Conception d'applications web géographiques guidée par les contenus et les usages : cadre méthodologique et opérationnalisation avec l'environnement WINDMash

Patrick Etcheverry, Sébastien Laborie, Christophe Marquesuzaà, Thierry Nodenot, The Nhân Luong

Conception d'applications web géographiques guidée par les contenus et les usages : cadre méthodologique et opérationnalisation avec l'environnement WINDMash

PATRICK ETCHEVERRY, SEBASTIEN LABORIE, CHRISTOPHE MARQUESUZAÀ, THIERRY NODENOT, THE NHAN LUONG
UNIV PAU & PAYS ADOUR, LABORATOIRE D'INFORMATIQUE DE L'UNIVERSITE DE PAU ET DES PAYS DE L'ADOUR, EA3000, 64600, ANGLET, France

Résumé : Cet article présente un cadre de conception d'applications Web géographiques interactives s'appuyant sur des modèles de conception génériques permettant d'élaborer une application selon trois dimensions : les contenus géographiques manipulés, la manière de les afficher mais aussi les comportements interactifs associés. Les modèles de conception proposés permettent de produire ces applications au service de tâches élémentaires nécessaires à la réalisation d'un objectif. Ces modèles se veulent riches pour être en mesure de décrire une grande variété d'applications tout en conservant la capacité à traduire leurs instances sous forme de code exécutable. L'exécution des modèles permet ainsi de proposer une approche de conception basée sur des cycles courts dans lesquels le concepteur affine ses besoins en enchaînant, autant de fois que nécessaire, des phases de spécification, d'exécution et d'évaluation de son application. Pour faciliter le processus de conception, le travail de spécification est réalisé de manière visuelle y compris la dimension interactive qui demeure la plus complexe à décrire et pour laquelle nous proposons un langage dédié inspiré du diagramme de séquence UML. Ce langage visuel se veut simple afin de pouvoir être appréhendé par des experts d'un domaine (enseignement, tourisme, culture…) n'ayant pas forcément des compétences fortes en informatique. Ce langage se veut également riche dans la description de l'interactivité. Pour mesurer l'apport de cette approche mais aussi des modèles et des langages de conception proposés, nous avons mis au point un environnement-auteur nommé WINDMash. Ce démonstrateur vise à mesurer l'expressivité des modèles définis en les expérimentant sur la conception d'applications géographiques variées. Il vise également à évaluer la pertinence de l'approche de conception et des outils de spécification visuels proposés.

Mots clés : Conception centrée utilisateur, Conception guidée par les contenus et l'interaction, Langage visuel pour l'interaction, Modélisation de tâches par essai-erreur via des cycles courts, Valorisation de données géographiques, Génération d'applications web, Environnement-auteur, Mashup.

Adresse des auteurs : Patrick Etcheverry (patrick.etcheverry@iutbayonne.univ-pau.fr), Sébastien Laborie (sebastien.laborie@iutbayonne.univ-pau.fr), Christophe Marquesuzaà (christophe.marquesuzaa@iutbayonne.univ-pau.fr), Thierry Nodenot (thierry.nodenot@iutbayonne.univ-pau.fr), The Nhân Luong (luongthenhan@gmail.com), IUT de Bayonne et du Pays Basque, 2, Allée du Parc Montaury 64600 Anglet, France.

Abstract: This paper presents a framework dedicated to the design of geographic web applications. This framework includes three generic design models that allow designers to build a geographic application according to three dimensions: geographic content, content rendering and display, and interactive behaviour. The three design models allow designers to develop applications for elementary tasks to achieve a specific goal. They are expressive enough to describe a wide variety of geographic applications. They are operational as they are translated into executable code. This executability supports an agile design process based on short cycles where designers can refine their needs as many times as necessary by specifying, executing and evaluating their application. Modeling is performed using visual languages. In particular, the interactive behaviour is specified with a language inspired from the UML sequence diagram. We have demonstrated that this language is sufficiently simple to be understood by domain experts (e.g., education, tourism, culture) with no computer science background. We have developed WINDMash, an authoring environment, to assess our approach, as well as the models and the design languages with a diversity of geographic Web applications.

Key words: User-Centered Design, Interaction Visual Language, Empirical Task Modeling with Short Lifecycle, Geographic Web Application Design, Web Application Generation, Graphical Authoring Environment, Mashup.

1. INTRODUCTION

Depuis plusieurs années, la disponibilité d'informations géographiques n'a cessé de croître avec la mise à disposition de données publiques sur le Web (Open data). Ainsi, les internautes ont accès à de grandes bases de connaissances décrivant divers territoires, comme Paris[1] ou Toulouse[2], et collectant des informations du monde entier (DBpedia[3]). Par ailleurs, les utilisateurs d'aujourd'hui peuvent également acquérir et produire ces données de façon transparente via leurs dispositifs mobiles (smartphones, tablettes...) ou bien en utilisant des outils communautaires, tel que Open Street Map[4].

Nous souhaitons dans cet article mettre l'accent sur la conception d'applications mettant en valeur ces données au travers d'interactions, en allant au-delà du simple moteur de recherche ou de l'affichage de données sur une carte. En effet, de multiples domaines requièrent ce type d'applications interactives : le tourisme ou la culture (encyclopédies multimédias par exemple), la sécurité (vidéo-surveillance...), l'éducation (découverte de territoires...), etc. L'étude de ces applications nous a permis d'identifier un certain nombre de spécificités en fonction du domaine métier mais également les invariants suivants : toutes ces applications intègrent des contenus géoréférencés, les présentent sur des cartes ou des frises chronologiques et offrent à l'utilisateur des possibilités d'interaction pour manipuler, rechercher, synthétiser ou calculer des données géographiques. Dans ce contexte, développer des applications géographiques devient une tâche de plus en plus difficile :

– il existe une multitude de modèles décrivant les informations géographiques.
– des services de traitement de l'information de plus en plus complets doivent pouvoir être combinés.
– de nombreux langages et bibliothèques de programmation doivent être exploités.

Compte tenu de la grande variabilité des applications, de la diversité des domaines métiers et de la complexité de mise en œuvre, de nombreux travaux se sont orientés vers des environnements facilitant la conception et la diffusion de ces applications. Ces environnements ne peuvent être directement exploités par un concepteur sans connaissances techniques approfondies car ils nécessitent très souvent des compétences en développement ainsi qu'une bonne maîtrise des technologies dédiées.

Nos travaux ciblent particulièrement des concepteurs qui doivent être dotés de dispositifs leur permettant de spécifier, concevoir et déployer, par prototypage rapide, des applications géographiques. Les concepteurs que nous visons ont les spécificités suivantes :

– Ce sont des experts de leur domaine : des enseignants, des géographes, des personnes travaillant dans un office de tourisme, etc. Ils connaissent les spécificités de leur application et leurs choix de conception sont guidés par des critères issus de leur domaine d'expertise métier (critères pédagogiques, touristiques...) et non pas du domaine informatique.
– Ce ne sont pas des experts en informatique : nous considérons qu'ils n'ont pas les capacités d'abstraction nécessaires pour mener une activité de conception basée sur des méthodes traditionnelles issues du génie logiciel leur permettant de spécifier leur application

[1] opendata.paris.fr
[2] data.grandtoulouse.fr
[3] dbpedia.org
[4] www.openstreetmap.org

selon différentes couches (modèles de données, traitements, tâches, dialogue, présentation...). Nous considérons également qu'ils ne peuvent être impliqués dans des activités de production ou de modification de code.

En effet, si ces utilisateurs ne sont jamais en situation d'exprimer formellement la tâche visée par une application à élaborer, nous faisons l'hypothèse qu'un dispositif adéquat peut leur permettre de construire et tester des spécifications fonctionnelles par expression de leurs intentions via des langages visuels adaptés. Nous nous situons ici dans une approche *End-User Programming* telle que définie dans [Ko et al. 2011].

Les approches reposant sur la modélisation des tâches répondent en partie à ces problématiques et nous avons souhaité aborder de manière originale le problème en permettant à l'utilisateur final d'exprimer lui-même, de manière empirique, selon un processus basé sur des essais-erreurs, la manière de réaliser une tâche pour atteindre un but donné ainsi que les informations (géographiques) qui sont nécessaires à la réalisation de cette tâche. Nous nous intéressons donc à un processus de conception dans lequel l'utilisateur final spécifie et construit lui-même l'application qui lui permettra de mener une tâche donnée. Les applications visées sont de taille "raisonnable" dans le sens où elles sont au service d'une et une seule tâche; la réalisation d'une tâche complexe pouvant être traitée par plusieurs applications, chacune au service d'une sous-tâche.

Dans le cadre du projet de recherche "*Pyrénées itinéraires éducatifs*"[5], nous avons participé au développement d'une application spécifiquement dédiée aux classes de découvertes. À cette fin, nous avons travaillé avec un professeur des écoles pour définir les spécifications d'une application de lecture active d'un texte ayant pour thématique le Tour de France cycliste 2012. Les différentes réunions ont permis de construire incrémentalement un cahier des charges. Dans la première version, ce cahier des charges comportait uniquement des informations qui traitaient des lieux parcourus cités dans le texte et de l'intérêt de les visualiser sur un fond cartographique (*cf.* zones 1 et 2 de la figure 1).

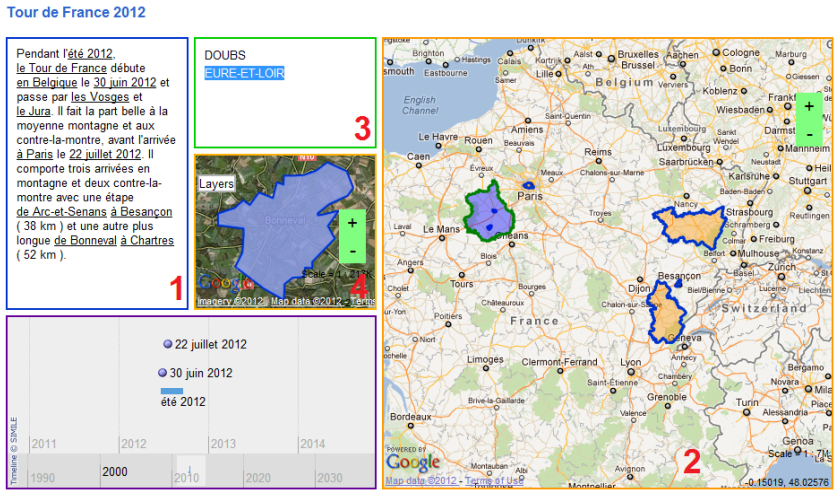

Fig. 1. Une application Web géographique sur le Tour de France 2012

[5] www.aberouat-itinerairesseducatifs.fr

Après la présentation de différents prototypes que nous avons codés, le professeur a affiné ses besoins tant au niveau du comportement souhaité que des contenus à afficher au sein de l'application. La dernière version contient des éléments mettant en exergue le département de chacune des villes du texte, plus précisément les noms (*cf.* zone 3) et les contours spécifiques (*cf.* zone 4) des départements, ainsi qu'une frise chronologique pour identifier les dates et périodes citées.

L'enseignant tient à ce que ce soit une activité de lecture active de textes à caractère spatio-temporel et non pas une simple application de consultation de cartes géographiques. Seules nous intéressent des interactions émanant du texte vers les cartes quels que soient les afficheurs textuels utilisés. Dans le cadre de cette activité, nous avons construit durant chaque étape de réflexion le modèle de tâches sous-jacent à l'application désirée. La dernière version de l'application validée par l'enseignant avant son utilisation en classe est présentée dans le modèle suivant (Figure 2) sous forme de CTT [Paternò et al. 1997].

Nous avons constaté que nous avions du mal à étayer nos discussions avec l'enseignant sur la base de tels diagrammes et que nous ne pouvions pas lui faire construire / corriger de tels modèles. Par contre, un tel modèle est fort utile à des informaticiens pour s'approprier le domaine métier et documenter un cahier des charges servant de base pour coder des applications.

L'objet de cet article est donc de présenter un cadre méthodologique et son opérationalisation avec un environnement-auteur pour que des concepteurs non-spécialistes en informatique puissent spécifier des interactions pour des applications métiers dédiées. Il ne s'agit pas d'amener un concepteur à décrire un modèle de tâches mais plutôt de lui permettre d'opérationaliser des tâches sans passer par une modélisation de type CTT. Nous verrons en conclusion de l'article que les représentations des modèles créés grâce à l'environnement-auteur permettent de générer à la volée le code applicatif et donc de tester l'application mais que ces mêmes modèles pourraient permettre de retrouver en partie des éléments du diagramme CTT tels que ceux illustrés dans la figure 2. Notre approche scientifique du problème est de proposer des méthodes et des techniques permettant :

– la spécification (semi-automatique) des données géographiques nécessaires à la réalisation de la tâche. Ces données pourront être spécifiées / extraites / calculées à partir de sources multiples (textes fixes ou à saisir, bases de données géographiques, Web...).
– l'expression / l'évaluation via des langages visuels et le prototypage rapide de la tâche que l'expert veut mener en s'appuyant sur ces contenus géographiques. Compte tenu de la diversité des domaines d'application (surveillance environnementale, tourisme, éducation...), notre approche considère différentes facettes de conception qu'il s'agit de tisser les unes aux autres. Notre étude du domaine nous a conduits à identifier trois facettes qui apparaissent systématiquement dans toute application géographique : la facette "Contenus (géographiques)" définissant les données à intégrer, la facette "Interface" précisant la manière d'afficher les données et la facette "Interaction" définissant les actions déclenchables sur ces données pour mener la tâche considérée.

Plus précisément, la démarche de conception que nous proposons (Figure 3) est articulée de la manière suivante. Le concepteur spécifie visuellement, dans l'ordre souhaité, les données géographiques jouant un rôle dans la réalisation de la tâche visée, la manière dont il souhaite les présenter sur son interface utilisateur mais aussi les comportements interactifs déclenchables sur chacun de ces contenus pour exécuter finalement la tâche ciblée.

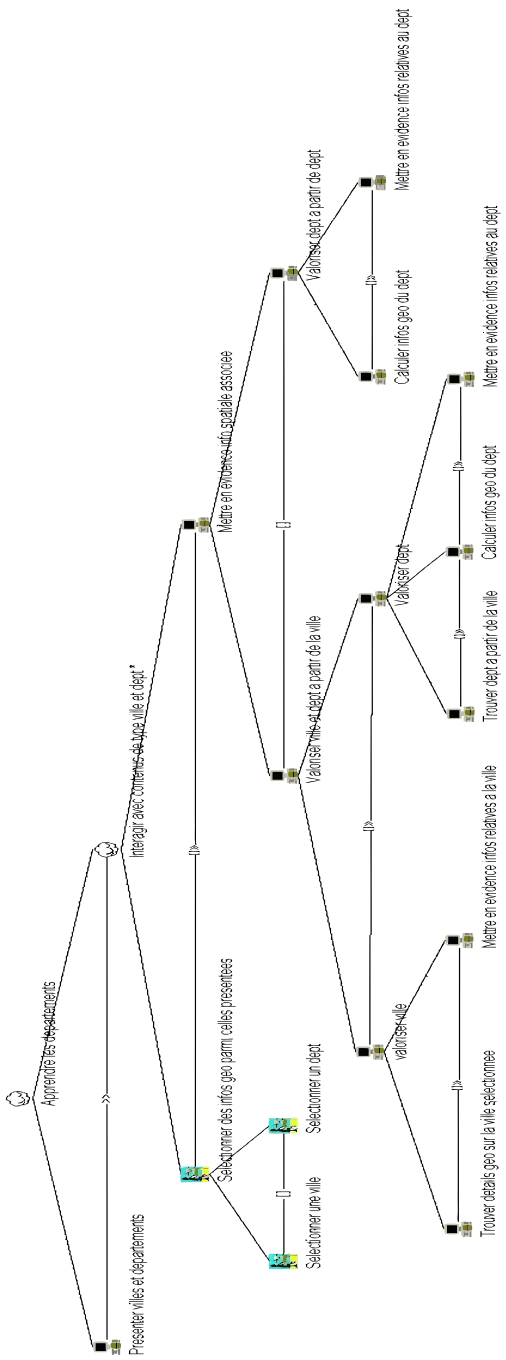

Fig. 2. Modèle de tâches CTT de l'application de lecture active du Tour de France

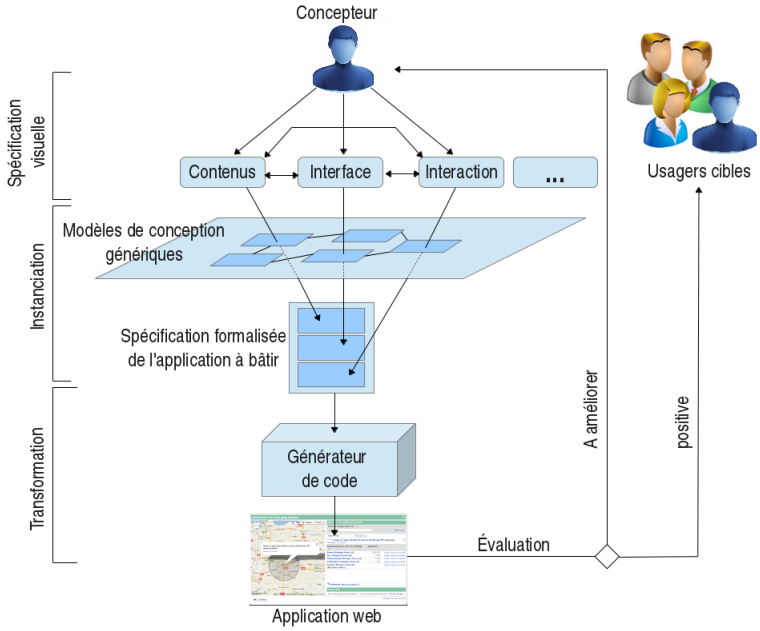

Fig. 3. Une démarche de conception d'applications interactives.

Cette spécification est réalisée en s'appuyant sur des modèles de conception génériques, chaque spécification se traduisant par une instance décrivant une propriété de l'application à élaborer. L'ensemble des instances constitue un cahier des charges structuré pouvant être lu par un générateur de code produisant automatiquement le code source (exécutable) d'une application (Web). Ceci permet au concepteur d'évaluer immédiatement sa spécification et de l'affiner si nécessaire ou bien de la publier vers des usagers cibles (et/ou lui-même) si elle est jugée satisfaisante.

Les travaux que nous avons menés depuis 2008 [Luong 2012] nous ont permis d'obtenir des résultats significatifs tant sur les modèles profonds de représentation des différents objets créés au niveau de chaque facette de conception [Luong et al. 2011], que sur les langages visuels rendant possible l'expression de ces modèles par des non-spécialistes de l'informatique [Luong et al. 2012]. Nous avons également conduit un ensemble de travaux de validation de l'approche en développant un démonstrateur nommé WINDMash[6] implémentant les principes de l'approche proposée. Ce démonstrateur qui offre un dispositif de conception pour décrire les trois facettes fondamentales d'applications géographiques a fait l'objet de nombreux travaux en amont [Luong et al. 2010] et d'expérimentations avec des utilisateurs en aval [Luong et al. 2011].

En complément de nos travaux cités ci-dessus, nous proposons dans cet article :

– La présentation globale du processus de conception illustré dans la figure 3. Plus précisément, nous montrons la complémentarité des différents modèles de conception et des outils visuels associés.

[6] `http://erozate.iutbayonne.univ-pau.fr/Nhan/windmash5s/`

– L'évaluation de la cohérence globale ainsi que de la flexibilité de notre processus de conception en analysant la façon avec laquelle les concepteurs exploitent et combinent étape par étape chacune des trois phases proposées (Contenus, Interface, Interaction).

La section suivante propose un état de l'art relatif aux divers points mis en évidence dans cette introduction. Tout d'abord, nous allons présenter un état de l'art sur les environnements-auteurs Web pour la gestion et l'affichage de contenus géographiques. Nous étudierons également les modèles conceptuels ainsi que les langages visuels permettant de décrire les interactions à mettre en œuvre pour réaliser la tâche visée par l'application. La section 3 présente ensuite nos contributions en termes de modèles et langage : notre modèle permettant de définir les informations géographiques utiles à la réalisation de la tâche, le modèle d'interface simple permettant de présenter ces informations (section 3.1), et enfin le modèle d'interaction sur lequel le concepteur pourra s'appuyer pour définir les fonctionnalités de son application ainsi que le langage visuel associé (section 3.2). La section 4 présente l'environnement-auteur Web nommé WINDMash (section 4.1) ainsi que le résultat des évaluations menées pour valider nos propositions (section 4.2). Nous dressons un bilan de nos contributions et présentons les perspectives s'ouvrant à nous dans la section 5.

2. ÉTAT DE L'ART

Les outils de conception que nous visons doivent permettre au concepteur de spécifier :

– les contenus de son application. Nous présentons dans la section 2.1 un état de l'art sur les Mashups qui facilitent la création et l'agrégation de données.
– le comportement de son application. Nous étudions dans la section 2.2, les différents modèles et langages permettant de décrire l'interaction de manière visuelle.

2.1 Environnement Web pour la gestion et l'affichage de contenus

Dans cette section, nous présentons certains outils Web permettant de faciliter la gestion et l'affichage de données telles que les informations géographiques. Étant donnés nos objectifs, nous focalisons notre présentation sur les outils de type Mashup (ou applications composites) qui permettent de définir des contenus par combinaison / agrégation de données hétérogènes, pour certains de manière visuelle. Ces outils permettent souvent de définir aussi la manière de présenter des contenus sur des composants prédéfinis de type tableau, cartes géographiques... Nous classons les Mashups selon le type d'utilisateurs : développeurs (informaticiens), utilisateurs avancés (connaissant certains détails de fonctionnement des technologies informatiques sous-jacentes) et utilisateurs professionnels (ne maîtrisant pas ces technologies informatiques sous-jacentes). Le tableau I offre une comparaison entre différents Mashups pour la gestion et l'affichage de contenus. En plus du type d'utilisateurs cibles, nous avons choisi les critères suivants pour la comparaison :

– **Méthode de conception** : Approche pour concevoir l'application de type Mashup ;
– **Outillage** : Quantité d'outils proposés ;
– **Technologies** : Technologies et langages utilisés pour implémenter les Mashups ;
– **Intégration** : Façon d'intégrer les Mashups dans un site Web ;
– **Services Web** : Indique l'usage possible ou non de service(s) Web ;
– **Interactivité** : Spécification par le concepteur de comportements interactifs.

	Méthode de conception	Outillage[7]	Technologies	Intégration	Services Web	Interactivité
Pour les développeurs						
GME[8]	Codage	Important	XML	Code XML à intégrer dans iGoogle	N/A	N/A
Exhibit [Huynh et al. 2007]	Codage	Important	HTML, JSON	Non	Non	Oui mais prédéfinie
Chickenfoot [Bolin et al. 2005]	Codage	Faible	JavaScript	Non	Non	Non
PiggyBank [Huynh et al. 2005]	Codage	Faible	RDF, JavaScript	Non	Non	N/A
Pour les utilisateurs avancés						
Yahoo! Pipes[9]	Flux de données	Important	Web standard, YUI	Code HTML à intégrer	Oui	Oui mais prédéfinie
Popfly[10]	Flux de données	Important	Silverlight	Code HTML à intégrer	Oui	Oui mais prédéfinie
Damia [Altinel et al. 2007]	Flux de données	Important	REST, XML	Non	Oui	Oui mais prédéfinie
Pour les utilisateurs professionnels						
Afrous[11]	Barre d'outils dédiés	Important	JavaScript	Code HTML	Oui	N/A
Marmite [Wong and Hong 2007]	Flux de données	Faible	Plugin Firefox, XUL, JavaScript	N/A	Oui	Oui mais prédéfinie
MashMaker [Ennals and Garofalakis 2007]	Barre d'outils dédiés	Important	Plugin Firefox	N/A	Oui	Oui mais prédéfinie
Mashlight [Albinola et al. 2009]	Flux de données	Faible	JavaScript, XML	Non	Oui	Données, Oui mais prédéfinie

Table I. Comparaison des environnements Web pour la gestion et l'affichage de contenus.

