à accueillir son amie anglaise Kate qui est passionnée par l'histoire française. Habitant Tours, Alice veut faire découvrir à son amie sa ville marquée par des paysages culturels uniques. Cependant, elle ne sait pas comment organiser la visite de deux jours. A cette fin, elle utilise la plateforme CART dans son mode *pas à pas* (PAP) pour construire son plan de façon incrémentale. Puisqu'elle veut commencer sa visite de chez elle, elle autorise la fonctionnalité de la geolocalisation sur la page d'accueil de CART et commence à construire son itinéraire. Elle introduit également la date de début et la date de fin du séjour puis clique sur le bouton Enregistrer.

Pour construire son plan, Alice commence par sélectionner l'activité « site touristique » et indique ses préférences dans cette catégorie comme indique la Figure 6.5a. Une fois qu'elle clique sur le bouton « valider », les POIs autour de son point de départ sont

affichés sur la carte lui permettant de choisir son premier lieu de visite. Grâce à la barre de configuration du temps de déplacement, Alice peut découvrir d'autres POIs, affichés, sur les rayons jaune et rouge, s'elle souhaite se déplacer plus loin. En choisissant le « château de Tours », son premier lieu de visite, il devient le centre d'un nouveau rayon ce qui permettra à Alice de choisir un nouveau POI. Elle veut visiter Amboise, elle ajoute le « château Royal d'Amboise » à son itinéraire et juste après elle cherche un restaurant à côté dont elle se souvient d'un bon moment avec un cadre sympathique, que son amie l'aimerait. Ayant construit progressivement son itinéraire pour le premier jour, Alice continue sa recherche pour planifier son second jour de visite où elle compter explorer le sud de la région. Le plan de visite est visualisé en entier sur la carte et sous forme de TimeLine comme l'illustre la figure 6.5.



(a) (b)

FIGURE 6.5 – Plan de visite d'Alice

6.2.2 Scénario 2 : Pierre découvre la Vallée de la Loire

Pierre est un jeune entrepreneur de Paris qui participe à un séminaire en région Centre à Tours. Il veut joindre l'utile à l'agréable pour découvrir les sites phares de la vallée de la Loire pendant son séjour de 2 jours à Tours. Afin de planifier sa visite, Pierre utilise CART l'application de mashup qui récupère et intègre l'information touristique. Ayant peu d'informations, contrairement à Alice qui est de la région, Pierre choisit le mode « bout en bout » qui lui recommande automatiquement un plan. Pour cette fin, il commence par introduire ses contraintes en termes de durée de visite et son point de départ. Ensuite, il définit sa requête de mashup en déposant le service des « Sites Touristiques », qui fournit

des données sur les monuments historiques autour d'une position donnée, puis il sélectionne l'opérateur « APRES » pour joindre un autre service à son mashup en série. La sélection du service « Restaurant » et son configuration, indiquant le budget de cette catégorie et le type de restaurant, permet de combiner les données générer par ces deux services. Une fois que Pierre clique sur le bouton « OK », son itinéraire est affiché sur la carte ainsi qu'une visualisation TimeLine est aussi disponible listant les différent POIs recommandés avec la durée de visite de chacun comme l'illustre la Figure 6.6.

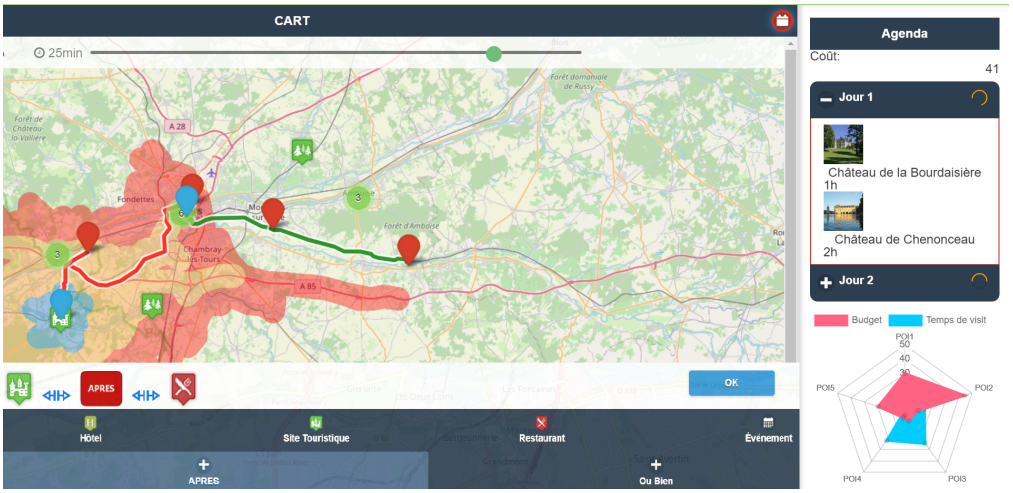
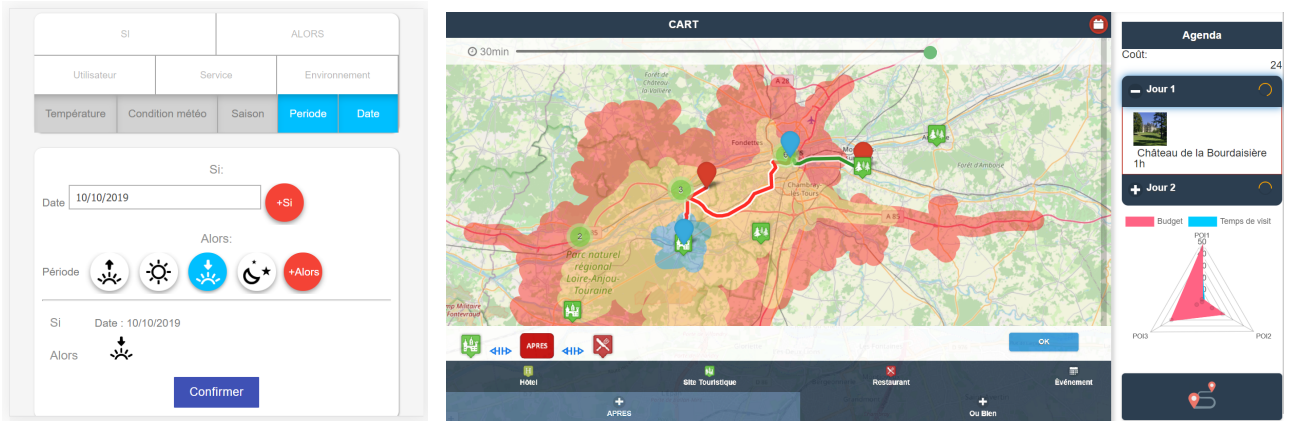


FIGURE 6.6 – Plan de visite initial de Pierre

Pierre cherche ensuite à rendre compatible le plan de visite recommandé avec le programme de son séminaire. Il utilise alors l'éditeur de règles de CART afin d'introduire ses disponibilités sous forme de règles 6.7a. Une fois sur l'interface de création de règles, l'assistant affiche les conditions à spécifier (bouton SI) puis les actions à appliquer (bouton ALORS) sur le mashup. Ainsi, il définit la règle *Si la date est 10/10/2019 qui correspond à son premier jour de visite alors il cherche à réduire sa visite pour la période de l'après-midi.* Le plan de visite adapté est illustré par la Figure 6.7b.

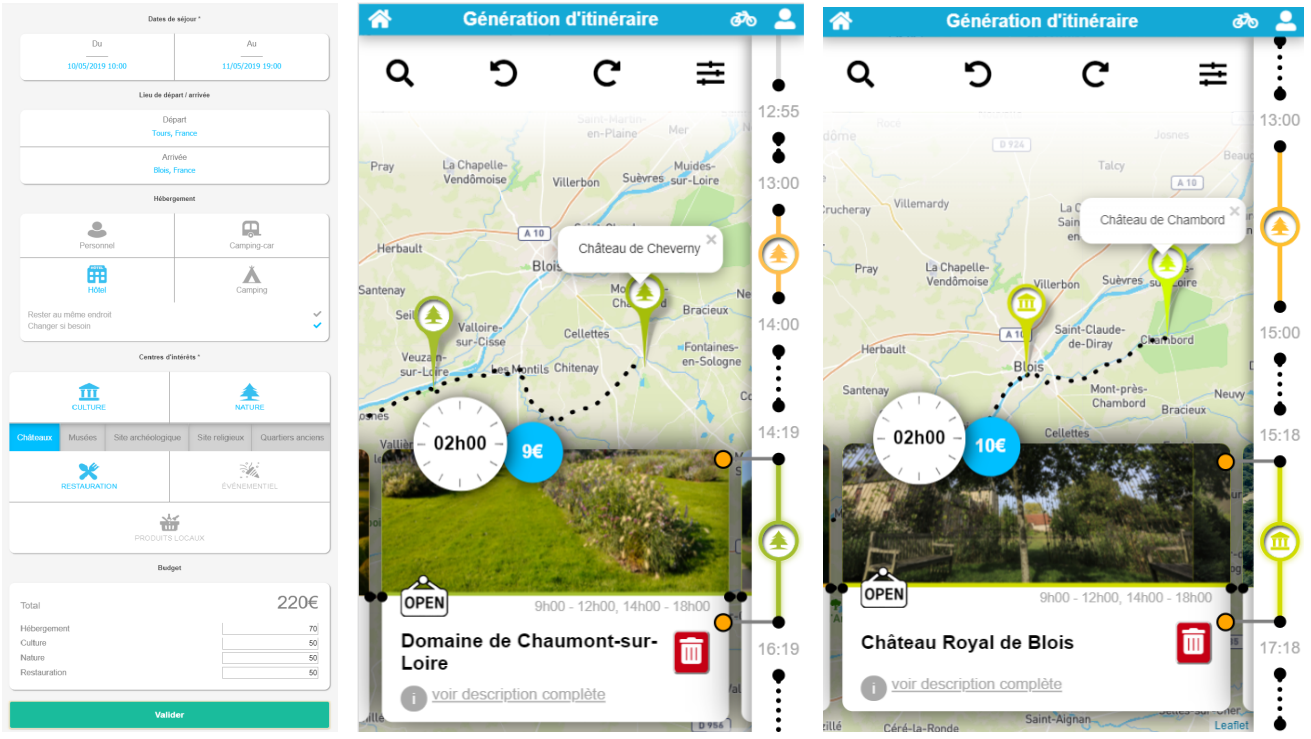
6.2.3 Scénario 3 : Jade remonte le temps en Val de Loire

Jade, une passionnée par l'art et l'histoire, veut planifier une visite en Val de Loire afin de la découvrir. Elle utilise l'outil CART lui permettant de spécifier ses préférences pour la visite. Jade indique une durée de visite de 2 jours pour explorer les châteaux de la Loire et la gastronomie locale comme décrit la Figure6.8a. Ainsi, CART calcule le plan de visite correspondant aux préférences et l'affiche sur une carte géographique illustré par la Figure 6.8b. Jade cherche à vivre une aventure lui permettant de remonter le temps jusqu'à la Renaissance durant sa visite. Elle exploite ainsi, la possibilité de reconfigurer son plan afin



(a) Spécification de règle (b) Plan de visite adapté

FIGURE 6.7 – Plan de visite adapté de Pierre



(a) Spécification des préférences (b) Plan de visite initial (c) Plan de visite reconfiguré

FIGURE 6.8 – Plan de visite de Jade

de construire *une visite au fil du temps*. Elle choisit la propriété de reconfiguration "Style architectural" avec la valeur "Renaissance" pour adapter son plan. CART reconfigure le plan au niveau du jour 2 de la visite en remplaçant *le château de chaumont sur Loire* et *le*

château de Cheverny, représentés par des marqueur vert sur la Figure 6.8b, par *le château Royal de Blois* et *le château de Chambord* qui sont représentés par la Figure 6.8c.

6.3 Étude utilisateur et Évaluation

Nous présentons dans cette section une évaluation de notre approche de configuration de mashup personnalisé afin de vérifier si les utilisateurs sont effectivement plus à l'aise avec le paradigme de mashup spécifique à un domaine et tester le bon compromis entre la flexibilité et la complexité de la contextualisation et la personnalisation du mashup. Pour ce faire, nous avons mené une étude utilisateur sur la plateforme CART où nous avons émis l'hypothèse que ce dernier est plus facile à utiliser pour la création d'itinéraires plus personnalisés. La première étude vise à comparer CART avec des systèmes de recommandation touristique sur la base d'un ensemble de critères traitant la construction interactive du mashup. Une seconde étude auprès des utilisateurs a été menée permettant de valider la performance de CART en termes de caractéristiques liées à la satisfaction des utilisateurs.

6.3.1 Étude comparative

Afin de mener une étude comparative, nous avons établit un framework conceptuel permettant de comparer les systèmes de recommandation touristique se focalisant sur l'interaction des utilisateurs avec le système. Nous définissons deux niveaux de granularité pour l'interaction : le niveau POI et le niveau plan. En prenant comme point de départ les lignes directrices de [Pu *et al.*, 2011], nous déterminons 4 critères de comparaisons.

— **C1 - Expression souple des préférences**
  Il s'agit d'éviter des schémas rigides pour la définition de la requête. Différents utilisateurs ont également différents niveaux d'expertise et d'aisance, de sorte qu'un processus progressif adapté à leurs connaissances et à leur expérience serait approprié.
  — Au niveau POI : le système devrait suggérer des POIs potentiels à inclure dans la visite.
  — Au niveau plan de visite : le système peut avoir besoin de préférences liées au plan telles que la destination, la durée, etc.
— **C2 - Préférence ++**
  Ce critère consiste à aller au-delà des préférences exprimées et propose aux utilisateurs d'éventuelles préférences supplémentaires qui permettraient d'étendre une visite. Ceci peut également concerner des suggestions qui ne sont peut être pas encore optimales, mais qui le deviendront en raffinant les préférences.
  — Au niveau POI : Afficher des POIs dans des catégories potentiellement intéressantes pour l'utilisateur.

  - Au niveau plan de visite : suggérer une séquence de POI qui représente une visite partielle, à laquelle l'utilisateur pourrait ajouter d'autres POI pour la compléter.
- **C3 - Présentation des résultats**

  Un utilisateur ne devrait pas être submergé par la quantité d'informations affichées
  - Au niveau POI : pour les affichages sur mobile, seuls quelques POI individuels peuvent être affichés.
  - Au niveau plan de visite : afficher qu'un plan à la fois pour les affichages mobiles.
- **C4 - Explications**

  Le dernier critère consiste à fournir de l'information sur la façon dont le système a calculé le résultat.
  - Au niveau POI : expliquer comment les POIs correspondent aux préférences.
  - Au niveau plan de visite : expliquer comment la visite correspond aux préférences.

