La traçabilité d'une interface tangible, à l'aide d'une base de données spatio-temporelle

Samuel Baillaud, Jean Caelen

Laboratoire LIG, Domaine universitaire
BP 53, 220 rue de la Chimie
38041 Grenoble Cedex 9
Samuel.Baillaud@imag.fr, Jean.Caelen@imag.fr

RÉSUMÉ. *L'article décrit la couche dite de traçabilité dans une interface tangible, constituée d'une table et d'objets interactifs matériels. La conception de l'interface pose le problème de la représentation des états, événements et comportements des objets évoluant sur la table, au cours du temps. L'article détaille la spécification d'un SGBD avec contraintes, spatio-temporel, actif, et son usage pour maintenir, et donner accès à, une telle représentation.*

ABSTRACT. *This paper describes the layer called traceability layer in a tangible interface, composed of a table and of material interactive objects. The interface's design faces us with the problem of representing the states, events, and behaviours of the objects operating on top of the table, in the course of time. This paper specifies in detail an active, spatio-temporal, constraint DBMS, and its possible use to maintain, and give access to, such a representation.*

MOTS-CLÉS : *interface tangible, table interactive, traçabilité, base de données active, base de données avec contraintes, base de données spatio-temporelle*

KEYWORDS : *tangible interface, interactive table, traceability, active database, constraint database, spatio-temporal database*

1. Introduction

Le domaine des interfaces tangibles est en pleine évolution. Avec ce type d'interface, il ne s'agit plus seulement d'interagir avec des objets virtuels sur un écran à l'aide de la souris et du clavier, mais de manipuler de véritables objets physiques dotés de capacités numériques d'interaction. Dans cette mouvance nous avons développé une table constituée de dalles d'antennes pouvant détecter des objets munis d'étiquettes RFID (projet TTT, labellisé en 2007 par l'ANR). Cette table est une sorte de « rétine 2D

personnages d'histoires, et projette des textes, images, vidéos, à la surface d'une table. Il permet de parcourir les histoires selon le point de vue des différents personnages, en manipulant les pions qui les représentent. Les systèmes « pen and paper » (Yeh *et al.*, 2008) localisent précisément la mine d'un crayon, et projettent éventuellement des textes, tableaux, diagrammes, schémas, images, à la surface de feuilles de papier. Ils permettent de manipuler et communiquer des données numériques, sans être devant un ordinateur. Le système « Xenakis » (Bishof *et al.*, 2008) localise des objets, et projette des images et vidéos à la surface d'une table. Il permet de composer de la musique dans un environnement sonore et visuel plaisant. On pourrait citer encore de très nombreux exemples d'un domaine en pleine effervescence.

Le concept d'interface tangible a été précisé par Ullmer *et al.* (Ullmer *et al.* 2000) : « Tangible User Interfaces (TUIs) give physical form to digital information, employing *representations* *controls*

per se

Ainsi les systèmes interactifs que nous souhaitons réaliser, ne se démarquent pas seulement par le canal interactif, canal tangible plutôt que graphique, mais ils se démarquent aussi par la nature des associations liant les objets aux fonctions qu'ils remplissent à travers les actions qu'ils effectuent, nature qu'il faut prendre en compte dans la maintenance des connaissances du système interactif au-delà d'une représentation de ce qui se passe à l'instant actuel, dans le monde où opèrent les utilisateurs avec leurs objets.

Les systèmes interactifs de ce type entrent dans la catégorie dite « d'initiative mixte » évoquée par Hearst *et al.* (Hearst *et al.*, 1999) : d'une part, les messages adressés par l'utilisateur à l'application, ne sont pas seulement des commandes, ce peuvent être aussi des informations sur le monde et les objets physiques qui s'y trouvent étant donné que la machine ne les comprend pas. L'application peut alors construire une représentation de la situation, de l'activité, ou des habitudes/habiletés (profil) de l'utilisateur ou encore donner à voir cette activité à quelqu'un (pour apprendre, corriger ou comprendre cette activité), d'autre part, les messages adressés par l'application à l'utilisateur, ne sont pas uniquement des

informations concernant l'état de l'application mais également des informations concernant l'activité de l'utilisateur à travers les objets qu'il manipule.