Synthèse : Du point de vue des Mashups, et plus précisément sur l'aspect "Technologies", nous pouvons constater qu'ils utilisent principalement le langage JavaScript, ce qui facilite la spécification de l'interface des applications Web. Pour les non-développeurs, les contenus ainsi que la méthode de conception de l'application reposent globalement sur la spécification de flux de données car l'usage des Mashups par codage n'est pas à la portée de tout le monde. Ces systèmes sont génériques et ne sont pas exclusivement conçus pour construire des applications géographiques, d'où le fait qu'ils ne proposent pas de plateforme pour la conception de ce type particulier d'applications. En outre, pour les quatre autres critères, la plupart des systèmes cités ne prennent pas en compte toutes ces caractéristiques à la fois, c'est-à-dire un outillage important, une intégration aisée au sein d'un site Web, l'appel possible de services Web variés ainsi que la spécification libre d'interactions entre les contenus affichés sur l'interface de l'application. L'environnement-auteur que nous proposons dans la section 4 couvre l'ensemble de ces critères.

[7] Le seuil subjectif entre les valeurs Faible et Important se situe à une dizaine d'outils.
[8] Google Mashup Editor a été arrêté et migré dans Google App Engine.
[9] `http://pipes.yahoo.com/pipes/`
[10] Popfly de Microsoft a pris fin à partir de l'été 2009.
[11] `http://www.afrous.com/en/`

2.2 Modèles et langages visuels pour décrire l'interaction

Dans cette section, nous nous intéressons aux modèles et langages permettant de décrire l'interaction au sein d'une application. L'exploration de la littérature relative aux Interactions Homme-Machine [Silva 2000], [Mori et al. 2004], [Caffiau et al. 2009], [Rogers et al. 2011], et notamment les travaux de [Hewett et al. 1992] et [Sears and Jacko 2007] montrent qu'il n'existe pas de définition convenue de ce qu'est une interaction. Dans le cadre de nos travaux, nous croisons les définitions de [Marion 1999] et [Vuillemot and Rumpler 2009] pour définir une interaction comme "*une action généralement mise en œuvre par un acteur humain sur un système et qui déclenche une réaction du système qui peut être directe (perceptible par l'acteur humain) ou indirecte (interne au système comme par exemple un calcul)*".

2.2.1 *Modèles pour spécifier l'interaction.* La dimension complexe que revêt la notion d'interaction est retranscrite en partie par la variété des définitions présentes dans la littérature mais aussi par le nombre de modèles proposés pour la décrire [Kolski 2001; Diaper and Stanton 2004]. Nous retrouvons :

– Les *modèles de tâches* dédiés à la spécification des actions que l'utilisateur peut effectuer sur le système final en vue d'atteindre un objectif précis : [Card et al. 1983], MAD [Scapin and Pierret-Goldbreich 1989] et son extension MAD* [Gamboa and Scapin 1997], CTT [Paternò et al. 1997], TOOD [Mahfoudhi et al. 2001; Ormerod and Shepherd 2004], AMBOSS [Giese et al. 2008], Diane+ [Tarby and Barthet 1996], UAN [Hix and Hartson 1993], GTA [Veer et al. 1996] ou encore UsiXML [Limbourg and Vanderdonckt 2004] qui regroupe plusieurs modèles (dont un modèle de tâches) pour décrire l'interaction.
– Les *modèles de dialogue* permettant de décrire les échanges entre l'utilisateur et le système : [Green 1986; Churcher et al. 1997; Britts 1987; Pallota 2003; Luyten et al. 2003; Caffiau et al. 2009].
– Les *modèles de présentation* qui visent à décrire les propriétés et les éléments composant l'interface utilisateur : [Guerrero-Garcia et al. 2009; Normand 1992].
– Les *modèles architecturaux* qui visent à organiser le code d'une application interactive de manière raisonnée, de sorte à faciliter son développement, sa maintenance et son évolution : Seeheim [Pfaff 1985; Dragicevic and Fekete 2004], AMF [Samaan and Tarpin-Bernard 2004], Arch [Bass et al. 1992], PAC [Coutaz 1987] ou encore MVC [Krasner and Pope 1988].

Des cadres de modélisation tels que UsiXML [Limbourg and Vanderdonckt 2004], UIML [Abrams et al. 1999], CAMELEON [Calvary et al. 2003] ou encore TERESA [Paternò and Santoro 2003; Correani et al. 2005] offrent au concepteur des "packs" de modèles complémentaires permettant de décrire les couches tâches et/ou dialogue et/ou présentation et/ou architecturale.

Ces différents modèles offrent un cadre de modélisation riche qui permet de décrire les différentes facettes de l'interaction. Généralement, la modélisation de l'interaction débute par la formalisation des tâches que l'utilisateur doit accomplir. À la vue des tâches à réaliser, il s'agit ensuite de définir les dialogues que le système devra pouvoir mettre en œuvre afin d'assister l'utilisateur dans la réalisation de ses tâches. Les modèles de présentation viennent donner un visage à l'interface concrète avec laquelle l'utilisateur final interagira pour accomplir ses tâches. Les modèles architecturaux proposent, en bout de

chaîne, un cadre logiciel pour produire un code applicatif dont le comportement interactif est conforme aux propriétés établies à l'aide des modèles précédents.

L'usage de ces différents modèles nécessite toutefois des capacités d'abstraction et de modélisation qui demeurent l'apanage de spécialistes en informatique. Il faut en effet être capable de séparer clairement les différents niveaux d'abstraction qui permettent de décrire les différentes couches interactives d'un système.

Dans le cas des Systèmes d'Information Géographique (SIG), tels que ArcGIS ou IGN par exemple, les données sont ordonnées selon des couches spécifiques [Gaio et al. 2008] : cours d'eau, réseaux de transport, communes, etc. Cette organisation par couche permet de définir des interactions propres aux données de la couche telles que l'affichage d'attributs spécifiques et la façon dont ils sont représentés (modification de l'apparence de la couche, placement de la couche en arrière plan, etc.).

Dans le cadre de nos travaux, il nous semble important de reprendre cette organisation et de permettre au concepteur de définir des interactions sur des données géographiques en tenant compte de la couche à laquelle elles appartiennent. Ceci sous-entend d'être en mesure de pouvoir qualifier / typer chaque donnée pour permettre au concepteur de définir des interactions sur l'ensemble des données appartenant à une couche spécifique (par exemple, surligner l'ensemble des données de type "oronyme"). D'autre part :

– Nous nous centrons, à ce stade, sur la conception d'applications géographiques ciblant une et une seule tâche.
– Nous souhaitons également, à terme, que la création de ces applications puisse être réalisée en autonomie par des concepteurs non-informaticiens.

Ces deux principales raisons nous poussent à écarter tout cadre de modélisation qui s'avérerait trop complexe :

– Les *modèles de tâches* tels qu'ils sont définis dans les travaux précédemment cités nous semblent trop lourds à utiliser vis-à-vis de la taille des applications visées mais aussi des capacités d'analyse et d'abstraction des concepteurs que nous souhaitons toucher. En effet, ces modèles sont généralement exploités par un informaticien ou un ingénieur des connaissances (parfois appelé cogniticien ou ingénieur pédagogique dans le domaine de l'éducation) afin de modéliser les connaissances de l'expert du domaine [Matta 2004], [Rialle 1990]. Même si nous délaissons la richesse des modèles de tâches, il nous semble néanmoins important de pouvoir identifier les informations qui entrent en jeu dans la résolution de la tâche ainsi que les fonctionnalités que l'application finale devra intégrer pour mener à bien la tâche.
– Les *modèles de dialogue* nous semblent incontournables dans le sens où ils décrivent les actions possibles d'un utilisateur sur un système ainsi que les réactions de ce dernier. Ce type de modèle reste au cœur de l'interaction et notre solution doit donc offrir les moyens d'exprimer cette dimension de l'interaction.
– Les *modèles de présentation* recouvrent également un aspect important de l'interaction puisqu'ils décrivent la couche visible de l'interaction. Au-delà du fait qu'elle est incontournable, cette couche, de par sa nature visuelle, reste particulièrement importante pour des concepteurs non-experts qui auront plus de facilité à raisonner sur des éléments visuels de l'interface utilisateur finale.
– Les *modèles architecturaux* jouent un rôle secondaire pour les concepteurs que nous visons. Étant donné que nous souhaitons mettre en place des outils de génération au-

tomatique de code, les concepteurs n'ont pas besoin de se soucier de la manière dont le code final est organisé.

Nos travaux visent à proposer des modèles permettant de décrire l'interaction de manière visuelle. Selon les modèles cités précédemment nous nous situons à la croisée des modèles de dialogue et des modèles de présentation, notre objectif étant de les combiner pour, au final, obtenir une application capable d'assister l'utilisateur final sur une tâche spécifique. Le but de la tâche visée et la manière de l'atteindre ne seront donc pas formellement définis. La tâche ne sera pas expressément formalisée sous la forme d'un graphe mais sera représentée par l'agrégation :

– des informations géographiques que le concepteur aura définies car jugées utiles à la réalisation de la tâche ;

– des interactions que le concepteur aura prévues car jugées nécessaires à l'exécution de la tâche.

Selon cette approche, le concepteur décrira les possibilités interactives de son application en fonction des données affichées, sachant que ces possibilités interactives consisteront à spécifier des dialogues possibles entre l'utilisateur et le système (couche dialogue). Il s'agit donc de proposer une approche hybride permettant à des concepteurs non-spécialistes en informatique de décrire les informations qui peuvent être rendues interactives sur une interface et les conséquences de chaque action utilisateur sur ces informations affichées.

2.2.2 *Langages visuels pour spécifier l'interaction.* Les modèles offrent un cadre de réflexion en fournissant au concepteur un ensemble de concepts et de propriétés permettant de décrire l'objet à concevoir. Les modèles évitent ainsi au concepteur d'identifier les propriétés sur lesquelles il doit se pencher pour décrire l'objet à élaborer. Si les modèles permettent d'apporter un cadre de réflexion structuré, il est souhaitable qu'ils proposent aussi un langage visuel ainsi que des outils adaptés permettant au concepteur d'instancier les concepts du modèle afin de décrire l'objet qu'il élabore.

En outre, dans le cadre des modèles de tâches, ces modèles reposent sur un méta-modèle qui liste les caractéristiques permettant de spécifier ce qu'est une tâche. À titre d'exemple, CTT fournit un méta-modèle (Figure 4, [Limbourg et al. 2001]) accompagné d'un langage graphique spécialisé pour décrire des tâches ([Paternò 2003]) ainsi qu'un outil d'édition graphique (CTTe). Notons que notre contribution (*cf.* section 3) va également dans ce sens en proposant un modèle de conception global accompagné d'un langage visuel de spécification, lui-même outillé par un éditeur graphique dédié (*cf.* section 4.1).

Nous centrons ici notre étude sur les langages de nature graphique jugés en général plus simples à lire et à interpréter que des langages textuels ou mathématiques. Nous portons donc ici notre attention sur les travaux relatifs aux langages visuels au sens de [Myers 1990] et plus précisément aux langages de programmation visuels tels que présentés dans [Myers 1990; Shu 1999; Koegel and Heines 1993; Narayanan and Hübscher 1998]. Nous définissons un *langage visuel de programmation* comme un "*langage graphique qui permet à des programmeurs de créer des programmes en manipulant des éléments du programme représentés graphiquement*". Le fait, qu'il s'agisse d'un langage de programmation sous-entend l'existence d'un système capable de traduire une description graphique sous forme

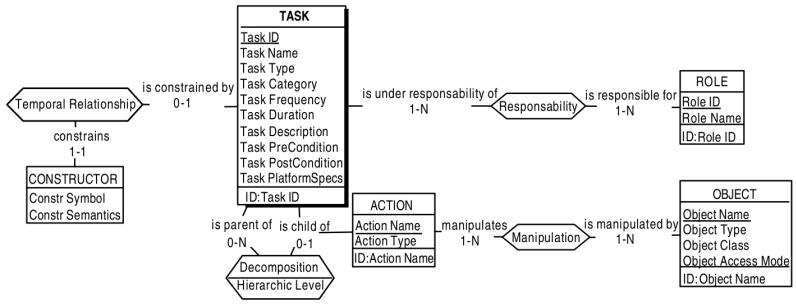

Fig. 4. Méta-modèle de CTT d'après [Limbourg et al. 2001]

de code exécutable comme c'est par exemple le cas pour Scratch[12], StarLogo[13] ou App Inventor[14].

2.2.2.1 *Langages visuels non spécifiques aux applications géographiques.* La notation ICON [Dragicevic and Fekete 2004] permet de décrire des interactions dans lesquelles les entrées utilisateur dépassent les techniques usuelles clavier-souris : interaction gestuelle, interaction multi-digitale, interaction 3D... Comme présenté sur la Figure 5, la notation et l'environnement de conception associé proposent des boîtes à outils qui permettent de décrire des interactions à l'aide de trois familles de dispositifs : les dispositifs d'entrée physique tels que le clavier ou la souris, les dispositifs de traitement pour effectuer des transformations de données et les dispositifs à retour graphique. L'interaction est décrite en plaçant ces dispositifs sur un plan de travail et en les connectant les uns aux autres via des ports spécifiques à chaque dispositif. Nous obtenons alors une interaction décrite selon un flux qui part d'un dispositif d'entrée et se termine sur un ou plusieurs dispositifs à retour graphique. Les interactions décrites fonctionnent avec des composant Java / Swing et le concepteur peut simuler directement l'interaction décrite au sein d'une fenêtre Java.

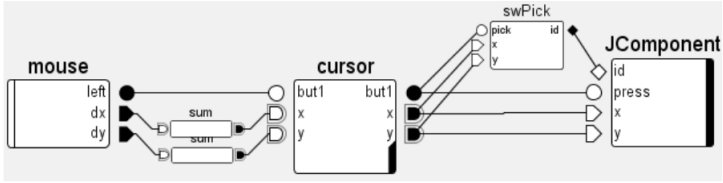

Fig. 5. Programmation visuelle avec ICON d'après [Dragicevic and Fekete 2004]

Comme ICON, Squidy [König et al. 2010] permet la spécification d'interactions qui vont au-delà des interfaces graphiques classiques : reconnaissance vocale, multi-touch, suivi de gestes... Le framework fournit un environnement graphique qui permet au concepteur de faire abstraction de la complexité technique nécessaire à la mise en œuvre de ce type

[12] http://scratch.mit.edu/
[13] http://education.mit.edu/starlogo/
[14] http://appinventor.mit.edu/explore/

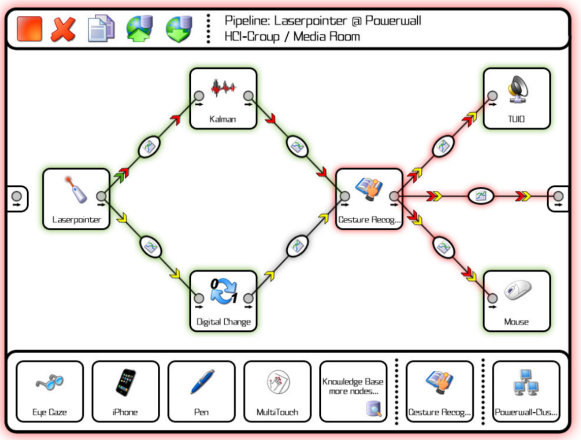

Fig. 6. Programmation visuelle avec Squidy d'après [König et al. 2010]

d'interactions. L'environnement fournit une notation visuelle basée sur la description de flux de données entre dispositifs (à la manière de ICON). La Figure 6 présente un exemple de flux dans lequel des données issues d'un pointeur laser sont envoyées vers différents filtres dont la sortie permet de simuler le comportement d'une souris traditionnelle.

2.2.2.2 *Langages visuels spécifiques aux applications géographiques.* L'idée d'intégrer des langages visuels de programmation dans les systèmes d'informations géographiques n'est pas nouvelle. Des systèmes tels que ArcGIS[15], Mapinfo[16] ou encore AutoCAD Map3D[17] restent massivement réservés à des spécialistes du domaine. Dans ce cadre, les langages visuels tentent de rendre accessibles de tels systèmes à des utilisateurs non-spécialistes.

Sur ArcGis, le langage visuel prend la forme d'un workflow construit à partir de deux entités graphiques de base : des rectangles aux bords arrondis représentent des traitements et des ellipses représentent les données qui sont en entrée et en sortie des traitements. Les données sont connectées aux traitements par des flèches dont le sens précise si les données sont en entrée ou en sortie. Pour donner du sens au workflow élaboré, le concepteur peut ajouter des annotations à divers endroits du diagramme. La Figure 7 croise des données géographiques concernant l'emplacement de collèges et l'emplacement de zones inondables pour construire un atlas permettant d'identifier les collèges compris dans une zone d'inondation spécifique.

De manière similaire à ArcGIS, AutoCAD Map3D permet de décrire graphiquement un ensemble de traitements à appliquer sur des données géographiques. Cette description est réalisée sous forme d'un workflow où les seules entités graphiques représentées sont les traitements. La Figure 8 met en place un workflow dans lequel 2 connexions à une banque de données sont réalisées en parallèle. Les données résultant de ces connexions sont ensuite croisées et superposées sur un calque unique.

[15] www.arcgis.com
[16] www.pbinsight.com/welcome/mapinfo
[17] www.autodesk.fr/map3d

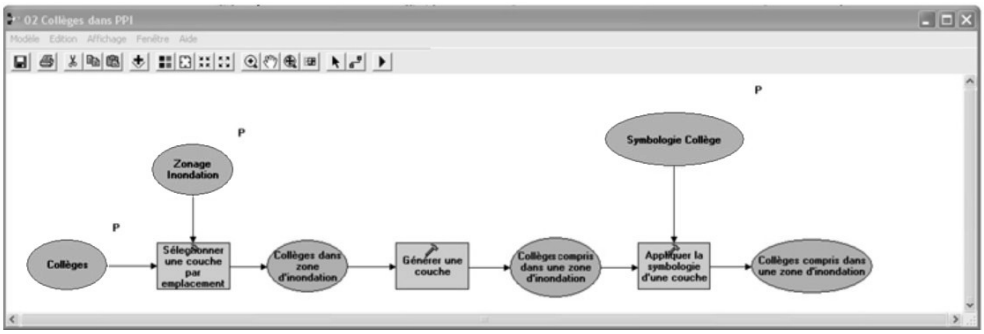

Fig. 7. Langage visuel utilisé sur ArcGis (*cf.* `www.arcgis.com`)

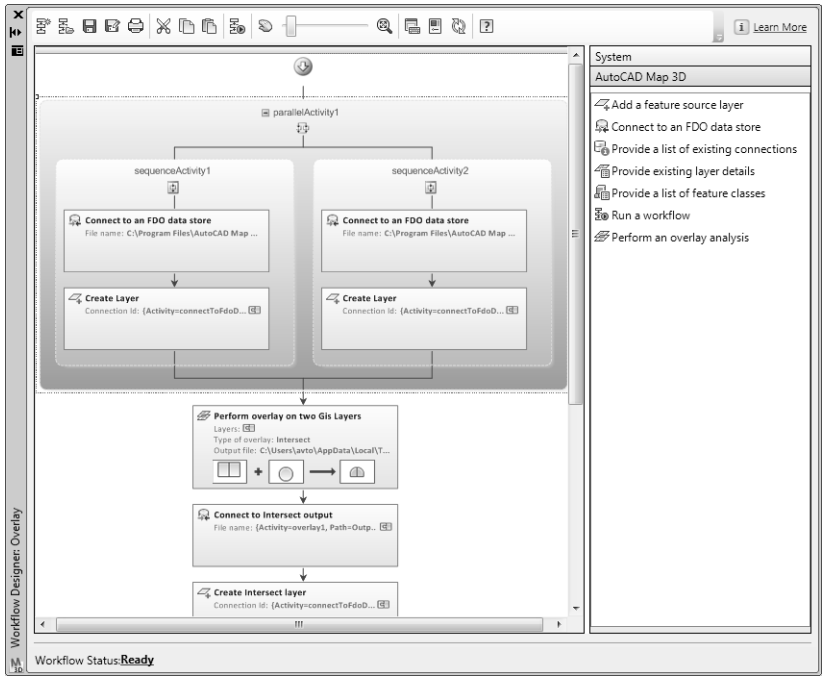

Fig. 8. Langage visuel utilisé sur AutoCAD Map3D (*cf.* `www.autodesk.fr/map3d`)

Les langages visuels décrits précédemment proposent de décrire l'interaction en termes de flux depuis un dispositif d'entrée jusqu'à un composant de sortie retranscrivant les réactions du système. Ces flux peuvent traverser des composants internes au système pour décrire des traitements visant à modéliser les répercussions de l'action utilisateur au sein du système. L'interaction reste à l'initiative de l'utilisateur et cette initiative est modélisée par l'usage d'un périphérique d'entrée. Les actions de l'utilisateur sont donc décrites par le biais de périphériques de saisie mais les éléments d'interface sur lesquels l'utilisateur agit via ces périphériques ne sont pas explicitement décrits. En ce sens, la partie de l'interaction

qui touche à la couche de présentation n'est que partiellement décrite et impose au concepteur de décrire le déclenchement de l'interaction à partir d'un périphérique et non pas en fonction des éléments graphiques disponibles sur l'interface.

2.2.2.3 *Cas particulier d'UML.* UML offre un standard de modélisation pour spécifier les différentes facettes d'une application informatique. Depuis sa version 2, UML propose treize modèles spécialisés permettant au concepteur de décrire les différentes caractéristiques d'une application en considérant plusieurs angles de vue. Ces diagrammes peuvent être assimilés à des langages visuels car ils offrent au concepteur des notations graphiques lui permettant de spécifier les caractéristiques de l'application qu'il souhaite concevoir. Beaucoup d'Ateliers de Génie Logiciel[18] et travaux de recherche ont montré qu'il était possible de générer du code à partir d'une spécification UML. Comme présenté dans [Barbier 2008; Kohler et al. 1999; Ziadi et al. 2009], UML peut être utilisé comme langage de modélisation pour générer du code Java à partir de diagrammes de classes et de diagrammes états-transitions. Les aspects interactifs du système sont décrits par des diagrammes états-transitions qui spécifient les possibilités interactives de l'utilisateur en fonction de l'état du système mais aussi les réactions internes de ce dernier lorsque les éléments qui le composent changent d'état suite à une action de l'utilisateur. Les travaux présentés dans [Harel and Marelly 2003] mettent en avant les capacités intéressantes des diagrammes de séquence UML pour décrire en temps réel, à l'exécution, les interactions de l'utilisateur avec son application.

À la manière d'ICON ou de Squidy, les diagrammes de séquence décrivent l'interaction sous forme de flux matérialisant les échanges entre l'utilisateur et le système mais aussi entre les composants du système. Les diagrammes de séquence permettent, par contre, de faire abstraction des périphériques utilisés par l'utilisateur et soulignent, sous forme de messages, ce que l'utilisateur peut faire sur le système indépendamment du périphérique d'entrée utilisé. Ces diagrammes sont d'ailleurs quasiment les seuls à intégrer une représentation de l'utilisateur et de son rôle dans l'interaction. La représentation de l'interaction sous forme de flux décrivant les échanges entre l'utilisateur et le système nous apparaît comme une manière naturelle pour décrire ce qui se passe entre un utilisateur et un système au cours d'une interaction. De plus, contrairement aux diagrammes états-transitions, ils permettent au concepteur de décrire la dimension interactive du système en plusieurs étapes et non pas dans sa globalité et pour une seule classe. Cette propriété de décomposition de l'interaction nous semble également intéressante pour des concepteurs non-informaticiens qui peuvent avoir du mal à considérer la dimension interactive d'une application dans sa globalité, en particulier si le système à concevoir offre des possibilités interactives nombreuses.