Notre étude comparative comprend des systèmes proposés dans des travaux universitaires et d'autres systèmes qui correspondent à des produits commerciaux. Pour la sélection des systèmes universitaires, nous nous sommes basés sur les critères suivants : 1) l'article doit décrire un système de planification d'itinéraire ; 2) l'article fournit une description de l'interface utilisateur de ce système. Ainsi, nous avons sélectionné 9 articles. De plus, nous avons inclus un autre travail qui ne vérifie pas complètement les critères mentionnés mais traite l'interactivité à l'échelle de POI. Quant aux systèmes commerciaux, nous avons utilisé une recherche Google sur les planificateurs d'itinéraires touristiques et nous avons retenu 3 systèmes. Les résultats de notre évaluation sont présentés dans le tableau 6.1 où nous indiquons pour chaque système s'il vérifie les critères définis au niveau du POI et au niveau du plan de visite.

Nous concluons que la majorité d'entre eux suivent une stratégie de planification identique. Initialement, le visiteur indique ses préférences en terme de destination, les catégories de POIs et éventuellement la durée de la visite puis le système recommande le plan satisfaisant les contraintes. De même la majorité des systèmes propose d'apporter des modifications interactives sur le plan généré où les utilisateurs retirent un POI de l'itinéraire ou en insèrent un nouveau. Lors de l'insertion d'un nouveau POI, l'utilisateur dispose d'une grande liste de POIs potentiellement intéressants à sélectionner. Une autre alternative suivie par RoutePerfect, CT-Planner, PersTour et TourRec consiste à ajuster les degrés de préférence pour différentes catégories et le système recalcule automatiquement l'itinéraire. Bien que ces systèmes fournissent un environnement interactif, ce dernier est limité à la modification de l'itinéraire généré ce qui réduit le contrôle de l'utilisateur sur la recommandation. Avec le mode PAP pour la construction progressive, CART offre aux utilisateurs un moyen de maîtriser la recommandation.

Pour les critères qui sont à peine ou pas du tout pris en charge, à savoir C2 et C4, nous

présentons des exemples d'utilisation typiques. Le critère C2 visant à dépasser les préférences indiquées par l'utilisateur pour lui proposer des POIs potentiellement intéressants nécessite l'exploration de l'historique de l'utilisateur et l'emploi d'un modèle d'apprentissage. CART se base sur les données liées pour proposer des POIs au-delà des préférences utilisateur. Quant au critère C4, nous pensons que la justification est l'un des aspects importants de l'interaction entre les utilisateurs et le système garantissant ainsi une transparence des recommandations. Il est beaucoup plus facile pour un utilisateur de comparer deux options si elles sont expliquées en termes de préférences utilisateur. Il s'agit de souligner l'implication des POIs à travers l'exposition des règles qui ont inféré l'ensemble de POIs comme le présente SPLIS ou présenter des itinéraires alternatifs au plan de visite.

	C1		C2		C3		C4	
	POI	Plan de visite	POI	Plan de visite	POI	Plan de visite	POI	Plan de visite
Interactive Design [Rodríguez *et al.*, 2012]	-	-	-	-	-	+	-	-
CT-Planner4 [Kurata et Hara, 2013]	+	-	-	-	+	-	-	-
Trip Builder [Brilhante *et al.*, 2013]	-	+	+	-	-	+	-	-
PathRec [Chen *et al.*, 2017]	-	+	+	-	+	+	-	-
TourRec [Herzog *et al.*, 2018]	+	+	-	-	-	+	-	+
PersTour [Lim *et al.*, 2016]	+	+	-	-	+	-	-	-
Scenic Route [Gavalas *et al.*, 2017]	+	+	+	-	+	+	-	+
Aurigo [Yahi *et al.*, 2015]	+	+	-	+	+	+	-	-
3cixty [Troncy *et al.*, 2017]	-	+	+	-	+	-	-	-
SPLIS [Viktoratos *et al.*, 2014b]	-	-	+	-	-	+	+	-
Route Perfect https ://www.routeperfect.com	+	+	-	-	+	-	-	+
Triphobo https ://www.triphobo.com/trip	+	+	-	-	+	-	-	-
Sygic Travel https ://travel.sygic.com/fr	+	+	+	-	+	-	-	-
CART (Notre Solution)	+	+	+	+	-	+	+	+

TABLE 6.1 – Tableau comparatif des systèmes de recommandation touristique

6.3.2 Étude utilisateur

Cette étude vise à évaluer l'utilisabilité du paradigme de composition mis en œuvre dans CART par rapport au " modèle mental " des touristes. Comme l'objectif ici n'est pas de mesurer la performance en termes de temps d'exécution, nous n'avons pas collecté les temps et la durée des tâches individuelles mais nous nous sommes concentrés sur l'interaction avec les éléments de l'environnement. Ainsi, pour mieux comprendre l'opinion des participants, nous avons recueilli des informations sur leurs profils via deux questionnaires. Le questionnaire initial est consacré à la collecte d'informations démographiques sur les participants, tandis que le second traite les performances utilisateur. Une autre entrevue "non structurée" a été organisée avec chaque participant afin de recueillir d'autres suggestions sur la façon d'améliorer l'environnement de CART.

6.3.3 Participants

Pour cette évaluation, nous avons contacter 15 participants ayants des compétences différentes. Nous avons réparti les participants en 2 groupes : les locaux (résidents permanents de la région, familiarisés avec les principales attractions) et les non locaux (peu familiers avec la région). Étant donné que les utilisateurs cibles peuvent éventuellement être des utilisateurs non techniques, c'est-à-dire des personnes sans compétences techniques en programmation informatique, nous avons considéré nos contacts comme des utilisateurs non-techniques. La Figure 6.9 résume les caractéristiques des participants.

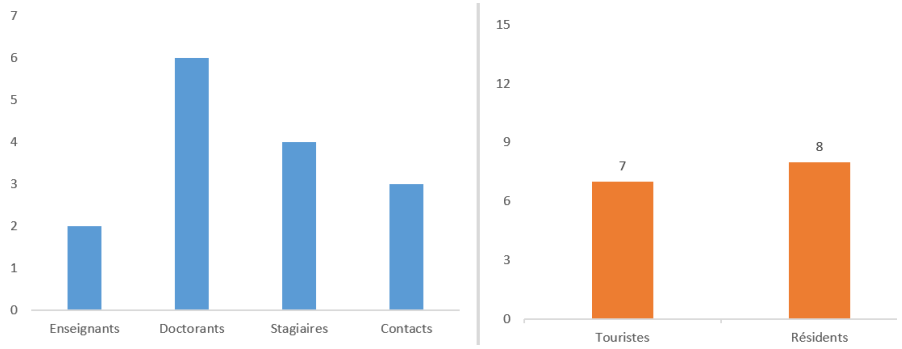


FIGURE 6.9 – caractéristiques des participants à l'évaluation

6.3.4 Procédure

Nous avons demandé aux participants de remplir le questionnaire initial pour collecter leurs données démographiques et évaluer dans quelle mesure ils étaient habitués à voyager et à construire leur itinéraire. Il s'agit de questions portant sur l'identité du participant, son

niveau d'études, son âge, son genre, le dispositif utilisé pour le test, (local ou touriste), son degré de familiarité avec les outils de recommandation et ses compétences en informatique. Lors de la session de test, nous avons demandé aux participants d'effectuer ces trois tâches : 1) Exploration de l'environnement de mashup pour construire l'itinéraire suivant l'un des deux modes proposés par CART 2) Définir des règles de contexte via l'éditeur 3) Utiliser l'éditeur de personnalisation pour ajuster le plan de visite.

A la suite des interactions avec la plateforme, chaque participant a répondu aux questionnaires mesurant la performance utilisateur. Ce dernier est structuré en trois parties :

1. SUS (System Usability Scale) [Bangor *et al.*, 2008] : une échelle de Likert en 10 questions permettant d'évaluer l'utilisabilité de CART (Figure 6.10).
2. CUSQ (Computer Usability Satisfaction Questionnaire) [Lewis, 1995] : une échelle de Likert en 12 question mesurant l'utilisabilité et la satisfaction de l'utilisateur (Figure 6.11)
3. UEQ (User Experience Questionnaire) [Laugwitz *et al.*, 2008] : une échelle de Likert en 10 points qui permet de mesurer l'expérience utilisateur (Figure 6.12).

Questions	(désaccord total) 1	2	3	4	(Accord total) 5
Je pense que le système est facile à utiliser					
Je pense que le système est complexe					
Je pense que la construction du plan est facile					
Je pense que j'ai besoin d'aide pour utiliser le système					
Je pense que l'éditeur de règle est facile à utiliser					
Je pense que j'ai besoin d'apprendre des choses avant d'utiliser le système					
Je pense que la personnalisation du plan est facile					
Je pense que le système contient des incohérences					
Je trouve que les fonctionnalités du système sont bien intégrées					
Je pense que le système est compliqué à utiliser					

FIGURE 6.10 – Questionnaire SUS

Questions	Désaccord Total 1	2	3	4	5	6	Accord Toal 7
Globalement, le système est facile à utiliser							
Le système est simple à utiliser							
Le système est confortable							
Le système est simple à prendre en main							
La spécification des règles contextuelles est facile							
Les informations concernant les POIs sont utiles							
La navigation dans le système est facile							
La représentation de l'itinéraire est simple à comprendre							
J'ai apprécié les interfaces du système							
Le système fournit les fonctionnalités attendues							
Je recommande le système							
Globalement, je suis satisfait du système							

FIGURE 6.11 – Questionnaire CUSQ

6.3.5 Résultat

L'impression générale sur CART semble être positive, d'après les réponses des participants sur les questionnaires SUS et CUSQ dans l'intervalle 1 (désaccord total) à 5 (accord

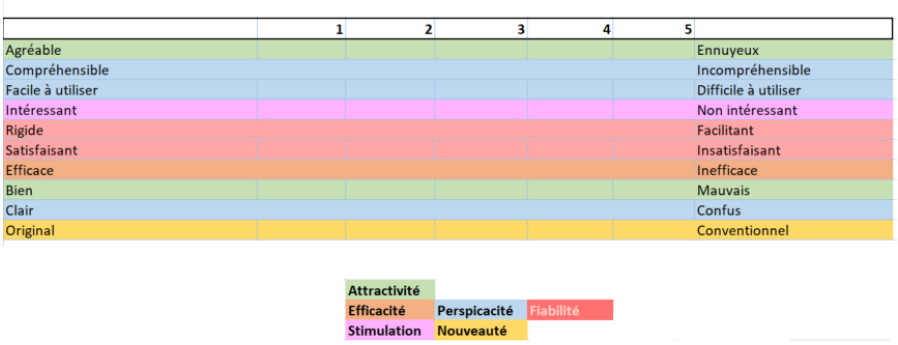


FIGURE 6.12 – Questionnaire UEQ

total) ainsi que la moyenne et l'écart type. L'analyse détaillée est fournie dans les sous-sections suivantes.

6.3.5.1 SUS

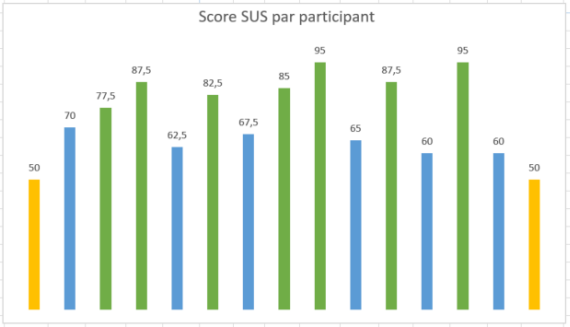
Le score global SUS est de 73 supérieur à la référence (71,4), ce qui est considéré bon selon [Bangor *et al.*, 2009]. Les résultats encourageants, illustrés par la Figure 6.13, nous donnent des indications sur la perception de la convivialité et de l'apprentissage du système. En effet, [Lewis et Sauro, 2009] divisent ce score en deux facteurs, soit l'aptitude à l'apprentissage du système (question 4 et 6) et l'aptitude à utiliser le système (les autres questions). Le score d'utilisation du système est de 70,9 (75), tandis que le score d'apprentissage du système est de 19,1 (25), ce qui montre que la convivialité du système peut encore être améliorée. Parmi les questions posées, les deux qui ont obtenu les résultats les plus élevés sont "Je pense que la construction du plan de visite est facile à utiliser" et "Je pense que le système est facile à utiliser". Cela suggère que les participants ont reconnu l'utilité du système et ne sont pas frustrés de l'utiliser. Parmi les questions négatives, celle qui a abouti à la moyenne la plus basse est "Je pense que j'ai besoin d'aide pour utiliser le système", qui se justifie par le fait que les applications se basant sur ce type de raisonnement ne sont pas encore très répondues. Nous pensons qu'avec la prolifération de la technologie de l'Internet des objets, les utilisateurs seront plus à l'aise avec ce type d'applications.

6.3.5.2 CUSQ

Le CUSQ est une variante du PSSUQ (Post-Study System Usability Questionnaire), conçu par [Lewis, 2002] pour l'étude de la convivialité du système via la facette de la satisfaction de l'utilisateur en suivant la pratique psychométrique standard.

Le modèle CUSQ mesure 4 facteurs, à savoir l'utilité du système (SysUse), la qualité de l'information (InfoQual), la qualité de l'interface (IntQual) et la satisfaction globale

Questions	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	Moyen
Je pense que le système est facile à utiliser	2	5	5	4	5	4	5	4	5	3	5	3	5	4	4	63
Je pense que le système est complexe	4	2	2	1	4	2	4	1	2	3	2	3	1	4	5	40
Je pense que la construction du plan est facile	4	4	4	4	5	5	5	5	5	5	4	5	4	5	2	66
Je pense que j'ai besoin d'aide pour utiliser le système	4	2	2	1	4	2	4	3	2	3	2	2	1	4	3	39
Je pense que l'éditeur de règle est facile à utiliser	3	4	5	5	4	4	4	3	4	3	2	4	4	4	2	55
Je pense que j'ai besoin d'apprendre des choses avant d'utiliser le système	4	2	3	2	4	2	4	2	2	3	1	4	2	3	4	42
Je pense que la personnalisation du plan est facile	4	3	4	4	5	4	4	4	4	5	5	4	5	5	4	64
Je pense que le système contient des incohérences	2	4	4	3	5	4	2	1	1	1	1	5	1	5	2	41
Je trouve que les fonctionnalités du système sont bien intégrées	3	3	4	4	4	4	4	4	5	3	4	3	4	4	4	57
Je pense que le système est compliqué à utiliser	4	4	3	3	4	2	2	2	1	4	2	4	2	4	4	45



73

FIGURE 6.13 – Résultats du questionnaire SUS

(Overall). La figure 6.14 illustre le résultat de ce questionnaire où nous avons fait apparaître 3 groupes représentant respectivement SysUse (bleu), InfoQual (vert), IntQual (rouge). La colonne moyenne décrit la moyenne obtenue pour chaque question. Les résultats CUSQ sont cohérents avec ceux du questionnaire SUS. En effet, les questions «Les informations concernant les POIs sont utiles» et «La représentation de l'itinéraire est facile à interpréter» ont obtenus le score le plus élevé (5,67/7 et 5,4/7). Cela reflète la reconnaissance des participants de l'utilité du système. La question traitant l'interaction avec l'éditeur de règles est la question qui a eu le score le plus faible, révélant ainsi un besoin d'aide pour ce module. Cependant, 67% des participants ont réclamé une satisfaction globale du système.