Ainsi dans une architecture logicielle d'interface tangible il apparaît nécessaire d'introduire une couche que nous appelons « Traçabilité », couche qui n'est pas nécessaire dans les architectures classiques d'interface graphique comme MVC, ARCH ou PAC (Coutaz, 1990, Duval *et al.*, 1999). Cette couche a pour objet de garder l'historique des mouvements effectués, des positions des objets et de leurs relations spatiales, et de les rattacher à leurs actions pour ainsi donner de la « sémantique » aux objets manipulés que l'utilisateur ne peut sinon comprendre.

3. Présentation hardware de la table TTT

TTT est une table magnétique, carrée de 1 m de côté, pilotable depuis un ordinateur, comprenant une matrice d'antennes RFID organisée sous forme de "dalles" contenant chacune 64 antennes. Les propriétés de cette table sont :

–

4. L'architecture logicielle de la table TTT

L'architecture logicielle de la table est composée de trois couches allant du logiciel enfoui dans la table aux API applicatives permettant à un programmeur de développer des systèmes multi-agents et des interfaces utilisateurs. Les trois couches sont :

Couche « SMA » (système multi-agent) pour le noyau fonctionnel de l'application et les interfaces utilisateurs, (il s'occupe aussi de définir les fonctions des objets).

Couche « Traçabilité » qui correspond à une sorte d'adaptateur de noyau fonctionnel (avec le vocabulaire ARCH) et de gestion de la couche middleware

(c’est un SGBD spatio-temporel qui gère aussi les empreintes des objets sur la table en termes de formes et de faces).

Couche « Interface matériel » qui est une couche d’interaction bas niveau.

Dans la suite nous nous intéresserons uniquement à la couche « Traçabilité » pour laquelle nous avons défini un outil logiciel qui permet de construire une composante qui trace les objets à la surface de la table, c'est à dire qui maintient une représentation de ces objets et des actions qu’ils subissent au cours du temps. C’est une sorte de « miroir » en mémoire des objets de leur activité sur la table.

Les problèmes à résoudre à ce niveau sont :

5. Un SGBD spatio-temporel pour la couche Traçabilité

Pour répondre aux questions ci-dessus nous avons considéré une technique basée sur un système de gestion de base de données active spatio-temporelle, qui permet en théorie d’apporter une solution pratique, puisqu’elle maintient une représentation des objets dans l’espace et au cours du temps, et que leur comportement peut-être décrit à l’aide de faits, interrogeables par des requêtes posées à la base.

Le schéma des relations de cette base, correspond à une ontologie, qui décrit quelles classes d'objets peuvent exister à la surface de la table, et quelles propriétés intemporelles, états, et événements peuvent concerner ces objets. Propriétés, états, et événements, peuvent concerner plusieurs objets à la fois. Par exemple, « la cuiller c est à côté de l'assiette a », est un état concernant c et a ; « Les pièces p et q du jeu de construction, sont de la même couleur » est une propriété concernant p et q. « La princesse p entre dans le château c », est un événement concernant p et c. Cette ontologie peut être représentée par un schéma entités-associations avec une relation d'héritage entre entités (Elmasri *et al.*, 2003) : où les classes d'objets sont représentées par des entités, les propriétés, états et évènements par des associations ou attributs d'entités.

Les faits concernant les relations du schéma de la base, constitutifs de son contenu, représentent les objets qui existent et ont existé à la surface de la table, et les propriétés, états et événements qui les concernent et les ont concernés. Le langage de requêtes de la base, permet de définir des vues sur celle-ci. Ces vues

correspondent encore à des classes d'objets pouvant exister à la surface de la table, à des propriétés, états et évènements pouvant les concerner ; elles sont définies à partir des relations du schéma. On peut demander au SGBD, quel est l'ensemble des faits concernant une relation ou une vue. On peut aussi lui demander, de notifier tout changement de cet ensemble, dès qu'il s'accomplit. Un SGBD capable de faire de telles notifications est dit actif ; une vue concernant laquelle le SGBD peut faire ces notification est dite matérialisée (Gupta *et al.*, 1993).