3. MODÈLES ET LANGAGES

L'état de l'art présenté en section 2 fait ressortir un ensemble de modèles permettant de bâtir des applications web géographiques. Nous remarquons toutefois que ces modèles n'ont pas été pensés pour être facilement couplés. Ceci sous-entend soit un travail supplémentaire du concepteur (pour les interconnecter) soit une modélisation partielle focalisée sur un des modèles au détriment des autres.

[18] IBM Rational Rose (`www.ibm.com/software/fr/rational`), Modelio (`www.modeliosoft.com`), BOUML (`www.bouml.fr`), etc.

À partir de ce constat nous proposons un ensemble de modèles interconnectés (modèle global présenté sur la figure 9 et détaillé sur les figures 10, 11 et 12) permettant de spécifier de manière complémentaire les données manipulées (Contenu - *cf.* section 3.1.1) au sein de l'application, la manière de les présenter (Interface - *cf.* section 3.1.2) mais aussi les comportements interactifs associés (Interaction - *cf.* section 3.2). Avec le modèle proposé, les contenus annotés restent au centre de la démarche de conception : le concepteur les définit, les présente sur un écran et les rend interactifs. Une annotation est définie comme un contenu qui sera valorisé au sein de l'application. Dans cette approche, le concepteur conçoit son application en raisonnant sur des contenus à valoriser. Ces contenus sont identifiés lors d'une phase d'annotation dans laquelle le concepteur définit (annote) les contenus qui seront présentés à l'utilisateur ou qui joueront un rôle dans une interaction avec l'utilisateur. Dans le cadre d'une interaction, ces contenus annotés pourront initier l'interaction (clic sur un contenu présenté à l'écran par exemple) ou bien résulter d'une interaction (apparaître à l'écran suite à une action de l'utilisateur sur un autre contenu par exemple). La phase d'annotation des contenus à valoriser est présentée et illustrée dans la section 4.1.1 et dans la figure 20.

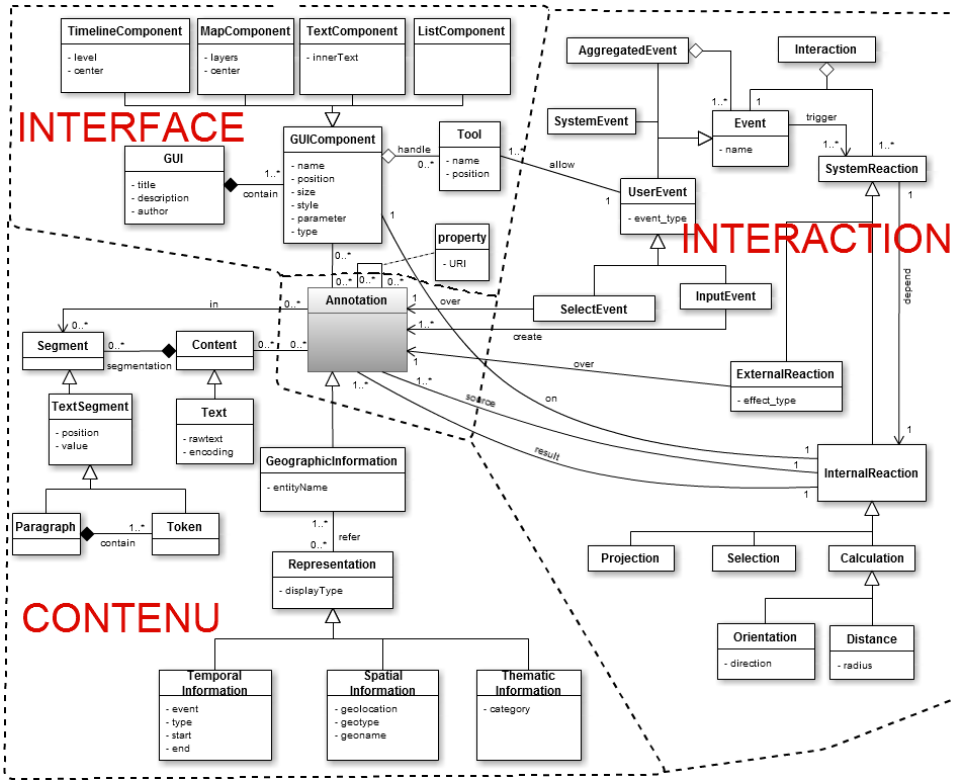

Fig. 9. Modèle global pour la conception d'applications géographiques

3.1 Modèles de contenu et d'interface

Cette phase se concentre sur les contenus (géographiques) que les concepteurs voudraient présenter à l'utilisateur lors de l'exécution. Les contenus représentent le concept central de notre processus car notre approche de conception a pour but de valoriser des données choisies par le concepteur. Cette valorisation se concrétise par :

– une présentation de ces contenus (à l'écran) selon des modes de représentation variés ;
– une mise en valeur de ces données auprès de l'utilisateur via des interactions.

3.1.1 *Modèle de contenu.* Nous proposons un modèle (Figure 10) pour décrire les contenus d'applications Web géographiques [Luong et al. 2011]. Nous considérons des contenus (`Content`) (1) qui peuvent être composés de plusieurs segments (`Segment`) (2). Comme illustré en figure 10, un contenu peut détenir plusieurs annotations (`Annotation`) (3) faisant référence à des segments spécifiques. Une annotation peut être connectée avec d'autres annotations grâce à des propriétés (`property`) (4) différentes. Par exemple, une annotation sur "Biarritz" est liée à une annotation sur "Anglet" par la propriété dont l'`URI` (Uniform Resource Identifier[19]) est `wind:nextTo` (à côté de).

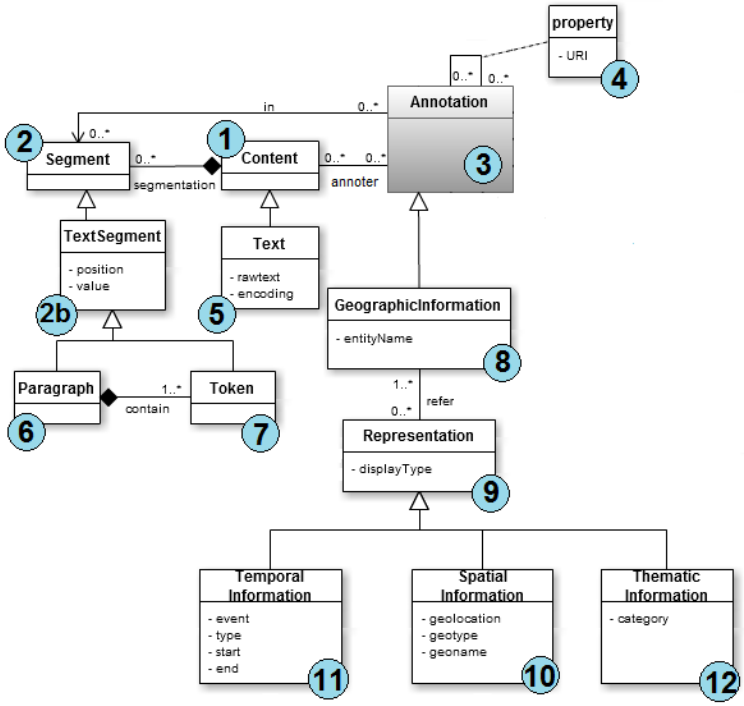

Fig. 10. Modèle de description des contenus

Actuellement, nous envisageons exclusivement le contenu textuel comme point de départ dans une application géographique. Par conséquent, comme présenté dans la figure

[19] `http://www.w3.org/TR/cooluris/`

10, les textes (`Text`) (5) peuvent être segmentés en paragraphes (`Paragraph`) (6) et tokens (`Token`) (7) qui héritent d'un segment textuel (`TextSegment`) (2b). Un segment textuel peut avoir une position dans le texte de départ qui a servi à l'extraire (par exemple, 6^{e} mot, 4^{e} paragraphe), une valeur présentée sous forme d'une chaîne de caractères (par exemple, "*Paris*", "*J'habite à 10 kilomètres au sud de Paris.*").

Par ailleurs, dans notre modèle de *Contenu*, l'information géographique (`Geographic-Information`) (8) est une annotation (3) (lien d'héritage). L'information géographique est composée de trois facettes : spatiale, temporelle et thématique. Ainsi, une information géographique (`GeographicInformation`) (8) peut avoir plusieurs représentations (`Representation`) (9) de nature spatiale (`SpatialInformation`) (10) ou temporelle (`TemporalInformation`) (11) ou thématique (`ThematicInformation`) (12).

La représentation spatiale d'une annotation se traduit toujours par des coordonnées géographiques permettant de situer l'annotation sur une carte. Elle est décrite selon une forme qui peut être un point (par exemple une localisation ou un endroit), une ligne (par exemple un fleuve ou une route), un polygone (par exemple une zone ou une ville) ou un ensemble de coordonnées géographiques correspondant à cette forme. Pour cadrer avec les SIG traditionnels (*cf.* section 2.2.1), chaque information spatiale doit pouvoir être typée pour permettre au concepteur de définir ensuite des interactions sur la couche à laquelle appartient cette information. La représentation temporelle d'une annotation peut être un instant dans le temps (par exemple la date "*2014-08-16*") ou une période (par exemple "*du 21 septembre 2003 au 23 octobre 2008*"). La représentation thématique d'une annotation peut prendre une forme textuelle et peut être catégorisée via des concepts dont la sémantique est définie dans une ontologie par exemple.

3.1.2 *Modèle d'interface.* Cette phase permet au concepteur d'organiser l'interface utilisateur de l'application générée (taille, position, fournisseur de la carte, le niveau de zoom...). Le concepteur peut définir le rendu visuel de l'application géographique finale composée des différents afficheurs adaptés aux contenus géographiques manipulés. Une interface se compose d'afficheurs qui affichent les informations (contenus géographiques) de la phase *Contenu* et le concepteur peut décider où et comment chaque afficheur est présenté à l'écran.

Cette section a pour but de présenter le modèle élaboré pour présenter les contenus dans l'interface graphique de l'application finale [Luong et al. 2011]. L'interface de l'application est vue comme une couche de visualisation permettant de présenter des contenus à l'utilisateur sous diverses formes. Une interface utilisateur est construite par l'assemblage de plusieurs composants d'interface, chaque composant est spécialisé pour présenter des contenus sous une forme particulière.

Bien que la phase *Interface* ne soit pas une contribution forte en terme de recherche, elle est nécessaire (Figure 11) pour prendre en compte les éléments de la phase *Contenu* jusqu'à l'exécution.

Une application géographique contient une interface utilisateur graphique (`GUI`) (1). Celle-ci peut être composée de plusieurs composants d'interface (`GUIComponent`) (2) organisés dans la mise en page de l'application. Actuellement, nous considérons quatre types de composant d'interface : texte (`TextComponent`) (3), carte (`MapComponent`) (4), frise chronologique (`TimelineComponent`) (5) et liste (`ListComponent`) (6) qui sont les sous-classes de `GUIComponent` (2) dans la figure 11. Chaque composant d'interface prend en charge ses propres paramètres de configuration et d'affichage. Une

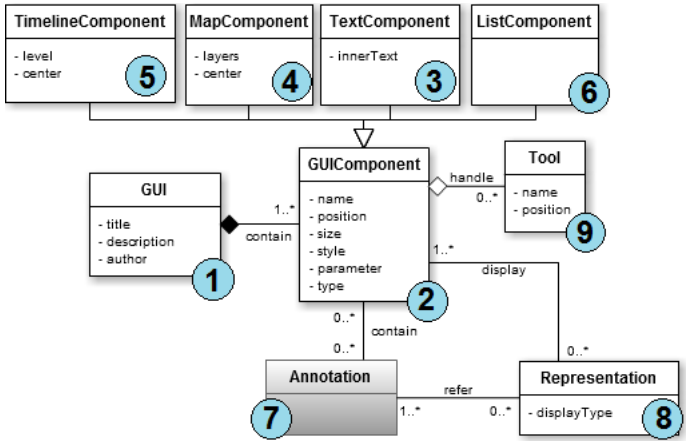

Fig. 11. Modèle de description des interfaces utilisateur

annotation (7) peut apparaître dans un ou plusieurs composants d'interface (2) pouvant contenir une ou plusieurs annotations (7) et afficher les représentations (8) de ces annotations qui sont compatibles avec le type de composant d'interface :

– Pour un `TextComponent` (3), la représentation du contenu est textuelle sous forme de chaîne de caractères ou encore de sa (ou ses) position(s) dans le texte. Les éléments textuels peuvent être automatiquement marqués par exemple avec des services Web qui extraient automatiquement les entités nommées.

– Pour un `MapComponent` (4), les entités géographiques sont marquées comme des géométries sur la carte. Un point représente un endroit, un lieu... Une ligne représente une route, une rivière ou un itinéraire par exemple. Un polygone représente une région, une ville... Le concepteur peut choisir les fonds cartographiques pour son application. Nous mettons en évidence la puissance de la prise en charge multi-couches de `MapComponent`, qui peut intégrer plusieurs couches de différents fournisseurs (IGN Geoportail, GoogleMaps . . .).

– Le `TimelineComponent` (5) affiche les informations temporelles (par exemple, date, période) dans une frise chronologique.

– Le `ListComponent` (6) affiche les annotations sous forme de liste à puces.

Chaque composant d'interface peut posséder un ou plusieurs outils (`Tool` (9)) permettant à l'utilisateur de créer une nouvelle annotation. Un composant textuel pourra proposer par exemple des outils pour annoter ou surligner du texte. Un composant cartographique pourra quant à lui proposer des outils pour tracer des points, des lignes ou des polygones.

Lorsqu'une annotation est projetée sur un composant d'interface et que cette annotation peut avoir plusieurs représentations possibles sur ce même composant, le concepteur doit préciser les représentations à utiliser pour représenter l'annotation sur ce composant d'interface. Par exemple, une annotation concernant "Bayonne" peut avoir trois représentations :

– une représentation textuelle au 5e token dans le premier paragraphe d'un texte donné[20] ;

– une représentation cartographique qui est un point dont la longitude est -1.475 et la latitude est 43.4936, sa géolocation est donc `POINT(-1.475 43.4936)` ;

– une autre représentation cartographique dont la géolocation est sous forme d'un polygone : `MULTIPOLYGON(((-1.49222 43.50884,-1.49223 43.50901,...,-1.49222 43.50884)))`. Cette représentation cartographique est plus intéressante car elle couvre la représentation ponctuelle.

3.2 Modèle et langage pour l'interaction

Nous décrivons ici un modèle et un langage visuel facilitant la conception et l'implémentation d'interactions sur des contenus géographiques à manipuler pour réaliser une tâche.

3.2.1 *Un modèle d'interaction axé sur les contenus.* Dans cette section, nous proposons un modèle d'interaction élaboré selon l'hypothèse suivante : une application finale met à disposition de l'utilisateur des contenus sur une interface avec lesquels il peut interagir dans le but de réaliser une tâche. Le résultat d'une interaction se traduit par la valorisation et la création de nouveaux contenus nécessaires à l'exécution de la tâche. Dans cette approche, la conception de l'interaction est guidée par des contenus (géographiques) jouant un rôle dans la tâche à réaliser.

Le modèle d'interaction que nous proposons est décrit sur la figure 12. Une interaction (1) est définie comme un événement particulier (2) déclenchant une réaction du système (3). L'événement déclencheur peut être envoyé par le système (4), créé par une action de l'utilisateur (5) ou il peut encore s'agir d'une agrégation (6) d'événements système et/ou utilisateur ayant eu lieu (par exemple, il est 10h et l'utilisateur a réalisé une action particulière).

Un événement utilisateur (5) peut être défini à partir de deux types d'actions possibles : il peut s'agir d'une action de sélection ou d'une action de saisie. Une action de sélection (7) est définie par la sélection d'une annotation (8) affichée à l'interface utilisateur (9) tandis qu'une action de saisie (10) est caractérisée par la création d'une nouvelle annotation (8) dans un composant d'interface (9). Dans ce modèle, les actions de l'utilisateur se résument à désigner des annotations présentées à l'interface (par exemple, `click`, `mouseover`, `dragdrop`...) ou bien à créer de nouvelles annotations via des éléments d'interface spécifiques (e.g., des boutons d'annotation manuelle dans un composant textuel ou des outils à dessiner des lignes, polygones sur un composant cartographique).

Les réactions du système (3) peuvent être de deux natures : externes ou internes. Les réactions externes (11) sont des réactions du système perceptibles par l'utilisateur. Elles sont définies par un effet visuel (`show`, `hide`, `highlight`, `zoom`...) appliqué sur une annotation (8) présentée sur l'interface utilisateur (`GUIComponent`) (9). Les réactions internes (12) correspondent à des opérations qui, au cours d'une interaction, vont créer une nouvelle annotation, modifier ou déplacer une annotation existante. Le modèle actuel considère trois types de réactions internes :

– les opérations de projection (`projection`) (13) permettent de copier une annotation (8) existante d'un composant d'interface (9) vers un autre ;

[20] Je suis allé à Bayonne le 15 août 2014.

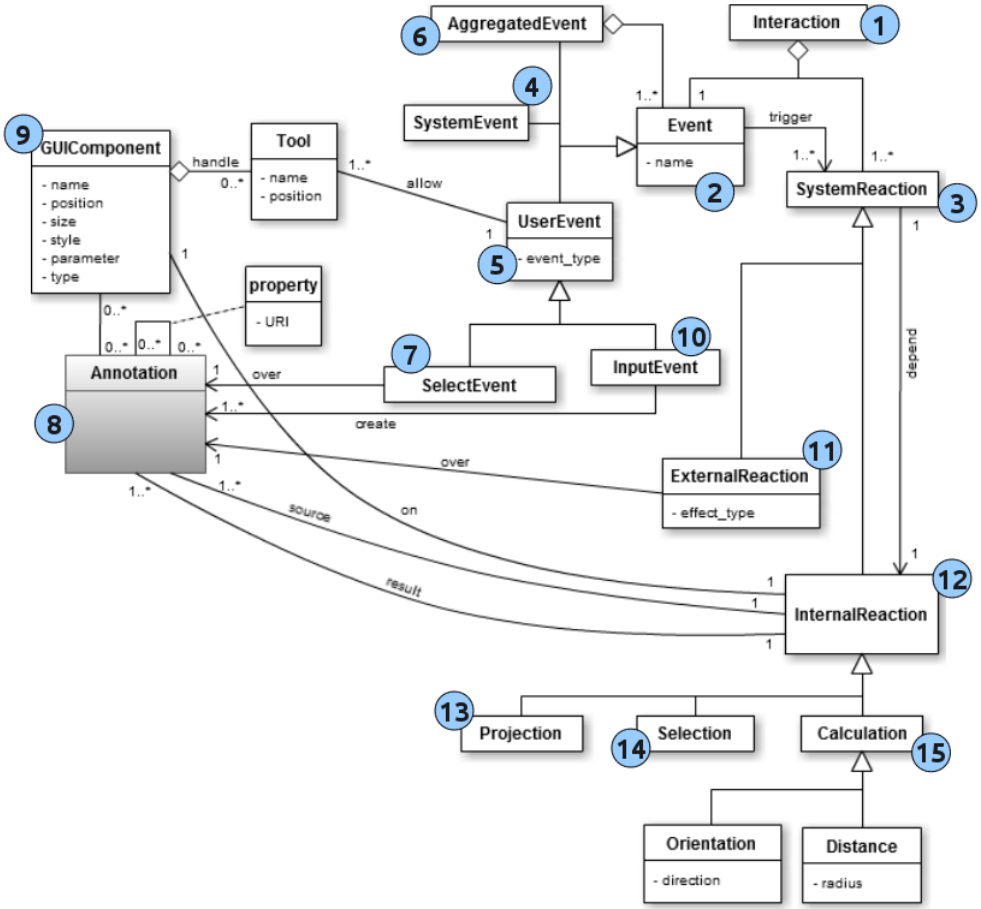

Fig. 12. Modèle de description des interactions

– les opérations de sélection (`sélection`) (14) permettent au système de déterminer quelle annotation (8) a été sélectionnée par l'utilisateur parmi toutes les annotations affichées sur un composant d'interface (9) ;
– les opérations de calcul (`calculation`) (15) permettent de calculer de nouvelles annotations à partir d'annotations existantes ou précédemment calculées. Dans le cadre des applications géographiques, ces opérations de calcul seront de nature spatiale ou temporelle (par exemple, déterminer le département auquel appartient une ville, calculer le nombre de jours entre deux dates...)

Quelle que soit l'interaction considérée, une action de l'utilisateur sur le système pourra engendrer une ou plusieurs réactions internes mais l'interaction devra, à terme, se conclure par une ou plusieurs réactions externes appliquant ainsi des effets visibles par l'utilisateur sur une ou plusieurs annotations présentées sur l'interface utilisateur.

3.2.2 *Un langage visuel pour décrire l'interaction.* Le modèle d'interaction présenté dans la section 3.2.1 propose un ensemble de concepts pour décrire l'interaction au sein d'une application. Le modèle représente en lui-même un cadre de réflexion qui permet de guider le travail du concepteur en lui proposant de spécifier l'interaction à partir de contenus interactifs pouvant déclencher des réactions du système.

Pour faciliter la conception de l'interaction, nous proposons dans cette section un langage visuel permettant au concepteur de décrire les éléments constituant chaque interaction qu'il souhaite mettre en place. Basé sur le modèle d'interaction présenté sur la figure 12, le langage visuel proposé permet de caractériser, pour chaque interaction, des actions utilisateur, les contenus sur lesquels portent ces interactions, les réactions internes du système (projection, sélection, calcul) ainsi que les réactions externes (effets visuels résultant de l'interaction).

3.2.2.1 *Principes du langage visuel.* Comme annoncé en synthèse (*cf.* section 2.2.2), nous retenons les potentialités intéressantes des diagrammes de séquence UML pour décrire des interactions. L'intérêt porté à ces diagrammes est né des observations suivantes :

– Ils sont plutôt faciles à maîtriser car ils se basent sur peu de concepts pour décrire les interactions entre un utilisateur et un système.
– Ils distinguent clairement les actions de l'utilisateur sur le système (messages allant de l'utilisateur vers le système) et les réactions du système vers l'utilisateur (messages allant du système vers l'utilisateur).
– Ils permettent aussi bien de décrire les interactions entre l'utilisateur et le système que les interactions entre les composants du système.
– La chronologie des messages est clairement et simplement exprimée avec la ligne de vie dans sa dimension verticale.

Comme l'ensemble des langages visuels présentés en section 2.2.2, ils permettent de décrire l'interaction sous forme de flux échangés entre l'utilisateur et le système. Cette représentation par flux nous apparaît comme une manière intuitive de décrire les différentes phases d'une interaction depuis l'initialisation de l'interaction via une action utilisateur, en passant par les répercussions de cette action sur les composants internes du système et en allant jusqu'aux retours visuels finaux qui clôturent l'interaction et restituent un résultat à l'utilisateur.

Le mode d'expression de ces diagrammes permet de retranscrire assez naturellement des interactions décrites sous la forme "*lorsque l'utilisateur réalise telle action sur tel contenu affiché sur tel composant de l'application, le système réagit de la manière suivante*". Cette façon informelle de décrire une interaction nous semble assez proche du schéma de pensée utilisé par un concepteur non-informaticien pour décrire le comportement de l'application qu'il souhaite réaliser.

Chaque diagramme décrit une et une seule interaction : une action de l'utilisateur sur le système provoquant une ou plusieurs réactions de la part de ce dernier. Ces diagrammes permettent donc au concepteur de décrire la couche interactive par étape, interaction après interaction. L'ensemble des diagrammes de séquence décrits spécifieront ainsi l'ensemble des capacités interactives de l'application construite.