Les scores factoriels du CUSQ sont 4,66 pour SysUse ; 5,08 pour InfoQual et 4,81 pour IntQual. Étant supérieur au moyenne définies par le benchmark de [Lewis, 2014], CART a révélé une bonne satisfaction utilisateur.

Nous faisons remarquer que, selon une étude comparative des questionnaires d'utilisabilité de [Lewis, 2018], le SUS et CUSQ sont les questionnaires les plus populaires et les plus rapides à converger vers sa moyenne.

6.3.5.3 UEQ

Le questionnaire UEQ est un standard permettant d'évaluer l'expérience utilisateur pour les produits interactifs. Il est conçu dans un format qui permet aux utilisateurs d'exprimer instantanément les sentiments, impressions et attitudes qu'ils ressentent lorsqu'ils

Questions	1	2	3	4	5	6	7	Moyenne
Globalement, le système est facile à utiliser	0	2	3	1	3	3	3	4,73
Le système est simple à utiliser	1	1	1	2	4	4	2	4,8
Le système est confortable	1	2	2	2	3	3	2	4,4
Le système est simple à prendre en main	2	2	3	3	5	2	1	4,73
La spécification des règles contextuelles est facile	2	1	2	3	4	2	1	**4,07**
Les informations concernant les POIs sont utiles	0	0	1	2	3	4	5	**5,67**
La navigation dans le système est facile	0	0	2	2	5	3	3	5,2
La représentation de l'itinéraire est simple à comprendre	0	0	2	2	3	4	4	5,4
J'ai apprécié les interfaces du système	2	1	2	2	2	4	2	4,4
Le système fournit les fonctionnalités attendues	1	0	1	3	3	4	3	5,07
Je recommande le système	1	1	2	2	3	3	3	4,73
Globalement, Je suis satisfait du système	0	1	1	2	4	4	3	5,2

FIGURE 6.14 – Résultats du questionnaire CUSQ

utilisent un produit. La figure, illustrant le questionnaire, décrit deux concepts ayant une signification opposée à évaluer sur une échelle de Likert. Dans sa version initiale, présentée par [Laugwitz *et al.*, 2008], UEQ comprend 26 items qui couvrent un large éventail d'expérience d'utilisation, mesurée en termes d'attractivité, de perspicacité, d'efficacité, de fiabilité, de stimulation et de nouveauté. Selon [Schrepp *et al.*, 2017], ces 6 facteurs peuvent être structurés en 2 groupes. Alors que l'attractivité peut être considérée comme une valeur autonome, décrivant une impression générale du produit, la perspicacité (degré de familiarité), l'efficacité (résoudre les tâches) et la fiabilité (sens du contrôle) peuvent être regroupées dans une dimension qualité pragmatique (axée sur l'objectif), et la stimulation et la nouveauté sont regroupées dans la dimension qualité hédonique (non axée sur l'objectif). Partant de cette classification, nous avons extrait une version courte du questionnaire, plus appropriée à notre plateforme. La figure présente le résultat de l'expérience utilisateur selon les facteurs de la méthodologie UEQ. Le graphique de la figure 6.15 montre une évaluation positive avec des valeurs $> 0{,}8$. Afin d'interpréter ces résultats, nous nous sommes basés sur le graphe de benchmark construit par [Schrepp *et al.*, 2017], illustré par la figure 6.16. Nous pouvons remarquer que tous les facteurs ont obtenu une valeur dépassant la moyenne ce qui confirme une évaluation positive. Le facteur ayant le score le plus élevé est la fiabilité reflétant ainsi un fort sentiment de contrôle de l'interaction avec le système.

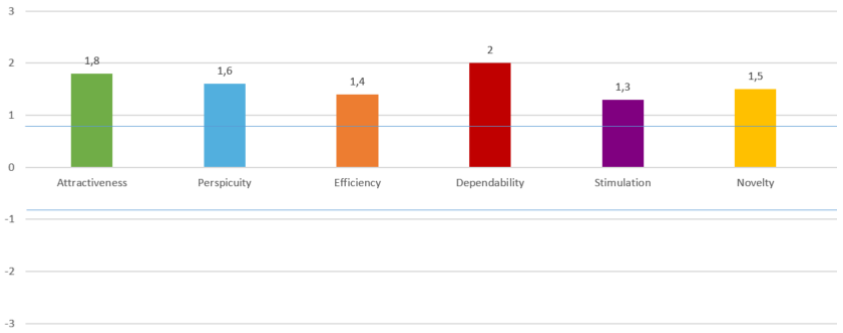


FIGURE 6.15 – Résultats du questionnaire sur l'expérience utilisateur

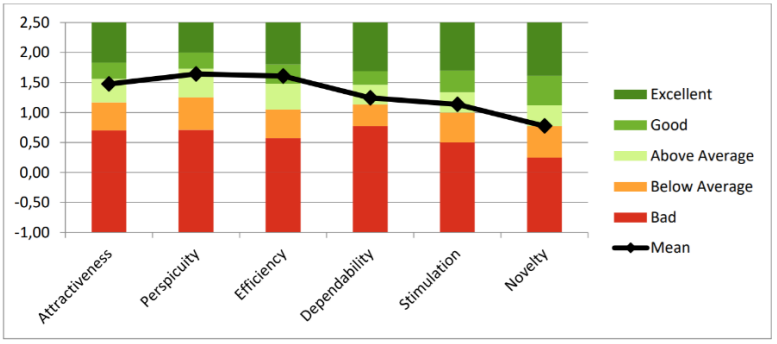


FIGURE 6.16 – Benchmark pour UEQ

6.3.5.4 Questionnaire de rétroaction

En plus des questions à l'échelle Likert, nous avons demandé aux utilisateurs de fournir une rétroaction ouverte sur le système et de suggérer des améliorations. Parmi les commentaires positifs, nous mentionnons que les participants ont apprécié les deux modes de construction du plan de visite notamment la construction progressive avec l'aide du rayon de recherche qui a bien joué son rôle pour les locaux. En plus, la fonctionnalité de personnalisation du plan de visite par l'ajustement flexible des POIs a été reconnue intéressante par les participants. Les commentaires négatifs/constructifs se rapportent à la création de règles. L'un des participants a suggéré de mettre en évidence les opérateurs logiques de combinaison des conditions et des actions. Une autre proposition concerne l'ajout d'une synthèse en langage naturel de la règle créée en plus de la description graphique. Des remarques sur la sauvegarde des règles créées a été également mentionnées.

6.3.6 Menaces à la validité

Nous analysons certains effets qui peuvent menacer la validité de l'étude utilisateurs et mettre en évidence dans quelles conditions la conception de l'étude offre des avantages et dans quelles circonstances elle pourrait échouer.

Effet de la démonstration sur l'expérience utilisateur : nous avons essayé d'atténuer cette menace en ne divulguant pas beaucoup d'informations au cours de la courte démonstration de formation afin qu'elle n'influence pas sur l'apprentissage de la plateforme.

Représentativité des résultats : compte tenu du contexte dans lequel l'étude s'est déroulée, les résultats pourraient être représentatifs malgré l'échantillon réduit. Notre objectif était d'évaluer l'utilisabilité du paradigme d'interaction, et non la capacité de l'outil à améliorer la productivité des utilisateurs du domaine. Ainsi, nous croyons que la représentativité des résultats n'est pas très affectée. Cette menace peut être corrigée par l'étendu

de l'étude. Nous supposons que ceci ne contrasterait pas avec les résultats de notre étude, mais accentuerait plutôt la tendance vers une expérience utilisateur positive en termes de fiabilité.

Conclusion

Dans ce chapitre, nous avons présenté la plateforme CART, l'outil de mashup implémentant notre approche de configuration pour la planification touristique en région Centre Val de Loire. Dans l'ensemble, les évaluations ont fourni des résultats intéressants soulignant un réel potentiel pour l'approche de mashup spécifique à un domaine pour la conception de systèmes interactifs personnalisés Les participants ont fait preuve d'un bon niveau de compréhension des fonctionnalités proposées par CART. L'une des principales constatations est liée à la facilité avec laquelle notre échantillon a compris que les composantes devaient être reliées entre elles pour que l'information puisse circuler entre les différents services. Cela est dû à la représentativité visuelle des composants et surtout la visualisation immédiate de leurs effets ce qui a augmenté l'utilité perçue par l'utilisateur et l'efficacité, c'est-à-dire la perception que les utilisateurs ont le contrôle du processus.

Bien que la puissance d'expression de l'éditeur de règles n'a pas diminué la performance et la satisfaction de l'utilisateur, certaines suggestions d'amélioration ont été mentionnées. En s'intéressant au « DO It Yourself », nous sommes conscients que la simplicité des règles est un aspect important à renforcer afin d'assurer « une pente douce de difficulté ». Ceci fera l'objectif d'une perspective que nous entamons où une étude sur un échantillon plus large sera effectuée afin de cerner les difficultés rencontrées.

Chapitre 7

Conclusion Générale

Apperçu

Bien que l'information soit accessible facilement, les utilisateurs se retrouvent souvent submergés par une masse importante de données, ce qui donne lieu au phénomène **d'infobésité**. En particulier, le tourisme est une activité de loisir bien appréciée, mais la planification d'itinéraire est devenue une tâche fastidieuse avec la dispersion de l'information. Quels sont les lieux d'intérêt à visiter ? Comment se déplacer ? Où manger ? sont quelques questions typiques que les visiteurs se posent quand ils s'apprêtent à voyager. Si les systèmes de recommandation sont la solution qui suggère des éléments potentiellement intéressants pour l'utilisateur, un effort d'agrégation reste encore à faire par ce dernier pour construire son itinéraire. En effet, ces systèmes prédisent les POIs les plus appropriés en fonction des préférences de l'utilisateur et les retournent souvent sous forme de liste de classement en négligeant les dépendances entre les éléments recommandés et les contraintes de l'utilisateur. Par conséquent, pour planifier sa visite, un utilisateur est tenu de visiter plusieurs sites Web/applications et explorer les données pour accéder à l'information adaptée à ses besoins. Il est ainsi évident qu'une plateforme qui unifie l'information dont le touriste a besoin est utile. Elle lui permettrait de construire facilement son itinéraire personnalisé.

Le mashup de données est un ensemble de techniques et d'approches conviviales permettant aux utilisateurs de récupérer les données à partir de sources multiples, de les fusionner et de les visualiser dans un seul espace numérique. Les utilisateurs de mashups peuvent être classifiés en deux types : les utilisateurs programmeurs finaux et les utilisateurs ordinaires/novices finaux, n'ayant pas des compétences techniques. L'analyse des outils de mashup a révélé qu'ils ne conviennent qu'aux développeurs utilisateurs finaux. Ceci s'explique principalement, comme décrit dans le chapitre 2, par le fait qu'il n'existe souvent pas de représentation de haut niveau qui puisse permettre aux utilisateurs finaux

de formuler leurs besoins en information. Le développement orienté utilisateur final est un moyen de résoudre ce problème dans le but de rendre autonomes les utilisateurs de manière à ce qu'ils puissent participer efficacement au processus de développement. Malgré les efforts, il est encore difficile pour les utilisateurs, aujourd'hui, avec les solutions existantes, de réaliser leurs besoins individuels. Par conséquent, le consensus qui est dégagé est que le problème de la vue unifiée personnalisable n'est toujours pas résolu.

Ainsi, nous proposons dans cette thèse une approche qui *fait le mashup* de techniques afin de fournir aux utilisateurs du domaine une solution pour configurer leurs propres itinéraires personnalisés satisfaisant leurs contraintes.

7.1 Résumé des contributions

Dans cette thèse, nous nous sommes intéressés à la problématique d'intégration de données afin de répondre à la question principale : *Comment offrir une vision globale unifiée de l'information qui soit personnalisable ?*. Notre but se focalise sur le domaine touristique afin d'améliorer l'expérience du visiteur par la construction de visites sur mesure. A cette fin, nous avons proposé une approche de mashup qui s'appuie sur la technique de configuration, impliquant efficacement les utilisateurs finaux, potentiellement novices n'ayant pas des compétences en programmation, dans le processus d'intégration. Contrairement aux solutions de mashup existantes, notre approche de configuration présente une idée de mashup spécifique au domaine et contrôlée par l'utilisateur. Convaincus par le besoin d'outil qui "parle le langage des utilisateurs" afin de les aider efficacement dans la prise de décision, nous nous sommes concentrés sur un domaine donné.

À cet égard, nous avons présenté le méta-modèle de mashup et le modèle de concept de domaine, le modèle syntaxique du domaine et avons montré comment ceux-ci peuvent être fusionnés dans un méta-modèle de mashup spécifique au domaine. Ils fournissent une base consolidée et une puissance expressive à un outil de mashup orientée utilisateur final.

Le chapitre 3 décrit cette méthodologie de mashup spécifique au domaine et présente son implémentation sous forme de modèle fonctionnel de configuration. Ce dernier adopte un paradigme visuel pour guider les utilisateurs dans le processus d'agrégation ce qui permet de répondre au premier verrou à savoir, *Comment l'utilisateur final peut faire le mashup des APIs Web ?*. Dans une logique de composition incrémentale, l'utilisateur connecte les composants de configuration pour définir son mashup qui est exécuté par le framework de configuration. De cette manière, les utilisateurs ne se préoccupent pas de tâches de mapping des données. Cette tâche est déléguée au framework de configuration qui intègre les données en s'appuyant sur les opérateurs de configuration. Ce dernier s'articule en deux phases. La première phase de mapping visuel permet d'identifier les ressources de

données à invoquer La seconde est une phase de composition qui vise à appliquer les opérateurs de configuration sur les propriétés des ressources pour les connecter et générer un jeu de solutions de configuration. L'ensemble de solutions de configuration décrit la caractéristique de validité vérifiant la satisfaction des contraintes de l'utilisateur, en plus de la représentativité qui exprime une notion de couverture, en l'occurrence géographique, en relation avec le domaine touristique et qui se généralise pour d'autres domaines comme nous le montrons dans la section suivante.