Nous spécifions maintenant en détail notre SGBD spatio-temporel. Pour cela, comme cela se fait parfois (Ullman, 1988), nous décrivons une logique, affiliée à la logique du premier ordre, par rapport à laquelle notre SGBD est spécifié.

La logique associée à notre SGBD, met en jeu des types de termes, une relation d'héritage entre ces types, et des fonctions, constantes, prédicats sur ces types. Comme cela est permis par la norme SQL2, des bibliothèques de types prédéfinis sont associées à notre SGBD. Quelques-uns de ces types sont donnés dans la table 1 : types temporels, spatiaux (comme ceux d'un SGBD spatial (Stolze, 2003), ou d'une API géométrique (Sun Microsystems, 2001)), types spatio-temporels (comme ceux du SGBD spatio-temporel de Erwig *et al.* (Güting *et al.*, 2003)).

La logique associée à notre SGBD, met en jeu des formules et des relations. A une formule ou relation F, on attribue une signature :

$F : T(a_1 : E_1, ..., a_n : E_n)$,

Type/fonction/prédicat/constante	**Définition informelle**
≤	≤
∞	∞

Table 1. *Des types de termes simples*

avec T un type temporel de formule, a_1, ..., a_n des attributs, E_1, ..., E_n des types de termes. Un fait concernant F, est une affirmation de la forme :

$F(a_1 : c_1, ..., a_n : c_n)$,

avec c_1, ..., c_n des constantes de types E_1, ..., E_n. Le type temporel de F, précise quelle est sa signature, et comment l'ensemble des faits concernant F, peut être représenté, par un ensemble fini d'affirmations. Ce type peut être l'un des cinq suivants : propriété (intemporelle), état, évènement instantané, évènement duratif, ou formule (quelconque). Ces cinq types temporels sont décrits dans la table 2. Les relations du schéma, peuvent être des types propriété, état, ou événement.

L'idée que l'ensemble des faits concernant une relation ou formule peut être infini, pourvu qu'il soit représentable par un ensemble fini d'affirmations, a d'abord été celle de la programmation logique avec contraintes. Elle a été adaptée aux bases de données par Kanellakis (Kanellakis, 1994). Nous tirons les notions d'état, événement, propriété, de travaux d'IA visant à définir des « logiques temporelles » affiliées à celle du premier ordre (Shoham 1987, Shanahan, 1999, Galton, 2005).

Comme cela est permis par la norme SQL3, l'utilisateur de notre SGBD peut introduire de nouveaux types de termes (« user-defined types »), et de nouveaux prédicats, fonctions, constantes. Des opérateurs permettent de construire des types de tuples, et d'ensembles de tuples (« nested relations »). Ils sont décrits dans la table 3. Nous appelons les fonctions qui, comme la fonction interpolation, ont pour paramètre un ensemble de tuples, des fonctions d'agrégation. Comme une formule ou relation, un ensemble de tuples a un type temporel : propriété, état, événement

F : Property($a_1 : E_1$, ..., $a_n : E_n$)
F : State(time : Instant, $a_1 : E_1$, ..., $a_n : E_n$) ∀ ≤
F : I-Event(time : Instant, $a_1 : E_1$, ..., $a_n : E_n$)
F : D-Event(beginTime : Instant, endTime : Instant, $a_1 : E_1$, ..., $a_n : E_n$)
F : F -m- □ □ tF □ m>b □ tF s tanx , ...1rm- □ t(ti dTimepf □ uY a tandm- □ t(

Table 2. *Les types temporels de formules et relations*

instantané ou duratif, ou ensemble (quelconque) de tuples. Ce type temporel précise quels sont les attributs des tuples que l'ensemble contient, et comment il peut être représenté, comme une union finie d'ensembles de tuples. Une table analogue à la table 2, devrait décrire ces cinq types temporels d'ensembles de tuples.