Le langage que nous proposons s'inspire du formalisme des diagrammes de séquence UML mais les diagrammes résultants ne peuvent pas être considérés comme des diagrammes de séquence en soi ni une de leurs extensions. Les adaptations majeures que

nous proposons sont les suivantes :

– Les seuls composants du système que le langage considère sont des composants définissant l'interface utilisateur (texte, carte...). Cette restriction a été adoptée afin que l'interaction soit décrite selon une approche visuelle dans laquelle le concepteur décrit ce qui se passe pour chaque composant d'interface lorsque l'utilisateur interagit avec l'application. En ce sens nous rejoignons et adhérons aux idées présentées dans [Hennicker and Koch 2001] où l'interaction est décrite à partir des composants visuels constituant l'interface utilisateur.
– Les annotations manipulées au cours de l'interaction doivent être représentées et liées aux composants d'interface sur lesquels elles apparaissent en début d'interaction ou vont apparaître en fin d'interaction.

3.2.2.2 *Composants du langage visuel.* Chaque interaction est décrite à partir d'un diagramme qui spécifie quelle est l'action utilisateur qui initie l'interaction et quelles sont les réactions du système qui en résultent. Les annotations impliquées dans l'interaction sont représentées et mises en évidence sur les diagrammes.

A. Spécification d'une action utilisateur

Comme défini précédemment, une action utilisateur peut être de deux types : une action de sélection ou bien une action de saisie.

L'action de sélection permet à l'utilisateur de sélectionner une annotation présentée sur son interface. Elle est matérialisée par un événement utilisateur (`click`, `mouseover`...) appliqué sur une annotation particulière qui est affichée sur un composant d'interface donné. Les événements déclencheurs d'une interaction sont à l'initiative de l'utilisateur et sont représentés par une flèche ayant pour origine l'utilisateur et pour étiquette le nom de l'événement déclenchant l'interaction (Figure 13). La destination de la flèche désigne d'une part l'annotation avec laquelle l'utilisateur souhaite interagir mais aussi le composant d'interface sur lequel cette annotation est affichée. L'annotation, qui dans ce cas devient interactive, est représentée sur le composant d'interface où elle est affichée.

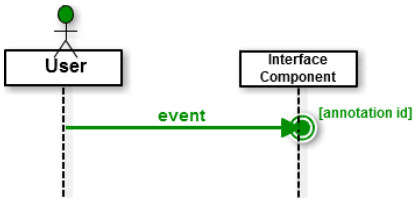

Fig. 13. Spécification d'une action de sélection de la part de l'utilisateur

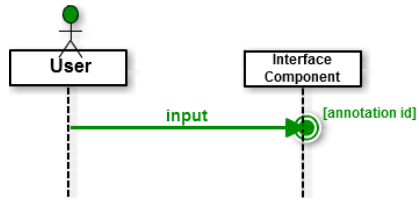

Fig. 14. Spécification d'une action de saisie de la part de l'utilisateur

L'action de saisie permet à l'utilisateur de créer une nouvelle annotation à partir d'un outil disposé dans chaque composant d'interface (par exemple, des boutons d'annotation manuelle dans un composant textuel ou des outils à dessiner des lignes, polygones sur un composant cartographique). Cette action est représentée par une flèche allant de l'utilisateur vers le composant d'interface. La destination de la flèche désigne l'annotation créée par

l'utilisateur (Figure 14). Les modalités de la saisie sont définies par le concepteur au niveau du composant d'interface (dans la phase *Interface*). Au niveau interactif, le concepteur ne se soucie plus de la manière dont l'annotation a été saisie par l'utilisateur mais uniquement des réactions du système que cette annotation saisie peut déclencher.

B. Spécification d'une réaction externe du système

Les réactions externes du système sont les réactions perceptibles par l'utilisateur. Dans notre cas, elles se traduisent par la modification visuelle d'une annotation présentée sur l'interface utilisateur. Cette modification est réalisée par le système en appliquant un ou plusieurs effets sur l'annotation à valoriser. Étant donné qu'une réaction externe provient du système et est perceptible par l'utilisateur, nous la représentons (Figure 15) par une flèche ayant pour origine le système (le composant sur lequel se trouve l'annotation) et pour destination l'utilisateur (celui qui perçoit la réaction). De manière plus précise, la flèche prend pour origine l'annotation qui doit être modifiée visuellement et l'étiquette de la flèche précise l'effet appliqué pour mettre en valeur cette annotation.

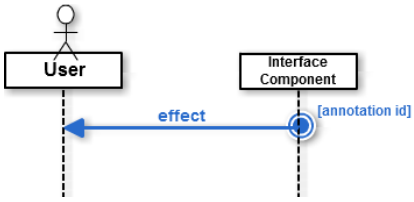

Fig. 15. Spécification d'une réaction externe du système

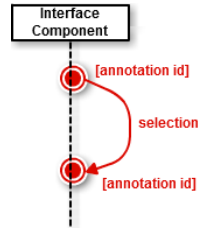

Fig. 16. Spécification de la sélection (réaction interne du système)

C. Spécification des réactions internes du système

Le modèle d'interaction proposé dans la section 3.2.1 propose trois types de réactions internes au système : la sélection, la projection et le calcul.

L'opération de sélection permet au système de déterminer quelle annotation a été sélectionnée par l'utilisateur parmi toutes les annotations affichées sur un composant d'interface. L'annotation sélectionnée par l'utilisateur devient une annotation à part entière clairement identifiée par le système et qui peut être valorisée dans la suite de l'interaction. L'opération de sélection est représentée graphiquement par une flèche ayant pour origine un ensemble d'annotations de départ parmi lesquelles l'utilisateur va faire sa sélection (Figure 16). Cet ensemble est rattaché à un composant d'interface donné. La destination de la flèche désigne l'annotation sélectionnée par l'utilisateur et identifiée par le système. La nouvelle annotation créée appartient par défaut au composant d'interface qui présente l'ensemble des annotations dans lequel la sélection a été réalisée. L'annotation créée peut ensuite être affichée via une réaction externe ou bien transférée sur un autre composant d'interface par une opération de projection par exemple.

L'opération de projection consiste à transférer une annotation présente sur un composant d'interface vers un autre composant d'interface. Dans cette opération, le système doit calculer, en cours d'interaction, la représentation de l'annotation transférée vers le composant de destination. L'origine de la flèche détermine l'annotation à projeter tandis que la destination définit le composant vers lequel l'annotation doit être projetée (Figure 17).

Soulignons que l'opération de projection n'affiche pas l'annotation sur le composant d'interface de destination. Elle détermine seulement comment cette annotation doit être représentée : la position ou la valeur de l'annotation dans un composant textuel ou encore les coordonnées géographiques de l'annotation si cette dernière est projetée sur un composant cartographique.

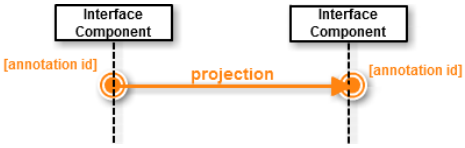

Fig. 17. Spécification de la projection (réaction interne du système)

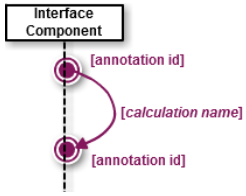

Fig. 18. Spécification du calcul (réaction interne du système)

Une opération de calcul permet de créer une nouvelle annotation à partir d'une annotation de départ sur laquelle un calcul est appliqué. Du fait que les annotations sont de nature géographique, les opérations autorisées sont elles-mêmes de nature géographique : calcul de distance, d'orientation... L'opération de calcul (Figure 18) est représentée par une flèche portant le nom du service de calcul utilisé. Cette flèche a pour origine l'annotation qui sert de point d'entrée au calcul et pour destination l'annotation créée en sortie du calcul. Cette nouvelle annotation est représentée sur le même composant d'interface que celui où se trouve l'annotation de départ.

3.2.3 *Exemple de mise en œuvre.* Le diagramme suivant (Figure 19) montre la spécification d'une interaction combinant la plupart des briques du langage visuel proposé. Il spécifie l'interaction suivante : "lorsque l'utilisateur clique (1) sur une annotation de type Ville (2) située dans le texte (3) affiché sur son interface, le système identifie (4) la ville sélectionnée (5) puis calcule (6) toutes les villes situées à moins de 20 km de la ville sélectionnée. Cet ensemble de villes (7) est ensuite transféré (8) sur le composant cartographique (9) puis mis en évidence (10) via un effet de type *highlight*".

4. ENVIRONNEMENT, OUTILS ET ÉVALUATIONS

Pour définir les caractéristiques d'une approche de conception permettant à un concepteur novice de bâtir une application géographique en toute autonomie, nous nous sommes intéressés aux approches traditionnelles issues du génie logiciel, aux paradigmes de programmation accessibles à des non-experts ainsi qu'aux outillages permettant de produire des applications sans connaissance technique spécifique. Ainsi, nous avons retenu l'intérêt et les potentialités :

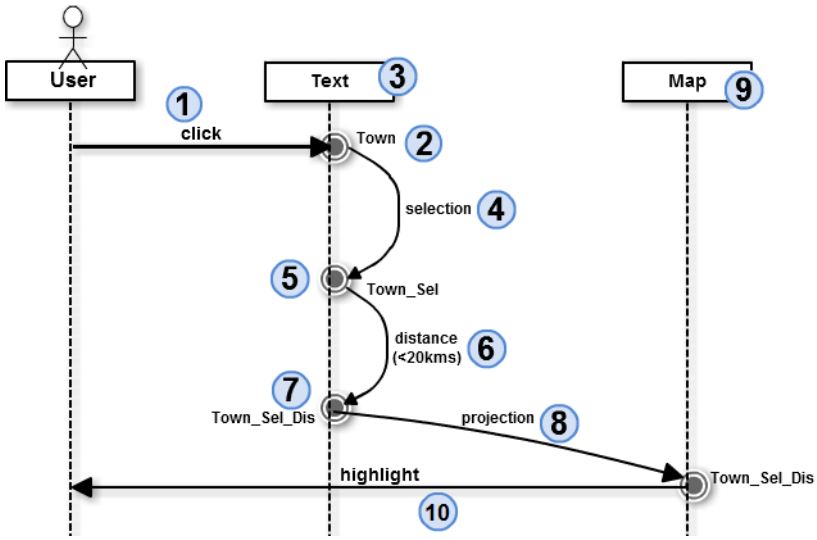

Fig. 19. Exemple d'un diagramme d'interaction

– des approches de conception "agile" qui préconisent de travailler par cycles courts en impliquant l'utilisateur final dans l'activité de conception.
– des langages visuels de programmation qui, par leur pouvoir d'expression graphique, peuvent hypothétiquement être pris en main par des non-experts.
– des outils de type Mashup qui permettent de bâtir de petites applications de manière visuelle et sans phase de programmation.

Le cahier des charges de l'environnement de conception doit avoir les propriétés suivantes :

– proposer un ensemble de services permettant de définir les informations nécessaires à la réalisation de la tâche visée par l'application. Ces informations doivent pouvoir être extraites à partir de bases de données géographiques ou de textes à caractère géographique (récits de voyage, topoguides...). Il faut également prévoir des services permettant au concepteur d'affiner chaque donnée extraite ou de combiner plusieurs données pour créer de nouvelles données utiles à la réalisation de la tâche visée.
– proposer un ensemble de composants permettant de bâtir des interfaces utilisateur capables d'afficher des contenus géographiques jouant un rôle dans la tâche à réaliser. Étant donnée la nature tripartite de ces contenus (*cf.* section 3.1.1), ces composants d'interface devront permettre de représenter aussi bien la dimension spatiale, temporelle que thématique de chaque donnée géographique manipulée par le concepteur. Ce dernier devra donc être en mesure de bâtir des interfaces utilisateur présentant des cartes (pour valoriser la dimension spatiale d'une donnée), des calendriers ou frises chronologiques (pour valoriser la dimension temporelle) et des objets textuels (pour représenter n'importe quelle dimension).
– proposer un langage visuel permettant au concepteur de préciser le détail de chaque interaction que l'utilisateur final pourra déclencher pour accomplir sa tâche.

– permettre, dans la mesure du possible, de bâtir l'application selon différents points d'entrée : définir d'abord les informations nécessaires à la réalisation de la tâche, la manière de les présenter sur une interface utilisateur finale puis les interactions associées ou bien bâtir d'abord l'interface utilisateur finale, puis les contenus que l'interface devra présenter et enfin les interactions déclenchables pour exécuter la tâche. Cette souplesse permet de prendre en compte partiellement les approches de conception empiriques que peuvent adopter des concepteurs non-informaticiens.
– permettre de générer automatiquement le code de l'application spécifiée et, si possible, quel que soit le stade d'avancement de la conception. La génération automatique du code représentera le moyen de mettre en place des cycles de conception courts et d'offrir au concepteur des retours immédiats sur ce qu'il a conçu.

Pour assurer l'exécution des modèles de *Contenu*, d'*Interface* et d'*Interaction* (présentés dans les sections 3.1.1, 3.1.2 et 3.2.1), et pour masquer la complexité de codage inhérente aux technologies à utiliser, nous avons développé une API Javascript nommée WIND[21] ("*Web INteraction Design*") [Luong et al. 2009] et l'environnement WINDMash que nous présentons ci-après.

4.1 Environnement-auteur WINDMash

Nous avons implémenté notre processus de conception dans un environnement-auteur nommé WINDMash. Quelques vidéos présentant ce prototype sont disponibles sur YouTube[22].

L'environnement propose trois modules correspondant aux trois phases du processus de conception :

(1) Un éditeur de flux qui permet de combiner différents services et de spécifier les contenus géographiques utiles à la réalisation de la tâche (phase *Contenu*);
(2) Un éditeur de présentation graphique qui est utilisé pour organiser, par exemple, des composants cartographiques ou des contenus multimédias (phase *Interface*);
(3) Un éditeur visuel inspiré par le diagramme de séquence UML qui permet de spécifier les interactions à mettre en œuvre pour que l'utilisateur final puisse réaliser sa tâche (phase *Interaction*).

La conception avec WINDMash repose sur le modèle unifié que nous avons proposé (*cf.* Figure 9) : chaque module crée des instances (sous forme de représentation RDF[23] [Manola and Miller 2004; Beckett and McBride 2004]) d'une partie du modèle. À l'issue de la conception, les instances du modèle sont finalement combinées pour générer le code exécutable de l'application finale basée sur l'API WIND.

4.1.1 *Phase Contenu.* Afin de gérer les données associées à la tâche utilisateur, nous avons développé un éditeur de flux de données. Cet outil, inspiré de l'éditeur Yahoo! Pipes[24], permet au concepteur de créer une chaîne de traitement contenant différents services (Figure 20).

[21] `http://erozate.iutbayonne.univ-pau.fr/windapi/`
[22] `https://www.youtube.com/watch?v=TTY5py1POjQ`, `https://www.youtube.com/watch?v=8NkMcZj2KxE`
[23] *Resource Description Framework*
[24] `http://pipes.yahoo.com`

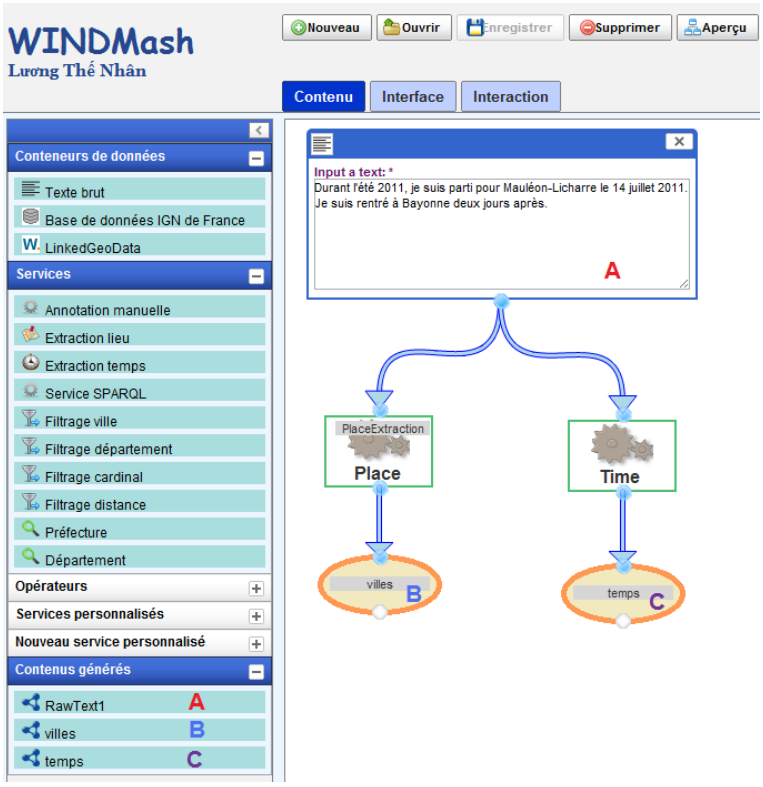

Fig. 20. Éditeur de flux de WINDMash (Phase *Contenu*)

À partir d'un ou de plusieurs textes bruts, le concepteur peut facilement créer une chaîne de traitement en sélectionnant des modules dédiés. Cette chaîne de traitement peut transformer automatiquement un texte en entrée en un document structuré selon le modèle de *Contenu* qui peut être visualisé ultérieurement (phase *Interface*) par les afficheurs (`Displayer`) dédiés : textuel, cartographique et calendaire. Les modules disponibles peuvent être paramétrés par le concepteur pour atteindre un objectif spécifique. Nous identifions deux types de modules : les *Conteneurs de données* et les *Services* décrits ci-dessous.

– Un *conteneur de données* peut être soit un texte brut, soit une base de données IGN, soit des données Web à connotation géographique (*LinkedGeoData*). Un texte brut est un document textuel (par exemple, le récit de voyage) utilisé par le concepteur et est relié à un module `Service`. Une base de données IGN comprend des tables d'informations géographiques (communes, départements, montagnes, cours d'eau...). Le module de données *LinkedGeoData* permet d'exploiter des données géographiques ouvertes sur le Web[25].

– Les modules *services* (e.g., Annotation manuelle, Extraction lieu, Extraction temps, Service SPARQL...) permettent de traiter un conteneur de données et produire des contenus

[25] `http://linkedgeodata.org`

générés (données structurés selon le modèle *Contenu* présenté dans la section 3.1.1). Un module *Extraction lieu* implémente le service Web GeoStream [Sallaberry et al. 2009] pour extraire les informations géographiques dans les documents textuels et un module *Extraction temps* implémente le service Web TempoStream [Loustau et al. 2009] pour extraire les informations temporelles dans les documents textuels. Un module *Annotation manuelle* permet à l'utilisateur d'annoter par lui-même les entités sur un texte selon trois facettes : spatiale, temporelle et thématique. Le *service SPARQL* est utilisé avec des données Web sémantiquement définies (*LinkedGeoData*) pour extraire des entités géographiques en utilisant une requête SPARQL[26] permettant d'inférer des résultats ayant du sens pour l'utilisateur.

La figure 20 illustre un exemple de chaîne de traitements : à partir d'un texte donné (A), nous tenons à extraire automatiquement les lieux (B) (en occurrence des "`villes`") et les références temporelles (C) (nommées "`temps`"). Pour cet objectif, nous appelons le service Web `Extraction lieu` qui a été détaillé dans [Sallaberry et al. 2009] et le service Web `Extraction temps`. De plus, le service Web `Extraction lieu` peut identifier et marquer les types des entités extraites ; dans ce cas les entités "*Mauléon-Licharre*" et "*Bayonne*" sont identifiées comme des villes (`Town`).

Une fois la construction des flux réalisée, il est possible de visualiser à tout moment les données calculées en sélectionnant par double clic les éléments "`villes`" (B) et "`temps`" (C).

Chaque fois qu'un ensemble de données est calculé tel que la liste des lieux extraits (B), WINDMash génère une description RDF/XML[27] qui correspond à la phase de *Contenu*. Notons que ces descriptions intègrent des concepts sémantiquement définis qui peuvent être exploités dans notre environnement par des opérateurs, par exemple, d'inférence ou de filtrage sémantique. Ces descriptions sont accessibles en bas à gauche du prototype WINDMash dans la zone des *Contenus générés* (Figure 20).

Nous montrons ci-après que ces descriptions peuvent être utilisées dans notre éditeur de présentation graphique pour afficher les données à l'intérieur des afficheurs.

4.1.2 *Phase Interface.* Notre éditeur de présentation graphique permet à un concepteur de spécifier l'interface qui sera proposée à l'utilisateur final pour accomplir sa tâche. Le concepteur décide quel type d'afficheur il souhaite dans son application (par exemple, `Afficheur texte`, `Afficheur carte`, `Afficheur liste`, `Afficheur frise`) et comment ces afficheurs sont organisés à l'intérieur de la présentation graphique (taille et position).

La figure 21 illustre comment un concepteur peut préciser, grâce à notre outil, la présentation graphique de son application. Le menu à gauche indique le type des afficheurs qui peuvent être manipulés par le concepteur, l'ensemble des données disponibles qui ont été calculées avec notre éditeur de flux (*cf.* section 4.1.1) et les afficheurs qui sont actuellement utilisés. Ici aussi, les différents afficheurs et contenus générés (issus de la phase *Contenu*) sont placés dans la zone de travail par glisser-déposer du concepteur. Ce dernier choisit la taille et la position de chaque afficheur qu'il peut également paramétrer en précisant notamment le nom qui s'affichera aussi dans la zone des *Afficheurs générés* (et sera utilisé ultérieurement dans la phase *Interaction* - *cf.* Section 4.1.3). Dans la figure 21, trois af-

[26] `http://www.w3.org/TR/sparql11-query/`
[27] `http://www.w3.org/TR/REC-rdf-syntax/`

ficheurs ont été spécifiés : un afficheur textuel (1), un afficheur cartographique (2) et un afficheur de frise chronologique (3).

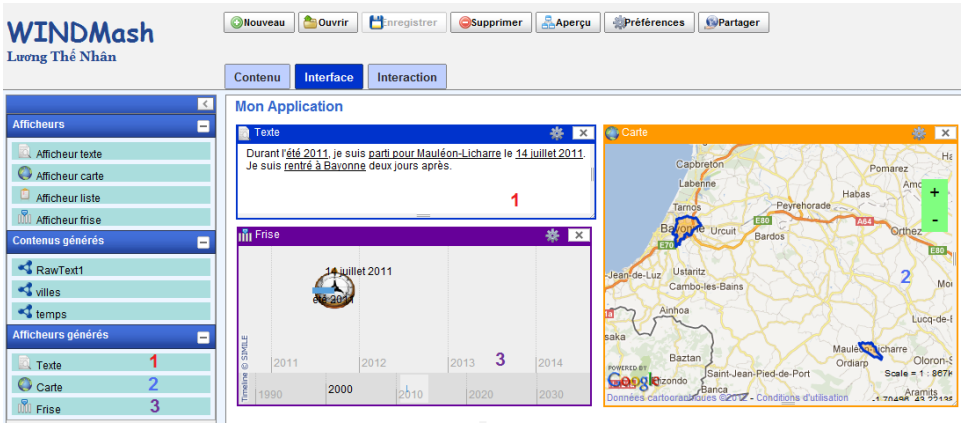

Fig. 21. Éditeur de la mise en page graphique de WINDMash (Phase *Interface*)

Initialement, lorsque le concepteur glisse et dépose un afficheur à l'intérieur de la zone de travail, cet afficheur est vide, à l'exception de l'afficheur cartographique qui contient une carte. Si le concepteur souhaite afficher quelques informations (au lancement de l'application) à l'intérieur des afficheurs, à partir du menu il doit glisser les données calculées et les déposer dans un afficheur spécifique. Par exemple, si le concepteur veut voir à l'intérieur de l'afficheur textuel (1) le texte qui a été écrit dans la figure 20, il doit glisser l'élément `RawText1` du menu et déposer cet élément à l'intérieur de l'afficheur `Texte`. Par la suite, si le concepteur veut souligner les lieux et les temps extraits de ce texte, il doit glisser les éléments `villes` et `temps` du menu et le déposer dans l'afficheur `Texte`.