Notre seconde contribution vise à répondre à la question suivante : *Quel rôle joue le contexte dans le développement du mashup ?*. Elle consiste à enrichir le modèle fonctionnel de configuration de mashup par une couche de sensibilité au contexte afin de fournir un support pour la prise en considération du contexte dans le développement d'applications de mashup dans le but d'adapter le contenu intégré aux besoins situationnels des utilisateurs. Le modèle fonctionnel et contextuel vise à transformer le mashup agnostique au contexte en un mashup contextuel par la modélisation explicite du comportement adaptatif. La spécification de la situation contextuelle suit le paradigme événementiel utilisant les règles de type événement-condition-action, offrant un bon compromis entre l'expressivité et la simplicité. Afin de pouvoir adapter le mashup aux besoins contextuels des utilisateurs, nous transformons cette syntaxe visuelle en langage compréhensible par la machine. Ainsi, nous réalisons une conversion des règles contextuelles abstraites en règles définies en format RuleML. L'information contextuelle extraite est exploitée dans le modèle d'arbre de dimensions contextuelles qui permet la représentation de toutes les perspectives possibles caractérisant le contexte au moyen du concept générique de dimension de contexte. Ce modèle conceptuel, dérivé de la technique 5W-1H, est employé pour construire le schéma de mashup décrivant un répertoire de ressources permet la transformation de la requête initialement agnostique au contexte en requête contextuelle.

Enfin notre troisième contribution vise à augmenter l'approche de configuration par un modèle de reconfiguration permettant d'attribuer une réponse au dernier verrou *Comment rendre le mashup plus personnalisable ?*. Dans la logique de l'innovation orientée utilisateur, nous proposons le mashup élastique qui offre l'opportunité d'accommoder des besoins de personnalisation multiples en déplaçant la responsabilité aux utilisateurs finaux pour créer leurs propres applications. Cette optique de personnalisation implique en effet, une reconfiguration des composants du mashup. A cet égard nous avons couplé les techniques de mashup avec l'exploitation des Données Ouvertes Liées dans un modèle de reconfiguration. Ce dernier, décrit dans le chapitre 5, part du modèle ontologique de DATAtourisme qui décrit l'information touristique fournie en open data. Ainsi, nous proposons un enrichissement du modèle DATAtourisme pour construire une source d'information riche. Cette source de données est exploitée ensuite par la mesure de similarité LODS dans le but de

calculer les alternatives de reconfiguration via la navigation dans un graphe de connaissance. Cette représentation permet de décrire les relations sémantiques entre les entités et d'exprimer par conséquent une complémentarité de la configuration qui se traduit par une compatibilité entre les ressources.

Comme preuve de faisabilité, nous avons développé l'outil CART, décrit dans le chapitre 6, qui implémente l'approche de configuration pour le mashup touristique, guidant les utilisateurs à planifier leurs séjours en région Centre-Val de Loire. Pour l'évaluation de notre approche, nous avons effectué une évaluation par les utilisateurs par le biais de questionnaires standards mesurant les différentes facettes de la satisfaction utilisateur. Le résultat de cette étude montre un potentiel de notre approche de configuration de mashup en offrant une expérience complètement personnalisable. De nombreux commentaires et propositions d'améliorations ont été recueillis au cours de cette étude que nous allons intégrer dans la prochaine version de CART.

7.2 Discussion

Nous présentons dans cette section une vue globale de notre approche de configuration de mashup annotée des leçons apprises. Bien que nous nous sommes focalisés, dans cette thèse, sur le domaine touristique, nous pensons que l'approche de configuration se transpose dans d'autres scénarii, où les utilisateurs finaux cherchent des solutions personnalisées qui fournissent un environnement propice à l'agrégation flexible de données.

Par exemple, lors d'un achat en ligne, les utilisateurs cherchent à construire un paquet de produit afin de bénéficier d'un rabais. Dans ce type de recommandation de produit, en plus de la liste des iPhones disponibles dans les limites du budget, il est également souhaitable de présenter, avec chaque iPhone, un ensemble de forfaits, dont chacun se compose d'articles compatibles qui peuvent être achetés avec l'iPhone et dont le prix total est dans le budget de l'utilisateur. Le mashup ainsi configuré vérifie bien les caractéristiques de validité et de complémentarité mentionnés dans cette thèse. Dans ce cas, la validité est exprimée par la limite budgétaire et la complémentarité est déterminée par la compatibilité entre l'Iphone et chaque article recommandé dans le forfait qui peut être dérivé des historiques de co-navigation et de co-achat d'articles ou issue des recommandations du fabricant. Ainsi, la configuration est représentée sous forme d'étoile où un ensemble d'éléments compatibles gravitent autour d'un élément central.

La formation d'équipe est un autre scénario d'application de la configuration de mashup, où il s'agit de construire une équipe d'experts d'une taille limitée ayant des compétences complémentaires (un gestionnaire, un avocat, un ingénieur, etc). De même pour une plateforme de crowdsourcing, l'affectation de tâches aux travailleurs peut être vue comme un

problème de configuration de mashup. Puisque chaque travailleur ne s'intéresse pas à chaque tâche, il est important de configurer un mashup personnalisé de tâches à chaque travailleur. Cette configuration regroupe des tâches compatibles avec les compétences du travailleur et valides par rapport à la rémunération souhaitée. Le mashup ainsi configuré, représentant un ensemble de tâches diverses, est plus intéressant pour les travailleurs qu'une liste de tâches classées. [Binzagr et Medjahed, 2018] proposent CrowdMashup, une approche de crowdsourcing pour la recommandation des équipes de travail via une analyse des besoins de la communautés de développeurs et des répertoires d'APIs afin de déduire les intérêts des développeurs pour les APIs.

De ce fait, l'aspect spécificité au domaine dans notre approche vise à donner une puissance expressive aux utilisateurs leur permettant une exploration de l'espace de mashup via des représentations visuelles pour masquer la complexité technologique. L'idée est d'intégrer efficacement l'humain dans la boucle de mashup avec une logique interactive qui est essentielle pour un processus de personnalisation, en particulier dans un domaine de nature subjective. Les défis abordés dans cette thèse nous ont permis de tirer quelques conclusions. La première se rapporte à l'impact visuel qui joue un rôle important dans la construction interactive du mashup contrairement aux objets « plats ». Toutefois, nous avons constaté l'impossibilité d'éliminer complètement la barrière technique entre un outil et l'utilisateur final en raison de la complexité qui s'installe en développant les fonctionnalités. Ceci découle de l'équation $utilite = valeur \ / \ effort$ qui implique que l'effort requis pour le mashup diminue la valeur et neutralise donc l'utilité. A travers l'approche de configuration, nous avons fait le compromis entre la flexibilité et la puissance expressive dans le processus de mashup.

7.3 Perspectives

Au cours de ce travail de thèse, nous avons observé un certain nombre de directions intéressantes que, lorsqu'elles seront appliquées, peuvent renforcer notre approche de configuration de mashup. Nous présentons dans ce qui suit ces orientations futures.

7.3.1 Mashup pour l'Internet des objets

Nous prévoyons étendre notre approche de configuration pour intégrer les objets connectés dans le processus de mashup. Le potentiel de l'Internet des Objets (IOT) est de plus en plus reconnu permettant l'accès à un pool de ressources. l'IOT fournit ainsi, une nouvelle valeur ajoutée en connectant les dispositifs physiques aux environnements virtuels en fonction de leur contexte. Ceci permet d'étendre le concept de mashup aux applications IOT, combinant ainsi l'espace physique et l'espace informationnel si chaque objet expose

ses fonctionnalités comme un service Web.

Le défi ici pour le mashup des services de l'internet des objets est la nature hétérogène des capteurs qui nécessite une abstraction de la couche inférieure du dispositif à une couche d'accès commune afin de collecter les données provenant de différentes plates-formes de capteurs. Un middleware idéal de mashups de services de l'IOT devrait fournir des abstractions à différents niveaux, tels qu'un environnement hétérogène pour les périphériques physiques et les interfaces logicielles, et le processus de création de services intelligents. De ce fait, le concept de mashup est enrichi par une intégration transparente des objets physiques intelligents fournissant un nouveau type d'applications. Le smart tourisme est un exemple de ce type d'application introduisant de nouvelles opportunités pour l'industrie du tourisme en facilitant l'accès et l'interaction avec un large éventail d'informations. Par exemple, les capteurs distribués dans les points d'intérêt peuvent détecter des informations sur la température, le niveau de pollen ou la pollution, ce qui permet, à leur intégration dans le processus de mashup et d'affiner ainsi la phase de sélection où ce type de POIs ne sera pas proposé aux utilisateurs à risque afin d'éviter les accidents et contribuer à accroître le niveau de satisfaction des touristes.

7.3.2 Recommandation de règles pour les utilisateurs finaux

La gestion actuelle des règles contextuelles est réduite à la création des règles, nous souhaitons l'enrichir afin de permettre aux utilisateurs de les réutiliser dans des contextes différents ou de les partager avec d'autres utilisateurs, voir de les utiliser comme point de départ pour créer de nouvelles règles. Il est donc important de fournir aux utilisateurs des outils leur permettant de sauvegarder les règles pour une utilisation ultérieur. Par exemple, Patrick, un directeur d'entreprise, a tendance à planifier plusieurs séjours dans des contextes différents (seul, en couple ou en famille). Il souhaite disposer d'un mécanise qui lui permet de réutiliser les règles créées au paravent pour les adapter au nouveau contexte sans avoir à créer une nouvelle règle, en partant de rien. Nous prévoyons d'intégrer des techniques de recommandation afin d'améliorer à la fois la réutilisation et la définition des règles de déclenchement, aidant ainsi les utilisateurs à personnaliser facilement leurs mashups (applications Web).

En fait, lorsqu'il s'agit de règles de navigation déjà créées et partagées par d'autres utilisateurs, un système de recommandation (RecRules) pourrait suggérer des règles pertinentes à réutiliser. Lors de la définition d'une nouvelle règle, un système de recommandation pourrait aider les utilisateurs à compléter leurs règles, par exemple, en suggérant dynamiquement des règles pertinentes sur la base de ce que l'utilisateur a déjà défini. L'idée derrière RecRules est de recommander des règles CA sur la base des comportements que les utilisateurs souhaitent définir. C'est la recommandation par fonctionnalité qui définit une

nouvelle catégorie de systèmes de recommandation pour les utilisateurs finaux s'appuyant sur les besoins réels des utilisateurs finaux. Outre les développeurs, quelques travaux se focalisent sur les utilisateurs finaux, en suggérant des objets intelligents basées sur les préférences et les intérêts des utilisateurs, afin d'optimiser le temps et le coût d'utilisation de l'IOT dans une situation particulière. Au-delà des objets, RecRules propose de recommander des règles qui relient des paires d'entités. A cette fin, nous pensons qu'un processus de raisonnement sémantique pour enrichir les règles avec des informations sémantiques est nécessaire pour permettre de découvrir des connexions cachées entre les règles en termes de fonctionnalités.

7.3.3 Représentation de haut-niveau des règles contextuelles

Lors de la rétro-action avec les participants à l'évaluation de CART, nous avons constaté le besoin d'une représentation de haut niveau des règles contextuelle permettant aux utilisateurs de définir des règles abstraites de type "s'il fait mauvais, alors je préfère faire des visites à l'intérieur". Ce besoin nous a guidé à réfléchir à une représentation ontologique pour fournir des représentations abstraites. Le raisonnement sémantique nous offre un support solide pour définir des règles de haut niveau. Ainsi, nous souhaitons transformer le modèle de représentation des dimensions contextuelles en modèle ontologique et l'intégrer au schéma DATAtourisme décrivant les lieux d'intérêt afin de construire un modèle sémantique riche capable d'interpréter des comportements contextuels plus génériques.

7.3.4 Mashup cognitif

Bien que le paradigme événementiel de la programmation par condition-action soit adopté par la plupart des outils destinés aux utilisateurs finaux, il présente ses propres limites. En effet, on ne peut pas toujours supposer que tous les utilisateurs finaux seraient capables de définir des règles contextuelles. Pour soulager les utilisateurs finaux de la surcharge cognitive, nous devons leur fournir des interactions plus naturelles leur permettant de décrire leurs besoins de manière plus simples. L'informatique cognitive émerge comme le moteur qui alimente les interactions naturelles entre les humains, les logiciels et les dispositifs soutenus par les progrès de l'Intelligence Artificielle. Il s'agit de stimuler le processus de la pensée humaine dans un modèle informatisé en mesure d'identifier les intentions de l'utilisateur et de convertir les données brutes en donnés intelligentes afin de répondre à ses besoins en connaissance de sa situation.

Les chatbots ou les assistants numériques est une instance de services cognitifs permettent à l'utilisateur d'interroger un contenu complexe, de sorte qu'une interaction plus humaine avec lui est activée. Ces services cognitifs peuvent se présenter comme des APIs

qui exposent des modèles d'apprentissage manipulant des données multimédia pour la reconnaissance vocale, le traitement du langage, etc, appelés MLaaS- Machine Learning as a Services. C'est ce dont nous avons besoin pour profiter facilement du potentiel de l'apprentissage et améliorer le processus de mashup. Le défi ainsi consiste à reconstruire le processus de mashup afin de créer une application intelligente capable d'abstraire les données en information et à intégrer un comportement plus humain comme les sentiments ou les émotions dans le processus de composition. Ceci nécessite de nouvelles techniques d'intégration d'APIs car la compréhension des intentions des utilisateurs est plus probabiliste que déterministe. Il s'agit d'un passage d'un monde de services déterministes, traitant des instructions précises par invocation de petits ensembles de services, à un raisonnement probabiliste pour clarifier et identifier les intentions des utilisateurs via un mapping entre les intentions et un ensemble évolutif de services s'appuyant sur modèle conversationnel configurable.