La logique associée à notre SGBD met en jeu des opérateurs primitifs de construction de formules. La table 4 donne les constructions qu'ils permettent. Les opérateurs R, ⊤, ∧, ∧ où f est id_E ou c_E, ∧ où p est =, ∃, ∨, ∧¬, appartiennent au calcul des relations de Codd (Ullman, 1988). Les opérateurs ∧ où f est quelconque, ∧ où p est quelconque, ∧ , respectent la norme SQL2. L'opérateur ∧ ∈ (opérateur d'« unnesting ») respecte la norme SQL3. L'opérateur (a_1 : Instant) fait de notre SGBD un SGBD avec contraintes d'ordre entre instants (Kanellakis, 1994), comme le SGBD temporel de Toman (Toman, 1998, Chomicki *et al.*, 2005). Les formules qui sont construites grâce à ces opérateurs primitifs, à partir des relations du schéma, et qui sont des types propriété, état, ou événement, constituent le langage de requêtes. Ces formules sont appelées des requêtes ou des vues.

Les opérateurs primitifs de construction de formules, tels que nous les avons présentés jusqu'à présent, permettent de construire toutes les formules de la logique du premier ordre. Afin de garantir la calculabilité, de l'ensemble des faits concernant toute formule que ces opérateurs permettent de construire, des restrictions de l'usage autorisé de ces opérateurs, complètent leur définition. La troisième colonne de la table 4 devrait être complétée avec ces restrictions. Ces restrictions mettent en jeu diverses propriétés des formules, données dans la table 5. La quatrième colonne de la table 4 donne la définition récursive de la propriété TYPE ; elle devrait être complétée par d'autres colonnes, donnant les définitions récursives des autres propriétés des formules. La propriété ATTRIBUTS, et les restrictions la mettant en jeu, ont été définies par Codd pour son calcul des

Type/fonction/prédicat/constante	**Définition informelle**
∈	∈
Σ	Σ
	∈
	∈

Table 3. *Les types de termes complexes*

F =	$F(<a_1 : c_1, ..., a_n : c_n>) :=_{def}$	Restrictions	TYPE(F) =
⊤	⊤		
∧	∧		
∧	∧		
∧	∧		
∃	∃		
∨	∨		
∧ ¬	∧ ¬		
∧	∧		
∧ ∈	∧ ∈		

Table 4. *Les opérateurs primitifs de construction de formules*

relations. Par exemple, pour être autorisé à construire la formule ($F_1 \wedge (\neg F_2)$), on doit avoir que ATTRIBUTS(F_1) $\supseteq$ ATTRIBUTS(F_2). Un exemple de restriction mettant en jeu la propriété TYPES, est que pour pouvoir construire ($F_1 \wedge F_2$), les attributs communs à F_1 et F_2 doivent avoir des types compatibles. Les propriétés INSTANCIES et EGAUX, et les restrictions les mettant en jeu, sont inspirées des spécifications des prédicats prédéfinis (« built-in ») données pour les systèmes de programmation logique (1997). Par exemple, pour pouvoir construire ($F_1 \wedge a_n = f(a_{k(1)}, ..., a_{k(m)})$), on doit avoir que ($a_{k(1)}$, ..., $a_{k(m)}$) $\subseteq$ INSTANCIES(F_1) ; ou encore, pour pouvoir construire ((State) F_1), on doit avoir que ATTRIBUTS(F_1)\(time) $\subseteq$ INSTANCIES(F_1). La propriété DEPENDANTS et les restrictions la mettant en jeu, sont une version simplifiée et clarifiée de celles définies par Goldin *et al.* (Chomicki *et al.*, 2003). Par exemple, pour pouvoir construire ($F_1 \wedge a_n = g(\{<a_{k(1)}, ..., a_{k(m)}>\})$), ($a_{k(1)}$, ..., $a_{k(m)}$) et ATTRIBUTS(F_1)\($a_{k(1)}$, ..., $a_{k(m)}$) doivent être séparés dans le graphe DEPENDANTS(F_1), par INSTANCIES(F_1) (deux ensembles de points d'un graphe, sont séparés par un troisième, si tout chemin reliant un point