De la même manière avec les autres types d'afficheurs, si le concepteur veut voir les lieux sur la carte, il doit glisser l'élément `villes` et le déposer dans l'afficheur `Carte`. Enfin, si le concepteur veut voir les temps sur la frise chronologique, il doit glisser l'élément `temps` et le déposer à l'intérieur de l'afficheur `Frise`.

Jusqu'à cette étape, le concepteur a utilisé WINDMash pour générer automatiquement deux instances de deux modèles de *Contenu* et d'*Interface*. Il peut prévisualiser l'application Web géographique (statique) en cliquant sur le bouton `Aperçu` dans la barre de menu (en haut) de notre prototype. Il peut également sauvegarder et modifier l'application quand il le souhaite. L'application générée est dite "statique" car elle ne contient aucun comportement dynamique sensible aux actions de l'utilisateur, telles que le clic ou le survol. Toutefois, nous allons prendre en compte cet aspect dynamique dans la section suivante (section 4.1.3).

4.1.3 *Phase Interaction.* Pour faciliter l'implémentation des interactions nécessaires à la réalisation de la tâche visée, nous avons intégré dans l'environnement WINDMash le langage visuel proposé dans la section 3.2.2. L'exemple traité (Figure 22) est le suivant : "*lorsque l'utilisateur clique sur un lieu dans le texte, le système identifie le lieu sélectionné,*

le surligne dans le texte, puis envoie ce lieu sur la carte afin de faire un zoom avant à cet endroit."

L'espace de conception est divisé en deux zones :

– une zone à gauche qui présente dans sa partie basse l'ensemble des composants d'interface sur lesquels le concepteur va pouvoir définir des interactions à partir de cinq blocs disponibles (ceux présentés en section 3.2.2). Cette zone présente aussi dans sa partie haute les différents composants d'interface (les afficheurs) qui ont été créés/paramétrés dans la phase Interface ;

– une zone de travail centrale dans laquelle le concepteur va spécifier chacune de ses interactions via le langage visuel que nous avons proposé.

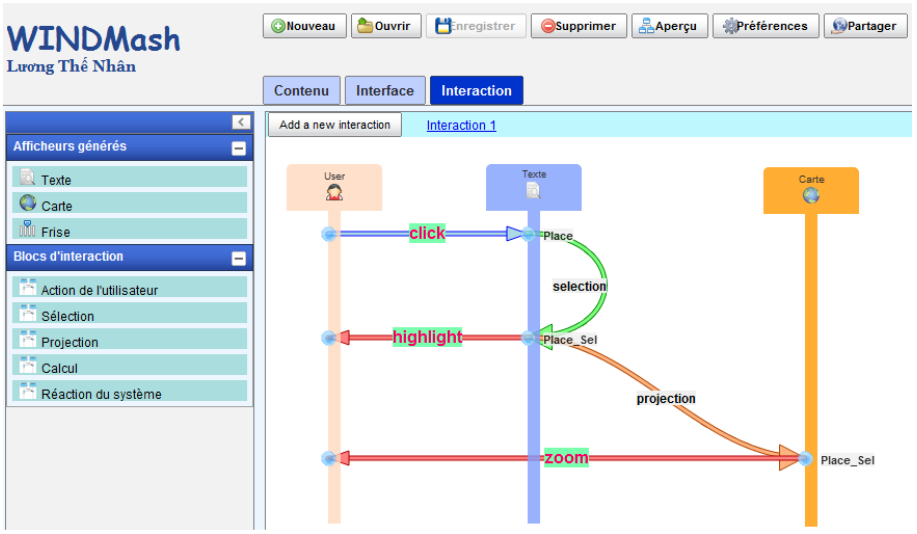

Fig. 22. Spécification graphique d'une interaction dans WINDMash

Comme pour les autres espaces de travail de WINDMash, la spécification d'une interaction est réalisée graphiquement par des opérations de glisser-déposer visant à assembler des briques de notre langage visuel. Lors de la spécification d'une action utilisateur, d'une réaction interne ou externe sur un composant d'interface, l'environnement présente au concepteur la liste des annotations présentes sur ce composant[28] et demande à l'utilisateur quelles sont les annotations qu'il souhaite impliquer dans l'interaction en cours de définition. Selon les spécifications de l'interaction qu'il veut décrire avec le langage visuel, le concepteur fait successivement glisser-déposer des blocs disponibles dans le menu de gauche et renseigne leurs paramètres.

Comme dans les étapes de conception précédentes, la spécification graphique d'une interaction est ensuite traduite au format RDF. Rappelons qu'il existe également, suite au résultat de la phase *Interface* une description RDF précisant les composants d'interface

[28] Ceci constitue un premier niveau d'assistance au concepteur.

(afficheurs textuels, cartographiques ou calendaires) utilisés. De même, les contenus géographiques manipulés ont été définis dans la phase *Contenu* et ont donné lieu à une autre description RDF qu'il est possible d'utiliser en phase *Interface* pour lier ces contenus et les afficheurs.

Par conséquent, la description RDF de l'interaction correspond à un cahier des charges structuré de l'interaction que le système doit rendre exécutable. L'opérationnalisation de chaque interaction est ensuite réalisée en parcourant les différents fichiers RDF générés puis en générant les objets JavaScript correspondants via l'API WIND. Le code JavaScript généré qui est interprété par un navigateur Web devient alors très synthétique. Cette génération automatique de code permet ainsi au concepteur d'exécuter son application, d'avoir un retour immédiat sur les interactions spécifiées et, en cas d'insatisfaction, de pouvoir revenir en phase de conception pour modifier ses diagrammes d'interaction.

Dans la section suivante 4.2, nous proposons d'évaluer les usages de nos outils : l'environnement-auteur WINDMash basé sur notre API WIND.

4.2 Evaluations

Nous avons conduit un ensemble d'évaluations portant sur trois points :

(1) Aptitude du modèle et du langage visuel pour spécifier et implémenter des interactions portant sur des contenus géographiques.
(2) Capacité de concevoir et mettre en œuvre facilement des applications géographiques répondant aux besoins d'enseignants.
(3) Capacité pour des non-informaticiens de concevoir des applications géographiques en toute autonomie.

Chacune de ces trois évaluations a reposé sur un protocole de test en plusieurs étapes reposant sur des critères d'évaluation pré-établis [Luong 2012].

La première évaluation, publiée dans la Conférence IUI 2012 [Luong et al. 2012], a porté sur l'évaluation de notre langage visuel et de son modèle sous-jacent permettant aux concepteurs de concevoir facilement le comportement interactif d'une application Web géographique. Suite à cette évaluation nous avons pu mettre en évidence deux principaux atouts du langage :

– la facilité d'usage du langage (car composé de cinq briques simples) ;
– la flexibilité du langage (possibilité de décrire une interaction complexe dans un seul diagramme ou en la décomposant sur plusieurs diagrammes).

La deuxième évaluation portait sur le processus global de conception pour des usages pertinents à l'école élémentaire. Dans ce dernier cas, pour son mémoire de Master 2, [Paillas 2011] avait proposé un scénario complexe mêlant interactions simples et annotations de corrélation. Avec cette évaluation, nous avons établi que WINDMash permettait de concevoir et implémenter ce scénario de façon complète [Luong 2012].

Pour la dernière évaluation portant sur le processus complet de conception, nous avons sollicité treize étudiants en reconversion professionnelle préparant un Diplôme d'Université dans le domaine des Technologies de l'Internet (DU TIC[29]). Aucun d'eux n'était informaticien mais tous étaient des utilisateurs réguliers de l'outil informatique ayant un profil initial en communication, infographie ou audiovisuel. L'évaluation est présentée ci-dessous.

[29] www.iutbayonne.univ-pau.fr/diplomes-universitaires/tic/

4.2.1 *Description du protocole expérimental.* Le travail consistait à construire une application Web géographique en utilisant notre environnement WINDMash. La procédure d'évaluation a été organisée en trois étapes. Durant 45 minutes, nous avons effectué une présentation générale de l'environnement WINDMash et montré une vidéo guidant la conception d'application (étape 1). Les étudiants ont travaillé ensuite en autonomie et à distance pour construire une application décrite ci-après (étape 2). Ils ont enfin rempli un questionnaire d'enquête (étape 3).

Supposons qu'un enseignant de cycle 3 (CM1-CM2) souhaite élaborer une application dédiée à une activité pédagogique sollicitant des connaissances sur les villes et départements français. Il utilise des références du Tour de France 2012 pour construire une application présentant un texte décrivant une partie de la course et affichant les étapes sur une carte et les dates correspondantes sur une frise chronologique. Cette application a deux buts principaux. Elle doit introduire la notion de département et améliorer leurs connaissances en géographie sur les principales villes et départements traversés durant la course. Le texte qui a servi à cette évaluation est le suivant :

"*Pendant l'été 2012, le Tour de France débute en Belgique le 30 juin 2012 et passe par les Vosges et le Jura. Il fait la part belle à la moyenne montagne et aux contre-la-montre, avant l'arrivée à Paris le 22 juillet 2012. Il comporte trois arrivées en montagne et deux contre-la-montre avec une étape de Arc-et-Senans à Besançon (38 km) et une autre plus longue de Bonneval à Chartres (52 km).*".

Ce texte est intéressant car il comporte à la fois des références spatiales et temporelles. De plus, les références spatiales concernent non seulement des villes mais aussi des départements, certains explicitement et d'autres devant être calculés à partir des villes citées. L'application visée de la figure 1 est accessible en ligne[30] et dispose des cinq zones suivantes d'affichage :

(1) Une zone en haut à gauche contenant le texte initial ;
(2) Un afficheur cartographique à droite pour présenter non seulement les départements (mis en surbrillance) mais aussi pour zoomer en avant sur les villes étapes du département ;
(3) Un afficheur de type texte situé en haut au centre pour écrire le nom des départements qui sont soit cités dans le texte soit calculés à partir des villes citées ;
(4) Un petit afficheur cartographique situé au centre et permettant de zoomer sur les villes citées dans le texte ;
(5) Une frise chronologique située en bas à gauche pour afficher les dates et périodes citées dans le texte.

Le comportement de l'application est le suivant :

– Quand un utilisateur clique sur une ville dans le texte (afficheur 1), le système doit automatiquement mettre en surbrillance le nom du département dans l'afficheur textuel 3 et dans la carte principale (afficheur 2). Le système doit aussi zoomer sur la ville dans l'afficheur cartographique 4.
– Quand un utilisateur clique sur un nom de département dans le texte principal (afficheur 1), ce nom doit s'afficher dans le petit afficheur textuel 3.

[30] `http://erozate.iutbayonne.univ-pau.fr/Nhan/windmash/demo/tourdefrance/`

– Quand un utilisateur clique sur un nom de département de cet afficheur 3, la carte principale (afficheur 2) doit zoomer sur ce département.

Les participants ont disposé de 20 minutes pour la phase *Contenu*, de 15 minutes pour la phase *Interface*, de 30 minutes pour la phase *Interaction*. À la fin de la conception / réalisation, nous avons posé aux participants 35 questions dont 28 QCM à réponse unique et 7 questions à réponse ouverte courte. Les questionnaires que les étudiants ont du remplir étaient accessibles sur l'espace numérique de travail de notre Université[31].

Nous avons divisé l'expérimentation en deux parties : réaliser une application Web géographique statique (phases *Contenu* et *Interface*) puis dynamique (phase *Interaction*). Une application statique contient des composants d'interface affichant seulement des contenus sans aucune possibilité d'interaction ; une application dynamique ajoute des comportements interactifs entre les composants d'interface. Pour chaque partie, nous présentons ci-dessous un tableau des questions ainsi que la synthèse des réponses des participants.

4.2.2 *Résultats et discussion.* Pour réaliser une application web géographique statique (sans Interaction), nous avons posé des questions relatives à la manipulation des contenus, d'autres relatives à l'interface utilisateur puis présenté le questionnaire de synthèse suivant (Table II).

	Questions	Type
1	Si vous aviez à développer une autre application Web géographique statique, souhaitez-vous réutiliser cet outil ?	Oui / Non
2	Connaissez-vous d'autres outils permettant de développer des applications Web géographiques ?	Oui / Non
2bis	Si oui à la question précédente, précisez	Ouverte
3	Donnez une note /20 à l'outil (en terme d'utilisabilité) de manière générale (pour développer une application Web géographique statique).	Score entre 1 et 20

Table II. Questionnaire d'évaluation de la synthèse des phases *Contenu* et *Interface*

Voici une synthèse des réponses :

(1) Tous les participants souhaitent réutiliser l'environnement WINDMash pour développer une application Web géographique statique. L'outil est donc fortement apprécié.
(2) 92% des participants ont répondu qu'ils ne connaissent pas d'autres outils permettant de développer des applications Web géographiques statiques (un participant n'a pas répondu). Pour eux, WINDMash est original.
(3) La note moyenne obtenue était environ de 14 sur 20. Nous en concluons que WINDMash est utilisable par des utilisateurs non-informaticiens pour créer des applications Web géographiques statiques (sans interaction).

Après l'ajout des comportements interactifs à cette application géographique, nous avons proposé aux participants de répondre aux questions suivantes (Table III).

Voici une synthèse des réponses :

(1) 75% des participants ont répondu que le processus actuel en 3 phases (*Contenu* <-> *Interface* <-> *Interaction*) facilite l'amélioration progressive de la conception effectuée de l'application.

[31] `https://webcampus.univ-pau.fr/courses/EVALUATIONWINDMASH/`

	Questions	Type
1	Le processus actuel en 3 phases (*Contenu* <-> *Interface* <-> *Interaction*) facilite-t-il l'amélioration progressive de la conception effectuée de l'application ?	Oui / Non
2	Ce processus pour lier les objets d'une phase à l'autre vous a-t-il paru naturel ou pas ?	Tout à fait, Plutôt Oui, Plutôt Non, Pas du tout
3	Imagineriez-vous d'autres moyens de lier les différentes phases et d'autres chronologies d'enchaînement des phases ?	Ouverte
4	Connaissez-vous d'autres outils permettant de développer des applications Web géographiques dynamiques ?	Oui / Non
4bis	Si oui à la question précédente, précisez	Ouverte
5	Donnez une note /20 (en terme d'utilisabilité) à WINDMash de manière générale.	Score entre 1 et 20

Table III. Questionnaire d'évaluation de la synthèse finale

(2) 50% ont trouvé le processus naturel pour lier les objets d'une phase à l'autre.

(3) Un seul participant a proposé de combiner les phases *Interface* et *Interaction*.

(4) 100% des participants ont répondu qu'ils ne connaissent pas d'autres outils permettant de développer des applications Web géographiques dynamiques.

(5) Seuls 7 participants ont fourni une note qui était comprise entre 10 et 16 sur 20. De manière générale, nous constatons donc que WINDMash sera utilisable pour les utilisateurs non-informaticiens après correction des bogues identifiés.

Au-delà de l'environnement WINDMash, nous avons essayé de cerner la diversité des applications qui pouvaient être développées avec notre approche de conception. L'ensemble des modèles de conception présentés dans les sections 3.1.1, 3.1.2 et 3.2.1 sont rendus opérationnels via une API Javascript nommée WIND ("*Web INteraction Design*")[32] qui permet de retranscrire chaque instance de modèle sous forme de code exécutable. Nous avons mis à disposition cette API pour divers projets de développement menés avec des partenaires et nous avons pu constater que cette API était capable de répondre à des besoins variés qui vont au-delà des applications présentées précédemment[33].

5. BILAN ET PERSPECTIVES

Dans cet article, nous avons présenté un nouveau cadre de conception d'applications Web géographiques interactives. Afin de faciliter le processus de conception, nous avons proposé des représentations visuelles qui s'appuient sur un modèle générique profond suffisamment riche pour couvrir une large palette d'applications interactives. Actuellement, ce modèle de conception s'articule autour de trois facettes fondamentales de conception : (1) les contenus à mettre en valeur, (2) la mise en place de l'interface et (3) la spécification des interactions.

Ce cadre permet à un concepteur d'exprimer la manière dont il souhaite réaliser une tâche spécifique à l'aide d'une application. Ce travail de conception est réalisé en enchaînant des phases de spécification, d'exécution et d'évaluation. La tâche en elle-même n'est pas expressément formalisée. En définitive, le concepteur dispose d'une application dédiée à une tâche spécifique mais le but de cette tâche et les éventuelles sous-tâches à accomplir pour atteindre ce but n'ont pas été clairement exprimés. L'analyse de l'application finale permet néanmoins d'identifier en partie les informations qui entrent en jeu dans la réalisation de la tâche ainsi que les fonctionnalités déclenchables par l'utilisateur pour exécuter

[32] `http://erozate.iutbayonne.univ-pau.fr/windapi/`

[33] Aperçu des applications consultable sur `http://erozate.iutbayonne.univ-pau.fr/Nhan/`

la tâche.

Lorsque le concepteur est aussi l'utilisateur final, il spécifie les informations qui lui sont nécessaires pour réaliser la tâche, la manière dont ces informations seront présentées sur l'interface utilisateur finale ainsi que les fonctionnalités interactives dont il souhaite disposer pour réaliser la tâche. Dans ce contexte particulier, nous nous situons à mi-chemin entre la modélisation de la tâche et la spécification de l'activité étant donné que les caractéristiques de l'application sont pensées par le concepteur-utilisateur en fonction de la manière dont il envisage de mener concrètement son travail.

Nous avons montré l'intérêt de l'environnement-auteur WINDMash pour concevoir rapidement une application par prototypage et nous avons évalué l'utilisabilité de cet outil par des concepteurs non-informaticiens. Finalement, ces concepteurs non-spécialistes en informatique ont pu produire (sans connaissance en termes de modélisation de tâches), une application au service d'une tâche unique. Ces expérimentations avec des acteurs de terrain nous amènent à établir les potentialités de notre contribution en tant qu'outil d'aide à l'analyse et à la modélisation de tâches.

Traditionnellement, la construction d'un modèle de tâches est réalisée par un ingénieur cogniticien. Cette construction passe par une analyse des besoins des utilisateurs pour identifier ce que les utilisateurs veulent faire avec le logiciel. Généralement, cette analyse est menée de la manière suivante. Il y a tout d'abord des interviews permettant d'identifier la tâche prévue (ce qui doit être fait) puis des séances d'observation des utilisateurs en situation réelle pour formaliser l'activité effectivement réalisée. Les résultats de ces analyses prennent ensuite la forme d'un graphe représentant le modèle de tâches.

Nous avons montré qu'un outil de prototypage rapide tel que WINDMash contribue à cette phase d'analyse en permettant aux utilisateurs finaux d'exprimer eux-mêmes la manière dont ils aimeraient réaliser les tâches à l'aide d'une application. Cette approche ne vise pas à s'opposer aux méthodes traditionnelles d'analyse mais à ajouter une plus-value dans un processus d'analyse long et complexe qui vise, finalement, à bâtir une application au service d'un utilisateur. Dans ce contexte, nous pensons qu'un expert pourrait exploiter les résultats de l'activité de prototypage et en déduire des éléments d'analyse relatifs à plusieurs niveaux de modélisation.

Dans le cadre de l'ingénierie des interactions homme-machine, nous pensons qu'en plaçant l'utilisateur final en situation de commenter son activité de conception face à un expert, il devient possible pour ce dernier de déduire des éléments d'analyse relatifs :

– au modèle des tâches : le fait de permettre à l'utilisateur d'expliquer la présence d'une information ou de justifier le pourquoi d'une interaction contribue à faciliter la compréhension de la tâche menée.
– au modèle de dialogue : les interactions imaginées et implémentées par l'utilisateur décrivent en partie la manière dont ce dernier souhaite échanger / dialoguer avec son application.
– au modèle de présentation : les composants d'interface retenus par l'utilisateur mais aussi sa manière de les agencer peuvent rendre compte de propriétés de présentation incontournables à intégrer sur la couche visuelle de l'application finale.

Le travail de l'expert pourrait être appuyé / complété par des mécanismes d'analyse semi-automatique de l'application conçue. Actuellement, chaque choix de conception de l'utilisateur se traduit par une instanciation de nos modèles de conception. Chacune de ces

instanciations prend la forme d'une description RDF et, finalement, l'ensemble des descriptions RDF produites constitue un cahier des charges structuré servant de point d'entrée pour la génération du code final de l'application. Cette représentation RDF présente également un intérêt pour analyser et extraire automatiquement les éléments relatifs aux modèles des tâches, de dialogue et de présentation cités ci-dessus. Grâce aux technologies liées au formalisme RDF, le système de requêtage reposant sur la sémantique des données permet, par exemple, d'extraire facilement l'ensemble des informations ajoutées par l'utilisateur à l'application dans le but de mener sa tâche.

L'analyse des interactions implémentées par l'utilisateur peut également être un point de départ intéressant pour essayer d'analyser la structure de la tâche et éventuellement sa décomposition en sous-tâches. Les informations initialement placées sur l'interface, mais aussi les interactions associées, sont censées jouer un rôle pour débuter la réalisation de la tâche. Suite à une interaction, de nouvelles informations peuvent être calculées et présentées, et de nouvelles interactions peuvent être attachées à ces informations. Il devient donc possible d'établir un graphe de dépendances temporelles entre l'ensemble des interactions spécifiées par l'utilisateur (une interaction ne pouvant être déclenchée sur une information que lorsque celle-ci a été précédemment calculée et affichée). Ce graphe de dépendances temporelles pourrait être complété / consolidé en observant et en traçant l'activité de l'utilisateur en phase d'usage de l'application générée. Bien entendu, un tel graphe reste encore trop pauvre pour identifier automatiquement un "pack" d'informations / interactions pouvant désigner une tâche ou une sous-tâche... Néanmoins ce graphe pourrait fournir des informations significatives à un expert soucieux d'avoir un retour sur le modèle de tâches déduit selon une approche traditionnelle (interviews et observations des utilisateurs en situation d'usage).

Dans nos futurs travaux, nous approfondirons cette piste en identifiant, de manière plus formelle, les liens pouvant être établis / inférés entre les éléments de notre modèle de conception et les éléments des modèles de tâches classiques (CTT par exemple). Ceci nous amènera à utiliser de manière plus systématique des moteurs de transformation de modèles (du modèle WIND vers CTT par exemple).

RÉFÉRENCES

ABRAMS, M., PHANOURIOU, C., BATONGBACAL, A. L., WILLIAMS, S. M., AND SHUSTER, J. E. 1999. UIML: an appliance-independent XML user interface language. *31*, 1695–1708.

ALBINOLA, M., BARESI, L., CARCANO, M., AND GUINEA, S. 2009. Mashlight: a lightweight mashup framework for everyone. In *2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web*.

ALTINEL, M., BROWN, P., CLINE, S., KARTHA, R., LOUIE, E., MARKL, V., MAU, L., NG, Y.-H., SIMMEN, D., AND SINGH, A. 2007. Damia - a data mashup fabric for intranet applications. In *Proceedings of the 33rd International Conference on Very Large Data Bases*. 1370 – 1373.

BARBIER, F. 2008. Supporting the UML state machine diagrams at runtime. In *Model Driven Architecture Foundations and Applications*, I. Schieferdecker and A. Hartman, Eds. Lecture Notes in Computer Science, vol. 5095. Springer Berlin / Heidelberg, 338–348.

BASS, L., FANEUF, R., LITTLE, R., MAYER, N., PELLEGRINO, B., REED, S., SEACORD, R., SHEPPARD, S., AND SZCZUR, M. R. 1992. A metamodel for the runtime architecture of an interactive system: the UIMS tool developers workshop. *SIGCHI Bull. 24,* 1 (Jan.), 32–37.

BECKETT, D. AND MCBRIDE, B. 2004. RDF/XML Syntax Specification (Revised). Recommendation, W3C. February. `http://www.w3.org/TR/rdf-syntax-grammar/`.

BOLIN, M., WEBBER, M., RHA, P., WILSON, T., AND MILLER, R. C. 2005. Automation and customization of rendered web pages. In *Proceedings of the ACM Conference on User Interface Software and Technology (UIST)*. 163 – 172.

BRITTS, S. 1987. Dialog management in interactive systems: a comparative survey. *SIGCHI Bull. 18,* 3 (Jan.), 30–42.