7.3.5 Mashup hautement personnalisé

L'injection numérique a transformé l'industrie économique en industrie intelligente proposant des services personnalisés. En particulier, dans l'industrie du tourisme, Airbnb et Uber offrent un modèle de consommateur à consommateur visant à améliorer l'expérience client. Ainsi, le besoin de combiner argent, technologie et connaissance semble nécessaire pour répondre aux changements et fournir des plateformes innovantes. Nous pensons étendre notre approche de mashup pour couvrir à la fois la planification, la réservation et le paiement. Il s'agit d'ouvrir le processus de configuration pour faire participer en plus de l'utilisateur les autres partenaires impliqués dans la prise de décision. Nous constituons, ainsi, un écosystème permettant aux utilisateurs de planifier leurs déplacements puis réserver des places ou acheter des tickets sans avoir à passer d'une ressource à une autre et profiter d'une offre sur mesure leur permettant d'économiser de l'argent par le biais de la désintermédiation. Nous parlons alors d'un mashup de services hautement personnalisé via une configuration ouverte impliquant plusieurs collaborateurs. A cette fin, nous proposons de s'appuyer sur la technologie Blockchain comme technologie collaborative.

Grâce à un réseau alimenté par une chaîne de blocs distribués et un ensemble de contrats intelligents, nous pouvons fournir une solution de personnalisation intelligente par le biais d'une négociation transparente entre les opérateurs touristiques et le touriste, lui permettant de bénéficier de la meilleure offre. Le consensus des paires est automatiquement implémenté par un contrat intelligent assurant une réduction des coûts.

Annexes

Annexe A

Annexe 1

Listing A.1 – itinerary.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning" vc:minVersion="1.0">
  <xs:element name="itinerary">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="periode" />
        <xs:element ref="days" maxOccurs="unbounded"/>
        <xs:element ref="budget"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="periode">
    <xs:complexType>
      <xs:attribute name="end" use="required" type="xs:dateTime"/>
      <xs:attribute name="start" use="required" type="xs:dateTime"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="days">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="day"/>
      </xs:sequence>
```

```
  </xs:complexType>
</xs:element>
<xs:element name="budget">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:attribute name="devise" use="required"  type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="day">
  <xs:complexType>
    <xs:sequence >
      <xs:element ref="steps"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"/>
  </xs:complexType>
</xs:element>
<xs:element name="steps">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="step"/>

    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="step">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" ref="POI"/>
    </xs:sequence>
    <xs:attribute name="end" use="required" type="xs:dateTime"/>
    <xs:attribute name="start" use="required" type="xs:dateTime"/>
  </xs:complexType>
</xs:element>
<xs:element name="POI">
  <xs:complexType>
```

```
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="rating"/>
      <xs:element ref="position"/>
      <xs:element ref="handicap"/>
      <xs:element ref="contact"/>
      <xs:element ref="description"/>
      <xs:element ref="styles"/>
      <xs:element ref="themes"/>
      <xs:element ref="types"/>
      <xs:element ref="photos"/>
      <xs:element ref="prices"/>
      <xs:element ref="opening"/>
      <xs:element ref="duration"/>
    </xs:sequence>
    <xs:attribute name="id" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
<xs:element name="rating" type="xs:decimal"/>
<xs:element name="position">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="latitude"/>
      <xs:element ref="longitude"/>
      <xs:element ref="address"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="latitude" type="xs:decimal"/>
<xs:element name="longitude" type="xs:decimal"/>
<xs:element name="address" type="xs:string"/>
<xs:element name="handicap" type="xs:boolean" default="false"/>
<xs:element name="contact">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Telephone"/>
      <xs:element ref="Mail"/>
      <xs:element ref="Web"/>
```

```
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Telephone" type="xs:string"/>
<xs:element name="Mail" type="xs:string"/>
<xs:element name="Web" type="xs:anyURI"/>
<xs:element name="description" type="xs:string"/>
<xs:element name="styles">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="style"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="style" type="xs:string"/>
<xs:element name="themes">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="theme"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="theme" type="xs:string"/>
<xs:element name="types">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="type"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="type" type="xs:string"/>
<xs:element name="photos">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="photo"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="photo" type="xs:anyURI"/>
<xs:element name="prices">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="price"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="price">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="value"/>
      <xs:element minOccurs="0" ref="conditions"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="value" type="xs:integer"/>
<xs:element name="conditions">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="condition"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="condition" type="xs:string"/>
<xs:element name="opening">
  <xs:complexType id="openingType">
    <xs:sequence>
      <xs:element name="periode"  >
      <xs:complexType >
        <xs:sequence>
          <xs:element name="start" type="xs:date"/>
          <xs:element name="end" type="xs:date"/>
          <xs:element name="day" >
            <xs:complexType >
              <xs:sequence>
                <xs:element ref="name"/>
                <xs:element ref="timesSlot" maxOccurs="unbounded" />
```