Propriété de F	Définition informelle
	≤

Table 5. *Les propriétés des formules et relations*

du premier ensemble et un point du deuxième, passe par un point du troisième).

Il est possible de définir d'autres opérateurs de construction de formules, comme des compositions des opérateurs primitifs de la table 4. La table 6 donne quelques-uns de ces opérateurs, propres à l'aspect temporel de notre SGBD. On peut en trouver des semblables dans d'autres travaux (Chomicki *et al.*, 2005, Galton, 2005).

6. Usage du SGBD spatio-temporel, par la couche Traçabilité

Nous présentons maintenant l'usage que la couche Traçabilité peut faire du SGBD que nous avons défini, pour représenter ce qui se passe à la surface de la table au cours du temps. Nous donnons dans la table 7 une partie du schéma des relations de la base, que peut maintenir la couche Traçabilité. Puis nous donnons dans la table 8, sept exemples de vues que l'on peut définir sur cette base.

Le premier exemple de vue, utilise des opérateurs logiques de définition d'un état à partir d'autres états. De tels opérateurs se retrouvent dans les SGBDs temporels de Snodgrass (Böhlen *et al.*, 1995) et de Toman, dans le SGBD spatio-temporel avec contraintes linéaires de Rigaux *et al.* (Grumbach *et al.*, 2003), et dans des logiques temporelles de l'IA. Le premier exemple utilise aussi la possibilité d'intégrer dans la définition d'un état, un prédicat, une fonction ou une constante : ici, le prédicat into est intégré. D'autres exemples de ce genre, mettent en jeu les autres opérateurs logiques de construction d'un état, et les autres constantes, fonctions, prédicats, d'une bibliothèque de types de termes spatiaux.

Les deuxième et troisième exemples de vues, utilisent la possibilité d'intégrer dans la définition d'une vue, des agrégations d'états. Le deuxième exemple utilise cette possibilité, car les successions discrètes de positions perçues, sont des approximations de trajectoires continues. Ici on veut leur appliquer la fonction intersects. Ce problème est aussi traité par les SGBDs spatio-temporels de Rigaux *et al.* et Erwig *et al.*. D'autres exemples de ce genre, mettent en jeu d'autres

F =	Définition informelle	Condition
^		
^		
^		
^		
^		

Table 6. *Des opérateurs composés de construction de formules*

existingTobject : Property(tObject)
existingTside : Property(tSide)
existingTshape : Property(tShape)
side : Property(tObject, tSide, index)
outline : Property(tSide, tShape)
onTable : State(tObject, tSideIndex, time)
location : State(tShape, shape, time)

Table 7. *Des relations du schéma, de la base maintenue par la couche Traçabilité*

objectIntoShape : State(tObject, tShape, time)
objectIntersectsShape : State(tObject, tShape, time)
stillObject : State(tObject, time)
objectEntersIntoShape : I-Event(tObject, tShape, time)
objectJumps : D-Event(tObject, beginTime, endTime)
objectNotOnTable : State(tObject, time)
objectEntersIntoARedShape : I-Event(tObject, time)
shapeIsRed : State(tShape, time)
objectJumpsTwice : D-Event(tObject, beginTime, endTime)