CAFFIAU, S., GIRARD, P., GUITTET, L., AND SCAPIN, D. 2009. Hierarchical structure: A step for jointly designing interactive software dialog and task model. In *Proceedings of the 13th International Conference on HCI*. Springer-Verlag, 664–673.

CALVARY, G., COUTAZ, J., THEVENIN, D., LIMBOURG, Q., BOUILLON, L., AND VANDERDONCKT, J. 2003. A unifying reference framework for multi-target user interfaces. *Interacting with Computers 15,* 3, 289–308.

CARD, S. K., MORAN, T. P., AND NEWELL, A. 1983. *The Psychology of Human-Computer Interaction*, 1st ed. CRC Press.

CHURCHER, G. E., ATWELL, E. S., AND SOUTER, C. 1997. Dialogue management systems: a survey and overview. Tech. rep., School of Computer Science, University of Leeds.

CORREANI, F., MORI, G., AND PATERNÒ, F. 2005. Supporting flexible development of multi-device interfaces. In *Proceedings of the 2004 International Conference on Engineering Human Computer Interaction and Interactive Systems*. EHCI-DSVIS'04. Springer-Verlag, Berlin, Heidelberg, 346–362.

COUTAZ, J. 1987. PAC, an object oriented model for dialog design. In *Interact'87*. 431–436.

DIAPER, D. AND STANTON, N. 2004. *The Handbook of Task Analysis for Human-Computer Interaction*. Lawrence Erlbaum.

DRAGICEVIC, P. AND FEKETE, J.-D. 2004. Support for input adaptability in the icon toolkit. In *Proceedings of the 6th international conference on Multimodal interfaces*. ICMI '04. ACM, New York, NY, USA, 212–219.

ENNALS, R. AND GAROFALAKIS, M. 2007. Mashmaker: Mashups for the masses. In *Proceedings of the 27th ACM SIGMOD International Conference on Management of Data*. 1116 – 1118.

GAIO, M., SALLABERRY, C., ETCHEVERRY, P., MARQUESUZAÀ, C., AND LESBEGUERIES, J. 2008. A global process to access documents' contents from a geographical point of view. *Journal of Visual Languages and Computing 19*, 3–23.

GAMBOA, F. AND SCAPIN, D. 1997. Editing MAD* task descriptions for specifying user interfaces, at both semantic and presentation levels. In *Proceedings of Fourth Int. Workshop on Design, Specification, and Verification of Interactive Systems*. 193–208.

GIESE, M., MISTRZYK, T., PFAU, A., SZWILLUS, G., AND DETTEN, M. 2008. Amboss: A task modeling approach for safety-critical systems. In *Proceedings of the 2nd Conference on Human-Centered Software Engineering and 7th International Workshop on Task Models and Diagrams*. Springer-Verlag, 98–109.

GREEN, M. 1986. A survey of three dialogue models. *ACM Trans. Graph. 5*, 244–275.

GUERRERO-GARCIA, J., GONZALEZ-CALLEROS, J. M., VANDERDONCKT, J., AND MUNOZ-ARTEAGA, J. 2009. A theoretical survey of user interface description languages: Preliminary results. In *Proceedings of the 2009 Latin American Web Congress*. LA-WEB '09. IEEE Computer Society, 36–43.

HAREL, D. AND MARELLY, R. 2003. *Come, Let's Play: Scenario-Based Programming Using LSC's and the Play-Engine*. Springer-Verlag New York, Inc.

HENNICKER, R. AND KOCH, N. 2001. Modeling the user interface of web applications with UML. In *Workshop of the pUML-Group held together with the "UML"2001 on Practical UML-Based Rigorous Development Methods - Countering or Integrating the eXtremists*. 158–172.

HEWETT, BAECKER, CARD, CAREY, GASEN, MANTEI, PERLMAN, STRONG, AND VERPLANK. 1992. *ACM SIGCHI Curricula for Human-Computer Interaction*. Association for Computing Machinery.

HIX, D. AND HARTSON, H. R. 1993. *Developing User Interfaces: Ensuring Usability Through Product & Process*. John Wiley & Sons.

HUYNH, D. F., KARGER, D. R., AND MILLER, R. C. 2007. Exhibit: Lightweight structured data publishing. In *Proceedings of the 16th Int. World Wide Web Conference*. 737 – 746.

HUYNH, D. F., MAZZOCCHI, S., AND KARGER, D. R. 2005. Piggy bank: Experience the semantic web inside your web browser. In *Proceedings of the International Semantic Web Conference (ISWC)*. 413 – 430.

KO, A. J., ABRAHAM, R., BECKWITH, L., BLACKWELL, A., BURNETT, M., ERWIG, M., SCAFFIDI, C., LAWRANCE, J., LIEBERMAN, H., MYERS, B., ROSSON, M. B., ROTHERMEL, G., SHAW, M., AND

WIEDENBECK, S. 2011. The state of the art in end-user software engineering. *ACM Comput. Surv. 43,* 3 (Apr.), 21:1–21:44.

KOEGEL, J. F. AND HEINES, J. M. 1993. Improving visual programming languages for multimedia authoring. In *ED-MEDIA '93, World Conference on Educational Multimedia and Hypermedia*. 286–293.

KOHLER, H.-J., NICKEL, U., NIEREAND, J., AND ZANDORF, A. 1999. Using UML as visual programming language. *Technical Report tr-ri-99-205*.

KOLSKI, C. 2001. *Analyse et conception de l'IHM : Tome 1, Interaction homme- machine pour les SI*. Hermès.

KÖNIG, W. A., RÄDLE, R., AND REITERER, H. 2010. Interactive design of multimodal user interfaces - reducing technical and visual complexity. *Journal on Multimodal User Interfaces 3,* 3 (Feb), 197–213.

KRASNER, G. E. AND POPE, S. T. 1988. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program. 1*, 26–49.

LIMBOURG, Q., PRIBEANU, C., AND VANDERDONCKT, J. 2001. Towards uniformed task models in a Model-Based approach. In *Proceedings of the 8th International Workshop on Interactive Systems: Design, Specification, and Verification-Revised Papers*. DSV-IS '01. Springer-Verlag, London, UK, 164–182.

LIMBOURG, Q. AND VANDERDONCKT, J. 2004. USIXML: A user interface description language supporting multiple levels of independence. In *ICWE Workshops*. 325–338.

LOUSTAU, P., NODENOT, T., AND GAIO, M. 2009. Design principles and first educational experiments of PIIR, a platform to infer geo-referenced itineraries from travel stories. *International Journal of Interactive Technology and Smart Education 6*, 23 – 29.

LUONG, T. N. 2012. Modélisation centrée utilisateur final appliquée à la conception d'applications interactives en géographie : une démarche basée sur les contenus et les usages. Ph.D. thesis, Université de Pau et des Pays de l'Adour.

LUONG, T. N., ETCHEVERRY, P., AND MARQUESUZAÀ, C. 2011. An interaction model and a framework dedicated to web-based geographic applications. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*. MEDES '11. ACM, New York, NY, USA, 235–242.

LUONG, T. N., ETCHEVERRY, P., MARQUESUZAÀ, C., AND NODENOT, T. 2012. A visual programming language for designing interactions embedded in web-based geographic applications. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*. IUI '12. ACM, 207–216.

LUONG, T. N., ETCHEVERRY, P., NODENOT, T., AND MARQUESUZAÀ, C. 2009. WIND: an interaction lightweight programming model for geographical web applications. In *Geospatial Free and Open Source Software in the 21st Century*, E. Bocher and M. Neteler, Eds. Lecture Notes in Geoinformation and Cartography. Springer Berlin Heidelberg, 211–225.

LUONG, T. N., ETCHEVERRY, P., NODENOT, T., MARQUESUZAÀ, C., AND LOPISTÉGUY, P. 2010. End-user visual design of web-based interactive applications making use of geographical information: The WINDMash approach. In *Fifth European Conference on Technology Enhanced Learning*. 536 – 541.

LUONG, T. N., LABORIE, S., AND NODENOT, T. 2011. A framework with tools for designing web-based geographic applications. In *Proceedings of the 11th ACM symposium on Document engineering*. DocEng '11. ACM, 33–42.

LUYTEN, K., CLERCKX, T., CONINX, K., AND VANDERDONCKT, J. 2003. Derivation of a dialog model from a task model by activity chain extraction. In *DSV-IS*. Lecture Notes in Computer Science, vol. 2844. Springer, 203–217.

MAHFOUDHI, A., ABED, M., AND TABARY, D. 2001. *From the formal specifications of user tasks to the automatic generation of the HCI specifications*. Springer, London, 331–347.

MANOLA, F. AND MILLER, E. 2004. RDF Primer. Recommendation, W3C. February. `http://www.w3.org/TR/rdf-syntax/`.

MARION, C. 1999. What is interaction design and what does it mean to information designers? `http://mysite.verizon.net/resnx4g7/PCD/WhatIsInteractionDesign.html`.

MATTA, N. 2004. Ingénierie des connaissances en conception pour la mémoire de projets. Université de technologie de Compiègne. Habilitation à diriger des recherches.

MORI, G., PATERNÒ, F., AND SANTORO, C. 2004. Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Transaction on Software Engineering. 30*, 507–520.

MYERS, B. A. 1990. Taxonomies of visual programming and program visualization. *Journal of Visual Languages and Computing 1*, 97–123.

NARAYANAN, N. H. AND HÜBSCHER, R. 1998. *Visual language theory: towards a human computer interaction perspective*. Springer-Verlag, New York, NY, USA, 87–128.

NORMAND, V. 1992. Le modèle Siroco : de la spécification conceptuelle des interfaces utilisateur à leur réalisation. Ph.D. thesis. Thèse de doctorat Informatique préparée au Laboratoire de Génie Informatique (IMAG), Université Joseph Fourier 258 pages.

ORMEROD, T. C. AND SHEPHERD, A. 2004. *Using task analysis for information requirements specification: The SGT method.* Lawrence Erlbaum Associates, 1–24.

PAILLAS, V. 2011. Intérêt des applications "WIND" pour l'exploitation pédagogique de textes décrivant des itinéraires : les pratiques d'annotation au service du lire et interpréter différents langages. Master EFE - 2I2N - FEN de l'Université Toulouse 2 (IUFM Toulouse).

PALLOTA, V. 2003. Computational dialogue models. Tech. Rep. Report IM2.MDM-02, Faculty of Computer and Communication Sciences, Swiss Federal Institute of Technology - Lausanne.

PATERNÒ, F., MANCINI, C., AND MENICONI, S. 1997. ConcurTaskTrees: A diagrammatic notation for specifying task models. In *Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*. INTERACT '97. Chapman & Hall, Ltd., London, UK, UK, 362–369.

PATERNÒ, F. AND SANTORO, C. 2003. A unified method for designing interactive systems adaptable to mobile and stationary platforms. *Interacting with Computers 15,* 3, 349–366.

PATERNÒ, F. 2003. Concurtasktrees: An engineered approach to model-based design of interactive systems. *The Handbook of Task Analysis for Human-Computer Interaction*, 483–503.

PFAFF, G. E., Ed. 1985. *User Interface Management Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

RIALLE, V. 1990. Contribution à la définition du profil de cogniticien : caractéristiques fonctionnelles, savoir et savoir-faire dans l'élaboration d'une base de connaissances. In *APPLICA-90, Second Congrès Multimédia, Intelligence Artificielle et Formation*. 1–10.

ROGERS, Y., SHARP, H., AND PREECE, J. 2011. *Interaction Design: Beyond Human - Computer Interaction*, 3rd ed. John Wiley & Sons.

SALLABERRY, C., ROYER, A., LOUSTAU, P., GAIO, M., AND JOLIVEAU, T. 2009. GeoStream: Spatial information indexing within textual documents supported by a dynamically parameterized web service. In *Proceedings of the International Opensource Geospatial Research Symposium*. OGRS'09. `http://hal.archives-ouvertes.fr/docs/00/45/19/49/PDF/ogrs.pdf`.

SAMAAN, K. AND TARPIN-BERNARD, F. 2004. Task models and interaction models in a multiple user interfaces generation process. In *Proceedings of the 3rd Annual Conference on Task Models and Diagrams*. TAMODIA '04. ACM, New York, NY, USA, 137–144.

SCAPIN, D. AND PIERRET-GOLDBREICH, C. 1989. Towards a method for task description : MAD. In *Proceedings of Work with Display Units (WWU '89)*, L. Berlinguet and D. Berthelette, Eds. North-Holland: Elsevier Science, 27–34.

SEARS, A. AND JACKO, J. A., Eds. 2007. *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, 2nd ed. Boca Raton, London, New-York.

SHU, N. C. 1999. Visual programming: Perspectives and approaches. *IBM Systems Journal 38,* 2/3, 199–221.

SILVA, P. P. D. 2000. User interface declarative models and development environments: A survey. In *Proceedings of DSV-IS2000*, LNCS, Ed. Vol. 1946. Springer-Verlag, 207–226.

TARBY, J.-C. AND BARTHET, M.-F. 1996. The DIANE+ method. In *CADUI* (2005-05-12), J. Vanderdonckt, Ed. Presses Universitaires de Namur, 95–120.

VEER, G. C. V. D., LENTING, B. F., AND BERGEVOET, B. A. J. 1996. GTA: Groupware task analysis - modeling complexity. *Acta Psychologica 91*, 297–322.

VUILLEMOT, R. AND RUMPLER, B. 2009. A web-based interface to design information visualization. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*. MEDES '09. ACM, 26:172–26:179.

WONG, J. AND 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 (CHI)*. 1435 – 1444.

ZIADI, T., BLANC, X., AND RAJI, A. 2009. From requirements to code revisited. In *Proceedings of the 2009 IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*. IEEE Computer Society, 228–235.

Patrick Etcheverry a obtenu son doctorat en informatique en décembre 2002 à l'Université de Pau et des Pays de l'Adour (UPPA). Actuellement Maître de Conférences au sein de l'équipe T2i du Laboratoire d'Informatique de l'Université (LIUPPA - `http://liuppa.univ-pau.fr/`), ses activités de recherche portent sur l'ingénierie des interactions. Ses travaux concernent l'élaboration de modèles, méthodes et outils permettant de concevoir des applications en considérant la dimension interactive comme critère de conception central, l'objectif étant de produire des applications dotées d'un comportement interactif tendant vers "l'idéal".

Sébastien Laborie a soutenu sa thèse en Informatique en mai 2008 à l'Université Joseph Fourier (Grenoble 1). Sa thèse intitulée "*Adaptation sémantique de documents multimédias*" a été co-encadrée par Jérome Euzenat (équipe EXMO, INRIA Grenoble Rhône-Alpes) et Nabil Layaïda (équipe WAM, INRIA Grenoble Rhône-Alpes). Ses thématiques de recherche sont à la croisée des domaines suivants : spécification et adaptation de documents multimédias, représentation et raisonnement spatio-temporel quantitatif/qualitatif, ainsi que les technologies issues du Web Sémantique. Depuis septembre 2010, il est Maître de Conférences au sein de l'équipe T2i du LIUPPA.

Christophe Marquesuzaà est Maître de Conférences à l'IUT de Bayonne et du Pays Basque depuis 1999. Membre de l'équipe T2i du LIUPPA, ses activités de recherche portent sur la modélisation des systèmes d'information basés sur des informations géographiques afin qu'un utilisateur humain non-spécialiste en informatique puisse concevoir de façon autonome une application (orientée web et interactive) conforme à ses besoins. Les contenus manipulés sont valorisés en utilisant l'interaction comme levier. Un domaine d'application visé est celui des EIAH.

Thierry Nodenot est Professeur des Universités en poste à l'IUT de Bayonne et il dirige l'équipe T2i du LIUPPA. Ses domaines de recherche portent sur les Environnements Informatiques pour l'Apprentissage Humain ainsi que les modèles et outils pour l'Interaction Homme-Machine. Membre du conseil d'administration de l'ATIEF (Association des Technologies de l'Information pour l'Education et la Formation), il participe aux comités de programme de nombreuses revues et conférences du domaine.

The Nhan Luong a soutenu sa thèse en Informatique en décembre 2012 au sein de l'équipe T2i du LIUPPA. Sa thèse intitulée "*Modélisation centrée utilisateur final appliquée à la conception d'applications interactives en géographie : une démarche basée sur les contenus et les usages*". Il met aujourd'hui ses compétences à disposition d'une start-up.

Dimensions of User Experience - from the Product Design Perspective

Kerstin Bongard-Blanchy, Carole Bouchard

Dimensions of User Experience – from the Product Design Perspective

KERSTIN BONGARD-BLANCHY
CAROLE BOUCHARD
Arts et Métiers ParisTech

Résumé : Le domaine EU (Expérience Utilisateur) a été étroitement lié au développement des logiciels. Les méthodes UX trouvent cependant de plus en plus d'applications dans le Design de Produits. Aujourd'hui le Designer Produit doit mettre en œuvre des compétences qui vont bien au-delà de la seule définition de l'apparence. L'objet de cet article est de mettre en lumière ces dimensions du design que les Designers Produit soucieux de concevoir dans le respect de l'UX ne sauraient ignorer. L'article apporte ainsi une vue globale sur les dimensions susceptibles d'impacter l'UX. L'identification des dimensions pertinentes puise à la fois dans les théories de la psychologie cognitive, dans les modèles d'interaction homme-machine, ainsi que dans les résultats de la recherche en design. Ces dimensions sont ensuite regroupées sous quatre catégories : les dimensions de la perception humaine, du produit et du contexte de l'utilisation, ainsi que la dimension temporelle. Enfin, ces dimensions sont mises en relation dans un schéma qui illustre le cours de l'expérience entre un utilisateur et un produit, dans un contexte et avec sa temporalité.

Mots clés : Design Produit, dimension UX, produit interactif, design conceptuel, interaction utilisateur-produit

Abstract: The UX domain has so far been strongly associated with software development. However, its methods are finding their way into domains like Product and Service Design. Product Designers now need competencies far beyond classical form-giving. The objective of this paper is to show Product Designers which design dimensions they need to attend to when designing for UX. The paper gives an overview of design dimensions that potentially impact how users' experience products. These dimensions are brought together from theories of Cognitive Science, models of Human-Computer Interaction and findings from Design Research. They are presented under four categories: dimensions of human perception, dimensions of products, dimensions of the context of use and the temporal dimension. In the final part, the identified dimensions are connected into a schema, illustrating their interplay and therefore the journey of UX between a user and a product, in a certain context over a certain time.

Key words: Product Design, UX dimension, interactive product, conceptual design, user-product interaction.

Adresse des auteurs : Kerstin Bongard-Blanchy (kerstin.blanchy@gmail.com), Carole Bouchard (carole.bouchard@ensam.eu) LCPI, Arts et Métiers ParisTech, 151 boulevard de l'hôpital, 75013 Paris, France.

1. Introduction

Do you remember the first TV sets of the 1960's? They were based on analogue technology and had a single functionality. Now look at the latest devices presented at tech fairs such as CES (International Consumer Electronics Show). Today, TV is digital and "smart". The devices are multifunctional. You can access content anytime, get additional information and comment on content via social media. With the arrival of the Internet of Objects the consumer can now personalize more and more products to his/her needs and desires. As a consequence, the role of the Product Designer has gained complexity far beyond form-giving. Today s/he has to design for an experience for the user.

UX (User Experience) has become a topic of interest for researchers and companies even outside its traditional domains. As can be seen in

Figure 1, researchers in cognitive psychology were the first to look at the human perception of objects (Gibson 1986). With the arrival of personal computers, the domain of Human-Computer Interaction emerged as a means to improve the usability of graphical interfaces (Forlizzi and Ford 2000). In the Product Design domain, the study of human factors gained importance for the same objective: that of optimizing usability (Norman 1988; Green and Jordan 1999). Researchers of "Kansei Engineering" (Tomico et al. 2008) in Asia were the first to anticipate the emotions, sensations and semantics conveyed by a Product Design.

In the Western world, the start of the new century saw the advent of research into "Emotional Design"; the study of the emotional value of products (Desmet 2002; Norman 2004). Furthermore researchers of "Affective Engineering" investigate the sensorial experience evoked by materials and textures (Salvia et al. 2010).

While UX is still commonly associated with the realm of Human-Computer Interaction, other domains such as Product or Service Design integrate the findings of these various domains under the paradigm of UX Design / Experience Design (Hassenzahl 2011).

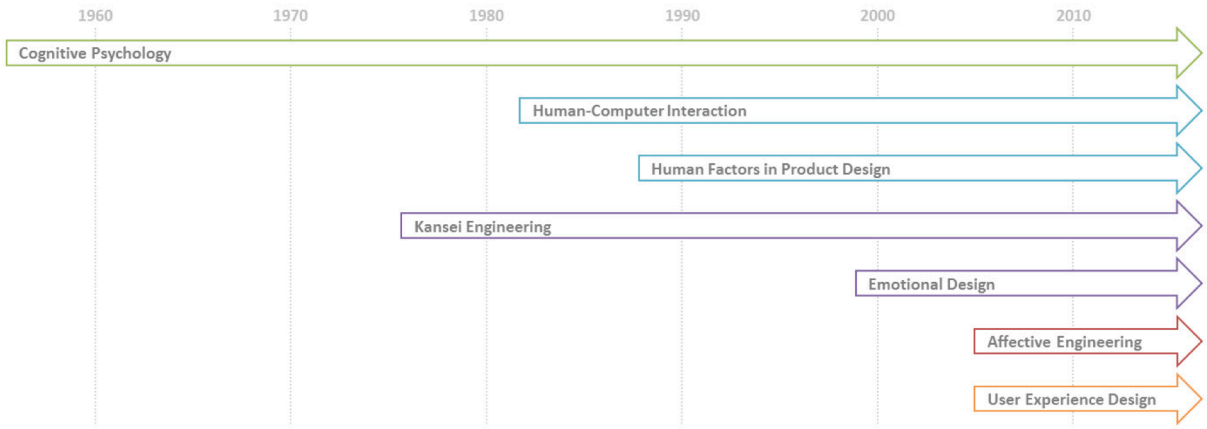

Figure 1: Evolution of research domains related to the UX of Products.

Designing for and researching UX require the consideration of a multitude of dimensions (Robert and Lesage 2011). A review of studied dimensions in UX related papers revealed an array of UX dimensions including affect/emotion, enjoyment/fun, aesthetics/appeal, hedonic quality, engagement/flow, motivation, enchantment and frustration (Bargas-Avila and Hornbæk 2011). This disparate list demonstrates the inconsistencies that remain in the classification of UX dimensions.

UX is defined as "a person's perceptions and responses that result from the use or anticipated use of a product, system or service" (ISO 9241-210). In this definition appear the human and the product with their respective perceptive and responsive capabilities. These capabilities can be addressed by Product and Interaction Design (Djajadiningrat et al. 2004). In this paper we seek to consolidate the insights from the afore mentioned research domains. We look at dimensions that influence the user's perception and then define the range of product dimensions that can evoke a specific reaction of the user. The context of use and the temporality of experience are added as third and fourth layer. Together they lead us towards a schema of UX in Product Design.

2. BASICS OF USER-PRODUCT INTERACTION

The manner in which users experience products is similar to how humans perceive their environment and the objects within it – the mechanisms that humans employ to do this have been investigated by Cognitive Psychology, Neuroscience, Sociology, Semiotics, and Philosophy for many decades.

The human nervous system interacts with the environment through sensory inputs and motor outputs. Organisms use sensors to capture changes of the environment in order to anticipate and coordinate actions that help them to fulfil their goals (e.g. nutrition, reproduction, and belonging). These sensors constantly adapt their state upon interaction with the environment or object. Humans do not perceive characteristics of the environment but rather opportunities for action, so called affordances, like climb-ability, pull-ability, etc. (Gibson 1986). When people perceive their surroundings they recognise coherent patterns of objects or events that have a meaning to them (Russell 2003). For UX design this means visual properties like form or colour do not become stimuli just by themselves but through the affordance they communicate.