```
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="duration" type="xs:integer"/>
  <xs:element name="timesSlot">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="start" minOccurs="1"/>
        <xs:element ref="end"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="start" type="xs:time"/>
  <xs:element name="end" type="xs:time"/>
</xs:schema>
```

Bibliographie

[Abdulkarem *et al.*, 2019] Abdulkarem, H. F., Abozaid, G. Y. et Soliman, M. I. (2019). Context-aware recommender system frameworks, techniques, and applications : A survey. *In 2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pages 180–185. IEEE.

[Adomavicius et Tuzhilin, 2005] Adomavicius, G. et Tuzhilin, A. (2005). Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*, (6):734–749.

[Aghaee et Pautasso, 2014] Aghaee, S. et Pautasso, C. (2014). End-user development of mashups with naturalmash. *Journal of Visual Languages & Computing*, 25(4):414–432.

[Alegre *et al.*, 2016] Alegre, U., Augusto, J. C. et Clark, T. (2016). Engineering context-aware systems and applications : A survey. *Journal of Systems and Software*, 117:55–83.

[Alliance, 2013] Alliance, O. M. (2013). Oma emml documentation.

[Amer-Yahia *et al.*, 2014] Amer-Yahia, S., Bonchi, F., Castillo, C., Feuerstein, E., Mendez-Diaz, I. et Zabala, P. (2014). Composite retrieval of diverse and complementary bundles. *IEEE Transactions on Knowledge and Data Engineering*, 26(11):2662–2675.

[Ardito *et al.*, 2018] Ardito, C., Buono, P., Desolda, G. et Matera, M. (2018). From smart objects to smart experiences : An end-user development approach. *International Journal of Human-Computer Studies*, 114:51–68.

[Ardito *et al.*, 2017] Ardito, C., Costabile, M. F., Desolda, G. et Matera, M. (2017). A three-layer meta-design model for addressing domain-specific customizations. *In New Perspectives in End-User Development*, pages 99–120. Springer.

[Arigi *et al.*, 2018] Arigi, L. R. H., Baizal, Z. A. et Herdiani, A. (2018). Context-aware recommender system based on ontology for recommending tourist destinations at bandung. *In Journal of Physics : Conference Series*, volume 971, page 012024. IOP Publishing.

[Asahiro *et al.*, 2000] Asahiro, Y., Iwama, K., Tamaki, H. et Tokuyama, T. (2000). Greedily finding a dense subgraph. *Journal of Algorithms*, 34(2):203–221.

[Auer *et al.*, 2007] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. et Ives, Z. (2007). Dbpedia : A nucleus for a web of open data. *In The semantic web*, pages 722–735. Springer.

[Baltrunas *et al.*, 2012] Baltrunas, L., Ludwig, B., Peer, S. et Ricci, F. (2012). Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing*, 16(5):507–526.

[Bangor *et al.*, 2009] Bangor, A., Kortum, P. et Miller, J. (2009). Determining what individual sus scores mean : Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123.

[Bangor *et al.*, 2008] Bangor, A., Kortum, P. T. et Miller, J. T. (2008). An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594.

[Bao *et al.*, 2015] Bao, J., Zheng, Y., Wilkie, D. et Mokbel, M. (2015). Recommendations in location-based social networks : a survey. *GeoInformatica*, 19(3):525–565.

[Baral et Li, 2016] Baral, R. et Li, T. (2016). Maps : A multi aspect personalized poi recommender system. *In Proceedings of the 10th ACM conference on recommender systems*, pages 281–284. ACM.

[Baral *et al.*, 2018] Baral, R., Li, T. et Zhu, X. (2018). Caps : Context aware personalized poi sequence recommender system. *arXiv preprint arXiv :1803.01245*.

[Barricelli *et al.*, 2019] Barricelli, B. R., Cassano, F., Fogli, D. et Piccinno, A. (2019). End-user development, end-user programming and end-user software engineering : A systematic mapping study. *Journal of Systems and Software*, 149:101–137.

[Barricelli et Valtolina, 2017] Barricelli, B. R. et Valtolina, S. (2017). A visual language and interactive system for end-user development of internet of things ecosystems. *Journal of Visual Languages & Computing*, 40:1–19.

[Benouaret et Lenne, 2016] Benouaret, I. et Lenne, D. (2016). A package recommendation framework for trip planning activities. *In Proceedings of the 10th ACM Conference on Recommender Systems*, pages 203–206. ACM.

[Berners-Lee, 2006] Berners-Lee, T. (2006). Linked data.

[Bhaskara *et al.*, 2010] Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U. et Vijayaraghavan, A. (2010). Detecting high log-densities : an o (n 1/4) approximation for densest k-subgraph. *In Proceedings of the forty-second ACM symposium on Theory of computing*, pages 201–210.

[Binzagr et Medjahed, 2018] Binzagr, F. et Medjahed, B. (2018). Crowdmashup : recommending crowdsourcing teams for mashup development. *In International Conference on Service-Oriented Computing*, pages 679–693. Springer.

[Bizer *et al.*, 2011] Bizer, C., Heath, T. et Berners-Lee, T. (2011). Linked data : The story so far. *In Semantic services, interoperability and web applications : emerging concepts*, pages 205–227. IGI Global.

[Bleier *et al.*, 2018] Bleier, A., De Keyser, A. et Verleye, K. (2018). Customer engagement through personalization and customization. *In Customer Engagement Marketing*, pages 75–94. Springer.

[Bobadilla *et al.*, 2013] Bobadilla, J., Ortega, F., Hernando, A. et Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-based systems*, 46:109–132.

[Bolchini *et al.*, 2013] Bolchini, C., Quintarelli, E. et Tanca, L. (2013). Carve : Context-aware automatic view definition over relational databases. *Information Systems*, 38(1):45–67.

[Boley *et al.*, 2010] Boley, H., Paschke, A. et Shafiq, O. (2010). Ruleml 1.0 : the overarching specification of web rules. *In International Workshop on Rules and Rule Markup Languages for the Semantic Web*, pages 162–178. Springer.

[Bouguettaya *et al.*, 2014] Bouguettaya, A., Sheng, Q. Z. et Daniel, F. (2014). *Web services foundations*. Springer.

[Braunhofer *et al.*, 2013] Braunhofer, M., Elahi, M., Ge, M., Ricci, F. et Schievenin, T. (2013). Sts : Design of weather-aware mobile recommender systems in tourism. *In AI* HCI@ AI* IA*.

[Braunhofer *et al.*, 2014] Braunhofer, M., Elahi, M. et Ricci, F. (2014). Sts : A context-aware mobile recommender system for places of interest. *In UMAP Workshops*. Citeseer.

[Braunhofer et Ricci, 2016] Braunhofer, M. et Ricci, F. (2016). Contextual information elicitation in travel recommender systems. *In Information and Communication Technologies in Tourism 2016*, pages 579–592. Springer.

[Braunhofer et Ricci, 2017] Braunhofer, M. et Ricci, F. (2017). Selective contextual information acquisition in travel recommender systems. *Information Technology & Tourism*, 17(1):5–29.

[Brilhante *et al.*, 2013] Brilhante, I., Macedo, J. A., Nardini, F. M., Perego, R. et Renso, C. (2013). Where shall we go today ? : planning touristic tours with tripbuilder. *In Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 757–762. ACM.

[Brilhante *et al.*, 2015] Brilhante, I. R., Macedo, J. A., Nardini, F. M., Perego, R. et Renso, C. (2015). On planning sightseeing tours with tripbuilder. *Information Processing & Management*, 51(2):1–15.

[Buqing *et al.*, 2014] Buqing, C., Tang, M. et Huang, X. (2014). Cscf : A mashup service recommendation approach based on content similarity and collaborative filtering. *International Journal of Grid & Distributed Computing*, 7(2).

[Cabitza *et al.*, 2017] Cabitza, F., Fogli, D., Lanzilotti, R. et Piccinno, A. (2017). Rule-based tools for the configuration of ambient intelligence systems : a comparative user study. *Multimedia Tools and Applications*, 76(4):5221–5241.

[Cai *et al.*, 2018] Cai, L., Xu, J., Liu, J. et Pei, T. (2018). Integrating spatial and temporal contexts into a factorization model for poi recommendation. *International Journal of Geographical Information Science*, 32(3):524–546.

[Cano *et al.*, 2011] Cano, A. E., Burel, G., Dadzie, A.-S. et Ciravegna, F. (2011). Topica : A tool for visualising emerging semantics of pois based on social awareness streams. *In 10th Int. Semantic Web Conf (ISWC2011)(Demo Track)*.

[Cano *et al.*, 2013] Cano, A. E., Dadzie, A.-S. et Ciravegna, F. (2013). Travel mashups. *In Semantic Mashups*, pages 321–347. Springer.

[Cao *et al.*, 2013] Cao, B., Liu, J., Tang, M., Zheng, Z. et Wang, G. (2013). Mashup service recommendation based on user interest and social network. *In 2013 IEEE 20th international conference on web services*, pages 99–106. IEEE.

[Cappiello *et al.*, 2015] Cappiello, C., Matera, M. et Picozzi, M. (2015). A ui-centric approach for the end-user development of multidevice mashups. *ACM Transactions on the Web (TWEB)*, 9(3):11.

[Cappiello *et al.*, 2012] Cappiello, C., Matera, M., Picozzi, M., Caio, A. et Guevara, M. T. (2012). Mobimash : end user development for mobile mashups. *In Proceedings of the 21st International Conference on World Wide Web*, pages 473–474. ACM.

[Casati, 2011] Casati, F. (2011). How end-user development will save composition technologies from their continuing failures. *In International Symposium on End User Development*, pages 4–6. Springer.

[Casati *et al.*, 2012] Casati, F., Daniel, F., De Angeli, A., Imran, M., Soi, S., Wilkinson, C. R. et Marchese, M. (2012). Developing mashup tools for end-users : on the importance of the application domain. *International Journal of Next-Generation Computing*, 3(2).

[Casillo *et al.*, 2016] Casillo, M., Cerullo, L., Colace, F., Lemma, S., Lombardi, M. et Pietrosanto, A. (2016). An adaptive context aware app for the tourism. *In*

Proceedings of the The 3rd Multidisciplinary International Social Networks Conference on SocialInformatics 2016, Data Science 2016, pages 1–5.

[Casillo *et al.*, 2017] Casillo, M., Colace, F., De Santo, M., Lemma, S. et Lombardi, M. (2017). A context-aware mobile solution for assisting tourists in a smart environment.

[Cassani *et al.*, 2016] Cassani, V., Gianelli, S., Matera, M., Medana, R., Quintarelli, E., Tanca, L. et Zaccaria, V. (2016). On the role of context in the design of mobile mashups. *In International Rapid Mashup Challenge*, pages 108–128. Springer.

[Cervantes *et al.*, 2017] Cervantes, F., Ramos, F., Gutiérrez, L. F., Occello, M. et Jamont, J.-P. (2017). A new approach for the composition of adaptive pervasive systems. *IEEE Systems Journal*, 12(2):1709–1721.

[Chen *et al.*, 2014] Chen, C., Zhang, D., Guo, B., Ma, X., Pan, G. et Wu, Z. (2014). Tripplanner : Personalized trip planning leveraging heterogeneous crowdsourced digital footprints. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1259–1273.

[Chen *et al.*, 2017] Chen, D., Kim, D., Xie, L., Shin, M., Menon, A. K., Ong, C. S., Avazpour, I. et Grundy, J. (2017). Pathrec : Visual analysis of travel route recommendations. *In Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 364–365.

[Cheng *et al.*, 2017] Cheng, B., Zhai, Z., Zhao, S. et Chen, J. (2017). Lsmp : A lightweight service mashup platform for ordinary users. *IEEE Communications Magazine*, 55(4):116–123.

[Cheniki *et al.*, 2016] Cheniki, N., Belkhir, A., Sam, Y. et Messai, N. (2016). Lods : A linked open data based similarity measure. *In 2016 IEEE 25th International Conference on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE)*, pages 229–234. IEEE.

[CHOI, 2019] CHOI, Y. (2019). Ontolgy-based mashup model for context-aware services in internet of things application environments. *Journal of Theoretical and Applied Information Technology*, 97(4).

[Chudnovskyy *et al.*, 2012] Chudnovskyy, O., Nestler, T., Gaedke, M., Daniel, F., Fernández-Villamor, J. I., Chepegin, V., Fornas, J. A., Wilson, S., Kögler, C. et Chang, H. (2012). End-user-oriented telco mashups : the omelette approach. *In Proceedings of the 21st International Conference on World Wide Web*, pages 235–238. ACM.

[Corno *et al.*, 2019] Corno, F., De Russis, L. et Roffarello, A. M. (2019). A high-level semantic approach to end-user development in the internet of things. *International Journal of Human-Computer Studies*, 125:41–54.

[Cruz et Xiao, 2005] Cruz, I. F. et Xiao, H. (2005). The role of ontologies in data integration. *Engineering intelligent systems for electrical engineering and communications*, 13(4):245.

[Czarnecki *et al.*, 2004] Czarnecki, K., Helsen, S. et Eisenecker, U. (2004). Staged configuration using feature models. *In International Conference on Software Product Lines*, pages 266–283. Springer.

[Dalipi *et al.*, 2016] Dalipi, E., Van den Abeele, F., Ishaq, I., Moerman, I. et Hoebeke, J. (2016). Ec-iot : An easy configuration framework for constrained iot devices. *In 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 159–164. IEEE.

[Daniel *et al.*, 2009] Daniel, F., Casati, F., Benatallah, B. et Shan, M.-C. (2009). Hosted universal composition : Models, languages and infrastructure in mashart. *In International Conference on Conceptual Modeling*, pages 428–443. Springer.

[Daniel *et al.*, 2012] Daniel, F., Imran, M., Kling, F., Soi, S., Casati, F. et Marchese, M. (2012). Developing domain-specific mashup tools for end users. *In Proceedings of the 21st International Conference on World Wide Web*, pages 491–492.

[Daniel et Matera, 2008] Daniel, F. et Matera, M. (2008). Mashing up context-aware web applications : a component-based development approach. *In International Conference on Web Information Systems Engineering*, pages 250–263. Springer.

[Daniel et Matera, 2014] Daniel, F. et Matera, M. (2014). Mashups. *In Mashups*, pages 137–181. Springer.

[Daniel *et al.*, 2018] Daniel, F., Matera, M., Quintarelli, E., Tanca, L. et Zaccaria, V. (2018). Context-aware access to heterogeneous resources through on-the-fly mashups. *In International Conference on Advanced Information Systems Engineering*, pages 119–134. Springer.

[Daniel *et al.*, 2007] Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M. et Saint-Paul, R. (2007). Understanding ui integration : A survey of problems, technologies, and opportunities. *IEEE Internet Computing*, 11(3):59–66.

[Das *et al.*, 2017] Das, D., Sahoo, L. et Datta, S. (2017). A survey on recommendation system. *International Journal of Computer Applications*, 160(7).

[Deng *et al.*, 2013] Deng, T., Fan, W. et Geerts, F. (2013). On the complexity of package recommendation problems. *SIAM Journal on Computing*, 42(5):1940–1986.

[Deng *et al.*, 2015] Deng, T., Fan, W. et Geerts, F. (2015). On recommendation problems beyond points of interest. *Information Systems*, 48:64–88.

[Desolda *et al.*, 2017a] Desolda, G., Ardito, C., Costabile, M. F. et Matera, M. (2017a). End-user composition of interactive applications through actionable ui components. *Journal of Visual Languages & Computing*, 42:46–59.

[Desolda *et al.*, 2015] Desolda, G., Ardito, C. et Matera, M. (2015). Efesto : a platform for the end-user development of interactive workspaces for data exploration. *In International Rapid Mashup Challenge*, pages 63–81. Springer.

[Desolda *et al.*, 2016] Desolda, G., Ardito, C. et Matera, M. (2016). End-user development for the internet of things : Efesto and the 5w composition paradigm. *In International Rapid Mashup Challenge*, pages 74–93. Springer.

[Desolda *et al.*, 2017b] Desolda, G., Ardito, C. et Matera, M. (2017b). Empowering end users to customize their smart environments : model, composition paradigms, and domain-specific tools. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 24(2):1–52.

[Desolda *et al.*, 2017c] Desolda, G., Ardito, C. et Matera, M. (2017c). Specification of complex logical expressions for task automation : an eud approach. *In International Symposium on End User Development*, pages 108–116. Springer.

[Dey et Abowd, 2000] Dey, A. K. et Abowd, G. D. (2000). Providing architectural support for building context-aware applications.

[Di Lorenzo *et al.*, 2009] Di Lorenzo, G., Hacid, H., Paik, H.-y. et Benatallah, B. (2009). Data integration in mashups. *ACM Sigmod Record*, 38(1):59–66.

[Doan *et al.*, 2012] Doan, A., Halevy, A. et Ives, Z. (2012). *Principles of data integration*. Elsevier.

[Domínguez et Ko, 2018] Domínguez, N. et Ko, I.-Y. (2018). Mashup recommendation for trigger action programming. *In International Conference on Web Engineering*, pages 177–184. Springer.

[Duan *et al.*, 2015] Duan, Y., Fu, G., Zhou, N., Sun, X., Narendra, N. C. et Hu, B. (2015). Everything as a service (xaas) on the cloud : origins, current and future trends. *In 2015 IEEE 8th International Conference on Cloud Computing*, pages 621–628. IEEE.

[Eirinaki *et al.*, 2018] Eirinaki, M., Gao, J., Varlamis, I. et Tserpes, K. (2018). Recommender systems for large-scale social networks : A review of challenges and solutions.

[Ekaputra *et al.*, 2017] Ekaputra, F. J., Do, B.-L., Kiesling, E., Novak, N. M., Trinh, T.-D., Tjoa, A. M. et Aryan, P. R. (2017). Towards open data mashups for data journalism. *In SEMANTICS Posters&Demos*.