Table 8. *Sept exemples de vues, sur la base maintenue par la couche Traçabilité*

constantes, fonctions, prédicats, d'une bibliothèque de types de termes spatio-temporels. Le troisième exemple, utilise la possibilité d'intégrer des agrégations d'états dans la définition d'une vue, car la procédure de calcul d'une fonction, appliquée à la succession de positions qu'est la trajectoire, ne peut pas être réduite à la réitération d'un traitement, appliqué à chaque position prise isolément des autres. Ici la fonction appliquée est still : on ne peut voir qu'un objet est immobile à un instant, en ne voyant sa position qu'à cet instant. Ce problème est aussi traité par le SGBD spatio-temporel de Erwig *et al.*. D'autres exemples de ce genre, mettent en jeu des fonctions calculant des caractéristiques d'une trajectoire, comme les points de rebroussement ou d'inflexion, les vitesses ou les courbures, ou des fonctions de reconnaissance de formes dans la trajectoire (formes pouvant témoigner qu'un geste a été fait).

Les quatrième et cinquième exemples de vues, utilisent la possibilité de définir un événement à partir d'un état. Le quatrième exemple définit un événement comme le commencement de la persistance d'un état à un instant. Le cinquième exemple définit un évènement, comme la persistance d'un état pendant une période.

Les sixième et septième exemples de vues, utilisent la possibilité de définir un événement à partir d'autres événements. Le sixième exemple, définit un événement à partir d'un événement, et d'un état servant de précondition. D'autres exemples de ce genre, mettent en jeu d'autres manières dont un état peut servir de condition dans la définition d'un événement ; par exemple, condition au sujet de la période ou de l'instant d'accomplissement de l'événement, ou postcondition. Le septième exemple, définit un événement comme une séquence d'événements. D'autres exemples de ce genre, mettent en jeu d'autres compositions simples d'événements, comme celles utilisées pour définir un modèle de tâche (Coutaz, 1990) : événements en parallèle, ou simultanés ou presque simultanés, ou alternatifs. Encore d'autres exemples, mettent en jeu des compositions plus générales, comme dans des travaux d'IA sur les logiques temporelles (Kautz, 1991, Shanahan, 1999).

7. Conclusion

Nous avons montré la nécessité d’adjoindre une couche particulière de « traçabilité » à une interface tangible du fait de la rupture de représentation forme/ fonction à la fois chez l’utilisateur et en machine. Nous avons spécifié un SGBD spatio-temporel, qui maintient et met à disposition, une représentation de ce qui se passe sur la table. C'est une base de données avec contraintes, active, pour laquelle sont définis un schéma de relations, un contenu, et des vues (qui peuvent être matérialisées).

Une telle couche ne revêt pas les mêmes caractéristiques dans les interfaces graphiques bien qu’elle ait pout but aussi de maintenir une représentation de ce qui se passe, dans le monde constitué des objets, grâce auxquels l'utilisateur et le système peuvent interagir (objets tels que des fenêtres, icônes, menus, etc.). Les différences essentielles résident dans le fait que les objets « disparaissent » à la

fermeture d'une application ou du moins gardent leurs propriétés d'une session à une autre lorsqu'on les réactive. Ces objets restent des objets numériques mono-fonctionnels (si l'on veut changer de fonction il est plus facile d'en créer de nouveaux).

Ici chaque objet tangible se double d'un ou plusieurs objets numériques en « miroir ». Chacun d'eux peut répondre à des questions sur l'objet qu'il représente, il peut notifier des évènements concernant cet objet, et il peut exécuter des commandes d'action sur cet objet. Mais une composante habituelle a une structure « orientée objet », que n'a pas la base de données que permet de construire notre outil. De manière consensuelle, il est admis que le passage à cette structure orientée objet, est un progrès des interfaces, apparu lorsqu'elles sont devenues graphiques (Coutaz, 1990), c'est à dire, dès qu'elles ont du représenter un petit monde composé d'objets affichés sur un écran.

Un outil habituel de construction d'interfaces graphiques, laisse au programmeur la charge de programmer la détection de tous les états, évènements, propriétés, qu'il souhaite définir lui-même. Dans notre système, le programmeur peut utiliser le langage de requêtes associé à la composante de traçabilité, pour formuler des définitions d'états, évènements, propriétés, propres à l'interface particulière qu'il souhaite réaliser. Puis il peut les déclarer comme des vues. La composante de traçabilité se chargera alors de les détecter.