The path that sensory input takes inside the nervous system can be modelled as 'perception loops'. Either the input is treated quickly and leads to a direct response – treatment on 'system 1' level– or it enters a more profound treatment of measurement, computation, prediction, evaluation – the so-called 'system 2'. In the latter case, sensed patterns are compared to previous patterns and knowledge templates are continually created or altered (Axelrod 1973; Cariani 2001; Kahneman 2011).

The nervous system determines which actions to take in response to the sensed input. The action organs (e.g. muscles) respond and potentially alter the state of the environment or the object they interact with. This leads to new sensory inputs for the human. The human learns to understand an unfamiliar object by interacting with it (Russell 2003) because perception is based on successive reception and actions (Lenay 2006).

The course of these interaction sequences between the human and the product are furthermore influenced by the context in which they happen and by previous experiences the human had with this environment or object.

Various researchers in the UX domain have proposed schemas that illustrate the interplay of dimensions that form the UX. Among them are Forlizzi and Ford's (2000) 'Initial framework of experience', Hassenzahl's (2003) 'Model of User Experience', Crilly et al.'s (2004) 'Framework for consumer response to the visual domain in product design', Krippendorff's (2005) 'Interaction protocol of an interface', Schifferstein and Hekkert's (2008) 'Model of human-product interaction' and Locher et al.'s (2009) 'Framework for aesthetic interaction', and Robert and Lesage's (2011) 'The inputs and outputs of UX'. Each schema looks at UX from its particular point of view. Together, they give an overview on the multitude of dimensions that constitute UX and that should play a part in early Product Design and Product Design evaluation. In the following, we will try to consolidate the various dimensions inside one schema and give detailed descriptions for each dimension. The dimensions are organised and presented under the four categories (see Figure 2):

1. Dimensions of Human Perception / the User
2. Dimensions of Products
3. Dimensions of the Context of Use
4. The Temporal Dimension of UX

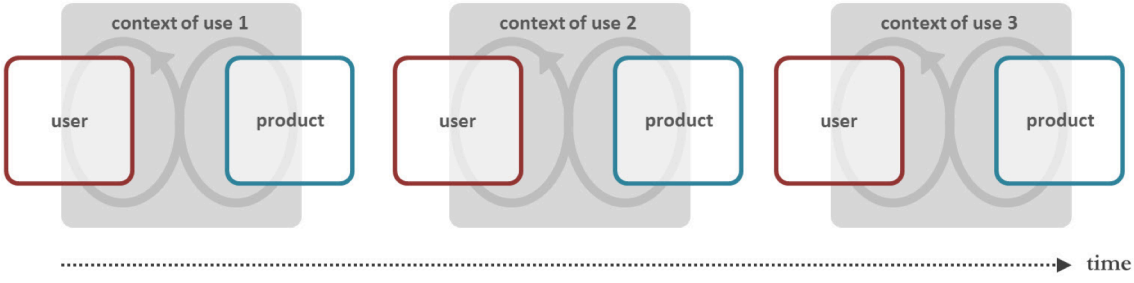

Figure 2: The dimensions of UX are presented under four categories: user, product, context of use and time.

3. DIMENSIONS OF HUMAN PERCEPTION

A user is a person who is targeted to utilise a product. S/he approaches the product with a specific extrinsic goal – what do I consciously expect from this product? – and intrinsic expectations that have been formed by former experiences in addition to his/her personal living conditions (Robert, Lesage 2011). The dimensions that define the user profile are the person's age and gender (Crilly, Moultrie, and Clarkson 2004), cultural background, living environment (Locher, Overbeeke, and Wensveen 2009), education, occupation and interests.

The user encounters a product. Neuroscientists do not yet agree on the brain mechanism involved (Cariani 2001; Bonnardel 2012) in the human perception of objects. What is known is that sensing, cognition and affect, and response are embedded in neural discharge activities. In the following we look at dimensions of these three steps of human perception: human sensors, human cognition and affect, and human responses.

HUMAN SENSORS

In order to perceive their surroundings, humans dispose of a limited range of physiological sensors. They can be classified into exteroceptive, proprioceptive, interoceptive sensors and chronoception (LaMuth 2011). Table 1 gives an overview and definition of each type of human sensor.

Table 1: Types of human sensors.

Sensor	Definition
EXTEROCEPTIVE	Capture stimuli that are external to the organism (Oxford dictionary), in order to keep him/her aware of the environmental changes. They are *visual* (eyes/ophthalmoception), *auditory* (ears and bones/audioception), *gustatory* (tongue with taste buds/gustaoception), *olfactory* (nose/olfacoception), and *somesthetic.* The somesthetic sensors form a complex system able to sense *heat or cold* (thermoreceptors), *pain* (nociceptors), *pressure* (pacinian corpuscles), and *touch* (mechanoreceptors) (Amsel 2005).
PROPRIOCEPTIVE	Arise from the spatial body orientation. They include the sense of *balance* (vestibular input from the inner ear/equilibrioception), *position* (limb and joint afferents/ stretch receptors), and *movement* (muscles for kinesthesioception).
INTEROCEPTIVE	Relate to stimuli that are produced within the organism from the *visceral, digestive,* and *autonomic* systems to secure bodily functions. They work through *chemoreceptors* that regulate hunger, thirst, body temperature, blood pressure, and sexual behaviour.
CHRONOCEPTION	Humans also have limited capabilities to sense of time. However the perception of event duration is subjective and variable. There is no direct sensor for time, it is mostly reconstructed from other sensed information (Tangient 2013).

Humans are also capable of cross-sensory transfers. For example people who lose sight can partially substitute the visual sense with enhanced auditory and tactile senses. Furthermore, sensory substitution devices can be employed as a new means to acquire the non-accessible information to compensate for the loss of one sense (Bach-y-Rita and Kercel 2003; Lenay et al. 2003). Synaesthesia is a phenomenon "in which stimulation of one sensory modality causes unusual experiences in a second, unstimulated modality" (Hubbard and Ramachandran 2005) like letters or numbers that are associated with colours (grapheme-colour synaesthesia), sounds that induce colours (Grossenbacher and Lovelace 2001), or lexical-gustatory synaesthesia where sounds evoke a taste (Ward and Simner 2003). Real synaesthesia happens involuntary and concerns about 4 % of the population (Simner et al. 2006).

This is a brief overview on the type of information a human is able to capture. So far, products address only a few of these capabilities. The focus lies on the visual and auditory information and an emerging trend is tactile and olfactory stimuli. Product Design has high potential to address a wide spectrum of sensors. For example, when touch becomes part of design considerations, the texture but also the felt temperature of the material or the weight of the object are dimensions that humans can sense and appreciate.

HUMAN COGNITION & AFFECT

Once stimulus information has been captured by the human sensors, it enters the perception of the human preparing responses to this stimulus. The perception involves two closely related mechanisms: cognition and affect. Cognition enables the human to understand the environment, affect allows him/her to judge what s/he perceives (Bonnardel 2012). Cognitive events are essentially related to an object/a situation/something. Contrarily, humans experience emotional episodes induced by an object but also independently of any stimulus (Russell 2003). Neuroscientists remain discordant as to whether each are distinct or if cognition is included in the affective process. They do however agree that affect has a cognitive component (Lane et al. 2000; Russell 2003; Scherer 2004; Bonnardel 2012). Khalid (2006) suggests that humans treat stimuli in parallel on the affective and on the cognitive level. Affect is intuitive and experiential. Cognition is the analytical, rational part of information treatment. It creates lasting knowledge and accords meanings.

To understand cognitive and affective mechanisms of human perception is important for Design. Several cognitive and affective components are intentionally or unintentionally addressed by a product or interface. The

better we understand these mechanisms, the more consciously we can design objects to achieve the desired cognitive and affective response.

Cognitive processes link the external stimuli information with the brain (the user's knowledge and memories) in order to reach an interpretation of the stimuli based on their semantic and aesthetic qualities (Hassenzahl 2003). They also allow for evaluation based on the user's concerns/values/preferences and lead to the prediction, planning, and coordination of outputs (Cariani 2001).

In the beginning, a stimulus holds different types of information on several cases at a specific instance in time (e.g. the current condition of an object property such as the fill level of a glass of water). The information is processed in the following way: when a message/input is received, the human seeks to find an already available interpretation of this case. If that exits, s/he will verify if the new information is in accordance with the old interpretation. If that is the case, this interpretation will be reinforced and the behaviour is equivalent to that following a previous similar experience. This is the quickest cognitive treatment – only going through the so-called 'system 1'. If the new information does not fit well with old interpretations, old knowledge patterns are modified or if no interpretation exists so far, other schemas are sought to help with the interpretation. The so-called system 2 enters a more complex cycle of analysis than system 1. The human will try one schema after another starting with the schemas that are the easiest to reach / that have served often – because human cognition is lazy. Once s/he finds a satisfying interpretation, s/he will decide for the appropriate response. The applied schema will become more easily accessible. If no schema can be found to interpret the information, no interpretation is done and no response follows (Axelrod 1973; Kahneman 2012).

Contrary to cognitive processes, the affective part of information processing is still little understood and models are more based on folklore than on scientific concepts (Russell 2003). Some researchers call it affects, others emotions. Yet emotions are only one part of affects. Wundt was the first to propose an affect model. It is based on three continuous dimensions: pleasant-unpleasant (valence), tension-relaxation (tension) and excitement-calm (arousal) (Wundt 1914). Studies suggest that unpleasant stimuli are more arousing than pleasant ones (Bradley and Lang 2000). It has also been shown that events that are accompanied by strong arousal are better remembered. However this seems to be stronger for negative than for positive valence (Kensinger 2009). As a synthesis of the types of affects defined by Scherer (2004) and Russell (2003), Table 2 gathers types of human affects that can be distinguished:

Table 2: Types of human affects.

Affect type	Definition
CORE AFFECT (MOOD)	Diffuse affect states of low intensity, usually rather long lasting, independent of an object/person/event (irritable, depressed, cheerful, buoyant, etc.).
ATTRIBUTED AFFECT	Affect attributed to an object/person/event.
UTILITARIAN EMOTIONS	Brief episodes of synchronised response "of all or most of the five organismic subsystems" to an external or internal stimulus event that is relevant to the organism, personal goals and bodily needs (fear, anger, sadness, joy, disgust, etc.).
AESTHETIC EMOTIONS	Brief episodes experienced through evaluations of sensorial stimuli on their intrinsic qualities, irrespective of bodily needs, current goals or social concerns, addressed to a cause like a piece of art, a design, music, etc. (being full of wonder, admiration, fascination, ecstasy, harmony, rapture, etc.).
PREFERENCES	Stable evaluative judgements on the pleasantness of a stimulus (like/dislike, positive/negative).
CONCERNS/VALUES	Enduring beliefs or predispositions towards an object or a person (loving, hating, desiring, rejecting, etc.). They are based on values that are "desirable, trans-situational goals […] that serve as guiding principles in people's lives" (Schwartz and Sagiv 1995).
AFFECT DISPOSITIONS	Stable personality traits that lead a person to frequently experience certain moods and to react with similar emotions and behaviours to certain types of objects, persons or events (nervous, anxious, reckless, jealous, etc.).

Affect dispositions, concerns/values and preferences are rather stable elements of human affect. They slowly evolve over the person's lifetime as a result of interaction with the environment (Desmet and Hekkert 2007). Core affect can last for a day, some days or even weeks. Attributed affect alters instantly with each new stimulus. There are long-lasting affect components that influence the experienced temporary core affect and attributed affect. As such affect dispositions, concerns/values and preferences are always present, but only addressed once there is an external stimulus to be evaluated. Contrary to this, humans always experience some state of core affect, even if they are not conscious of it. Core affect is an intrinsic state of the human. It describes a mood or a lasting state. At any one time anybody is able to define his/her core affect state.

Products can be an external cause that alters core affect (Russell 2003). When the human perceives an external stimulus, an emotional episode starts. Emotional episodes are evoked by the affective quality of the stimulus, which can be perceived as pleasant or unpleasant, and activating or deactivating. The perception of the affective quality of a stimulus does not necessarily change the core affect. Both can be experienced at the same time even in a non-congruent way. Somebody who feels depressed will find a joke funny, and still feel sad. An affective quality that is perceived by the human becomes an attributed affect (Russell 2003). Scherer's utilitarian and aesthetic emotions can be considered as two different types of attributed affect.

Design evaluations should try to distinguish between attributed affects and core affects in order to draw relevant conclusions. People are able to evaluate the affective quality of objects or events even without being in the real situation (Russell 2003). However, core affects can only be measured in the real interaction. Otherwise there is the risk of misinterpretation caused by misattribution or mood-congruent judgement. Misattribution means that the core affects evoked by one source is mistakenly attributed to the wrong object (Schwarz and Clore 1983). Mood-congruent judgement has the effect that a person who is feeling happy processes more positive information of the stimulus and therefore rates it more positively than she would if she was in a gloomy mood (Bower and Forgas 2012). Negative moods are more easily attributed to an external cause than positive moods (Schwarz and Clore 1983).

To evaluate the affective quality of a product, it might therefore be useful to first measure the core affect of the test person, and then the quality the person attributes to the product. When we downgrade the attributed affects for persons with a highly positive core affects and upgrade it for those with a negative core affects we might get an overall idea of the actual affective quality of the product. It is also relevant to understand the values, preferences and disposition of a person to interpret her evaluation of a design (Bouchard et al. 2009).

Now that we have seen human sensors and cognitive and affective treatment of stimulus information, let us take a look at the responding side of human perception.

HUMAN RESPONSES

Human responses constitute the third element of the human percept-action loop. There are three different response types: physiological/somatic, motor, motivational (Scherer 2004; Scherer 2005). Motor and motivational responses are related to controllable events. Physiological responses can also be of uncontrollable nature (see Table 3).

Table 3: Types of human responses.

Response type	Definition
PHYSIOLOGICAL EFFECTS bodily symptoms	Responses of the human metabolism. Symptoms are temperature sensations as a consequence of blood circulation (red cheeks, cold hands, pale face), respiratory accelerations/decelerations, cardiovascular accelerations/decelerations, muscle tension (weak limbs, trembling, relaxing), constriction in internal organs (stomach ache, lump in the throat), and body fluids (transpiration, salvia, odorant) (Scherer 2004). These responses happen when events appear that disturb the on-going body state. They have for purpose to set the human body in the condition necessary to cope with the situation. An augmented blood rate for example prepares the body to run away from an aggressor. They can also be an unconscious means of communication.
MOTOR EFFECTS facial, gestural, posture, vocal (action)	Occur on three different body parts: the limbs (gestural, postural), the face (mimic), and the speech organs (vocal). Gestures, postures, mimics and voice are more or less controllable means of human communication that enable him/her to visually and audibly show the outsider in which state the situation put him/her and how s/he will react (Scherer 2004). Values that distinguish motor effects are the speed, the amplitude, the frequency/rhythm, pauses, and patterns. The meanings of facial expressions have been subject of research in psychology from Darwin, to Paul Ekman, and Nico H. Frijda.
MOTIVATIONAL EFFECTS action tendencies and readiness (behaviour)	Stable action tendencies towards a specific object or situation (Scherer 2005), in general approach or withdrawal (Russell 2003). Become visible in human comportment. The person directs her attention to the subject, like somebody who always buys a specific brand or acts with consciousness on the environmental impact of his/her behaviour.

Responses that users show to products can be observed or measured with techniques such as eye tracking or with physiological measurements such as EEG (electroencephalogram) or GSR(galvanic skin response) (Kim 2011). They provide relatively objective data to researchers. However, even though these responses are the result of cognitive and affective judgements, “no specific action or action tendency is produced by or is necessary for a specific emotion” (Russell 2003). That means an observed behaviour does not represent one specific affect felt by the person. It is still necessary to look at the totality of responses and the person’s subjective self-evaluations in order to draw conclusions on his/her experience.

This first part introduced the dimensions of human perception. We saw that humans draw upon a wide range of sensors to capture stimulus information. Here lies a first entry point for original ideas to design for UX. We also talked about the manner in which stimulus information is processed through the interplay of human cognition and affect. To address certain cognitive responses, designers can work on a semantic expression of a product. Designs can also transport a certain affective quality. But the designer cannot control all individual dimensions of human perception. A person’s current mood or previous memories are outside of the design scope. Nevertheless, this points at a huge potential for intelligent products that can adapt their properties to the user’s individual affective disposition and knowledge. Finally, we saw the range of human responses. They can be of physiological, behavioural or motivational nature. A design that causes reactions on each of these three levels is likely to be more engaging than one that only addresses one level. Product developers should envision ways to enable their products to not only react on behavioural but also on physiological responses. In order to

do so, the designer needs to know which product dimensions s/he can potentially influence through his/her design. This is therefore the topic of the next section.

4. DIMENSIONS OF PRODUCTS

A product is a specific type of object or interface or service that serves a certain purpose and that can be classed in a certain sector such as automobile, food, cosmetics, etc. (Krippendorff 2005). Hassenzahl (2003) assigns four major functions to products: 1. Enable people to manipulate their environment, 2. Stimulate personal development, 3. Express identity, and 4. Evoke memories. To do so, products must possess certain properties that facilitate these functions. Classical dimensions of Product Design are functionalities, semantics, form, colour, material, texture etc. They will not be further discussed here since it seems more interesting to look at properties that enable the product to interact with the environment or the user. Today products too come with sensors that facilitate intelligent responses to user responses. Some product properties can also be designed with their own dynamic behaviour. An analogy can therefore be drawn with the human perceptive system. In the following, potential sensory and responsive capabilities of products will be presented.

PRODUCT SENSORS

In order to react to user inputs, products need to be capable of perceiving human responses. Classic consumer products (furniture or decoration, or the external parts of industrial designs like car chassis, telephone housing, etc.) are usually not equipped with sensors. Nowadays there are more and more products that show a dynamic reaction to user inputs through tangible or graphic user interfaces. These products come equipped with a great variety of sensors that can be classified into the types listed in Table 4 (Robotworx 2013):

Table 4: Types of sensors for consumer products.

Sensors categories	Sensors types	Definition
PHYSICAL SENSORS	Active sensors	Emit some form of energy - ultrasonic, laser, infrared
	Passive sensors	Receive energy Example: a camera
LOGICAL SENSORS		Supply robot with a percept from the physical sensor
PROPRIOCEPTIVE SENSORS		Monitor self-maintenance, control internal status Examples: Global Positioning System (GPS), Inertial Navigation System (INS), Shaft/rotary Encoder, Compass, Inclinometer
EXTEROCEPTIVE SENSORS	Contact Sensors	Emit a signal on physical contact Measure the interaction force and torque Tactile sensors, conductivity (linear, circular, discontinued)
	Range Sensors	Measure distance to an object Two principles: time-of-flight and triangulation (reflection, sonar, capacity)
	Vision Sensors	Extract, characterize and interpret visual information
EXPROPRIOCEPTIVE SENSORS		Combine proprioceptive and exteroceptive monitoring to measure the relative position through directional sensors, to measure difference between internal and external heart; Examples: panning sonar sensors, force sensors
FORCE SENSORS		Measure torque or force or weight

Today Product Designers have access to a wide range of sensors and actuators (for behavioural response) that are easy to employ in the phases of rapid prototyping – nearly like a pen for sketching or some 3D software for modelling. There are for example the Arduino kits (Arduino 2012) or Phidgets (Phidgets 2012) with MAX/MSP, Processing or Adobe Flash with ActionScript that allow designers to model interactive behaviour coupled with sensing technology. These techniques find their way into design education and once design graduates master them, they will enrich the interactivity of products and with it the UX (Helm, Aprile, and Keyson 2008).

PRODUCT RESPONSES

All externally perceivable product dimensions are potential stimuli for the user. Products can be characterised through concrete dimensions of lower order and abstract dimensions of higher order (Snelders 2003).

Concrete dimensions include the properties of functional, behavioural, semantic and sensorial properties. They are designed to evoke abstract dimensions on a pragmatic and a hedonic level (Hassenzahl 2003). The term ‘pragmatic’ refers to the product utility and usability. A product with a strong pragmatic quality simply fulfils a functional purpose, e.g. using a pair of scissors to cut paper. The hedonic side addresses the user emotions, values, and memories. A product with a strong hedonic quality, such as a family photo, might have no pragmatic quality but can be very valuable to the person. While the perception of abstract properties might differ strongly between users, the perception of concrete properties is essentially the same between people of different

backgrounds. In user-centred design, abstract properties define the design objective. The Product Designer then materialises this objective through the concrete properties (Snelders 2003).

Classical Product Designs stimulate the user through function and appearance/semantics. Since intelligent technologies have found their way into consumer products, more and more products are connected to a network, are pro-active and capable of adaption to events (Ross and Wensveen 2010). The design of intelligent products goes beyond traditional Product Design and can even be independent of physical materials (Lim, Lee, and Lee 2009). To conceive interactive products, designers require "a new language of form that incorporates the dynamics of behavior" (Ross and Wensveen 2010).

Dynamic product responses mean that the product adapts its properties to the situation. It may for example alter the orientation of some of its components (like Nabaztag moving his ears) or its colour (e.g. clicked links). These changes can be instantaneous or follow a fixed pattern (Lim, Lee, and Lee 2009; Lin and Cheng 2011). Shape change is a specific behaviour that waits to be exploited by Product Designers. Examples are changes of orientation, form, volume, texture, viscosity, spatial changes, as well as addition or subtraction of elements or changes of permeability (Kirkegaard Rasmussen et al. 2012). Related to it are spatial distribution and motion patterns that can express complex meanings that people interpret on a social dimension (approaching, avoiding, etc.) (Mutlu et al. 2006). Other examples of dynamic product responses can be seen in Table 5.

Table 5: Possible dynamic responses of products.

Dimension	Property	Behaviour
Material	(Depending on the material)	Elasticity, colour, temperature, conductivity…(Ashby and Johnson 2010)
Colour		Hue, saturation, lightness
Texture	Visual texture	Transparency…
	Tactile texture	Smoothness...
Illumination		Intensity, colour, movement pattern, fade pattern
Sound	Volume	Increase or decrease
(Özcan 2008)	Timbre	Change type
	Pitch	Increase or decrease
	Melody / pattern	Change in a pattern or randomly
	Spatiality (source)	Orientation change
Form		Geometrically defined
		Free-forms
		Adapt to human dimensions (ergonomic)
	Body volume	Increase or decrease
Orientation		Between two states or seamlessly
		360°, on 3 spatial axes
Permeability		Binary or seamlessly
		From complete permeability to impermeable
Spatial distribution		of product elements in the available space systematic (meaningful) or random way;
		the product itself can change its position in the room/space
Components		Can be added/subtracted
Functionality		Can change when the purpose is distorted, e.g. a bottle becomes a vase, a chair used as a ladder
Language style		Casual / formal style
		Natural / artificial style (Blanchy 2010)

Products are formed by a wide range of design dimensions. They define the product appearance as well as its behaviour. To this day Product Design is often seen as something static. But products can be equipped for dynamic responses. Designers have the possibility to explore, for example, form changes thanks to flexible materials, or spatiality and motion changes on graphic interfaces in order to communicate with the user or to adapt the behaviour to the context of use. To design such adaptive behaviour, Product Designers require knowledge about the different types of human or environmental responses that a product can capture if equipped with the corresponding sensors and they should exploit the possibilities of the here presented response types.

5. DIMENSIONS OF THE CONTEXT OF USE

Any UX is embedded in a specific interaction situation. The context of use also influences which properties the human or product sensors will capture. It might furthermore influence the treatment of the stimulus information. The externalities of the context of use that influence UX are situational, cultural, and social dimensions (Forlizzi and Ford 2000; Crilly, Moultrie, and Clarkson 2004; Krippendorff 2005; Locher, Overbeeke, and Wensveen 2009; Robert and Lesage 2011). Table 6 provides an overview of potential context dimensions:

Table 6: Dimensions of the context of use.