[Elmaghraoui *et al.*, 2017] Elmaghraoui, H., Benhlima, L. et Chiadmi, D. (2017). And/or directed graph for dynamic web service composition. *In International Conference of Cloud Computing Technologies and Applications*, pages 351–368. Springer.

[Elmeleegy *et al.*, 2008] Elmeleegy, H., Ivan, A., Akkiraju, R. et Goodwin, R. (2008). Mashup advisor : A recommendation tool for mashup development. *In 2008 IEEE International Conference on Web Services*, pages 337–344. IEEE.

[Ennals et Garofalakis, 2007] Ennals, R. J. et Garofalakis, M. N. (2007). Mashmaker : mashups for the masses. *In Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1116–1118.

[Fargier *et al.*, 2016] Fargier, H., Gimenez, P.-F. et Mengin, J. (2016). Recommendation for product configuration : an experimental evaluation.

[Feige *et al.*, 2001] Feige, U., Peleg, D. et Kortsarz, G. (2001). The dense k-subgraph problem. *Algorithmica*, 29(3):410–421.

[Felfernig *et al.*, 2016] Felfernig, A., Erdeniz, S. P., Azzoni, P., Jeran, M., Akcay, A. et Doukas, C. (2016). Towards configuration technologies for iot gateways. *In 18 th International Configuration Workshop*, volume 73.

[Felfernig *et al.*, 2014] Felfernig, A., Hotz, L., Bagley, C. et Tiihonen, J. (2014). *Knowledge-based configuration : From research to business cases*. Newnes.

[Fielding et Taylor, 2000] Fielding, R. T. et Taylor, R. N. (2000). *Architectural styles and the design of network-based software architectures*, volume 7. University of California, Irvine Doctoral dissertation.

[Franke et Piller, 2003] Franke, N. et Piller, F. (2003). Key research issues in user interaction with configuration toolkits in a mass customization system. *International Journal of Technology Management*, 26(5/6):578–599.

[Gandhi et Gheewala, 2017] Gandhi, S. R. et Gheewala, J. (2017). A survey on recommendation system with collaborative filtering using big data. *In 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pages 457–460. IEEE.

[Gao *et al.*, 2013] Gao, H., Tang, J., Hu, X. et Liu, H. (2013). Exploring temporal effects for location recommendation on location-based social networks. *In Proceedings of the 7th ACM conference on Recommender systems*, pages 93–100. ACM.

[Gao et Wu, 2017] Gao, W. et Wu, J. (2017). A novel framework for service set recommendation in mashup creation. *In 2017 IEEE International Conference on Web Services (ICWS)*, pages 65–72. IEEE.

[Gavalas *et al.*, 2017] Gavalas, D., Kasapakis, V., Konstantopoulos, C., Pantziou, G. et Vathis, N. (2017). Scenic route planning for tourists. *Personal and Ubiquitous Computing*, 21(1):137–155.

[Gavalas *et al.*, 2014] Gavalas, D., Konstantopoulos, C., Mastakas, K. et Pantziou, G. (2014). A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*, 20(3):291–328.

[Ghiani *et al.*, 2017] Ghiani, G., Manca, M., Paternò, F. et Santoro, C. (2017). Personalization of context-dependent applications through trigger-action rules. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 24(2):14.

[Ghiani *et al.*, 2016] Ghiani, G., Paternò, F., Spano, L. D. et Pintori, G. (2016). An environment for end-user development of web mashups. *International Journal of Human-Computer Studies*, 87:38–64.

[Gottschalk *et al.*, 2009] Gottschalk, F., Wagemakers, T. A., Jansen-Vullers, M. H., van der Aalst, W. M. et La Rosa, M. (2009). Configurable process models : Experiences from a municipality case study. *In International Conference on Advanced Information Systems Engineering*, pages 486–500. Springer.

[Govardhan et Feuerlicht, 2007] Govardhan, S. et Feuerlicht, G. (2007). Itinerary planner : A mashup case study. *In International Conference on Service-Oriented Computing*, pages 3–14. Springer.

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220.

[Gutierrez, 2019] Gutierrez, F. (2019). Introduction to spring boot. *In Pro Spring Boot 2*, pages 31–44. Springer.

[Hallerbach *et al.*, 2010] Hallerbach, A., Bauer, T. et Reichert, M. (2010). Capturing variability in business process models : the provop approach. *Journal of Software : Evolution and Process*, 22(6-7):519–546.

[Hansen et Golbeck, 2009] Hansen, D. L. et Golbeck, J. (2009). Mixing it up : recommending collections of items. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1217–1226. ACM.

[He *et al.*, 2016a] He, C., Parra, D. et Verbert, K. (2016a). Interactive recommender systems : A survey of the state of the art and future research challenges and opportunities. *Expert Systems with Applications*, 56:9–27.

[He *et al.*, 2017] He, J., Li, X. et Liao, L. (2017). Category-aware next point-of-interest recommendation via listwise bayesian personalized ranking. *In IJCAI*, pages 1837–1843.

[He *et al.*, 2016b] He, J., Li, X., Liao, L., Song, D. et Cheung, W. K. (2016b). Inferring a personalized next point-of-interest recommendation model with latent behavior patterns. *In Thirtieth AAAI Conference on Artificial Intelligence*.

[Herzog *et al.*, 2018] Herzog, D., Lass, C. et Wörndl, W. (2018). Tourrec : a tourist trip recommender system for individuals and groups. *In Proceedings of the 12th ACM Conference on Recommender Systems*, pages 496–497.

[Hirmer et Behringer, 2016] Hirmer, P. et Behringer, M. (2016). Flexmash 2.0–flexible modeling and execution of data mashups. *In International Rapid Mashup Challenge*, pages 10–29. Springer.

[Hirmer et Mitschang, 2015] Hirmer, P. et Mitschang, B. (2015). Flexmash–flexible data mashups based on pattern-based model transformation. *In International Rapid Mashup Challenge*, pages 12–30. Springer.

[Husmann *et al.*, 2014] Husmann, M., Nebeling, M., Pongelli, S. et Norrie, M. C. (2014). Multimasher : providing architectural support and visual tools for multi-device mashups. *In International Conference on Web Information Systems Engineering*, pages 199–214. Springer.

[Hvam, 2006] Hvam, L. (2006). Mass customisation in the electronics industry : based on modular products and product configuration. *International Journal of Mass Customisation*, 1(4):410–426.

[Imed et Graiet, 2017] Imed, A. et Graiet, M. (2017). An automatic configuration algorithm for reliable and efficient composite services. *IEEE Transactions on Network and Service Management*, 15(1):416–429.

[Imran *et al.*, 2012] Imran, M., Soi, S., Kling, F., Daniel, F., Casati, F. et Marchese, M. (2012). On the systematic development of domain-specific mashup tools for end users. *In International Conference on Web Engineering*, pages 291–298. Springer.

[Jaffe *et al.*, 2006] Jaffe, A., Naaman, M., Tassa, T. et Davis, M. (2006). Generating summaries and visualization for large collections of geo-referenced photographs. *In Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, pages 89–98. ACM.

[Jafri *et al.*, 2013] Jafri, R., Alkhunji, A. S., Alhader, G. K., Alrabeiah, H. R., Alhammad, N. A. et Alzahrani, S. K. (2013). Smart travel planner : A mashup of travel-related web services. *In 2013 International Conference on Current Trends in Information Technology (CTIT)*, pages 181–185. IEEE.

[Jang *et al.*, 2005] Jang, S., Ko, E.-J. et Woo, W. (2005). Unified user-centric context : Who, where, when, what, how and why. *In ubiPCMM*.

[Jarrar et Dikaiakos, 2008] Jarrar, M. et Dikaiakos, M. D. (2008). Mashql : a query-by-diagram topping sparql. *In Proceedings of the 2nd international workshop on Ontologies and information systems for the semantic web*, pages 89–96.

[Jiang *et al.*, 2016] Jiang, S., Qian, X., Mei, T. et Fu, Y. (2016). Personalized travel sequence recommendation on multi-source big social media. *IEEE Transactions on Big Data*, 2(1):43–56.

[Jin *et al.*, 2016] Jin, Y., Seipp, K., Duval, E. et Verbert, K. (2016). Go with the flow : effects of transparency and user control on targeted advertising using flow charts. *In Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 68–75. ACM.

[Jones et Churchill, 2009] Jones, M. C. et Churchill, E. F. (2009). Conversations in developer communities : a preliminary analysis of the yahoo ! pipes community. *In Proceedings of the fourth international conference on Communities and technologies*, pages 195–204. ACM.

[Jordan *et al.*, 2007] Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y. *et al.* (2007). Web services business process execution language version 2.0. *OASIS standard*, 11(120):5.

[Kanellopoulos *et al.*, 2010] Kanellopoulos, D. N., Hadaya, P., Pellerin, R., Gao, F., Liu, X., Guo, J., Chen, J.-S. et Li, E. Y. (2010). 457 mashups for travellers : integrating web applications based on the purchase stages of travel products. *Int. J. Electronic Business*, 8(6).

[Kongdenfha *et al.*, 2008] Kongdenfha, W., Benatallah, B., Saint-Paul, R. et Casati, F. (2008). Spreadmash : A spreadsheet-based interactive browsing and analysis tool for data services. *In International Conference on Advanced Information Systems Engineering*, pages 343–358. Springer.

[Kunaver et Požrl, 2017] Kunaver, M. et Požrl, T. (2017). Diversity in recommender systems–a survey. *Knowledge-Based Systems*, 123:154–162.

[Kurata et Hara, 2013] Kurata, Y. et Hara, T. (2013). Ct-planner4 : Toward a more user-friendly interactive day-tour planner. *In Information and communication technologies in tourism 2014*, pages 73–86. Springer.

[Laß *et al.*, 2017] Lass, C., Herzog, D. et Wörndl, W. (2017). Context-aware tourist trip recommendations. *In Proceedings of the 2nd Workshop on Recommenders in Tourism co-located with 11th ACM Conference on Recommender Systems (RecSys 2017), Como, Italy, August 27*.

[Laugwitz *et al.*, 2008] Laugwitz, B., Held, T. et Schrepp, M. (2008). Construction and evaluation of a user experience questionnaire. *In Symposium of the Austrian HCI and Usability Engineering Group*, pages 63–76. Springer.

[Lédeczi *et al.*, 2001] Lédeczi, Á., Bakay, A., Maroti, M., Volgyesi, P., Nordstrom, G., Sprinkle, J. et Karsai, G. (2001). Composing domain-specific design environments. *Computer*, 34(11):44–51.

[Ledent *et al.*, 2017] Ledent, V. *et al.* (2017). Comment transformer l'infobésité en intelligence stratégique pour les entreprises et les organisations ?

[Lee et Joo, 2013] Lee, E. et Joo, H.-J. (2013). Developing lightweight context-aware service mashup applications. *In 2013 15th International Conference on Advanced Communications Technology (ICACT)*, pages 1060–1064. IEEE.

[Lefrançois *et al.*, 2017] Lefrançois, M., Zimmermann, A. et Bakerally, N. (2017). A sparql extension for generating rdf from heterogeneous formats. *In European Semantic Web Conference*, pages 35–50. Springer.

[Lemos *et al.*, 2016] Lemos, A. L., Daniel, F. et Benatallah, B. (2016). Web service composition : a survey of techniques and tools. *ACM Computing Surveys (CSUR)*, 48(3):33.

[Leroy *et al.*, 2015] Leroy, V., Amer-Yahia, S., Gaussier, E. et Mirisaee, H. (2015). Building representative composite items. *In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1421–1430. ACM.

[Lewis, 1995] Lewis, J. R. (1995). Ibm computer usability satisfaction questionnaires : psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 7(1):57–78.

[Lewis, 2002] Lewis, J. R. (2002). Psychometric evaluation of the pssuq using data from five years of usability studies. *International Journal of Human-Computer Interaction*, 14(3-4):463–488.

[Lewis, 2014] Lewis, J. R. (2014). Usability : lessons learned... and yet to be learned. *International Journal of Human-Computer Interaction*, 30(9):663–684.

[Lewis, 2018] Lewis, J. R. (2018). Measuring perceived usability : The csuq, sus, and umux. *International Journal of Human–Computer Interaction*, 34(12):1148–1156.

[Lewis et Sauro, 2009] Lewis, J. R. et Sauro, J. (2009). The factor structure of the system usability scale. *In International conference on human centered design*, pages 94–103. Springer.

[Li *et al.*, 2015a] Li, X., Cong, G., Li, X.-L., Pham, T.-A. N. et Krishnaswamy, S. (2015a). Rank-geofm : A ranking based geographical factorization method for point of interest recommendation. *In Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 433–442. ACM.

[Li *et al.*, 2015b] Li, X., Eckert, M., Martinez, J.-F. et Rubio, G. (2015b). Context aware middleware architectures : survey and challenges. *Sensors*, 15(8):20570–20607.

[Lim, 2015] Lim, K. H. (2015). Recommending tours and places-of-interest based on user interests from geo-tagged photos. *In Proceedings of the 2015 ACM SIGMOD on PhD Symposium*, pages 33–38. ACM.

[Lim *et al.*, 2015] Lim, K. H., Chan, J., Leckie, C. et Karunasekera, S. (2015). Personalized tour recommendation based on user interests and points of interest visit durations. *In Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[Lim *et al.*, 2016] Lim, K. H., Wang, X., Chan, J., Karunasekera, S., Leckie, C., Chen, Y., Tan, C. L., Gao, F. Q. et Wee, T. K. (2016). Perstour : A personalized tour recommendation and planning system. *In HT (Extended Proceedings)*.

[Liu *et al.*, 2012] Liu, Q., Chen, E., Xiong, H., Ge, Y., Li, Z. et Wu, X. (2012). A cocktail approach for travel package recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):278–293.

[Lu *et al.*, 2016] Lu, C., Laublet, P. et Stankovic, M. (2016). Travel attractions recommendation with knowledge graphs. *In European Knowledge Acquisition Workshop*, pages 416–431. Springer.

[Machado *et al.*, 2017] Machado, A., Maran, V., Augustin, I., Wives, L. K. et de Oliveira, J. P. M. (2017). Reactive, proactive, and extensible situation-awareness in ambient assisted living. *Expert Systems with Applications*, 76:21–35.

[Markl *et al.*, 2008] Markl, V., Altinel, M., Simmen, D. et Singh, A. (2008). Data mashups for situational applications. *In International Workshop on Model-Based Software and Data Integration*, pages 12–18. Springer.

[Matera *et al.*, 2013] Matera, M., Picozzi, M., Pini, M. et Tonazzo, M. (2013). Peudom : a mashup platform for the end user development of common information spaces. *In International Conference on Web Engineering*, pages 494–497. Springer.

[Maximilien *et al.*, 2007] Maximilien, E. M., Wilkinson, H., Desai, N. et Tai, S. (2007). A domain-specific language for web apis and services mashups. *In International Conference on Service-Oriented Computing*, pages 13–26. Springer.

[Mayer *et al.*, 2016] Mayer, S., Verborgh, R., Kovatsch, M. et Mattern, F. (2016). Smart configuration of smart environments. *IEEE Transactions on Automation Science and Engineering*, 13(3):1247–1255.

[McIlraith et Son, 2002] McIlraith, S. et Son, T. C. (2002). Adapting golog for composition of semantic web services. *KR*, 2:482–493.

[Mendes *et al.*, 2011] Mendes, P. N., Jakob, M., García-Silva, A. et Bizer, C. (2011). Dbpedia spotlight : shedding light on the web of documents. *In Proceedings of the 7th international conference on semantic systems*, pages 1–8.

[Mendes *et al.*, 2012] Mendes, P. N., Mühleisen, H. et Bizer, C. (2012). Sieve : linked data quality assessment and fusion. *In Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 116–123.

[Meng et Xu, 2010] Meng, J. et Xu, N. (2010). A mobile tourist guide system based on mashup technology. *In The 2nd International Conference on Information Science and Engineering*, pages 1716–1719. IEEE.

[Mernik *et al.*, 2005] Mernik, M., Heering, J. et Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, 37(4):316–344.

[Mesmoudi *et al.*, 2011] Mesmoudi, A., Mrissa, M. et Hacid, M.-S. (2011). Combining configuration and query rewriting for web service composition. *In 2011 IEEE International Conference on Web Services*, pages 113–120. IEEE.

[Middleton *et al.*, 2004] Middleton, S. E., Shadbolt, N. R. et De Roure, D. C. (2004). Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88.

[Muhammad *et al.*, 2012] Muhammad, I., Florian, D., Fabio, C. et Maurizio, M. (2012). Reseval mash : a mashup tool that speaks the language of the user. *In CHI'12 Extended Abstracts on Human Factors in Computing Systems*, pages 1949–1954. ACM.

[Namoun *et al.*, 2010] Namoun, A., Nestler, T. et De Angeli, A. (2010). Conceptual and usability issues in the composable web of software services. *In International Conference on Web Engineering*, pages 396–407. Springer.

[Nurbakova *et al.*, 2016] Nurbakova, D., Laporte, L., Calabretto, S. et Gensel, J. (2016). Anastasia : recommandation de séquences d'activités spatio-temporelles.

[Nurbakova *et al.*, 2017] Nurbakova, D., Laporte, L., Calabretto, S. et Gensel, J. (2017). Recommendation of short-term activity sequences during distributed events. *Procedia Computer Science*, 108:2069–2078.

[O'reilly, 2009] O'reilly, T. (2009). *What is web 2.0.* " O'Reilly Media, Inc.".

[Paik *et al.*, 2017] Paik, H.-y., Lemos, A. L., Barukh, M. C., Benatallah, B. et Natarajan, A. (2017). *Web Service Implementation and Composition Techniques*, volume 256. Springer.

[Papazoglou, 2008] Papazoglou, M. (2008). *Web services : principles and technology.* Pearson Education.