8. Bibliographie

Bishof M., Conradi B., Xenakis - Combining tangible interaction with probability-based musical composition, *Proc.2nd int. conf. on tangible embedded interaction*, 2008.

Böhlen M., Jensen C., Snodgrass R., Evaluating and enhancing the completeness of TSQL2. Tech. report 95-05, Computer science dep., University of Arizona, 1995.

Chomicki J., Goldin D., Kuper G., Toman D., Variable independence in constrainst databases, *IEEE Transactions on knowledge and data engineering*, 15(6), 2003.

Chomicki J., Toman D., Temporal databases, *Handbook of temporal reasoning in AI*, Elsevier, 2005.

Coutaz J., *Interface homme-ordinateur : conception et réalisation*, Dunod, 1990.

Duval T., Nigay L., Implémentation d'une application de simulation selon le modèle PAC-Amodeus, *Actes des Journées IHM*, Cépaduès, 1999.

Elmasri R., Navathe S., *Fundamentals of database systems*, 4th ed., Addison-Wesley, 2003.

Galton A., Eventualities, *Handbook of temporal reasoning in AI*, Elsevier, 2005.

Gibson J., *The ecological approach to visual perception*, Boston : Houghton Mifflin, 1979.

Grumbach S., Koubarakis M., Rigaux P., Spatio-temporal models and languages : an approach based on constraints, *Spatio-temporal databases : the CHOROCHRONOS approach*, LNCS 2520, Springer, 2003.

Gupta A., Mumick I., Subrahmanian V., Maintaining views incrementally, *Proc. SIGMOD int. conf. on management of data*, 1993.

Güting R., Böhlen M., Erwig M., Jensen C., Spatio-temporal models and languages : an approach based on data types, *Spatio-temporal databases : the CHOROCHRONOS approach*, LNCS 2520, Springer, 2003.

Hearst M. (ed.), Allen J., Guinn C., Horvitz E., Mixed-initiative interaction, Trends and controversies, *IEEE Intelligent Systems*, 14(5), 1999.

Kanellakis P., Goldin D., Constraint programming and database query languages. *Proc. 2nd conf. on theoretical aspects of computer software*, LNCS 789, Springer, 1994.

Kautz H., A formal theory of plan recognition and its implementation, *Reasoning about plans*, Morgan Kaufmann, 1991.

Mazalek A., Davenport G., Ishii H., Tangible viewpoints : a physical approach to multimedia stories. *Proc. 10th int. conf. on multimedia*, 2002.

Shanahan M., The event calculus explained, *AI today*, LNAI 1600, Springer, 1999.

Shoham Y., Temporal logics in AI : semantical and ontological considerations, *Artificial intelligence*, 33, Elsevier, 1987.

Stolze K.. SQL/MM Spatial - the standard to manage spatial data in a relational database system. *Proc. 10th conf. on database systems for business, technology and web*, 2003.

Sun Microsystems, *Programmer's guide to the Java 2D API*, Sun Microsystems, 2001.

Toman D., Point-based temporal extensions of SQL and their efficient implementation. *Temporal databases : research and practice*, LNCS 1399, Springer, 1998.

Ullman J., *Principles of database and knowledge-base systems*, vol. 1, Computer science press, 1988.

Ullmer B., Ishii H., Emerging frameworks for tangible user interfaces, *IBM Systems journal*, 39(3-4), 2000.

Underkoffler J., Ishii H., Urp : a luminous-tangible workbench for urban planning and design. *Proc. SIGCHI conf. on human factors in computing systems*, 1999.

Wielemaker J., *SWI-Prolog reference manual*. University of Amsterdam, 1997.

Yeh, R., Paepcke A., Klemmer S., Iterative design and evaluation of an event architecture for pen-and-paper interfaces. *Proc. 21st annual symposium on user interface software and technology*, 2008.