Situational dimensions	Viewing time
	Related products/features/things
	Place
	Time
	Event/activity
Cultural dimensions/references	Similar products/brands/activities
	Clichés/stereotypes
	Trends/fashions/tastes/conventions
Social dimensions	Communication, sharing, collaboration

6. THE TEMPORAL DIMENSION OF EXPERIENCE

Last but not least, time is another important UX dimension. Some experiences such as a surprise can only be lived once while other experiences grow over various use episodes. Hassenzahl (2010) demands a "longitudinal approach" when designing for UX. He distinguishes between a micro (an hour of usage), meso (a few weeks of usage), and macro (years of usage) perspective on the UX with a product. Roto et al. identified four types of UX over time: the anticipated UX (before usage), the momentary UX (during usage), the episodic UX (after usage), and the cumulative UX (multiple use situations over time) (Roto et al. 2011). From the Product Design point of view, there are two different types of temporality that can be effectively shaped by the designer: The interaction sequence and certain aspects of the cumulative UX.

Interaction sequences: A dynamically changing product is conceived as a continuous cycle of sensing and response on the human and on the product side. The human as well as the artefact are both capable of sensing input, processing this data, and responding with distinct behaviour (output actions). The data circulates between the user and the product and is transformed (Krippendorff 2005). The emotions felt by the user change over the different interaction sequences (Lin and Cheng 2011).

Long-term UX: An interactive product might show reoccurring response patterns to user actions, but the user might respond differently after a while. Surprise can only be triggered once. Excitement changes into comfort once the user gets familiar with the product. Even when the product disappears from his/her sight, the memory of it can still trigger emotions (Norman 2004). Karapanos et al. (2009) studied the experience of iPhone users from the intention of purchase, until after a few months of usage. They proved a shift in UX over time. During the initial orientation phase, the experience is mainly formed by stimulation and learnability. This leads to a certain familiarity with the product. In the following incorporation phase usefulness and long-term usability are important. A functional dependency of the user on the product appears. Finally the user can enter the phase of identification where s/he shows an emotional attachment to the product.

Designers should anticipate the course of UX over interaction sequences as well as the evolvement of UX over the time of product use during their design process.

7. CONCLUSION: FROM PRODUCT DESIGN TO DESIGN FOR UX

For centuries, people have created and refined products. A common understanding of Gestalt laws or colour harmonies has developed within and beyond cultural borders. In the past, engineers dealt with functions, components and performance; while Product Designers were asked to work on form, colour, and semantic (Djajadiningrat et al. 2004). Now microcontrollers have considerably enriched the capabilities of everyday products to process the user input and to respond with discrete output behaviour (Lim and Kim 2011) and therefore, to adapt their behaviour to the context of use and user actions/responses (Djajadiningrat et al. 2004).

This brings new opportunities as well as challenges to Product Designers. The user perception of interactive products goes far beyond form-giving. Interaction impacts the affective experience (Schuster Smith 2008). Hassenzahl illustrates this with the example of the Philips Wake-Up Light: "…it substantially changes the way one wakes up. It changes the experience. The object itself, its form, is rather unremarkable" (Hassenzahl, Eckoldt, and Thielsch 2009).

Even though people might not expect great emotional stimulation from all products, it cannot be neglected that each artefact causes a reaction in its user – appeal, indifference or rejection. Thus, all goods and services made for consumers have to respond to some kind of physiological, psychological or social human need (Lim 2003). Today form-giving is only one part of Product Design.

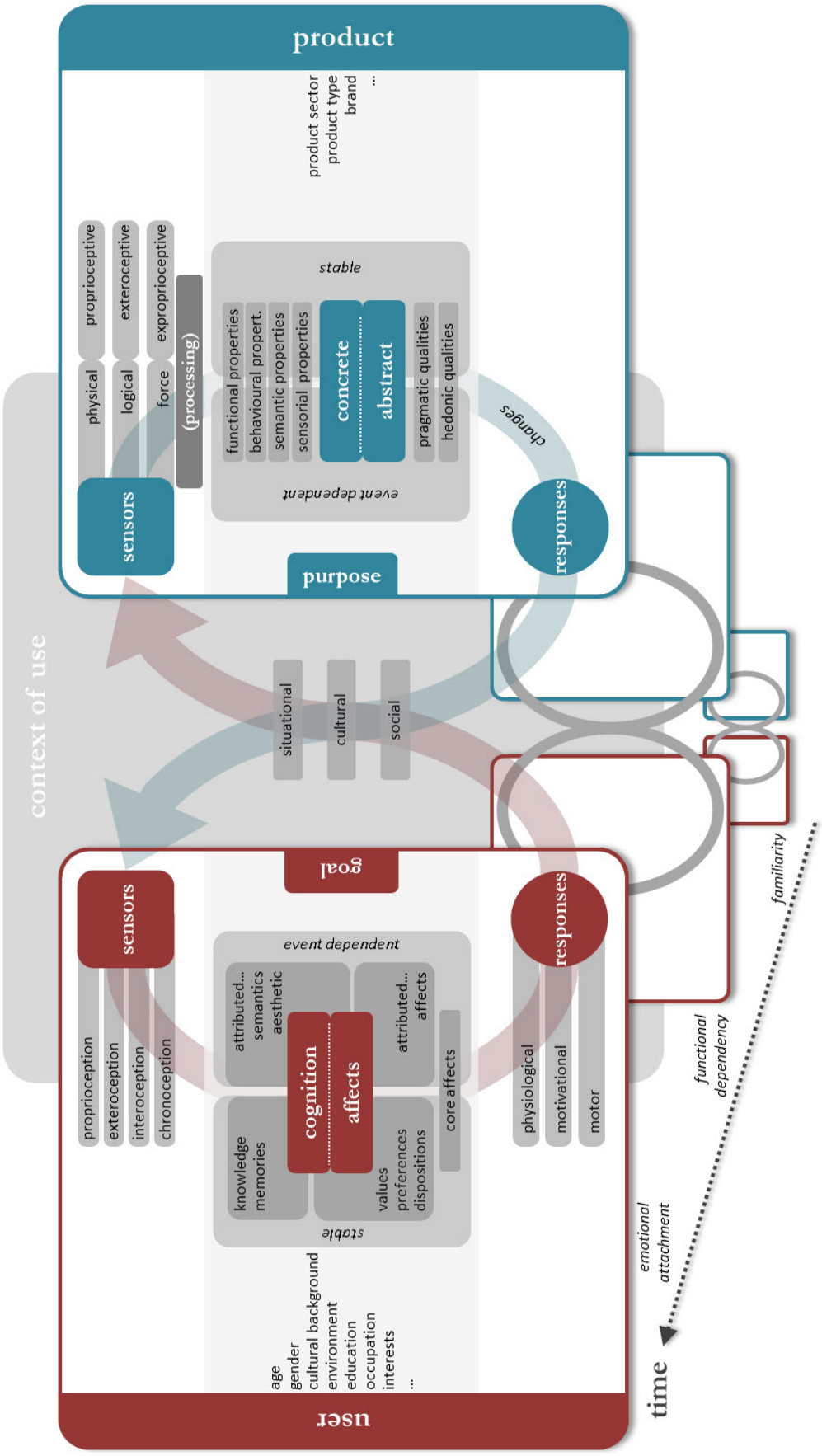

Figure 3: Schéma of dimensions of the User-Product Experience.

Through this literature review, we have sought to demonstrate how UX results from the interplay of a wide range of concrete (form, colour, semantic, function) and abstract (affective and sensorial quality, semantic quality, aesthetic quality) product dimensions, together with their perception by the target user, the context in which s/he encounters the product and the temporality of the experience (see Figure 3).

Today the challenge for Product Designers is to adopt a holistic view of UX during conceptual design and design evaluation. That means, conceptual design should always target congruence between identified (and potentially evolving) user goals and the product purpose. The Product Designer defines function, semantic, sensorial and behavioural properties with regard to their hedonic and pragmatic quality. To validate and improve the designed experience, concepts and prototypes should then be regularly evaluated through user tests. These evaluations verify which pragmatic and hedonic qualities the user really attributes to the designed product (Figure 4).

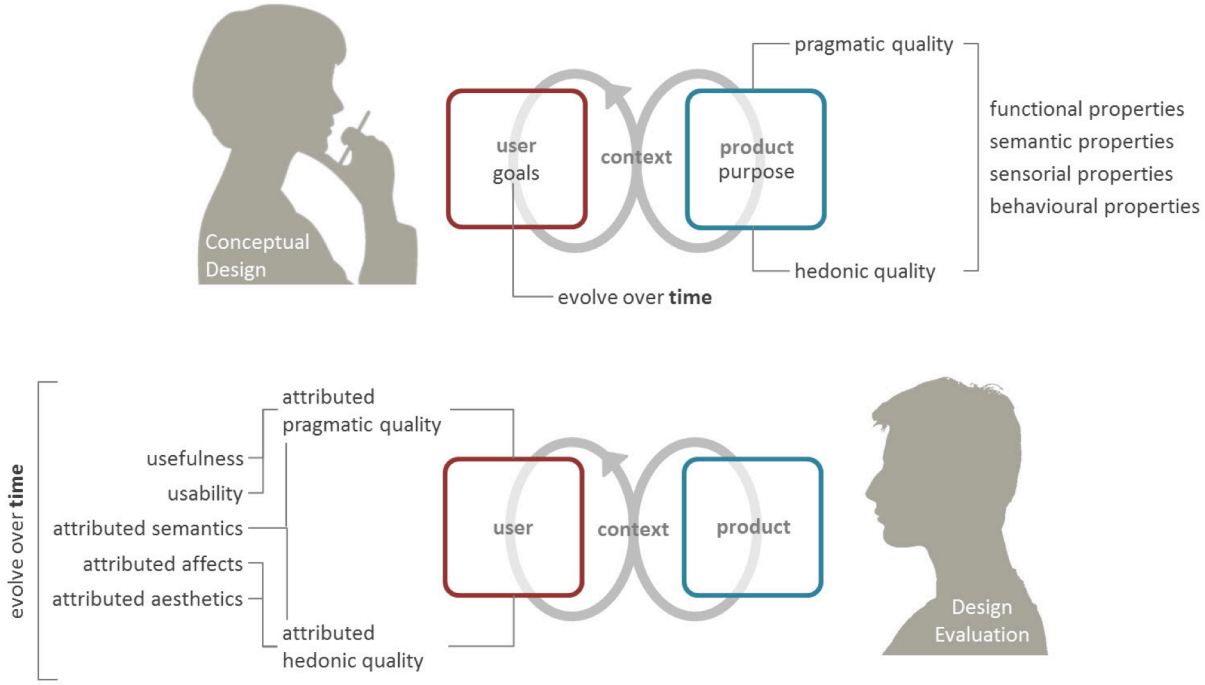

Figure 4: Dimensions of conceptual design (top) and design evaluation (bottom) when designing for UX.

ACKNOWLEDGEMENT

This literature review is part of a thesis undertaken in the context of the ANR project Skippi. We thank the Agence Nationale de la Recherche for its financial and pedagogical support.

REFERENCES

AMSEL, S. 2005. "'Special Senses.' Touch. Exploring Nature Educational Resource." http://exploringnature.org/db/detail.php?dbID=25&detID=47.

ARDUINO. 2012. "Arduino." http://www.arduino.cc/.

ASHBY, M. F., and JOHNSON, K. 2010. *Materials and Design*. Oxford: Butterworth-Heinemann.

AXELROD, R. 1973. "Schema Theory: An Information Processing Model of Perception and Cognition." *The American Political Science Review* 67 (4): 1248–1266.

BACH-Y-RITA, P., AND KERCEL, S. W. 2003. "Sensory Substitution and the Human–machine Interface." *Trends in Cognitive Sciences* 7 (12) (December): 541–546.

BARGAS-AVILA, J. A., and HORNBÆK, K. 2011. "Old Wine in New Bottles or Novel Challenges: A Critical Analysis of Empirical Studies of UX." In *Conference on Human Factors in Computing Systems*, 2689-2698.

BLANCHY, K. 2010. "Interface Design Based on the Philosophy of Japanese Hospitality." In *Design and Emotion*. Chicago.

BONNARDEL, N. 2012. "Cognition and Emotion in Creative Design." In *Attention, Representation, and Human Performance: Integration of Cognition, Emotion, and Motivation*, edited by S. Masmoudi, A. Naceur, and D. Yun Dai, 187-200. London: Psychology Press.

BOUCHARD, C., MANTELET, F., AOUSSAT, A., SOLVES C., GONZALEZ J., PEARCE, K., LOTTUM C., and COLEMAN, S. 2009. "A European Emotional Investigation in the Field of Shoe Design." *International Journal of product Development* 7 (1): 3–27.

BOWER, G. H., and FORGAS, J. P. 2012. "Mood and Social Memory." In *Handbook of Affect and Social Cognition*, 97–121. Taylor & Francis.

BRADLEY, M. M., and LANG, P. J. 2000. "Measuring Emotion: Behavior, Feeling, and Physiology." In *Cognitive Neuroscience of Emotion*, 242–276. New York: Oxford University Press.

CARIANI, P. 2001. "Symbols and Dynamics in the Brain." *Bio Systems* 60 (1-3): 59–83.

CRILLY, N., MOULTRIE, J., and CLARKSON, P. J. 2004. "Seeing Things: Consumer Response to the Visual Domain in Product Design." *Design Studies* 25 (6): 547–577.

DESMET, P. 2002. "Designing Emotions". *PhD thesis*. TU Delft.

DESMET, P., and HEKKERT, P. 2007. "Framework of product Experience." *International Journal of Design* 1: 57–66.

DJAJADININGRAT, T., WENSVEEN, S., FRENS, J., and OVERBEEKE, K. 2004. "Tangible products: Redressing the Balance between Appearance and Action." *Personal and Ubiquitous Computing* 8 (5) (July 31): 294–309.

FORLIZZI, J., and Ford, S. 2000. "The Building Blocks of Experience: An Early Framework for Interaction Designers." In *Symposium on Designing Interactive Systems*, 419–423. New York, New York, USA.

GIBSON, J. J. 1986. *The Ecological Approach to Visual Perception*. Tayler & F. New York, New York, USA: Psychology Press.

GREEN, W., JORDAN, P. W. 1999. *Human Factors in Product Design: Current Practice and Future Trends.* London: Tayler and Francis.

GROSSENBACHER, P G., and LOVELACE C. T. 2001. "Mechanisms of Synesthesia: Cognitive and Physiological Constraints." *Trends in Cognitive Sciences* 5 (1) (January 1): 36–41.

HASSENZAHL, M. 2003. "The Thing and I: Understanding the Relationship Between User and product." In *Funology: From Usability to Enjoyment*, edited by M. A. Blythe, K. Overbeeke, A. F. Monk, and P. C. Wright, 31–42. Dordrecht: Kluwer Academic Publishers.

HASSENZAHL, M. 2010. *Experience Design - Technology for All the Right Reasons*. Edited by John Caroll. Morgan & Claypool.

HASSENZAHL, M. 2011. "UX and Experience Design." In *Encyclopedia of Human-Computer Interaction*, edited by Mads Soegaard and Rikke Friis Dam. Aarhus, Denmark: The Interaction Design Foundation. http://www.interaction-design.org/encyclopedia/user_experience_and_experience_design.html.

HASSENZAHL, M., ECKOLDT, K. and THIELSCH, M. T. 2009. "UX Und Experience Design – Konzepte Und Herausforderungen." In *Usability Professionals 2009. Berichtband Des Siebten Workshops Des German Chapters Der Usability Professionals Association e.V.*, edited by H. Brau, 233–237. Stuttgart: Fraunhofer-Verlag.

Helm, A. VAN DER, APRILE, W. and KEYSON, D. 2008. "Experience Design for Interactive products : Designing Technology Augmented Urban Playgrounds for Girls." *PsychNology* 6 (2): 173–188.

HUBBARD, E. M., and RAMACHANDRAN, V. S. 2005. "Neurocognitive Mechanisms of Synesthesia." *Neuron* 48 (3) (November 3): 509–20.

KAHNEMAN, D. 2012. *Thinking, Fast and Slow*. London: Penguin Books.

KARAPANOS, E., ZIMMERMAN, J., Forlizzi, J., and Martens, J.-B. 2009. "UX over Time: An Initial Framework." In *Conference on Human Factors in Computing Systems*. Boston, Massachusetts.

KENSINGER, E. A. 2009. "Remembering the Details: Effects of Emotion." *Emotion Review : Journal of the International Society for Research on Emotion* 1 (2) (January): 99–113.

KHALID, H. M. 2006. "Embracing Diversity in User Needs for Affective Design." *Applied Ergonomics* 37 (4) (July): 409–18.

KIM, J. 2011. "Modeling Cognitive and Affective Processes of Designers in the Early Stages of Design: Mental Categorization of Information Processing." *PhD thesis*. Arts & Métiers ParisTech.

KIRKEGAARD RASMUSSEN, M., PEDERSEN, E. W., PETERSEN, M. G., and HORNBÆK, K. 2012. "Shape-Changing Interfaces: A Review of the Design Space and Open Research Questions." In *Conference on Human Factors in Computing Systems*, 1–10. Austin, Texas.

KRIPPENDORFF, K. 2005. *The Semantic Turn: A New Foundation for Design*. Boca Ratan, London, New York: CRC Press.

LAMUTH, J. E. 2011. "Input Specificity - Exteroceptive, Interoceptive, and Proprioceptive." http://www.angelfire.com/rnb/fairhaven/inputspecificity.html.

LANE, R. D., NADEL, L., ALLEN, J. J. B. and KASZNIAK, A. W. 2000. "The Study of Emotion from the Perspective of Cognitive Neuroscience." In *Cognitive Neuroscience of Emotion*, 3–11. New York: Oxford University Press.

LENAY, C. 2006. "Énaction , Externalisme et Suppléance Perceptive." *Intellectica* 1 (43): 27–52.

LENAY, C., GAPENNE, O., HANNETON, S., Marque, C., and GENOUËLLE, C. 2003. "Sensory Substitution: Limits and Perspectives." In *Touching for Knowing*, edited by Yvette Hatwell, Arlette Streri, and Edouard Gentaz, 275–290. Paris: John Benjamins B.V.

LIM, D. 2003. "Modelisation Du Processus de Conception Centrée Utilisateur, Basée Sur L'intégration Des Méthodes et Outils de L'ergonomie Cognitive: Application À La Conception d'IHM Pour La Télévision Interactive." *PhD thesis*. Arts & Métiers ParisTech.

LIM, Y., and KIM, D.-J. 2011. "Interactivity Attributes for Expression-Oriented Interaction Design." *International Journal of Design* 5 (3): 113–128.

LIM, Y., LEE, S.-S., and LEE, K.-Y. 2009. "Interactivity Attributes: A New Way of Thinking and Describing Interactivity." In *Conference on Human Factors in Computing Systems*, 105–108. Boston.

LIN, M.-H., and CHENG, S.-H. 2011. "Semantic Shifting within the Interaction Sequence." In *IASDR*, 1–10. Delft.

LOCHER, P., OVERBEEKE, K. and WENSVEEN, S. 2009. "A Framework for Aesthetic Experience." In *Conference on Human Factors in Computing Systems*. Boston.

MUTLU, B., FORLIZZI, J., NOURBAKHSH, I., and HODGINS, J. 2006. "The Use of Abstraction and Motion in the Design of Social Interfaces." In *DIS*. Pennsylvania.

NORMAN, D. A. 2004. *The Design of Everyday Things*. New York: Basic Books.

NORMAN, D. A. 2004. *Emotional Design*. Cambridge, MA: Basic Books.

Özcan, E. 2008. "Product Sounds". *PhD thesis.* TU Delft.

PHIDGETS, Inc. 2012. "Phidgets." http://www.phidgets.com/.

ROBERT, J.-M., Lesage, A. 2011. "Designing and evaluating user experience." In *Handbook of Human-Computer Interaction. A human-centered design approach*, edited by G.A. Boy, 321-338. Surrey, UK: Ashgate Publishing Limited.

ROBOTWORX. 2013. "Robot Sensors." Accessed February 5. http://www.robots.com/education/exteroceptive-sensors.

ROSS, P. R., and WENSVEEN, S. A. G. 2010. "Designing Behavior in Interaction: Using Aesthetic Experience as a Mechanism for Design." *International Journal of Design* 4 (2): 3–13.

ROTO, V., Law, E., VERMEEREN, A., and Hoonhout, J. 2011. "UX White Paper: Bringing Clarity to the Concept of UX." *Seminar*. Dagstuhl, Germany.

RUSSELL, J. A. 2003. "Core Affect and the Psychological Construction of Emotion." *Psychological Review* 110 (1): 145–172.

SALVIA, G., ROGNOLI, V., MALVONI, E., and LEVI, M. 2010. "The Objectivity of Users' Emotional Experience with Textiles Biological and Mechanical Tests for the Prediction of the Sensorial Profile of Fabrics." In *Seventh International Conference on Design and Emotion*, 1–12. Chicago, IL.

SCHERER, K. R. 2004. "Which Emotions Can Be Induced by Music? What Are the Underlying Mechanisms? And How Can We Measure Them?" *Journal of New Music Research* 33 (3) (September): 239–251.

SCHERER, K. R. 2005. "What Are Emotions? And How Can They Be Measured?" *Social Science Information* 44 (4) (December): 695–729.

SCHIFFERSTEIN, H.N.J., HEKKERT, P. (Eds.) 2008. Product Experience. Elsevier Science.

SCHUSTER Smith, H. 2008. "Emotional Evaluation of a product/System". *PhD thesis.* University of Central Florida.

SCHWARTZ, S. H., and SAGIV, L. 1995. "Identifying Culture-Specifics in the Content and Structure of Values." *Journal of Cross-Cultural Psychology* 26 (1) (January 1): 92–116.

SCHWARZ, N., and CLORE, G. L. 1983. "Mood, Misattribution, and Judgements of Well-Being: Informative and Directive Functions of Affective States." *Journal of Personality and Social Psychology* 45 (3): 513–523.

SIMNER, J., MULVENNA, C., SAGIV, N., TSAKANIKOS, E., WITHERBY, S. A., Fraser, C., Scott, K., and Ward, J. 2006. "Synaesthesia: The Prevalence of Atypical Cross-Modal Experiences." *Perception* 35 (8): 1024–1033.

SNELDERS, D. 2003. "An Exploratory Study of the Relation between Concrete and Abstract product Attributes." *Journal of Economic Psychology* 25 (December): 803–820.

TANGIENT LLC. 2013. "Chronoception." http://scalometer.wikispaces.com/chronoception.

TOMICO, O., MIZUTANI, N., LEVY, P., TAKAHIRO, Y., and CHO, Y. 2008. "Kansei Physiological Measurements and Constructivist Psychological Explorations for Approaching User Subjective Experience During and After product Usage." In *International Design Conference - Design*, 529-536. Dubrovnik.

WARD, J., and SIMNER, J. 2003. "Lexical-Gustatory Synaesthesia: Linguistic and Conceptual Factors." *Cognition* 89 (3) (October): 237–261.

WUNDT, W. M. 1914. *Grundriss Der Psychology*. Edited by Jazzybee Verlag.

Kerstin Bongard-Blanchy is a UX designer und researcher. During her PhD thesis at the Product Design and Innovation Laboratory, Arts et Métiers ParisTech, France she investigated ideation and evaluation tools for UX Design (2010-2013). Following her studies of mechanical engineering/industrial design at Dresden University of Technology, Germany (2006), she had worked as Interaction Designer for Panasonic in Japan (2006–2010) where she designed mobile phone interfaces, product sounds and lights as interaction means. She currently develops and tests user interfaces for Dassault Systèmes.

Carole Bouchard is professor at Arts et Métiers, ParisTech, France. She teaches and guides research in the Product Design and Innovation Laboratory. She obtained her PhD in industrial engineering in 1997 in the field of automotive design and has been professor since 2012. Her research focus is on Kansei Design, as well as creativity and innovation in early design stages. She pilots various research projects that seek to develop innovative design tools to efficiently integrate Kansei principles into the design process. Current projects include user-centred Eco Design methods, a word-based conception software for multidisciplinary teams and the development of an immersive spatialized design workspace.