[Parameswaran *et al.*, 2011] Parameswaran, A., Venetis, P. et Garcia-Molina, H. (2011). Recommendation systems with complex constraints : A course recommendation perspective. *ACM Transactions on Information Systems (TOIS)*, 29(4):20.

[Paschke *et al.*, 2012] Paschke, A., Boley, H., Zhao, Z., Teymourian, K. et Athan, T. (2012). Reaction ruleml 1.0 : standardized semantic reaction rules. *In International Workshop on Rules and Rule Markup Languages for the Semantic Web*, pages 100–119. Springer.

[Paternò et Santoro, 2017] Paternò, F. et Santoro, C. (2017). A design space for end user development in the time of the internet of things. *In New perspectives in end-user development*, pages 43–59. Springer.

[Pautasso, 2009] Pautasso, C. (2009). Restful web service composition with bpel for rest. *Data & Knowledge Engineering*, 68(9):851–866.

[Perera *et al.*, 2013] Perera, C., Zaslavsky, A., Christen, P. et Georgakopoulos, D. (2013). Context aware computing for the internet of things : A survey. *IEEE communications surveys & tutorials*, 16(1):414–454.

[Preuveneers et Novais, 2012] Preuveneers, D. et Novais, P. (2012). A survey of software engineering best practices for the development of smart applications in ambient intelligence. *Journal of Ambient Intelligence and Smart Environments*, 4(3):149–162.

[Pu *et al.*, 2011] Pu, P., Faltings, B., Chen, L., Zhang, J. et Viappiani, P. (2011). Usability guidelines for product recommenders based on example critiquing research. *In Recommender systems handbook*, pages 511–545. Springer.

[Quadrana *et al.*, 2018] Quadrana, M., Cremonesi, P. et Jannach, D. (2018). Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)*, 51(4):66.

[Radeck *et al.*, 2013] Radeck, C., Blichmann, G. et Meissner, K. (2013). Capview–functionality-aware visual mashup development for non-programmers. *In International Conference on Web Engineering*, pages 140–155. Springer.

[Ricci *et al.*, 2011] Ricci, F., Rokach, L. et Shapira, B. (2011). Introduction to recommender systems handbook. *In Recommender systems handbook*, pages 1–35. Springer.

[Richardson *et al.*, 2013] Richardson, L., Amundsen, M., Amundsen, M. et Ruby, S. (2013). *RESTful Web APIs : Services for a Changing World.* " O'Reilly Media, Inc.".

[Rodríguez *et al.*, 2012] Rodríguez, B., Molina, J., Pérez, F. et Caballero, R. (2012). Interactive design of personalised tourism routes. *Tourism Management*, 33(4):926–940.

[Rosa *et al.*, 2017] Rosa, M. L., Van Der Aalst, W. M., Dumas, M. et Milani, F. P. (2017). Business process variability modeling : A survey. *ACM Computing Surveys (CSUR)*, 50(1):2.

[Roy *et al.*, 2011] Roy, S. B., Das, G., Amer-Yahia, S. et Yu, C. (2011). Interactive itinerary planning. *In 2011 IEEE 27th International Conference on Data Engineering*, pages 15–26. IEEE.

[Roy Chowdhury *et al.*, 2013] Roy Chowdhury, S., Chudnovskyy, O., Niederhausen, M., Pietschmann, S., Sharples, P., Daniel, F. et Gaedke, M. (2013). Complementary assistance mechanisms for end user mashup composition. *In Proceedings of the 22nd International Conference on World Wide Web*, pages 269–272. ACM.

[Roy Chowdhury *et al.*, 2014] Roy Chowdhury, S., Daniel, F. et Casati, F. (2014). Recommendation and weaving of reusable mashup model patterns for assisted development. *ACM Transactions on Internet Technology (TOIT)*, 14(2-3):21.

[Roy Chowdhury *et al.*, 2012] Roy Chowdhury, S., Rodríguez, C., Daniel, F. et Casati, F. (2012). Baya : assisted mashup development as a service. *In Proceedings of the 21st International Conference on World Wide Web*, pages 409–412. ACM.

[Sabatucci *et al.*, 2016] Sabatucci, L., Lopes, S. et Cossentino, M. (2016). A goal-oriented approach for self-configuring mashup of cloud applications. *In 2016 International Conference on Cloud and Autonomic Computing (ICCAC)*, pages 84–94. IEEE.

[Sadou *et al.*, 2005] Sadou, N., Tamzalit, D. et Oussalah, M. (2005). How to manage uniformly software architecture at different abstraction levels. *In International Conference on Conceptual Modeling*, pages 16–30. Springer.

[Samanta et Liu, 2017] Samanta, P. et Liu, X. (2017). Recommending services for new mashups through service factors and top-k neighbors. *In 2017 IEEE International Conference on Web Services (ICWS)*, pages 381–388. IEEE.

[Sang *et al.*, 2015] Sang, J., Mei, T. et Xu, C. (2015). Activity sensor : Check-in usage mining for local recommendation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):41.

[Sansonetti, 2019] Sansonetti, G. (2019). Point of interest recommendation based on social and linked open data. *Personal and Ubiquitous Computing*, 23(2):199–214.

[Sansonetti *et al.*, 2019] Sansonetti, G., Gasparetti, F., Micarelli, A., Cena, F. et Gena, C. (2019). Enhancing cultural recommendations through social and linked open data. *User Modeling and User-Adapted Interaction*, 29(1):121–159.

[Sassi *et al.*, 2017] Sassi, I. B., Mellouli, S. et Yahia, S. B. (2017). Context-aware recommender systems in mobile environment : On the road of future research. *Information Systems*, 72:27–61.

[Schaller *et al.*, 2014] Schaller, R., Harvey, M. et Elsweiler, D. (2014). Relating user interaction to experience during festivals. *In Proceedings of the 5th Information Interaction in Context Symposium*, pages 38–47. ACM.

[Schrepp *et al.*, 2017] Schrepp, M., Hinderks, A. et Thomaschewski, J. (2017). Construction of a benchmark for the user experience questionnaire (ueq). *IJIMAI*, 4(4):40–44.

[Schultz *et al.*, 2012] Schultz, A., Matteini, A., Isele, R., Mendes, P. N., Bizer, C. et Becker, C. (2012). Ldif-a framework for large-scale linked data integration. *In 21st International World Wide Web Conference (WWW 2012), Developers Track, Lyon, France.*

[Sekkal *et al.*, 2018] Sekkal, N., Benslimane, S. M., Mrissa, M. et Boudaa, B. (2018). Combining proactive and reactive approaches in smart services for the web of things. *In IFIP International Conference on Computational Intelligence and Its Applications*, pages 509–520. Springer.

[Shah *et al.*, 2016] Shah, L., Gaudani, H. et Balani, P. (2016). Survey on recommendation system. *International Journal of Computer Applications*, 137(7).

[Sheng *et al.*, 2009] Sheng, Q. Z., Benatallah, B., Maamar, Z. et Ngu, A. H. (2009). Configurable composition and adaptive provisioning of web services. *IEEE Transactions on Services Computing*, 2(1):34–49.

[Shukla *et al.*, 2017] Shukla, Y. *et al.* (2017). State of art survey of travel based recommendation system. *International Journal of Advanced Research in Computer Science*, 8(3).

[Silva *et al.*, 2018] Silva, C. A., Toasa, R., Guevara, J., Martinez, H. D. et Vargas, J. (2018). Mobile application to encourage local tourism with context-aware computing. *In International Conference on Information Theoretic Security*, pages 796–803. Springer.

[Singh *et al.*, 2017] Singh, M., Borromeo, R. M., Hosami, A., Amer-Yahia, S. et Elbassuoni, S. (2017). Customizing travel packages with interactive composite items. *In 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 137–145. IEEE.

[Soi *et al.*,] Soi, S., Daniel, F. et Casati, F. Domain-specific mashup platforms for end users : A conceptual development approach. *TWEB (2014, submitted).*

[Soi *et al.*, 2014] Soi, S., Daniel, F. et Casati, F. (2014). Conceptual development of custom, domain-specific mashup platforms. *ACM Transactions on the Web (TWEB)*, 8(3):14.

[Soualah-Alila *et al.*, 2016] Soualah-Alila, F., Coustaty, M., Rempulski, N. et Doucet, A. (2016). Datatourism : designing an architecture to process tourism data. *In Information and communication technologies in tourism 2016*, pages 751–763. Springer.

[Sutcliffe *et al.*, 2003] Sutcliffe, A., Lee, D. et Mehandjiev, N. (2003). Contributions, costs and prospects for end-user development. *In Proceedings of the Tenth International Conference on Human-Computer Interaction*, volume 3.

[Tanenbaum et Van Steen, 2007] Tanenbaum, A. S. et Van Steen, M. (2007). *Distributed systems : principles and paradigms.* Prentice-Hall.

[Taylor *et al.*, 2018] Taylor, K., Lim, K. H. et Chan, J. (2018). Travel itinerary recommendations with must-see points-of-interest. *In Companion Proceedings of the The Web Conference 2018*, pages 1198–1205. International World Wide Web Conferences Steering Committee.

[Teije *et al.*, 2004] Teije, A. t., Harmelen, F. v. et Wielinga, B. (2004). Configuration of web services as parametric design. *In Proceedings of the 16th European Conference on Artificial Intelligence*, pages 1097–1098. IOS Press.

[Ten Teije *et al.*, 2004] Ten Teije, A., Van Harmelen, F. et Wielinga, B. (2004). Configuration of web services as parametric design. *In International Conference on Knowledge Engineering and Knowledge Management*, pages 321–336. Springer.

[Thorat *et al.*, 2015] Thorat, P. B., Goudar, R. et Barve, S. (2015). Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4):31–36.

[Trattner *et al.*, 2016] Trattner, C., Oberegger, A., Eberhard, L., Parra, D., Marinho, L. B. *et al.* (2016). Understanding the impact of weather for poi recommendations. *In RecTour@ RecSys*, pages 16–23.

[Triki, 2016] Triki, R. B. C. (2016). *RECODYN : Une approche d'ingénierie d'application basée sur la recommandation et la configuration interactive des lignes de produits.* Thèse de doctorat.

[Trinh *et al.*, 2016] Trinh, T.-D., Wetz, P., Do, B.-L., Kiesling, E. et Tjoa, A. M. (2016). Linked widgets platform for rapid collaborative semantic mashup development. *In International Rapid Mashup Challenge*, pages 51–73. Springer.

[Troncy *et al.*, 2017] Troncy, R., Rizzo, G., Jameson, A., Corcho, O., Plu, J., Palumbo, E., Hermida, J. C. B., Spirescu, A., Kuhn, K.-D., Barbu, C. *et al.* (2017). 3cixty : Building comprehensive knowledge bases for city exploration. *Journal of Web Semantics*, 46:2–13.

[Ur *et al.*, 2014] Ur, B., McManus, E., Pak Yong Ho, M. et Littman, M. L. (2014). Practical trigger-action programming in the smart home. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 803–812. ACM.

[Ur *et al.*, 2016] Ur, B., Pak Yong Ho, M., Brawner, S., Lee, J., Mennicken, S., Picard, N., Schulze, D. et Littman, M. L. (2016). Trigger-action programming in the wild : An analysis of 200,000 ifttt recipes. *In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 3227–3231. ACM.

[Urbieta *et al.*, 2017] Urbieta, A., González-Beltrán, A., Mokhtar, S. B., Hossain, M. A. et Capra, L. (2017). Adaptive and context-aware service composition for iot-based smart cities. *Future Generation Computer Systems*, 76:262–274.

[Vansteenwegen *et al.*, 2011] Vansteenwegen, P., Souffriau, W. et Van Oudheusden, D. (2011). The orienteering problem : A survey. *European Journal of Operational Research*, 209(1):1–10.

[Viktoratos *et al.*, 2014a] Viktoratos, I., Tsadiras, A. et Bassiliades, N. (2014a). Providing a context-aware location based web service through semantics and user-defined rules. *In Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*, page 9. ACM.

[Viktoratos *et al.*, 2015] Viktoratos, I., Tsadiras, A. et Bassiliades, N. (2015). A context-aware web-mapping system for group-targeted offers using semantic technologies. *Expert Systems with Applications*, 42(9):4443–4459.

[Viktoratos *et al.*, 2017] Viktoratos, I., Tsadiras, A. et Bassiliades, N. (2017). Modeling human daily preferences through a context-aware web-mapping system using semantic technologies. *Pervasive and Mobile Computing*, 38:14–40.

[Viktoratos *et al.*, 2014b] Viktoratos, I., Tsadiras, A. K. et Bassiliades, N. (2014b). Geosocial splis : A rule-based service for context-aware point of interest exploration. *In Challenge+ DC@ RuleML*.

[Vulbeau, 2015] Vulbeau, A. (2015). Contrepoint-l'infobésité et les risques de la surinformation. *Informations sociales*, (5):35–35.

[Wajid *et al.*, 2011] Wajid, U., Namoun, A. et Mehandjiev, N. (2011). Alternative representations for end user composition of service-based systems. *In International Symposium on End User Development*, pages 53–66. Springer.

[Wang *et al.*, 2009] Wang, G., Yang, S. et Han, Y. (2009). Mashroom : end-user mashup programming using nested tables. *In Proceedings of the 18th international conference on World wide web*, pages 861–870.

[Wang *et al.*, 2017] Wang, L., Zhong, S.-S. et Zhang, Y.-J. (2017). Process configuration based on generative constraint satisfaction problem. *Journal of Intelligent Manufacturing*, 28(4):945–957.

[Weber *et al.*, 2013] Weber, I., Paik, H.-Y. et Benatallah, B. (2013). Form-based web service composition for domain experts. *ACM Transactions on the Web (TWEB)*, 8(1):2.

[Weiss *et al.*, 2013] Weiss, M., Sari, S. et Noori, N. (2013). Growth of the mashup ecosystem and niche formation.

[Wong et Hong, 2007] Wong, J. et Hong, J. I. (2007). Making mashups with marmite : towards end-user programming for the web. *In Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1435–1444. ACM.

[Xie *et al.*, 2011] Xie, M., Lakshmanan, L. V. et Wood, P. T. (2011). Comprec-trip : A composite recommendation system for travel planning. *In 2011 IEEE 27th International Conference on Data Engineering*, pages 1352–1355. IEEE.

[Xie *et al.*, 2012] Xie, M., Lakshmanan, L. V. et Wood, P. T. (2012). Composite recommendations : from items to packages. *Frontiers of computer science*, 6(3):264–277.

[Yahi *et al.*, 2015] Yahi, A., Chassang, A., Raynaud, L., Duthil, H. et Chau, D. H. P. (2015). Aurigo : an interactive tour planner for personalized itineraries. *In Proceedings of the 20th international conference on intelligent user interfaces*, pages 275–285. ACM.

[Yang et Dong, 2013] Yang, D. et Dong, M. (2013). Applying constraint satisfaction approach to solve product configuration problems with cardinality-based configuration rules. *Journal of Intelligent Manufacturing*, 24(1):99–111.

[Yang *et al.*, 2012] Yang, D., Dong, M. et Chang, X.-K. (2012). A dynamic constraint satisfaction approach for configuring structural products under mass customization. *Engineering Applications of Artificial Intelligence*, 25(8):1723–1737.

[Yang et Fang, 2014] Yang, P. et Fang, H. (2014). Exploration of opinion-aware approach to contextual suggestion. Rapport technique, DELAWARE UNIV NEWARK DEPT OF ELECTRICAL AND COMPUTER ENGINEERING.

[Yao *et al.*, 2018] Yao, L., Wang, X., Sheng, Q. Z., Benatallah, B. et Huang, C. (2018). Mashup recommendation by regularizing matrix factorization with api co-invocations. *IEEE Transactions on Services Computing*.

[Yee, 2008] Yee, R. (2008). *Pro Web 2.0 mashups : remixing data and web services*. Apress.

[Yu *et al.*, 2008] Yu, J., Benatallah, B., Casati, F. et Daniel, F. (2008). Understanding mashup development. *IEEE Internet computing*, 12(5):44–52.

[Yu et Chen, 2015] Yu, Y. et Chen, X. (2015). A survey of point-of-interest recommendation in location-based social networks. *In Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[Yu *et al.*, 2016] Yu, Z., Xu, H., Yang, Z. et Guo, B. (2016). Personalized travel package with multi-point-of-interest recommendation based on crowdsourced user footprints. *IEEE Transactions on Human-Machine Systems*, 46(1):151–158.

[Yuan *et al.*, 2013] Yuan, Q., Cong, G., Ma, Z., Sun, A. et Thalmann, N. M. (2013). Time-aware point-of-interest recommendation. *In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 363–372. ACM.

[Zhang *et al.*, 2016] Zhang, H., Yang, Y. et Zhang, Z. (2016). Cts : combine temporal influence and spatial influence for time-aware poi recommendation. *In International Conference of Pioneering Computer Scientists, Engineers and Educators*, pages 272–286. Springer.

[Zhang et Chow, 2015a] Zhang, J.-D. et Chow, C.-Y. (2015a). Geosoca : Exploiting geographical, social and categorical correlations for point-of-interest recommendations. *In Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 443–452.

[Zhang et Chow, 2015b] Zhang, J.-D. et Chow, C.-Y. (2015b). Spatiotemporal sequential influence modeling for location recommendations : A gravity-based approach. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(1):11.

[Zhao *et al.*, 2016a] Zhao, S., King, I. et Lyu, M. R. (2016a). A survey of point-of-interest recommendation in location-based social networks. *arXiv preprint arXiv :1607.00647*.

[Zhao *et al.*, 2016b] Zhao, S., Zhao, T., Yang, H., Lyu, M. R. et King, I. (2016b). Stellar : spatial-temporal latent ranking for successive point-of-interest recommendation. *In Thirtieth AAAI conference on artificial intelligence*.

[ZHOU, 2017] ZHOU, M. (2017). A hybrid approach for automatic mashup tag recommendation. *Journal of Web Engineering*, 16(7&8):676–692.

[Ziegler *et al.*, 2005] Ziegler, C.-N., McNee, S. M., Konstan, J. A. et Lausen, G. (2005). Improving recommendation lists through topic diversification. *In Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM.