Université de Montpellier II
Laboratoire LIRMM - UMR 5506

Présentation de travaux en vue de l'obtention de
l'Habilitation à Diriger des Recherches

De la Conception à l'Extraction de Connaissances dans les Bases de Données

Pascal Poncelet

soutenue le jeudi 4 janvier 2001 devant le jury composé de :

Président	Mr Claude Chrisment	Professeur, Université Toulouse I
Rapporteurs	Mme Christine Collet	Professeur, ENSIMAG Grenoble
	Mr Georges Gardarin	Professeur, Université Versailles-Saint Quentin
	Mme Geneviève Jomier	Professeur, Université Paris Dauphine
Examinateurs	Mr Michel Habib	Professeur, Université Montpellier II
	Mme Danièle Hérin	Professeur, Université Montpellier II
	Mme Violaine Prince	Professeur, Université Montpellier II
	Mme Colette Rolland	Professeur, Université Paris I

Table des matières

Synopsis

L'apparition des nouvelles technologies, l'ouverture à d'autres domaines d'applications, l'émergence de nouvelles catégories d'utilisateurs ont eu un impact considérable à tous les niveaux (entreprises, organisations, grand public) et tout particulièrement sur l'activité des informaticiens, concepteurs et développeurs.

Ces dernières années, les travaux de recherche en bases de données ont intégré ces évolutions et ont porté sur la définition de nouveaux modèles logiques plus riches et plus puissants [9, 19, 3, 20, 69], le développement des systèmes associés [51, 33, 32, 46, 47] et la proposition d'approches conceptuelles [112, 89, 43, 17, 110, 19, 32].
Plus récemment, l'aide à la décision a fait l'objet de nombreux travaux qui peuvent se scinder en deux grandes tendances complémentaires. La première s'intéresse, avant tout, à la matière première que constituent les données collectées. Elle recouvre les démarches de gestion d'entrepôts de données, leur conception, le calcul de données synthétisées, plus compactes et plus "informatives" [20, 13, 1, 5]. Dans la seconde tendance, sont regroupées en particulier toutes les approches de fouille de données (ou Data Mining) [4, 24, 64, 52]. Leur objectif est d'extraire, à partir d'ensembles de données souvent très volumineux et donc inexploitables manuellement, la connaissance la plus pertinente possible par rapport aux besoins des décideurs.

Le travail présenté dans ce mémoire concerne à la fois les aspects Conception et Fouille de données. Au travers de ces deux aspects nous nous situons à la fois en amont et en aval des applications et même s'ils semblent a priori très éloignés, ils sont cependant très proches notamment par les variations offertes autour du comportement. Alors que dans le premier cas, en amont du développement des applications, il est indispensable de modéliser le comportement d'une application. Dans le second cas, en aval des applications, l'intérêt de l'analyse du comportement des utilisateurs ou du système est le garant d'une meilleure compréhension de la dynamique de l'application.

Largement influencées par les travaux menés en génie logiciel et sur la spécification de systèmes temps réel [63, 84, 86, 94, 91] dont la problématique est voisine, les approches de conception comportementales visent à représenter la dynamique des applications tout au long de leur vie, i.e. leurs réactions au cours du temps quand certains événements interviennent [17, 43, 110, 112, 119]. Il convient donc de pouvoir spécifier les opérations que doit réaliser l'application, mais le point clef, au niveau conceptuel, est de pouvoir décrire leurs déclenchements et enchaînements, et là, le concept d'événement est essentiel.

Dans le cadre de la fouille de données, l'idée est de tirer profit des données historiques

accumulées pour rechercher toute information pertinente pour l'expert [24, 23]. Cet historique, cependant, se contente de répertorier des événements, de manière chronologique et évidemment sans exprimer de façon explicite les liens de causalité ou de synchronisation entre eux. La recherche d'associations entre événements peut permettre de cerner de tels liens. L'idée est de découvrir des combinaisons caractéristiques d'événements en analysant l'histoire de chacun des individus.

Cette symétrie par rapport au comportement se retrouve également dans la structuration des applications. En proposant un modèle structurel, nous souhaitons modéliser les composants statiques d'une application. En analysant les données de grandes bases il est possible de retrouver la structure sous-jacente d'une application notamment via l'extraction de dépendances fonctionnelles.
S'il est naturel que la structuration d'une application ait des conséquences sur les techniques de fouilles de données à mettre en œuvre (règles d'association multi-relation, règles d'association de bases de données orientées objet [16]), il ne faut pas négliger qu'il existe cependant des possibilités inverses, i.e. où les techniques de fouilles de données offrent une aide à la spécification de la structuration (par exemple, dans le cas de bases de données semi-structurées ou de rétro-conception à partir de règles obtenues).

La dualité introduite par la problématique posée se retrouve dans le plan de ce mémoire. La présentation adoptée est thématique mais également chronologique. La première partie de ce mémoire présente l'approche de conception d'applications avancées et décline le verbe modéliser selon différents angles puisque la modélisation y est vue comme un moyen de structurer, de faire agir ou de rechercher. C'est ce dernier verbe qui est ensuite abordé, toujours en suivant différentes variations, dans la seconde partie traitant des travaux menés dans le cadre de la fouille de données et d'extraction de connaissances : rechercher pour comprendre, pour s'adapter et enfin rechercher pour structurer. Nous présentons dans les paragraphes suivants un résumé des travaux que nous avons réalisés. Nous précisons également les résultats obtenus, les projets associés [1] ainsi que les encadrements réalisés.

Conception d'applications avancées

Les travaux réalisés dans le cadre de la conception d'applications avancées ont fait l'objet du projet IFO_2 dont l'objectif était d'offrir une approche de conception complète de la modélisation de la structure et du comportement à la génération de code vers des systèmes cibles.

Modéliser pour structurer
Dans le cadre de mon travail de thèse, ma contribution a porté sur la définition d'un modèle formel IFO_2 permettant la représentation des aspects statiques des applications. Ce modèle étend le modèle sémantique IFO, défini par S. Abiteboul et R. Hull, en intégrant les concepts du paradigme objet. Outre une représentation de haut niveau adaptée aux besoins des nouvelles applications, la démarche proposée permet la génération automatique de code à partir de spécifications IFO_2. Les systèmes cibles visés pour cette phase d'implantation sont aussi bien des systèmes classiques (en l'occurrence le SGBD relationnel

1. Dans chaque partie, un chapitre détaille les résultats obtenus et le contexte de la recherche.

Oracle) qu'orientés objet (le système O_2). L'approche définie permet ainsi de couvrir les différentes phases de développement d'application au niveau statique. Elle propose également des possibilités originales de prise en compte des évolutions de la représentation conceptuelle. Cet aspect s'appuie sur la formalisation des règles d'évolution de schéma et la définition d'invariants garantissant la cohérence des manipulations opérées.
Le travail effectué a fait l'objet de publications notamment dans une revue nationale (ISI) et des conférences internationales (VLDB, CAiSE) et nationales (BDA, INFORSID) dans le domaine. Il a également donné lieu à la réalisation de plusieurs prototypes. Ce travail s'est inscrit dans le cadre d'un contrat EERP (External European Research Project) avec le centre de Recherche et Développement de Digital Europe à Ferney Voltaire.

Modéliser pour faire agir
Mon activité de recherche s'est ensuite poursuivie au sein du projet IFO_2 en participant à la proposition d'un modèle dynamique, complétant le précédent, en permettant la représentation du comportement des applications. Ce modèle privilégie la symétrie entre les représentations statique et dynamique ainsi que la modularité et la réutilisabilité des spécifications. L'approche propose également une possibilité particulièrement originale de génération de code, en considérant comme cibles d'implantation des modèles de Bases de Données Actives. D'autre part, je me suis également intéressé au problème de la vérification du comportement modélisé. Cette composante de vérification propose différents niveaux de contrôles permettant de filtrer certaines anomalies ou erreurs de conception. L'idée est de pouvoir opérer de tels contrôles le plus tôt possible au cours du processus de développement et en particulier de garantir certaines propriétés des règles actives avant même leur génération.
Outre des publications dans un ouvrage MIT Press, une revue nationale (ISI) et des conférences internationales (VLDB, CAiSE) et nationales (BDA, INFORSID), ce travail, faisant toujours partie du contrat EERP avec Digital Europe, a permis notre implication dans un projet d'Action de Soutien Programmé (ASP) du GDR Bases de Données (1994-1996) dont l'objectif général était la validation des systèmes actifs comme outils d'implantation du comportement de Systèmes d'Information. Au cours de l'année 1996, j'ai également participé à un projet de recherche, mené par Z. Bellahsène au LIRMM (Montpellier), adoptant une démarche complémentaire d'IFO_2, pour la prise en compte des évolutions de Bases de Données. Cette approche s'inscrit dans le contexte des Bases de Données Orientées Objet et s'appuie sur le mécanisme des vues pour permettre d'éviter des réorganisations coûteuses de la base. Ce travail a donné lieu à une publication dans une conférence internationale (CAiSE). Une partie des travaux sur la partie comportementale a été réalisée dans le cadre du co-encadrement des DEA de C. Lallouche et de H.O Mohamed au Laboratoire d'Informatique de Marseille (LIM).

Modéliser pour rechercher
Nous nous sommes ensuite intéressés à un nouveau type d'application décisionnelle dite d'auto-surveillance visant au diagnostic et au traitement des pathologies des barrages. Ce travail consistait à représenter des scénarios de comportement, décrivant des comportements des objets de la base que l'on souhaite détecter au plus tôt. Une démarche générique couvrant les différentes étapes du processus de développement a été proposée pour permettre la modélisation des applications visées. Pour répondre à la problématique particulière du traitement des pathologies des barrages, une approche de conception spécifique

basée sur le modèle IFO_2 a été définie pour la représentation des scénarios.
Outre une publication internationale (CAiSE), ce travail entrait dans le cadre d'un projet mené avec le CEMAGREF (CEntre national de Machinisme Agricole du Génie Rural des Eaux et Forêts). Les travaux, réalisés dans le cadre de la modélisation de scénarios, ont fait l'objet du co-encadrement de la thèse de F. Cathala.

Extraction de connaissances et fouilles de données

Les travaux réalisés dans le cadre de la fouille de données font partie d'un projet plus général que je dirige au LIRMM dans l'équipe "Bases de Données - Systèmes d'Information" depuis 1996. L'objectif de ce projet est la définition d'un environnement d'extraction de connaissances.

Rechercher pour comprendre
Les travaux menés dans le projet consistent à découvrir, à partir des données descriptives du comportement d'objets dans le temps, des traits de comportements pertinents pour l'utilisateur final, i.e. des séries d'événements marquants. Ainsi, nous nous sommes particulièrement intéressés à la recherche de motifs séquentiels dans les grandes Bases de Données et un premier algorithme, appelé PSP, très efficace en temps de réponse, a été proposé aussi bien pour la recherche de règles d'association que de motifs séquentiels. A partir de ces travaux, une extension, appelée PSP+, a été réalisée dans le cadre du projet avec le CEMAGREF pour rechercher des motifs séquentiels introduisant la multi-occurrence d'événements, i.e. prenant en compte le nombre d'événements de même nature intervenus de façon parallèle.
Pour permettre de répondre complètement à certains types d'applications, une approche de gestion efficace des contraintes de temps liées à l'étude des comportements à court et à long terme a été définie. L'approche GTC permet ainsi, à l'aide d'un pré-traitement des données pour prendre en compte les contraintes de temps, de résoudre la problématique des motifs séquentiels de manière très efficace aussi bien du point de vue des temps de réponse que de l'adéquation des résultats aux objectifs de l'utilisateur final. En effet, la prise en compte de contraintes temporelles offre une analyse affinée des données.
Ces travaux ont donné lieu à publication dans une conférence internationale (PKDD) et une conférence nationale (BDA). Une partie de ces travaux a également donné lieu à un projet de transfert de technologies avec la société Albert-Inc. La définition d'une approche efficace de recherche de motifs séquentiels (PSP et GTC) rentre dans le cadre de l'encadrement du DEA de F. Masseglia. La définition de la partie multi-occurrence a été réalisée dans le cadre du co-encadrement de la thèse de F. Cathala.

De la même manière qu'une composante évolution doit être proposée pour un modèle conceptuel, il est indispensable pour une approche de fouille de données de proposer un tel mécanisme. Dans ce cadre, cependant, il ne s'agit plus d'offrir des mécanismes pour modifier un schéma qu'il soit structurel ou dynamique mais plutôt de prendre en compte l'évolution du contenu des bases de données au cours du temps. Les données manipulées (Bases de Données ou serveurs Web) évoluant sans cesse, il n'est pas concevable de recommencer entièrement le processus d'extraction. Il est donc nécessaire de proposer une approche efficace de recherche de motifs séquentiels capable de prendre en compte les connaissances extraites lors d'une précédente recherche. Une telle approche est appelée

incrémentale. Un algorithme, appelé ISE, a été développé afin d'optimiser la recherche de connaissances en calculant le minimum d'informations, i.e. les informations nécessaires pour que les connaissances extraites soient représentatives de la nouvelle base de données. Dans le contexte de ce travail, nous avons également transposé cette approche hors de son cadre incrémental. En découpant la base de données origine en une base et son incrément, nous avons montré au cours des évaluations qu'avec ISE, la recherche de motifs séquentiels pouvait, dans certains cas, être nettement optimisée.
Ces travaux ont donné lieu à une conférence internationale (PKDD) et une conférence nationale (BDA). Les travaux concernant la définition d'une recherche incrémentale rentrent dans le cadre du co-encadrement de la thèse de F. Masseglia.

Rechercher pour s'adapter
Pour compléter l'étude des algorithmes sur des applications réelles, nous nous sommes ensuite intéressés à la recherche de connaissances sur le Web. En effet, avec la popularité du Web, de très grandes quantités de données comme l'adresse des utilisateurs ou les URLs demandées sont automatiquement récupérées par les serveurs et stockées dans des fichiers Access Log. L'analyse de tels fichiers peut alors offrir des informations très utiles pour améliorer les performances du trafic sur le réseau, restructurer un site ou même cibler le comportement des clients dans le cadre du commerce électronique, etc. En appliquant des algorithmes de fouille de données sur les informations issues de tels serveurs (Web Mining), nous sommes à même de trouver les corrélations qui existent entre les différentes pages d'un serveur mais également les comportements typiques des utilisateurs du serveur. Dans ce cadre, nous avons réalisé un système, appelé WebTool, permettant de prendre en compte la recherche de motifs séquentiels et de règles d'association sur des données issues de serveurs Web. L'approche proposée est très originale car elle offre à l'utilisateur la possibilité de spécifier les différentes contraintes de temps qu'il souhaite afin d'affiner son analyse. Cet aspect n'existe pas dans les approches de Web Mining actuelles qui ne gèrent peu ou pas la prise en compte du temps. L'utilisateur final ne peut donc pas tirer profit des connaissances acquises (trop de règles, pas d'enchaînement temporel entre les règles, etc.). Conjointement à WebTool, une approche de modification dynamique de pages Web a été proposée : il s'agit d'analyser en temps réel le parcours d'un usager sur le Web pour adapter le contenu des pages du serveur afin d'optimiser le parcours et d'offrir l'information la plus en adéquation avec son intention (profil de comportement type).
Outre le système WebTool ces travaux ont donné lieu à deux revues internationales (NIS, ACM SigWeb), ainsi qu'à une conférence internationale (DEXA) et nationale (INFORSID). Dans le cadre de ces travaux, une collaboration avec le laboratoire d'Immuno-Génétique de l'Université de Montpellier II a été réalisée. Les travaux menés sur l'analyse de serveurs Web entrent dans le cadre du co-encadrement de la thèse de F. Masseglia.

Rechercher pour structurer
Depuis cette année, conjointement aux travaux menés précédemment, nous nous intéressons à la recherche de régularités structurelles dans des objets complexes. Ces travaux concernent aussi bien la recherche de régularités dans des transactions imbriquées que dans des ensembles de documents. Dans le premier cas, il s'agit de rechercher, dans une base de transactions, quelles sont celles apparaissant régulièrement. Cette composante est complémentaire de la composante comportementale du projet IFO_2 concernant la transformation vers des règles actives. Par exemple, en analysant des fichiers de transactions de

règles actives, cela permettrait d'assister la phase de rétro-conception de règles de manière à optimiser le code. Dans le second cas, l'analyse des pages sur un serveur Web, par des techniques de fouille de données, offre également la possibilité de déterminer quelles sont les structures sous-jacentes associées à un site. A l'heure actuelle, la plupart des documents "en ligne" (e.g. fichiers HTML, Latex, Bibtex, etc.) sont semi-structurés. Contrairement aux données structurées (telles que les bases de données objets ou relationnelles), les données semi-structurées n'ont pas de schéma parfaitement défini ou de classes fixées à l'avance et, de manière générale, chaque objet d'un serveur contient son "propre schéma". Cependant ces irrégularités n'impliquent pas qu'il n'existe pas de similarité structurelle entre objets semi-structurés. Il est même assez courant de trouver que des objets semi-structurés décrivant le même type d'information possèdent des structures similaires. A partir de la découverte de telles structures, il est alors possible de proposer une réorganisation du site et éventuellement de proposer une aide à la génération de balises XML afin de mieux décrire la structuration d'un document et ainsi faciliter son interrogation.
Outre deux publications internationales (PKDD, AIMSA), ce travail a permis notre implication dans le projet RNTL "CONTEXTE Bourse". Les travaux menés dans le cadre de la recherche de régularités rentrent dans le cadre de l'encadrement du DEA et de la thèse de P.A. Laur.

Bibliographie

[1] S. Agarwal, R. Agrawal, P. Deshpande, A. Gupta, and al. On the Computation of Multidimensional Aggregates. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB'96)*, pages 506–521, Bombay, India, September 1996.

[2] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD Conference*, pages 207–216, Washington DC, USA, May 1993.

[3] A. Albano, L. Cardelli, and R. Orsini. GALILEO: A Strongly-Typed, Interactive Conceptual Language. *ACM Transactions on Database Systems*, 10(2):230–260, 1985.

[4] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S.B. Zdonik. The Object-Oriented Database System Manifesto. In *Proceedings of the 1st Deductive and Object-Oriented Databases Conference (DOOD'89)*, pages 40–55, Kyoto, Japan, December 1989.

[5] K. Beyer and R. Ramakrishnan. Bottom-Up Computation of Sparse and Iceberg CUBEs. In *Proceedings of the Internation Conference on Management of Data (SIGMOD'99)*, pages 359–370, New-York, USA, September 1999.

[6] G. Booch. *Object-Oriented Design with Applications*. Benjamin/Cumming Comp, 1991.

[7] M. Bouzeghoub, G. Gardarin, and P. Valduriez. *OBJETS, Du C++ à Merise Object*. Eyrolles, 1994.

[8] M. Bouzeghoub and E. Métais. Semantic Modelling of Object-Oriented Databases. In *Proceedings of the 17th International Conference on Very Large Data Bases (VLDB'91)*, pages 3–14, Barcelona, Spain, September 1991.

[9] S. Chakravarthy, B. Blaustein, A. P. Buchmann, M. Carey, U. Dayal, D. Goldhirsch, M. Hsu, R. Jauhari, and al. HiPAC: A Research Project in Active, Time-Constrained Database Management. Technical report, Xerox Advanced Information Technology, Cambrige, MA, August 1990.

[10] P. Coad and E. Yourdon. *Object-Oriented Analysis*. Yourdon Press Computing Series, 1990.

[11] C. Collet. Bases de Données Actives : des systèmes relationnels aux systèmes à objets. Habilitation à diriger des recherches, Université Joseph Fourier, Grenoble, Octobre 1996.

[12] C. Collet, T. Coupaye, and T. Svensen. NAOS Efficient and modular reactive capabilities in an Object-Oriented Database System. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB'94)*, Santiago, Chile, September 1994.

[13] G. Colliat. Relational, and Multidimensional Database Systems. *ACM SIGMOD Record*, 3:64–69, 1996.

[14] O. Deux. The Story of O_2. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):91–108, March 1990.

[15] U.M. Fayad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, 1996.

[16] J. Han, S. Nishio, H. Kawano, and W. Wang. Generalization-Based Data Mining in Object-Oriented Databases Using an Object-Cube Model. *Data and Knowledge Engineering*, 25(1-2):55–97, 1998.

[17] D. Harel. On Visual Formalisms. *Communications of the ACM*, 31(5):514–530, 1988.

[18] R. Hull. Four Views of Complex Objects: A Sophisticate's Introduction. In *Proceedings of the Nested Relations and Complex Objects in Databases*, volume 361 of *Lecture Notes in Computer Science*, pages 87–116, 1989.

[19] N. Kettani, D. Mignet, P. Paré, and C. Rosenthal-Saboux. *De Merise à UML*. Eyrolles, 1998.

[20] R. Kimball. *Entrepôts de Données*. Thomson Publishing, 1997.

[21] Z. Manna and A. Pnueli. Specification and Verification of Concurrent Programs by $\forall$-Automata. In *Proceedings of the Temporal Logic in Specification Conference*, volume 398 of *Lecture Notes in Computer Science*, pages 124–164, Altrincham, UK, April 1987.

[22] Z. Manna and A. Pnueli. A Hierarchy of Temporal Properties. In *Proceedings of the 9th Annual ACM Symposium on Principles of Distributed Computing*, pages 377–408, Quebec, CA, August 1990.

[23] H. Mannila. Methods and Problems in Data Mining. In *Proceedings of the International Conference on Database Theory (ICDT'97)*, pages 210–215, Delphi, Greece, January 1997.

[24] H. Mannila, H. Toivonen, and A. I. Verkano. Discovering Frequent Episodes in Sequences. In *Proceedings of the 1st International Conference on Knowledge Discovery in Databases and Data Mining (KDD'95)*, pages 210–215, Montreal, Canada, August 1995.

[25] A. Mueller. Fast Sequential and Parallel Algorithms for Association Rules Mining: A Comparison. Technical Report CS-TR-3515, Department of Computer Science, University of Maryland-College Park, August 1995.

[26] P.A. Muller. *Modélisation objet avec UML*. Eyrolles, 1998.

[27] J. S. Ostroff and W. Murray Wonham. A Framework for Real-Time Discrete Event Control. *IEEE Transactions on Automatic Control*, 35(4):386–397, April 1990.

[28] J.S. Ostroff. *Temporal Logic for Real Time Systems*. Wiley and Sons, 1989.

[29] C. Rolland and C. Cauvet. Modélisation Conceptuelle Orientée Objet. In *Actes des 7ièmes Journées Bases de Données Avancées*, pages 299–325, Lyon, France, Septembre 1991.

[30] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.

[31] C. Sernadas and J. Fiadeiro. Towards Object-Oriented Conceptual Modeling. *Data & Knowledge Engineering*, 6:479–508, 1991.

[32] Rational Soft. *UML 1.1, Documentation*. http://www.rational.com/uml, 1998.

[33] J. Widom and S. Ceri. *Active Database Systems. Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann Publishers, 1996.

Première partie

Conception d'applications avancées

Introduction

Proposant à la fois une approche de modélisation structurelle et comportementale, l'originalité du projet IFO_2 est de les aborder en leur fixant les mêmes objectifs globaux et, en s'appuyant sur certaines similitudes, d'en uniformiser la vision.
Les concepts introduits dans les parties statique et dynamique du modèle sont symétriques avec, comme conséquence immédiate, l'uniformité offerte au concepteur pour la modélisation structurelle et la représentation du comportement.
Dans le projet CEMAGREF, nous étendons les concepts précédemment introduits pour s'adapter au contexte d'applications décisionnelles.

Modélisation structurelle

Sur le plan statique, les modèles sémantiques [71, 69] sont la référence pour évaluer la puissance d'expression mais, d'autre part, les modèles objet ont des abstractions et des qualités conceptuellement intéressantes tout en étant des cibles potentielles d'implantation. Il apparaît dès lors que la définition d'un modèle conceptuel structurel ne peut se justifier que combinant ces différents avantages. C'est une des ambitions de la partie statique d'IFO_2, qui étend le modèle sémantique IFO, défini par S. Abiteboul et R. Hull [4], et adopte une représentation "tout-objet" du réel.
Proposant la manipulation d'objets complexes via différents constructeurs, IFO_2 permet aussi une modélisation orientée attribut, jugée plus conceptuelle [69], et, contrairement aux autres modèles [17, 20, 43, 8, 97, 110, 112, 125], offre d'exercer modularité et réutilisabilité lors du travail de spécification.
Les capacités d'évolution proposées permettent de modifier un schéma conceptuel décrit à l'aide d'IFO_2 en garantissant sa cohérence après l'opération. Basés sur une approche par invariants, les contrôles mis en jeu assurent ainsi le filtrage d'anomalies de conception avant la dérivation du schéma "implantable" de l'application. Les dérivations structurelles, proposées dans IFO_2, visent des systèmes relationnels ou objet et exploitent les mécanismes offerts par ces systèmes cibles pour traduire toute la sémantique du schéma conceptuel, en particulier les contraintes applicatives donnant lieu, suivant le contexte, à la génération du code de méthodes de contrôle ou la définition de vues relationnelles.

Modélisation comportementale

Sur le plan comportemental, l'originalité d'IFO$_2$ est de poser les problèmes dans les mêmes termes que pour la modélisation structurelle et de viser les mêmes qualités de représentation. Ainsi, la vision globale des spécifications, offertes par certaines approches classiques mais pas forcément par les modèles les plus récents [17, 43, 112, 125], est privilégiée. Le comportement de l'application est représenté au sein d'un schéma dynamique. Mais le concepteur se voit aussi offrir une vision détaillée et une vision macroscopique de certains aspects comportementaux du schéma qu'il peut, à ce double niveau, manipuler comme un tout et réutiliser.
Prolongeant la philosophie "tout-objet" de sa partie structurelle, IFO$_2$ adopte une représentation "tout-événement" pour modéliser le comportement des applications. Des constructeurs événementiels sont définis pour exprimer des conditions de synchronisation permettant soit de contraindre l'émission de nouveaux événements, soit d'organiser des déclenchements combinés. Enfin, des liens nuancés d'enchaînements d'événements sont introduits pour traduire les relations de cause à effet dans les cascades de déclenchements.
Afin d'offrir les fonctionnalités requises pour manipuler les représentations élaborées, une composante d'évolution comportementale est proposée. Adaptant les possibilités et les contrôles structurels au contexte dynamique, elle renforce les vérifications opérées en contrôlant, non seulement la bonne utilisation des concepts dans un schéma, mais aussi que le comportement représenté a certaines qualités garantes du fonctionnement possible du système.

L'apparition des modèles de bases de données actives aura forcément à terme un impact sur les démarches conceptuelles. En effet, permettant de spécifier les circonstances d'exécution des opérations, en termes de règles Evénement-Condition-Action (E-C-A) [7, 15, 32, 31, 33, 48, 46, 50, 53, 70], ces modèles nous apparaissent comme des cibles naturelles d'implantation pour les approches comportementales et les SGBD actifs, relationnels ou objet, seront sans doute très rapidement les systèmes de mise en œuvre privilégiés par les démarches de conception. Dans ce cadre, la définition d'une composante de dérivation généralement absente des approches conceptuelles comportementales (à quelques exceptions près, [80, 64]), est tout aussi importante sur un plan dynamique que statique.

En offrant des mécanismes de vérifications du comportement modélisé, nous apportons également une réponse aux besoins de spécification de plus haut niveau qu'éprouvent les développeurs d'applications actives [23, 30, 46, 52, 135].

Modélisation de scénarios

L'objectif des applications décisionnelles que nous abordons est l'étude, au cours du temps, d'une ou plusieurs "populations" particulières dont les "individus" évoluent. Ces individus peuvent aussi bien être les clients d'une entreprise, les utilisateurs d'un service, mais aussi des mécanismes, des produits ou des constructions plus ou moins complexes. Ces individus ou objets sont décrits à travers un ensemble de propriétés intrinsèques de manière tout à fait analogue à celle des applications classiques. Cependant leur évolution au cours du temps n'a pas trait à la modification des propriétés statiques (comme c'est généralement

le cas dans les applications traditionnelles) mais à l'observation et à la préservation de leur histoire, i.e. de tous les faits significatifs qui sont survenus au cours de leur vie. Suivant la nature des objets considérés, ces faits peuvent être des événements (au sens usuel du terme), des phénomènes, l'observation de divers critères, etc. Le point commun est que ces faits s'inscrivent dans une dimension temporelle. La motivation essentielle dans la construction d'un tel historique, est de pouvoir contrôler l'évolution des individus, afin principalement de prévenir ou d'anticiper des situations délicates voire dangereuses, mais aussi de reconnaître, et pourquoi pas de gratifier, des évolutions exemplaires. Ainsi, contrairement aux entrepôts de données, exploités pour connaître les grandes tendances d'une population ou de son comportement, nous nous intéressons à l'observation individuelle des objets la composant. La modélisation des scénarios pose certaines difficultés au niveau conceptuel mais aussi en termes d'organisation de données et il s'agit d'offrir, dans les deux cas, une approche de représentation adaptée.

Organisation

Le projet IFO_2 est d'abord décrit dans sa dimension structurelle au chapitre 1 puis comportementale au chapitre 2. Au travers de ces deux chapitres, le projet IFO_2 est présenté dans sa globalité. Pour assurer l'homogénéité des présentations, un paragraphe concernant l'évolution comportementale (Chapitre 2, paragraphe 2.3.2) est proposé. Les travaux réalisés dans ce paragraphe ne rentre pas dans ma contribution mais dans celle des autres membres du projet. La modélisation de scénarios liée au projet CEMAGREF est décrite dans le chapitre 3.
Pour présenter les différentes contributions apportées, ces chapitres adoptent un même plan. Après un exposé détaillé de la problématique et un exposé de l'état de l'art du domaine, les contributions apportées sont détaillées et comparées aux autres travaux dans une discussion terminant chaque chapitre.

Un bilan des contributions est proposé en fin de partie. Celui-ci est suivi par une présentation de la liste des publications, des prototypes associés aux thèmes abordés ainsi que des conditions de la recherche pour les différents projets menés.

Pour illustrer les modèles présentés tout au long de cette partie, nous avons choisi un exemple inspiré de [62] qui consiste en un ensemble d'ascenseurs fonctionnant dans un même bâtiment.

Chapitre 1

Modélisation structurelle

Les objectifs que poursuivent les approches de conception actuelles n'ont guère variés depuis les modèles sémantiques. Il s'agit d'offrir à l'utilisateur une représentation la plus fidèle possible de l'univers réel modélisé, donc suffisamment riche en termes d'abstractions proposées et prohibant les choix a priori, i.e. indépendante d'un système d'implantation. Il s'agit de faciliter la perception de cette représentation en offrant une vision globale, éventuellement graphique, des spécifications élaborées. Il s'agit de rationaliser et fiabiliser le processus de développement d'application en proposant une dérivation de la représentation définie, si possible automatique, vers ses systèmes d'implantation [18].

Dans ce chapitre, nous nous proposons de détailler l'aspect modélisation à travers une vision purement structurelle. Avant de brièvement présenter les approches contribuant à la représentation statique (paragraphe 1.2), nous nous attacherons à expliciter la problématique posée en décrivant les objectifs d'un modèle conceptuel structurel (paragraphe 1.1) que nous avons adopté pour décrire les aspects statiques dans le projet IFO_2 dont une synthèse est proposée au paragraphe 1.3. Le bilan de la contribution est donné sous forme d'une discussion (paragraphe 1.4).

1.1 Exposé de la problématique

La Modélisation Structurelle

Parmi les approches conceptuelles, nous distinguons trois grandes tendances, selon qu'elles sont basées sur un modèle orienté valeur (ou enregistrements selon certains auteurs), sémantique ou objet.

Les modèles de la première tendance offrent une représentation du réel par l'organisation généralement simple, de valeurs, soumises à certaines contraintes d'intégrité [12, 44]. Le modèle relationnel en est l'exemple typique.
Dans un cadre conceptuel, ces modèles souffrent de plusieurs lacunes. Notons tout d'abord, des insuffisances sémantiques préjudiciables pour une représentation naturelle, aisée et la plus fidèle possible de la réalité : absence de concepts permettant de traduire les entités du réel et incapacité de spécifier certaines contraintes sémantiques, notamment de cardinalité, pour ne citer que deux exemples. De plus, la simplicité de l'organisation des données

ne permet la représentation que de structures "plates". Dans le modèle relationnel, par exemple, le respect des différentes formes normales signifie une multiplication du nombre de relations. En d'autres termes, l'utilisateur ne peut manipuler les données qu'en les reliant par des clés de jointure, pour lui, généralement abstraites. Ce point est d'autant plus critique que les objets représentés sont complexes.

Les extensions du modèle relationnel [1, 2, 73, 115], en autorisant une utilisation plus libre des constructeurs existants, n'ont pas, sur ces différents plans, apporté de solution satisfaisante [116]. Il est vrai que les modèles orientés valeur, bien que détournés par certains auteurs de leur vocation initiale, l'implantation, n'affichent pas véritablement d'ambition conceptuelle [69].

Les approches de la deuxième classe sont basées sur des modèles sémantiques [22, 71, 98, 140]. Leur principe est d'offrir des concepts suffisamment puissants pour obtenir la spécification la plus fidèle possible de l'univers modélisé. Contrairement aux précédents, ces modèles se veulent totalement indépendants de l'implantation des données [71, 8] et rejoignent, par la même, l'idée, devenue principe, selon laquelle ce n'est pas à l'utilisateur à s'adapter au système sous-jacent. Ainsi, il n'existe pas de distorsion majeure entre le schéma conceptuel élaboré selon ces modèles et l'univers du discours.
Par la représentation explicite des entités du réel, la spécification des contraintes sémantiques et la proposition d'abstractions suffisamment riches, comme la spécialisation ou la généralisation, ces modèles ont des qualités conceptuelles indéniables. Aujourd'hui pourtant, le développement des nouvelles applications fait apparaître certains de leurs défauts [111]. Si la vision globale des spécifications qu'elles permettent est un atout, il n'en demeure pas moins que la modularité de ces spécifications et la réutilisabilité des composants définis sont des qualités, essentielles lors de la conception, auxquelles ne peuvent pas prétendre tous ces modèles. De plus à travers, ces démarches le développement d'applications s'opère selon différents niveaux, entre lesquels les transitions peuvent être automatisées [66, 136]. Cette décomposition en plusieurs étapes, parfaitement adaptée aux modèles logiques classiques, peut être aujourd'hui perçue comme problématique. En effet, la philosophie de ces approches s'accorde mal à celle des nouveaux outils de développement que sont les Systèmes de Gestion de Bases de Données Orientés Objet (SGBDOO) [81].

De ce constat sont nées les démarches conceptuelles basées sur des modèles orientées objet [139]. Ces dernières préconisent une description des objets (unités de conception) qui encapsule les aspects structuraux et comportementaux [17, 43, 112, 125]. Leur objectif principal est d'associer, à la structure des entités de l'univers réel, les aspects dynamiques, voire à ne considérer, parfois, que ces derniers [28, 99, 113, 114].
A travers le paradigme objet [9], la modularité et la réutilisabilité deviennent effectives et la représentation d'objets de structures éventuellement très complexes est un atout majeur. Ces approches semblent donc très prometteuses pour une modélisation naturelle et aisée de l'univers réel. Pourtant, en offrant des concepts structurels souvent limités à ceux des modèles d'implantation, elles retombent dans le travers des modèles orientés valeurs. Elles préconisent notamment une représentation optimisée, i.e. exploitant au maximum les possibilités de construction de types alors qu'une modélisation orientée attribut est jugée préférable au niveau conceptuel [69]. Avec une approche orientée type, le concepteur se trouve obligé d'effectuer des choix de représentation a priori, occultant une partie de l'uni-

vers réel à modéliser. C'est ainsi que les modèles servant de base à des SGBDOO tels qu'O_2 [51], ORION [13], GEMSTONE [83], ... "préconisés" comme modèles conceptuels [69, 112], se sont avérés mal adaptés à un tel rôle. Les démarches orientées objet ont d'autres défauts dans notre contexte. Les contraintes structurelles y sont généralement exprimées sous forme de méthodes, l'antinomie entre codage et conception étant pourtant flagrante ! Les abstractions proposées ne sont pas aussi riches que celles des approches sémantiques. Par exemple, l'union de types, permettant la manipulation de structures hétérogènes [3, 72] n'est pas toujours offerte et quand elle l'est, ce n'est pas forcément avec les contraintes afférentes, d'exclusivité par exemple. Le recours aux méthodes, toujours possible pour pallier cette lacune souligne la dépendance par rapport à l'implantation. Enfin, soulignons que la vision globale du schéma de l'application ne peut, conceptuellement, être satisfaisante pour plusieurs raisons. Dans ce schéma, en effet, les objets sont présentés au sein d'une hiérarchie d'héritage, ne laissant pas transparaître les différentes constructions élaborées, telles que les agrégations ou les compositions, mais proposant par contre des classes primitives typiquement implantables [13, 59].

Pour conjuguer les qualités à la fois conceptuelles et objet, une démarche fréquemment suivie [19, 43, 8, 112, 97] est de s'appuyer sur un modèle sémantique classique, en en préservant les divers avantages, et de l'étendre pour intégrer les concepts objet. Cette intégration peut se faire en respectant la philosophie du modèle sémantique sous-jacent. Le concept de classe est alors adopté comme unité de description structurelle, avec l'apport immédiat de pouvoir spécifier les entités conceptuelles à travers à la fois leur structure et leurs opérations.

La Dérivation

Une condition sine qua non pour justifier de la définition d'un nouveau modèle conceptuel, outre ses qualités et sa puissance de représentation est qu'il soit dérivable, si possible de manière automatique, vers un système cible. Pour optimiser et fiabiliser le développement, des transformations ou dérivations doivent donc être définies [18, 112]. Il s'agit donc d'exploiter toutes les capacités des modèles cibles pour conserver la plus grande richesse à la représentation.

L'Evolution Structurelle

Enfin, complétant les possibilités de modélisation et de dérivation, des capacités d'évolution doivent permettre de modifier un schéma conceptuel. Le concepteur doit pouvoir rectifier, de manière incrémentale, ses spécifications ou les faire évoluer conjointement aux changements du réel. De telles modifications structurelles sont peu étudiées au niveau conceptuel alors que le caractère itératif du processus de conception les rend indispensables. Elles sont par contre offertes par les différents systèmes existants pour maintenir l'application implantée [13, 45, 77, 79, 83, 88, 133, 141].

Les différents aspects que nous venons de présenter concernant les attentes d'un modèle conceptuel ne sont, en fait, jamais entièrement prises en compte dans les diverses approches proposées. C'est ce que nous tentons de mettre en évidence dans le survol de l'état de l'art

suivant.

1.2 Aperçu des travaux antérieurs

Nous nous limiterons à l'examen des approches les plus récentes dont la préoccupation est de concilier qualités classiques et objet. Ces démarches peuvent être considérées à la fois comme sémantiques et orientées objet. Nous donnons, ci-dessous, un aperçu général de ces approches objet en ne retenant que les plus significatives parmi la pléthore de propositions (une cinquantaine de méthodes selon [89]). Une synthèse plus complète, englobant modèles sémantiques et orientés valeur, est donnée dans [101]. [69] et [71] proposent un panorama complet des modèles sémantiques et [19] présente différentes méthodes de conception orientées objet. Enfin, en nous intéressant à UML [89], nous tirons profit des objectifs généraux fixés pour la démarche, à savoir une unification des approches antérieures les plus probantes et les plus utilisées.

Des modèles, comme OMT [112] et OOA/OOD (Object Oriented Analysis and Design) [43], choisissent d'étendre le classique modèle Entité-Relation, en substituant le concept de classe (avec ses attributs et méthodes) à celui de type d'entités et en introduisant de nouvelles capacités de représentation. Des possibilités d'agrégation de classes (similaires à des associations mais dotées d'une sémantique "Composant/Composé") sont notamment proposées. OMT distingue les abstractions de généralisation et spécialisation (avec disjonction ou intersection des sous-classes) mais en propose une seule représentation et permet une prise en compte explicite de contraintes liées aux cardinalité, domaine, clefs candidates, conditions sur les associations, éléments dérivés, etc. Par rapport à OMT, OOA/OOD introduit une spécificité : la représentation des envois de message dans le modèle objet. Par contre, seules des associations binaires peuvent être représentées et les contraintes explicites sont moins nombreuses que dans OMT.
L'approche OOA proposée par S. Schlaer et S. J. Mellor [120, 121] reste dans l'intégration des concepts objet en deçà des deux modèles précédents puisque le principe d'identification des objets n'est pas retenu et les méthodes ne sont pas spécifiées dans les classes. Des objets associatifs permettent la représentation des attributs d'association binaire avec la particularité de pouvoir être réutilisés et raffinés par héritage.

Ces approches ne remettent pas véritablement en cause la philosophie du modèle Entité-Relation ni les principes de modélisation du concepteur. Par exemple, une propriété complexe d'une entité réelle est représentée par une classe, sous la même forme que l'entité elle-même. Il n'y a donc pas véritablement de manipulation d'objets complexes, mais manipulation de plusieurs objets de sémantique différente et des divers liens de construction entre eux. Le principal inconvénient est que ces divers éléments ne forment pas un tout, manipulable tel quel.

Ces approches inscrivent la définition d'une composante de modélisation structurelle dans un cadre méthodologique, mais, plutôt qu'une conception véritablement objet, elles ont évolués vers l'harmonisation de leurs concepts avec ceux du niveau implantable. Elles offrent d'ailleurs une implantation automatique des spécifications en utilisant non seulement les langages ou SGBD classiques, mais aussi objet.

Egalement basés sur le modèle Entité-Relation, ERC+ [97, 8] et OOM [19] permettent la manipulation d'objets complexes par la définition de propriétés composites pour les entités.

La méthode OOD de G. Booch [17], dont la vocation est le développement plutôt que la conception et le domaine d'application le génie logiciel plutôt que les bases de données, propose une description de l'application en termes de classes associées par des liens d'héritage, d'instanciation, de méta-classes et d'utilisation. Ces derniers, explicitant des envois de messages, peuvent être différenciés selon que l'objet utilise la spécification d'un autre objet dans son interface ou son implantation. Des cardinalités leur sont attribuées. Cette approche ne propose pas de représentation explicite pour les objets composites qui peuvent être pris en compte via plusieurs classes associées par des liens d'utilisation.

Basé sur les réseaux sémantiques, MORSE propose une perception uniforme du réel en termes de types d'objets, pouvant être atomiques ou moléculaires [20, 18]. L'organisation de ces types d'objets met en jeu différents constructeurs et offre une vision structurelle globale de l'application. La dérivation de spécifications MORSE vers un modèle objet, en l'occurrence O_2, s'appuie sur des équivalences naturelles entre concepts. Elle permet, en particulier, de traduire des objets atomiques MORSE soit sous forme d'objets O_2, soit sous forme de valeurs, selon le contexte, et assure la prise en compte des contraintes spécifiées, par génération du code des méthodes de contrôle.

Proposant un consensus sur les concepts objet, UML est né de l'unification et l'intégration de plusieurs propositions précédentes, en particulier OMT et l'approche de G. Booch. UML propose les concepts objets déjà évoqués et un ensemble de notations graphiques dans la même philosophie que celles des diagrammes de classe d'OMT mais avec quelques enrichissements (nature plus précise des opérations, directions des agrégations, concept de paquetage de conception, etc.).

Avec des philosophies parfois différentes, les modèles évoqués dans ce paragraphe offrent un pouvoir d'expression relativement comparable, incluant au moins des possibilités de spécialisation, agrégation et l'expression d'associations sémantiques entre objets. Certains modèles permettent une meilleure prise en compte des contraintes en proposant une plus grande richesse d'expression et la production de programme de contrôle. Cependant ces approches ne distinguent généralement pas spécialisation et généralisation (contrairement à ce que préconisent les modèles sémantiques) et ne proposent pas de constructions exclusives explicites (offertes par certains modèles objet). Soulignons enfin que les modèles véritablement objet ont l'avantage de permettre une représentation plus uniforme de l'application mais également plus fidèle, notamment par une représentation d'entités de structure arbitrairement complexe.

1.3 Synthèse des contributions

En tentant de définir une nouvelle approche conceptuelle, notre objectif général a été d'offrir un cadre complet et rigoureux au développement d'applications, à travers la définition

formelle de trois composantes structurelles : Modélisation, Evolution et Dérivation.

Sur le plan de la modélisation, notre ambition est de conjuguer les qualités des approches traditionnelles et des modèles objet. C'est sur cette hypothèse que s'appuie l'approche que nous proposons : une démarche de conception structurelle basée sur le modèle IFO_2, extension du modèle IFO [4] défini par S. Abiteboul et R. Hull.
Le choix de ce dernier mérite quelques éléments de justification. Tout d'abord IFO offre les avantages, précédemment évoqués, des modèles sémantiques : représentation explicite des entités réelles ; proposition d'abstractions de haut niveau qui si elles ne sont pas tout à fait complètes dans un cadre conceptuel constituent cependant une base solide et extensible ; possibilité de spécification des contraintes structurelles et enfin vision globale du schéma conçu. Un autre avantage de ce modèle, suffisamment rare dans les approches sémantiques pour être fortement souligné, est la rigueur de sa définition qui évite toute ambiguïté dans la structuration élaborée. De plus IFO peut être perçu comme précurseur des modèles objet. Il se dote, en effet, de concepts assurant une très grande modularité des schémas et de constructeurs devenus classiques dans ces modèles. Il se situe donc dans la droite ligne que nous nous sommes fixée.

En adoptant cette base de travail pour la composante Modélisation d'IFO_2 [105], notre objectif est de conserver la puissance de représentation d'IFO voire de la renforcer sur certains points, tout en respectant le paradigme objet. Ainsi, IFO_2 redéfinit l'ensemble des concepts pour permettre une prise en compte explicite de la notion d'identifiant et étend le pouvoir d'expression d'IFO en introduisant de nouveaux constructeurs d'objets composites (alternative, composition et groupement) et en enrichissant la liste des contraintes sémantiques explicitement spécifiées.

L'approche de conception structurelle que nous proposons à travers le projet IFO_2 [103, 129, 100, 39] s'articule selon les trois composantes dont nous venons de tracer les objectifs et dont un résumé est proposé à travers les paragraphes suivants.

1.3.1 Composante Modélisation

Comme d'autres modèles, IFO_2 propose une vision uniforme du réel en adoptant une approche "tout-objet". Tout élément stucturel descriptif de la réalité, qu'il soit élémentaire ou complexe, est, en IFO_2, modélisé comme un objet, doté d'un identifiant et d'une valeur. Cette valeur peut être simple ou structurée suivant le *type d'objets* considéré, *de base* ou *complexe*. Les types d'objets peuvent être organisés et reliés entre eux selon un double niveau d'abstraction, i.e. au sein d'un *fragment* puis d'un *schéma*, qui offrent respectivement une vision parcellaire et globale des spécifications.
Les concepts évoquées de type d'objets, fragment et schéma, successivement utilisées pour une conception allant du "plus détaillé" au "plus global", sont présentées ici de manière intuitive et illustrée. Les définitions formelles du modèle sont données dans [101, 107].

Les types de base du modèle, dont la représentation graphique est donnée par la figure 1.1, sont les suivants :

- le *Type d'Objets Imprimable (TOI)* qui représente les entrées/sorties de l'application,

donc les éléments d'information matérialisables. Un TOI correspond à la notion de type d'attributs dans le modèle Entité-Relation [132] ;

- le *Type d'Objets Abstrait (TOA)* modélisant un type d'entités identifiable du monde réel qui n'est pas décrit par sa structure interne mais par les propriétés qui lui sont spécifiées. Il peut être comparé à la notion de type d'entités dans le modèle Entité-Relation. Les valeurs des objets de type abstrait sont nulles car les objets reflètent simplement des entités réelles ;
- le *Type d'Objets Représenté (TOR)* qui symbolise n'importe quel autre type d'objets (de base ou complexe) décrit par ailleurs dans les spécifications. Il peut être manipulé comme un tout et sans en connaître la description précise via un type représenté. La valeur des objets d'un TOR dépend bien sûr du type d'objets symbolisé.

Exemple 1 Considérons un système de contrôle d'un ensemble d'ascenseurs fonctionnant dans un même bâtiment. Dans ce contexte, parmi les types de base identifiés (C.f. figure 1.1), "Position" est un type imprimable indiquant l'étage actuel de la cabine d'un ascenseur. "Cabine", reflètant une entité du réel, est modélisée par un type abstrait. Ce dernier peut aussi être manipulé via le type représenté "Cabine-Ascenseur" introduit pour le symboliser.

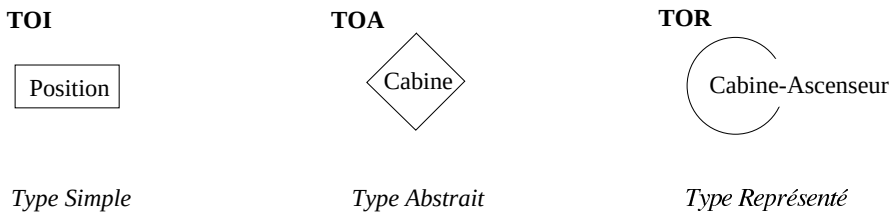

FIG. 1.1 – *Exemples de types de base*

Afin de permettre la spécification d'objets complexes, IFO_2 propose cinq constructeurs qui peuvent être appliqués, de manière récursive, sur les types de base et introduit une distinction entre constructions exclusives ou non. Ces constructeurs sont les suivants :

- l'*agrégation* et la *composition* correspondent à l'abstraction agrégation des modèles sémantiques [71] traduit par le constructeur tuple des modèles objet. Le domaine de ces types est défini comme le produit Cartésien de l'ensemble des objets composants. Par rapport à l'agrégation, la composition est enrichie d'une contrainte d'exclusivité permettant de vérifier que chaque objet d'un type composant ne participe qu'à une unique construction ;
- la *collection* et le *groupement* correspondent au constructeur set-of des modèles objet : une construction est un ensemble d'objets d'un même type. Le groupement permet une construction exclusive ;
- l'*alternative* (union) permet de manipuler indifféremment des types d'objets structurellement différents. Ce constructeur représente l'abstraction généralisation et permet de prendre en compte les types d'objets à structure variable [72] car il est défini comme l'union des structures des types le composant (qui doivent être disjoints). Le

type ainsi défini permet de remplacer à un niveau supérieur n'importe lequel de ses composants.

En utilisant ces constructeurs et les types de base, des types d'objets arbitrairement complexes peuvent être construits. De manière générale, un type est un arbre identifié par sa racine et dont les feuilles sont des types imprimables ou représentés et les nœuds des constructeurs.

Une double vision extensive des types d'objets est proposée à travers les notions d'instance et d'objets attachés. L'instance d'un type décrit l'ensemble des objets de ce type à un instant donné. Les objets attachés explicitent quant à eux, pour un type et son instance, l'ensemble des objets qui en sont effectivement composants. Ils sont déterminés pour chaque sommet d'un type.

Exemple 2 Au cours de la description de la composition matérielle d'un ascenseur, le concepteur veut pouvoir spécifier un élément de type "Moteur" comme étant constitué d'un axe et d'un ensemble de générateurs, chacun d'entre eux comprenant une bobine et un aimant. Reflétant cette description, le type composite "Moteur" (C.f. figure 1.2) est défini comme la composition d'un type imprimable, "Axe", et du type "Générateurs", lui-même construit comme un groupement de "Générateur". Ce dernier type est modélisé comme une composition de deux types imprimables, "Bobine" et "Aimant".
Les constructeurs utilisés étant tous exclusifs, chaque objet composant ne peut être utilisé que dans la construction d'un seul moteur.

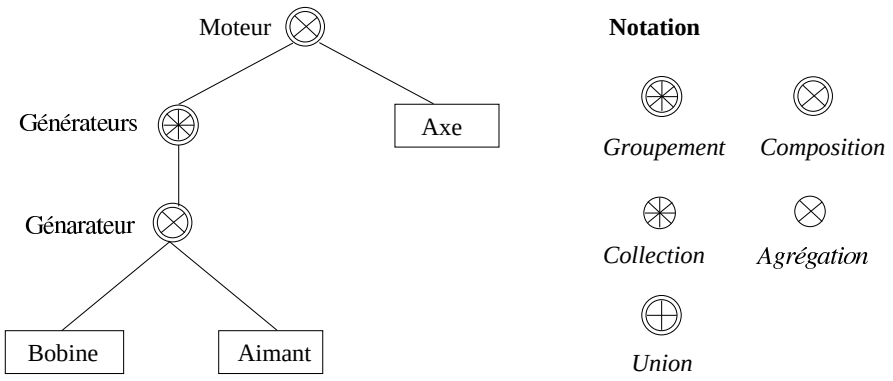

FIG. 1.2 – *Le type complexe "Moteur"*

Les types d'objets ayant été décrits, il convient d'affiner les liens qui peuvent les réunir, en dehors de leur participation à une même construction. Pour décrire un aspect précis de la réalité, les types d'objets sont organisés au sein de fragments et reliés entre eux par des

fonctions. La vocation d'un fragment est de spécifier complètement une portion du réel en se focalisant sur un type d'objets central, appelé le cœur du fragment, et en lui attachant toutes les propriétés devant être prises en compte selon ce point de vue. Les fonctions, permettant ce rattachement, expriment aussi des contraintes de cardinalité en combinant les caractéristiques suivantes : simples ou complexes (multivaluées) et partielles ou totales (obligatoires).
Il est possible de rattacher des combinaisons de propriétés, entre lesquelles existent des liens sémantiques, au cœur d'un fragment, à travers la notion de sous-fragment : un type d'objets descriptif du cœur peut lui-même être décrit par des propriétés. Il est également possible de spécifier des fonctions inverses permettant d'exprimer des contraintes de dépendance fonctionnelle.
Le concept de fragment, directement hérité d'IFO, est essentiel car il est la base de la modularité de spécification dans IFO_2. Pouvant se centrer sur une entité réelle ou sur une construction plus artificielle, il symbolise des unités conceptuelles dont la spécification, détaillée par un concepteur, peut être rendue transparente aux autres. Ces unités conceptuelles ont un comportement, que nous décrivons dans le chapitre suivant, et réalisent des opérations de base. Ces méthodes, spécifiées au sein du fragment, se réduisent dans notre contexte à une signature dont les types participants appartiennent nécessairement à ceux du fragment concerné.
De la même manière que pour les types d'objets, nous introduisons les notions d'instance d'un fragment, associant chacun des objets du cœur aux objets qui le décrivent, et d'objets attachés pour tout sommet d'un fragment.

Exemple 3 Une portion de l'application exemple est proposée à travers le fragment de la figure 1.3. Cette vision se focalise sur une entité réelle, représentée par le TOA "Cabine", cœur du fragment, qu'il s'agit de complètement décrire. Une cabine est notamment caractérisée par sa position, propriété obligatoire et monovaluée, et par l'ensemble des étages qu'elle peut desservir en utilisant ses différents boutons de sélection. Pour prendre en compte ces informations, le cœur du fragment est lié au TOI "Position" par une fonction totale et simple et au type "Etage-Cabine" via une fonction totale et complexe. Nous imaginons ici que le type "Etage-Cabine" symbolise un type "Etage" dont la description précise n'est pas encore donnée. Pour une cabine, chaque étage est doté d'un bouton de sélection. Cette information est représentée au sein d'un sous-fragment en associant le TOI "Bouton", via une fonction simple et totale, au type "Etage-Cabine", cœur du sous-fragment. Ainsi, chaque objet de "Cabine" est associé à un ensemble de couples, objets de "Etage-Cabine" et "Bouton". Des exemples d'instances de types sont précisés sur le schéma.
Parmi les méthodes du fragment "Cabine", "Fermeture" et "Ouverture" permettent respectivement de fermer et d'ouvrir les portes de la cabine.

Pour obtenir une représentation complète de l'application, les vues partielles que fournissent les fragments sont réunies au sein d'un schéma structurel IFO_2, en utilisant des liens IS_A. Plus précisément, ces liens associent un type représenté dans un fragment au type d'objets qu'il symbolise et qui doit être cœur d'un autre fragment. Par ce biais, le fragment est réutilisable comme un tout.
Les concepts de fragment, type représenté et lien IS_A proposent une modularité et réutilisabilité à un niveau de description plus global qu'une classe ou type d'objets, donc plus

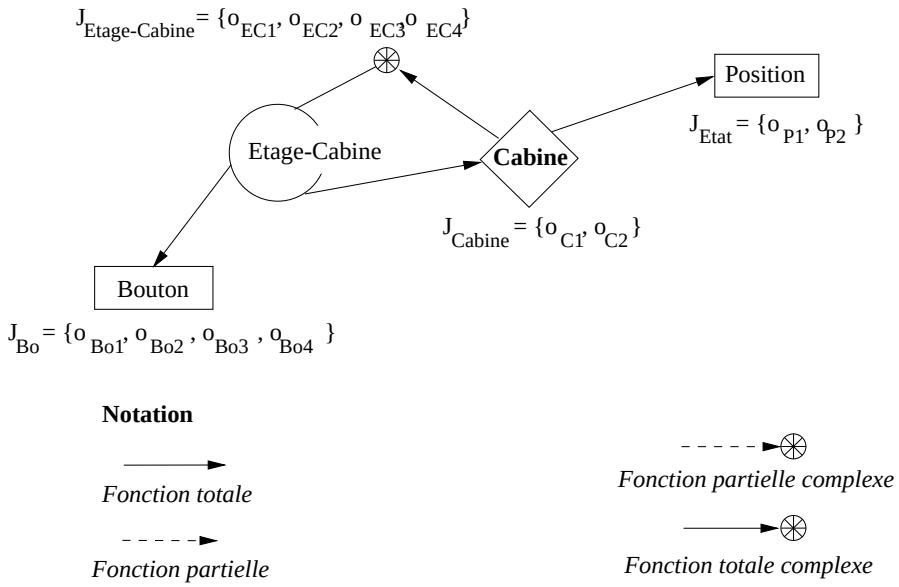

FIG. 1.3 – *Le fragment "Cabine"*

fonctionnel dans un cadre conceptuel. Les liens IS_A permettent de représenter des liens d'utilisation de type client/fournisseur [87] ou de spécialisation.
La différence entre ces deux rôles n'est pas soulignée par un formalisme différent simplement car elle est explicite dans la représentation du schéma. En effet, dans le cadre d'une spécialisation, le type représenté raffinant le type initial est forcément le cœur de son propre fragment puisqu'il doit lui-même posséder des propriétés descriptives. S'il n'est pas le cœur, il représente une propriété qui "est un" objet du type symbolisé. Dans les deux cas, un même type représenté peut être associé à plusieurs types cœurs (héritage ou utilisation multiple, ou combinaison des deux).
Outre les contraintes sur les sources et cibles de lien IS_A, la construction d'un schéma interdit les cycles de liens IS_A (pour éviter les interblocages de référence) et impose, si un type hérite de plusieurs autres, l'existence d'une source commune d'héritage (direct ou via un chemin de liens IS_A).
L'instance d'un schéma est alors définie comme l'union des instances des fragments associés, ces instances devant respecter la propagation des propriétés (les objets attachés du cœur) via les liens IS_A (le type représenté hérite ainsi de la description du type origine).
L'ensemble des méthodes associé au schéma est défini par l'union de l'ensemble des méthodes de chacun des fragments qui le composent.

Exemple 4 La figure 1.4 propose une représentation simplifiée de l'application "Ascenseurs", en réunissant trois visions parcellaires, les fragments "Ascenseur", "Etage" et "Cabine", déjà décrit dans l'exemple précédent.
Dans le fragment "Ascenseur", des propriétés factuelles plus ou moins complexes, comme la référence ou la charge, décrivent les entités "Ascenseur", qui sont aussi caractérisées par leurs composants physiques, notamment le moteur et la cabine. Cette dernière est symbolisée par le type représenté "Cabine-Ascenseur" qui référence "Cabine" via un lien IS_A.

Le troisième fragment est consacré à la description des étages, en supposant que pour l'application, les seuls éléments d'information utiles sont le numéro de l'étage et le bouton d'appel qui y est situé. Le cœur de ce fragment, le TOA "Etage" est réutilisé dans le fragment "Cabine" via le type représenté "Etage-Cabine" et un lien IS_A les reliant.
"Montée", "Descente", "Arrêt" et "Arrivée" sont les méthodes du fragment "Ascenseur". Les deux premières permettent de réaliser un mouvement ascendant ou descendant de la cabine, la troisième correspond à l'arrêt du moteur de l'ascenseur et enfin, la dernière renvoie le numéro de l'étage atteint par la cabine.

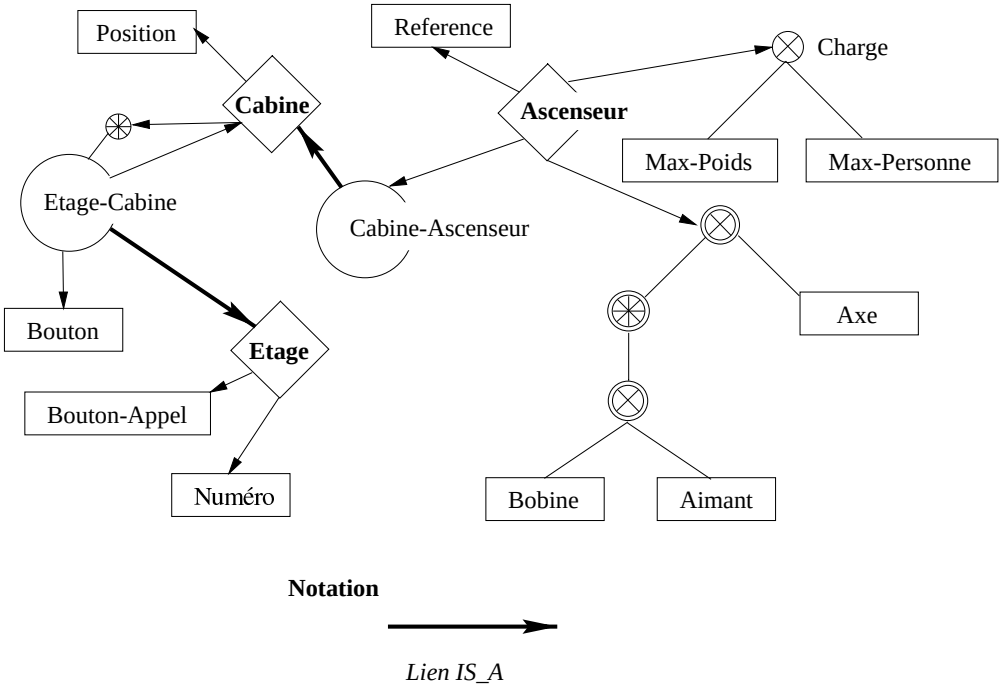

FIG. 1.4 – *Schéma IFO_2 de l'application "Ascenseurs"*

A travers ses trois niveaux d'abstraction successifs, de type d'objets, fragment et schéma, la composante de modélisation d'IFO_2 permet tour à tour une vision détaillée, parcellaire et globale de la description structurelle d'une application. Ces concepts confèrent au modèle des qualités sur lesquelles nous revenons dans la discussion du paragraphe 1.4, mais auparavant, nous les examinons sous l'angle de l'évolution puis de la dérivation.

1.3.2 Composante Evolution

Pour offrir une grande souplesse tout au long du processus de conception, les possibilités d'évolution doivent être complètes et contrôlées. Il importe donc d'une part de recenser les fonctionnalités attendues et d'autre part de leur fixer un cadre d'application précis. Sur ces deux points, nous nous inspirons des travaux réalisés pour différents systèmes comme OTG [79], ORION [13, 78], GEMSTONE [83], SHERPA [90], MORE [134], ESSE [45, 141]

et COCOON [133].

Nous basons notre approche sur un ensemble d'invariants que doit respecter un schéma IFO_2 pour être cohérent. Ces invariants régissent la construction des types d'objets (contrôle de cardinalité et de la nature des composants), des fragments (vérification de la nature des types y participant) et du schéma lui-même. La liste de ces invariants est donné dans [102, 101].

D'un point de vue général, certaines modifications sont très simples, comme la suppression d'un type imprimable, propriété descriptive d'une entité, alors que d'autres conduisent à une réorganisation complète du schéma, une modification sur un fragment pouvant se propager aux autres. La question est donc de proposer un ensemble de primitives minimal dont la combinaison permette d'exprimer les mises à jour. En s'inspirant des taxonomies proposées dans un contexte implantable [13, 45, 77, 79, 83, 88, 133, 141], une première proposition d'opérations de mise à jour est formalisée dans [102, 101], englobant des primitives sur les types d'objets (création, suppression, modification, substitution), sur les fonctions de fragment (création, suppression, modification des caractéristiques) et sur les liens IS_A (création, suppression).

Ces opérations de base sont complétées dans [101] par la définition des règles de mise à jour qui explicitent les séquences de primitives nécessaires pour des manipulations plus globales et bien sûr prennent en compte les propagations de modifications. L'application de ces règles est également soumise au respect a priori des invariants.

Exemple 5 Si le concepteur désire modifier le schéma IFO_2 de la figure 1.4, les possibilités qu'il a sont restreintes aux seules opérations valides. Par exemple, la destruction du type imprimable "Max-Poids" est interdite car elle conduit à n'avoir plus qu'un seul composant pour le type agrégation "Charge". Par application de règles de mise à jour, une telle situation est traitée par propagation de la destruction au type composite et le rattachement direct du dernier composant "Max-Personne" au cœur du fragment.

La définition des invariants structurels et des primitives de mise à jour est le support formel sur lequel peuvent s'appuyer les fonctionnalités offertes par un outil d'aide à la conception, puisque l'élaboration du schéma se traduit par une séquence de primitives.

1.3.3 Composante Dérivation

La composante de dérivation propose la transformation de spécifications structurelles IFO_2 vers deux modèles cibles représentatifs des systèmes actuels : le modèle relationnel et le modèle O_2 [51]. Les fonctions de transformation définies sont présentées dans [106, 101].

La transformation permettant de générer un schéma O_2 à partir de spécifications IFO_2 suit un principe analogue à celui de MORSE. Elle traite individuellement chaque fragment, pour engendrer au moins une classe O_2, généralement plusieurs. Même si elle procède

par fragments, la transformation doit évidemment tenir compte des liens existants entre fragments, ce qui est fait au cours de la traduction des types représentés. Les liens entre fragments IFO_2 sont des liens IS_A de spécialisation ou d'utilisation, ils sont traduits par des liens entre classes O_2 pouvant être soit des relations d'héritage entre classes, soit des références. La différence tient en fait à la position du type représenté au sein de son propre fragment. Soit il y est cœur, il affine donc la description proposée par le type qu'il symbolise et est alors traduit par une classe héritant de celle reflétant le type symbolisé (spécialisation). Soit le type représenté est une propriété descriptive d'un cœur, sa traduction est alors opérée par référence à la classe du type symbolisé (utilisation).

Exemple 6 La traduction en O_2 du fragment "Ascenseur" (C.f. figure 1.4) génère quatre classes. La première "Ascenseur" possède un attribut simple ("Référence"), un attribut de type tuple ("Charge" composée de "Max-Poids" et "Max-Personne") et deux attributs "Moteur" et "Cabine" référençant d'autres classes. Les trois autres classes créées correspondent respectivement aux types imprimables "Axe", "Bobine" et "Aimant".
En fait, l'attribut "Moteur" de la classe "Ascenseur" est composite et est construit comme une combinaison de références aux objets de "Axe", "Bobine" et "Aimant".
Quant à la référence introduite par "Cabine", elle ne fait que refléter l'utilisation d'un type d'objets à structure complexe comme propriété descriptive de "Ascenseur".

Complétant cette possibilité d'implantation objet, une transformation relationnelle de schémas IFO_2 est définie dans [101].

1.4 Discussion

IFO_2 peut être qualifié de modèle sémantique car il intègre les principales caractéristiques des approches sémantiques. Il peut également être perçu comme un modèle fonctionnel dans la mesure où les liens entre types peuvent être représentés par l'intermédiaire de fonctions. Enfin, il peut être considéré comme objet puisqu'il définit explicitement le concept d'identifiant d'objet indépendant de sa valeur. Avec cette philosophie du "tout-objet", tout élément du modèle est doté d'un identifiant indépendant de sa valeur. Ce choix est, pour les modèles conceptuels, une originalité dans la mesure où la distinction classique entre une entité qui peut être identifiée et un attribut de l'univers réel n'est pas reflétée dans la représentation IFO_2 associée. Cette particularité n'est pas en elle-même gênante puisque le concept d'identifiant, relevant d'un niveau plus logique où il s'agit d'obtenir une représentation optimisée des données, est totalement transparent. Ce choix impose cependant une redéfinition des concepts du modèle IFO dont la formalisation perd sa simplicité initiale. Mais, lors de l'extension d'IFO, la notion d'identifiant est essentielle pour toute construction d'objet, en particulier celle d'objet composite où la contrainte d'exclusivité ne peut, en aucun cas, se satisfaire d'une identification par valeur.

Nous conservons les trois types d'objets atomiques définis dans IFO et introduisons les trois abstractions supplémentaires suivantes : la composition, le groupement et l'alternative

(union), comme constructeurs de types d'objets complexes. Ainsi, nous offrons la possibilité de définir "naturellement" et graphiquement les objets composites. Via le constructeur union, le concept de généralisation des modèles sémantiques est explicitement représenté. Il permet d'éviter les problèmes de mise à jour inhérents à la généralisation sans disjonction décrits dans [4] et d'offrir à l'utilisateur une abstraction non ambiguë. La notion de fragment est également étendue en intégrant les fonctions inverses afin de pouvoir spécifier, encore plus finement, certaines contraintes du réel. Pour expliciter cet atout, nous en proposons l'illustration suivante. Pour modéliser les relations entre entités, certains modèles imposent un choix de représentation souvent déterminé par l'utilisation ultérieure des entités et occultant une partie de l'univers réel. Par exemple, pour représenter le fait "qu'un usager utilise un ascenseur et qu'un ascenseur est utilisé par un usager", la décision de représentation privilégie l'une ou l'autre des entités, soit "Usager" : l'ascenseur utilisé est alors un attribut d'"Usager" (Figure 1.5 cas b), soit "Ascenseur" : l'usager est alors vu comme une propriété d'"Usager" (Figure 1.5 cas a). En intégrant le lien inverse, IFO_2 permet au concepteur, quelle que soit la vision choisie, de spécifier complètement la relation considérée.

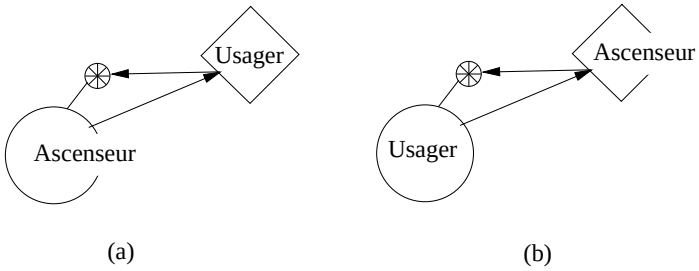

FIG. 1.5 – *Utilisation du lien inverse*

Cependant, notons qu'IFO_2 ne contraint pas le concepteur à privilégier l'une des entités participant à une association. Il peut, en effet, modéliser cette dernière via le constructeur d'agrégation. De plus, à l'intérieur d'un fragment IFO_2 les types d'objets sont reliés entre eux par l'intermédiaire de fonctions simples ou complexes mais en leur attribuant la possibilité d'être partielles ou totales, nous enrichissons la sémantique des contraintes de cardinalité offertes par IFO.

Le niveau d'abstraction supplémentaire introduit par les fragments est un avantage d'IFO_2 comparé aux approches conceptuelles proposées. Les fragments, en organisant des types d'objets sémantiquement liés, permettent de réutiliser des portions entières de spécifications. Complétant les qualités équivalentes des types d'objets, ils offrent modularité et réutilisabilité à un niveau mieux adapté au cadre conceptuel. Ils peuvent bien sûr contribuer à la clarté de la représentation graphique. La vision globale du schéma est simplifiée en considérant chaque fragment comme une "boîte noire" dont un zoom est offert par la représentation détaillée du fragment (une telle fonctionnalité est d'ailleurs offerte dans le système IFO_2). Ces qualités de présentation sont intéressantes car elles atténuent la rapide illisibilité de spécifications graphiques pour de grandes applications d'autant que des spécifications IFO_2 peuvent sembler plus difficiles à appréhender que d'autres représentations, essentiellement du fait de la richesse des concepts (mais cette richesse est aussi la base d'une représentation fidèle et complète).

Nous pouvons aussi souligner le fait qu'IFO_2 s'appuie aussi bien sur des éléments qui peuvent être qualifiés de conceptuels comme le fragment et les types représentés que sur des éléments implantables comme les identifiants d'objets. Il est ainsi accessible à la fois aux concepteurs novices et expérimentés car il autorise aussi bien une modélisation conceptuelle (orientée attribut) qu'optimisée (orientée type).

La définition des concepts structurels, base formelle de l'approche, permet notamment d'expliciter les règles de cohérence de la représentation. Leur contrôle est mis en œuvre à travers la composante Evolution qui formalise les primitives de modification nécessaires et propose des opérations macroscopiques, exprimées en termes de primitives, qui confèrent plus de souplesse à l'élaboration du schéma car elles autorisent des états intermédiaires incohérents.

Enfin, la composante Dérivation montre qu'il est possible, à partir de spécification IFO_2 cohérentes, d'obtenir des schémas implantables, relationnel et objet.

Chapitre 2

Modélisation comportementale

Les objectifs généraux des modèles conceptuels, que nous avons décrit dans le chapitre précédent, sont partagés à la fois pour la représentation structurelle et la description dynamique des applications. La principale caractéristique du modèle IFO_2 est d'offrir un cadre de travail parfaitement uniforme pour les modélisations structurelles et comportementales des applications. En fait, IFO_2 harmonise descriptions statique et dynamique, en adoptant des concepts parfaitement symétriques dans les deux contextes et une même philosophie de représentation. Ainsi, la spécification du comportement est globale et orientée événement.

Dans ce chapitre, nous présentons l'aspect comportemental du modèle IFO_2 en mettant en avant l'approche événementielle retenue. Dans le paragraphe 2.1, nous exposons la problématique en détaillant les ambitions d'un modèle comportemental. Le paragraphe 2.2 présente les approches contribuant à la représentation comportementale. Nous nous attachons particulièrement à décrire les travaux orientés objet et orientés processus. La présentation de la partie comportementale du modèle IFO_2 est proposée au paragraphe 2.3. Enfin, pour conclure ce chapitre, le bilan de la contribution est donné sous la forme d'une discussion.

2.1 Exposé de la problématique

La Modélisation Comportementale

Comme les modèles structurels, les approches comportementales les plus récentes ont adopté le concept d'objet et perçoivent, dès lors, la dynamique de l'application comme l'ensemble des actions et réactions des différents objets. Les objets ont généralement un comportement autonome ou local qui les fait agir indépendamment des autres, i.e. de manière concurrente, en exécutant leurs propres opérations. Ils peuvent aussi interagir, il s'agit alors de comportement global, en sollicitant les services d'autres objets ou en répondant aux messages qui leur sont destinés.

Cette perception comportementale dirigée par les objets a un défaut : la perte de la vision globale de l'application. Les réactions en chaîne sont, par exemple, délicates à appréhender, car leur description est éparpillée entre différentes classes, et, de manière générale, les in-

teractions entre objets ne sont pas toujours facilement perceptibles. En fait, le concepteur se voit proposer une vision complète des objets, i.e. structurelle et comportementale, au détriment d'une vision globale des spécifications. Cette vision est tout aussi importante sur le plan dynamique que statique, car elle facilite l'appréhension du réel et la détection d'anomalies voire d'interblocages. Le besoin est véritablement crucial car les difficultés de conception ne proviennent pas des actions des objets par nature même concurrents mais de leurs interactions. Ainsi, l'adoption d'une vision comportementale par objet compromet certaines qualités conceptuelles classiques, comme la vision globale ou la fidélité des spécifications, sans réellement offrir une transposition des qualités objet au niveau conceptuel.

Les représentations comportementales s'appuyant sur les états des objets rencontrent d'autres problèmes. Tout d'abord, les états d'un objet sont, au niveau conceptuel, parfois très difficiles à appréhender. Ils peuvent ne pas seulement refléter la valeur d'un attribut, mais expliciter certaines combinaisons de valeurs pour l'objet ou même de liens avec d'autres objets. De plus, selon divers points d'observation, les états significatifs d'un objet peuvent varier. Enfin, même en considérant un même point de vue sur l'application, les concepteurs ne perçoivent pas forcément les mêmes états pour un objet. Ces différentes raisons contribuent à la difficulté d'identifier les états remarquables, mais probablement moins que le caractère artificiel que peuvent, dans certains cas, revêtir ces états. En effet, quand il est nécessaire de connaître le comportement passé du système pour savoir comment il va agir, de nouveaux états, correspondant à des attributs artificiels de classes existantes, voire à de nouvelles classes, doivent être introduits dans la représentation structurelle.

La richesse des capacités d'expression est également essentielle. Il s'agit plus précisément de pouvoir représenter des événements de diverses sémantiques, de pouvoir les combiner pour spécifier des conditions (arbitrairement complexes) de déclenchement d'opérations mais aussi de pouvoir décrire des cascades de déclenchements reflétant des réactions en chaîne. Pour satisfaire ce dernier point, il est impératif de permettre l'expression de conditions variées portant éventuellement sur l'histoire du système. Ces conditions peuvent être extrêmement fines et précises et leur spécification nécessite, dans ce cas, une connaissance très détaillée du réel.

En résumé, un modèle comportemental doit offrir au concepteur les mêmes avantages qu'un modèle structurel : tout d'abord une représentation naturelle et fidèle des spécifications, la possibilité d'en avoir une vue globale mettant en évidence plus particulièrement les interactions entre objets et les enchaînements d'opérations à réaliser (y compris par des objets différents) mais aussi la souplesse du processus de conception. Comme pour la description statique, les atouts d'un formalisme graphique sont également reconnus, pour une plus grande facilité d'appréhension et une exploitation automatique [63].

La Dérivation

Les exemples de dérivation dynamique sont fort rares. A l'exception de REMORA qui offre la spécification de triggers [80], de STATEMATE permettant la génération de code ADA et des approches proposées dans [52, 135] introduisant, pour des règles E-C-A, une spécification déclarative de plus haut niveau (sans être véritablement conceptuelle), les démarches dynamiques actuelles voient simplement les spécifications élaborées comme définissant le

cadre de travail des développeurs et ne proposent pas de possibilités d'implantation, automatique ou guidée, de la description comportementale. Ainsi, la continuité du processus de développement, préconisée sur le plan structurel [18], n'est-elle généralement pas préservée par les approches comportementales.
Pouvoir combler cette lacune est à notre avis, un objectif clef d'une approche dynamique. Par exemple, les modèles de bases de données actives, permettant, entre autres, de spécifier la dynamique des applications sous forme de règles E-C-A, sont des cibles naturelles d'implantation. De plus, ces règles sont spécifiées de manière déclarative et il est alors possible d'envisager l'automatisation de la production de code beaucoup plus facilement qu'avec des langages procéduraux.

L'Evolution

Pour les approches statiques, nous avons souligné, au chapitre précédent, le besoin d'inclure des capacités d'évolution. Elles sont tout aussi importantes sur le plan comportemental car elles y sont aussi la base d'une conception incrémentale et guidée, i.e. garantissant, pas à pas, la cohérence des spécifications et ont évidemment une fonction de maintenance. Comme pour les aspects structurels, il s'agit d'offrir un ensemble exhaustif d'opérations de modification, permettant la prise en compte de tout changement.

La Vérification Comportementale

La représentation dynamique d'une application peut comporter un certains nombres d'anomalies conduisant à un fonctionnement critique du système modélisé : interblocage, cycles infinis, etc. Proposer un mécanisme de vérification comportementale à un niveau conceptuel permet de limiter les anomalies avant toute génération de code. Ainsi, les concepteurs d'applications trouvent non seulement la garantie de certaines propriétés de fonctionnement du système modélisé mais aussi une aide à la correction des anomalies détectées.

Les approches comportementales ne proposant pas de capacités d'évolution et les dérivations en étant généralement absentes, c'est essentiellement sur les aspects de modélisation et de vérification que nous revenons, dans le paragraphe suivant, en examinant les contributions apportées par les travaux les plus récents.

2.2 Aperçu des travaux antérieurs

Pour présenter les approches de description du comportement, deux grandes tendances se dégagent. Elles sont qualifiées respectivement d'orientées objet ou processus. Dans le premier cas, les approches sous-jacentes s'attachent à représenter le comportement individuel des objets du système. Elles s'appuient généralement sur des diagrammes de transitions d'états et s'avèrent bien adaptées à la représentation de traitements orientés individu. La seconde tendance cherche à appréhender la dynamique du système d'information comme un ensemble de processus globaux et s'intéresse donc à l'organisation générale des réactions et/ou activités du système modélisé. Nous adoptons la division selon ces deux tendances

et montrons les atouts des démarches décrites mais également leurs points faibles. En, fait, les deux types de démarches sont complémentaires et doivent souvent cohabiter lors de l'élaboration d'applications pouvant partager des données communes. Ainsi, certains modèles comme UML ou IFO_2 (nous en décrivons les composantes dans le paragraphe 2.3) ont proposé un cadre de travail uniforme aux concepteurs. Enfin, nous présentons les approches de vérifications de comportements.

2.2.1 Comportement orienté individu

A de très rares exceptions près (comme OOM qui préserve la philosophie de Merise [19]), les modèles proposent au concepteur d'appréhender le comportement de l'application à travers celui de ses objets. Certains des modèles, évoqués dans ce paragraphe, sont décrits, de manière plus détaillée, dans [126]. Ils s'appuient très fréquemment sur des diagrammes de transition d'états, ou s'en inspirent, pour représenter la dynamique des objets.

Largement éprouvés car extrêmement utilisés pour la spécification de système temps réel [84, 85, 86, 94, 91], ces diagrammes ont des qualités notables. Ils offrent une représentation claire, facile à appréhender et naturelle pour le concepteur et permettent de décrire le comportement de systèmes complexes [37].

Dans OOA [121], OOA/OOD [43] ou OOD [17], ces diagrammes sont "plats" et consistent en un ensemble d'états, significatifs pour les objets de la classe étudiée, reliés entre eux par des transitions explicitant les changements d'états possibles. Les transitions sont déclenchées par des événements. Elles peuvent être contraintes par des conditions et permettent de réaliser des actions.

De tels diagrammes ont le défaut, largement évoqué dans la littérature [63, 84, 91, 94, 112], d'être rapidement illisible. Outre l'explosion exponentielle du nombre d'états et de transitions survenant pour de grandes applications, simplement parce que le nombre d'objets s'accroît, états et transitions peuvent, dans certaines situations, se trouver démultipliés inutilement. Par exemple, si un objet, considéré selon différents points de vue, a plusieurs cycles de vie logiquement distincts, il convient de créer autant d'états que de combinaisons possibles des divers points de vue. Les états modélisés résultent alors du produit Cartésien des états réels perceptibles par le concepteur. De même, si un événement déclenche, quel que soit l'état de l'objet, le passage à un état particulier donné, ce dernier est cible d'autant de transitions qu'il existe d'états dans le diagramme.
Pour éviter ces problèmes, Z. Manna et A. Pnueli [84] proposent, pour la spécification de systèmes concurrents, les $\forall$-automates permettant l'imbrication de diagrammes d'états. A un premier niveau, le comportement local des objets est encapsulé dans un diagramme, doté d'une interface (canaux de communication), permettant les interactions avec les autres objets. En utilisant le même mécanisme, différents diagrammes peuvent être réunis dans un diagramme de plus haut niveau, modélisant le comportement d'un sous-système de l'application et communiquant avec les autres. Ainsi, il est possible de manipuler comme un tout les comportements locaux d'objets sémantiquement différents ainsi que l'ensemble des interactions entre ces objets.
Cette approche est adaptée aux systèmes temps réel par J.S Ostroff, notamment en intégrant des bornes de temps pour contraindre l'exécution des transitions [92, 93, 95].

En proposant ses diagrammes d'états imbriqués [63, 64], utilisés plus tard dans OMT [112] et UML [89], D. Harel uniformise et étend ces possibilités. Tout état dans un diagramme peut être spécifié sous forme d'un diagramme plus détaillé, décrivant un micro-comportement, en termes de sous-états et de transitions entre eux. Les sous-états héritent des transitions d'entrée et de sortie de leur super-état mais il est possible de limiter cet héritage, en explicitant des sous-états comme points d'entrée ou de sortie du super-état. Dans ce modèle, un objet peut avoir plusieurs comportements en parallèle. Il a donc un état courant dans chacun de ses cycles de vie concurrents. La modélisation de telles situations est beaucoup plus naturelle pour le concepteur qu'avec des diagrammes "plats" et minimise notablement le nombre d'états (et donc de transitions) représentés. De plus, le concepteur dispose d'une échelle de perception variable des spécifications et d'une modularité flexible selon les différents niveaux de granularité de la description.

Pourtant ces techniques semblent plus probantes pour la qualité de la représentation obtenue, souvent plus naturelle pour le concepteur et graphiquement beaucoup plus concise, que pour permettre des spécifications véritablement modulables et réutilisables. En effet, lorsque des diagrammes sont imbriqués pour décrire le comportement complexe des objets d'une même classe, ils ne peuvent être perçus comme le support du partage des spécifications, mais plutôt comme celui d'un raffinement de plus en plus détaillé d'un comportement local. Enfin, malgré des mécanismes d'intégration très précis, la définition de diagrammes de plus haut niveau que ceux des classes a forcément lieu a posteriori puisqu'elle nécessite la connaissance des interactions entre objets.

Malgré ces défauts, les approches évoquées permettent de réutiliser des composants de l'application, à différents niveaux de détail.

Dans OMT, ces possibilités sont complétées par l'héritage du comportement spécifié pour une super-classe par ses sous-classes. Cependant, quand deux objets de classes distinctes ont les mêmes réactions, celles-ci doivent être spécifiées dans chacun des diagrammes. Elles ne peuvent être véritablement partagées qu'avec l'introduction, dans la description structurelle, d'une classe généralisant les précédentes et totalement artificielle, puisque ne répondant qu'à un souci d'optimisation de la représentation dynamique.

Dans [65], D. Harel étudie des possibilités de recouvrement de diagrammes permettant un véritable partage d'états et donc offrant la réutilisabilité à tous les niveaux de la description. Il n'en préconise pourtant pas l'utilisation en raison de l'extrême complexité de mise en œuvre.
Ainsi la représentation par imbrication de diagrammes n'est pas totalement satisfaisante sur le plan de la modularité et de la réutilisabilité. La vision offerte est certes globale mais elle privilégie certains aspects alors que d'autres sont beaucoup moins explicites dans la description, comme les interactions et les réactions synchronisées entre classes. Ce type de comportement est décrit en faisant appel à des mécanismes variés : mêmes événements déclenchant les transitions d'objets différents ; envoi de messages ou production par un objet d'événements déclencheurs de transitions pour un autre objet ; conditions de déclenchement de transitions pouvant être étroitement liées pour des objets différents ou pouvant dépendre de l'état actuel d'autres objets. Ainsi, pour pouvoir étudier une réaction

en chaîne, le concepteur doit-il examiner les diagrammes des différents objets concernés et, en utilisant les événements et conditions, retracer la séquence, à travers tous les diagrammes, des transitions devant être chronologiquement exécutées (une telle séquence est, dans OMT, un "chemin de contrôle"). La difficulté de reconstituer ce type d'interactions s'accroît bien sûr avec le nombre de classes participant à la réaction globale du système, avec le nombre d'états et de transitions pour chaque classe, mais aussi avec les niveaux de raffinement successifs de chaque état et la possibilité de comportements parallèles pour un même objet.
Dans ces approches, l'expression de conditions sur le comportement passé du système s'exprime en termes d'états : à chaque événement significatif pour un objet est associé un état (ou simplement, pour limiter le nombre d'états, la valeur particulière d'un attribut), leur enchaînement (ou leur séquence) est la "portion" d'histoire à mémoriser [112]. Le principal défaut de cette démarche est lié au caractère artificiel des attributs sous-jacents.

OMT complète les diagrammes d'états imbriqués en introduisant, outre les opérations exécutées lors des transitions, des activités réalisées tant que l'objet demeure dans un état, et en organisant les événements au sein d'une hiérarchie de spécialisation (avec héritage des attributs d'événements). Pour éviter de surcharger la représentation graphique, des opérations peuvent être associées aux états plutôt qu'aux transitions et alors déclenchées automatiquement à l'entrée ou la sortie de l'état ou encore par l'occurrence d'un événement donné.

En s'inspirant des diagrammes d'états, le modèle O* distingue les comportements locaux, dont la description est encapsulée dans les classes, et les comportements globaux [110]. Deux graphes, de cycle de vie et de dépendance, en permettent une représentation graphique naturelle. Comparé à OMT, O* permet de spécifier des événements dans l'expression des conditions de déclenchement d'opérations. Les contraintes sur l'historique de l'objet sont ainsi prises en compte de manière beaucoup plus naturelle. Les interactions entre objets, déclenchées par des événements partagés sont décrites à l'extérieur des classes et donc exclues des possibilités d'héritage.
Avec une philosophie assez proche de O* mais une technique de représentation différente, Oblog propose l'encapsulation dans les classes des comportements locaux et globaux, représentés sous forme d'équations d'interactions [58, 54, 74, 117]. En s'appuyant sur la logique temporelle, l'approche préconise une spécification multi-niveaux permettant au concepteur d'appréhender le comportement de l'application du global au particulier [24, 55, 57, 119].

Le modèle ORM [99] (et sa variante [8]) est assez proche des précédents mais son originalité est d'intégrer, dans le comportement des objets, leur évolution structurelle (modification de l'ensemble des attributs). Ainsi, ce comportement est-il appréhendé comme une succession de rôles joués, au cours du temps, par l'objet et dont l'enchaînement est contrôlé. Si le concept de rôle apporte une alternative originale à l'abstraction de spécialisation envisagée dans une perspective temporelle, il ne peut pourtant pas subvenir au besoin d'évolutions structurelles reflétant les modifications du réel, par essence même imprévisibles. Cette approche, enfin, se limite à la représentation des comportements locaux et ne permet pas de représenter les interactions entre objets.

Pour la représentation dynamique, UML [89] s'appuie largement sur OMT [112] (plus

riche sur ce plan que les autres méthodes à l'origine d'UML) et donc sur les diagrammes d'états de D. Harel. Les concepts fondamentaux sont similaires, cependant, UML introduit quelques variantes ou améliorations. Par exemple, des transitions simples (d'état à état) mais aussi complexes sont proposées. Ces dernières, utilisées dans le contexte de diagrammes concurrents, sont aussi présentes dans OMT (split et fork). La notion d'état de synchronisation est nouvelle et combinée aux transitions complexes, elle permet de mieux spécifier des comportements concurrents.

Pour compléter cette vision conceptuelle du comportement, il est intéressant d'évoquer celle proposée, à un autre niveau d'abstraction, par les modèles de bases de données actives [7, 15, 31, 33, 48, 50, 53, 70]. Dans cette vision logique, des combinaisons éventuellement complexes d'événements permettent le déclenchement de règles, i.e. l'exécution de séries de méthodes ou l'appel d'autres règles. Ces déclenchements peuvent bien sûr être contraints par certaines conditions sur les états de la base. La granularité de modularité de la description est la règle. Mais sa définition n'impose aucune contrainte particulière sur les objets concernés, elle en invoque seulement les opérations ou les soumet à des conditions. Il n'y a donc plus de distinction entre comportements locaux et globaux mais une description véritablement uniforme de toute "micro" ou "macro-réaction".

2.2.2 Comportement orienté processus

La vision classique du comportement d'un système d'information, i.e. une vision globale, a récemment été adoptée par les modèles et systèmes de workflows. Initialement nés de la notion de processus dans l'industrie et le travail de bureau, les workflows font aujourd'hui l'objet de différents travaux et de nombreux systèmes sont proposés [61, 25, 10, 108, 123]. Depuis 1993, le groupe de travail Workflow Management Coalition (WfMC) s'attache en particulier aux problèmes de normalisation [68].

Les travaux, s'inscrivant dans un contexte de gestion de workflows, abordent différents aspects [11, 61, 68], dont particulièrement : la modélisation de processus et la spécification de workflows. Il s'agit d'offrir des modèles de représentation et des méthodologies d'analyse et de conception pour représenter les processus. Les modèles de workflows offrent les concepts de base permettant la description des processus. Ces concepts sont ceux de tâches, dépendances entre tâches et rôle (i.e. compétence d'un agent pour réaliser une tâche). Des langages de spécification graphiques ou textuels, basés sur ces modèles, sont définis.

Concernant la spécification de processus, les approches existantes poursuivent les mêmes objectifs que les approches de conception de systèmes d'information ou les méthodes de génie logiciel [21, 22, 80, 43, 125, 17, 112, 121, 89]. Il s'agit, en effet, d'offrir des abstractions de haut niveau permettant une représentation fidèle et sémantiquement riche des processus. L'indépendance par rapport aux outils d'implémentation est recherchée et les travaux de standardisation, menés par la WfMC, visent en particulier à l'adoption d'une norme concernant la définition de processus. Les points communs ne s'arrêtent pas là puisque, tout comme dans les approches orientées objet (pour leurs aspects dynamiques), les modèles de workflows privilégient des spécifications modulaires et réutilisables [68, 137, 108] et certains proposent des approches formelles permettant une description sans ambiguïté

dont certaines propriétés peuvent être prouvées [10, 123, 124].

Les approches de modélisation de processus doivent permettre la spécification des tâches, des flux de contrôles et de l'affectation des tâches aux exécutants (éventuellement en fonction de leur compétence ou rôle). Les langages associés offrent des capacités de structuration des tâches (control flow) et d'échanges de données entre tâches (data flow). En d'autres termes, ils permettent de préciser quelles tâches peuvent être exécutées en parallèle ou encore quelles tâches doivent attendre les résultats de telle(s) autre(s). Ces langages offrent aussi des possibilités de spécification des exceptions (que faire en cas d'échec d'une tâche ou de non achèvement d'un workflow), de gestion des durées d'exécution des tâches ainsi qu'un système de priorité entre tâches. La structuration des tâches s'appuie sur la définition de routines qui peuvent être classées en conditional, rule-based et parallel [61]. Les premières introduisent des gardes pour autoriser le déclenchement de tâches et les dernières permettent une exécution parallèle de différentes tâches.

Une différence fondamentale entre workflows et approches conceptuelles concerne la vision des données qui jouent un rôle central, à travers le schéma structurel, dans les approches de conception alors qu'elles ne constituent, dans un workflow, que le flux échangé entre tâches et agents. Les modèles de workflows proposent une spécification plus fine de certains aspects organisationnels que ne le font les méthodes objets comme OMT, OOA-OOD ou encore UML.

Une autre différence concerne les cibles d'implémentation. Dans les approches de conception, les environnements de développement visés sont soit des langages de programmation (classiques ou objets) soit des modèles et systèmes de bases de données et leurs langages hôtes ou de développement. Enfin, les approches de workflows visent l'intégration des processus modélisés dans un environnement technique existant [61, 108], ce qui permet de limiter les réalisations à mener.

Dans les approches comportementales évoquées, certains points de modélisation restent flous. Comment peut-on raffiner un comportement hérité ? Quid des mécanismes de surcharge ? Comment peut-on représenter des interactions entre objets de la même classe ?

Il n'en demeure pas moins qu'en se basant sur des techniques de représentation puissantes, ces approches offrent toutes les capacités d'expression requises et peuvent tirer profit des nombreux travaux menés sur les systèmes concurrents, en particulier les possibilités de vérification proposées [33].

L'opposition entre les deux visions précédentes de la dynamique d'un système s'explique par le souci de répondre à deux types de besoins différents. Néanmoins, nous avons vu que certains aspects de la problématique abordée sont communs et les démarches proposées étant complémentaires, leur dualité est souvent indispensable. Ainsi, en introduisant des diagrammes d'activité dédiés à la modélisation de processus qui ne sont que des variantes de diagrammes dynamiques, UML réconcilie les deux grandes tendances de modélisation du comportement et met en évidence leurs points communs et leurs spécificités. Les transitions entre activités sont dépourvues d'événement déclencheur puisqu'elles n'ont d'autre objectif que de mettre en évidence l'organisation et le déroulement des activités. Les points d'entrée

ou de sortie sont uniques puisqu'ils marquent le début et la fin d'un processus. Par contre, les mécanismes de synchronisation sont similaires et des conventions graphiques uniformes sont offertes.

2.2.3 Vérification du comportement

Les problèmes de vérification comportementale ont été abordés dans différents domaines, incluant les systèmes experts, les bases de données déductives et actives, les spécifications de systèmes temps réel. Les approches proposées dans ces différents domaines partagent certaines préoccupations, en particulier, le problème de la terminaison lors du traitement de règles ou de l'exécution de programmes.

Z. Manna et A. Pnueli proposent une approche de spécification et de vérification pour des programmes concurrents réactifs, basée sur les $\forall$-automates [84, 85, 86]. Leur objectif est de contrôler les propriétés fondamentales d'initialisation, consécutivité, terminaison, justice et équité. Ces deux derniers besoins garantissent à divers processus de pouvoir progresser de manière concurrente, tout au long de l'exécution de programmes. Cette approche est adaptée aux systèmes temps réel par J.S. Ostroff [92, 93, 91].

Plus proche de notre contexte de recherche, des techniques de vérification sont proposées pour analyser le comportement de systèmes distribués [35, 36, 33]. Elles sont inspirées de l'analyse de flux de données, initialement proposée comme une technique permettant de déterminer les propriétés de programme [109]. S'appuyant sur des diagrammes d'états, ces techniques offrent la détection automatique de l'inatteignabilité des états et de certains interblocages entre processus concurrents dus à des anomalies de communication. La méthode proposée ne révèle pas toutes les erreurs de synchronisation mais elle est réellement utile pour une meilleure compréhension préliminaire du comportement du processus et une détection, à moindre coût, d'anomalies loin d'être triviales.

Dans [75], une approche est développée pour diagnostiquer des comportements non voulus de règles de production dans des bases de données déductives et orientées objet. En suivant les idées introduites dans [30], un graphe de déclenchement est construit. Dans ce cadre, les conflits de mise à jour peuvent être mis en évidence ainsi que les cycles. Une analyse plus poussée des cycles est proposée en considérant des "cycles essentiels", correspondant à des situations critiques, et "non essentiels" (dont l'exécution est finie). [76] propose une approche, basée sur la réécriture de termes, pour contrôler les propriétés de terminaison et de confluence dans un contexte de bases de données actives.
De telles propriétés ainsi que le déterminisme observable sont étudiés dans [52, 6] pour des bases de données actives. Dans cette approche, des comportements éventuellement infinis sont détectés et signalés à l'utilisateur. L'analyse de la confluence est basée sur la commutativité des règles. Le déterminisme observable fait référence à la perception qu'a l'utilisateur du traitement des règles. Dans [6], cette approche est brièvement comparée avec des techniques équivalentes pour les systèmes experts et ses avantages sont mis en évidence. Elle est mise en œuvre dans le système Starburst [138]. Notons que les contrôles de terminaison sont également implantés dans Chimera [14, 29] ainsi que dans d'autres approches comme le système VITAL [16].

2.3 Synthèse des contributions

En définissant la partie comportementale du projet IFO_2, nous montrons qu'il est possible de s'inspirer des concepts définis dans la composante de modélisation statique et de les adapter au contexte dynamique pour définir des mécanismes adaptés de représentation comportementale.

La vision orientée événement confère à IFO_2 certaines qualités conceptuelles absentes des autres modèles, mais il est essentiel de montrer qu'elle permet d'offrir des capacités d'expression au moins équivalentes. Cet aspect est plus particulièrement étudié à travers une présentation descriptive de la composante de modélisation (paragraphe 2.3.1) et comparative dans la discussion du paragraphe 2.4.

Pour compléter l'approche proposée, nous adjoignons aux possibilités de modélisation, des composantes Evolution et Dérivation dont les objectifs généraux sont tout à fait symétriques de ceux adoptés pour les composantes structurelles équivalentes. Ainsi, les possibilités d'évolution proposées doivent permettre d'opérer toute réorganisation de la description et garantir la cohérence du résultat obtenu ; les dérivations doivent contribuer à la réalisation de l'application. Pour ces dernières, nous nous intéressons tout particulièrement aux SGBD actifs car ils sont des systèmes naturels d'implantation et permettent d'envisager une production automatique de code par génération de règles E-C-A. Nous précisons également les optimisations à adopter pour une génération vers le système actif NAOS[46].

Enfin, pour garantir la validité des spécifications comportementales, nous proposons des mécanismes de vérifications complémentaires à ceux couramment utilisés dans les approches conceptuelles.

L'approche de conception comportementale que nous proposons à travers le projet IFO_2 s'articule selon les composantes dont nous venons de tracer les objectifs et dont un résumé est proposé à travers les paragraphes suivants.

2.3.1 Composante Modélisation

IFO_2 ne se préoccupant que d'événements, ils y sont perçus, d'une manière très générale comme la représentation de faits participant aux réactions du système modélisé. Les concepts de la partie comportementale d'IFO_2 sont formellement définis dans [130, 126]. Les événements interviennent de façon spontanée et aléatoire ou non (événements externes ou temporels) ou sont engendrés par l'application elle-même (événements internes). Dans les deux cas, leur occurrence est instantanée, i.e. ils n'ont pas de durée. Comme dans [34, 49], nous adoptons l'hypothèse simplificatrice suivante : il ne peut survenir à un instant donné qu'au plus un événement, contrainte atténuée en jouant sur la "granularité" du temps (dont l'échelle est perçue comme infiniment dense). De plus, cette occurrence concerne certains objets représentés dans le système et va donc influer sur leur vie. Ces objets sont considérés dans IFO_2 comme des paramètres pour les événements.

Comme dans les autres modèles, un *événement* en IFO_2 est caractérisé par son instant d'occurrence et ses paramètres représentant les objets qui réagissent à l'événement considéré. Mais de plus, un événement est identifié (exactement comme un objet dans la partie structurelle du modèle) et il a une valeur déterminée par son type. En fait, différentes catégories d'événements sont fréquemment distinguées dans bien des modèles. Cependant, une telle classification ne relève pas d'une problématique de représentation. A l'opposé, IFO_2 fait de la définition des types d'événements un point clef dans la modélisation dynamique. Les types de base proposés, symétriques des types structurels (leur formalisme graphique est illustré par la figure 2.1), sont les suivants :

- le *Type d'Evénements Simple* (TES) représente les événements déclencheurs d'une méthode, i.e. directement matérialisables à travers leurs effets sur les objets. La méthode concernée est considérée comme la valeur de l'événement et sa spécification (en fait sa signature) est incluse dans la représentation structurelle de l'application [1]. En distinguant les méthodes de leur appel, IFO_2 offre l'uniformité de la représentation dynamique ("tout fait est un événement") et le respect du principe d'encapsulation de ses types d'objets structurels ;
- le *Type d'Evénements Abstrait* (TEA) modélise des événements externes ou temporels, émis par l'environnement de l'application. Il est également utilisé pour représenter des événements internes qui ne sont intéressants qu'à travers leurs conséquences, i.e. les autres événements qu'ils déclenchent. Leur valeur est nulle ;
- le *Type d'Evénements Représenté* (TER) symbolise un autre type, décrit ailleurs dans les spécifications, qui peut ainsi être utilisé sans en connaître précisément la description. Sa valeur dépend bien sûr du type d'événements symbolisé.
  Les types représentés sont essentiels car ils permettent de différer la description d'un type ou de la confier à un autre concepteur tout en poursuivant la partie des spécifications à élaborer.

Exemple 1 Nous continuons, pour illustrer la partie comportementale d'IFO_2, à utiliser l'application exemple "Ascenseurs", introduite au chapitre précédent. Dans ce contexte, "Montée" est un type d'événements simple, représentant les ordres de mise en mouvement de la cabine d'un ascenseur. La méthode associée provoque le mouvement ascendant de la cabine d'un seul étage. Le type d'événements abstrait "Demande-Etage" décrit les événements externes survenant lorsque les usagers appellent l'ascenseur d'un palier ou demandent un étage de l'intérieur de la cabine. "Satis-Demande" est également un type abstrait, mais il modélise les événements internes émis lorsque les usagers atteignent l'étage désiré. Enfin, le type représenté "Aller-Etage" est introduit dans la description comportementale afin de réutiliser le TEA "Demande-Etage". □

Pour modéliser le comportement d'un système, il est nécessaire d'exprimer des conditions de synchronisation des événements, i.e. différentes variantes de conjonction et disjonction d'événements. Pour répondre à ce besoin, nous avons choisi de représenter les événements complexes en utilisant des constructeurs. Avec cette approche, IFO_2 offre non seulement le pouvoir d'expression requis mais aussi l'uniformité par rapport à la composante struc-

1. En fait, dans la description du fragment structurel réalisant l'opération.

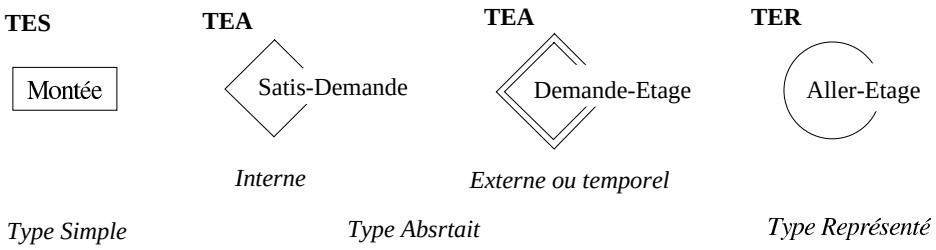

FIG. 2.1 – *Exemples de Types d'Evénements de base*

turelle du modèle. Ainsi, IFO_2 propose les constructeurs événementiels suivants, dont le formalisme graphique est indiqué figure 2.2.

- La *composition* reflète la conjonction d'événements de différents types.
- La *séquence* est définie comme le constructeur précédent mais inclut une contrainte chronologique entre les occurrences de ses événements composants.
- Le *groupement* exprime la conjonction d'événements de même type. Il est comparable au constructeur "closure" du modèle HiPAC [32].
- L'*union* représente une disjonction d'événements.

La valeur des éléments de ces types composites est définie en utilisant la valeur de leurs composants et la structure de tuple (pour les deux premiers) ou d'ensemble (pour le groupement). La valeur d'un événement de type union est la valeur de l'un de ses événements composants.

Exemple 2 Imaginons que le concepteur veuille spécifier les événements déclencheurs d'un mouvement de la cabine d'un ascenseur. Il peut utiliser le type union illustré par la figure 2.2, décrivant une alternative entre "Montée" et "Descente", i.e. entre deux appels de méthodes, chacune provoquant le mouvement de la cabine d'un seul étage. □

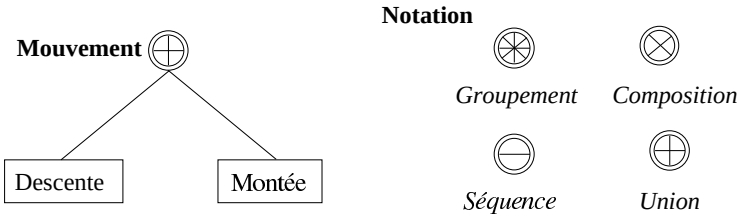

FIG. 2.2 – *Exemple de Type d'événements complexe*

Pour pouvoir exprimer des conditions de synchronisation arbitrairement complexes, les constructeurs définis peuvent être appliqués récursivement sur les types de bases. De façon générale, un type est un arbre identifié par sa racine, dont les nœuds sont des constructeurs et les feuilles des types de base IFO_2.
Comme pour les types structurels, la vision extensive des types d'événements est proposée

à travers les notions d'instance de types et d'événements attachés.

Exemple 3 Dans notre application exemple, le type complexe "Stop", illustré par la figure 2.3, permet de spécifier les conditions de déclenchement d'un processus d'arrêt de la cabine d'un ascenseur. En fait, un ordre d'arrêt est donné si la cabine vient de s'immobiliser à un étage (fait symbolisé par le TER "Arrivée") et qu'il existe une ou plusieurs demandes d'usagers pour cet étage (représentées par le TER "Aller-Etage"). Ainsi le type "Stop" est-il conçu comme la composition du TER "Arrivée" et d'un groupement de "Aller-Etage". Des exemples d'instances sont indiqués sur le schéma. □

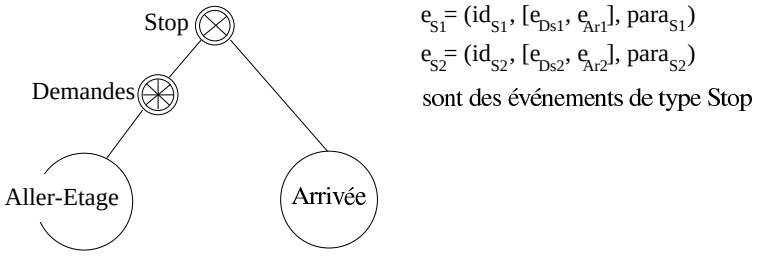

FIG. 2.3 – *Le type complexe "Stop"*

Hormis les conditions de synchronisation, d'autres relations entre événements doivent être modélisées : les enchaînements de déclenchements. Ces liens de causalité sont exprimés en reliant les types d'événements entre eux par des fonctions. En fait, ces relations de cause à effet sont décrites à travers le concept de *fragment événementiel*, dont la construction obéit à des règles précises. Ce concept, directement hérité du modèle IFO [4], est essentiel car il permet la modularité des spécifications. Son objectif est de décrire une portion du réel modélisé, i.e. les réactions du système dans une certaine situation, cette description se focalisant sur un type principal appelé le *cœur* du fragment.
Les fonctions de fragment, reliant les types, expriment des conditions générales sur l'enchaînement des événements car elles peuvent combiner les caractéristiques suivantes :

- *simple* ou *complexe*, i.e. un événement de leur type origine déclenche un ou plusieurs événements de leur type cible ;
- *partielle* ou *totale*, i.e. un événement de leur type origine peut ou doit déclencher un événement de leur type cible ;
- *différée* ou *immédiate*, s'il existe ou non un délai entre les occurrences des événements de l'origine et de la cible.

De plus, nous introduisons une distinction entre les fonctions de *déclenchement* et de *précédence*, qui expriment le fait qu'un événement du cœur du fragment déclenche d'autres événements ou qu'il est précédé d'autres événements. Pour souligner l'intérêt de cette différence, il suffit de considérer un événement externe ou temporel. Par nature même, il ne peut être déclenché par aucun autre événement, pourtant il est parfois nécessaire d'exprimer le fait qu'il doit être précédé par l'occurrence d'autres événements. Contrairement aux fonctions de déclenchement, une fonction de précédence, si elle existe, est unique dans un fragment car si les événements déclenchés peuvent être contraints par des conditions

spécifiques différentes, les événements déclencheurs doivent, eux, être synchronisés par les constructeurs décrits.

Il est ainsi possible, en adoptant un point spécifique d'observation (le cœur du fragment), d'avoir une vision complète du "sous-comportement" considéré, i.e. incluant non seulement les événements générés (déclenchés) mais aussi les événements générateurs (précédents). Le fragment peut ainsi être réellement considéré comme une unité de description macroscopique du comportement du système.

Exemple 4 La figure 2.4 illustre un fragment, dépourvu de fonction de précédence, dont le cœur est un type d'événements externe "Demande-Etage". Ce fragment décrit le comportement du système lorsqu'un usager désire se rendre à un étage. Le cœur du fragment est relié par une fonction partielle et différée au type simple "Fermeture" déclenchant la fermeture des portes. La fonction est partielle car cette fermeture n'est pas systématique, elle peut être provoquée non pas par la demande d'étage considérée mais par une demande antérieure. Le caractère différé de la fonction tient compte du cas où l'usager appelle l'ascenseur alors qu'il est occupé.
Le TEA est également lié au type composite "Mouvement" par une fonction partielle, différée et complexe. La fonction est partielle car toute demande d'un usager ne déclenche pas forcément un mouvement de la cabine : l'étage demandé peut être desservi en satisfaisant une demande antérieure. Pour traduire l'éventuel délai entre la demande d'un usager et le mouvement résultant de la cabine, la fonction est différée. Enfin, les méthodes correspondant aux TES "Montée" et "Descente" provoquent le mouvement de la cabine d'un seul étage. C'est pourquoi la fonction de déclenchement est complexe.
Le type union "Mouvement" est cœur d'un sous-fragment. La fonction qui le lie au type représenté "Arrivée-Etage" (dont les conséquences sont décrites dans un autre fragment) est totale et immédiate, ce qui signifie que tout mouvement de la cabine se conclut par l'arrivée à un étage. □

Les vues partielles que fournissent les fragments, sont regroupées au sein de *schémas événementiels* qui offrent une vision globale du comportement spécifié. Plus précisement, les types représentés sont reliés aux cœurs des fragments au moyen de liens IS_A événementiels. Comme sur le plan structurel, ces liens IS_A expriment la spécialisation avec les possibilités afférentes de surcharge et affinement, ou l'utilisation. Un trait comportemental hérité peut être enrichi aussi bien au niveau des ses conséquences que de ses précédents : il suffit de faire du TER cible le cœur d'un nouveau fragment pour pouvoir lui spécifier de nouveaux événements générateurs ou générés. Un TER pouvant avoir plusieurs sources, IFO_2 prend en compte l'héritage multiple, ce qui permet de combiner différents traits comportementaux en manipulant un unique type.

L'instance d'un schéma est alors définie comme l'union des instances des fragments associés, ces instances devant respecter la propagation des propriétés (les événements attachés du cœur) via les liens IS_A (le type représenté hérite ainsi de la description du type source).

Exemple 5 La figure 2.5 présente le schéma événementiel IFO_2 "Ascenseurs",

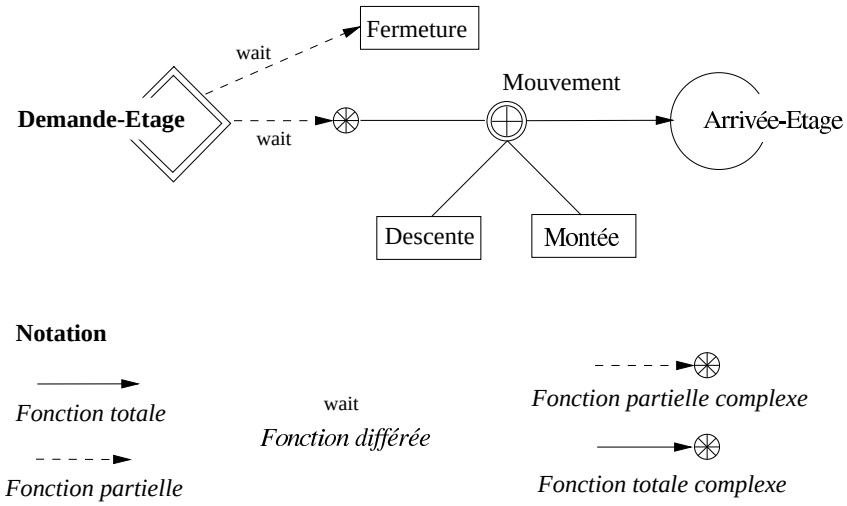

FIG. 2.4 – *Le fragment événementiel "Demande-Etage"*

incluant trois fragments, chacun dédié à un aspect précis des réactions du système. Outre le fragment "Demande-Etage", le schéma inclut le fragment "Satis-Demande", consacré aux réactions de l'ascenseur quand la cabine arrive à l'étage demandé. Ce fragment comprend le type représenté "Aller-Etage" dont tout événement "est un" événement de "Demande-Etage", selon la sémantique du lien IS_A reliant ces deux types. De manière symétrique, tout événement de "Arrivée" est un événement du cœur relié à ce type représenté, i.e. un événement de "Arrivée-Cabine". Les types représentés de ce fragment participent à la construction du type complexe "Stop", source d'une fonction de précédence. Cela signifie que l'occurrence d'un événement du cœur "Satis-Demande" ne peut se produire qu'après celle d'un événement de "Stop". Ce dernier est en fait une combinaison d'une ou plusieurs demandes d'usagers (groupement de "Aller-Etage") et (composition) de l'arrivée de la cabine à un étage. Ainsi, un ordre d'arrêt n'est donné à la cabine que si celle-ci est effectivement localisée à un étage précis et qu'il existe des usagers à satisfaire. Il y a alors émission d'un événement de "Satis-Demande", lequel déclenche l'immobilisation de la cabine puis l'ouverture des portes.
Enfin, "Arrivée-Cabine" est un fragment particulier car il est réduit à son cœur, un type simple déclencheur d'une méthode retournant le numéro de l'étage atteint par la cabine. La spécification de ce dernier fragment permet de réutiliser l'appel de méthode dans les deux autres fragments via des liens IS_A et les types représentés "Arrivée-Etage" et "Arrivée". □

Les concepts du modèle ayant été décrits, il convient maintenant d'expliciter la sémantique des schémas IFO_2.

La simulation du comportement d'une application spécifié en IFO_2 consiste en une propagation de déclenchements d'événements. Nous l'appelons l'*activité* du schéma IFO_2. Traduisant les réactions du système face à son environnement, cette propagation est initiée

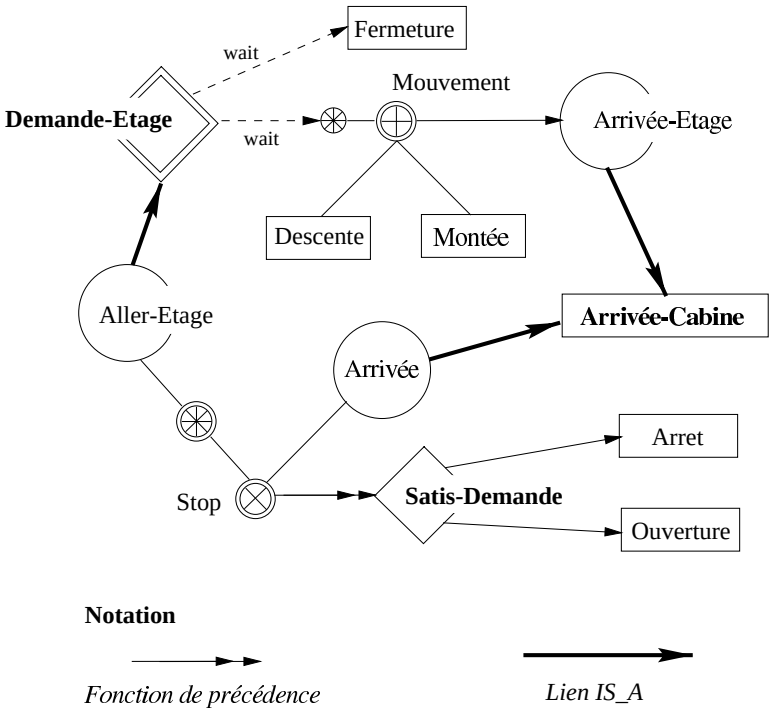

FIG. 2.5 – *Schéma événementiel "Ascenseurs"*

par l'occurrence d'un événement externe ou temporel ou d'une combinaison de tels événements. Dans ce cas, le TEA sous-jacent (ou le type construit à partir de ces TEA) doit être déclenchable spontanément (i.e. ne pas avoir de précédent). Il est alors considéré comme une *entrée* du graphe événementiel et est le point de départ de l'activité. La cascade de déclenchements, représentée en IFO_2 par un *chemin d'activité*, s'arrête quand toutes les actions, reflétant le but poursuivi par le système, sont réalisées. Ces actions sont décrites, dans le schéma, au sein d'un ou de plusieurs fragments appelés *fragments de satisfaction.*

Exemple 6 Dans notre schéma exemple, les seuls événements spontanément émis sont ceux de "Demande-Etage", qui est donc l'unique entrée du graphe. Effectivement, c'est la demande d'un usager qui fait réagir le système et initie la cascade de déclenchements. Son point d'arrêt (et donc le but du système) est que chaque usager ayant demandé un étage doit, à terme, y arriver. Ce but est représenté à travers le seul fragment de satisfaction du schéma, "Satis-Demande". □

Entre ces deux points, de démarrage et de fin de l'activité, la propagation des déclenchements s'effectue :

- au sein de chaque fragment, en évaluant les fonctions de précédence et de déclenchement ;
- entre les fragments, en naviguant le long des liens IS_A.

Contrairement à la propagation au sein d'un fragment, qui est dirigée par les fonctions, la cascade de déclenchements entre les fragments n'est pas orientée par le sens des liens IS_A,

elle peut se faire dans les deux sens, i.e. du type représenté vers le cœur ou inversement. En fait, chaque lien IS_A a un sens de parcours déterminé par le style de modélisation adoptée. Schématiquement, si, au cours de l'activité du schéma, les événements d'un cœur sont les premiers déclenchés, ils provoquent l'émission d'événements du type représenté. Au contraire, si le type représenté est le premier atteint par la propagation, l'événement émis engendre un événement du cœur.

Exemple 7 Considérons le schéma de la figure 2.5 et supposons qu'un usager appelle l'ascenseur, alors inoccupé, d'un palier pour se rendre à un étage supérieur. Selon cette hypothèse, l'événement externe déclenche la fermeture des portes puis une occurrence d'un événement du type composite "Mouvement" et de "Montée". La propagation se poursuit au niveau du sous-fragment par le déclenchement d'un événement de "Arrivée-Etage". Ce dernier provoque alors l'émission d'un événement du type "Arrivée-Cabine" en naviguant le long du lien IS_A du type représenté vers le cœur. Examinons maintenant les deux derniers types représentés de notre schéma : "Arrivée" et "Aller-Etage". Participant à la construction d'un type précédent dans leur fragment, ils ne peuvent être déclenchés que par le déclenchement de leur cœur respectif. Dans ces deux cas, la navigation le long des liens IS_A se fait du cœur vers le type représenté. Ainsi, l'émission d'un événement de "Arrivée-Cabine" engendre celle d'un événement de "Arrivée" et de manière symétrique, la demande de l'usager provoque le déclenchement d'un événement du type représenté "Aller-Etage". Il existe donc, à cet instant, des événements permettant de construire un événement complexe du type "Stop". Supposons que la demande ne puisse être satisfaite car la cabine n'a pas encore atteint l'étage demandé par l'usager. Cette condition est exprimée dans la spécification de la fonction de précédence reliant "Stop" au cœur de fragment, en utilisant un langage algébrique manipulant des séquences d'événements. □

Le principe général de propagation de l'activité, décrit jusqu'à présent, doit être complété en précisant le rôle essentiel des fragments de satisfaction. Ils représentent non seulement le but du système mais ils relancent également l'activité en déclenchant des itérations dans le parcours du schéma, jusqu'à ce que le but du système soit atteint. Ces itérations sont réalisées en réexaminant les fonctions complexes et différées, évaluées lors du parcours de l'activité.

Exemple 8 Reprenons l'activité de notre schéma exemple là où nous l'avons laissée, i.e. avec l'impossibilité de satisfaire l'usager. Une première itération est alors réalisée pour déclencher à nouveau la fonction complexe entre "Demande-Etage" et "Mouvement". Ce processus est renouvelé autant de fois que nécessaire pour atteindre l'étage demandé. Quand ceci se produit, la condition de la fonction de précédence dans le fragment "Satis-Demande" est vraie et la propagation de déclenchements se poursuit en engendrant un événement du cœur de fragment qui provoque immédiatement un événement de "Arrêt" puis déclenche l'ouverture des portes de l'ascenseur. □

Pour permettre de spécifier des contraintes fines, un langage algébrique, basé sur le concept de *trace* introduit dans [67], a été défini [126, 127]. La combinaison des opérateurs permet d'isoler ou de localiser un événement ou trait comportemental selon différents critères, temporels, sémantiques ou encore selon sa portée structurelle. Pour offrir une spécification plus complète, des opérateurs liés particulièrement à l'activité du schéma ont également été définis (restitution d'un événement d'origine, vérification de la satisfaction d'un événement, etc.).

2.3.2 Composante Evolution

De par la symétrie des concepts structurels et comportementaux, la composante Evolution[2], permettant d'élaborer pas à pas et de rectifier les schémas événementiels IFO_2, tire profit du travail réalisé sur le plan statique. Elle adopte une démarche similaire, i.e. propose un ensemble minimal et complet de primitives de modification [128], qui transposent, dans un contexte dynamique, les propositions structurelles équivalentes. L'application de ces primitives est soumise au respect de règles de cohérence.
Cependant, dans notre contexte comportemental, les contrôles doivent être étendus au-delà de la vérification de la construction correcte du schéma, car un schéma bien construit peut tout de même représenter un comportement absurde.

Outre les règles de cohérence régissant la construction des types, des fragments et des schémas événementiels, que nous appelons invariants syntaxiques, des vérifications de l'activité du schéma sont définies sous forme d'invariants sémantiques. Ils assurent en particulier que l'activité peut démarrer et s'achever dans un schéma, en effectuant certains contrôles sur la spécification des entrées et des fragments de satisfaction, et permettent de détecter certains blocages simples de cette activité. Contrairement aux contrôles syntaxiques, les invariants sémantiques ne peuvent être vérifiés systématiquement au cours d'une élaboration pas à pas du schéma, car leur application nécessite une spécification complète de l'application. Ils sont donc perçus comme un mécanisme de vérification a posteriori.
[128] propose deux algorithmes pour cette composante Evolution. Le premier, dit de vérification syntaxique, détermine si le résultat d'une modification, opérée sur un schéma bien construit, est lui-même bien construit. Le deuxième algorithme réalise effectivement l'opération, si elle est cohérente. Enfin, dans [104], le troisième niveau de contrôle met en œuvre les vérifications sémantiques évoquées dont nous proposons une illustration à travers l'exemple suivant.

Exemple 9 Imaginons que le concepteur veuille dissocier les demandes d'étages, de l'intérieur de la cabine, des appels de l'ascenseur à partir d'un palier. Il propose la représentation de la figure 2.6. Le schéma élaboré est bien construit, il satisfait toutes les règles de cohérence du modèle, pourtant il est en conflit avec certains invariants sémantiques.
Dans la représentation proposée, le type représenté "Demande" est déclenché par le type d'événements externe "Demande-Etage". Par ailleurs, il symbolise le type externe "Appel", selon la sémantique du lien IS_A les reliant. Cependant,

2. Ce paragraphe a pour but d'assurer l'homogénéité des présentations du modèle IFO_2. Les travaux réalisés dans ce paragraphe ne rentre pas dans ma contribution mais dans celle des autres membres du projet.

par nature même, les événements de "Appel" surviennent spontanément et ne peuvent jamais être déclenchés par un autre événement. Ainsi, en examinant le comportement sous-jacent, son absurdité apparaît : l'usager qui a demandé un étage reste bloqué dans la cabine jusqu'à ce qu'une autre personne appelle l'ascenseur et, si cela arrive, il n'atteint jamais l'étage demandé (mais simplement l'étage d'appel de l'autre usager). L'erreur de conception décrite est filtrée par l'application des invariants sémantiques. □

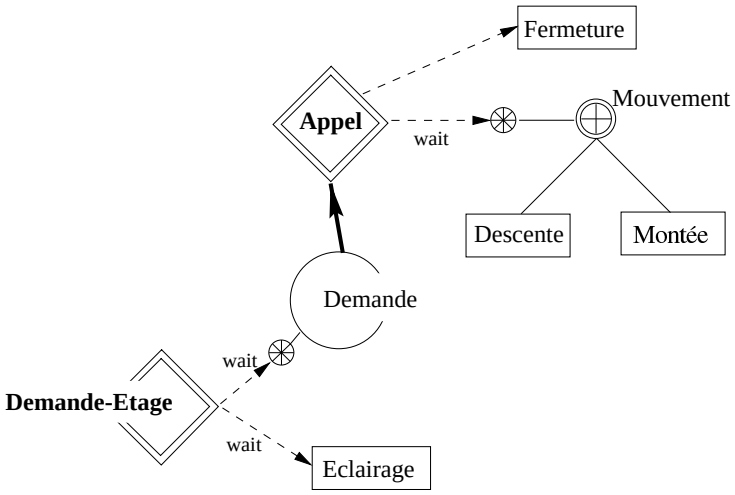

FIG. 2.6 – *Exemple de représentation incorrecte*

Si la composante Evolution répond à nos objectifs de richesse fonctionnelle, elle n'offre pas un contrôle exhaustif des spécifications. Néanmoins, elle a permis d'aborder la problématique de vérification du comportement, sur laquelle nous revenons dans le paragraphe 2.3.4.

2.3.3 Composante Dérivation

Nous nous sommes attachés à compléter la partie de modélisation comportementale d'IFO$_2$ en définissant une composante de dérivation vers des modèles cibles permettant l'implantation des spécifications [131]. Voulant avant tout mettre en évidence la faisabilité de la dérivation (et ne visant pas l'exhaustivité des traductions produites), nous avons opté, comme base théorique de dérivation d'IFO$_2$, pour un modèle de règles E-C-A de type HiPAC [49]. Il peut, en effet, être perçu comme un "dénominateur commun" dans le sens où les autres travaux en adoptent les concepts fondamentaux et étendent ses capacités d'expression.
Nous avons défini un algorithme de transformation d'un schéma IFO$_2$ événementiel sous forme de règles E-C-A, dont nous ne présentons ici que les principes généraux. Cet algorithme est présenté dans [131] et largement détaillé et illustré dans [126]. Il convient de noter que si notre préoccupation initiale était avant tout conceptuelle, le modèle IFO$_2$ comportemental, à travers la dérivation qui en est proposée, est également très intéressant

pour les développeurs d'applications actives. La difficulté de définir des règles E-C-A a été à plusieurs reprises évoquée dans la littérature et le besoin de spécification de plus haut niveau reconnu [6, 23, 30, 52, 135, 46].

Au niveau logique, l'activité d'une BD s'exprime en termes d'actions dont la réalisation est contrainte par à la fois l'occurrence d'un ou plusieurs événements et des conditions portant sur les objets. Une première recherche intuitive d'équivalence entre les concepts IFO_2 conduit à rapprocher les types d'événements simples (déclencheurs de méthodes) et les actions, et à localiser les conditions de déclenchement des règles actives dans la spécification des fonctions. Ces conditions posent un problème particulier. Exprimées en termes d'événements dans IFO_2, elles doivent être traduites en termes d'états. Sans être formalisée, cette réécriture est intuitivement présentée dans [126]. Nous supposons donc, dans la suite, que les conditions sur les états remarquables ont été exhibées, dans une phase préliminaire, à partir de la spécification des fonctions de fragment.

De manière schématique, le processus de génération de règles E-C-A débute par les entrées du graphe IFO_2 en examinant les fragments correspondants. Il se poursuit par la transformation des fragments liés aux fragments d'entrée en suivant les liens IS_A de leur cible vers leur source en adoptant comme philosophie de parcours celle de l'activité (Cf. paragraphe 2.3.1). Tous les fragments non encore dérivés sont ensuite examinés.

Chaque fragment donne lieu à la création d'au moins une règle E-C-A, en général plusieurs, sauf dans le cas où il est réduit à un TES. De façon schématique, lorsqu'un type est origine d'une fonction de précédence, sa traduction consiste en une clause événement, éventuellement complexe si le type est composite.
Un type cible d'une fonction de déclenchement est traduit, dans le cas le plus simple d'une composition ou séquence dont les fils sont des TES, par une séquence d'actions.
Lorsque le type composite généré est une union ou un groupement, il convient pour prendre en compte l'alternative ou la répétition d'actions, de produire de nouvelles règles déclenchées ou activées par les actions appropriées spécifiées dans la règle reflétant le comportement du fragment. Dans le cas d'une union, il y a autant d'actions que de fils composants et il y a génération pour chacune d'elles d'une nouvelle règle E-C-A. Dans le cas d'un groupement, une seule action est générée et une seule nouvelle règle E-C-A est introduite, mais nous l'appelons récursive car son exécution doit être répétée plusieurs fois, afin de traduire la multiplicité des événements modélisés en IFO_2. Pour conférer ce caractère récursif à la règle engendrée, nous la réactivons systématiquement après chaque exécution et la privons d'événement déclencheur.

Les TER sont traités d'une manière particulière car leur rôle est, dans IFO_2, de représenter un autre type dans un souci de modularité des spécifications. S'ils participent à l'origine d'une fonction de précédence d'un fragment, leur traduction influe sur la partie Evénement de la règle dérivée du fragment. En fait, cette partie comprend une disjonction d'événements correspondant aux différentes sources du TER. S'ils participent à la cible d'une fonction de déclenchement d'un fragment, c'est au niveau des actions de la règle générée que leur traduction a des répercussions. Avec la même philosophie que pour la dérivation des types unions, il y a création d'autant d'actions de déclenchement ou activation de nouvelles règles qu'il y a de sources pour les TER et ces nouvelles règles sont ensuite

générées.

Enfin, les caractéristiques des fonctions influent sur la dérivation effectuée. Les fonctions de déclenchement différées d'un fragment peuvent introduire un mode de couplage différé entre la composante Evénement et la composante Condition. Elles peuvent aussi être traduites dans le cas d'une génération de règle récursive, par l'activation (au lieu du déclenchement) d'une nouvelle règle. Les fonctions complexes sont traduites dans le même esprit que les types groupements, puisqu'elles représentent une itération au niveau des types d'événements cibles, i.e. via la génération de règles récursives déclenchées par une action de la règle correspondant au fragment.

Dans le cadre du projet d'ASP (Action de Soutien Programmé), un de nos objectifs était d'enrichir les cibles potentielles d'implantation du comportement, en proposant une génération de code pour le système actif NAOS [47, 46]. Dans ce cadre, il était important d'examiner et comparer les systèmes HiPAC et NAOS. Ces deux SGBD étant actifs, ils ont en commun divers concepts et mécanismes fondamentaux. Cependant, ils diffèrent par divers éléments significatifs que nous détaillons dans le paragraphe suivant.

HiPAC vs NAOS

Nous nous intéressons aux seuls éléments de différence entre ces deux systèmes, significatifs par rapport à notre problématique de génération de règles actives. Après quelques éléments descriptifs, nous explicitons les problèmes posés. Les solutions adoptées et les algorithmes associés sont détaillés dans [60].

Spécification des événements
Dans HiPAC, les événements primitifs considérés sont : des opérations sur la BD (création, suppression, modification, lecture et appel de méthodes (événements de début et fin d'exécution)) et des événements temporels et externes.
Dans NAOS, les mêmes types d'opération sur la base sont proposés (création, suppression et modification d'objets ou de valeurs y compris complexes, lecture (avec distinction entre types simple et complexe), appel de méthode (début et fin d'exécution)). Des événements de type "programme", "application" et "utilisateur" peuvent être spécifiés. Pour ceux des deux premiers types, deux événements peuvent être détectés : le début et la fin d'exécution. Les événements "utilisateur" correspondent à des appels de méthodes O_2 ("*signal*") qui peuvent être intégrés soit dans la partie Action d'une règle soit dans un programme, une transaction, une fonction ou une méthode. Les événements temporels peuvent être : absolu, périodique ou relatif. Les événements externes ne sont pas pris en considération.

Dans les deux systèmes, des événements composites peuvent être définis. Dans HiPAC, les constructeurs d'événements sont : la disjonction, la séquence et la fermeture (n occurrences d'un même type d'événement).
Dans NAOS, les disjonctions d'événements peuvent être exclusives ou pas et les séquences peuvent être contraintes (strictes). Sont également proposées : la négation d'événements et l'itération (choix du nième événement dans un ensemble).

Les principales difficultés rencontrées, lors de la réécriture des spécifications d'événements, sont les suivantes :

- Absence d'événements externes dans NAOS ;
- Richesse des événements temporels dans NAOS ;
- Richesse des constructeurs proposés par NAOS ;
- Absence de certains constructeurs dans NAOS.

Spécification des conditions et actions
Dans HiPAC, les conditions sont des expressions du langage de requête du SGBD (i.e. un ensemble de requêtes). Les actions peuvent être des opérations de manipulation de la base ou des appels de programmes. Dans NAOS, les conditions sont exprimées en utilisant une extension du langage de requête qui permet de conserver le résultat d'une requête afin de le réutiliser dans la partie Action de la règle concernée (introduction d'une clause into). Cette partie Action consiste en code O_2C.
Les conditions proposées dans IFO_2 sont de deux types : des conditions simples portant sur les valeurs d'objets et des conditions historiques portant sur les événements survenus au cours de la vie de l'application. C'est sur ce deuxième aspect [3] qu'IFO_2 et les systèmes actifs comme HiPAC et NAOS diffèrent le plus. En effet, l'expression des contraintes historiques est ramenée dans les systèmes actifs à la formulation de requêtes sur la base. La connaissance d'un fait du passé revient à lui associer un état remarquable (valeur d'attribut ou combinaison de telles valeurs), cette tâche étant bien sûr à la charge de l'utilisateur. Sur ce plan, la philosophie d'IFO_2 est totalement différente, puisque ces contraintes sont exprimées sous forme d'expressions événementielles.

Spécification des modes de couplage
Dans HiPAC, quatre possibilités sont offertes pour le couplage aussi bien Evénement-Condition que Condition-Action : immédiat, différé, détaché dépendant, détaché indépendant. Dans NAOS, seules les deux premiers modes de couplage sont offerts (pas de mode séparé).

IFO_2 proposant une description conceptuelle du comportement, nous ne nous préoccupons pas de la spécification des transactions. Cependant, certains éléments des schémas IFO_2 (en particulier l'éventuel caractère différé des fonctions) nous permettent de préciser des modes de couplage immédiat et différé, qui sont proposés par NAOS.

Les mécanismes d'enchaînement de règles
Les mécanismes offerts pour spécifier l'enchaînement des règles sont essentiels dans l'algorithme de génération proposé. Ils permettent en effet de lier les règles entre elles et déchargent de cette tâche le programme d'application. Ce dernier est donc extrêmement succinct, voire inexistant si la cascade de déclenchement peut être initiée. Les mécanismes de HiPAC utilisés dans l'algorithme de dérivation sont : fire et enable qui permettent à une règle de déclencher ou d'activer une autre règle. NAOS propose les possibilités : enable et disable, pour activer et désactiver une règle. Elles sont utilisables dans la partie Action des règles.

3. Dans l'algorithme de génération de règles HiPAC, les conditions ne sont pas considérées.

2.3.4 Composante Vérifications Comportementales

Malgré le respect des règles de cohérence définies, la représentation dynamique d'une application peut encore comporter diverses anomalies, conduisant à un fonctionnement critique du système modélisé (interblocage, cycles infinis, etc.). Une telle problématique est abordée dans différents domaines comme les bases de données actives ou déductives, les systèmes experts ou encore la spécification de systèmes concurrents, avec certaines préoccupations communes, ne serait-ce que le problème de terminaison [6, 33, 75, 76, 84]. L'intérêt de situer ce type de contrôles à un niveau conceptuel est évidemment de limiter les anomalies avant toute génération ou écriture de code. De plus, lorsqu'une anomalie est détectée, elle est localisée dans le schéma événementiel IFO_2 examiné. Nous pensons qu'à travers cette représentation de haut niveau, le concepteur peut mieux appréhender, et donc corriger, les erreurs de modélisation. Les vérifications sont présentées dans [40] et une version étendue est proposée dans [41]

Ainsi, les concepteurs d'applications peuvent trouver, dans l'approche de vérifications proposée, non seulement la garantie de certaines propriétés de fonctionnement du système modélisé, mais aussi une aide à la correction des anomalies observées. En aval dans notre démarche de développement, la composante de dérivation peut alors être perçue comme un mécanisme de production de code mais également comme la base d'un outil de validation permettant d'effectuer diverses simulations, tout particulièrement pour les applications non transactionnelles.

Le niveau de vérification met en œuvre différents contrôles dont nous distinguons deux catégories :

- Les premiers sont largement inspirés des travaux proposés pour certains modèles de bases de données actives, en particulier Starbusrt [5]. Ils visent à contrôler les propriétés de terminaison, confluence et déterminisme observable, garantes d'un bon fonctionnement du système. L'adaptation de ces vérifications à un modèle conceptuel apporte l'uniformité de la représentation pour l'utilisateur, car il est inutile de construire une autre représentation que celle qu'il a lui-même élaborée (par opposition au graphe de déclenchement des règles construit dans les analyses statiques des modèles actifs).
- Deux autres possibilités originales sont proposées. La première appelée contrôle de la "déclenchabilité des événements" permet de vérifier que "tout est utile" dans les spécifications proposées par le concepteur. La seconde vise à contrôler leur "cohérence" historique. Elle permet de détecter des interblocages liés à des séquences de déclenchements contradictoires.

Les contrôles de la seconde catégorie sont, dans notre contexte, particulièrement intéressants car ils seraient très délicats à mettre en œuvre au niveau d'un ensemble de règles actives. C'est pourquoi nous avons choisi de les développer dans les paragraphes suivants. Quant aux premiers, nous les résumons ci-dessous, en précisant au préalable le contexte dans lequel ils s'inscrivent.

En offrant divers mécanismes permettant d'enchaîner les types d'événements, le modèle IFO_2 met un accent particulier sur les cascades de déclenchements. Ces cascades sont re-

présentées par des *chemins d'activité*. Chaque chemin correspond à une navigation au sein d'un schéma et décrit un comportement type possible pour le système sous-jacent. Un schéma peut inclure un ou plusieurs chemins. En fait, par rapport à l'ensemble de règles actives générées à partir d'un schéma IFO_2 et qui sont enchaînées pour traduire les liens de déclenchement, un chemin représente les exécutions autorisées de règles. Ainsi, en construisant les chemins d'activité d'un schéma, il est possible de simuler le comportement futur de son ensemble de règles équivalent et donc d'anticiper ses éventuels dysfonctionnements en détectant certaines situations critiques.

En premier lieu, nous pouvons vérifier que l'activité peut être initiée, i.e. qu'il existe, dans le schéma, au moins un type d'événement déclenchable spontanément et notifiant au système qu'un fait réel est survenu. De tels types d'événements, qui peuvent être externes, temporels ou représenter une combinaison des précédents, sont appelés les *Entrées* du schéma. A partir de ces entrées, les cascades de déclenchement atteignent les différents types dans les fragments concernés et sont orientées par les fonctions de précédence et déclenchement. Elles se propagent alors aux autres fragments via les liens IS_A les reliant. La navigation le long des liens IS_A peut être effectuée du cœur de fragment vers le type représenté ou inversement. Pour résumer son principe, le premier type déclenché déclenche l'autre. Enfin, les cascades de déclenchement se terminent au sein de *Fragments de satisfaction*. En effet, la modélisation IFO_2 impose au concepteur de spécifier le ou les différents objectifs que doit atteindre le système, à travers de tels fragments.

En examinant tous les chemins d'activité d'un schéma, il est alors possible de répondre aux questions suivantes :

- "Les types d'événement sont-ils tous au moins inclus dans un chemin d'activité ?" ou, dans les termes du développeur d'applications actives : "Existe-t-il des règles indéfiniment indéclenchables ?"
- "Existe-t-il des cycles dans les chemins d'activité ?" ou "Existe-t-il des règles dont le traitement peut être infini ?"
- "L'état final du système mais aussi la perception de ses réactions par l'utilisateur dépendent-ils du chemin d'activité suivi ?" ou "Peut-on garantir la confluence et le déterminisme observable d'un ensemble de règles ?"
- "Chaque fois qu'un type d'événement peut être atteint par différents chemins, y-a-t'il ou non interblocage historique ?" ou "Un interblocage peut-il se produire, au cours du traitement des règles, à cause de clauses contradictoires entre des règles interagissant ?"

Pour le programmeur d'applications actives, la garantie de la terminaison du traitement des règles est cruciale. Transposée dans notre contexte conceptuel, la problématique est de montrer la terminaison de toute instance d'un chemin d'activité. L'ensemble des types d'événements d'un schéma IFO_2 étant fini, ainsi que celui des fonctions et des liens IS_A, les seuls chemins d'activité pouvant être infinis proviennent de cycles. La détection de tels cycles peut être facilement mise en œuvre lors de la construction des chemins d'activité. De plus, ils apparaissent clairement dans la représentation graphique associée.
Cependant l'examen des cycles détectés est à la charge du concepteur qui doit vérifier s'il existe des conditions d'arrêt pour l'itération des déclenchements de types et si elles peuvent être satisfaites.

La vérification de confluence et de déterminisme observable est largement inspirée de l'approche développée dans Starburst [6]. Dans notre contexte, la confluence signifie que l'état final du système ne dépend pas de l'ordre dans lequel sont exécutés les chemins d'activité. Pour la mise en œuvre d'un tel contrôle, les types simples invoquant des opérations bases de données doivent être localisés dans les divers chemins d'activité concernés. Par l'examen de leurs paramètres et en s'appuyant sur le principe de commutativité de règles de [6], il est possible de détecter des types en conflit par rapport à la confluence.
Lors du contrôle du déterminisme observable, les types d'événements simples correspondant aux opérations observables par l'utilisateur sont mis en évidence au cours de la construction des chemins. Les paramètres de ces types sont à nouveau examinés pour établir cette propriété.

Dans les paragraphes suivants, nous nous focalisons sur les deux vérifications originales proposées.

Déclenchabilité des événements

Pour vérifier la déclenchabilité des événements, l'algorithme défini transpose les techniques de vérification de l'atteignabilité des états [35, 36, 37] (ou d'instructions dans un programme [109]). Ces dernières procèdent au parcours des différents diagrammes d'états modélisant le système afin de s'assurer que tout état peut être atteint lors de ce parcours.

Dans notre contexte, l'objectif est de vérifier que tout type d'événements peut être effectivement déclenché, au cours de la vie de l'application. Ce premier niveau de contrôle permet de localiser des parties de spécification qui ne seront jamais utilisées.

Pour contrôler que "tout est utile" dans la représentation du concepteur, l'algorithme proposé réalise un parcours du graphe que constitue le schéma IFO_2, en mettant en œuvre les principes de propagation de l'activité des schémas, exposés dans le paragraphe précédent. Ainsi, à partir des entrées d'un schéma, tous les parcours autorisés sont suivis, i.e. tous les comportements possibles du système sont simulés. Tout type d'événements ne figurant pas dans un de ces parcours est alors déclaré indéclenchable et rien ne justifie sa spécification dans la modélisation proposée.

Remarquons que les erreurs décelées sont plus difficiles à percevoir avec une représentation de la dynamique basée sur les états, pour deux raisons essentielles : les événements ne sont pas typés et la "reconstruction" d'une cascade de déclenchements nécessite le parcours des diagrammes des différents objets et, surtout, l'examen des conditions de déclenchement.

Au niveau de l'implantation, les anomalies observées induisent des règles E-C-A n'étant jamais activables ou jamais déclenchées. Leur détection nécessiterait, outre un examen exhaustif de l'ensemble des règles spécifiées, une analyse très fine non seulement de la partie Evénement de ces règles mais surtout des instructions de leur partie Action, qui peuvent directement ou indirectement (par exemple par le biais d'opérations de mise à jour) provoquer le déclenchement d'autres règles. Signalons enfin qu'un tel contrôle est très étroitement lié au modèle actif visé, car les mécanismes proposés pour l'enchaînement des règles varient d'un système à l'autre.

Exemple 10 Considérons le schéma de la figure 2.7. La seule entrée de ce schéma, le type externe "Demande-Etage", permet de déclencher tous les autres types du même fragment, i.e. "Fermeture" et "Mouvement" (avec ses composants), ainsi que le type représenté "Aller-Etage" du second fragment (par propagation le long du lien IS_A).
Dans ce deuxième fragment, dépourvu de fonction de précédence, le cœur "Stop" ne peut être déclenché que par le déclenchement de ces deux types composants : "Aller-Etage", déclenchable via "Demande-Etage", et "Arrivée-Cabine" qui étant un type simple ne peut jamais être déclenché.
Ainsi, "Arrivée-Cabine" et "Stop" sont indéclenchables et par conséquent les types "Arrêt" et "Ouverture", cibles de fonctions de déclenchement, le sont aussi. □

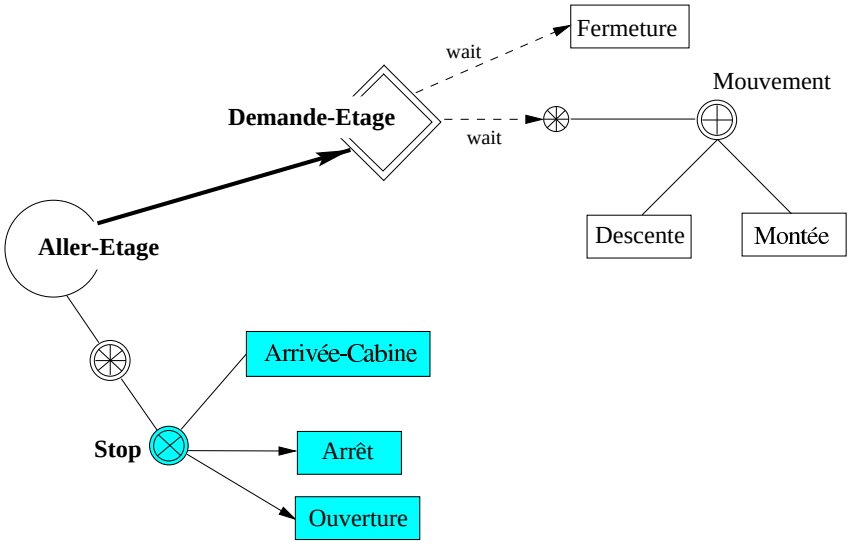

FIG. 2.7 – *Exemple de schéma avec types indéclenchables*

Vérification de cohérence historique

Les anomalies auxquelles nous nous intéressons dans ce paragraphe peuvent survenir lorsque plusieurs chemins d'activité concurrents se rejoignent en déclenchant un type d'événement précis dans un fragment donné.
Dans certains cas, un des chemins peut exprimer l'"histoire" du type en question, i.e. ce qu'il s'est passé avant le déclenchement de ce type. S'il y a contradiction entre l'histoire et un autre chemin, ce dernier est dit historiquement incohérent. Ces anomalies historiques induisent des comportements absurdes, aussi, quand elles sont détectées, elles doivent être corrigées par le concepteur.

La mise en œuvre de contrôles détectant le même type d'anomalies dans le contexte d'une représentation de la dynamique par diagramme d'états, se heurte aux mêmes problèmes que précédemment, i.e. la difficulté de reconstituer les cascades de déclenchements et la

nécessité d'une analyse des conditions des transactions.

L'implantation d'un schéma incluant un tel chemin, s'il était traduit sous forme d'un ensemble de règles E-C-A conduirait à une suspension non voulue, voire un blocage, des déclenchements des règles. Ceci survient quand une règle en déclenche une autre, qui ne peut, à cet instant là, être déclenchée parce que "quelque chose" doit arriver avant. Par "quelque chose", nous entendons l'occurrence d'un événement, le déclenchement d'une règle, l'exécution d'une opération ou une combinaison de ces divers faits. De telles erreurs sont délicates à isoler au niveau implantation, car il s'agit de réinterpréter le comportement réel représenté pour le comparer au résultat attendu. Elles sont par contre localisables, de manière automatique au sein d'un schéma IFO_2.

Pour détecter les chemins historiquement incohérents, l'algorithme de vérification proposé calcule d'une part les chemins d'activité et d'autre part l'histoire de chacun des types dans un schéma. Cette histoire est locale, i.e. qu'elle ne peut inclure que les types d'événement du fragment examiné. En effet, puisque les fragments sont concurrents a priori, l'histoire d'un type est simplement la séquence des types obligatoirement déclenchés, dans son propre fragment, pour le rendre à son tour déclenchable. Il est particulièrement facile de calculer l'histoire des types car elle est dictée par les fonctions de précédence et de déclenchement. Ainsi, par nature même, un type source d'une fonction de précédence n'a aucune histoire. L'histoire possible d'un cœur de fragment est restreinte à son éventuel type précédent. Enfin, les types cibles de fonction de déclenchement ont forcément une histoire incluant au moins le cœur de fragment auquel ils sont liés.
De manière générale, l'algorithme défini comprend trois étapes et s'applique après la détection des types d'événement non déclenchables.
La première étape calcule, pour chaque fragment du schéma, ce que nous appelons "l'historique" des types d'événements. Il s'agit simplement de construire, pour chacun des types d'événements, la séquence des types qu'il a fallu obligatoirement déclencher, au sein de son fragment, pour que lui-même devienne déclenchable. Ainsi, un type source d'une fonction de précédence n'a, par définition même, pas d'historique. L'historique d'un cœur de fragment ne peut, s'il existe, qu'être réduit au type précédent du fragment. Les types déclenchés ont, quant à eux, forcément un historique incluant au moins le cœur du fragment.
La deuxième étape de l'algorithme met en œuvre les principes de l'activité de schéma IFO_2 pour calculer les chemins d'activité, i.e. tous les comportements possibles. Son principe de parcours du schéma est donc analogue à celui du premier algorithme décrit. Pour chaque type, est construit l'ensemble des séquences conduisant à son déclenchement. Evidemment, les premiers fragments examinés sont ceux comprenant une entrée de schéma et pour chaque type de tels fragments, s'il n'existe qu'une seule séquence d'exécution, elle sera identique à l'historique du type considéré. Pour les autres fragments, les chemins d'activités sont construits en naviguant dans le schéma comme nous l'avons détaillé dans le paragraphe 2.3.1.
En s'appuyant sur les résultats précédents, la dernière étape réalise le contrôle de validité dont le principe est le suivant : pour que la spécification soit valide, il faut que, pour chaque type, toute séquence d'exécution inclue l'historique du type en question. En effet, comme nous l'avons précédemment souligné, l'historique d'un type a un caractère obligatoire : pour qu'un type soit déclenchable, il faut impérativement que tous les types de son historique aient été successivement déclenchés. Aussi, une trace d'exécution n'incluant

pas cet historique relève d'une erreur de conception : un comportement possible ne respectant pas l'histoire imposée. Lors de ce contrôle, les types représentés constituent un cas un peu particulier. Outre leur "propre histoire" déterminée par leur fragment, toutes leurs séquences d'exécution doivent aussi inclure l'historique hérité du ou des types qu'ils symbolisent. C'est ainsi que nous vérifions la compatibilité entre fragments.

Dans un souci d'optimisation, l'algorithme défini ne calcule par systématiquement toutes les traces d'exécution pour tous les types. En fait, la construction d'une trace se faisant pas à pas (i.e. type après type), elle est suspendue dès qu'elle ne respecte plus la contrainte d'inclusion de l'historique du type examiné. Ainsi seules les traces autorisées sont construites. Dans ce contexte, les seuls types qu'il convient d'examiner lors de la troisième phase de contrôle, sont les types représentés afin de vérifier que toutes leurs séquences d'exécution possibles incluent l'historique hérité.

Exemple 11 La figure 2.8 propose une tentative de modélisation IFO_2 et une partie des règles actives automatiquement engendrées par la composante de dérivation du modèle est donnée figure 2.9.
Nous supposons, dans cet exemple, que les appels de l'ascenseur d'un palier et les demandes d'étage, de l'intérieur de la cabine, sont dissociés. Ces événements externes sont respectivement représentés par les types abstraits "Appel" et "Demande", chacun source d'une fonction de précédence dans un des fragments du schéma. Ces deux types sont donc des entrées du graphe événementiel IFO_2.
Imaginons un comportement du système ainsi spécifié, en supposant qu'un événement de "Demande" est émis. Il provoque l'occurrence d'un événement de "D-Mouvement', et, par propagation le long du lien IS_A, l'émission d'un événement de "Mouvement". Cependant, pour que ce dernier soit à son tour déclencheur, i.e. qu'un ordre de montée ou de descente soit effectivement donné, il faut impérativement que le type précédent de "Mouvement", i.e. "Appel", soit déclenché.
Ainsi, avec la modélisation proposée, un usager ayant sélectionné un étage dans la cabine doit attendre pour que celle-ci se mette en mouvement qu'un autre usager appelle l'ascenseur. Evidemment, la même attente aberrante peut être observée dans le déclenchement des règles E-C-A (Cf. figure 2.9). Un événement de demande d'étage va permettre le déclenchement de la règle $r_{\texttt{D-Mouvement}}$ dont la première action consiste à déclencher la règle $r_{\texttt{Mouvement}}$. Or l'exécution de cette dernière ne peut être réalisée que si un événement d'appel intervient.

Considérons maintenant l'application de l'algorithme de vérification de la cohérence historique. L'historique du type "Mouvement" y est simplement réduit à sont type précédent "Appel". De manière symétrique, celui de "D-Mouvement" est constitué de son propre type précédent "Demande". Cet historique est aussi une trace d'exécution possible pour le type "D-Mouvement". Or cette trace ne contient pas l'historique hérité de "Mouvement", i.e. "Appel". Ainsi, les modélisations des types "Mouvement" et "D-Mouvement" sont détectées comme étant incompatibles." □

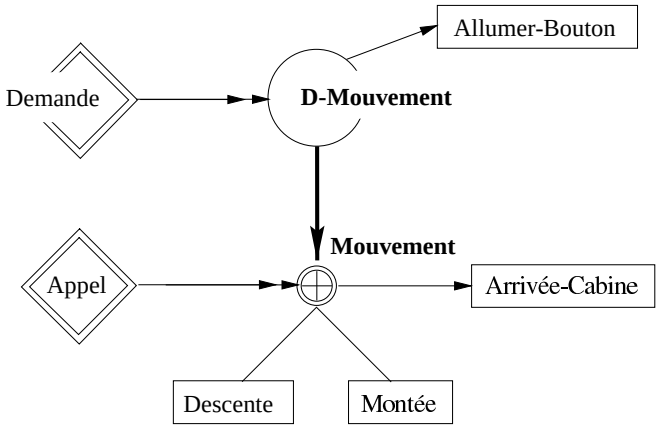

FIG. 2.8 – *Exemple de spécifications avec suspension de l'activité*

rD-Mouvement
E: Demande
Deferred
C: c_2
Immediate
A: fire rMouvement
Allumer-Bouton

rMouvement
E: Appel
Deferred
C: c_1
Immediate
A: fire rMontée
fire rDescente
Arrivée-Cabine

FIG. 2.9 – *Règles E-C-A associées au schéma de la figure 2.8*

2.4 Discussion

Dans les approches modélisant la dynamique des applications, le comportement est perçu en terme de comportement des objets (encapsulé dans la description de la classe) et d'interactions entre objets. C'est donc une vision complète des objets, structurelle et comportementale, qui est privilégiée au détriment d'une vue globale de la dynamique fatalement morcelée en autant de classes. Outre que cette philosophie s'accorde mal à celle de leurs systèmes d'implantation naturels, les SGBD actifs, elle n'est pas sans défaut dans un cadre conceptuel. La séparation entre comportements locaux et globaux prive le concepteur d'une appréhension générale du système et implique des mécanismes de représentation distincts (comme les événements partagés [110] ou les équations d'interaction [118]). A cela, nous opposons dans IFO_2 la définition d'un schéma événementiel dédié à la modélisation comportementale, traduite dans sa globalité, et complétant le schéma structurel.

La différence majeure entre les possibilités d'héritage dans IFO_2 et celles existant dans les autres approches est qu'une partie du comportement peut être réutilisé, dans notre modèle, sans être soumise à des contraintes structurelles. En effet, dans les autres travaux [99, 110, 118], un trait dynamique ne peut être hérité qu'entre deux objets liés structurellement : l'un doit être une spécialisation de l'autre.

Dans la plupart des approches, une sémantique, comparable à celles des types atomiques

d'IFO_2, est attribuée aux événements (externes, appels de méthode, etc.). Les conditions de synchronisation des événements sont, dans la majorité des modèles exprimées sous forme de combinaisons booléennes. Dans IFO_2, nous choisissons une approche par constructeurs, qui non seulement est au moins aussi expressive mais de plus participe à la vision globale du comportement, tout en restant en concordance avec les possibilités de construction d'objets structurels. Une telle démarche, alliée au caractère "tout-événement" d'IFO_2, permet de plus de spécifier la synchronisation des actions.
Enfin, l'expression de déclenchement d'événements est de manière générale, dans les autres travaux, traduite en termes de conditions d'occurrence (éventuellement pré et post-conditions). Sur ce plan, la philosophie d'IFO_2 est fort différente, car le double niveau de spécification préconisé permet de distinguer des contraintes globales d'enchaînement, apparaissant dans la nature des fonctions de fragment mais aussi au niveau des types construits utilisés, et des conditions précises de déclenchement explicitées dans la spécification des fonctions via le langage algébrique.

Les schémas événementiels IFO_2 ont, à l'image des graphes de transitions d'états largement utilisés dans les approches comportementales, un rôle de représentation des comportements possibles d'un système. Cependant, nous avons montré leur plus grand pouvoir d'expression, notamment par l'introduction de conditions de synchronisation, bien sûr entre les événements déclencheurs mais aussi entre les événements déclenchés. Dans les autres approches, les actions sont simplement perçues comme un enchaînement séquentiel d'appels de méthode et/ou d'émission d'événements. Outre l'obligation de spécifier un ordre chronologique entre ces éléments, même s'il n'existe pas dans le réel, le problème essentiel réside dans la représentation d'actions itérées. Pour le résoudre, IFO_2 propose un mécanisme de modélisation simple et explicite. Il en est de même pour les alternatives qui ne peuvent, en général, être spécifiées qu'à travers la description de plusieurs comportements locaux ou globaux. Les fonctions de précédence des fragments sont aussi un point fort des schémas IFO_2.
Une autre critique, formulé à l'encontre des graphes de transitions d'états [63] est l'impossibilité d'une conception modulaire, pas à pas, ascendante ou descendante. L'obligation de spécifier tous les états et toutes les transitions, y compris celles provoquées par un même événement (par exemple, une interruption de haut niveau) et conduisant à un même état final, rend rapidement illisible les graphes. D. Harel les juge même infaisables. A cela, nous opposons avec le concept de schéma événementiel, une représentation plus riche et extrêmement plus concise. De plus, nous conservons la qualité de lisibilité de toute représentation graphique, en évitant l'explosion exponentielle inhérente aux graphes d'objets [63].

A travers la composante de dérivation définie, IFO_2 peut apporter une réponse aux besoins des développeurs d'applications actives, en leur offrant les moyens d'organiser les déclenchements de règles ainsi que leur génération. Même si l'approche proposée ne vise pas l'exhaustivité, l'illustration au travers d'HiPAC et de NAOS, nous montre la faisabilité. Enfin, le choix d'une approche "tout-événement" pour IFO_2 n'est pas remis en cause par la définition de la composante dérivation, au contraire puisque non seulement les actions des règles actives peuvent être facilement déterminées mais surtout, le concepteur se voit proposer un cadre de travail totalement uniforme où son seul souci est la spécification de l'enchaînement des événements. Comparé à une transformation d'une approche basée sur

les états[4] , la dérivation dans IFO_2 permet de traiter les modes de couplages entre les différentes clauses d'une règle. En effet, la modélisation permet de prendre en compte la notion de délai inexistante dans UML par exemple. En outre, le fait d'avoir une vision globale, et non pas classe par classe, facilite grandement la transformation. En fait, les limites de l'approche de transformation à partir de graphes d'états proviennent des lacunes de ce modèle. Une extension intéressante serait par exemple de pouvoir spécifier des modes de couplage entre les différents éléments d'une transition et également de permettre les exceptions.

Les mécanismes de vérification présentés permettent, en amont de la dérivation, de localiser des erreurs de conception directement dans un cadre conceptuel. Par des techniques voisines, les systèmes de vérification proposés par les diagrammes d'états [38] détectent des erreurs de synchronisation entre objets mais ne peuvent isoler les anomalies que la composante de validation IFO_2 met en évidence. En effet, la constitution d'un historique, tel que nous l'entendons dans ce chapitre, i.e. un passage obligé pour le comportement, correspond à un chemin de contrôle [112] parcourant les différents diagrammes d'états dont l'expression fait appel aux conditions sur les transactions non prises en compte dans l'analyse de flux. Sans être exhaustif, le système de vérification défini permet la détection d'erreurs non triviales, délicates à percevoir dans les autres représentations comportementales ce qui nous conforte dans le choix d'une modélisation totalement événementielle pour IFO_2.

4. Des algorithmes de transformations à partir de diagrammes de transitions d'états via une représentation très proche de celle d'UML sont décrits formellement dans [96].

Chapitre 3

Modélisation de scénarios pour l'aide à la décision

Les applications auxquelles nous nous intéressons dans ce chapitre constituent une famille d'applications décisionnelles (puisque leur vocation est l'aide à la prise de décision) dotée de certaines spécificités notables (absentes dans d'autres types d'applications décisionnelles). L'idée générale du type d'applications visé est de représenter l'évolution de divers systèmes (dans notre cas des ouvrages) et d'extraire parmis ces évolutions modélisés les probabilités de défaillance d'un système. L'étape ultime vise quant à elle à définir une stratégie d'intervention. Ces applications mettant en type deux problématiques différentes : modélisation des systèmes et extractions de connaissances, nous nous focaliserons dans ce chapitre sur la première et décrirons la seconde dans la deuxième partie de ce mémoire. Pour contrôler l'évolution des systèmes observés, des scénarios de comportements doivent être élaborés par des experts. Nous montrons dans ce chapitre comment une approche basée sur une extension du modèle IFO_2 est parfaitement adaptée à la problématique. En outre, nous montrons comment stocker automatiquement les scénarios élaborés à l'aide d'une approche basée sur une méta-modélisation.

Dans ce chapitre, nous proposons de détailler l'aspect modélisation et stockage de scénarios. Après une présentation de la problématique liée à ce type d'application, nous proposons une synthèse de nos travaux dans le paragraphe 3.2. L'état de l'art associé à la modélisation comportementale est décrit dans le chapitre précédent. Enfin, en guise de conclusion, le bilan de la contribution est donné sous la forme d'une discussion.

3.1 Exposé de la problématique

La Modélisation

L'objectif des applications auxquelles nous nous intéressons est l'analyse de l'évolution d'une population, dont les individus peuvent être modélisés comme des objets éventuellement complexes. Cependant, pour contrôler l'évolution des individus observés, des scénarios de comportement type doivent être élaborés par les experts. Ces scénarios consistent en des enchaînements organisés d'événements éventuellement complexes. La problématique liée à de tels scénarios est double, puisqu'il s'agit, dans un premier temps, de pouvoir représenter

de manière adaptée ces scénarios (ce qui sous-entend notamment richesse et fidélité de la représentation, modularité, réutilisabilité de la description, ...), pour pouvoir ensuite les exploiter efficacement, en mettant en autre en œuvre des capacités de comparaison entre historiques particuliers et scénarios génériques.

Dans notre contexte, la modélisation structurelle doit permettre la représentation de la (ou des) population(s) étudiée(s). Leur description est "classique" dans le sens où les individus considérés sont caractérisés par un ensemble de propriétés intrinsèques (généralement stables dans le temps), qui peuvent être simples ou complexes. En termes de capacités d'abstraction, cette dimension structurelle peut être parfaitement représentée à travers les modèles statiques ou structurels des méthodes de conception classiques ou objets.

Il s'agit à présent de décrire la dimension comportementale. En effet, une fois l'historique de l'application type conservé, l'intérêt est de pouvoir comparer cette trace (historique) à des comportements types que nous appelons scénarios (ces "patterns comportementaux" pouvant aussi bien correspondre à de bons ou de mauvais comportements). Lorsque des individus adoptent réellement de tels comportements, s'ils peuvent être détectés automatiquement, alors il est envisageable de contrôler l'évolution dans le temps de notre population et d'anticiper les situations critiques, préalablement identifiées. Au vu des besoins évoqués, les modèles de représentation de la dynamique des systèmes d'information permettent la description du comportement d'un système en planifiant et organisant ses réactions. Qu'ils offrent des capacités de représentation de la dynamique individuelle des objets ou des processus globaux, ces modèles proposent des mécanismes permettant de séquencer et synchroniser les opérations à réaliser. Les premiers, orientés individus, semblent a priori mieux adaptés à notre problème. Ils intègrent la notion d'événement comme déclencheur des actions effectuées par les objets. Néanmoins, la richesse des concepts proposée peut s'avérer gênante sans parfaitement répondre aux besoins de modélisation nécessaires. En effet, la dynamique des objets s'exprime, dans ces modèles, en termes d'événements généralement élémentaires, d'opérations réalisées en réaction à ces événements et, fréquemment, d'états des objets qui constituent des pré et des post-conditions pour les opérations évoquées. Or, la représentation des scénarios d'évolution n'intègre pas forcément l'aspect réactif des objets. Dans ce contexte, la perception des états des objets peut s'avérer très délicate pour le concepteur. Les modèles dynamiques orientés processus ont, comparés aux précédents, l'avantage de proposer une vision unifiée du comportement en termes d'activité. Les mécanismes de synchronisation mis en jeu sont puissants. Cependant la notion d'événement n'existe pas dans ces approches (hormis les événements internes de début ou fin d'activité).

Basée sur une approche "tout-événement", l'approche IFO_2 semble être toute à fait adaptée à la modélisation de scénarios. Cependant, dans un contexte particulier, i.e. la modélisation de barrages, une adaptation des concepts est indispensable.

La Méta Modélisation

Si les modèles conceptuels dynamiques s'avèrent les plus probants pour décrire les scénarios, les outils d'implantation disponibles pour organiser les informations véhiculées par ces scénarios s'appuient quant à eux sur des modèles de bases de données, i.e. des modèles structurels relevant d'une représentation logique. Ainsi les difficultés de conception obser-

vées se doublent d'un problème d'implantation : comment implanter automatiquement les scénarios élaborés par un expert ? L'approche que nous proposons pour y remédier permet de constituer le dictionnaire de spécifications comportementales, en produisant les structures de stockage adéquates (basées sur tel ou tel modèle de base de données) et en les instanciant.

Nous détaillons dans le paragraphe suivant l'approche d'implantation des différents scénarios élaborés et qui est basé sur un méta-schéma. Puis nous nous intéressons au problème de la modélisation de tels scénarios.

3.2 Synthèse des contributions

Notre démarche, dont les principes généraux sont illustrés par la figure 3.1, tire profit de ces approches de modélisation et s'appuie sur l'élaboration d'un méta-schéma, décrivant les concepts dynamiques de l'approche de conception choisie en utilisant ses propres abstractions structurelles. Des approches par dictionnaire, ou plus généralement l'utilisation de méta-données, ont été depuis longtemps utilisées pour enrichir la sémantique, l'organisation ou l'objectif des données (certaines sont résumées dans [122]). C'est en se basant sur ce concept de méta-description que nous proposons notre approche. Une version préliminaire [27] propose le méta-schéma correspondant aux spécificités dynamiques de plusieurs méthodes de conception, dont OMT. Nous l'enrichissons ici en considérant le nouveau standard UML.
Lorsque le méta-schéma est spécifié, tout comportement d'application, modélisé par les concepteurs, peut être simplement perçu comme une des instances de ce méta-schéma. D'un point de vue implantation, les structures de stockage nécessaires peuvent être générées, à partir du méta-schéma, en utilisant les mécanismes de transformation fournis par la méthode conceptuelle sous-jacente. Cependant, ces fonctionnalités doivent être complétées afin de procéder au traitement des instances. En fait, une procédure d'instanciation automatique peut être définie. Elle s'applique aux schémas conceptuels dynamiques et crée les objets ou les tuples nécessaires dans les classes ou les relations générées à partir du méta-schéma.

Le dictionnaire dynamique de l'application, ainsi obtenu, peut alors être simplement manipulé par les développeurs, à travers le langage de requête du système cible choisi. En outre, sur ce dictionnaire peuvent être appliquées des techniques de fouilles de données de manière à extraire des patterns comportementaux utilisables à des fins d'aide à la décision.

En appliquant les principes donnés, nous examinons la mise en œuvre de notre démarche avec deux approches conceptuelles relativement éloignées : UML et IFO$_2$. La méthode UML, standard actuel, s'imposait d'elle-même dans notre choix. Contrairement à UML, la description dynamique d'IFO$_2$ met l'accent sur une perception globale : il s'agit de spécifier la réaction du système face à certains événements et non pas les réactions individuelles des objets des différentes classes. Tout en explicitant les cascades d'interactions entre objets (particulièrement délicates à appréhender avec les diagrammes d'états [112]), cette perception n'interdit pas, chaque fois que c'est possible, une représentation claire et naturelle du cycle de vie des objets (avantage reconnu des diagrammes d'états).

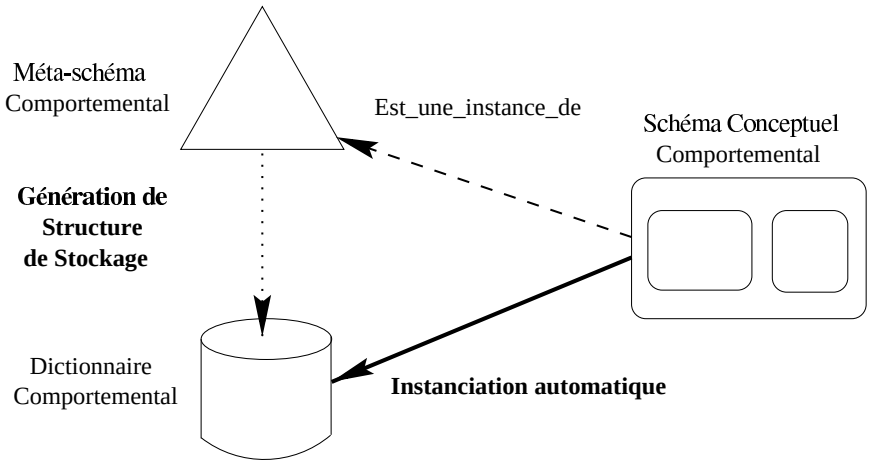

FIG. 3.1 – *Principes généraux de l'approche*

Appliqué au domaine des barrages, les modèles comportementaux nécessitent d'être redéfini ou affiné. C'est ce que nous proposons en décrivant les adaptations à apporter à IFO_2 pour offrit une méthode de modélisation de scénarios.

3.2.1 Composante Méta Modélisation

Dans UML [89], la dynamique des objets est modélisée à travers des diagrammes de transitions d'états. Le méta-schéma comportemental devant être spécifié doit permettre la représentation non seulement des concepts de base du modèle dynamique mais aussi des abstractions d'agrégation et de généralisation d'état. La figure 3.2 n'illustre que partiellement ce méta-schéma, car nous y mettons l'accent sur les aspects les plus intéressants rencontrés lors de la modélisation et ne détaillons pas la description de tous les concepts.

De par son caractère récursif, la généralisation d'état requiert des abstractions structurelles adéquates et une vision claire de la propagation de l'activité à travers les différents niveaux de description, puisque les diagrammes peuvent être arbitrairement imbriqués. Pour refléter ce mécanisme d'imbrication, nous utilisons l'agrégation récursive du modèle objet d'UML, et proposons une vision uniforme des états et des transitions. Plus précisément, un diagramme d'état (STATECHART DIAGRAM) est représenté par un état sommet (STATE), pouvant être spécialisé au niveau le plus bas, en un état simple (SIMPLE STATE) ou un pseudo-état (PSEUDO STATE), ou alors en un sous-diagramme (COMPOSITE STATE). Les pseudo-état modélisent simplement l'état initial et final d'une transition ; quant aux sous-diagrammes, ils sont modélisés comme une agrégation récursive d'états sommets (STATE), la notion de sous-diagramme concurrent étant spécifiée à travers le booléen isConcurrent. Chaque état sommet est relié à l'entité TRANSITION par trois associations de rôles différents (transition d'entrée , de sortie de diagramme, ou transitions internes à un état). Une transition dite "externe" (vs. les transitions internes) est ainsi reliée à deux états sommets (le diagramme d'entrée et de sortie), et est perçue comme une composition des éléments suivants : EVENT, CONDITION, ACTION. Ces derniers

correspondent respectivement, à l'événement déclenchant une transition, à son éventuelle expression conditionnelle, et à l'action qu'elle réalise. Avec ce style de modélisation, il est possible de représenter non seulement des transitions entre états mais aussi des transitions entre diagrammes de plus haut niveau, ou encore celles indiquant les points d'entrée et de sortie d'un diagramme. Lorsqu'elles sont spécifiées, ces dernières doivent obligatoirement être déclenchées pour entrer ou sortir d'un diagramme. Elles peuvent correspondre au début ou à la fin du cycle de vie d'un objet et jouent un rôle restrictif dans l'héritage des transitions d'un diagramme de plus haut niveau. En fait, à l'entrée dans un diagramme, les points d'entrée sont les seuls sous-diagrammes (ou états) pouvant être atteints. D'autre part, les points de sortie doivent être obligatoirement atteints pour pouvoir déclencher une transition sortant du diagramme.

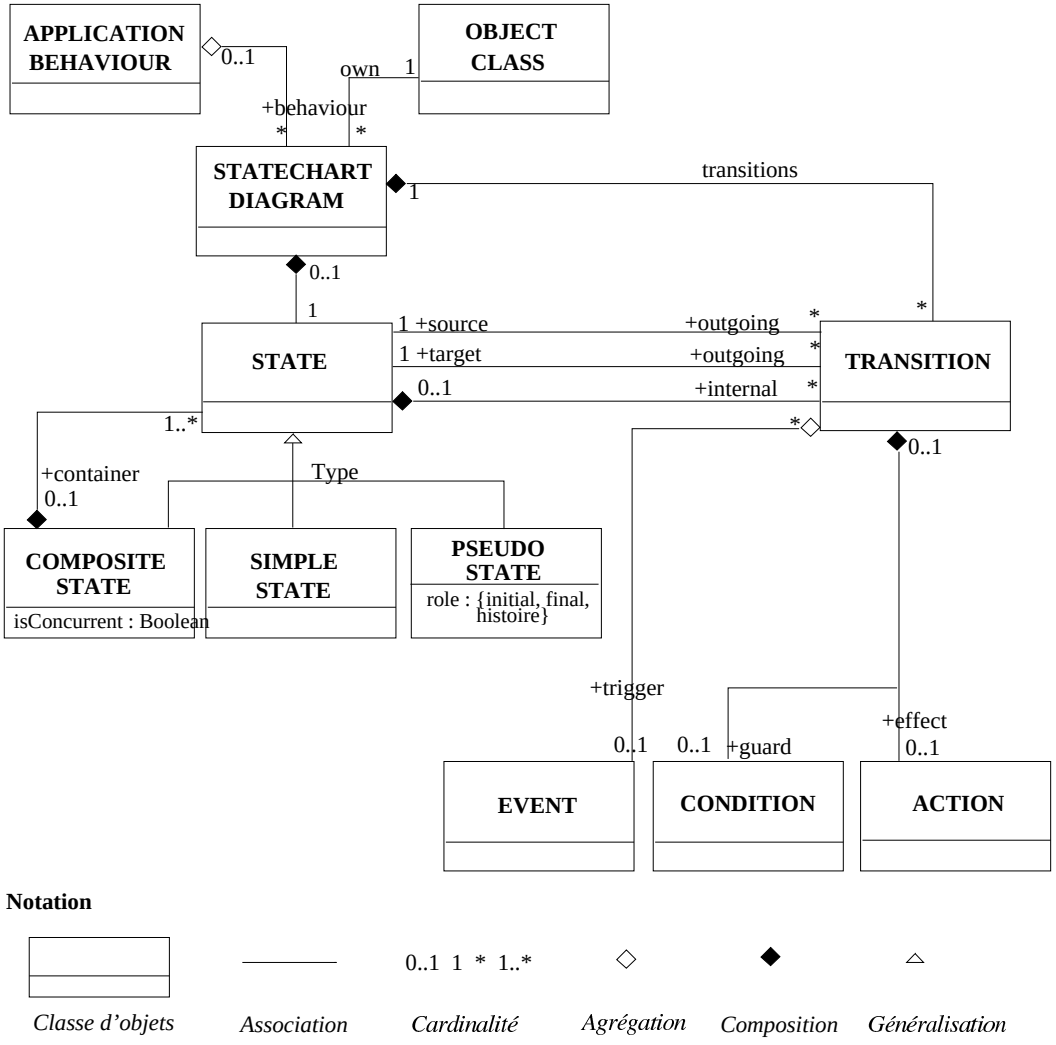

FIG. 3.2 – *Le méta-schéma comportemental avec UML*

Le méta-schéma étant spécifié, la deuxième étape de notre démarche s'applique en utilisant les mécanismes de transformation définis pour le modèle objet d'UML [89]. Dans

le cadre de notre travail, nous avons choisi le modèle relationnel comme cible de transformation. L'algorithme d'instanciation, décrit dans [13], produit donc le dictionnaire des spécifications comportementales de l'application.
Si le modèle dynamique d'UML est particulièrement bien adapté à la représentation des objets évoluant au cours du temps, il s'avère cependant inefficace pour la représentation de traitements orientés processus. Nous proposons ainsi, une deuxième expérimentation à travers l'approche IFO_2.

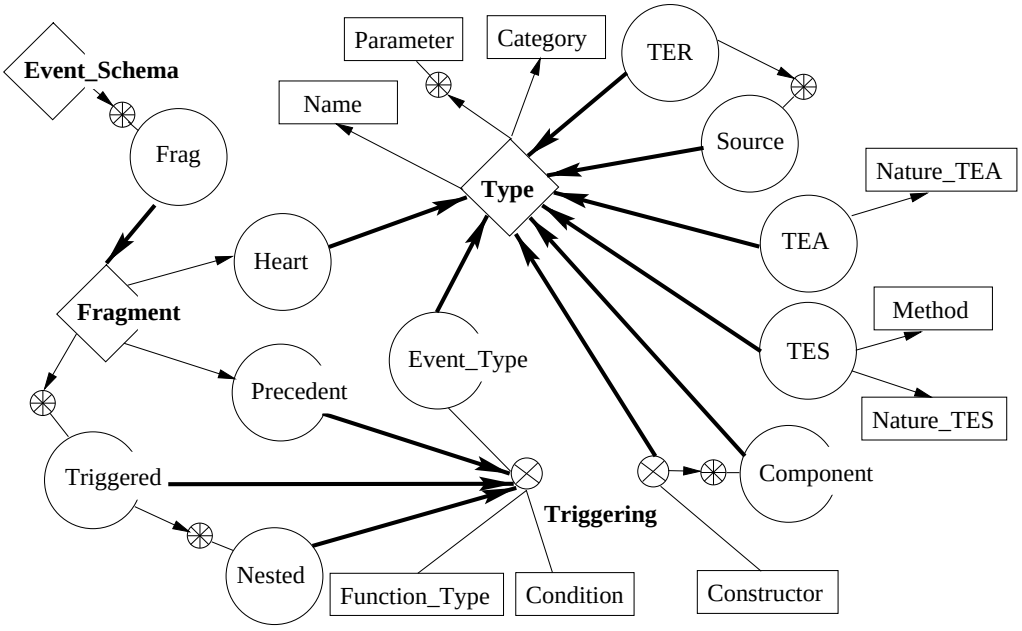

FIG. 3.3 – *Le méta-schéma comportemental avec IFO_2*

Le méta-schéma spécifié en considérant les concepts structurels d'IFO_2 est décrit par la figure 3.3. A partir des schémas structurels IFO_2, les mécanismes de génération de code peuvent être mis en œuvre, pour des systèmes relationnels, le SGBD orienté objet O_2 ou des langages comme C++. Pour créer les objets dans les classes appropriées, un mécanisme d'instanciation qui examine successivement les fragments événementiels est défini et décrit dans [27].

3.2.2 Composante Modélisation

Ce paragraphe est consacré à la mise en œuvre de la démarche proposée, à travers une application réelle. Elle s'inscrit dans le contexte plus général d'une des activités d'expertises menées par le CEMAGREF (CEntre national de MAchinisme Agricole du Génie Rural des Eaux et Forêts) : celle du diagnostic et traitement des pathologies de barrages.

L'objectif de ce projet est de calculer la probabilité de défaillance d'un système ou d'un ouvrage puis de la croiser avec l'évaluation des conséquences de cette défaillance. L'étape ultime vise à définir une stratégie d'intervention et de gestion adaptée, en tenant compte des résultats de la comparaison du "risque calculé" avec le "risque acceptable". Appliquée

d'abord dans le domaine de la sûreté des installations industrielles (centrales nucléaires, en particulier), la méthode commence à être étendue au domaine des barrages, sous l'impulsion d'experts anglo-saxons [56].

Le principal problème lié à la représentation de l'application Barrage tient à la grande diversité des informations manipulées, qui sont de différentes natures. Nous les scindons en trois dimensions : structurelle, historique et comportementale que nous détaillons dans les paragraphes suivants.

Représentation des dimensions structurelle et historique

Les données structurelles, descriptives des barrages, s'inscrivent dans divers domaines classiques (administratif, juridique, ...) ou techniques. Elles peuvent alors avoir trait aux divers éléments physiques constituant les barrages (corps, parements, évacuateurs, ...) ou correspondre à des caractéristiques de leur environnement (géologiques, climatiques, ...). Dans le premier cas, une approche de modélisation traditionnelle s'avère très bien adaptée, car il s'agit simplement de décrire des entités conceptuelles et les associations les reliant. Elle l'est probablement moins dans le second. En effet, la description physique d'un barrage peut nécessiter des capacités d'abstraction plus étendues, intégrant la spécification d'objets complexes.

Les données typiquement temporelles, décrivant l'histoire d'un barrage, peuvent être perçues comme des faits véhiculant une certaine information et estampillés par une date. La principale difficulté dans l'appréhension de cette dimension est liée au niveau de granularité de la description. En effet, les données historiques sont par nature même "inflationnistes" (un fait semblant anodin lors de son observation peut, des années plus tard, se révéler un symptôme significatif, si bien qu'il faut en garder la trace [42]), et les barrages peuvent être perçus comme des objets arbitrairement complexes dont chaque élément peut avoir sa propre histoire. A partir de ces rapports de visite, les faits considérés comme pertinents peuvent être isolés (par exemple : apparition d'un suintement sur le parement aval en telle année, installation de piézomètres tel jour, etc.) et raffinés en deux sous-catégories : les incidents et les interventions. La représentation des données historiques ne pose pas, a priori, de problèmes de modélisation spécifiques. En fait les approches conceptuelles classiques (a fortiori les plus récentes) permettent d'intégrer la dimension temporelle dans le schéma structurel d'une application et donc de représenter, et en particulier d'identifier, des faits historiques.

Représentation de la dimension comportementale

C'est dans cette dimension que s'inscrivent les scénarios de vieillissement des barrages. Un scénario est constitué d'un ensemble de phénomènes, intervenant au cours de la vie des barrages, ainsi que des relations les reliant. Fondamentalement, ces liens revêtent deux sémantiques différentes. Ils peuvent exprimer des relations de cause à effet entre phénomènes mais aussi des contraintes de synchronisation, organisant des combinaisons de "phénomènes causes" mais aussi "effets". Les phénomènes peuvent être externes au barrage considéré, i.e. provenir de son environnement (par exemple des crues, un séisme, etc.) ou internes. Ils peuvent, dans ce dernier cas, correspondre à des réactions du système, à son environne-

ment, mais aussi au temps.

En proposant une approche spécifique de représentation de scénarios basée sur le modèle IFO_2, notre objectif est de tirer parti de ses points forts tout en offrant une modélisation adaptée et naturelle pour le spécialiste. Cependant, pouvoir adapter IFO_2 à la modélisation de scénarios nécessite en premier lieu une comparaison des notions d'événements et de phénomènes.

Les événements dans IFO_2 sont définis par un triplet : (Id, Val, $Para$), où Id, l'identifiant de l'événement, est son instant d'occurrence, Val, sa valeur, est fonction du type de l'événement considéré et $Para$ est l'ensemble des objets structurels réagissant à l'événement considéré.
Adapter cette définition à la notion de phénomène impose d'avoir un identifiant dénué de sémantique particulière (l'instant d'occurrence d'un phénomène étant généralement inconnu). La notion de valeur d'événements doit également être reconsidérée. En fait, cette valeur correspond pour les types simples à la méthode invoquée, elle n'a donc, dans ce cas, aucun intérêt pour la représentation de phénomènes. Par contre, comme les événements, les phénomènes opèrent sur des objets structurels, en fait plus précisément sur un barrage ou certains de ses composants. Ainsi, la notion de paramètres doit être conservée. Cependant, un scénario décrit, par nature même, le cycle de vie d'un barrage et donc la dynamique particulière qu'il véhicule est toujours "orientée instance" (et jamais "orientée set" car, dans notre contexte, les traitements globaux sont inapplicables). Cette spécificité induit une simplification notable des paramètres d'un phénomène, qui ne peuvent alors référencer qu'un barrage, ou une ou plusieurs de ses propriétés ou composants. De plus, les phénomènes sont dotés de caractéristiques particulières. Leur caractère visible ou pas est primordial (dans une perspective de manipulation combinée des dimensions historique et comportementale). Ils ont une durée, sans que leur commencement et leur fin puissent toujours être précisément datés.

Outre l'adaptation de la définition des événements afin de représenter des phénomènes, les types de base proposés doivent également être examinés.
En respectant la philosophie du modèle, les types abstraits et représentés sont conservés avec la même sémantique. Bien que semblant à priori inutiles, les types simples sont également retenus mais ils revêtent dans notre contexte une autre sémantique. Ainsi, les types de phénomènes de base considérés dans notre approche sont les suivants :

- *les types simples* permettent de modéliser les préconisations incluses dans les scénarios. Bien que ne représentant pas l'invocation d'opérations effectives, ces préconisations d'interventions, réparations ou auscultations suggèrent ou recommandent des actions sur le terrain ;
- *les types abstraits*, permettant la représentation de phénomènes externes ou internes. Plus précisément, les types de phénomènes abstraits sont répartis en trois sous-ensembles *TPAE* (qui représentent les phénomènes externes pouvant survenir au barrage : les crues, les séismes, ...), *TPAI* (les phénomènes internes : les suintements, les dépôts, ...), *TPAI/R* (représentant les interventions ou les réparations sur les barrages);
- *les types représentés* introduits pour des besoins de modularité et réutilisabilité.

Pour décrire le comportement d'un système, il est essentiel d'exprimer des combinaisons éventuellement complexes de types d'événements. Les quatre constructeurs proposés (C.f. formalisme graphique figure 3.4), composition, séquence, groupement et union, permettent de spécifier les conditions de synchronisation sur les événements. Ces constructeurs sont conservés et définis de la même manière que dans IFO_2, avec cependant l'intégration de la portée dans la définition des types de phénomènes complexes.

Exemple 1 La figure 3.4 illustre un type de phénomène de type Composition. Plus précisément, le phénomène complexe est décrit comme une conjonction de deux phénomènes internes : "Suintements" et "Dissolution liant-béton". Le premier est visible et le second déduit, si bien que la valeur de la portée pour le phénomène complexe est $false$: il suffit qu'un de ses composants soit non visible pour que le phénomène complexe soit globalement considéré comme non visible. □

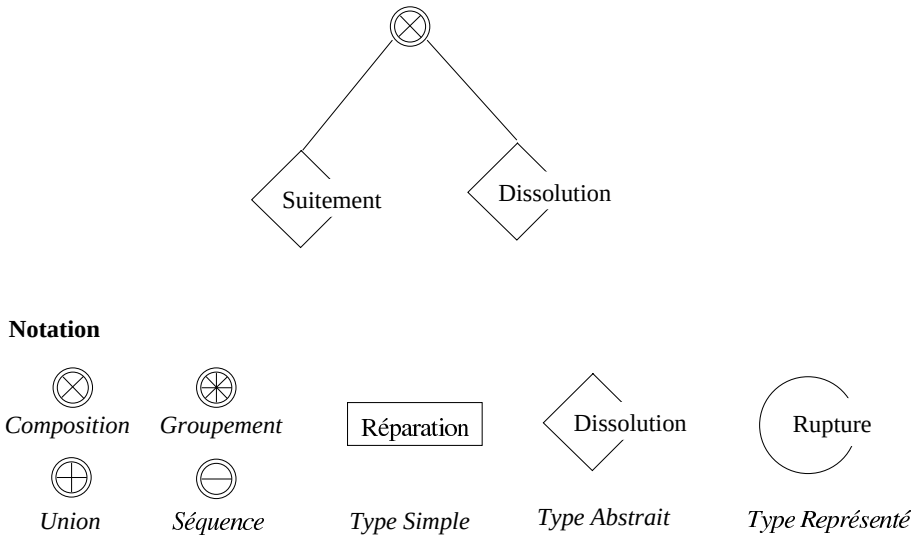

FIG. 3.4 – *Exemple de composition*

Les autres types de phénomènes complexes sont les suivants :

- *Type de phénomènes Séquence.* Le constructeur Séquence joue un rôle similaire à celui d'une composition et est défini de la même manière (donc même définition pour la "portée") à ceci près qu'il impose un ordre chronologique sur l'occurrence des phénomènes des types fils.
- *Type de phénomènes Groupement.* Le groupement représente une collection de phénomènes, i.e. une conjonction de phénomènes appartenant au même type (dans ce cas là, il n'y a aucune ambiguïté pour la définition de la portée, puisqu'il s'agit d'occurrence de même type).
- *Type de phénomènes Union.* Une disjonction de différents types de phénomènes est décrite par le constructeur Union. Dans ce cas, un phénomène composite ne peut

être engendré que par l'occurrence d'un unique phénomène des types composants. La valeur de la portée est donc égale à la valeur du phénomène composant concerné.

Les autres concepts du modèle restant inchangés, nous n'en redonnons pas la définition. Par contre, nous les utilisons pour la représentation d'un scénario à travers l'illustration suivante.

Exemple 2 La figure 3.5 propose la modélisation d'un scénario de vieillissement pour les barrages poids en béton. Les causes originelles du vieillissement sont ici des suintements ou fuites et une dissolution du liant. La combinaison de ces deux phénomènes peut entraîner des dépôts de calcite conduisant à une augmentation des sous-pressions dans le corps du barrage. Remarquons qu'un tel phénomène peut aussi résulter d'une dissolution du liant ayant provoqué d'autres types de dépôts. L'augmentation des sous-pressions peut provoquer une diminution de la résistance interne de l'ouvrage ou d'un plot puis, à terme, soit un mouvement du plot, soit des fissures mais aussi la rupture du barrage examiné. Ces trois conséquences, toutes visibles, sont modélisées via un type de phénomène Union. □

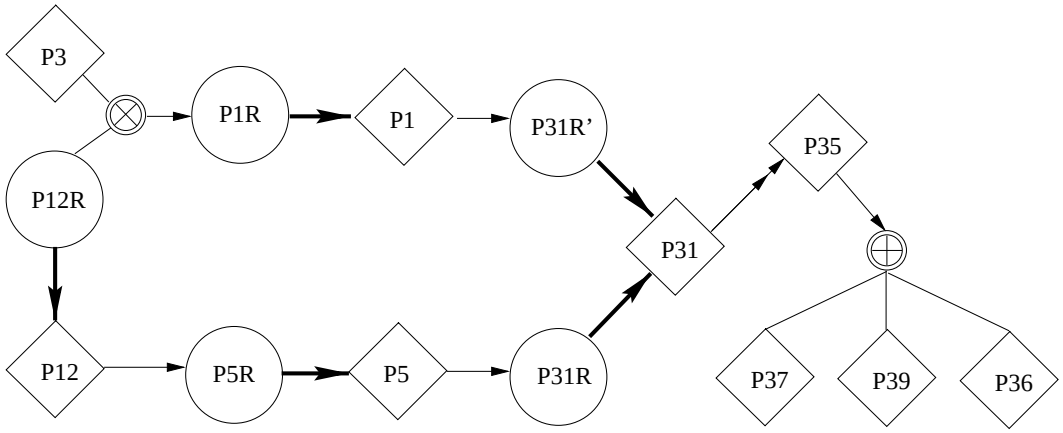

FIG. 3.5 – *Exemple de scénario pour les barrages en béton*

L'approche définie permet une modélisation adaptée et naturelle des scénarios de vieillissement et plus globalement de l'application barrage. Elle est actuellement utilisée au sein du CEMAGREF. Les experts utilisent donc l'extension d'IFO_2 pour modéliser le comportement des barrages et tous les scénarios sont automatiquement stockés, via la composante de méta-modélisation, pour pouvoir rechercher des probabilités de défaillance.

3.3 Discussion

Nous revenons, dans cette partie, sur les problèmes liés à la représentation des scénarios. Dans ce contexte, les concepts proposés dans UML s'avèrent délicats à utiliser. Intuitivement, les phénomènes de vieillissement peuvent être rapprochés de la notion d'événement,

P1	Dépôt de calcite sur zones courantes du parement aval	observable
P3	Suintements ou fuites sur zones courantes du parement aval	observable
P5	Dépôts de calcite sur ligne(s) verticale(s) du parement aval	observable
P12	Dissolution du liant du béton	déduit
P31	Augmentation des sous-pressions dans le corps de l'ouvrage	auscultable
P35	Diminution de la résistance interne du barrage	déduit
P36	Rupture	observable
P37	Mouvement irréversible d'un plot vers l'aval	auscultable
P39	Fissures (sub)verticales du corps de l'ouvrage	observable

TAB. 3.1 – *Liste des phénomènes*

déclencheur de transitions. Mais les phénomènes ont une durée (comme les états) alors que l'occurrence des événements est considérée comme instantanée. En fait, la principale difficulté provient de la dualité des concepts d'états et d'événements, tout aussi cruciaux dans la représentation dynamique, alors qu'un scénario peut être simplement perçu comme un enchaînement de phénomènes. L'absence de types d'événements est également une limite de l'approche, tout comme la perte de la vision globale du scénario s'il est suffisamment complexe pour être décrit à travers différents diagrammes (imbriqués ou non). Dans ce cas, c'est à l'utilisateur qu'incombe la tâche de "reconstituer" un scénario en suivant les transitions exécutées en séquence, le long des chemins de contrôle [112].

Le modèle n'offre pas de capacité de construction d'événements complexes, si bien qu'il est nécessaire d'avoir recours à la concurrence entre états et aux mécanismes de synchronisation associés. Le résultat est une représentation peu explicite et peu naturelle. Cette technique n'est, de plus, pas systématiquement utilisable car les états concurrents doivent faire référence à un même objet.
Enfin, les notions d'action et d'activité, correspondant aux opérations réalisées par le système, sont dénuées de sens, dans notre contexte. Mais leur "non utilisation" ne pose pas de problèmes spécifiques (contrairement aux états). Par contre, ce modèle permet la spécification de conditions globales sur les objets, au niveau de diagrammes d'états (typiquement un prédicat sur les objets de la classe dynamiquement décrite). Ainsi, il est possible d'associer à des transitions internes d'entrée de diagrammes (action "entry"), les conditions explicitant les objets concernés. A travers un tel mécanisme, peut être décrit le lien entre dimensions structurelle et comportementale de l'application.
Devant les difficultés rencontrées, pour certaines directement liées à une vision de la dynamique dirigée par les objets, il nous semble intéressant d'examiner une démarche classique, qui au contraire met l'accent sur les enchaînements d'événements.

De manière générale, le comportement d'un système est décrit avec IFO_2 comme un graphe de types d'événements qui peuvent être associés par des liens de différentes natures. La synchronisation des événements est spécifiée par le biais de constructeurs et les liens de causalité sont représentés sous forme de fonctions (avec deux sémantiques distinctes : déclenchement et précédence). Les enchaînements de déclenchement sont organisés de façon modulaire et réutilisable au sein de schémas événementiels. Ce modèle n'est évidemment pas dénué de limites ou défauts. Comme dans les autres démarches, les événements sont

considérés comme se produisant instantanément, et les types d'événements simples, symbolisant les opérations, sont inutiles pour la représentation de scénarios. Enfin, certaines spécificités du domaine d'application ne peuvent être prises en compte (le caractère visible ou non d'un phénomène par exemple). Malgré ces restrictions IFO_2 reste une base très probante pour définir une extension réellement adaptée aux scénarios.

En offrant à la fois la composante de modélisation de scénarios et la possibilité de stocker automatiquement ces scénarios, nous offrons une base de travail sur laquelle peuvent s'appliquer des techniques de fouille de données. En effet, à partir des différents comportements de barrages modélisés, il devient possible de rechercher par exemple les comportements de barrages les plus fréquents, i.e. ceux qui interviennent suffisamment souvent pour être intéressants pour un expert mais également des scénarios de comportements à risque. L'idée étant alors de comparer le comportement d'un barrage par rapport à ces scénarios afin de procéder au plus tôt aux réparations sur un ouvrage. Nous revenons dans la seconde partie de ce mémoire sur la recherche de tels comportements.

Conclusion

Pour conclure le travail présenté, nous nous proposons, dans ce chapitre, de dresser un bilan de l'approche IFO_2, en résumant ses points forts déjà signalés dans les précédentes discussions et en développant ceux dont l'argumentation nécessite une vision globale de l'approche.

Sur le plan de la modélisation structurelle, IFO_2 offre les avantages des modèles sémantiques classiques, comme l'indépendance par rapport aux modèles implantables, une vue complète du schéma ou encore l'explicitation des contraintes structurelles. De plus, la modularité permise lors de l'élaboration du schéma et la possibilité de réutiliser des parties via les concepts de fragment et de type représenté sont des qualités dont peut se prévaloir le modèle IFO_2 aussi bien que les approches orientées objet.
Sans être aussi riche que l'éventail de certains de ces modèles, les abstractions proposées sont celles reconnues comme pertinentes voire indispensables pour la modélisation d'applications avancées et sont véritablement conceptuelles. Ainsi, les compositions et groupement sont explicitement exprimés sans recours aux méthodes, éléments typiquement implantables, qui, en outre, peuvent être engendrées ultérieurement de façon automatique.
Les capacités d'évolution complétant la composante modélisation constituent un autre atout de l'approche IFO_2. Elles permettent, en effet, une spécification réellement incrémentale de schéma et déchargent le concepteur des contrôles de cohérence induits par toute remise en cause du travail élaboré. Elles contribuent donc largement à l'amélioration de la productivité lors de la phase de conception d'une application, tout comme celle de dérivation définie participe à celle de l'étape d'implantation. De plus cette dernière composante confirme le caractère d'indépendance d'IFO_2 par rapport aux modèles cibles. La traduction vers des modèles aussi différents qu'O_2 ou ORACLE en est une illustration parlante.
IFO_2 adopte la philosophie du "tout-objet', tout élément du modèle est doté d'un identifiant indépendant de sa valeur. Ce choix est, pour les modèles conceptuels, une originalité dans la mesure où la distinction classique entre une entité qui peut être identifiée et un attribut de l'univers réel n'est pas reflétée dans la représentation IFO_2 associée. Cette particularité n'est pas en elle-même gênante puisque le concept d'identifiant, relevant d'un niveau plus logique où il s'agit d'obtenir une représentation optimisée des données, est totalement transparent. Elle peut néanmoins dérouter un concepteur qui y chercherait un critère pour choisir le type d'objet IFO_2 adéquat pour la représentation d'un élément du réel. Une telle attitude doit, à notre avis, être proscrite en raison du caractère non conceptuel de telles options.

Le modèle comportemental IFO_2 adapte la philosophie "tout-objet" de la partie structurelle

en proposant une approche "tout-événement", avec les mêmes conséquences bénéfiques que sur le plan statique. Elle n'altère donc pas la fidélité de la modélisation structurelle. Les différents concepts introduits permettent de définir des schémas événementiels qui offrent une vue globale, concise et parlante du comportement du système (ou plus exactement des comportements possibles du système). Ces qualités cependant ne nuisent en rien à la puissance d'expression du modèle, au contraire, nous avons mis un accent particulier sur la richesse sémantique supplémentaire qu'il apporte par rapport aux autres travaux. Citons les conditions de synchronisation sur les événements déclenchés, alors que les autres modèles n'en permettent la spécification qu'au niveau des événements déclencheurs, mais aussi la nuance originale qu'introduisent les liens de précédence entre événements, ou encore, le caractère explicite des contraintes globales de synchronisation et la richesse des possibilités offertes à travers les constructeurs proposés.

Un autre point fort de la composante comportementale d'IFO_2 est qu'il propose les mécanismes permettant la modularité et la réutilisabilité des spécifications. C'est donc une véritable hiérarchie d'héritage dynamique, symétrique de la hiérarchie structurelle, qui est offerte au concepteur. Cette absence d'interférence entre les deux aspects d'une application est, selon nous, un meilleur gage d'efficacité lors de la conception que des possibilités de réutilisabilité cantonnées au comportement d'objets spécialisés.

Outre l'aspect de modélisation, IFO_2 intègre une composante de dérivation vers des règles E-C-A. Cette phase est une étape clef dans le processus de développement d'applications et même un passage obligé pour tout modèle se souciant d'optimisation de la réalisation. A notre connaissance, aucune transformation n'a été définie entre des spécifications comportementales de haut niveau et des BD actives. La proposition de cette dérivation répond à un véritable besoin d'aide à la spécification de règles E-C-A [46].
En proposant des mécanismes de vérifications du comportement modélisé, nous ne faisons que renforcer le besoin d'outils d'aide à la spécification de règles actives.

En nous intéressant à un type particulier d'applications décisionnelles, dont l'objectif est l'étude et le contrôle au cours du temps des "individus" d'une ou plusieurs populations dont l'évolution doit être surveillée, nous avons dégagé une double problématique relevant non seulement de la conception de bases de données, mais aussi de l'extraction de connaissances que nous allons aborder dans les chapitres suivants.
La méthode que nous avons proposée possède deux avantages intéressants : sa simplicité et une mise en œuvre peu coûteuse. Elle peut en effet être utilisée par différentes approches de modélisation ; nous avons notamment examiné cette démarche à travers deux approches conceptuelles relativement éloignées : UML et IFO_2. Nous avons également complété cette étude par la définition d'un modèle spécifique, basé sur IFO_2.

Résultats obtenus

Liste des publications

Thèse

P. Poncelet. "Contribution à la conception de bases de données avancées : modélisation, évolution et dérivation" Thèse de doctorat de l'Université de Nice Sophia-Antipolis, Mai 1993.

Publication dans un ouvrage

M. Teisseire, P. Poncelet and R. Cicchetti. "Events as Behavioral Modeling Drivers". In "Advances in Object-Oriented Modeling", M.P. Papazoglou, S. Spaccapietra and Z. Tari (Eds), M.I.T-Press, pp. 41-64, Octobre 2000, ISBN 0-262-16189-3.

Revues nationales avec comité de lecture

R. Cicchetti, P. Poncelet et M. Teisseire. "Modélisation et vérifications comportementales". Revue Ingénierie des Systèmes d'Information, Vol. 5, N. 3, pp. 265-285, Août 1997.

P. Poncelet, M. Teisseire, R. Cicchetti et L. Lakhal. "Conception Formelle de Bases de Données Avancées : le projet IFO_2". Revue Ingénierie des Systèmes d'Information, Vol. 1, N. 4, pp. 467-510, Décembre 1993.

Conférences internationales avec comité de lecture

F. Cathala and P. Poncelet. "Preserving Behaviour: Why and How?". Proceedings of the 9th International Conference on Advanced Information Systems Engineering (CAiSE'97), Lecture Notes in Computer Science, Springer Verlag, pp. 333-346, Barcelona, Spain, June 1997.

Z. Bellahsene, P. Poncelet, and M. Teisseire. "Views for Information System Design without Reorganization". Proceedings of the 8th International Conference on Advanced Information Systems Engineering (CAiSE'96), Lecture Notes in Computer Science, Springer Verlag, pp. 496-513, Crete, Greece, June 1996.

M. Teisseire, P. Poncelet, and R. Cicchetti. "Towards Event-Driven Modeling for Database Design". Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94), pp. 285-296, Santiago, Chile, September 1994.

M. Teisseire, P. Poncelet, and R. Cicchetti. "Dynamic Modelling with Events". Proceedings of the 6th International Conference on Advanced Information Systems Engineering (CAiSE'94), Lecture Notes in Computer Science, Springer Verlag, Vol. 811, pp. 186-199, Utrecht, The Netherlands, June 1994.

P. Poncelet, M. Teisseire, R. Cicchetti, and L. Lakhal. "Towards a Formal Approach for Object Database Design". Proceedings of the 19th International Conference on Very Large Data Bases (VLDB'93), pp. 278-289, Dublin, Ireland, August 1993.

P. Poncelet and M. Teisseire. "Advanced Database Modeling and Design : The IFO_2 Approach". Proceedings of the 3rd International Conference for Young Computer Science (ICYCS'93), International Academic Publishers, pp. 79-82, Beijing, China, July 1993.

M. Teisseire, P. Poncelet and R. Cicchetti. "A Tool Based on a Formal Approach for Object-Oriented Database Modeling and Design". Proceedings of the 6th International Workshop on Computer-Aided Software Engineering (CASE'93), IEEE Publishers, Singapore, July 1993.

P. Poncelet and L. Lakhal. "Consistent Structural Updates for Object Database Design". Proceedings of the 5th International Conference on Advanced Information Systems Engineering (CAiSE'93), Lecture Notes in Computer Science, Springer-Verlag, Vol. 685, pp. 1-21, Paris, June 1993.

Conférences nationales avec comité de lecture

R. Cicchetti, P. Poncelet et M. Teisseire. "Une aide à la conception et au contrôle de règles actives". Actes des 12ièmes Journées Bases de Données Avancées (BDA'96), pp. 19-34, Cassis, Août 1996.

R. Cicchetti, P. Poncelet et M. Teisseire. "Modélisation et validation comportementales". Actes du 14ième Congrès Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID'96), pp. 407-425, Bordeaux, Juin 1996.

M. Teisseire, P. Poncelet, R. Cicchetti et L. Lakhal. "Conception de bases de données avancées : le projet IFO_2". Actes du congrès AFCET'93, Colloque "Bases de Données", pp. 105-114, Versailles, Juin 1993.

P. Poncelet. "Conception d'applications avancées : modèle, mécanismes d'évolution et dérivation". Actes du 11ième Congrès Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID'93), pp. 477-496, Lille, Mai 1993.

P. Poncelet, M. Teisseire et L. Lakhal. "IFO$_2$: modèle et principe pour la conception de bases de données avancées". Actes des 8ièmes Journées Bases de Données Avancées (BDA'92), pp. 320-338, Trégastel, Septembre 1992.

Article invité

M. Teisseire, P. Poncelet, and R. Cichetti R. "IFO$_2$: a Uniform Approach for Information System Modeling". Proceedings of the 5th International Workshop on the Deductive Approach to Information Systems and Databases (DAISD'94), pp. 33-53, Costa Brava, Spain, September 1994.

Rapports de projet

"Modélisation du comportement et contrôle de l'évolution d'une application persistante". Rapport final du projet ASP présenté par les équipes du GDR Bases de Données, pôles "Modélisation et environnement de développement" et "Gestion de la dynamique", Septembre 1997.

Autres publications liées à mon activité d'encadrement

F. Cathala. "Modélisation de scénarios comportementaux et application à la pathologie des barrages". Thèse de doctorat de l'Université d'Aix-Marseille II, Février 2000.

F. Cathala, P. Meriaux and P. Royet. "A Database Modelling Approach for Damageing Scenarios". In Dam Safety (ICOLD'98), pp. 629-637, Barcelona, Spain, Septembre 1998.

H.O. Mohamed. "Des diagrammes d'états à un formalisme de règles actives". Mémoire de DEA d'Informatique de l'Université d'Aix-Marseille II, Juin 1997.

A. Gaudillère. "Développement des algorithmes de transformation d'IFO$_2$ vers NAOS. Mise en place de l'environnement client-serveur pour le système IFO$_2$". Rapport de fin d'études d'Ecole d'Ingénieur, Août 1996.

P.E. Epaillard (IMERIR-Perpignan). "Validation, à l'aide de Lex/Yacc, du langage IFO$_2$". Rapport de fin d'études d'Ecole d'Ingénieur, Septembre 1996.

A. Meyer. "Réalisation d'un éditeur graphique IFO$_2$ en Smalltalk sous Windows". Rapport de fin d'études d'Ecole d'Ingénieur, Novembre 1995.

C. Lallouche. "Un outil d'aide à la vérification de spécifications comportementales IFO$_2$". Mémoire de DEA d'Informatique de l'Université d'Aix-Marseille II, Juin 1994.

Y. Abiza. "Etude des différentes méthodes de conception orientées objet et spécification d'un noyau de composants conceptuels adaptés au contexte des bases de données

orientées objet". Rapport de DEA de l'Université de Nice-Sophia Antipolis, Juin 1991.

Prototypes

Le système IFO_2

Le système IFO_2 vise à la réalisation d'un système complet d'aide au développement d'applications avancées dans un environnement relationnel, objet ou actif. Il s'intéresse plus particulièrement à la structuration d'objets complexes et à la spécification du comportement des applications. En fait, sont principalement abordés, à travers ce système, les thèmes suivants : la spécification, la validation et la génération de code.
IFO_2 adopte une architecture modulaire dans le but de supporter efficacement le développement d'applications avancées. L'avantage majeur de cette architecture est de fournir un ensemble d'outils intégrés (langage, éditeur, générateur de codes, etc.) permettant de dériver le schéma des applications vers divers systèmes cibles.

L'architecture retenue repose sur un serveur d'applications qui active les communications entre les différents modules. De cette manière, les spécifications issues des différentes interfaces (`ifoshell` et `ifojava`) sont exprimées sous forme de requêtes envoyées au serveur (`ifo2`) qui effectue les traitements et retourne les résultats.
Un éditeur graphique (`ifoWindows`) offre toutes les facilités pour la spécification d'un schéma IFO_2. Il produit en sortie un code interprétable par `ifoshell` de manière à offrir directement toutes les fonctionnalités d'un outil CASE.
La gestion de la persistance des applications est assurée par le module `ifoshell` qui offre un ensemble de fonctions de bas niveau permettant la manipulation de fragments (insertion, destruction, mise à jour et recherche).
L'architecture fonctionnelle d'`ifo2` se décompose en cinq composants majeurs :

- la bibliothèque des opérations de mise à jour ;
- la validation sémantique ;
- le catalogue de validation ;
- les méthodes d'aide à l'identification des états remarquables ;
- la bibliothèque de générateurs de code.

Une description complète du système est donné dans [60].

Le système CASCADE

Afin de montrer la faisabilité de l'approche de modélisation et de représentation de scénarios, un prototype a été réalisé. Il se veut un outil d'aide à l'élaboration de scénarios et propose à la fois des capacités de modélisation et de stockage des scénarios. L'outil conçu met en œuvre la démarche spécifique de modélisation ainsi que la démarche d'obtention du dictionnaire des spécifications. Le modèle de données choisi est le modèle relationnel.
L'outil développé intègre toutes les caractéristiques nécessaires à la création, à la sauvegarde et à la manipulation de scénarios.

Condition de la recherche

Initiée en 1990, au sein de l'équipe Bases de Données au laboratoire Informatique, Signaux et Systèmes (I3S) de l'Université de Nice-Sophia Antipolis, avec une allocation MRT, mon activité de recherche s'est poursuivie au Laboratoire Informatique de Marseille (LIM) de 1994 à 1996.

Le projet IFO_2 s'est déroulé, de 1990 à 1993, sous la direction de L. Lakhal, au sein de l'équipe Bases de Données du Laboratoire I3S. Durant cette période et dans le cadre de mon travail de thèse, ma contribution a porté sur la définition de la composante structurelle d'IFO_2.

Le travail sur la partie comportementale du projet IFO2, faisant l'objet de la thèse de M. Teisseire, s'est poursuivi, sous la direction de R. Cicchetti, au Laboratoire d'Informatique de Marseille (LIM) au sein de l'équipe Bases de Données dirigée par J. Le Maitre. Dans ce cadre, mon activité de recherche s'est poursuivie en participant à la proposition du modèle dynamique qui est symétrique et complémentaire de la partie structurelle ainsi que sur l'aspect dérivation vers Naos et vérification. Ces derniers travaux s'intègrent dans le cadre de notre participation dans le projet d'Action de Soutien Programmé (ASP) du GDR Bases de Données regroupant les équipes des laboratoires de recherche : CRI (Université Paris I), LAFORIA (Université P. et M. Curie, Paris), LAMSADE (Université Paris Dauphine), LSR-IMAG (Université de Grenoble), LIM (Université de la Méditerranée, Marseille II), LIRMM (Université de Montpellier II) et du LIMOS (Université de Clermont-Ferrand II).

Les travaux menés dans le cadre du projet IFO_2 ont été financé par un contrat EERP (External European Research Project) avec Digital Equipment Corporation (centre de Recherche et Développement de Digital Europe à Ferney Voltaire) dont nous étions, avec M. Teisseire, co-responsables.

Le travail sur la modélisation des scénarios s'est déroulé au LIM de 1996 à 2000. Il a été réalisé dans le cadre d'un projet (dont nous étions avec R. Cicchetti, co-responsables) avec la division d'Aix en Provence du CEMAGREF (CEntre national de Machinisme Agricole du Génie Rural des Eaux et Forêts), département Equipement pour l'eau et environnement.

Bibliographie

[1] S. Abiteboul, P.C. Fisher, and H.J. Scheck (Eds.). *Nested Relations and Complex Objects in Databases*, volume 361 of *Lecture Notes in Computer Science*. Springer Verlag, 1989.

[2] S. Abiteboul and S. Grumbach. Bases de Données et Objets Structurés. *TSI - Technique et Science Informatique*, 5:384–404, 1987.

[3] S. Abiteboul and R. Hull. Restructuring of Complex Object. In *Proceedings of the International Conference on Database Theory*, Roma, Italy, 1986.

[4] S. Abiteboul and R. Hull. IFO: A Formal Semantic Database Model. *ACM Transactions on Database Systems*, 12(4):525–565, December 1987.

[5] A. Aiken and J. Widom and. J.M. Hellerstein. Behavior of Database Production Rules: Termination, Confluence, and Observable Determinism. In *Proceedings of the ACM Sigmod Conference*, pages 59–68, 1992.

[6] A. Aiken, J.M. Hellerstein, and J. Widom. Static Analysis Techniques for Predicting the Behaviour of Active Database Rules. *ACM Transactions on Database Systems*, 20(1):3–41, March 1995.

[7] E. Anwar, L. Maugis, and S. Chakravarthy. A New Perspective on Rule Support for Object-Oriented Databases. In *Proceedings of the ACM Sigmod Conference*, June 1993.

[8] C. Arapis. Temporal Specifications of Object Behavior. In *Proceedings of the 3rd Symposium on Fondamentals of Database and Knowledge Base Systems*, volume 495 of *Lecture Notes in Computer Science*, pages 308–324, Rostock, Germany, May 1991.

[9] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S.B. Zdonik. The Object-Oriented Database System Manifesto. In *Proceedings of the 1st Deductive and Object-Oriented Databases Conference (DOOD'89)*, pages 40–55, Kyoto, Japan, December 1989.

[10] P.C. Attie, M.P. Singh, E.A. Emerson, A. Seth, and M. Rubisinkiewics. Specifying and Enforcing Intertask Dependencies. *Distributed System Engineering Journal*, 3(4):222–238, 1996.

[11] M.I. Kellner B. Curtis and J. Over. Process Modeling. *SIGMOD Record*, 35(9):75–90, 1992.

[12] C.W. Bachman. Data Structure Diagrams. *Data Base*, 1(2):4–10, 1969.

[13] J. Banerjee, H.T. Chou, J.F. Garza, W. Kim, D. Woelk, N. Ballou, and H.J. Kim. Data Model Issues for Object-Oriented Applications. *ACM Transactions on Office Information Systems*, 5(1):3–26, January 1987.

[14] E. Baralis, S. Ceri, P. Fraternali, and S. Paraboschi. A Support Environment for Active Rule Design. Technical report, Politecnico di Milano, Idea Project, January 1996.

[15] P. Bayer. State of the Art Report on Reactive Processing in Database Technology and in Artificial Intelligence. Technical Report ECRC-93-8, European Computer Industry Research Centre (ECRC), 1993.

[16] E. Benazet, H. Guehl, and M.Bouzeghoub. VITAL: A Visual Tool for Analysis of Rules Behaviour in Actives Databases. In *Proceedings of the RIDS'95 Conference*, Greece, Septembre 1995.

[17] G. Booch. *Object-Oriented Design with Applications*. Benjamin/Cumming Comp, 1991.

[18] M. Bouzeghoub. Ingénierie des Bases de Données, Techniques et Outils CASE. Habilitation à diriger des recherches de l'Université de Paris VI, Mars 1992.

[19] M. Bouzeghoub, G. Gardarin, and P. Valduriez. *OBJETS, Du C++ à Merise Object*. Eyrolles, 1994.

[20] M. Bouzeghoub and E. Métais. Semantic Modelling of Object-Oriented Databases. In *Proceedings of the 17th International Conference on Very Large Data Bases (VLDB'91)*, pages 3–14, Barcelona, Spain, September 1991.

[21] M.L. Brodie. On Modelling Behavioural Semantics of Databases. In *Proceedings of the 7th International Conference on Very Large Data Bases (VLDB'81)*, pages 32–42, Cannes, France, September 1981.

[22] M.L. Brodie. On the Development of Data Models. In *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*, pages 19–48. Springer Verlag, 1984.

[23] A.P. Buchman. Active Object Systems. In *Proceedings of the NATO-ASI Conference*, pages 1–27. Springer Verlag, August 1993.

[24] J. Carmo and A. Sernadas. A Temporal Logic Framework for a Layered Approach to Systems Specification and Verification. In *Proceedings of the Temporal Aspects in Information Systems Conference (IFIP 88)*, pages 31–46. Elsevier Science, 1988.

[25] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Conceptual Modeling of Workflows. Technical report, Politecnico di Milano, Milano, January 1996.

[26] F. Cathala. Modélisation de Scénarios Comportementaux et Application à la Pathologie des Barrages. Thèse de Doctorat d'Informatique, Université d'Aix-Marseille II, Février 2000.

[27] F. Cathala and P. Poncelet. Preserving Behaviour : Why and How. In *Proceedings of the 9th international Conference on Advanced Information Systems Engineering (CAISE'97)*, pages 333–346, Barcelona, Spain, 1997.

[28] C. Cauvet and C. Rolland. O* : un modèle pour la conception de bases de données orientées objet. In *Actes du congrès INFORSID'89*, pages 265–284, Lille, France, 1989.

[29] S. Ceri, P. Fraternali, S. Paraboschi, and L. Tanca. Active Rule Management in Chimera. Technical report, Politecnico di Milano, Idea Project, January 1996.

[30] S. Ceri and J. Widom. Deriving Production Rules for Constraint Maintenance. In *Proceedings of the 16th International Conference on Very Large Data Bases (VLDB'90)*, pages 566–577, Brisbanne, Australia, August 1990.

[31] S. Chakravarthy. Rule Management and Evaluation: An Active DBMS Perspective. *Sigmod Record*, 18(3):20–28, September 1989.

[32] S. Chakravarthy, B. Blaustein, A. P. Buchmann, M. Carey, U. Dayal, D. Goldhirsch, M. Hsu, R. Jauhari, and al. HiPAC: A Research Project in Active, Time-Constrained Database Management. Technical report, Xerox Advanced Information Technology, Cambrige, MA, August 1990.

[33] S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S.K. Kim. Composite Events for Active Databases: Semantics, Contexts and Detection. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB'94)*, pages 606–617, Santiago, Chile, September 1994.

[34] S. Chakravarthy and D. Mishra. SNOOP: An Expressive Event Specification Language for Active Databases. Technical report, University of Florida, March 1993.

[35] S. C. Cheung and J. Kramer. Enhancing Compositional Reachability Analysis with Context Constraints. In *Proceedings of the 1st International Symposium Foundations of Software Engineering*, pages 115–125, 1993.

[36] S. C. Cheung and J. Kramer. Tractable Flow Analysis for Anomaly Detection in Distributed Programs. In *Proceedings of the 4th European Software Engineering Conference*, pages 283–300, 1993.

[37] S. C. Cheung and J. Kramer. Tractable Dataflow Analysis for Distributed Systems. *IEEE Transactions on Software Engineering*, 20(8):579–593, 1994.

[38] S.C. Cheung and J. Kramer. Enhancing Compositional Reachability Analysis with Context Constraints. In *Proceedings of the 1st International Symposium Foundations of Software Engineering*, pages 115–125, September 1993.

[39] R. Cicchetti. Bases de Données Avancées : modéliser pour concevoir, structurer et faire agir. Habilitation à diriger des recherches, Université d'Aix-Marseille II, Janvier 1995.

[40] R. Cicchetti, P. Poncelet, and M. Teisseire. Une Aide à la Conception et aux Contrôle de Règles Actives. In *Actes des 12ièmes Journées Bases de Données Avancées*, pages 19–34, Cassis, France, Septembre 1996.

[41] R. Cicchetti, P. Poncelet, and M. Teisseire. Modélisation et vérifications comportementales. *Revue Ingénierie des Systèmes d'Information*, 5(3):265–286, Août 1997.

[42] CIGB. Viellissement des barrages et des ouvrages annexes - synthèse et recommandations. *Bulletin du C.I.G.B*, 1993.

[43] P. Coad and E. Yourdon. *Object-Oriented Analysis*. Yourdon Press Computing Series, 1990.

[44] E.F. Codd. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6):377–387, 1970.

[45] A. Coen-Porisini, L. Lavazza, and R. Zicari. The ESSE Project: an Overview. In *Proceedings of the 2nd Far-East Workshop on Future Database Systems*, pages 28–37, 1992.

[46] C. Collet. Bases de Données Actives : des systèmes relationnels aux systèmes à objets. Habilitation à diriger des recherches, Université Joseph Fourier, Grenoble, Octobre 1996.

[47] C. Collet, T. Coupaye, and T. Svensen. NAOS Efficient and modular reactive capabilities in an Object-Oriented Database System. In *Proceedings of the 20th Interna-*

tional Conference on Very Large Databases (VLDB'94), Santiago, Chile, September 1994.

[48] A. Cornelio and S. B. Navathe. Using Active Database Techniques for Real Time Engineering Applications. In *Proceedings of the 9th International Conference on Data Engineering (ICDE'93)*, pages 100–107, Vienna, Austria, April 1993.

[49] U. Dayal, A. P. Buchmann, and D. R. McCarthy. Rules Are Objects Too: A Knowledge Model for An Active, Object-Oriented Database System. In *Advances in Object-Oriented Database Systems*, volume 334 of *Lecture Notes in Computer Science*, pages 129–14, September 1988.

[50] U. Dayal, M. Hsu, and R. Ladin. A Transactional Model for Long-Running Activities. In *Proceeding of the 17th International Conference on Very Large Data Bases (VLDB'91)*, pages 113–122, Barcelona, Spain, September 1991.

[51] O. Deux. The Story of O_2. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):91–108, March 1990.

[52] O. Diaz and S. M. Embury. Generating Active Rules From High-Level Specifications. In *Proceedings of the 10th British National Conference on Databases (BNCOD'10)*, volume 618 of *Lecture Notes in Computer Science*, pages 227–243, Aberdeen, Scotland, July 1992.

[53] O. Diaz, N. Paton, and P. Gray. Rule Management in Object-Oriented Databases: A Uniform Approach. In *Proceedings of the 17th International Conference on Very Large Data Bases (VLDB'91)*, pages 317–326, Barcelona, Spain, September 1991.

[54] H.D. Ehrich, M. Gogola, and A. Sernadas. Objects and Their Specification. volume 655 of *Lecture Notes in Computer Science*, pages 40–65, 1991.

[55] H.D. Ehrich, A. Sernadas, and C. Sernadas. Abstract Object Types for Databases. In *Proceedings of the 2nd International Workshop on Object-Oriented Database Systems*, volume 334 of *Lecture Notes in Computer Science*, pages 144–149, Bad Munster am Stein-Ebernburg, FRG, September 1988.

[56] R. Fell. Essential Components of Risk Assesment for Dams. In *Proceedings of the Workshop on Risk-Based Dam Safety Evaluations*, Norway, Septembre 1997.

[57] J. Fiadeiro and A. Sernadas. Behavioural Aspects of Intelligent Knowledge-Based Information Systems. In *Proceedings of the Temporal Aspects in Information Systems Conference (IFIP'88)*, pages 77–92. Elsevier Science, 1988.

[58] J. Fiadeiro, C. Sernadas, T. Maibaum, and A. Sernadas. *Describing and Structuring Objects for Conceptual Schema Development*, pages 117–138. in [8], 1992.

[59] G. Gardarin and P. Valduriez. *SGBD Avancés : bases de données objets, déductives, réparties*. Eyrolles, 1990.

[60] GDR97. Modélisation du comportement et contrôle de l'évolution d'une application persistante. Rapport final du projet ASP présenté par les équipes du GDR Bases de Données, pôles "Modélisation et environnement de développement" et "Gestion de la dynamique", Septembre 1997.

[61] D. Georgakopoulos and M. Hornick. A Overview of Workflow Management : From Process Modelinbg to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.

[62] R. Hale. Using Temporal Logic for Prototyping: The Design of a Lift Controller. In *Proceedings of the Temporal Logic in Specification Conference*, volume 398 of *Lecture Notes in Computer Science*, pages 375–408, Altrincham, UK, April 1987.

[63] D. Harel. On Visual Formalisms. *Communications of the ACM*, 31(5):514–530, 1988.

[64] D. Harel and al. STATEMATE: A working environment for the development of complex reactive systems. *IEEE Transaction on Software Engineering*, 16(4):403–414, 1990.

[65] D. Harel and C.A. Kahana. On Statechart with Overlapping. *ACM Transactions on Software Engineering and Methodology*, 1(4):399–421, 1992.

[66] A. Heuer. A Data Model for Complex Objects Based on a Semantic Database Model and Nested Relations. In *Proceedings of the Nested Relations and Complex Objects in Databases Conference*, volume 361 of *Lecture Notes in Computer Science*, pages 297–311, 1989.

[67] C.A.R Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[68] D. Hollingsworth. The workflow reference model, v.1.1. Technical Report TC00-1003, Workflow Management Coalition (WfMC), http://www.wfmc.org, January 1995.

[69] R. Hull. Four Views of Complex Objects: A Sophisticate's Introduction. In *Proceedings of the Nested Relations and Complex Objects in Databases*, volume 361 of *Lecture Notes in Computer Science*, pages 87–116, 1989.

[70] R. Hull and D. Jacobs. Language Constructs for Programming Active Databases. In *Proceedings of the 17th International Conference on Very Large Data Bases (VLDB'91)*, pages 455–467, Barcelona, Spain, September 1991.

[71] R. Hull and R. King. Semantic Database Modelling: Survey, Applications and Research Issues. *ACM Computing Surveys*, 19(3):201–260, September 1987.

[72] R. Hull and C.K. Yap. The Format Model: A Theory of Database Organization. *Journal of the ACM*, 31(3):518–537, 1984.

[73] B. Jaeschke and H.J. Schek. Remarks on The Algebra of Non First Normal Form Relations. In *Proceedings of the ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 124–138, 1982.

[74] R. Junglaus, G. Saake, and C. Sernadas. Formal Specification of Object Systems. In *Proceedings of the International Joint Conference on Theory and Practice of Software Development (TAPSOFT'91)*, volume 494 of *Lecture Notes in Computer Science*, pages 60–82, 1991.

[75] A. Karadimce and S. Urban. Diagnosis Anomalous Rule Behavior in Databases with Integrity Maintenance Production Rules. In *Proceedings of the 3rd Workshop on Foundations of Models and Languages for Data and Objects*, pages 77–102, Aigen, Austria, September 1991.

[76] A. Karadimce and S. Urban. Conditional Term Rewriting as a Formal Basis for Analysis of Active Database Rules. In *Proceedings of the 4th International Workshop on Research Issues in Data Engineering, RIDE-ADS'94*, pages 156–162, Houston, Texas, February 1994.

[77] L.M. Keszenheimer. Utilizing Behavioral Abstractions to Facilitate Maintenance During Class Evolution. In *Proceedings of the 6th International Conference on Advanced Information Systems Engineering (CAiSE'94)*, Lecture Notes in Computer Science, pages 325–338, Utrecht, The NetherLands, June 1994.

[78] W. Kim, E. Bertino, and J.F. Garza. Composite Object Revisited. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 337–347, Portland, June 1989.

[79] B.S. Lerner and A.N. Haberman. Beyond Schema Evolution to Database Reorganization. *Proceedings of the ECOOP/OOPSLA'90, Sigplan Notices*, 25(10):67–76, 1990.

[80] J.Y. Lingat, P. Nobecourt, and C. Rolland. Behaviour management in database application. In *Proceedings of the 13th International Conference on Very Large Databases (VLDB'87)*, Brighton, UK, September 1987.

[81] P.C. Lockemann. *Object-Oriented Information Management.* Elsevier Science, 1989.

[82] P. Loucopoulos and R. Zicari. *Conceptual Modeling, Databases and CASE: An Integrated View of Information Systems Development.* Wiley Professional Computing, 1992.

[83] D. Maier and J. Stein. Development of an Object-Oriented DBMS. *Proceedings of the OOPSLA'86, Sigplan Notices*, 21(11):472–482, 1986.

[84] Z. Manna and A. Pnueli. Specification and Verification of Concurrent Programs by ∀-Automata. In *Proceedings of the Temporal Logic in Specification Conference*, volume 398 of *Lecture Notes in Computer Science*, pages 124–164, Altrincham, UK, April 1987.

[85] Z. Manna and A. Pnueli. The Anchored Version of the Temporal Framework. In *Proocedings of the Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency Conference*, volume 354 of *Lecture Notes in Computer Science*, pages 201–284, 1989.

[86] Z. Manna and A. Pnueli. A Hierarchy of Temporal Properties. In *Proceedings of the 9th Annual ACM Symposium on Principles of Distributed Computing*, pages 377–408, Quebec, CA, August 1990.

[87] B. Meyer. *Object-Oriented Software Construction.* Prentice-Hall, 1988.

[88] M. Missikoff and M. Scholl. An Algorithm for Insertion into a Lattice: Application to Type Classification. In *Proceedings of the Foundations of Data Organization and Algorithms Conference*, volume 367 of *Lecture Notes in Computer Science*, pages 64–82, 1989.

[89] P.A. Muller. *Modélisation objet avec UML.* Eyrolles, 1998.

[90] G.T. Nguyen and D. Rieu. Schema Evolution in Object-Oriented Database Systems. *Data & Knowledge Engineering*, 4:43–67, 1989.

[91] J. S. Ostroff and W. Murray Wonham. A Framework for Real-Time Discrete Event Control. *IEEE Transactions on Automatic Control*, 35(4):386–397, April 1990.

[92] J.S. Ostroff. Automated Verification of Timed Transition Models. In *Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 247–256, Grenoble, France, June 1989.

[93] J.S. Ostroff. Synthesis of Controllers for Real-Time Discrete Event Systems. In *Proceedings of the 28th Conference on Decision and Control*, pages 138–144, Tampa, Florida, December 1989.

[94] J.S. Ostroff. *Temporal Logic for Real Time Systems.* Wiley and Sons, 1989.

[95] J.S. Ostroff. A Logic for Real-Time Discrete Event Processes. *IEEE Control Systems Magazine*, pages 95–102, June 1990.

[96] H. Ould-Mohamed. Des Diagrammes d'Etats à un Formalisme de Règles Actives. Mémoire de DEA d'Informatique, Université d'Aix-Marseille II, Juin 1997.

[97] C. Parent and S. Spaccapietra. *ERC+: An Object-Based Entity Relationship Approach.* in [8], 1992.

[98] J. Peckman and F. Maryanski. Semantic Data Models. *ACM Computing Surveys*, 20(3):153–189, September 1988.

[99] B. Pernici. Objects With Roles. In *Proceedings of the Conference on Office Information Systems*, pages 205–215, Cambridge, MA, April 1990.

[100] P. Poncelet. Conception d'applications avancées : modèle, mécanismes d'évolution et dérivation. In *Actes du 11ième Congrès Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID'93)*, pages 477–496, Lille, France, Mai 93.

[101] P. Poncelet. Contribution à la conception d'applications avancées : modèle, mécanismes d'évolution et dérivation. Thèse de doctorat de l'Université de Nice-Sophia Antipolis, Mai 93.

[102] P. Poncelet and L. Lakhal. Consistent Structural Updates for Object-Oriented Design. In *Proceedings of the 5th International Conference on Advanced Information Systems Engineering (CAiSE'93)*, volume 685 of *Lecture Notes in Computer Science*, pages 1–21, Paris, France, June 1993.

[103] P. Poncelet and M. Teisseire. Advanced Database Modeling and Design: The IFO_2 Approach. In *Proceedings of the 3rd International Conference for Young Computer Science (ICYCS'93)*, pages 79–82, Beijing, China, July 1993.

[104] P. Poncelet, M. Teisseire, and R. Cicchetti. Behavioural Evolutions and Verifications. Technical Report LIM 95-128, June 1995.

[105] P. Poncelet, M. Teisseire, R. Cicchetti, and L. Lakhal. IFO_2, une approche pour la conception de bases de données avancées. *Ingénierie des Systèmes d'Information (ISI)*, 1(4):467–510, Décembre 1993.

[106] P. Poncelet, M. Teisseire, R. Cicchetti, and L. Lakhal. Towards a Formal Approach for Object-Oriented Database Design. In *Proceedings of the 19th International Conference on Very Large Data Bases (VLDB'93)*, pages 278–289, Dublin, Ireland, August 1993.

[107] P. Poncelet, M. Teisseire, and L. Lakhal. IFO_2, modèle et principe pour la conception de Bases de Données Avancées. In *Actes des 8ièmes Journées Bases de Données Avancées*, pages 320–338, Trégastel, France, Septembre 1992.

[108] J. Puustjarvi, H. Tirri, and J. Veijalainen. Reusability and Modularity in Transactional Workflows. *Information System*, 22(2-3):101–120, 1997.

[109] J.H Reif and A. Smolka. Dataflow Analysis of Distributed Communicating Processes. *International Journal of Parallel Programming*, 19(1):1–30, 1990.

[110] C. Rolland and C. Cauvet. Modélisation Conceptuelle Orientée Objet. In *Actes des 7ièmes Journées Bases de Données Avancées*, pages 299–325, Lyon, France, Septembre 1991.

[111] C. Rolland and C. Cauvet. *Trends and Perspectives in Conceptual Modeling.* in [8], 1992.

[112] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall, 1991.

[113] G. Saake. Descriptive Specification of Database Object Behaviour. *Data & Knowledge Engineering*, 6:47–73, 1991.

[114] H. Sakai. An Object Behavior Modeling Method. In *Proceedings of the 1st International Conference on Database and Expert System Applications (DEXA'90)*, pages 42–48, 1990.

[115] H.J. Schek and M.H. Scholl. The Relational Model with Relation-Valued Attributes. *Information Systems*, 11(2):137–147, 1986.

[116] H.J. Schek and M.H. Scholl. Evolution of Data Models. In *Database Systems for 90's*, volume 466 of *Lecture Notes in Computer Science*, pages 135–153, 1990.

[117] A. Sernadas, J. Fiadeiro, C. Sernadas, and H.D. Ehrich. Abstract Object Types: A Temporal Perspective. In *Proceedings of the Temporal Logic in Specification Conference*, volume 398 of *Lecture Notes in Computer Science*, pages 324–349, Altrincham, UK, April 1989.

[118] A. Sernadas, C. Sernadas, and H. D. Ehrich. Object-Oriented Specification of Databases: An Algebraic Approach. In *Proceedings of the 13th International Conference on Very Large Data Bases (VLDB'87)*, pages 107–116, Brighton,UK, August 1987.

[119] C. Sernadas and J. Fiadeiro. Towards Object-Oriented Conceptual Modeling. *Data & Knowledge Engineering*, 6:479–508, 1991.

[120] S. Shlaer and S.J. Mellor. *Object-Oriented Systems Analysis: Modeling the World in Data.* Yourdon Press, Prentice Hall, 1988.

[121] S. Shlaer and S.J. Mellor. *Object Lifecycles Systems Analysis: Modeling the World in State.* Yourdon Press, Prentice Hall, 1992.

[122] M. Siegel and S. Madnick. A Metadata Approach to Resolving Semantic Conflicts. In *Proceedings of the 17th International Conference on Very Large Databases (VLDB'91)*, Spain, 1991.

[123] M.P. Singh. Formal Aspects of Workflow Management-part 1: Semantics. Technical report, North Carolina State University, USA, January 1996.

[124] M.P. Singh. Formal Aspects of Workflow Management-part 2: Distributed Scheduling. Technical report, North Carolina State University, USA, January 1996.

[125] I. Sommerville. *Software Engineering.* Addison-Wesley, 1991.

[126] M. Teisseire. Le modèle IFO_2 : de la modélisation comportementale à la dérivation. Thèse de doctorat de l'Université d'Aix-Marseille II, Juin 1994.

[127] M. Teisseire. Behavioural Constraints: Why using Events instead of States. In *Proceedings of the International Conference on Object-Oriented Entity-Relationship (O-O ER'95)*, Lecture Notes in Computer Science, pages 123–132, Gold Coast, Australia, December 1995. Springer Verlag.

[128] M. Teisseire. Event Schema Updating. In *Proceedings of the International Workshop on Database and Expert System (DEXA'95)*, London, UK, September 1995.

[129] M. Teisseire, P. Poncelet, and R. Cicchetti. A Tool Based on a Formal Approach for Object-Oriented Database Modeling and Design. In *Proceedings of the 6th International Workshop on Computer-Aided (CASE'93)*, IEEE, Singapore, July 1993.

[130] M. Teisseire, P. Poncelet, and R. Cicchetti. Dynamic Modelling with Events. In *Proceedings of the 6th International Conference on Advanced Information Systems Engineering (CAiSE'94)*, volume 811 of *Lecture Notes in Computer Science*, pages 186–199, Utrecht, The NetherLands, June 1994. Springer Verlag.

[131] M. Teisseire, P. Poncelet, and R. Cicchetti. Towards Event-Driven Modelling for Database Design. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB'94)*, Santiago, Chile, September 1994.

[132] T.J. Teorey, D. Yang, and J.P. Fry. A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. *ACM Computing Surveys*, 18(2):197–222, June 1986.

[133] M. Tresh and M.H. Scholl. Meta Object Management and its Application to Database Evolution. In *Proceedings of the 11th International Conference on the Entity-Relationship Approach*, Lecture Notes in Computer Science, pages 299–321, Karlsruhe, Germany, October 1992.

[134] K. Tsuda, K. Yamamoto, M. Hirakawa, and T. Ichikawa. MORE: An Object-Oriented Data Model with a Facility for Changing Object Structures. *IEEE Transactions on Knowledge and Data Engineering*, 3(4):444–460, December 1991.

[135] A. Tuzhilin. Templar: A Knowledge-Based Language for Software Specifications Using Temporal logic. *ACM Transactions on Information Systems*, 13(3):269–304, July 1995.

[136] S. Twine. Mapping between a NIAM Conceptual Schema and KEE Frames. *Data & Knowledge Engineering*, 4(4):125–155, December 1989.

[137] WFMC. Workflow Standard - Interoperability Abstract Specification, v.1.0. Technical Report TC-1012, Workflow Management Institution (WfMC), http://www.wfmc.org, 1996.

[138] J. Widom. The Starburst Rule System: Language design, Implementation, and Applications. *IEEE Data Engineering Bulletin*, 15(4):15–18, December 1992.

[139] R.J. Wirfs-Brock and R. E. Johnson. Surveying Current Research in Object-Oriented Design. *Communications of the ACM*, 33(9):104–124, 1990.

[140] S.B. Zdonik and D. Maier (Eds.). *Reading in Object-Oriented Database Systems*. Morgan Kauffmann, 1990.

[141] R. Zicari. A Framework for Schema Updates in an Object-Oriented Database. In *Proceedings of the 7th IEEE Data Engineering Conference*, pages 2–13, Kobe, Japan, April 1991.

Deuxième partie

Fouille de données et extraction de connaissances

Introduction

Parmi les approches se situant dans une problématique générale d'aide à la décision, des démarches récentes tentent de tirer les leçons des situations ou phénomènes passés pour lesquels de nombreuses données sont collectées dans de grandes bases. Ces démarches s'inscrivent dans un processus global d'extraction de connaissances dont l'une des étapes importantes est la fouille de données (ou "Data Mining").

Bien que le terme de fouille de données recouvre à lui-seul la découverte de connaissances, il ne constitue en fait qu'une seule des étapes du processus général de l'ECD [24] (Extraction de Connaissances dans les Bases de données), qui comprend globalement trois phases :

1. *Préparation des données* : l'objectif de cette phase consiste à sélectionner uniquement les données potentiellement utiles de la base. L'ensemble des données est ensuite soumis à un pré-traitement, afin de gérer les données manquantes ou invalides (opération de nettoyage). L'étape suivante dans cette phase consiste à formater ces données, pour les rendre compréhensibles au processus de fouille de données (opérations de transformation et réduction).
2. *Extraction* : en appliquant des techniques de fouilles de données, l'objectif de cette phase est de proposer des modèles ou motifs représentatifs du contenu de la base.
3. *Interprétation des résultats* : le but de cette dernière phase est d'interpréter la connaissance extraite lors de la phase précédente, pour la rendre lisible et compréhensible par l'utilisateur.

Les techniques de fouille de données sont utilisées dans de nombreux domaines d'applications. Les exemples les plus courants sont les compagnies d'assurance, les compagnies bancaires (crédit, prédiction du marché, détection de fraudes), le marketing (comportement des consommateurs, "mailing" personnalisé), la recherche médicale (aide au diagnostic, au traitement, surveillance de la population sensible), les réseaux de communication (détection de situations alarmantes, prédiction d'incidents), l'analyse de données spatiales ou encore statistiques. Nous proposons un rapide survol en nous intéressant plus particulièrement à celles développées pour manipuler de grandes bases de données.

Les Règles d'Association

Le problème de recherche de règles d'association a fait l'objet de nombreux travaux de recherche ces dernières années [4, 6, 10, 36, 51, 64, 75, 78, 86]. Introduit dans [4] à partir du problème du panier de la ménagère ("Market Basket Problem"), il peut être résumé ainsi : étant donné une base de données de transactions (les paniers), chacune composée d'items (les produits achetés), la découverte d'associations consiste à chercher des ensembles d'items, fréquemment liés dans une même transaction, ainsi que des règles les

combinant. Un exemple d'association pourrait révéler que "75% des gens qui achètent de la bière, achètent également des couches". Ce type de règles concerne un grand champ d'applications, telles que la conception de catalogues, la promotion de ventes, le suivi d'une clientèle, etc.

Les Motifs Séquentiels

Introduit dans [7], les motifs séquentiels peuvent être vus comme une extension de la notion de règle d'association, intégrant diverses contraintes temporelles. La recherche de tels motifs consiste ainsi à extraire des ensembles d'items, couramment associés sur une période de temps bien spécifiée. En fait, cette recherche met en évidence des associations inter-transactions, contrairement à celle des règles d'association qui extrait des combinaisons intra-transaction. Dans ce contexte, et contrairement aux règles d'association, l'identification des individus ou objets est indispensable, afin de pouvoir suivre leur comportement au cours du temps. Par exemple, des motifs séquentiels peuvent montrer que "60% des gens qui achètent une télévision, achètent un magnétoscope dans les deux ans qui suivent". Ce problème, posé à l'origine par souci marketing, intéresse à présent des domaines aussi variés que les télécommunications (détection de fraudes), la finance, ou encore la médecine (identification des symptômes précédant les maladies).

Les Dépendances Fonctionnelles

L'extraction de dépendances fonctionnelles à partir de données existantes, est un problème étudié depuis de nombreuses années, mais qui a été abordé récemment avec une "vision" fouille de données. Les résultats obtenus par ces dernières approches sont très efficaces [40, 37, 49, 69]. La découverte de dépendances fonctionnelles est un outil d'aide à la décision à la fois pour l'administrateur de la base, les développeurs d'application, les concepteurs et intégrateurs de systèmes d'information. En effet, les applications relèvent de l'administration et du contrôle des bases de données, de l'optimisation de requêtes ou encore de la rétro-conception de systèmes d'information.

La Classification

La classification, appelée également induction supervisée, consiste à analyser de nouvelles données et à les affecter, en fonction de leurs caractéristiques ou attributs, à telle ou telle classe prédéfinie [9, 92]. Bien connus en apprentissage, les problèmes de classification intéressent également la communauté Base de Données qui s'appuie sur l'intuition suivante : "plus importants sont les volumes de données traités, meilleure devrait être la précision du modèle de classification" [79]. Les techniques de classification sont par exemple utilisées lors d'opérations de "mailing" pour cibler la bonne population et éviter ainsi un nombre trop important de non-réponses. De la même manière, cette démarche peut permettre de déterminer, pour une banque, si un prêt peut être accordé, en fonction de la classe d'appartenance d'un client.

La Segmentation

La technique de segmentation ressemble à celle de la classification, mais diffère dans le sens

où il n'existe pas de classes prédéfinies [39] : l'objectif est de grouper des enregistrements qui semblent similaires dans une même classe. De nombreux algorithmes efficaces ont été proposés pour optimiser les performances et la qualité des classes obtenues dans de grandes bases de données. Parmi ceux-ci nous pouvons citer [23], [27, 28] et [94]. Les applications concernées incluent notamment la segmentation de marché ou encore la segmentation démographique par exemple en identifiant des caractéristiques communes entre populations.

Les Séries Chronologiques

L'objectif est de trouver des portions de données (ou séquences) similaires à une portion de données précise, ou encore de trouver des groupes de portions similaires issues de différentes applications [3, 5, 12]. Cette technique permet par exemple d'identifier des sociétés qui présentent des séquences similaires d'expansion, ou encore de découvrir des stocks qui fluctuent de la même manière.

Après ce rapide survol des principales techniques utilisées dans le domaine de la fouille de données, nous nous intéressons, dans la suite de ce mémoire, à la recherche de motifs séquentiels. Outre la définition de composantes pour cette recherche, nous généralisons, au travers de deux types d'applications très différentes (comportement d'utilisateurs sur le Web et données complexes), l'utilisation des composants mis en place et proposons ainsi un cadre général d'extraction de connaissances.

Recherche de motifs séquentiels

Sur le plan recherche de motifs séquentiels, l'originalité de l'approche PSP proposée est de prendre en compte une nouvelle structure plus adaptée à l'extraction de motifs. Ainsi, les moyens mis en œuvre permettent d'optimiser les temps de réponses des algorithmes et donc de traiter de plus grands volumes d'information. L'approche proposée, même si elle n'a pas été définie dans ce sens, offre également à l'utilisateur une solution particulièrement efficace pour traiter le problème de la recherche de règles d'association.

En introduisant la prise en compte du temps, l'exploration peut se faire à travers des "tranches" d'histoire sans perdre trace des comportements individuels au cours du temps. Cependant, dû à leur cadre d'application particulier, les approches proposées dans la littérature n'intègrent pas un point nécessaire dans la problématique liée à certains types d'applications décisionnelles (comme par exemple celle de la gestion de scénarios décrite dans la première partie de ce mémoire) : l'occurrence réitérée d'un même événement au cours du temps. En effet, dans un scénario par exemple, la répétition d'un événement peut être significative et nous avons vu que des mécanismes d'itération doivent être offerts pour la spécification de scénarios. En considérant cette extension dans la recherche de motifs séquentiels nous en étendons donc le champ d'application.

Dans [83], des extensions ont été proposées pour répondre à la problématique de certaines applications et concernent la prise en compte de contraintes temporelles. Dans ce cadre, la proposition d'une approche de gestion efficace des contraintes, absente des autres approches, est très importante. C'est ce que permet l'algorithme GTC dont une originalité

est d'offrir une approche basée sur un pré-calcul des contraintes de temps afin d'offrir une solution exploitable par d'autres approches existantes.

L'extraction de motifs dans une grande base de données est, par définition, une approche particulièrement coûteuse en temps. Ainsi, en proposant une approche incrémentale, nous offrons une solution efficace qui permet de mettre à jour la connaissance extraite pour refléter les modifications apportées à la base de données originale. Une autre originalité de la proposition est son indépendance vis à vis des approches de recherche de motifs utilisées.

Web usage mining

En s'intéressant à l'analyse du comportement des utilisateurs sur un ou plusieurs serveurs Web, nous montrons comment les techniques de recherche de motifs séquentiels sont exploitables à la fois pour analyser le parcours des utilisateurs dans le temps mais également pour connaître les pages d'un serveur qui sont fortement corrélées, i.e. qui sont fréquemment accédées par les utilisateurs.
Les travaux menés dans le cadre des contraintes temporelles pour la recherche de motifs séquentiels offrent la possibilité de proposer une approche particulièrement originale par rapport aux travaux existants. En effet, aucune approche actuelle ne propose de gérer les navigations sur le serveur à court ou à long terme. Cette originalité est renforcée par l'intégration de la recherche incrémentale qui permet de maintenir la connaissance extraite.

L'analyse du comportement des utilisateurs ne consiste cependant pas à simplement appliquer les algorithmes de recherche de motifs séquentiels. Aussi pour prendre en compte la nature très particulière des données manipulées, nous proposons une solution complète d'ECD : des données brutes stockées sur un serveur à l'extraction de connaissances.

Recherche de régularités dans des bases de données d'objets complexes

Dans les travaux sur les règles d'association et les motifs séquentiels, les types de données manipulés sont "plats", i.e. des ensembles d'items ou des listes d'ensembles d'items. Cependant, dans de nombreuses applications les items manipulés ont une description et il est souvent plus intéressant pour l'utilisateur final de connaître les associations entre les descriptions d'items plutôt qu'entre les items eux-mêmes. Par exemple, dans un cadre de rétro-conception de règles actives, il est utile de connaître les enchaînements des règles qui s'effectuent régulièrement. Or, les techniques actuelles ne proposent pas de solution dans la mesure où les enchaînements de règles possèdent des structures complexes dans lesquelles la description des actions peut être décrite par des descriptions détaillées qui elles-mêmes peuvent être décrites par d'autres descriptions et ainsi de suite... Un autre exemple d'application est lié à l'émergence des documents semi-structurés sur le World Wide Web. En effet, pour interroger des documents, il est indispensable de connaître la structure de ceux-ci. Or, par définition, les documents semi-structurés n'ont pas de structure explicite et il faut donc proposer des approches permettant d'extraire les régularités structurelles pour soit réorganiser le document, soit générer un schéma qui pourra être utilisé par un langage de requête.

En proposant une solution à cette problématique, nous montrons que le champ d'applications des motifs séquentiels est beaucoup plus étendu que la recherche de liste d'ensemble d'items.

Organisation

Les travaux réalisés dans le cadre de la recherche de motifs séquentiels sont décrits dans le chapitre 1. La recherche du comportement d'utilisateurs sur un serveur Web est présentée dans le chapitre 2. Le chapitre 3 aborde la recherche de régularités dans des bases de données d'objets complexes.

Pour présenter les différentes contributions apportées, ces chapitres adoptent un même plan. Après un exposé détaillé de la problématique et un exposé de l'état de l'art du domaine, les contributions apportées sont détaillées et comparées aux autres travaux dans une discussion terminant chaque chapitre.

Un bilan des contributions est proposé en fin de partie. Celui-ci est suivi par une présentation de la liste des publications, des prototypes associés aux thèmes abordés ainsi que des conditions de la recherche pour les différents projets menés.

Chapitre 1

Recherche de motifs séquentiels

Motivées par des problèmes d'aide à la décision, les techniques de fouille de données se sont principalement intéressées à la problématique des règles d'association. La recherche de motifs séquentiels, outre son grand intérêt pratique (e.g. gestion des alarmes dans les réseaux de télécommunication, analyse des comportements de clients, analyse des accès dans un serveur Web, etc.) nécessite la définition de nouveaux algorithmes spécifiques.

Dans ce chapitre, nous nous proposons de détailler l'aspect extraction de motifs séquentiels. Avant de brièvement présenter les approches contribuant à la recherche de règles d'association et de motifs séquentiels, paragraphe 1.2, nous explicitons la problématique en détaillant les composantes nécessaires à une approche de fouille de données dans le paragraphe 1.1. Nous proposons ensuite une synthèse de nos travaux au paragraphe 1.3. Le bilan de la contribution est donné sous la forme d'une discussion au cours de laquelle nous précisons quelques éléments d'expérimentation.

1.1 Exposé de la problématique

La Recherche de Motifs Séquentiels

Initialement introduite dans [7], la notion de séquence est définie de la manière suivante. Une *transaction* constitue, pour un client, l'ensemble des items achetés à une même date, elle est décrite sous forme d'un triplet : <id-client, id-date, itemset>. Un *itemset* est un ensemble non vide d'items noté $(i_1 i_2 \dots i_k)$. Une *séquence* ou motif séquentiel est une liste ordonnée, non vide, d'itemsets notée $< s_1 s_2 \dots s_n >$ où s_j, $1 \leq j \leq n$, est un itemset (une séquence peut être perçue comme une suite de transactions avec une relation d'ordre entre les transactions), représentant l'ensemble des achats d'un client. Une séquence regroupant k items est dite une *k-séquence*. Soient T_1, T_2, ..., T_m les transactions d'un client, ordonnées par date d'achat croissante et soit $itemset(T_i)$, $1 \leq i \leq m$, l'ensemble des items correspondants à T_i, alors la séquence de données de ce client est $< itemset(T_1)\ itemset(T_2) \dots\ itemset(T_m) >$.

Exemple 1 Soit C un client et S=< (30) (40 50) (80) >, la séquence de données représentant les achats de ce client. S peut être interprétée par "C a acheté l'item 30, puis, de façon simultanée, les items 40 et 50 et enfin l'item 80". □

Client	Date	Items
C_1	01/01/1998	20,60
C_1	02/02/1998	20
C_1	04/02/1998	30
C_1	18/02/1998	80,90
C_2	11/01/1998	10
C_2	12/01/1998	30
C_2	29/01/1998	40,60,70
C_3	05/01/1998	30,50,70
C_3	12/02/1998	10,20
C_4	06/02/1998	20,30
C_4	07/02/1998	40,70
C_4	08/02/1998	90

FIG. 1.1 – *Base de données exemple*

Dans un contexte d'aide à la décision, le but principal est de rechercher dans une base de données de transactions les comportements les plus fréquents. Pour réaliser cette tâche, chaque sous-séquence s dans la base de données DB possède une valeur de support ($supp(s)$) indiquant son nombre d'occurrences dans DB. Cependant, dans ce contexte, une séquence de données n'est prise en compte qu'une seule fois pour calculer le support d'une séquence fréquente, i.e. elle peut présenter plusieurs fois le même comportement, le processus de recherche de séquences considère qu'elle produit ce comportement sans tenir compte du nombre de ses apparitions dans la séquence de données.
Pour décider si une séquence est fréquente ou non, une valeur de support minimal ($minSupp$) est spécifiée par l'utilisateur et une séquence est dite fréquente si la condition suivante est vérifiée : $supp(s) \geq minSupp$. En outre nous recherchons uniquement les séquences fréquentes maximales, i.e. non incluses dans une autre séquence et l'inclusion de séquence est définie de la manière suivante : Soit $s_1 =< a_1 a_2 \ldots a_n >$ et $s_2 =< b_1 b_2 \ldots b_m >$ deux séquences de données. s_1 est *incluse* dans s_2 ($s_1 \prec s_2$) si et seulement s'il existe $i_1 < i_2 < \ldots < i_n$ des entiers tels que $a_1 \subseteq b_{i_1}$, $a_2 \subseteq b_{i_2}, \ldots\ a_n \subseteq b_{i_n}$.

Exemple 2 La séquence $s1 =<$ (3) (4 5) (8) $>$ est incluse dans la séquence $s2 =<$ (7) (3 8) (9) (4 5 6)(8) $>$ (i.e. $s1 \prec s2$) car (3) $\subseteq$ (3 8), (4 5) $\subseteq$ (4 5 6) et (8) $\subseteq$ (8). En revanche $<$ (3) (5) $>\not\prec<$ (3 5) $>$ (et vice versa). □

Pour illustrer le problème de la recherche de motifs séquentiels, considérons la base de données DB de la figure 1.1. Avec un support minimum de 50% (i.e. pour qu'une séquence soit retenue, il faut que deux clients dans la base de données supportent cette séquence), les séquences fréquentes sont alors les suivantes : $<$ (20) (90) $>$, $<$ (30) (90) $>$ et $<$ (30) (40,70) $>$. Les deux premières font partie du support pour C_1 et C_4, alors que la dernière apparaît dans les séquence de données des clients C_2 et C_4.

La Prise en Compte de la Multi-Occurrence

La définition proposée pour les itemsets introduit une forte contrainte d'unicité des items. Dans [83], les auteurs précisent que "nous ne considérons pas la quantité ou les valeurs

d'items dans une transaction, celles-ci peuvent cependant être très importantes selon les applications. Déterminer de telles règles nécessite de nouveaux travaux". En effet, lors de l'analyse des caractéristiques d'une population, cette condition correspond généralement à une contrainte réelle. Cependant les séries de données comportementales contiennent fréquemment le même item, au niveau des données brutes collectées, simplement car le même type d'événement peut survenir plusieurs fois pour un même individu ou une même transaction. Cette information peut s'avérer pertinente dans certaines applications et apporter une réponse plus fine aux décideurs. Par exemple, dans le cadre de la recherche de scénarios, l'occurrence réitérée d'un même événement au cours du temps est tout à fait significative et des mécanismes d'itération doivent être offerts pour la spécification des scénarios.

Afin de pouvoir tenir compte des occurrences multiples des items dans les séries traitées, nous étendons la notion d'itemset à celle de collection d'items, qui autorise la multi-occurrence d'items et que nous appelons par la suite itemcollection. Dans ce contexte, la recherche de motifs séquentiels consiste à extraire les séquences fréquentes à partir d'une base de données de séquences de données (une séquence étant définie comme une liste de transactions, i.e. comme une liste d'itemcollections). Les définitions du support ainsi que les propriétés sous-jacentes ne sont pas ou peu modifiées lors de la nouvelle formulation du problème : le support d'une séquence d'itemcollections s correspond au nombre de séquences de données contenant s ; ainsi, même si une séquence de données contient plusieurs occurrences de s, elles ne sont comptabilisées qu'une seule fois pour le calcul du support.

L'Intégration des Contraintes Temporelles

Telle qu'elle a été introduite, la notion de séquence présente une rigidité rédhibitoire pour certains types d'application. En effet, si l'intervalle de temps entre deux transactions est suffisamment court, il peut être alors envisagée de traiter celles-ci comme une unique transaction. De façon similaire, un intervalle trop long peut rendre inintéressante l'étude de la séquence des deux transactions. C'est pourquoi la notion de *séquences généralisées* a été introduite dans [83] pour pallier ces restrictions en proposant l'introduction de contraintes temporelles. Soient *minGap* une durée minimum, *maxGap* une durée maximum et *windowSize* une durée de rectification, spécifiées par l'utilisateur. Une séquence de données $d =< d_1...d_m >$ supporte une séquence $s =< s_1...s_n >$ s'il existe des entiers $l_1 \leq u_1 < l_2 \leq u_2 < ... < l_n \leq u_n$ tels que :

1. $s_i \subset \cup_{k=l_i}^{u_i} d_k$, $1 \leq i \leq n$;
2. date(d_{u_i}) - date(d_{l_i}) $\leq windowSize$, $1 \leq i \leq n$;
3. date(d_{l_i}) - date($d_{u_{i-1}}$) $> minGap$, $2 \leq i \leq n$;
4. date(d_{u_i}) - date($d_{l_{i-1}}$) $\leq maxGap$, $2 \leq i \leq n$.

Le *support* de s, noté $supp(s)$, est le pourcentage de toutes les séquences dans DB qui supportent s. Si $supp(s) \geq minsupp$, avec une valeur de support minimum $minsupp$, la séquence s est dite *fréquente*.

Ce concept de motifs séquentiels généralisés permet une manipulation plus souple des transactions des clients dans la mesure où il est désormais possible de :

- regrouper des itemsets lorsque leurs dates sont "assez" proches via la contrainte de windowSize,

- considérer des itemsets comme trop rapprochés pour apparaître dans la même séquence fréquente avec la contrainte de minGap,
- considérer des itemsets comme trop éloignés pour apparaître dans la même séquence fréquente avec la contrainte de maxGap.

Exemple 3 Pour illustrer la prise en compte des contraintes de temps, reprenons la base de données de la figure 1.1. Avec un support minimum strictement supérieur à 50%, i.e. pour qu'une séquence soit supportée elle doit être vérifiée par au moins deux séquences de la base de données, les motifs séquentiels sont : $<$ (20) (90) $>$, $<$ (30) (90) $>$ et $<$ (30) (40,70) $>$, les deux premières car elles peuvent être retrouvées pour les deux clients C_1 et C_4, et la dernière car elle intervient pour les clients C_2 et C_4.
Si nous considérons une fenêtre de temps de 2 jours (windowSize = 2 jours), nous pouvons regrouper ensemble tous les items qui ont été achetés dans un intervalle de 2 jours même s'ils sont dans des transactions différentes. Dans ce cas, une nouvelle séquence fréquente $<$ (20 30) (90) $>$ est extraite car elle se retrouve dans la première transaction de C_4 tout en étant détectée pour C_1 (avec un couple de transactions respectant le windowSize).
Considérons maintenant, du point de vue de l'utilisateur, que deux itemsets extraits successivement ne sont pas intéressant lorsqu'ils sont séparés par un intervalle de temps de 15 jours ou plus. Cette contrainte de maxGap a pour effet d'éliminer $<$ (30) (40,70) $>$ de l'ensemble des séquences fréquentes parce que, pour le client C_2, il y a 17 jours entre les deux transactions. La séquence de données de C_2 ne satisfait pas la contrainte de maxGap et la contrainte de support minimal n'est donc plus vérifiée.
Considérons maintenant la séquence extraite via la contrainte de windowSize, i.e. $<$ (20 30) (90) $>$, son premier itemset est composé de deux items différents intervenus dans un intervalle de temps de 2 jours. Si nous considérons l'item (30), nous voyons qu'il existe un intervalle de temps de 14 jours avec (90). Cependant, le délai observé est de 16 jours entre les items (20) et (90) donc la contrainte de maxGap n'est pas vérifiée. La séquence est alors supprimée de l'ensemble des séquences fréquentes. □

La Composante Incrémentale

Etant donné que les bases de données évoluent, le problème de la maintenance des motifs séquentiels sur une longue période de temps devient essentiel puisqu'un grand nombre de nouveaux enregistrements peuvent être ajoutés à la base. Pour refléter le nouvel état courant de la base pour lequel d'anciens fréquents peuvent disparaître alors que de nouveaux peuvent apparaître, il est nécessaire de définir des algorithmes assurant la maintenance de la connaissance extraite. L'objectif de cette composante est de tirer profit de la connaissance acquise précédemment (en l'occurrence les fréquents) pour obtenir les nouveaux motifs séquentiels. En plus d'être adapté à la problématique ces algorithmes se doivent d'être efficace car l'approche proposée nécessite d'être plus rapide que l'approche naïve qui consisterait à recommencer entièrement le processus d'extraction lors des mises à jour de la base.

Clients	Itemsets		
$C1$	10 20	20	50 70
$C2$	10 20	30	40
$C3$	10 20	40	30
$C4$	60	90	

(DB)

Itemsets	
50 60 70	*80 100*
50 60	*80 90*

(db)

FIG. 1.2 – *Une base de données d'origine (DB) et une base de données incrément avec de nouvelles transactions (db)*

Clients	Itemsets		
$C1$	10 20	20	50 70
$C2$	10 20	30	40
$C3$	10 20	40	30
$C4$	60	90	
$C5$			

(DB)

Itemsets	
50 60 70	*80 100*
50 60	*80 90*
10 40	*70 80*

(db)

FIG. 1.3 – *Une base de données d'origine (DB) et une base de données incrément contenant de nouvelles transactions et de nouveaux clients (db)*

La problématique de la fouille incrémentale peut être définie de la manière suivante. Soit DB la base de données d'origine et *minSupp* le support minimal. Soit *db* la base de données ajoutée contenant de nouvelles transactions et de nouveaux clients. Soit $U = DB \bigcup db$ la base de données mise à jour contenant toutes les séquences de *db* et de DB. Soit L^{DB} l'ensemble des séquences fréquentes de DB. Le problème de la fouille incrémentale de motifs séquentiels consiste à rechercher toutes les séquences fréquentes en considérant la même valeur de support minimal.

Exemple 4 Pour illustrer la problématique de la fouille incrémentale, considérons la base de données de la figure 1.2. Considérons que le support minimal est de 50%. L'ensemble des séquences fréquentes incluses dans DB sont les suivantes : $L^{DB} = \{< (10\ 20)(30) >, < (10\ 20)(40) >\}$. Considérons à présent la base de données contenant de nouvelles transactions ajoutées à la base de données initiale (C.f. figure 1.2). En conservant la même valeur de support, les séquences $< (60)\ (90) >$ et $< (10\ 20)(50\ 70) >$ deviennent alors fréquentes. En examinant en détail la première, nous pouvons constater qu'elle n'était pas fréquente dans la base initiale (elle n'apparaissait que pour le dernier client) alors qu'elle devient fréquente car elle apparaît, avec la base ajoutée, dans les séquences des clients $C3$ et $C4$. De la même manière deux nouvelles séquences fréquentes $< (10\ 20)\ (30)\ (50\ 60)\ (80) >$ et $< (10\ 20)\ (40)\ (50\ 60)\ (80) >$ sont détectées.

Considérons à présent une base de données incrément contenant à la fois de nouveaux clients et de nouvelles transactions (C.f. figure 1.3). La valeur du support minimal restant la même, pour être fréquente, une séquence doit main-

tenant être observée dans au moins trois clients de la base. A cause de cette mise à jour du support, l'ensemble des séquences fréquentes de DB devient $L^{DB} = \{< (10\ 20) >\}$. Les nouvelles séquences fréquentes de la base de données mise à jour deviennent alors : $L^U = \{< (10\ 20)\ (50) >, < (10)\ (70) >, < (10)\ (80) >, < (40)\ (80) >, < (60) >\}$. Examinons en détail, la séquence $< (10\ 20)\ (50) >$. Celle-ci pourrait être détectée pour le client $C1$ dans la base initiale mais n'est pas une séquence fréquente. Néanmoins comme l'item 50 devient fréquent avec la base incrément, cette séquence se retrouve également chez les clients $C2$ et $C3$.

1.2 Aperçu des travaux antérieurs

Pour présenter les travaux relatifs à l'extraction de corrélations, nous nous intéressons tout d'abord à la recherche des règles d'association en détaillant les principes généraux des algorithmes d'extraction, puis en proposant une synthèse des différentes approches. Les motifs séquentiels sont abordés dans un second temps, ce qui nous permet de positionner problématique et solutions par rapport à celles des règles d'association. Une présentation complète des similarités avec les autres domaines est présentée dans [83].

1.2.1 Règles d'associations

Formellement, le problème des règles d'associations est présenté dans [4] de la façon suivante : soit $I = \{i_1, i_2, \ldots, i_m\}$ un ensemble d'items. Soit DB un ensemble de transactions, tel que chaque transaction T, dotée d'un identifiant unique noté (TID), soit constituée d'un un ensemble d'items, appelés *itemset* vérifiant $T \subseteq D$. Une transaction T supporte X, un ensemble d'items de I, s'il contient tous les items de X, i.e. si $X \subseteq T$. Une règle d'association est une implication de la forme $X \rightarrow Y$, où $X \subseteq I$, $Y \subseteq I$ et $X \cap Y = \emptyset$. La confiance c dans la règle établit la probabilité qu'une transaction T supportant l'item X, supporte aussi l'item Y. Le support d'une règle $X \rightarrow Y$ correspond au pourcentage de toutes les transactions de DB supportant $X \cup Y$.
Ainsi, à partir d'une base de transaction DB, le problème consiste à extraire toutes les règles d'association dont le support est supérieur à $minSupp$ et dont la confiance est supérieure à $minConf$ (confiance spécifiée par l'utilisateur). Cette tâche peut être divisée en deux étapes :

1. Rechercher dans un premier temps tous les itemsets fréquents. Si la base possède m items, 2^m items sont alors potentiellement fréquents, d'où le besoin de méthodes efficaces pour limiter cette recherche exponentielle.
2. Générer, à partir de ces itemsets fréquents, des règles dont la confiance est supérieure à $minConf$. Cette étape est relativement simple dans la mesure où il s'agit de conserver les règles du type $A/B \rightarrow B$ (avec $B \subset A$), ayant un taux de confiance suffisant.

La plupart des algorithmes de recherche de règles d'association utilisent les mêmes principes généraux, que nous présentons ici (C.f. figure 1.1).
Extraire toutes les règles pertinentes d'une base signifie examiner toutes les combinaisons

d'items possibles et vérifier leur fréquence et confiance par rapport aux données effectivement stockées. L'examen de toutes les combinaisons d'items revient à explorer, comme espace de recherche, l'ensemble des parties de l'ensemble des itemsets I. Les algorithmes par niveaux s'avèrent très performants pour procéder à une telle exploitation. L'idée générale est d'opérer de manière itérative en traitant, à chaque pas de l'algorithme, des itemsets de longueur croissante.

function *algoGenerique*
Input : Un support minimum *minSupp* et une base de données DB.
Output : L'ensemble L des séquences ayant une fréquence d'apparitions supérieure à *minSupp*.
$k = 1$;
// Les itemsets fréquents
$C_1 = \{\{< i >\}/i \in I\}$;
$T = C_1$;
foreach $d \in D$ **do** *compterSupport*(C_1,*minSupp*,d);
$L_1 = \{c \in C_1/Support(c) > \sigma\}$;
while $L \neq \emptyset$ **do**
  genererCandidats(T,k);
  foreach $d \in D$ **do** *compterSupport*(C_k,*minSupp*,d);
  $L_k = \{c \in C_k/Support(c) > minSupp\}$;
  $k = k + 1$;
endwhile
return $L = \bigcup_{j=0}^{k} L_j$;
end function *algoGenerique*

Algorithme 1.1: *Algorithme générique*

Ainsi la première étape concerne des itemsets simplement réduits à un singleton, la deuxième manipule des couples d'items et, de manière générale, la i^{ieme} étape opère sur des itemsets contenant i éléments (ils sont appelés i-itemsets, ou itemsets de longueur i). Supposons qu'au début de la i^{eme} étape, l'ensemble de toutes les combinaisons de i items à examiner soit fourni, il s'agit de l'ensemble des candidats. Il suffit alors, pour chaque candidat, de vérifier la fréquence effective dans la base, par une simple opération de comptage (compterSupport). Cette opération sous-entend le parcours complet de la base. Si cette dernière satisfait le support minimum, i.e. si le nombre d'occurrences du candidat considéré est supérieur au support minimum, celui-ci est considéré fréquent. Dans ce cas, il se peut qu'un sur-ensemble de cette combinaison se révèle également fréquent lors de l'étape ultérieure. Le candidat examiné est donc conservé pour être traité à l'étape suivante. A l'opposé, si l'opération de comptage conclut à la non fréquence d'un candidat, il devient inutile de calculer ses sur-ensembles : contenant un sous-ensemble non fréquent, ils sont forcément non fréquents [6]. Une telle combinaison est donc éliminée de l'espace de recherche préparé pour l'étape suivante (phase d'élagage). Ce dernier est donc réduit à l'ensemble de tous les candidats de longueur i qui sont validés, i.e. fréquents, après le parcours de la base.
Il s'agit à présent, à partir de cet ensemble, de construire toutes les combinaisons de $(i+1)$-items qu'il conviendra d'examiner au pas suivant. Cette procédure d'extension, appelée phase de génération de candidats (genererCandidat), diverge selon les approches, mais la

plupart du temps opère de manière optimale en réutilisant la connaissance acquise au cours des pas précédents. Le i^{eme} pas s'achève donc en ayant exhibé les itemsets fréquents de longueur i qui serviront, à l'itération suivante, à la génération de nouveaux candidats de longueur $(i+1)$. Nous rejoignons donc l'hypothèse posée comme point de départ de chaque pas : un espace de recherche sur-ensemble de l'espace des solutions. La démarche exposée permet de plus de restreindre au plus tôt cet espace de recherche, en éliminant dès que possible toute combinaison non pertinente de l'espace des solutions.

De nombreux algorithmes sur la recherche d'associations ont été proposés dans la littérature [4, 10, 36, 71, 48, 78, 86, 15]. [73] propose un panorama complet de ces approches. Ces algorithmes sont généralement itératifs et procèdent à plusieurs passes sur la base : c'est le cas, par exemple, de l'algorithme pionnier AIS, développé par R. Agrawal, T. Imielinski et A. Swami [4] et de l'algorithme SETM [35].
Son successeur, l'algorithme Apriori [6], malgré une procédure efficace de génération de candidats, nécessite aussi de multiples passes, mais demeure une référence dans le domaine. Quant à l'algorithme DHP [71], qui mise également sur la diminution du nombre de candidats, il requiert, comme Apriori, autant de passes sur la base que la longueur du plus grand itemset fréquent. Le nombre de passes sur la base, à l'origine des accès disque, ralentit nettement le temps de recherche et donc la production des règles d'associations : c'est sur ce constat que les travaux suivants sont basés, avec l'objectif commun de minimiser le nombre de passes.

L'algorithme DIC [10] divise la base en plusieurs parties égales sur lesquelles un comptage dynamique de candidats de longueurs différentes est opéré, lors d'une même passe : il contribue ainsi à une réduction sensible du nombre de passes sur la base (moins de k passes pour déterminer des itemsets fréquents de taille k). [34] propose un nouvel algorithme en ligne, effectuant seulement deux passes sur la base : nécessitant moins de mémoire que DIC et Apriori, il est en partie basé sur un libre changement du support lors de la première passe sur les transactions. L'algorithme Partition [78] réussit à baisser fortement le taux d'accès disque en parcourant seulement deux fois la base : celle-ci est divisée en plusieurs parties, dont la taille permet une manipulation en mémoire. Dans une première passe, on génère l'ensemble de tous les itemsets fréquents possibles et dans la seconde, leur support global est compté. l'algorithme Partition [78] propose de diviser la base de données afin de déterminer les fréquents en seulement deux passes. Pour cela l'algorithme détermine un nombre de partitions de tailles identiques et met en œuvre une méthode similaire à Apriori pour calculer en mémoire les fréquents locaux à chaque partition. Notons L^i l'ensemble des fréquents de la partition i. Du fait de la différence entre les données stockées dans chaque partition, les fréquents qu'elles contiennent sont également différents. Pour déterminer les fréquents de manière globale, une autre passe sur la base (*counting phase*) est nécessaire. Pour créer les candidats qu'il faut tester dans cette phase, l'ensemble des *candidats-globaux* $C^G = \bigcup_i L^i$ est construit comme l'union de tous les fréquents locaux. L'algorithme DLG [15] utilise, pour sa part, une structure de données différente : chaque item est représenté par un champ de bits, qui spécifie dans quels itemsets il apparaît. La génération d'itemsets fréquents s'effectue alors via des opérations logiques sur ces vecteurs de bits. Dans le même contexte, les auteurs de [26] utilise des tables binaires hiérarchiques, qui permettent de diminuer le coût des opérations sur les ensembles d'itemsets. Un autre moyen de minimiser les coûts d'entrées/sorties consiste à travailler sur un petit échantillon

de la base plutôt que sur la totalité : [86] présente un algorithme utilisant la technique d'échantillonnage (ou "sampling") et qui assure de trouver toutes les règles, au maximum en deux passes. Enfin dans [99, 97] un algorithme efficace, combinant l'utilisation de vecteurs de bits (représentation TID-List) et le partitionnement, est proposé : en comptant le pré-traitement, seules deux passes sur la base sont effectuées, la production des résultats est ainsi nettement accélérée.

Les approches précédentes ont en commun, à part DIC qui a mis en avant cette problématique, le fait qu'aucune d'entre elles ne tienne compte d'une éventuelle corrélation des données. En effet dans [74, 73, 85], l'algorithme Close proposé offre des temps de réponse particulièrement performants sur des bases de données telles que les données de recensement. Si cet algorithme, basé sur l'utilisation de treillis de Galois, n'accélère pas les temps de réponse sur les benchmarks proposé par [6], il a l'avantage de garder les meilleures performances sur un type de données particulier : les données fortement corrélées. L'étude de ce type de données a également donné lieu aux approches FP-growth et CLOSET qui ont la particularité d'éviter de générer des candidats [88, 5].

Pour pallier l'augmentation constante du volume de données à traiter et accélérer les temps de réponses, de nombreux algorithmes ont été adaptés aux techniques du parallélisme. Ces algorithmes [71, 98, 8, 16, 30] impliquent néanmoins plusieurs passes sur la partition locale de la base (coûts d'accès disque important) et échangent à chaque itération des informations ou des données relatives aux partitions distantes, occasionnant des surcoûts non négligeables en communication et synchronisation. Plus récemment, les auteurs de [100] ont proposé des algorithmes partitionnant la base en plusieurs sections totalement indépendantes, afin d'éliminer les besoins de synchronisations : le travail est partagé entre les processeurs de façon à ce que chacun recherche séparément les itemsets fréquents ; la base étant parcourue seulement deux fois, les coûts d'accès disques sont ainsi réduits.
De nombreux autres travaux généralisant la recherche de règles d'association ont été menés. Citons l'introduction de taxonomie, i.e. la découverte d'associations entre catégories d'items [31, 81], l'ajout de contraintes sur des items ou combinaisons d'items [84, 68], ou encore la recherche de règles d'association quantitatives [11, 42]. Pour cette dernière, il s'agit de trouver des règles à partir d'une base contenant à la fois des données quantitatives et qualitatives. Les données qualitatives correspondent à des catégories ou critères caractérisant la population étudiée.
Enfin, face aux grandes quantités de règles produites, dont beaucoup sont insignifiantes ou redondantes, de nombreuses approches [10, 75] cherchent à ne conserver que les règles pertinentes et à les classer en fonction de mesures d'intérêt.

1.2.2 Motifs séquentiels

Le problème de recherche de motifs séquentiels a été introduit par R. Agrawal et R. Srikant dans [7]. Ils y présentent trois algorithmes, dont deux prévus pour trouver uniquement les motifs séquentiels maximaux. Le troisième algorithme, AprioriAll, est conçu pour trouver tous les motifs et leur support ; ses performances s'avèrent de plus comparables, voire meilleures, aux deux autres approches. Dans des travaux ultérieurs [83], les mêmes auteurs étendent le problème afin de manipuler des contraintes de temps (*minGap*, *maxGap*, *windowSize*) et la taxinomie. Ils proposent ainsi l'algorithme GSP (Generalized Sequential

Patterns), dont les performances sont bien meilleures qu'AprioriAll.
La recherche de similarités dans une base de données génétique, présenté dans [87, 77], aborde une problématique assez similaire. Cependant les motifs recherchés sont des sous-séquences composées de caractères consécutifs et séparés d'un nombre variable de caractères parasites. Or dans notre cas, une séquence consiste en une liste d'ensembles de caractères (les items), plutôt qu'une simple liste de caractères. De plus, nous sommes intéressés par la recherche de toutes les séquences de support minimum et non seulement de quelques motifs fréquents.
Un problème de découverte d'épisodes fréquents dans des séquences d'événements a été proposé dans [52]. Un épisode est un ensemble d'événements, sur lequel est défini un ordre partiel. Notre définition de séquence peut s'exprimer sous la forme d'épisode. Toutefois, leur objectif est de trouver des épisodes fréquents dans une seule longue séquence d'événements, alors que nous sommes intéressés par la découverte des séquences fréquentes à travers de nombreuses transactions différentes. Les algorithmes MEDD et MSDD [70] permettent de découvrir des motifs dans des séquences d'événements multiples, mais sont limités cependant par des séquences de longueur deux.
La découverte de séquences peut aussi s'apparenter à la recherche de règles d'associations [4]. L'ensemble de toutes les séquences fréquentes constitue en fait un sur-ensemble de l'ensemble des itemsets fréquents. L'utilisation de ces algorithmes pour la recherche de motifs séquentiels a montré que les temps de réponses étaient inacceptables [83]. Cependant les algorithmes de recherche de séquences tels que AprioriAll ou GSP, retiennent certains principes, notamment l'approche générique, initialement proposée pour la découverte des règles d'association.
Les algorithmes présentés dans [99] s'inspirent aussi fortement des techniques d'extraction des règles d'associations. Cependant, l'espace de recherche des séquences, plus complexe, impose de nouveaux algorithmes spécifiques : par exemple l'approche proposée par M.J Zaki dans [97] est basée sur le partionnement mais aussi sur de simples opérations d'intersection. L'efficacité de son approche tient au fait que seules trois passes sur la base sont nécessaires, contrairement aux autres démarches qui nécessitent souvent autant de passes que la longueur de la plus grande séquence fréquente. Malheureusement, cette approche ne s'intéresse qu'à la recherche de motifs séquentiels et ne prend pas en compte les motifs séquentiels généralisés.

Il existe très peu de travaux concernant la prise en compte des mises à jour de la base de données dans le cadre des motifs séquentiels. Dans [72], les auteurs proposent un algorithme de recherche incrémentale basé sur l'approche SPADE [15], qui met à jour les motifs séquentiels d'une base de données lorsque de nouvelles transactions ou de nouveaux clients sont ajoutés. L'algorithme est basé sur un treillis de séquence constitué de toutes les séquences fréquentes et de toutes les séquences de la bordure négative de la base de d'origine. Cette bordure négative correspond à toutes les séquences qui ne sont pas fréquentes mais dont toutes les sous-séquences sont fréquentes. En outre, le support de toutes les séquences et sous-séquences est conservé dans le treillis. L'idée principale de l'algorithme est, lors d'une mise à jour, de parcourir la base incrément une première fois pour rajouter les informations dans le treillis. Ces nouvelles données sont alors combinées avec les séquences fréquentes et la bordure négative afin de déterminer quelle partie de la base d'origine doit être reparcourue. Même si cette approche est efficace (elle ne nécessite que peu de parcours de la base), la maintenance de la bordure négative est très difficile à gérer en mémoire et

pour cela, les auteurs précisent que leur approche n'est utilisable que pour de petites bases.

L'algorithme GSP étant défini pour la problématique des motifs séquentiels avec contraintes de temps, nous nous attachons à en décrire les principes généraux dans le paragraphe suivant.

L'algorithme GSP - Generalized Sequential Patterns

Pour construire les séquences candidates et les séquences fréquentes, l'algorithme GSP [83] procède selon les étapes précisées dans l'algorithme générique. Il s'agit, tout d'abord, de calculer le support de chaque item de la base. A la fin du parcours de celle-ci, les items fréquents, i.e. vérifiant la contrainte de support minimum, sont découverts. Ils sont considérés comme des 1-séquences fréquentes (séquence composée d'un itemset réduit à un singleton). Cet ensemble initial est le point de départ de la deuxième étape. L'ensemble des 2-séquences candidates est alors construit en combinant tous les couples d'items fréquents, à la fois dans la même transaction et dans une transaction séparée. A partir de ce point, toute étape k ($k > 2$) réalise, en se basant sur l'ensemble des (k-1)-séquences fréquentes, les deux sous-étapes suivantes :

- La première sous-étape (*join phase*) concerne la génération des candidats. L'idée principale est de retrouver parmi toutes les séquences dans L_{k-1}, des couples de séquences (s, s') tels qu'en éliminant le premier item de s et le dernier item de s', nous retrouvions les deux mêmes sous-séquences. Quand un couple de séquence (s, s') vérifie la condition, un nouveau candidat est construit en ajoutant le dernier item de s' à s. Dans la séquence candidate, ajoutée à C_k, le découpage des transactions est respecté.
- La seconde sous-étape (*prune phase*) a pour but d'obtenir l'ensemble L_k des k-séquences fréquentes en éliminant de C_k les séquences ne vérifiant pas le support minimum. Il s'agit donc de compter le nombre de séquences candidates intervenant dans les séquences de données.

Les séquences candidates sont stockées dans une structure de *hachage* afin d'assurer un accès efficace. Plus précisément, les séquences candidates sont stockées dans les feuilles de l'arbre alors que les nœuds intermédiaires contiennent des tables de hachage pour optimiser la recherche. Chaque séquence de données d de la base de données est hachée pour trouver les candidats contenus dans d. En parcourant une séquence de données, les contraintes de temps doivent être également prises en compte. Ceci est réalisé en parcourant l'arbre de "haut en bas" et de "bas en haut" afin d'appliquer une première fois les contraintes de temps. Le parcours de l'arbre permet d'obtenir des ensembles de feuilles contenant des candidats potentiels. A partir de ces feuilles, GSP recherche tous les candidats effectivement présents dans la séquence de données et qui vérifient les contraintes de temps. Pour cela, chaque candidat est stocké dans une nouvelle structure et est comparé avec la séquence de données. La comparaison a lieu en parcourant en avant et en arrière la séquence candidate et la séquence de données. La phase "avant" est réalisée en examinant les items progressivement. Pendant cette phase, la condition de minGap est appliquée pour éliminer les itemsets qui sont trop proches de leur précédent. Tout en parcourant les itemsets, windowSize est appliqué pour redéfinir les limites des transactions. La phase "retour arrière" est nécessaire lorsque la contrainte de maxGap n'est plus vérifiée. Dans ce cas, il est nécessaire d'éliminer tous les itemsets pour lesquels maxGap n'est pas vérifié et de recommencer un parcours avant à partir du dernier item qui vérifiait la condition de maxGap. Toutes les séquences

fréquentes sont alors stockées dans une nouvelle structure de hachage multi-niveau [10] pour permettre d'optimiser la génération des candidats.

Exemple 5 La figure 1.4 illustre la façon dont GSP gère la structure de hachage. Les deux arbres servent à conserver les mêmes candidats mais les feuilles de l'arbre du haut peuvent contenir plus de deux séquences candidates alors que dans l'arbre de droite seulement deux séquences candidates peuvent être stockées dans la même feuille. Nous remarquons que la feuille qui porte l'étiquette 20 dans l'arbre de gauche (le deuxième fils de la racine) est utilisée pour stocker les séquences candidates < (20 30 40) > et < (20) (30 40) >. Quand GSP atteint cette feuille, il n'a aucun moyen de savoir si c'est la séquence < (20 30 40) > ou bien la séquence < (20) (30 40) > qui l'a conduit jusqu'à cette feuille. C'est pour cette raison que GSP doit tester les séquences candidates contenues dans les feuilles atteintes afin de savoir quel(s) support(s) incrémenter. □

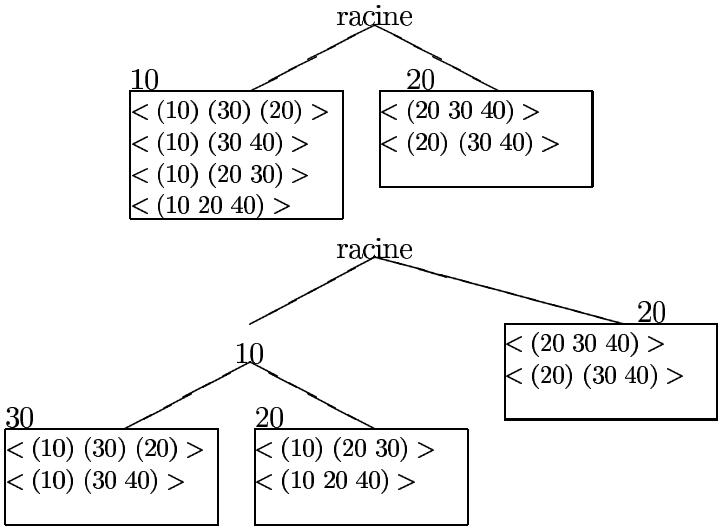

FIG. 1.4 – *La structure d'arbre de hachage de GSP*

1.3 Synthèse des contributions

En proposant une structure adaptée à la recherche de motifs séquentiels, nous montrons que la recherche peut être considérablement optimisée par rapport à GSP.

Basée sur la structure proposée, nous montrons qu'elle peut facilement être adaptée à la recherche de séquence intégrant la gestion de l'occurrence multiple d'items. La gestion de la multi-occurrence accentue encore l'avantage d'un point de vue performance de notre proposition par rapport à la version étendue de GSP.

Pour garantir une meilleure gestion des contraintes de temps, nous proposons une approche basée sur un pré-calcul des contraintes temporelles dans le processus d'extraction

de connaissances.

Enfin, pour compléter l'approche proposée nous adjoignons aux possibilités d'extraction, une composante incrémentale qui offre la possibilité d'obtenir les séquences fréquentes lors de mises à jour de la base de données initiale.

1.3.1 Composante Recherche de Motifs Séquentiels

La structure de données utilisée par GSP est mal adaptée à la recherche de séquences avec contraintes de temps. En effet, coûteuse en mémoire, elle nécessite de fréquents accès disque lors de la manipulation de grandes bases. En outre, le manque d'efficacité d'un parcours (aussi optimisé soit-il) utilisant cette structure est évident dans la mesure où pour prendre en compte les contraintes de temps, il est nécessaire d'effectuer des parcours arrière coûteux. Enfin le fait de stocker les séquences candidates dans des feuilles nécessite un parcours particulier pour chaque feuille avec de fréquents retours arrière et la mise en place d'une nouvelle structure.
Nous définissons donc une nouvelle structure, PSP (Prefix-tree for Sequential Patterns), destinée à accélérer les processus de vérification des séquences candidates. Les algorithmes associés à la manipulation de la structure et les expérimentations réalisées sont décrites dans [54, 55, 53].

Une nouvelle organisation des séquences candidates permet de trouver plus efficacement l'ensemble des candidats inclus dans une séquence de données. En effet, la structure de hachage utilisée dans GSP consiste à mettre en commun des préfixes sans faire de distinction entre deux items du même itemset et deux items avec changement d'itemsets. De ce fait, l'algorithme se voit contraint, lorsqu'il extrait une séquence de la feuille, de tester si les dates des items qu'elle représente sont adéquates par rapport à la séquence de données testée.
Par contre, la structure de données utilisée par PSP est une structure d'*arbre préfixé*, proche de celle utilisée dans [64]. Elle factorise les séquences candidates selon leur préfixe et tient compte des éventuels changements d'itemsets. Nous avons montré dans [53] que la structure d'arbre préfixé était plus efficace en mémoire que celle de hachage utilisée par GSP. En outre, la structure utilisée offre deux avantages principaux :

1. les séquences fréquentes sont stockées dans le même arbre que les séquences candidates afin d'optimiser la génération des candidats ;
2. contrairement à GSP qui nécessite une phase supplémentaire pour gérer les contraintes de temps, celles-ci peuvent être prises en compte directement lors du parcours de l'arbre.

De manière générale, à la k^{ieme} étape, l'arbre est d'une profondeur k. Il stocke toutes les k-séquences candidates de la manière suivante : chaque branche, de la racine à une feuille correspond à une séquence candidate et en considérant une simple branche, chaque nœud à la profondeur l ($k \geq l$) correspond au l^{ieme} item de la séquence. En outre, le support de la séquence de la racine à une feuille est associé à chaque nœud terminal. Pour distinguer les itemsets à l'intérieur d'une séquence (par exemple, (10 20) et (10)(20)) les fils d'un nœud sont séparés en deux catégories : "*same-transaction*" ou "*other-transaction*" (branche en traits pleins dans la figure 1.5).

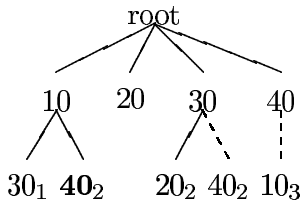

FIG. 1.5 – *La structure d'arbre préfixé de PSP*

Exemple 6 Considérons l'ensemble de 2-séquences fréquentes suivant : $L_2 = \{< (10)\ (30) >, < (10)\ (40) >, < (30)\ (20) >, < (30\ 40) >, < (40\ 10) >\}$. Il est organisé dans la structure d'arbre préfixé comme illustré figure 1.5. Chaque nœud terminal contient un item et un compteur. Si nous considérons le nœud correspondant à l'item **40**, la valeur 2 associée indique que deux occurrences de la séquence $\{< (10)\ (40) >\}$ ont déjà été détectées. L'item 20, fils de la racine, est un fréquent de longueur 1. □

1.3.2 Composante Multi-Occurrence

GSP généralise le problème défini dans [81] en incorporant des contraintes de temps dans la recherche des motifs séquentiels. Les contraintes de temps et plus particulièrement la contrainte du *Window Size*, engendre la création de transactions contenant plusieurs occurrences d'un même item. Chaque transaction étant estampillée d'une date, l'application de Window Size consiste à réunir les transactions considérées proches dans le temps pour former une et une seule transaction. Par exemple, si Window Size provoque la jointure de deux itemsets i_1 et i_2, respectivement estampillés par t_1 et t_2, l'itemset résultant i' sera composé de tous les items de i_1 datés au temps t_1, et tous les items de i_2 datés au temps t_2. Supposons que l'item a apparaisse dans i_1 et i_2, la contrainte d'unicité oblige à supprimer l'un des deux items (a^{t_1} ou a^{t_2}). Les auteurs ne parlent pas dans leurs travaux de cette configuration-là. Notre approche permet en revanche de pallier ce problème, en conservant ces deux items "similaires" (seule la date les différenciant) dans la nouvelle transaction i'.

Vouloir intégrer une composante multi-occurrence dans GSP revient à redéfinir la phase de génération des candidats. En effet, les autres composantes telles que celles dans le parcours de l'arbre ne sont pas modifiées dans la mesure où elles se basent uniquement sur les séquences candidates stockées dans la structure de hachage.

A travers l'exemple des figures 1.6 et 1.7, nous illustrons les modifications à apporter à GSP lors de la phase de génération des candidats, pour prendre en compte la contrainte de multi-occurrence (les nouvelles séquences produites sont inscrites en gras). Nous décrivons plus particulièrement le passage des 1-séquences fréquentes L_1, aux séquences candidates C_2, puis de L_2 à C_3.

1. de L_1 à C_2 : soit deux itemsets $< (1) >$ et $< (2) >$ de L_1, ils sont combinés pour former les 2-séquences candidates suivantes : $< (1)(1) >$, $< (1\ 2) >$, $< (1)(2) >$, $< (2)(1) >$ et $< (2)(2) >$. La base autorisant les multi-occurrences, il s'agit alors de produire des listes d'itemcollections. Pour cela, l'auto-jointure de L_1 est appliquée

1-séquences fréquentes	2-Séquences Candidates	
	jointure	élagage
< (1) >	**<(1 1)>**	
< (2) >	<(1) (1)>	
< (4) >	<(1 2)>	idem
	<(1) (2)>	
	<(1 4)>	
	<(2 2)>	
	...	

FIG. 1.6 – *Exemple de génération de candidats de taille 2 avec GSP (multi-occurrence)*

2-séquences fréquentes	3-Séquences Candidates	
	jointure	élagage
< (1)(1) >	<(1) (1) (1)>	<(1) (1) (1)>
< (13) >	<(1) (1 3)>	<(1) (1) (4)>
< (1)(4) >	<(1) (1) (4)>	**<(2 2 2)>**
<(2 2)>	**<(2 2 2)>**	<(2 2) (4)>
< (2)(4) >	<(2 2) (4)>	

FIG. 1.7 – *Exemple de génération de candidats de taille 3 avec GSP (multi-occurrence)*

avec lui-même pour permettre la génération de deux séquences supplémentaires : $< (1\ 1) >$ et $< (2\ 2) >$.

2. de L_2 à C_3 : de la même façon, pour générer des listes d'itemcollections, l'auto-jointure de L_2 avec lui-même est autorisée. Cependant, étant donné qu'il faut avoir (k-1)-itemsets en commun, seules les séquences composées d'un seul et unique item ont la possibilité de générer de telles séquences (ex : avec $< (2\ 2) >$, nous générons la 3-séquence candidate $< (2\ 2\ 2) >$). Dans ce cas, il est inutile d'exécuter la deuxième étape de l'algorithme (phase d'élagage), car les sous-séquences contiguës sont naturellement fréquentes. Ces modifications s'appliquent ensuite de manière identique aux étapes suivantes.

Client	Date	Items
C1	01/12/99	10, 10
C1	02/12/99	10, 40
C1	05/12/99	30
C2	02/12/99	10, 20, 20, 30
C2	10/12/99	10, 10
C3	01/12/99	10, 30
C3	03/12/99	30, 30
C3	10/12/99	30, 40

FIG. 1.8 – *Base de données exemple*

De la même manière que précédemment, nous proposons une description illustrée de l'inté-

gration de la multi-occurrence dans l'algorithme PSP. L'approche PSP+ ainsi que l'intégration de la multi-occurrence dans différents algorithmes de recherche de règles d'association sont définis dans [13]. Nous considérons pour cela la base de données illustrée par la figure 1.8, avec un support minimum de 50%.

Nous décrivons ci-dessous les deux premières itérations de l'algorithme, à travers les différentes étapes de l'approche : de la construction des candidats, au parcours des séquences dans l'arbre des candidats pour déterminer leur support.

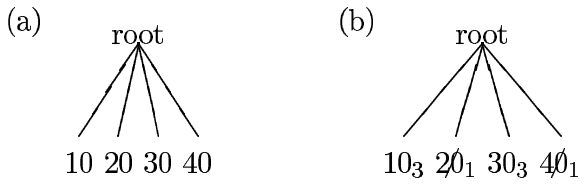

FIG. 1.9 – *1-séquences candidates puis fréquentes*

La figure 1.9(a) représente l'ensemble des 1-séquences candidates, chaque item de la base des clients constitue en fait une feuille de l'arbre. Leur support (en indice à droite des items sur la figure) est compté lors d'une passe sur la base et ceux dont le support est inférieur à 2 sont éliminés (c'est le cas des items 20 et 40, figure 1.9(b)).
A partir de ces 1-séquences fréquentes, l'algorithme de génération des candidats construit l'ensemble des 2-séquences candidates. La modification de cet algorithme afin de tenir compte de la multi-occurrence permet de générer les deux candidats suivants : (10 10) et (30 30) (C.f. figure 1.10(a)).

Pour déterminer quelles séquences candidates sont fréquentes, on parcourt l'arbre des candidats pour chaque séquence (ou client) de la base, selon une procédure récursive. Lorsqu'une feuille est atteinte, le candidat, représenté par le chemin de la racine à la feuille, est inclus dans la séquence considérée, son support est alors incrémenté.
Nous illustrons ci-dessous le parcours de la séquence du client $C1$ dans l'arbre des 2-séquences candidates. Nous considérons la représentation de $C1$ suivante :

$$C1 =< (10_i\ 10_j)\ (10_k\ 40)\ (30) >$$

Le premier item dans $C1$ (i.e. l'item 10_i) constitue le point de départ de la recherche dans l'arbre. La descente s'effectue alors dans la branche de gauche (item 10) et les séquences constituées par ce dernier et tous ses fils sont alors testées : si la séquence est incluse dans $C1$, son support s est incrémenté. Ces séquences sont les suivantes : $(10_i\ 10_j)$, $(10_i)(10_k)$, $(10_i)(30)$.
La recherche continue avec le prochain item de la séquence $C1$ (l'item 10_j). Identique à l'item précédent (i.e. même valeur d'item dans la même transaction), il n'est donc pas pris en compte : les 2-séquences considérées à partir de 10_j seront en effet similaires à celles considérées avec l'item 10_i et le support d'un candidat n'est incrémenté qu'une seule fois par client.
Avec l'item 10_k, les séquences suivantes sont considérées : $(10_k\ 40)$ (la séquence n'est pas trouvée dans l'arbre) et (10_k) (30) (le support a déjà été compté pour ce client).
L'item 40 n'étant pas fréquent, aucune séquence n'est testée à partir de celui-ci.

Le dernier item de la séquence (l'item 30) ne peut plus constituer de séquences candidates (la longueur devant être égale à deux), la recherche s'arrête alors pour le client $C1$, mais continue de manière identique pour les autres clients ($C2$ puis $C3$).

L'arbre des 2-séquences fréquentes est illustré par la figure 1.10(b). Seules les séquences de la branche gauche, considérées fréquentes car de support supérieur à deux, sont conservées et serviront lors de la prochaine itération, à la construction des candidats de longueur trois. Ces séquences sont les suivantes : < (10 10) >, < (10) (10) >, < (10 30) > et < (10) (30) >.

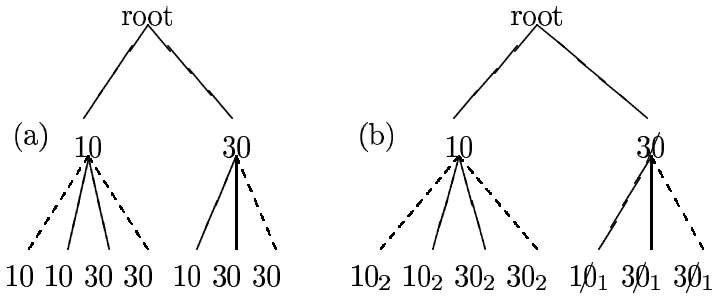

FIG. 1.10 – *Arbre des 2-séquences candidates et fréquentes*

1.3.3 Composante Intégration des contraintes temporelles

Si la structure utilisée dans PSP est bien adaptée à la recherche de séquences, l'algorithme PSP mis en œuvre peut être amélioré dans le cas de la gestion des contraintes de temps. Considérons, par exemple, la séquence de données suivante : < (1) (2) (3) (4) (5) > extraite de la base représentée par la figure 1.11.

Avec windowSize=1 et minGap=1, l'algorithme va tester les combinaisons suivantes afin de vérifier si elles sont susceptibles d'incrémenter le support d'une séquence candidate :

sans contraintes	avec windowSize & minGap
< (1) > •	< (1) (2) (5) > •
...	...
< (5) > •	< (1) (3) (5) > •
...	...
< (1) (2)> •	< (1) (2 3) (5) > •
...	...
< (3) (4)> •	< (1) (2) (4) (5)> •
...	< (1) (3) (4) (5)> •
< (1) (2) (3) (4) (5) > •	< (1) (2 3) (4) (5)> ∘

Nous remarquons que les séquences marquées par un • sont incluses dans la séquence marquée par un ∘. En effet si une séquence candidate fait partie du support pour < (1) (2 3) (4) (5) > alors elle fait également partie du support pour < (1) (2) (4) (5) > et < (1) (3) (4) (5) >, ce qui rend le test de ces deux séquences inutile car elles sont incluses dans une autre plus générale. Lié au nombre de combinaisons possibles, le nombre de séquences incluses est encore plus important en prenant en compte la contrainte

de windowSize.

Client	Date	Item
C1	01/04/99	1
C1	03/04/99	2
C1	04/04/99	3
C1	07/04/99	4
C1	17/04/99	5

FIG. 1.11 – *Exemple de base de données*

Le composante contrainte de temps propose, via l'algorithme GTC, de résoudre le problème des séquences incluses à l'aide d'une gestion efficace des contraintes de temps. L'algorithme, les preuves et les expériences sont présentés dans [53, 58].

Théorème :

L'algorithme GTC construit exactement toutes les séquences non incluses pour le problème des contraintes de temps.

L'objectif de l'algorithme est de gérer les contraintes de temps, non pas lors du parcours d'une séquence de données dans l'arbre des candidats comme dans l'approche PSP mais à l'aide d'un pré-calcul sur les séquences de données.

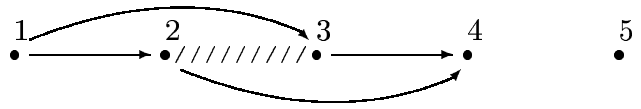

FIG. 1.12 – *Visualisation d'une séquence de données*

Reprenons le problème de la contrainte de minGap et maxGap sans tenir compte, pour l'instant, de windowSize.
Considérons la représentation graphique d'une séquence de données (C.f. figure 1.12) correspondant à la base de la figure 1.11. Le séparateur //////// entre deux items a et b indique que a et b sont "trop" proches au sens de la contrainte de minGap. Avec une contrainte de maxGap égale à 9, la séquence de données est séparée en deux sous-séquences sur lesquelles seront effectués les calculs : (1, 2, 3, 4) et (5).
Si nous observons uniquement le sous-graphe limité aux sommets (1, 2, 3, 4), nous pouvons extraire deux séquences qu'il est utile de chercher dans l'arbre des séquences candidates : < (1) (2) (4) > et < (1) (3) (4) >, qui correspondent aux deux chemins de source "1" du graphe.

Pour trouver l'ensemble complet des séquences non incluses, l'algorithme GTC utilise la structure de données suivante : un graphe orienté dont chaque sommet représente un itemset

et possède un vecteur *isPrec* de taille $|S|$ indiquant pour ce sommet quel autre sommet le précède par un chemin quelconque (il s'agit d'un vecteur de booléens qui indique au sommet x, pour chaque sommet y, si y est prédécesseur de x ou non). L'algorithme de construction du graphe explore chaque sommet x pour lui appliquer une méthode qui se décompose en deux étapes :

1. **Phase de propagation :** l'algorithme cherche le premier sommet y qui vérifie ($y.date() - x.date() > minGap$) [1] et construit l'arc (x,y). Ensuite pour chaque sommet z tel que ($z.date() - y.date() > minGap$) l'algorithme met à jour $z.isPrec[x]$ pour conserver le fait que x atteindra z quoiqu'il arrive en passant par y. Soit Z l'ensemble des sommets z ainsi traités.
2. **Phase de la main tendue :** le but dans cette phase est d'utiliser l'information "x atteindra $z(z \in Z)$" pour construire les arcs (x,t) sachant que ($t.date() - x.date() > minGap$) et $t.isPrec[x] \neq 1$ (donc $t \notin Z$ et x n'atteindra pas t si l'on ne construit pas l'arc (x,t)).

Pour prendre en compte la contrainte windowSize nous étendons l'algorithme GTC en calculant les combinaisons cohérentes de windowSize en début d'algorithme puis, une fois le graphe respectant minGap calculé, l'algorithme détecte alors les inclusions.
Pour énumérer les combinaisons de windowSize possibles, l'algorithme parcourt chaque sommet x, et détermine pour chacun d'entre eux quels autres sommets peuvent être "fusionnés" avec x (x et y peuvent fusionner si l'on peut grouper les itemsets qu'ils représentent, i.e. si $y.date() - x.date() < windowSize$). Nous adaptons la structure de données utilisée pour prendre en considération cette fusion. Désormais chaque sommet représente un itemset et possède un vecteur *isPrev* de taille $|S|$ indiquant pour ce sommet quel autre sommet le précède par un chemin quelconque et un itemset possède une date de début (date de transaction du premier item) et une date de fin (date de transaction du dernier item).
A partir du graphe respectant minGap et maxGap, l'algorithme détecte les inclusions de la manière suivante : pour chaque sommet x, déterminer l'ensemble de ses successeurs : $x.next$. Pour chaque sommet y dans $x.next$, si $y \subset z$,$z \in x.next$ et $y.next \subseteq z.next$ alors retirer le sommet y du graphe.

Une fois l'algorithme GTC appliqué à une sous-séquence de données d, l'ensemble des séquences à tester dans l'arbre des séquences candidates se construit en parcourant le graphe.

Exemple 7 Considérons la base de données représentée par la figure 1.13, le graphe résultant de la première étape est représenté par la figure 1.14. En effet, windowSize étant fixé à 4, les items 3, 4 et 5 peuvent être fusionnés. Cependant, il ne faut pas négliger la possibilité que < (3) (4 5) > ou encore < (3 4) (5) > peuvent faire partie d'une séquence candidate et donc doivent être testées. C'est pourquoi l'algorithme construit les sommets qui représentent les itemsets (3 4), (4 5) et (3 4 5).
le graphe résultant de la seconde étape est illustré par la figure 1.14. Il s'agit du graphe après l'exécution de GTC pour la contrainte de minGap. L'adaptation pour prendre en compte les possibilités ajoutées par windowSize se fait donc

1. *date*() est une fonction permettant d'accéder à la date de transaction de l'item ou de l'itemset.

Client	Date	Item
C1	01/04/99	1
C1	07/04/99	2
C1	13/04/99	3
C1	17/04/99	4
C1	18/04/99	5
C1	24/04/99	6

FIG. 1.13 – *Exemple de base de données*

FIG. 1.14 – *Graphe de séquence après la première et la seconde étape*

de manière immédiate.
La méthode utilisée pour détecter l'inclusion est illustrée figure 1.15. Nous constatons que les séquences < (3) (4) (6) > et < (3) (5) (6) > sont incluses dans la séquence < (3) (4 5) (6) >. En effet 3.*next*={(4),(5),(4 5)}, 4.*next* = 5.*next* = (4 5).*next* = 6 et comme 4 ⊂ (4 5) et 5 ⊂ (4 5) les sommets 4 et 5 peuvent être supprimés. En revanche, 2.*next* = {(3),(3 4),(3 4 5)} mais 3.*next* ⊄ (3 4 5).*next* donc le sommet 3 n'est pas supprimé.
Le graphe final utilisé pour la vérification des candidats est illustré par la figure 1.16. □

1.3.4 Composante Incrémentale

L'algorithme ISE (Incremental Sequence Extraction) résout le problème de la recherche incrémentale de séquences en utilisant les informations trouvées lors d'une extraction précédente. Considérons que k soit la longueur de la plus grande séquence trouvée lors d'une recherche précédente. Nous décomposons le problème de la manière suivante :

1. Rechercher toutes les nouvelles séquences de taille $j \leq (k+1)$. Le but de cette phase

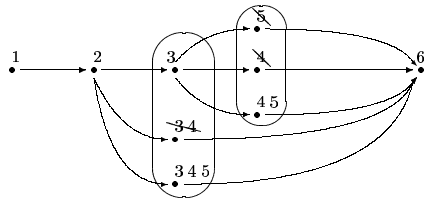

FIG. 1.15 – *Méthode de détection d'inclusions*

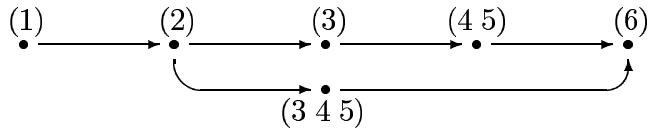

FIG. 1.16 – *Un graphe de séquence calculé avec windowSize*

Cust-Id	Itemsets		
$C1$	10 20	20	50 70
$C2$	10 20	30	40
$C3$	10 20	40	30
$C4$	60	90	

(DB)

Itemsets	
50 60 70	*80 100*
50 60	*80 90*

(*db*)

FIG. 1.17 – *Une base de données d'origine (DB) et une base de données incrément avec de nouvelles transactions (db)*

est de rechercher des séquences de DB qui deviennent fréquentes ; les séquences qui sont fréquentes dans l'incrément ; les extensions des fréquents de DB avec des items fréquents de *db*.

2. Rechercher toutes les séquences fréquentes de taille $j > (k+1)$.

Le second problème peut être résolu facilement en utilisant un algorithme comme PSP ou GSP dans la mesure où nous disposons, à la fin de la première phase, de toutes les séquences fréquentes de taille $(k+1)$. Dans la suite de ce paragraphe, nous nous intéressons au premier problème.

Pour découvrir les séquences fréquentes de taille $j \leq (k+1)$, l'algorithme ISE fonctionne de manière itérative.

Première étape

Dans la première passe sur *db*, nous comptons le support des items et nous obtenons ainsi l'ensemble *1-candExt* contenant les items qui interviennent au moins une fois dans *db*. En comparant ces items avec ceux de DB, nous obtenons l'ensemble L_1^{db} qui contient les items de *db* qui sont fréquents dans U. A la fin de cette phase, nous supprimons de L^{DB}, les séquences fréquentes qui ne vérifient plus le support.

Exemple 8 Considérons la base de données incrément de la figure 1.17. En parcourant *db*, nous obtenons le support de chaque item : {(< (50) >, 2), (< (60) >, 2), (< (70) >, 1), (< (80) >, 2), (< (90) >, 1), (< (100) >, 1)}. Considérons maintenant que nous avons obtenu les items suivants lors d'une extraction précédente sur DB.

item	10	20	30	40	50	60	70	90
support	3	3	2	2	1	1	1	1

En combinant ces items avec ceux de *db*, nous obtenons l'ensemble des 1-séquences fréquentes sur U et qui sont contenues dans *db* : $L_1^{db} = \{< (50) >, < (60) >, < (70) >, < (80) >, < (90) >\}$. □

Seconde étape - Génération des 2-candidats dans l'incrément
Les 1-séquences fréquentes de *db* sont utilisées pour générer de nouvelles 2-séquences candidates. Lors d'un nouveau parcours sur *db*, nous obtenons l'ensemble *2-candExt* composé des 2-séquences incluses au moins une fois dans *db*. Un parcours sur U avec les éléments de *2-candExt* permet de trouver les 2-séquences fréquentes qui sont insérées dans *2-freqExt*.

Exemple 9 Considérons l'ensemble L_1^{db} de l'exemple précédent. A partir de cet ensemble, nous pouvons générer les séquences suivantes $<$ (50 60) $>$, $<$ (50) (60) $>$, $<$ (50 70) $>$, $<$ (50) (70) $>$, ..., $<$ (80) (90) $>$. Pour découvrir 2-*candExt* dans la base de données mise à jour, nous examinons si un item intervient au moins une fois. par exemple, puisque le candidat $<$ (50) (60) $>$ n'apparaît pas dans *db*, il n'est pas considéré lors du parcours de U. A la fin du parcours de U avec les candidats restants, nous obtenons l'ensemble suivant des 2-séquences fréquentes 2-$freqExt = \{<$ (50 60) $>$, $<$ (50) (80) $>$, $<$ (50 70) $>$, $<$ (60) (80) $>\}$. □

Seconde étape - Recherche des itemsets fréquents précédents
Une opération supplémentaire, pour rechercher dans DB les sous-séquences fréquentes de L^{DB} précédants les items de *db*, est réalisée sur les items découverts fréquents dans *db*. Pour rechercher efficacement ces sous-séquences fréquentes nous utilisons un tableau qui a autant d'éléments que le nombre d'items fréquents dans *db*. En parcourant les séquences de données de U, pour chaque sous-séquence, nous examinons si elle est incluse. Dans ce cas, le support de chaque sous séquence précédent l'item est incrémenté.

Lors de la passe pour déterminer *2-freqExt*, nous obtenons aussi l'ensemble de toutes les sous-séquences précédants les items de *db*. A partir de cet ensemble, en ajoutant les items de *db* aux sous-séquences fréquentes, nous obtenons un nouvel ensemble $freqSeed$ contenant des nouvelles séquences fréquentes dont la taille est inférieure à $k+1$.

Items	Sous-séquences Fréquentes
50	$<$ (10) $>_3$ $<$ (20) $>_3$ $<$ (30) $>_2$ $<$ (40) $>_2$ $<$ (10) (30) $>_2$ $<$ (10) (40) $>_2$ $<$ (20) (30) $>_2$ $<$ (20) (40) $>_2$ $<$ (10 20) $>_3$ $<$ (10 20) (30) $>_2$ $<$ (10 20) (40) $>_2$
60	$<$ (10) $>_2$ $<$ (20) $>_2$ $<$ (30) $>_2$ $<$ (40) $>_2$ $<$ (10) (30) $>_2$ $<$ (10) (40) $>_2$ $<$ (20) (30) $>_2$ $<$ (20) (40) $>_2$ $<$ (10 20) $>_2$ $<$ (10 20) (30) $>_2$ $<$ (10 20) (40) $>_2$
70	$<$ (10) $>_2$ $<$ (20) $>_2$ $<$ (10 20) $>_2$
80	$<$ (10) $>_2$ $<$ (20) $>_2$ $<$ (30) $>_2$ $<$ (40) $>_2$ $<$ (10) (30) $>_2$ $<$ (10) (40) $>_2$ $<$ (20) (30) $>_2$ $<$ (20) (40) $>_2$ $<$ (10 20) $>_2$ $<$ (10 20) (30) $>_2$ $<$ (10 20) (40) $>_2$
90	-

FIG. 1.18 – *Sous-séquences fréquentes intervenant avant les items de db*

Exemple 10 Considérons l'item 50 dans L_1^{db}. Pour le client C_1, 50 est précédé par les sous-séquences fréquentes suivantes : $<$ (10) $>$, $<$ (20) $>$ et $<$ (10 20) $>$.

Si nous considérons maintenant le client C_2 avec la transaction mise à jour, nous obtenons l'ensemble de sous-séquences fréquentes précédants 50 suivant : $<(10)>$, $<(20)>$, $<(30)>$, $<(40)>$, $<(10\ 20)>$, $<(10)\ (30)>$, $<(10)\ (40)>$, $<(20)\ (30)>$, $<(20)\ (40)>$, $<(10\ 20)\ (30)>$ et $<(10\ 20)\ (40)>$. Ce principe est répété jusqu'à ce que toutes les transactions soient examinées. La figure 1.18 illustre les sous-séquences fréquentes ainsi que leurs supports sur U. L'ensemble $freqSeed$ est obtenu en ajoutant à chaque item de L_1^{db} sa sous-séquence fréquente associée. Par exemple, en considérant l'item 70, les sous-séquences suivantes sont insérées dans $freqSeed$: $<(10)\ (\mathbf{70})>$, $<(20)\ (\mathbf{70})>$ et $<(10\ 20)\ (\mathbf{70})>$. □

A la fin de la première passe sur U, nous disposons donc des 2-séquences fréquentes (dans *2-freqExt*) et d'un ensemble de séquences fréquentes dont la taille est inférieure ou égale à $k+1$ (dans *freqSeed*). Dans les itérations suivantes, nous sommes donc amenés à rechercher les séquences fréquentes qui ne sont pas encore dans ces deux ensembles.

Passes suivantes
Examinons maintenant les passes suivantes en considérant que nous sommes à la j^{ieme} passe avec $j \leq k+1$. Nous commençons par générer de nouveaux candidats à partir des ensembles obtenus précédemment. L'idée principale est de retrouver parmi les séquences de $freqSeed$ et de j-$freqExt$, deux séquences ($s \in freqSeed$, $s' \in j\text{-}freqExt$) tels qu'un item $i \in L_1^{db}$ soit le dernier item de s et le premier item de s'. Dès que la condition est vérifiée pour un couple (s, s'), une nouvelle séquence candidate est créée en supprimant le dernier item de s et en lui ajoutant s'. De manière complémentaire, nous générons à partir de j-$freqExt$, de nouvelles $(j+1)$-séquences candidates en utilisant la même génération que GSP. Le support de tous les candidats est obtenu en parcourant la base U et nous obtenons respectivement $freqInc$ et $(j+1)$-$freqExt$ qui sont utilisés pour générer de nouveaux candidats. Le processus s'arrête lorsque toutes les séquences fréquentes ont été découvertes ou que $j = k+1$.

A la fin de cette phase, nous obtenons à partir de L^{DB} et des séquences maximales de $freqSeed \bigcup freqInc \bigcup freqExt$, l'ensemble $L^{U_{k+1}}$ contenant toutes les séquences ayant une taille inférieure ou égale à $k+1$.

Exemple 11 Reprenons notre exemple, nous savons que $k = 3$, i.e. la plus grande taille des séquences fréquentes dans L^{DB}. Nous pouvons donc générer à partir de $freqExt$ la nouvelle séquence candidate $<(50\ 60)\ (80)>$ puisque sa taille est inférieure à k.
Examinons à présent comment sont générées les nouvelles séquences à partir de $freqSeed$ et 2-$freqExt$. En considérant la séquence $s =<(20)\ (40)\ (50)>$ de $freqSeed$ et $s' =<(50\ 60)>$ de 2-$freqExt$, la nouvelle séquence candidate $<(20)\ (40)\ (50\ 60)>$ est obtenue en supprimant 50 de s et en s' à la séquence restante.
Les séquences maximales fréquentes telles que $j \leq (k+1)$ sont précisées dans la figure 1.19. □

Une fois que toutes les séquences de taille $j \leq (k+1)$ sont découvertes, nous recherchons les nouvelles j-séquences fréquentes dans U avec $j > k+1$. Pour cela, nous récupérons des trois

freqInc	*freqSeed*	*freqExt*
< (10) (50 60) (80) >	< (10 20) (30) (50) >	< (60) (90) >
< (20) (50 60) (80) >	< (10 20) (40) (50) >	
< (30) (50 60) (80) >	< (10 20) (30) (60) >	
< (40) (50 60) (80) >	< (10 20) (40) (60) >	
< (20) (30) (50 60) >	< (10 20) (30) (80) >	
< (20) (40) (50 60) >	< (10 20) (40) (80) >	
< (10 20) (50 60) >		
< (10) (30) (50 60) >		
< (10) (40) (50 60) >		
< (10) (30) (50 60) >		
< (10) (30) (50) (80) >		
< (10) (40) (50) (80) >		
< (20) (30) (50) (80) >		
< (20) (40) (50) (80) >		
< (10 20) (50) (80) >		
< (10 20) (50 70) >		

FIG. 1.19 – *Séquences fréquentes maximales telles que* $j \leq (k+1)$

ensembles précédents ($freqSeed$, $freqExt$ et $freqInc$) les (k+1)-séquences fréquentes. De nouvelles (k+2)-séquences candidates sont alors générées en utilisant une approche similaire à GSP et le processus continue jusqu'à ce qu'il n'y ait plus de candidats à générer. En éliminant les séquences non maximales, i.e. les séquences incluses, nous obtenons L^U, l'ensemble de toutes les séquences fréquentes dans la nouvelle base de données mise à jour.

Exemple 12 A partir des (k+1)-séquences fréquentes découvertes dans l'exemple précédent, nous pouvons générer les séquences candidates suivantes : < (10 20) (30) (50 60) >, < (10 20) (40) (50 60) >, < (20) (30) (50 60) (80) > et < (20) (40) (50 60) (80) >. Comme elles sont fréquentes sur U, elles sont utilisées pour générer de nouveaux candidats à l'étape suivante. Nous obtenons donc à la fin du processus, i.e. dès qu'il n'est plus possible de générer des candidats, les deux séquences fréquentes suivantes : < (10 20) (30) (50 60) (80) > et < (10 20) (40) (50 60) (80) >. □

1.4 Discussion

L'algorithme GSP ne peut pas faire preuve d'une grande efficacité du fait de la nature même de la structure mise en œuvre. En effet l'approche PSP présentée dans ce chapitre montre qu'une structure efficace permet de retrouver plus rapidement une séquence dans un ensemble de séquences candidates. Pour mieux illustrer l'avantage d'une telle structure, considérons l'exemple suivant :

Exemple 13 La figure 1.20 illustre les structures utilisées par notre approche (arbre de gauche) et par l'approche GSP (arbre de droite). Dans les structures proposées, nous stockons l'ensemble des 3-candidats suivants : < (10) (10 40) >

$<$ (10) (30) (20) $>$ $<$ (10) (30 40) $>$ $<$ (30 40 10) $>$ $<$ (40) (10 30) $>$ $<$ (40) (1040) $>$. Nous pouvons constater que la structure proposée nécessite moins d'espace mémoire que celle de GSP. □

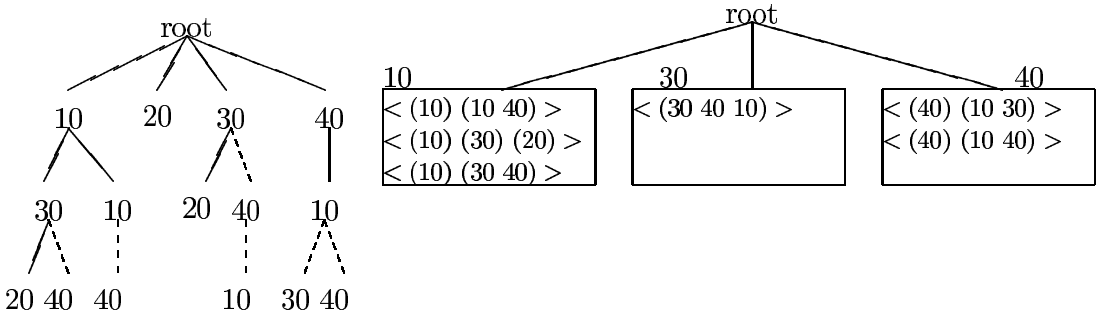

FIG. 1.20 – *Les deux structures mises en jeu.*

Pour évaluer quels candidats sont inclus dans une séquence de données , GSP parcourt l'arborescence jusqu'à atteindre une feuille puis applique un algorithme mettant en œuvre un retour arrière coûteux sur toutes les séquences candidates contenues dans cette feuille en parcourant chaque séquence candidate dans son intégralité. Notre approche permet de se limiter à la recherche des feuilles sans plus de calcul, quand une feuille est atteinte la seule opération restante est d'incrémenter son support.
Les séquences contenues dans les feuilles de la structure de [7] sont groupées dans des feuilles quand elles ont un préfixe commun mais GSP ne profite pas de cette organisation car il teste chaque séquence candidate de la feuille du premier au dernier item. Notre approche utilise le fait que toutes les feuilles "sœurs" ont un préfixe commun qu'il n'est pas nécessaire de tester pour chacune d'entre elles. Cela peut même se généraliser à l'argument suivant : quand le nœud N est atteint, le chemin parcouru de la racine vers ce nœud n'est plus à re-parcourir pour atteindre son frère.
Les éléments abordés lors de la discussion se sont également vérifiés lors des expérimentations avec PSP et GSP : PSP permet d'obtenir l'ensemble des fréquents 8 à 10 fois plus rapidement que l'approche GSP.

En étendant la structure PSP à la prise en compte de la multi-occurrence, nous avons, pour pouvoir comparer les deux approches, défini un algorithme GSP+, basé sur l'algorithme GSP mais intégrant la possibilité d'avoir plusieurs événements semblables dans un itemset. Les conclusions tirées sur la structure mise en jeux au travers de PSP se retrouvent dans le cas de la multi-occurrence. Pour nous en convaincre, considérons le parcours d'une séquence de données s, à travers un chemin de l'arbre des candidats. Les séquences s et c sont respectivement définies par $s =<$ (30)(30)(40 30) $>$ et $c =<$ (30 30 40)(30) $>$.
Dans la version multi-occurrence de GSP, la séquence s descend successivement dans les nœuds d'étiquettes 30, 30 et 40. Dans la feuille atteinte une seule séquence candidate est stockée et est comparée à la séquence s : comme c n'est pas incluse dans s, ce candidat n'est pas retenu et son support n'est pas incrémenté. Il a ainsi fallu parcourir trois niveaux dans l'arbre ainsi qu'une comparaison de deux 4-séquences pour s'apercevoir que c n'était pas

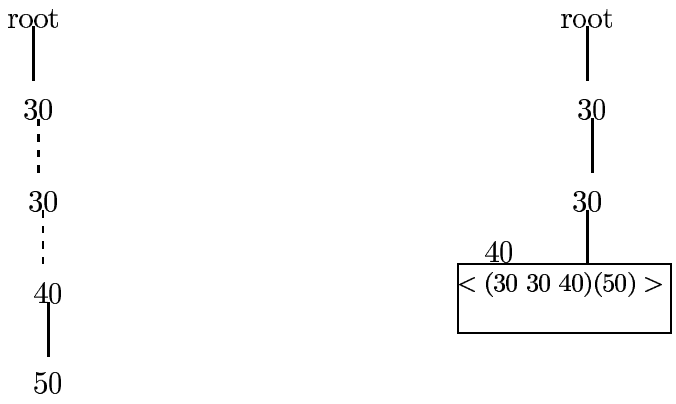

FIG. 1.21 – *Les deux structures mises en jeu.*

candidate pour la séquence s alors que dès le second item l'inclusion n'était pas possible. Les expérimentations menées avec PSP+ et GSP+ ont montré que notre approche était toujours plus efficace et que la différence entre les deux pouvait être d'un facteur 20.

En proposant un pré-calcul efficace des contraintes temporelles, nous généralisons la problématique des motifs séquentiels en offrant une approche suffisamment générique pour être indépendante des algorithmes de recherche utilisés. En effet, à la fin du prétraitement, nous obtenons l'ensemble des séquences candidates maximales vérifiant les contraintes de temps. A partir de cet ensemble, le parcours dans l'arbre des candidats (quelle que soit la structure envisagée : bitmap, arbre de hachage, TID-list, etc.) est immédiat et ne nécessite plus de retour arrière.

Lors des expérimentations avec GTC, PSP et GSP, nous avons constaté que l'algorithme de construction du graphe est un investissement rentable pour des séquences candidates de grande taille. Il est, par exemple, immédiat de constater que le temps de construction du graphe doit être inférieur au temps de parcours de PSP diminué du temps de parcours de GTC. Or, le temps nécessaire pour parcourir l'arbre des séquences candidates avec une séquence de données dans PSP croît de façon bien plus rapide que le temps nécessaire pour parcourir l'arbre des séquences candidates avec un graphe de séquence, alors que le temps de construction du graphe de séquence, lui, reste constant. Ces différences de temps requises sont, de plus, proportionnelles à l'importance des contraintes de temps spécifiées par l'utilisateur. Nous avons pourtant constaté que pour des contraintes de temps assez faible (windowSize=1 ou bien minGap=0) ou pour des fréquents relativement courts, la seconde passe sur la base de données (destinée à tester les candidats de taille 2) restait encore plus efficace avec l'algorithme PSP. Malgré cela, GTC affiche des temps de réponse plus performants que PSP dans de tels cas de figure (grâce à une plus grande rapidité d'exécution sur les passes suivantes). Pour exploiter ces comportements, nous avons crée un troisième algorithme (PG-*Hybrid*), hybride, utilisant la technique de PSP pour la seconde passe et ensuite GTC sur les passes suivantes. Cette technique s'avère efficace pour des fréquents courts, alors que pour des fréquents de longue taille la différence de temps devient minime en proportion du temps total requis.

L'approche incrémentale proposée se veut également générique dans la mesure où elle est indépendante des algorithmes de recherche de motifs utilisés. Les seuls éléments indispensables à la mise à jour sont les motifs séquentiels obtenus lors d'une première étape d'extraction. Les différentes expérimentations menées ont montré qu'ISE était nettement plus performant que l'utilisation de la méthode naïve qui consiste à ré-effectuer une extraction sur la nouvelle base de données. La différence est telle que nous avons mené d'autres expériences en utilisant ISE, non plus pour une recherche incrémentale, mais pour rechercher directement des motifs séquentiels dans une base en découpant celle-ci en une base d'origine et en un incrément. Les expériences ont montré que selon la taille de la base incrément ISE pouvait être deux fois plus rapide que GSP.

Quelques mots sur les expérimentations

Pour évaluer les différents algorithmes nous nous sommes attachés à les développer en C++ avec STL en utilisant la même philosophie et les mêmes structures. Toutes les optimisations ont été reportées sur les différentes approches. Concernant GSP, les optimisations précisées dans la thèse de R. Srikant ont été prises en compte mais il est évident que malgré celles-ci les résultats obtenus dans [83] ont bénéficié d'autres types d'optimisation (non précisées dans la littérature). Les approches proposées ont été implémentées en C++ (avec STL) et les expériences ont été réalisées sur un PC, 450 MHz Pentium III, 64 Mo de mémoire, système d'exploitation Linux et les données résidaient sur un disque dur de 9Go (IDE).
Deux types de jeux de données ont été utilisés : des données synthétiques obtenues à partir du générateur d'IBM et qui simulent le comportement des usagers ("panier de la ménagère")[2] et des jeux de données réelles. Ces derniers sont de types très différents puisqu'il s'agit de fichiers access log du LIRMM et de l'IUT d'Aix en Provence, de fichiers textes (au format pdf)[3], des requêtes en langages naturels et de requêtes XML sur le WEB. Les expérimentations concernant l'approche PSP se trouvent dans [54, 53, 55]. L'intégration de la multi-occurrence et les évaluations réalisées sont décrites dans [13]. Une comparaison détaillée des expérimentations entre PSP et GTC est proposée dans [58]. Enfin, dans [59], nous décrivons en détail les expérimentations menées dans le cadre de la recherche incrémentale.

2. Le programme de génération est disponible à l'URL suivante : *http://www.almaden.ibm.com/cs/quest.*
3. Ces fichiers ont été obtenus à partir du moteur de recherche suivant : http://searchpdf.adobe.com.

Chapitre 2

Web usage mining

Avec la popularité du World Wide Web, de très grandes quantités de données comme l'adresse des utilisateurs ou les URL demandées sont automatiquement récupérées par les serveurs Web et stockées dans des fichiers access log. L'analyse de tels fichiers peut offrir des informations très utiles pour, par exemple, améliorer les performances, restructurer un site ou même cibler le comportement des clients dans le cadre du commerce électronique. La découverte d'informations à partir du Web est généralement appelée Web Mining et peut recouvrir deux aspects : *Web content Mining* et *Web usage Mining* [19]. La première approche concerne la découverte et l'organisation d'informations extraites du Web. Par exemple, des approches basées sur les agents sont utilisées pour découvrir et organiser, de manière autonome, les informations extraites à partir du Web [47, 41, 63, 76] et des approches Bases de Données s'intéressent aux techniques d'intégration, d'organisation et d'interrogation de données hétérogènes et semi-structurées sur le Web [2, 61, 14, 25]. Le *Web usage Mining* s'intéresse par contre au problème de la recherche de motifs comportementaux des utilisateurs à partir d'un ou plusieurs serveur Web afin d'extraire des relations entre les données stockées.

Ce chapitre est organisé de la façon suivante. Dans le paragraphe 2.1, nous décrivons plus particulièrement la problématique associée au Web Usage Mining. Nous abordons, dans le paragraphe 2.2, les différentes approches contribuant à la recherche de comportement d'utilisateurs sur des serveurs Web. Une synthèse de nos travaux est proposée dans le paragraphe 2.3. Le bilan de la contribution est donné sous la forme d'une discussion (paragraphe 2.4).

2.1 Exposé de la problématique

Un Système d'Extraction de Connaissances

Dans notre contexte, les données brutes sont collectées dans des fichiers access log des serveurs Web. Une entrée dans le fichier log est automatiquement ajoutée chaque fois qu'une requête pour une ressource atteint le serveur Web (*démon http*). Spécifiée par le CERN et la NCSA [18], une entrée contient des enregistrements formés de 7 champs séparés par des espaces [67] :

```
host user authuser [date:time] ‘‘request’’ status bytes
```

Le tableau de la figure 2.1 décrit chaque champ et son contenu.

Champ	Contenu
`host`	Le nom ou l'adresse IP du client.
`user`	Toute information retournée par *identd* pour cet utilisateur, ou "-" par défaut.
`authuser`	l'identificateur utilisateur si celui-ci l'a envoyé. sinon "-".
`date`	La date dans le format JJ/Mois/Année.
`time`	L'heure dans le format hh:mm:ss.
`request`	La première ligne de la requête HTTP faite par le client (par exemple, PUT ou GET suivi par le nom de l'URL demandée).
`status`	Le code renvoyé par le serveur en réponse à cette requête, ou "-" par défaut.
`bytes`	Le nombre total d'octets envoyés (sans compter l'entête HTTP), ou "-" par défaut.

FIG. 2.1 – *Description des champs d'une entrée*

La figure 2.1 illustre un extrait du fichier access log du serveur Web de l'IUT d'Aix en Provence.

```
132.208.12.150 - - [29/Nov/1998:18:02:26 +0200] "GET /info/COINETUD.gif HTTP/1.0 " 200 1159
132.208.12.150 - - [29/Nov/1998:18:02:27 +0200] "GET /info/CONTACTER.gif HTTP/1. 0" 200 1137
132.208.12.150 - - [29/Nov/1998:18:02:28 +0200] "GET /info/DEBOUCHES.gif HTTP/1. 0" 200 1150
132.208.12.150 - - [29/Nov/1998:18:02:30 +0200] "GET /info/RECRUT.gif HTTP/1.0" 200 1141
132.208.12.150 - - [29/Oct/1998:18:03:07 +0200] "GET /info/recrut.html HTTP/1.0" 200 1051
132.208.127.200 - - [16/Oct/1998:20:34:32 +0100] "GET /geaaix/home.html HTTP/1.0 " 200 14617
148.241.148.34 - - [31/Oct/1998:01:17:40 +0200] "GET /info/index.html HTTP/1.0" 304 -
148.241.148.34 - - [31/Oct/1998:01:17:42 +0200] "GET /info/recrut.html HTTP/1.0" 304 -
192.70.76.73 - - [22/Nov/1998:11:06:11 +0200] "GET /info/program.html HTTP/1.0" 200 4280
192.70.76.73 - - [22/Nov/1998:11:06:12 +0200] "GET /info/MATIERES.gif HTTP/1.0" 200 2002
192.93.19.14 - - [07/Dec/1998:11:44:15 +0200] "GET /queldept.html HTTP/1.0" 200 5003
```

FIG. 2.2 – *Exemple de fichier access log*

Bien que des outils d'analyse [38] existent pour indiquer, par exemple, le nombre d'accès à des URL ou la liste des URL les plus demandées, les rapports existant entre les ressources demandées et les accès des clients ne sont pas analysés par de tels outils dont les performances sont assez limitées [95, 20].

Dans notre contexte, en analysant les informations stockées sur le serveur Web, un exemple de règle d'association peut être : *50 % des clients qui ont visité les URL* `plaquette/info-f.html` *et* `labo/infos.html` *ont également visité* `situation.html` ou bien *85% des clients qui ont visité les URL* `iut/general.html` `departement/info.html` *et* `info/program.html` *ont également consulté l'URL* `info/debouches.html`.

L'extraction de motifs séquentiels permet, d'autre part, de mettre en évidence le type de relation suivant : *60 % des clients qui ont visité* `/jdk1.1.6/docs/api/Package-java.io.html`

et
`/jdk1.1.6/docs/api/java.io.BufferedWriter.html`*, ont également visité, dans les 30 jours suivants, l'URL* `/jdk1.1.6/docs/relnotes/deprecatedlist.html` ou *34 % des clients ont visité*
`/relnotes/deprecatedlist.html` *entre le 20 septembre et le 30 novembre.*

Une Modification Dynamique de Site Web

Les règles et motifs découverts constituent une connaissance très adaptée à l'optimisation dynamique de l'organisation hypertexte d'un serveur. Très utiles dans des applications telles que la conception de catalogue "*on-line*" pour le commerce électronique, ces techniques exploitent les informations pertinentes sur le comportement des utilisateurs pour réorganiser un site afin d'améliorer son efficacité. Par exemple, les informations proposées peuvent être utilisées par un serveur Proxy dans le but d'améliorer l'accès aux pages. Quand une requête atteint le serveur Web, la requête est envoyée au Proxy qui charge les documents désirés. Les motifs séquentiels découverts peuvent alors être utilisés pour pré-charger les documents pour les utilisateurs [17].

2.2 Aperçu des travaux antérieurs

Un panorama des systèmes et approches de Web Usage Mining ainsi que de leur utilisation est proposé dans [20]. Nous nous intéressons dans cette partie, aux approches répondant à la problématique présentée.

Une approche pour découvrir des informations à partir de fichiers access log est présentée dans [62, 19]. Les auteurs proposent l'architecture d'un système pour le Web Mining appelée WEBMINER. Même si les contraintes de temps ne sont pas prises en compte par le système, une approche pour rechercher des motifs séquentiels est proposée. Dans ce cas, le fichier access log est réécrit de manière à regrouper toutes les transactions d'un utilisateur si elles sont suffisamment proches dans le temps. Une transaction temporelle est alors vue comme un ensemble d'URL et de temps d'accès tels que les entrées sont regroupées si elles s'inscrivent dans un intervalle de temps Δt précisé par l'utilisateur. Un algorithme de recherche de règles d'association, similaire à celui de [6], est adapté aux motifs séquentiels. Enfin, le système propose un langage d'interrogation, basé sur SQL, pour offrir un meilleur contrôle sur le processus d'extraction.
Dans [50], un algorithme efficace pour la recherche de séquences d'événements, MINEPI, est utilisé pour extraire des règles à partir du fichier access log de l'Université d'Helsinki. Chaque page consultée est considérée comme un événement et une fenêtre de temps similaire au paramètre Δt de [19] permet de regrouper les entrées suffisamment proches.
Dans le projet WebLogMiner, [95], les auteurs proposent d'utiliser un système OLAP (On-Line Analytical Processing) pour extraire des informations significatives. Les données sont tout d'abord filtrées pour éliminer les informations non pertinentes puis stockées dans une base de données relationnelle. Ensuite, une structure de tableau multidimensionnel, appelée *Web Log data cube*, est construite et chaque dimension représente un champ avec toutes les valeurs possibles décrites par les attributs. Le système OLAP est alors utilisé dans la troisième phase pour appliquer des opérations de *drill-down*, *rool-up*, *slice* et *dice* sur le

data cube. Ces opérations offrent ainsi la possibilité d'examiner les données sous différents angles. Enfin, des fonctions de data mining comme la caractérisation, la recherche de règles d'association, la prédiction ou la classification peuvent être utilisées sur le Web Log data cube. Une approche similaire est présentée dans [22].

2.3 Synthèse des contributions

En définissant un système d'extraction, nous montrons qu'il est possible d'analyser efficacement le comportement des utilisateurs sur un site Web. L'approche présentée, basée sur les algorithmes définis dans le chapitre 1, offre la possibilité d'intégrer des contraintes de temps dans le processus. Cette particularité est garante d'offrir une meilleure vision dans le temps de l'usage d'un serveur Web.
Pour compléter l'approche proposée nous décrivons succinctement une expérience menée pour modifier dynamiquement la structure d'un site Web de manière à adapter le contenu des pages au profil courant des usagers.

2.3.1 WebTool : un système pour analyser le comportement des utilisateurs

La démarche que nous proposons vise aussi bien l'extraction de règles d'associations que de motifs séquentiels dans le cadre du Web mining. Le système WebTool est décrit dans [56, 57]. Ces principes généraux, illustrés par la figure 2.3, sont similaires à ceux du processus d'extraction de connaissances exposés dans [24]. La démarche se décompose en trois phases principales. Tout d'abord, à partir d'un fichier de données brutes, un pré-traitement est nécessaire pour éliminer les informations inutiles. Dans la deuxième phase, à partir des données transformées, des algorithmes de data mining sont utilisés pour extraire les itemsets ou les séquences fréquents. Enfin, l'exploitation par l'utilisateur des résultats obtenus est facilitée par un outil de requête et de visualisation.

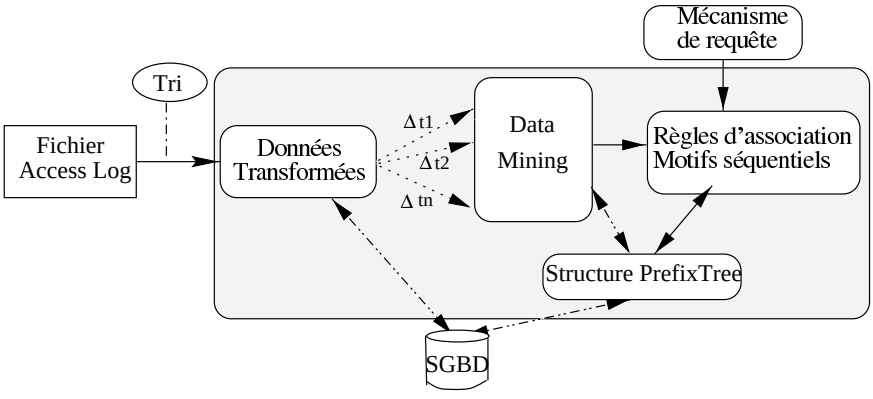

FIG. 2.3 – *Principe généraux de la démarche*

Les différentes phases introduites sont détaillées dans les paragraphes suivants.

Pré-traitement des données

Deux types de traitements sont effectués sur les entrées du serveur log. Tout d'abord, le fichier access log est trié par adresse et par transaction. Ensuite une étape d'élimination des données "non-intéressantes" est réalisée.

La plupart des outils d'analyse du Web profitent de cette étape pour éliminer des requêtes concernant des pages possédant des graphiques, des vidéos, des scripts CGI ou du son (par exemple, en éliminant les fichiers suffixés par *.GIF, .JPEG*) ou bien les requêtes issues d'agents ou de testeurs de liens. Même si WebTool offre également cette possibilité, nous préférons conserver ces informations comme dans le système WebLogMiner [95, 20]. En effet, ces données apportent des renseignements intéressants concernant aussi bien la structure du site Web, la motivation d'un utilisateur ou les performances du trafic. Par exemple, le fait de conserver des requêtes générées par des agents est utile pour analyser le comportement de l'agent et comparer le trafic généré par ces agents avec le reste du trafic [95]. De la même manière, si un utilisateur pose des requêtes pour des pages graphiques, certaines de ces requêtes sont importantes pour appréhender les actions de l'utilisateur.

Bien entendu, un tel choix nécessite la mise en œuvre d'algorithmes d'extraction efficaces car la taille du fichier access log reste très grande (au cours de nos expériences, nous avons constaté que la suppression des URL concernant des images réduisait la taille du fichier log de 40% à 85%). En outre, il est nécessaire d'offrir à l'utilisateur un outil de visualisation pour éliminer les données qui ne l'intéressent pas lors d'une analyse particulière.

Au cours de la phase de tri et afin de rendre plus efficace le traitement de l'extraction de données, les URL et les clients sont codés sous forme d'entiers. Toutes les dates sont également traduites en temps relatif par rapport à la plus petite date du fichier.

L'étape de pré-traitement des données offre également la possibilité de regrouper les entrées qui sont suffisamment proches. Contrairement au problème du supermarché où chaque transaction est définie comme un ensemble d'achats effectués par un client, les entrées dans le fichier log sont toutes des transactions séparées. De la même manière que [19], nous proposons de regrouper les entrées qui sont suffisamment proches en utilisant une contrainte de temps (Δt) spécifiée par l'utilisateur. La figure 2.4 illustre un exemple de fichier obtenu après la phase de pré-traitement.

Client	Date	URL traduite
1	01	10,30,40
1	02	20,30
2	11	10
2	12	30,60
2	23	20,50
3	01	10,70
3	12	30
3	15	20,30

FIG. 2.4 – *Exemple de fichier résultat issu de la phase de pré-traitement*

Outils d'Extraction

A partir des données transformées issues de la première étape, deux techniques d'extraction de connaissance peuvent être appliquées selon les besoins de l'analyste. Etant donné que le problème de la recherche de règles d'association est une réduction du problème de la recherche de motifs séquentiels, le principe retenu par WebTool est d'utiliser la même structure et les mêmes algorithmes pour obtenir des règles d'association. L'adaptation de PSP est prise en compte en considérant que toutes les transactions ont eu lieu en même temps. L'adaptation de PSP est détaillée dans [55]. L'intégration de la fouille incrémentale est définie dans [59]. Ainsi, lors de l'application de PSP, les temps des différentes transactions ne sont plus considérés et les résultats obtenus ne sont plus des séquences fréquentes mais des itemsets fréquents. La génération des règles basées sur ces itemsets est réalisée par l'outil de visualisation.

Outils de Visualisation

Devant le très grand nombre de motifs ou d'itemsets obtenus, il apparaît indispensable de proposer un mécanisme de requête pour offrir à l'utilisateur la possibilité de mieux analyser les informations découvertes lors la phase précédente. Un tel mécanisme peut intervenir de deux manières différentes :

1. lors de la phase d'obtention des règles d'association, i.e. après application des algorithmes de recherche de motifs ou d'itemsets fréquents. Il s'agit donc d'extraire des règles à partir des connaissances acquises par les outils d'extraction.
2. lors de l'extraction pour optimiser les temps de traitement réduisant la portion de base à extraire. Ce principe rejoint, par exemple, la phase de pré-traitement où les URL concernant des pages graphiques peuvent être éliminées de la base.

Dans le système WebTool nous privilégions, pour le moment, la première approche dans la mesure où nous considérons que l'utilisateur peut vouloir affiner ces recherches sans recommencer complètement le processus d'extraction.

A l'heure actuelle, de nombreux outils d'extraction proposent d'interroger les résultats via un langage de requête graphique [32]. Le système WebTool s'inspire de ces travaux et propose à l'utilisateur la possibilité de spécifier le type de règles désiré de la même manière que les "*templates*" utilisés dans le système Tasa [52, 50].

2.3.2 Modification dynamique de la structure hypertexte

Conjointement au système WebTool, nous avons développé un générateur de création de liens dynamiques dans des pages Web à partir des règles obtenues lors du processus d'extraction. Cette composante est détaillée dans [60]. Le but du générateur est de reconnaître un utilisateur et de vérifier son parcours dans les pages d'un serveur. Lorsque celui-ci correspond à une règle d'association ou à un motif séquentiel déterminé par WebTool, les pointeurs des pages sont dynamiquement mis à jour.

L'architecture fonctionnelle de l'approche est décrite dans la figure 2.5. Le serveur Web (démon *http*) réagit à l'envoi d'une requête par un client en lui retournant une page contenant une applet. Celle-ci est chargée de se connecter au serveur de clients (*gestionnaire de*

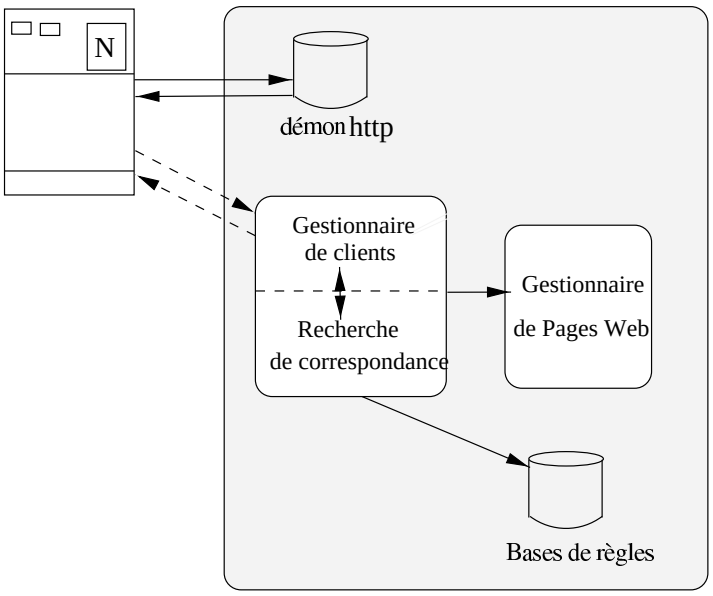

FIG. 2.5 – *Architecture générale*

clients) pour lui transmettre l'adresse Internet du client et la page demandée. A l'heure actuelle, le gestionnaire de clients est une application java qui fonctionne sur la même machine que le serveur *http* pour permettre à l'applet l'utilisation d'un mécanisme client/serveur.
Lors de la réception de l'adresse et de la page, le gestionnaire de clients examine le comportement du client via le module de recherche de correspondances qui est chargé de vérifier si le comportement du client est en adéquation avec une règle préalablement extraite par le processus de data mining.

Lorsqu'une entrée est jugée significative par le gestionnaire de correspondance, la page demandée par l'utilisateur est modifiée par le gestionnaire de pages en ajoutant dynamiquement des liens vers les conséquents de la règle reconnue. L'applet récupère alors l'URL d'accès à cette page et l'affiche sur le navigateur. Si aucune règle ne correspond au comportement du client, l'URL vers la page demandée est retournée à l'applet pour qu'elle puisse l'afficher.

2.4 Discussion

Le système WebTool aborde la problématique de la recherche de comportement d'usager sur un site Web de manière particulièrement originale. Le double objectif ayant guidé cette démarche est d'une part la proposition d'un système capable d'offrir à l'utilisateur final aussi bien une précision sur les pages corrélées que sur le comportement des usagers à travers le temps et d'autre part l'utilisation des résultats obtenus pour modifier dynamiquement un site.
L'approche proposée ne vise pas l'exhaustivité mais se veut suffisamment générique pour prendre en compte de nouveaux types d'algorithmes. Par exemple, une composante segmentation serait particulièrement utile dans un tel système de manière à permettre de re-

grouper les différents utilisateurs par catégorie. Cependant, en proposant une composante de recherche de motifs séquentiels et de règles d'association nous répondons en partie à la problématique de l'analyse du comportement des utilisateurs.

Outre, le fait que la proposition soit basée sur des algorithmes efficaces, ce qui est particulièrement important dans un tel contexte étant donné les quantités de données manipulées (la plupart des approches de recherche de motifs utilisées sont basées sur des extensions d'algorithmes de règles d'association [20] qui ont pourtant montré leur limite pour les motifs séquentiels [7, 83]), elle offre également une prise en compte des contraintes de temps. Cette composante, inexistante dans les autres propositions, permet d'offrir à l'utilisateur final une analyse plus précise du comportement des clients.

En intégrant la gestion de recherche incrémentale, nous étendons le caractère original de la proposition. En effet, l'une des particularités d'un serveur Web est d'avoir des données qui évoluent constamment. La seule approche, à l'heure actuelle qui permet de prendre en compte ces évolutions est l'approche du projet WebLogMiner [95], mais au travers d'un système OLAP. L'intégration de la composante recherche incrémentale se fait de manière tout à fait naturelle dans le système WebTool proposé et permet de maintenir une base de connaissance extraite.

Les expériences menées sur la modification dynamique de la structure hypertexte d'un serveur conforte le fait que la proposition permet de répondre à une problématique très actuelle des gestionnaires de serveur Web : comment maintenir les clients sur le serveur ? L'une des solutions préconisée est justement d'essayer d'adapter le contenu du serveur pour qu'il soit le plus en adéquation avec le comportement des utilisateurs. Notre proposition est un premier élément de réponse à ce problème.

L'approche proposée ouvre également la voie à un autre type d'application : la détection d'intrusion sur des systèmes [46]. En effet, dans ces applications, les données manipulées correspondent à des fichiers ordonnés dans le temps (tcpdump, BSM), et elles abordent deux problématiques principales : reconnaître les comportements usuels de ceux correspondant à une fraude et être capable de reconnaître en temps réel une tentative de fraude dans le système. Pour résoudre la première problématique les auteurs préconisent l'utilisation d'algorithme de recherche de motifs séquentiels efficaces dans la mesure où la prise en compte du temps est un élément particulièrement crucial dans l'examen du comportement des utilisateurs du système.

Chapitre 3

Régularités dans des bases de données complexes

Ces dernières années, les Systèmes de Gestion de Bases de Données (SGBD) ont considérablement évolués pour prendre en compte les besoins de nouvelles applications. Par exemple, un grand nombre de travaux ont été réalisés pour définir des systèmes capables de manipuler des données qui ne sont pas en première forme normale. Les extensions, proposées notamment par les systèmes objet, et même à l'heure actuelle par les entrepôts de données, permettent ainsi de stocker des structures de plus en plus complexes.
Malheureusement, comme nous l'avons signalé dans le chapitre d'introduction, les approches de fouilles de données actuelles ne prennent pas en compte de telles structures et se contentent d'analyser des structures "plates", i.e. des éléments atomiques comme les achats de clients par exemple. Pourtant, dans de nombreuses applications (e.g. les "workflows", les systèmes actifs, les systèmes de gestion de contraintes, les systèmes transactionnels, les applications fonctionnant sur des SGBD Orientés Objets, etc.), la recherche de régularités dans des transactions imbriquées peut permettre d'obtenir de nombreuses informations pour améliorer la qualité des applications, pour optimiser les accès, pour favoriser les techniques de rétro-conception, pour assurer un meilleur suivi de l'exécution de tâches ou de méthodes etc.

Avec la popularité du World Wide Web (Web), le nombre de documents semi-structurés produits augmente très rapidement. Alors que dans les applications de bases de données classiques, nous décrivons d'abord la structure des données, i.e. le type ou le schéma, et nous créons les instances de ces types, dans les données semi-structurées, les données n'ont pas de schéma prédéfini et chaque objet contient sa propre structure [14].
En effet, la majorité des documents "en-ligne", tels que les fichiers HTML, Latex, Bibtex ou SGML sont semi-structurés. Par conséquent, la structure des objets est irrégulière et il est judicieux de penser qu'une requête sur la structure des documents est aussi importante qu'une requête sur les données [89]. Cependant, cette irrégularité structurelle n'implique pas qu'il n'existe pas de similitudes structurelles parmi les objets semi-structurés. Il est même parfois assez fréquent de constater que des objets semi-structurés qui décrivent le même type d'information ont des structures similaires. L'analyse de telles régularités dans des données semi-structurées peut alors fournir des informations très utiles pour le concepteur ou l'utilisateur d'un site.

Dans ce chapitre, nous détaillons la recherche de régularités d'objets complexes définis sous la forme de transactions imbriquées ou d'objets semi-structurés. Dans le paragraphe 3.1, nous exposons la problématique en détaillant les deux types de données complexes. Le paragraphe 3.2 présente les approches contribuant à la recherche de ces données. La présentation de nos contributions est proposée au paragraphe 3.3. Enfin, une discussion conclue ce chapitre.

3.1 Exposé de la problématique

Les Transactions Imbriquées

Le problème de la recherche de régularités dans des transactions imbriquées est basé sur la recherche de régularités d'items complexes dans une base de transactions. Un item, dans notre contexte, est soit simple, soit complexe. Un item simple correspond à un objet élémentaire de l'application (nom de fonction, nom d'une tâche, etc.) Un item complexe quant à lui est composé d'autres items via des constructeurs "liste-de" ou "ensemble-de".

Exemple 1 Pour illustrer les notions d'items simples et complexes, considérons une base de données contenant des transactions constituées à partir de bibliothèques de programme. Chaque programme est défini soit par une tâche[1] élémentaire (un item simple) soit par une succession de tâches élémentaires (item complexe). Si nous considérons maintenant, plus en détail, une tâche complexe, celle-ci peut être vue comme la suite d'un ensemble de tâches complexes qui elles-mêmes sont formées de tâches complexes jusqu'à obtenir une tâche élémentaire. Considérons quatre tâches élémentaires a, b, c et d. Considérons les tâches complexes suivantes : $A = \{C,a,b\}$, et $C =< d,c >$. La tâche complexe A est en fait réalisée de la manière suivante : $\{< d,c > ,a,b\}$. En d'autres termes, la réalisation de la tâche A consiste à d'abord exécuter soit les tâches élémentaires d et c (dans cet ordre) puis après la tâche a et b (dans n'importe quel ordre), soit la tâche a et b (sans ordre) puis l'association des tâches d et c. □

La problématique liée à la recherche de régularités dans des transactions imbriquées est définie de la manière suivante : soit DB une base de données de transactions imbriquées. Une transaction t contient un élément E si et seulement si $E \subseteq t$. Le support de E est défini comme le pourcentage de transactions dans DB contenant E : $supp(E) = \|\{t \in DB \mid E \subseteq t\}\|/\|\{t \in DB\}\|$. Etant donné un support minimal, $minSupp$, donné par l'utilisateur, le problème de la recherche de régularités dans des transactions imbriquées consiste à rechercher tous les éléments E de niveau maximal tels que leur support est supérieur ou égal à $minSupp$.

Définie de cette manière, la problématique ressemble à celle de la recherche de règles d'association ou de motifs séquentiels (C.f. Chapitre 1). Cependant, contrairement à ces

1. Nous prenons volontairement la notion de tâche, semblable aux tâches Ada, pour permettre d'illustrer la possibilité d'avoir deux tâches consécutives et dans un ordre bien précis.

approches, les éléments manipulés sont composés de structures complexes. En effet, dans le cas des règles d'association, nous nous intéressons à la recherche de corrélation entre items, i.e. la recherche d'ensembles d'items maximaux dans la base. Dans le cas des motifs séquentiels, même si la notion de liste est prise en compte, il s'agit de liste d'ensembles d'items où les items sont atomiques.

Exemple 2 Pour illustrer la composante de recherche dans des transactions imbriquées, considérons une base de données constituée d'un ensemble de transactions contenant des déclenchements de méthodes (par exemple, une base similaire pourrait être obtenue en analysant les fichiers log de règles actives).

Id	Transactions
t_1	<create(client), modify(order.status), <modify (client.credit), modify (client.trusted)>, modify(amount), modify(order.status), <modify (client.credit), modify (client.trusted)> >
t_2	<create(client), modify(amount), <modify(order.status), <modify(client.credit), modify(client.trusted)>, modify(client.credit)>, modify(status), <modify(client.credit), modify(client.trusted)>, modify(amount)>
t_3	<create(client), <modify (client.credit), modify (client.trusted)> >
t_4	<create(client), modify(order.status), <modify (client.credit), modify (client.trusted)> >
t_5	<create(client), modify(order.status), <modify (client.credit), modify (client.trusted)>, modify(amount),<modify (client.credit), modify (client.trusted)>, modify(order.amount) >

Considérons que le support minimal spécifié par l'utilisateur est de 50%, c'est à dire que pour qu'une transaction imbriquée soit fréquente, il faut qu'elle apparaisse dans au moins trois transactions de la base. Nous trouvons les transactions imbriquées suivantes :

< *create(client)*, *modify(order.status)*, < *modify(client.credit)*, *modify(client.trusted)*> *modify(amount)*>
et < *create(client)*, *modify(client.credit)*, <*modify(client.trusted)* > >.

En effet, la première de ces transactions est imbriquée dans trois transactions de la base de données : t_1,t_2 et t_5. La seconde, par contre, est incluse dans toutes les transactions de la base. □

Un Système d'Extraction de Données Semi-Structurées sur le Web

Dans le cadre de la recherche de régularités dans des données semi-structurées issues du Web, le modèle de données utilisé est basé sur le modèle OEM que nous étendons pour pouvoir prendre en compte la notion de "liste-de".
La figure 3.1 décrit une partie des informations sur les restaurants dans la Bay Area [1]. Chaque cercle avec le texte associé représente un objet et son identificateur. Les arcs et les étiquettes associés représentent des références d'objet. Par exemple, *value(&44) = "El*

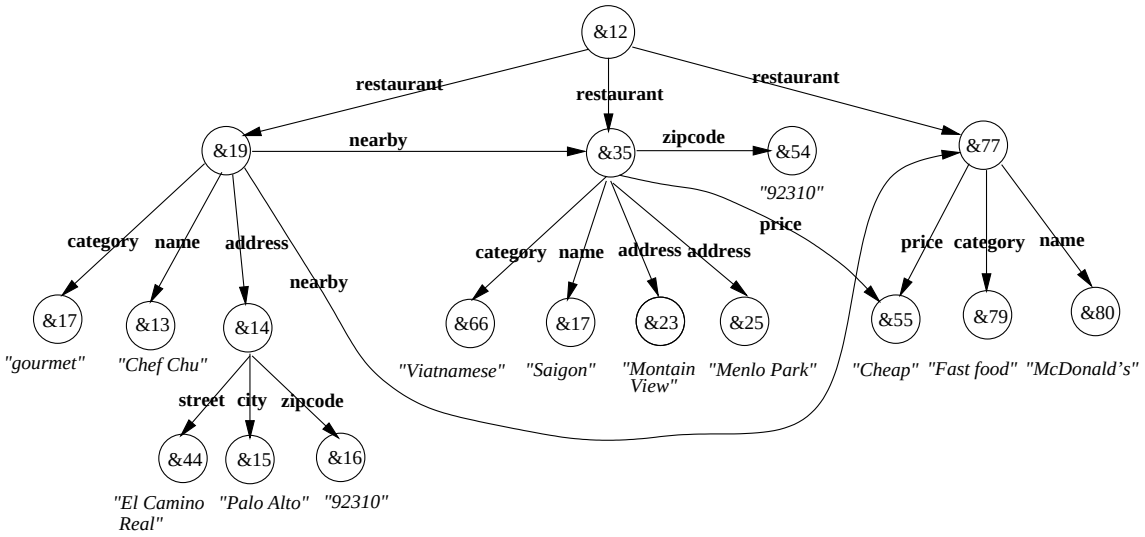

FIG. 3.1 – *Un exemple de graphe OEM*

Camino Real" et *value(&19)="setof"*, spécifie que *category*, *name* et *address* peuvent ne pas être ordonnés.

Un chemin simple dans le graphe est une séquence alternative d'objets et d'étiquettes:

$$< o_1 l_1 o_2 ... o_{k-1} l_{k-1} o_k >$$

commençant et se terminant avec des objets et dans laquelle chaque étiquette a pour nœud de départ et d'arrivée respectivement le nœud qui le précède immédiatement et le nœud qui lui succède immédiatement. Le nombre d'étiquettes k parcouru depuis un nœud de départ jusqu'au dernier nœud dans le chemin correspond à la longueur du chemin. Un chemin multiple ou *chemin* est un ensemble de chemins simples tels que l'objet source de ces chemins soit le même.

Soit DB un ensemble de transactions où chaque transaction T est composée d'un identifiant et d'un chemin inclus dans le graphe OEM. Toutes les transactions sont triées par ordre croissant. La figure 3.2 illustre un exemple d'une base de données de transactions contenant des expressions de chemins.
Soit *supp(p)*, une valeur de support pour un chemin correspondant au nombre d'occurrences de ce chemin dans la base de données DB. En d'autres termes, le support d'un chemin p est défini comme le pourcentage de toutes les transactions qui contiennent p. Une transaction contient p si est seulement si p est un sous-chemin de la transaction.

Etant donnée une valeur de support minimal spécifiée par l'utilisateur (*minSupp*), un chemin est *fréquent* si la condition $supp(p) \geq minSupp$ est vérifiée.
Comme nous nous intéressons à la recherche des plus longs chemins, nous précisons qu'un chemin p_m est un sous-chemin d'un chemin p_n si p_m est inclus dans p_n, où l'inclusion de chemin est définie de la manière suivante: un chemin $p_a = <o_{a_1} l_{a_1} o_{a_2} .. o_{a_{k-1}} l_{a_{k-1}} o_{a_n}>$est inclus dans un chemin $p_b = <o_{b_1} l_{b_1} o_{b_2} .. o_{b_{k-1}} l_{b_{k-1}} o_{b_m}>$ si et seulement s'il existe des entiers $i_1 < i_2 < \ldots < i_n$ tels que $o_{a_1} = o_{b_{i_1}}$, $o_{a_2} = o_{b_{i_2}}, \ldots, o_{a_n} = o_{b_{i_n}}$.

Trans_id	chemins
t_1	$person : \{identity: \{name: \perp, address; \perp\}\}$
t_2	$person : \{identity: \{name: \perp, address: < street: \perp, zipcode: \perp >,$ $company: \perp, director: < name: \perp, firstname: \perp >\}\}$
t_3	$person : \{identity: \{id: \perp, address: < street : \perp, zipcode : \perp >\}\}$
t_4	$person : \{identity: \{name: \perp, address: \perp, company: \perp\}\}$
t_5	$person : \{identity: \{name: \perp, address; \perp\}\}$
t_6	$person : \{identity: \{name: \perp, address: < street: \perp, zipcode: \perp >,$ $director: < name: \perp, firstname: \perp >\}\}$

FIG. 3.2 – *Un exemple de base de données de transactions*

Etant donnée une base de données DB de transactions, le problème de la recherche de régularités dans des données semi-structurées consiste donc à rechercher tous les chemins maximaux qui sont dans DB et dont le support est supérieur à *minSupp*.

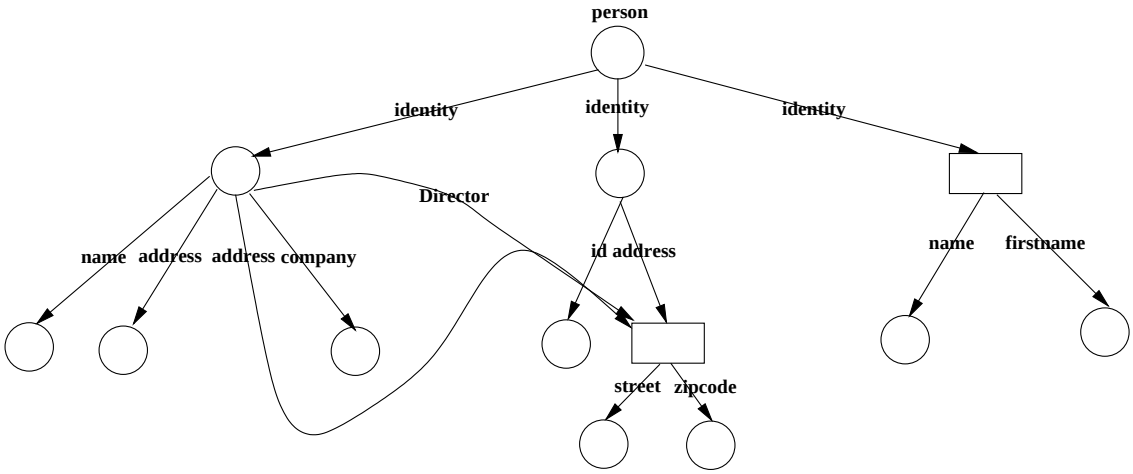

FIG. 3.3 – *Le graphe OEM associé*

Exemple 3 Pour illustrer le problème de la recherche de régularités structurelles dans une base de données d'objets semi-structurés, considérons la base DB de la figure 3.2. Supposons que la valeur de support minimum spécifié par l'utilisateur soit 50%, c'est à dire que pour être fréquent un chemin doit apparaître dans au moins trois transactions. La figure 3.3 décrit le graphe OEM associé aux transactions de la base de données.
Les seuls chemins fréquents dans DB sont les suivants :

identity: $\{name: \perp, address: \perp\}$,
et *identity*: $\{address: < street: \perp, zipcode: \perp >\}$.

Le premier car il est vérifié dans la transaction t_1 mais également dans les

transactions t_4 et t_5. De la même manière

identity: {*address*: < *street*: ⊥, *zipcode*: ⊥ >}

est inclus dans les transactions t_2, t_3 et t_6.
Par contre, le chemin

identity: {*name*: ⊥, *address*: < *street*: ⊥,*zipcode*: ⊥ > ,*director*: < *name*: ⊥, $firstname$: ⊥ >}

est vérifié par les transactions t_2 et t_6 mais n'est pas fréquent puisque le nombre de transactions qui possèdent ce chemin est inférieur à la contrainte de support minimal. □

3.2 Aperçu des travaux antérieurs

A notre connaissance, il existe très peu de travaux concernant la recherche de régularités structurelles dans de grandes bases de données. Néanmoins, notre approche est très proche de celle proposée dans [91, 89] pour la recherche d'associations structurelles dans des données semi-structurées. Les auteurs proposent une approche très efficace et des solutions basées sur une nouvelle représentation de l'espace de recherche. En outre, en proposant des optimisations basées sur des stratégies d'élagage, ils améliorent considérablement l'étape de génération des candidats.
Notre approche a cependant un certain nombre de différences avec l'approche de [91, 89]. Nous nous intéressons à toutes les structures stockées dans les transactions de la base alors qu'ils sont uniquement intéressés par la recherche de *tree expression* qui sont définis comme un chemin allant de la racine d'un graphe OEM à un nœud final du graphe. Avec cette définition de *tree expression*, ils ne peuvent pas trouver de régularités de la forme *identity*: {*address*: < *street*: ⊥, *zipcode*: ⊥ >} qui seraient incluses dans une transaction plus longue mais qui elle n'est pas fréquente. En fait, lors du parcours de la base de données pour rechercher les structures fréquentes, ils ne recherchent que les arbres maximaux et quand seulement une partie de cet arbre est fréquente, elle n'est pas découverte.

L'approche WARMR proposée dans [21] aborde la recherche de sous structures fréquentes dans des composants chimiques. Toutefois même si la problématique est similaire, ils considèrent que les sous-structures recherchées sont exprimées sous la forme de requête DATALOG. De manière générale, les motifs autorisés sont spécifiés via un langage déclaratif (de la même manière que les approches basées sur la logique inductive (ILP)) et sont stockés dans un treillis. A partir d'un algorithme par niveau ils déterminent quelles sont les clauses les plus fréquentes et trouvent ainsi les structures fréquences.

Dans [31, 81] des approches pour rechercher des règles d'association multi-niveaux ont été proposées. Les auteurs supposent que pour rechercher de telles règles, ils disposent d'une base de données de transactions de clients et d'une taxonomie (*hiérarchie is-a*) entre les items achetés par les clients. Les travaux proposés semblent proches de notre problématique dans la mesure où les éléments recherchés sont des ensembles d'items issus de différents niveaux de taxonomie. En fait, ces approches sont très différentes de la notre car elles s'intéressent uniquement à des données enrichies par une hiérarchie is-a alors que nous nous intéressons à des transactions dans lesquelles les données sont enrichies aussi bien par des

hiérarchies *ensemble-de* que *liste-de*.

La découverte d'information structurelle à partir de données semi-structurées a été très largement étudiée. Dans ce contexte, il existe un grand nombre de propositions pour extraire les types sous-jacents de données semi-structurées [66, 11, 65]. Par exemple, dans [65], les auteurs se sont intéressés à l'extraction de la structure implicite d'un schéma semi-structuré. Ces approches sont cependant éloignées de notre problématique dans la mesure où nous nous intéressons à la répétition de structures dans un schéma. Par contre, dans notre contexte, de telles approches peuvent être très utiles pour réaliser une étape de prétraitement des données afin de transtyper les données d'un graphe OEM. Dans ce cas, nous pourrions compléter la recherche de régularités en enrichissant les éléments manipulés avec les types de données associés.

3.3 Synthèse des contributions

En définissant une composante Transactions imbriquées, nous montrons qu'à l'aide d'une transformation efficace des données initiales, nous pouvons transformer la problématique initiale en recherche de motifs séquentiels. L'approche proposée dans cette composante est décrite dans [44]. Cette transformation est ensuite appliquée à la recherche de régularités sur des données semi-structurées issues du Web au travers de la spécification d'un système d'extraction dédié.

3.3.1 Composante Transactions Imbriquées

La recherche de régularités dans des transactions imbriquées fréquentes est réalisée en deux étapes.

1. *Phase de transformation* : cette étape est chargée de transformer les transactions initiales en données accessibles par la phase de recherche ;
2. *Phase de recherche* : cette étape recherche les transactions imbriquées fréquentes dans la base de données transformée.

La base de données est initialement triée avec l'identifiant de la transaction comme clé principale et les éléments simples stockés dans les ensembles d'éléments sont triés par ordre lexicographique. Cette dernière opération nous permet de réduire considérablement l'espace de travail dans la mesure où, les éléments d'un ensemble n'étant pas ordonnés, trop de combinaisons doivent être prises en compte.
Chaque transaction contenant des éléments complexes est alors transformée de la manière suivante :

1. *Prise en compte de la profondeur d'imbrication et du type complexe* : chaque élément simple est considéré et si cet élément est une partie d'un ensemble alors l'élément simple est précédé par la lettre 'S' (respectivement la lettre 'L' s'il s'agit de liste d'éléments simples). En outre, pour prendre en compte le niveau d'imbrication de l'élément, un entier, décrivant la profondeur d'imbrication de cet élément dans la structure complexe, est ajouté à l'élément.
2. *Création de liste d'ensembles d'éléments simples* : quand deux éléments sont au même niveau et si le premier est directement suivi par le second, nous les regroupons dans

un même ensemble autrement ils sont inclus dans deux ensembles séparés. La notion d'ordre de la valeur "liste-de" est, par contre, prise en compte en créant de nouveaux ensembles entre les éléments.
La transaction "composite" qui résulte de l'union de ces ensembles obtenue à partir des transactions initiales décrit une séquence d'éléments simples modifiés et l'ordre de cette séquence peut alors être vu comme une navigation en "profondeur d'abord" des transactions.

A la fin de cette phase, la base de données DB est transformée en une nouvelle base de séquence DB' dans laquelle l'imbrication dans les transactions a été modifiée en une liste ordonnée d'éléments simples.

Exemple 4 Pour illustrer la phase de transformation, considérons les deux transactions suivantes: $t_1 = \{a,\{c,\{d,f\}\}\}$ et $t_2 = \{a, < e,\{f,b\},d, < g,h >> ,c\}$. Dans la première transaction, étant donné que tous les éléments simples apparaissent dans un ensemble de valeurs, le symbole 'S' leur est affecté. Concernant le niveau d'imbrication de la transaction, nous affectons à chaque élément simple son niveau par rapport à l'ensemble le "plus haut" de la hiérarchie. Les éléments simples a,b,c,d et f sont alors transformés de la manière suivante: Sa_1,Sc_2,Sd_3,Sf_3. En affectant chaque élément simple dans un nouvel ensemble et en parcourant en profondeur d'abord la transaction, nous obtenons la transformation suivante pour la transaction t_1: $t_1 = (Sa_1)\ (Sc_2)\ (Sd_3\ Sf_3)$. En appliquant le même principe pour la transaction t_2, nous obtenons: $t_2 = (Sa_1)\ (Le_2)\ (Sf_3\ Sb_3)\ (Ld_2)\ (Lg_3)\ (Lh_3)\ (Sc_1)$. Nous pouvons constater que la modification de la transaction t_2 respecte l'ordre de parcours en profondeur d'abord. Examinons en détail la partie "liste-de" $< g : \perp, h : \perp >$ de la transaction t_2. Comme les éléments g et h sont ordonnés à cause de la valeur "liste-de", nous considérons que même s'ils interviennent au même niveau, et même si h suit directement g, ils ne peuvent pas être regroupés dans un même ensemble. □

Le fait de transformer les données initiales sous la forme de séquences d'éléments simples nous permet de nous rapprocher de la problématique de la recherche de motifs séquentiels (C.f. Partie II, Chapitre 1). En effet, nous n'avons plus qu'à rechercher dans les données transformées quelles sont les listes d'éléments simples qui interviennent suffisamment fréquemment pour être intéressantes, i.e. quelles sont les listes d'éléments simples dont le nombre d'occurrences dans la nouvelle base de données DB' est supérieur au support minimal spécifié par l'utilisateur. Les optimisations apportées à l'algorithme de recherche de motifs séquentiels sont décrites dans [43].

3.3.2 Composante Données Semi-Structurées

La problématique de la recherche de régularités dans des données semi-structurées est très similaire à celle que nous avons présentée précédemment. En fait, l'algorithme proposé permet également de résoudre ce problème. Cependant, l'analyse de tels documents pose d'autres types de problèmes qui nous ont amené à définir un système d'extraction de connaissances spécifique. Le système est décrit dans [45].

Inspiré des travaux menés dans le cadre du Web usage Mining, nous considérons que le mécanisme de la découverte de régularités dans de grandes bases de données semi-structurées est également un processus qui se fait en deux étapes. La première étape concerne la recherche des objets semi-structurés sur le Web et leur stockage dans une base. A partir de cette base, la phase de transformation modifie les données originales de la même manière que dans la composante précédente. Le résultat de cette transformation est une nouvelle base de données contenant les informations utiles sur lesquelles des techniques de fouille de données peuvent être appliquées pour rechercher les régularités dans les expressions des chemins. L'architecture fonctionnelle de notre approche est décrite dans la figure 3.4.

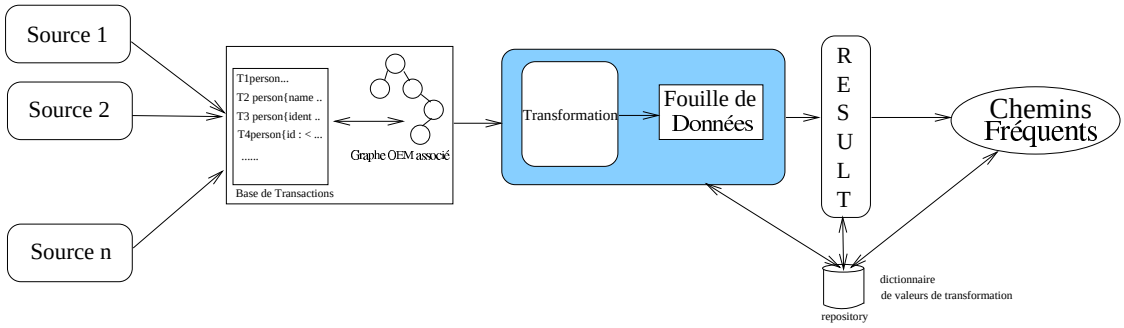

FIG. 3.4 – *Architecture générale*

Extraction des données

La structure de chaque objet de la source (page Web) doit être extraite au cours de cette phase. Tout d'abord une étape de filtrage est faite de manière à éliminer les données qui ne sont pas utiles dans l'analyse : image, vidéo, son, etc. En outre, en fonction du point de vue de l'utilisateur, les sous-structures "non intéressantes" sont également éliminées.
Cette extraction est réalisée sur un grand nombre de sources de manière à obtenir un ensemble suffisant de structures dans lesquelles nous souhaitons découvrir les sous-structures régulières.

Pour réaliser cette phase d'extraction, nous considérons que l'utilisateur final possède un parser ou un wrapper. Bien entendu, ce parser nécessite l'implémentation d'algorithmes efficaces pour extraire les structures de graphe sous-jacentes à une ou à un ensemble de pages. Par exemple, dans [29], un outil efficace d'extraction de données semi-structurées à partir d'un ensemble de pages HTML a été proposé. La particularité de cet outil est qu'il propose directement une conversion des données extraites dans un graphe OEM.

Extraction de connaissances

A partir des données extraites, nous appliquons l'algorithme décrit dans le chapitre précédent. Les données sont d'abord transformées puis l'algorithme de fouille de données est appliqué.
Enfin, à partir des résultats obtenus, une étape de transformation inverse est réalisée de manière à présenter les résultats sous la forme de transactions.

Trans_id	expressions de chemins transformées
t_1	$(Sidentity_1)$ $(Sname_2\ Saddress_2)$
t_2	$(Sidentity_1)$ $(Sname_2\ Saddress_2)$ $(Lstreet_3)(Lzipcode_3)$ $(Scompany_2\ Sdirector_2)$ $(Lname_3)$ $(Lfirstname_3)$
t_3	$(Sidentity_1)$ $(Sid_2\ Saddress_2)$ $(Lstreet_3)$ $(Lzipcode_3)$
t_4	$(Sidentity_1)$ $(Sname_2\ Saddress_2\ Scompany2)$
t_5	$(Sidentity_1)$ $(Sname_2\ Saddress_2)$
t_6	$(Sidentity_1)$ $(Sname_2\ Saddress_2)$ $(Lstreet_3)$ $(Lzipcode_3)$ $(Sdirector_2)$ $(Lname_3)$ $(Lfirstname_3)$

FIG. 3.5 – *Une base de données de transactions après la phase de transformation*

La figure 3.5 illustre la nouvelle base de données obtenue après l'application de la phase de transformation sur notre base de données exemple.

Expérimentation

Pour valider notre approche, nous avons, de la même manière que dans [89], réalisé des expériences sur la base de données des films sur Internet (http://us.imdb.com) afin de rechercher des structures typiques de documents concernant le cinéma. Le résultat des expériences est détaillé dans [43]. Cette base de données regroupe, en mai 2000, plus de 200 000 films dont toutes les informations sont organisées sous la forme de pages HTML.
Suite à l'examen de quelques exemples de films, nous nous sommes intéressés à la partie de la base qui concerne les 250 meilleurs films afin d'en extraire les informations concernant les acteurs.

Après avoir récupéré les données concernant les acteurs, nous avons extrait des pages HTML les informations significatives des pages concernant la structure des documents de type acteur. Cette étape a été réalisée en partie automatiquement à l'aide d'un parser mais également manuellement dans la mesure où le contenu des pages ne permettait pas d'extraire, sans ambiguïté, les structures sous-jacentes. Au total nous avons récupéré une base de 500 acteurs dont la profondeur maximale était de 5.

En appliquant les algorithmes de recherche, nous avons par exemple, pour un support de 50%, la structure commune suivante :

$\{actor : \{name, dateofbirth, filmographyas : \{title, notabletv\}\}\}$

indiquant que pour au moins 250 acteurs de la base, un acteur possède un nom, une date de naissance et une filmographie. Dans cette filmographie, il apparaît à la fois dans un film mais il a également effectué des apparitions à la télévision.

3.4 Discussion

Pour évaluer l'approche dans le cadre des transactions imbriquées, nous avons effectué différentes expérimentations sur des jeux de données synthétiques obtenus en modifiant le générateur d'IBM. Les expériences ont montré que l'approche proposée était efficace. Néanmoins, le générateur utilisé ne nous a pas permis de pousser plus en avant certaines

expériences. Par exemple, comment se comporte notre algorithme dans le cas de données vraiment très imbriquées (niveau d'imbrication supérieur à 10) ? Les premiers résultats nous ont permis de valider l'approche sur des niveaux d'imbrication de profondeur 6. Bien entendu, avoir des niveaux profonds implique comme conséquence première de rechercher des séquences de taille très longue et donc d'étendre les temps d'exécution de l'algorithme. Il s'agit actuellement d'une des limites des approches de recherche de règles d'association ou de motifs séquentiels qui recherchent généralement des séquences fréquentes de taille limitée. Il faut toutefois préciser cependant que dans la majorité des applications visées ces limites sont tout à fait suffisantes. Lors d'applications plus spécifiques, un pré-traitement des données plus efficace permet alors généralement de résoudre ces problèmes. Par exemple dans le cas du suivi de l'exécution de programme, à partir du moment où nous avons une connaissance sur le domaine traité, il est tout à fait possible de ne pas décrire complètement chaque procédure mais plutôt de les regrouper.

quid de la modélisation des données

Dans l'approche présentée, nous avons considéré que, même si les données étaient imbriquées, elles étaient stockées dans une relation unique composée d'un identifiant et des transactions associées. Cette approche est valide dans de nombreuses applications comme les "logs" où les éléments s'enchaînent les uns après les autres. Par contre, que se passe-t-il si les données sont modélisées au travers de plusieurs relations ? Pour illustrer ce problème de modélisation et ses conséquences, considérons la relation suivante qui contient un ensemble d'items simples et des items définis dans une autre relation.

- $t_1 = \{A,A,B,x\}$
- $t_2 = \{C,w,y\}$

Considérons maintenant que A, B et C sont respectivement définis par $A =< w,x >$, $B =< x,y >$, $C =< x,y,z >$.

Une approche de résolution triviale de ce problème est de remplacer directement A,B et C dans les transactions de manière à obtenir :

- $t_1 = \{< w,x > , < w,x > , < x,y > ,x\}$
- $t_2 = \{< x,y,z > ,w,y\}$

Cependant, cette approche est loin d'être efficace dans la mesure où elle engendre de nombreux problèmes de mise à jour.

Il est intéressant de souligner que cette modélisation des transactions est utilisée dans l'approche de recherche de régularités proposée dans [90]. Pour résoudre ce problème, les auteurs considèrent que les transactions sont stockées sur le disque alors que les opérations comme A, B ou C sont stockées en mémoire centrale. Cette approche a pour avantage bien entendu de supprimer les problèmes de mise à jour de la base de transaction et d'optimiser les recherches à l'aide des informations stockées en mémoire. Cependant, la limite principale de cette approche est que si le nombre d'opérations stockées en mémoire est trop grand, il est nécessaire de les retransférer sur disque et de faire une lecture sur le disque pour chaque appel de l'opération.

Concernant l'extraction de connaissances de données semi-structurées issues du Web, les premiers résultats obtenus sur la base de données internationale des films ont montrés que l'approche était adaptée. Cependant, la difficulté essentielle que nous avons rencontrée réside dans le fait qu'il est indispensable de définir un "parser" adapté à chaque type d'application. Dans le cas des films, la difficulté est étendue par la différence de représentation

de la structure des films dans la base.

Bien que l'approche proposée ait été validée à l'aide des acteurs, nous avons constaté au cours de nos expériences, qu'il existe des applications (typiquement, celle du cinéma) où il est possible d'améliorer la recherche de régularités. L'amélioration est basée sur le fait qu'il est possible d'optimiser la génération des candidats, i.e. ne générer que des candidats qui sont forcément inclus au moins une fois dans la base et donc de limiter le temps de comparaison entre les transactions et les candidats. Ceci peut être réalisé en parcourant une première fois la base et en stockant chaque séquence dans une structure adaptée. Cette optimisation cependant n'est réalisable que dans le cas d'applications très particulières contenant peu d'items et dont les séquences fréquentes sont nombreuses.

Bien que la problématique de la recherche de régularités dans des bases de données d'objets complexes n'ait pas encore été beaucoup étudiée dans les travaux actuels, il est indéniable qu'elle apparaisse comme le sujet d'une préoccupation forte dans les années à venir. En effet, avec la popularité sans cesse croissante du Web, le nombre d'objets semi-structurés ou d'applications réparties augmente très rapidement. Malheureusement la prise en compte de ces éléments n'est pas encore réalisée de manière satisfaisante : comment interroger une base de données semi-structurée ? comment analyser efficacement les déclenchements d'opérations réparties dans le cas de systèmes de bases de données actives couplées à un data warehouse ?. Les solutions proposées sont généralement basées sur le fait que la structure complexe est déjà connue à l'avance (par exemple les langages de requêtes pour les données semi-structurées). La solution envisagée dans ce chapitre est une réponse partielle à ces problèmes et à ceux des travaux actuels sur les données semi-structurées qui mettent en avant la nécessité d'utiliser des approches de fouille de données pour permettre une meilleure appréhension des données manipulées [14].

Conclusion

Pour conclure le travail présenté, nous nous proposons, dans ce chapitre, de dresser un bilan des contributions apportées en résumant les points forts déjà signalés dans les précédentes discussions.

Sur le plan de la recherche de motifs séquentiels, l'algorithme PSP propose, au travers d'une nouvelle structure, une recherche efficace des motifs d'une grande base de données. Dans une problématique générale d'aide à la décision, le fait de développer une approche efficace se justifie par la nécessité d'obtenir rapidement les résultats de manière à faciliter les prises de décisions.
Les expériences menées avec PSP ont montré que l'approche était très efficace en temps de réponses. Outre la proposition d'une structure mieux adaptée, des optimisations ont été proposées dans [53] de manière à améliorer les temps d'obtention de motifs fréquents.

Du ressort des experts du domaine, l'élaboration de scénarios peut s'avérer, suivant le cas très complexe. Néanmoins, à partir des données collectées, il est possible d'appliquer des techniques d'extraction de connaissances afin de rechercher des traits de comportement intéressants, pouvant aider l'expert à constituer un scénario ou vérifier une hypothèse. L'idée est ainsi de tirer profit des données historiques pour rechercher toute information pertinente. L'historique répertorie en fait des événements, sans tenir compte de liens de causalités ou de synchronisations. En proposant d'intégrer le contexte de multi-occurrence d'événements dans une transaction nous avons proposé l'approche PSP+. La proposition tire pleinement profit de la structure mise en jeux dans PSP et les expérimentations menées sur différents jeux de données ont montré son efficacité par rapport à l'intégration de la multi-occurrence dans une approche comme GSP.

L'élaboration de GTC a permis d'offir une réponse à la problématique de la recherche de motifs séquentiels intégrant des contraintes de temps. Adaptée aux applications nécessitant l'analyse de comportement à court et à long terme par exemple, cette composante offre en outre une solution particulièrement élégante à la prise en compte des contraintes temporelles. Son caractère générique, via le précalcul des contraintes, lui permet de s'adapter aisément à des structures utilisés dans des approches de recherche de motifs existantes. Par exemple, des expérimentations ont été réalisées sur la structure de hachage utilisée dans le cas de l'algorithme GSP.

En proposant une solution, également générique, à la mise à jour des connaissances extraites, l'algorithme ISE complète les composantes précédentes. L'approche proposée permet de répondre à un besoin de plus en plus crucial lors de l'utilisation de techniques de

data mining : comment maintenir la connaissance acquise?. Contrairement aux approches existantes qui nécessitent par exemple de stocker la bordure négative, les informations requises pour extraire la nouvelle connaissance sont minimales dans la mesure où elles se résument aux motifs découverts fréquents lors de la phase d'extraction précédente. Cette solution permet donc d'envisager l'approche incrémentale sur de grandes bases de données. L'utilisation d'ISE non plus comme algorithme incrémental mais en tant qu'approche de recherche de motifs séquentiels semble particulièrement intéressante et, même si des travaux sont indispensables pour connaître le contexte dans lequel la proposition est la plus performante, des expériences ont montrés qu'ISE était une solution efficace pour l'extraction de motifs.

Pour valider les composants définis précédemment, nous nous sommes intéressés à deux types particuliers d'applications. En étudiant le comportement des utilisateurs sur des serveurs Web, nous avons montré que les composants mis en place sont particulièrement adaptés (recherche de comportement dans le temps, recherche de page corrélées, prise en compte des mises à jour du serveur, etc.) et que l'approche de Web usage Mining proposée, en affinant l'analyse du comportement des utilisateurs, est très originale par rapport aux autres travaux du domaine. Recommandée dans le cadre de la détection d'intrusion dans un système, la recherche de régularités trouve dans le cadre de cette composante un nouveau champ d'investigation et une réponse particulièrement adaptée.
L'utilisation et l'adaptation des composants d'extraction de motifs séquentiels dans le contexte de la recherche de régularités dans des bases de données complexes est une approche originale qui offre une solution élégante à un nouveau domaine de recherche.

Les expérimentations menées aux travers de types d'applications variées : de l'analyse des scénarios à la recherche de structures récurrentes dans des schémas issus de pages Web, n'ont fait que renforcer notre opinion sur l'intérêt des motifs séquentiels par rapport aux règles d'association.

Résultats obtenus

Liste des publications

Revue internationale avec comité de lecture

F. Masseglia, P. Poncelet, and R. Cicchetti. "An Efficient Algorithm for Web Usage Mining". In Networking and Information Systems, Vol. 2, N. 5-6, pp. 571-603, December 1999.

Revue internationale invitée

F. Masseglia, P. Poncelet, and M. Teisseire. "Using Data Mining Techniques on Web Access Logs to Dynamically Improve Hypertext Structure". In ACM SigWeb Letters, pp. 13-19, Vol. 8, N. 3, October 1999.

Conférences internationales avec comité de lecture

P.A. Laur, F. Masseglia, P. Poncelet, and M. Teisseire. "A General Architecture for Finding Structural Regularities on the Web". Proceedings of the 9th International Conference on Artificial Intelligence (AIMSA'00), Lecture Notes in Artificial Intelligence, Springer Verlag, Varna, Bulgaria, September 2000.

P.A. Laur, F. Masseglia, and P. Poncelet. "Schema Mining: Finding Structural Regularity among Semistructured Data". Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'2000), Poster session, Lecture Notes in Artificial Intelligence, Springer Verlag, Lyon, France, September 2000.

F. Masseglia, P. Poncelet, and M. Teisseire. "Web Usage Mining: How to Efficiently Manage New Transactions and New Clients". Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'2000), Poster session, Lecture Notes in Artificial Intelligence, Springer Verlag, Lyon, France, September 2000.

F. Masseglia, P. Poncelet, and R. Cicchetti. "WebTool: An Integrated Framework for Data Mining". Proceedings of the 10th International Conference onf Database and Expert Systems Applications (DEXA'99'), Lecture Notes in Computer Science,

Springer Verlag, Vol. 1677, pp. 892-901, Florence, Italy, September 1999.

F. Masseglia, F. Cathala, and P. Poncelet. "The PSP approach for mining sequential patterns". Proceedings of the 2nd European Conference on Principles of Data Mining and Knowledge Discovery in Databases (PKDD'98), Lecture Notes in Artificial Intelligence, Springer Verlag, pp. 176-184, Nantes, France, September 1998.

Conférences nationales avec comité de lecture

F. Masseglia, P. Poncelet and M. Teisseire. "Incremental Mining of Sequential Patterns in Large Databases". Actes des 16ièmes Journées Bases de Données Avancées (BDA'00), Blois, France, Octobre 2000.

F. Masseglia, P. Poncelet et M. Teisseire. "Extraction efficace de motifs séquentiels : le prétraitement des données". Actes des 15ièmes Journées Bases de Données Avancées (BDA'99), pp. 341-360, Bordeaux, France, Octobre 1999.

F. Masseglia, P. Poncelet et R. Cicchetti. "Analyse du comportement des utilisateurs sur le Web". Actes du 17ième Congrés Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID'99), Toulon, France, Juin 1999.

Rapport de contrat

P. Poncelet. "Vers une approche uniforme pour analyser le comportement des utilisateurs d'un serveur Web en langage naturel". Rapport Intermédiaire de contrat CPER.

Autres publications liées à mon activité d'encadrement

P.A. Laur. "Recherche de régularités dans des bases de données d'objets complexes". Rapport de DEA de l'Université de Montpellier II, Juin 2000.

C. Besson, T. Chappe et P. Liénard. "Le Data Mining : la recherche au service d'Internet". Rapport de projet de seconde année d'Ecole d'Ingénieur, Juin 2000.

F. Masseglia. "L'extraction de motifs séquentiels". Rapport de DEA de l'Université de Montpellier II, Juin 1998.

Prototype

Depuis octobre 1998, au sein du projet Data Mining, nous réalisons le développement d'un environnement intégré d'extraction de connaissances assurant le pré-traitement, la fouille de données et le post-traitement. Les différents domaines d'applications privilégiés concernent jusqu'à présent l'analyse de données textuelles (Text Mining), l'analyse de comportement d'utilisateurs sur un serveur Web (Web Mining) et l'analyse de requêtes en

langage naturel ou de données XML et HTML issues du Web. Le système développé en C++ et Java a été expérimenté :

- pour le Web Mining, via les serveurs Web du Laboratoire LIRMM, du cache de la région Languedoc et de l'IUT d'Aix en Provence ;
- pour le Text Mining, via des fichiers pdf issues de recherches sur le Web ;
- pour des requêtes en langage naturel, via des jeux de données réels de la société Albert-inc ;
- pour des données HTML ou XML, via de grandes bases de données (bases de films) issues du Web et des jeux de données réels (requêtes sur l'International Immunogenetics Database) du Laboratoire d'Immuno-Génétique de l'Université de Montpellier II.

Pour valider les expériences sur le Web Mining, un prototype de mise à jour de pages HTML en temps réel a également été réalisé.

Condition de la recherche

Les travaux présentés dans ce chapitre ont été réalisés, depuis novembre 1996, dans le cadre du Projet "Data Mining" dont j'assure la responsabilité au sein de l'équipe "Bases de Données-Systèmes d'Information" du LIRMM.

Le travail sur la partie recherche de motifs séquentiels a donné lieu à un projet de Transfert Technologique de techniques (règles d'association et motifs séquentiels) de fouille de données appliquées à la recherche de requêtes en langage naturel avec la société Albert-Inc (www.albert-inc.com). Ce projet, dont je suis responsable, rentre dans le cadre d'un Contrat de Plan Etat Région depuis décembre 1999. Dans ce contexte, le principe général est d'utiliser des techniques de fouille de données (règles d'association et motifs séquentiels) pour optimiser les résultats de recherches d'information sur le Web (le système Albert est actuellement utilisé dans le moteur de recherche de www.free.fr). En analysant une requête utilisateur par rapport au profil général des utilisateurs, il est possible de corriger de manière automatique les erreurs syntaxiques ou grammaticales et d'optimiser les requêtes d'interrogations de moteurs de recherche de manière à offrir un résultat le plus possible en adéquation avec l'intention de l'utilisateur. Les données sur lesquelles sont appliquées les techniques de fouille de données sont à l'heure actuelle de 2Go par jour, il est donc indispensable de proposer des algorithmes très efficaces en temps de réponse mais également évolutifs de manière à maintenir une base de connaissances constamment mise à jour (fouille de données incrémentale).

Le travail effectué sur les motifs séquentiels appliqués au texte (Text mining) et sur la recherche de structures fréquentes rentre dans le cadre de notre participation, depuis octobre 1997, au projet "Extraction et synthèse de connaissances à partir de données réparties sur le Web" en collaboration avec le CNET (Lannion) et le LIRMM (Montpellier). Dans ce cadre, l'objectif est d'extraire les schémas sous-jacents de différents serveurs afin de les intégrer au sein d'un schéma global et d'offrir la possibilité d'interroger de manière transparente les différents serveurs. Une telle intégration nécessite une phase d'extraction de schéma où des techniques de fouille de données peuvent être appliquées. Cette phase doit également

être suivie par une étape de résolution des différentes incohérences.

Depuis juin 2000, le travail réalisé dans la recherche de structures régulières a permis notre intégration au projet RNTL "CONTEXTE Bourse : CONnaissances Temporelles EXTraites de données BOURSièrEs". L'objectif de ce projet est d'étendre les systèmes d'informations industriels et commerciaux via Internet en proposant un ensemble de composants logiciels pour la découverte de connaissances sur des données temporelles structurées et non-structurées. Ce projet est en collaboration avec le laboratoire PRiSM et les entreprises e-XMLMedia, elseware et Firstinvent.

Depuis juillet 2000, je suis co-responsable avec S. Cerri (Professeur de l'Université Montpellier II) du projet Génération de connaissances pour l'aide à la décision à partir du Web. Dans le cadre de ce projet, l'objectif est de proposer une approche de recherche d'informations et de consultation de connaissances évolutives sur le Web. Elle doit favoriser la création de structures dynamiques d'informations (pages Web et entrepôts de données) et prendre en compte le comportement de l'utilisateur final. Le cadre général de ce projet s'intègre dans l'apprentissage à distance et le commerce électronique.

Bibliographie

[1] S. Abiteboul. Querying Semi-Structured Data. In *Proceedings of International Conference on Database Theory (ICDT'97)*, pages 1–18, Delphi, Greece, January 1997.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J.L Wiener. The Lorel Query Language for Semi-Structured Data. *International Journal on Digital Libraries*, 1(1):68–88, April 1997.

[3] R. Agrawal, C. Faloutsos, and A. Swami. Efficient Similarity Search in Sequence Databases. In *Proocedings of the 4th International Conference on Foundations of Data Organization and Algorithms (FODO'93)*, USA, 1993.

[4] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD Conference*, pages 207–216, Washington DC, USA, May 1993.

[5] R. Agrawal, K. Lin, H.S. Sawhney, and K. Shim. Fast Similarity Search in Presence of Noise, Scaling, and Translation in Time-Series Database. In *Proocedings of the International Conference on Very Large Database*, Switzerland, 1995.

[6] R. Agrawal and R. Srikant. Fast Algorithms for Mining Generalized Association Rules. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB'94)*, Santiago, Chile, September 1994.

[7] R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, Tapei, Taiwan, March 1995.

[8] R. Agrawal and R. Srikant. Parallel Mining of Association Rules. *IEEE Transactions on Knowldege and Data Engineering*, 8(6):962–969, 1996.

[9] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Belmont, 1984.

[10] S. Brin, R. Motwani, J.D. Ullman, and S. Tsur. Dynamic Itemset Counting and Implication Rules for Market Basket Data. In *Proceedings of the International Conference on Management of Data (SIGMOD'97)*, pages 255–264, Tucson, Arizona, May 1997.

[11] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. Adding Structure to Unstructured Data. In *Proceedings of International Conference on Database Theory (ICDT'97)*, pages 336–350, Delphi, Greece, January 1997.

[12] d M. Ranganathan C. Faloutsos a and Y. Manolopoulos. Fast Sequence Matching in Time-Series Database. In *Proocedings of the International Conference on Management of Data (SIGMOD'94)*, 1994.

[13] F. Cathala. Modélisation de Scénarios Comportementaux et Application à la Pathologie des Barrages. Thèse de Doctorat d'Informatique, Université d'Aix-Marseille II, Février 2000.

[14] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In *Proceedings of the IPSJ Conference*, pages 7–18, Tokyo, Japan, October 1994.

[15] S-J. Chen and A.L.P. Chen. An Efficient Approach to Discovering Knowledge from Large Databases. In *Proceedings of the International Conference on Parallel and Distributed Information Systems*, 1996.

[16] D. Cheung, V. Ng, A. Fu, and Y. Fu. Efficient Mining of Association Rules in Distributed Databases. *IEEE Transactions on Knowldege and Data Engineering*, 8(6):911–922, 1996.

[17] D.W. Cheung, B. Kao, and J. Lee. Discovering User Access Patterns on the World-Wide Web. In *Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'97)*, February 1997.

[18] World Wide Web Consortium. httpd-log files. In *http://lists.w3.org/Archives*, 1998.

[19] R. Cooley, B. Mobasher, and J. Srivastava. Web Mining: Information and Pattern Discovery on the World Wide Web. In *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, November 1997.

[20] R.W. Cooley. Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data. In *PHD Thesis*, University of Minnesota, 2000.

[21] L. Dehaspe, H. Toivonen, and R.D. king. Fining Frequent Substructures in Chemical Coumpounds. In *Proocedings of the 4th International Conference on Knowldege Discovery and Data Mining*, pages 30–36, New York, 1998.

[22] C. Dyreson. Using an Incomplete Data Cube as a Summary Data Sieve. *Bulletin of the IEEE Technical Committee on Data Engineering*, pages 19–26, March 1997.

[23] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Database with Noise. In *Proocedings of the 2nd International Conference on Knowledge Dicovery and Data Mining (KDD'96)*, pages 226–231, Portland, USA, 1996.

[24] U.M. Fayad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, 1996.

[25] M. Fernández, D. Florescu, J. Kang, and A. Levy. Catching the Boat with Strudel: Experiences with a Web-Site Management System. *Proceedings of the International Conference on Management of Data (SIGMOD'98) - SIGMOD record*, 27(2):414–425, 1998.

[26] G. Gardarin, P. Pucheral, and F. Wu. Bitmap Based Algorithms For Mining Association Rules. In *Actes des journées Bases de Données Avancées (BDA'98)*, Hammamet, Tunisie, October 1998.

[27] S. Guha, R. Rastori, and K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In *Proocedings of the International Conference on Management of Data (SIGMOD'98)*, pages 73–84, Seattle, USA, 1998.

[28] S. Guha, R. Rastori, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. In *Proocedings of the 1(th International Conference on Data Engineering (ICDE'98)*, Sidney, Australia, 1998.

[29] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo. Extracting Semistructured Information from the Web. In *Proceedings of the Workshop on Management of Semistructured Data. See [14]*, Tucson, Arizona, May 1997.

[30] A.-H. Han, G. Karypis, and V. Kumar. Scalable Parallel Data Mining for Association Rules. In *Proceedings of International Conference on Management of Data (SIGMOD'97)*, 1997.

[31] J. Han and Y. Fu. Discovery of Multiple-Level Association Rules from Large Databases. In *Proceedings of the 21 st International Conference on Very Large Databases (VLDB'95)*, pages 420–431, Zurich, Switzerland, September 1995.

[32] J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaïane. Dmql: A Data Mining Query Language for Relational Databases. In *Proceedings of the SIGMOD'96 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'96)*, Montreal, Canada, 1996.

[33] J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In *Proceedings of International Conference on Management of Data (SIGMOD'2000)*, Dallas, Texas, 2000.

[34] C. Hidber. Online Association Rule Mining. In *Proceedings of the International Conference on Management of Data (SIGMOD'99)*, pages 145–156, USA, May 1999.

[35] M. Houtsma and A. Swami. Set-Oriented Mining of Association Rules. Technical Report RJ 9567, IBM Almaden Research Center, San Jose, California, October 1993.

[36] M. Houtsma and A. Swami. Set-Oriented Mining of Association Rules. In *Proceedings of International Conference on Data Engineering (ICDE'95)*, Taiwan, 1995.

[37] Y. Huhtala, J. Karkkainen, P. Porkka, and H. Toivonen. Efficient Dicsovery of Functional and Approximate Dependencies Using Partitions. In *Proceedings of the 14th International Conference on Data Engineering (ICDE'98)*, USA, 1998.

[38] HyperNews. Httpd log analyzers. In *http://www.hypernews.org/HyperNews/get/www/log-analyzers.html*, 1998.

[39] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[40] M. Kantola, H. Mannila, K.J. Raiha, and H. Sirtola. Discovering Functional and Inclusion Dependencies in Relational Database. *Journal of Intelligence Systems*, pages 591–607, 1992.

[41] C.A. Knoblock, S. Minton, J.L. Ambite, N.Ashish, P.J Modi, I. Musla, A.G. Philpot, and S. Tejada. Modeling Web Sources for Information Integration. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 211–218, Madison,Wisconsin, 1998.

[42] F. Korn, A. Labrinidis, Y. Kotidis, and C. Faloutsos. Ration Rules: A New Paradigm for Fast, Quantifiable Data Mining. In *Proceedings of the 24th International Conference on Very Large Databases (VLDB'98)*, September 1998.

[43] P.A. Laur. Schema Mining: vers une approche efficace. Technical Report (in preparation), LIRMM, France, June 2000.

[44] P.A. Laur, F. Masseglia, and P. Poncelet. Schema Mining: Finding Structural Regularity among Semistructured Data. In *Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'2000)*, Lyon, France, September 2000.

[45] P.A. Laur, F. Masseglia, P. Poncelet, and M. Teisseire. A General Architecture for Finding Structural Regularities on the Web. In *Proceedings of the 9th International Conference on Artificial Intelligence (AIMSA'00)*, Varna, Bulgaria, September 2000.

[46] W. Lee, S.Stolfo, and K. Mok. Mining in a Data-Flow Environment: Experience in Network Intrusion Detection. In *Proceeding of the 5th International Conference on Knowledge Discovery and Data Mining (KDD'99)*, San Diego, CA, August 1999.

[47] H. Lieberman. Letizia: An Agent that Assists Web Browsing. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95)*, 1995.

[48] D.-I. Lin and Z.M. Kedem. Pincer-Search: A New Algorithm for Discovering the Maximum Frequent Set. In *Proceedings of International Conference on Extending Database Technology (EDBT'98)*, Valencia, Spain, 1998.

[49] S. Lopez, J-M. Petit, and L. Lakhal. Discovery of Functional Dependencies and Armstrong Relations. In *Proocedings of the 7th International Conference on Extending Database Technology (EDBT'98)*, Germany, March 2000.

[50] H. Mannila, H. Toivonen, and A.I. Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1(3), September 1997.

[51] H. Mannila, H. Toivonen, and A. I. Verkano. Improved Methods for Finding Association Rules. Technical report, University of Helsinki, Finland, February 1994.

[52] H. Mannila, H. Toivonen, and A. I. Verkano. Discovering Frequent Episodes in Sequences. In *Proceedings of the 1st International Conference on Knowledge Discovery in Databases and Data Mining (KDD'95)*, pages 210–215, Montreal, Canada, August 1995.

[53] F. Masseglia. Le pré-calcul appliqué à l'extraction de sequential patterns en data mining. Mémoire de DEA d'Informatique, LIRMM, France, Juin 1998.

[54] F. Masseglia, F. Cathala, and P. Poncelet. The PSP Approach for Mining Sequential Patterns. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98), LNAI, Vol. 1510*, pages 176–184, Nantes, France, September 1998.

[55] F. Masseglia, P. Poncelet, and R. Cicchetti. An Efficient Algorithm for Web Usage Mining. *Networking and Information Systems Journal*, 2(5-6), December 1999.

[56] F. Masseglia, P. Poncelet, and R. Cicchetti. Analyse du comportement des utilisateurs sur le Web. In *Actes du 17ième Congrès Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID'99)*, Toulon, France, Juin 1999.

[57] F. Masseglia, P. Poncelet, and R. Cicchetti. WebTool: An Integrated Framework for Data Mining. In *Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA'99)*, Florence, Italy, September 1999.

[58] F. Masseglia, P. Poncelet, and M. Teisseire. Extraction efficace de motifs séquentiels : le prétraitement des données. In *Actes des Journées Bases de Données Avancées (BDA'99)*, Bordeaux, France, Octobre 1999.

[59] F. Masseglia, P. Poncelet, and M. Teisseire. Incremental Mining of Sequential Patterns in Large Databases. In *Actes des Journées Bases de Données Avancées (BDA'00)*, Blois, France, Octobre 1999.

[60] F. Masseglia, P. Poncelet, and M. Teisseire. Usinf Data Mining Techniques on Web Access Logs to Dynamically Improve Hypertext Structure. *ACM SigWeb Letters*, 8:13–19, October 1999.

[61] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. LORE: a Database Management System for Semi-Structured Data. *SIGMOD Record*, 26(3), September 1997.

[62] B. Mobasher, N. Jain, E. Han, and J. Srivastava. Web Mining: Pattern Discovery from World Wide Web Transactions. Technical Report TR-96-050, Department of Computer Science, University of Minnesota, 1996.

[63] L. Moreau and N. Gray. A Community of Agents Maintening Link Integrity in the World-Wide Web. In *Proceedings of the 3rd International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'98)*, pages 221–233, London, UK, March 1998.

[64] A. Mueller. Fast Sequential and Parallel Algorithms for Association Rules Mining: A Comparison. Technical Report CS-TR-3515, Department of Computer Science, University of Maryland-College Park, August 1995.

[65] S. Nestorov, S. Abiteboul, and R. Motwani. Extracting Schema from Semistructured Data. *Proceedings of the International Conference on Management of Data (SIGMOD'98) - SIGMOD record*, 27(2), 1998.

[66] S. Nestorov, J. Ullman, J. Wiener, and S. Chawathe. Representative Objects: Concise Representations of Semistructured, Hierarchical Data. In *Proceedings of the 13th International Conference on Data Engineering (ICDE'97)*, pages 79–90, Birmingham, U.K., April 1997.

[67] C. Neuss and J. Vromas. *Applications CGI en Perl pour les Webmasters*. Thomson Publishing, 1996.

[68] R.T. Ng, V.S. Lakshmanan, J. Han, and A. Pang. Exploratory Mining and Pruning Optimizations of Constrained Association Rules. *Proceedings of the International Conference on Management of Data (SIGMOD'98) - SIGMOD record*, 27(2):13–24, 1998.

[69] N. Novelli and R. Cicchetti. Mining Functional and Embedded Dependencies using Free Sets. In *Actes des journées Bases de Données Avancées (BDA'00)*, Blois, France, 2000.

[70] T. Oates, M.D. Schmill, D. Jensen, and P.R. Cohen. A Family of Algorithms for Finding Temporal Structure in Data. In *Proceedings of the 6th International Workshop on AI and Statistics*, 1997.

[71] J.S. Park, M. Chen, and P.S. Yu. An Effective Hash Based Algorithm for Mining Association Rules. In *Proceedings of the international Conference on Management of Data (SIGMOD'95)*, 1995.

[72] S. Parthasarathy, M.J. Zaki, M. Ogihara, and S. Dwarkadas. Incremental and Interactive Sequence Mining. In *Proceedings of the 8th International Conference on Information and Knowledge Management (CIKM'99)*, pages 251–258, Kansas City, MO, USA, November 1999.

[73] N. Pasquier. Data Mining: algorithmes d'extraction et de réduction des règles d'association dans les bases de données. In *Thèse de Doctorat, Université Blaise Pascal, Clermont-Ferrand II*, Janvier 2000.

[74] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient Mining of Association Rules Using Closed Itemset Lattices. *Information Systems*, 19(4):33–54, 1998.

[75] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering Frequent Closed Itemsets for Association Rules. In *Proceedings of International Conference on Database Theory (ICDT'99)*, pages 398–416, Israel, 1999.

[76] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill and Webert: Indentifying Interesting Web Sites. In *Proceedings of the AAAI Spring Symposium on Machine Learning In Information Access*, Portland, Oregon, 1996.

[77] I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences: the teiresias algorithm. *bioinformatic*, 4(1):55–67, 1998.

[78] A. Savasere, E. Omiecinski, and S. Navathe. An Efficient Algorithm for Mining Association Rules in Large Databases. In *Proceedings of the 21 st International Conference on Very Large Databases (VLDB'95)*, pages 432–444, Zurich, Switzerland, September 1995.

[79] J.C. Shafer, R. Agrawal, and M. Mehta. SPRINT: A Scalable Parallel Classifier for Data Mining. In *Proocedings of the 22th International Conference on Very Large Databases (VLDB'96)*, Bombay, India, September 1996.

[80] R. Srikant. Fast Algorithms for Mining Association Rules and Sequential Patterns. Technical Report PHD Dissertation, University of Wisconsin - Madison, 1996.

[81] R. Srikant and R. Agrawal. Mining Generalized Association Rules. In *Proceedings of the 21 st International Conference on Very Large Databases (VLDB'95)*, pages 407–419, Zurich, Switzerland, September 1995.

[82] R. Srikant and R. Agrawal. Mining Quantitative Association Rules in Large Relational Tables. In *Proceedings of the 1996 ACM SIGMOD Conference*, Montreal, Canada, June 1996.

[83] R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proceedings of the 5th International Conference on Extending Database Technology (EDBT'96)*, pages 3–17, Avignon, France, September 1996.

[84] R. Srikant, Q. Vu, and R. Agrawal. Mining Association Rules with Item Constraints. In *Proceedings of the 3rd International Conference on Knowledge Discovery in Databases and Data Mining*, pages 67–73, September 1997.

[85] R. Taouil. Algorithmique du treillis des fermés : application à l'analyse formelle de concepts et aux bases de données. In *Thèse de Doctorat, Université Blaise Pascal, Clermont-Ferrand II*, Janvier 2000.

[86] H. Toivonen. Sampling Large Databases for Association Rules. In *Proceedings of the 22nd International Conference on Very Large Databases (VLDB'96)*, September 1996.

[87] J.T. Wang, G-W Chirn, T. Marr, B. Shapiro, D. Shasha, and K. Zhang. Combinatorial Pattern Discovery for Scientific data: Some Preliminary Results. In *Proceedings of the 1994 ACM SIGMOD Conference on Management of Data*, Minneapolis, May 1994.

[88] K. Wang, Y. He, and J. Han. Mining Frequent Itemsets Using Support Constraints. In *Proceedings of International Conference on Very Large Database (VLDB'2000)*, Cairo, Egypt, 2000.

[89] K. Wang and H. Liu. Discovering Structural Association of Semistructured Data. *IEEE Transactions on Knowledge and Data Engineering*, 1999.

[90] K. Wang and H.Q. Liu. Mining Nested Association Patterns. In *Proceedings of SIGMOD'97 Workshop on Research Issues on Data Mining and Knowledge Discovery*, May 1997.

[91] K. Wang and H.Q. Liu. Discovering Typical Structures of Documents: A Road Map Approach. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 146–154, Melbourne, Austrialia, August 1998.

[92] S.M. Weiss and C.A. Kulikowski. *Computer System that Lean: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufman, 1991.

[93] Work97. The Workshop on Management of Semistructured Data. In *www.research.att.com/šuciu/workshop-papers.html*, Tucson, Arizona, May 1997.

[94] F. Wu and G. Gardarin. Gradual Clustering Algorithms for Metric Spaces. In *Actes des 15ièmes Journées Bases de Données Avancées (BDA'99)*, Bordeaux, France, 1999.

[95] O. Zaïane, M .Xin, and J. Han. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. In *Proceedings on Advances in Digital Libraries Conference (ADL'98)*, Santa Barbara, CA, April 1998.

[96] M. Zaki. Scalable Data Mining for Rules. Technical Report PHD Dissertation, University of Rochester - New York, 1998.

[97] M.J. Zaki. Efficient Enumeration of Frequent Sequences. In *Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM'98)*, 1998.

[98] M.J. Zaki, M. Ogihara, S. Parthasarathy, and W. Li. Parallel Data Mining for Association Rules on Shared-memory Multi-processors. In *Supercomputing'96*, 1996.

[99] M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New Algorithms for Fast Discovery of Association Rules. In *Proceedings of the 3rd International Conference on Knowldege Discovery and Data Mining*, 1997.

[100] M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New Parallel Algorithms for Fast Discovery of Association Rules. *Data Mining and Knowledge Discovery*, 1(4):343–373, 1997.

Troisième partie

Perspectives

Perspectives

Les perspectives envisagées à l'heure actuelle sont beaucoup plus liées à l'extraction de connaissances qu'à la conception de bases de données avancées. Il existe actuellement un très grand nombre de perspectives liés à la problématique de recherche de motifs séquentiels : recherche de règles multi-relation, recherche de règles négatives ou quantitatives, intégration de méta-connaissance du domaine lors du processus d'extraction, optimisation des algorithmes de recherches dans le cadre de données fortement corrélées, etc. En outre, associés aux travaux menés, de nombreuses problématiques restent ouvertes. Par exemple dans le cadre des travaux sur la régularités dans les bases de données d'objets complexes, nous avons considéré que les cycles n'étaient pas pris en compte. Cette contrainte est malheureusement très forte et il suffit d'examiner les données semi-structurées issues du Web pour constater leur importance. De la même manière, une extension originale de l'approche est également d'offrir une aide à la spécification de DTD en XML.

Nous présentons dans les paragraphes suivants quelques perspectives plus générale qui font directement suite aux différents travaux réalisés et qui rentrent dans le cadre des différents projets menés actuellement. Les trois premières perspectives concernent principalement le domaine de la fouille de données alors que la dernière perspective, qui s'intéresse à la mise à jour de schéma, prend en compte aussi bien les travaux menés dans le cadre de la conception que de la fouille de données.

Vers une meilleure gestion des candidats

Il existe des situations dans lesquelles les techniques mises en œuvre par l'approche "générer-élaguer" ne sont plus efficaces. Nous pouvons citer par exemple les longues séquences, un trop grand nombre de motifs fréquents ou encore une recherche avec un support minimal faible. En effet, en utilisant une approche générer-élaguer classique et quelque soit la structure envisagée, nous nous trouvons en présence d'un trop grand nombre de séquence ou d'itemsets à tester sur la base et un stockage intermédiaire sur le disque dur est indispensable. Considérons par exemple, 10^4 fréquents de taille 1, l'approche générer-élaguer classique nécessitera de générer plus de 10^7 candidats qui seront stockés soit à l'aide d'une structure efficace en mémoire centrale soit, au moins pour une partie d'entre eux, sur le disque dur. La recherche de fréquents de taille 2 est alors une phase très lente qui ralentit considérablement le processus d'extraction. La situation est similaire lors de la recherche de motifs de grande taille comme nous l'avons déjà signalé dans la discussion du chapitre 3. Par exemple, pour générer des itemsets fréquents de taille 100, les auteurs de [5] ont montré qu'il fallait générer 10^{30} candidats. Ainsi, la recherche de motifs nécessitera au pire, i.e. sans aucune optimisation, 100 passes sur la base mais en plus le processus sera ralenti

par la gestion des candidats.

Les extensions proposées aux algorithmes de recherche de motifs séquentiels ou de règles d'association ont permis de minimiser le temps soit en utilisant des structures mieux adaptées (comme PSP par exemple), soit en partitionnant la base (Partition, DIC ou Sampling). Même si un partitionnement ou une division de la base est tout en fait envisageable dans le cas de la recherche de motifs séquentiels, il n'en reste pas moins que la problématique liée à la présence d'un trop grand nombre de candidats reste présente.

Pour offrir une meilleure gestion des candidats, J. Han et al. [5] ont proposé une nouvelle structure de données basée sur une structure d'arbre préfixée (FP-Tree) qui n'est plus utilisée pour stocker les candidats mais plutôt pour représenter la base de données sous une forme compacte. Après deux parcours de la base, une structure d'arbre est créée et stocke les "chemins communs" avec leur support associé. En outre, pour chaque item, un tableau indique sa position dans le chemin. L'extraction se fait à l'aide d'une méthode récursive qui parcours la structure et détermine quels sont les chemins qui permettent d'arriver à cet item. Les itemsets fréquents sont alors obtenus en fonction du support associé au chemin.

Une autre perspective possible pour répondre à cette problématique est l'utilisation de chaînes de Markov. En considérant la base de données il est possible de déterminer une matrice de transition permettant de connaître pour chaque item de la base quelle est la probabilité de passer d'un état à un autre, i.e. quelle est la probabilité en connaissant un item d'obtenir l'autre item. Cependant, même si cette approche semble intéressante, elle ne permet pas de connaître quelle est la position d'un item dans une transaction. Cette contrainte est forcément limitative dans la mesure où dans le contexte d'extraction de connaissances, nous nous intéressons à la recherche de motifs ou d'itemsets consécutifs. Une solution plus adaptée consiste à utiliser les chaînes de Markov cachées qui offre la possibilité de modéliser des séquences d'observation, i.e. des enchaînements de transactions dans notre contexte. En modélisant la probabilité d'enchaînement entre les items ainsi que la probabilité liée au positionnement entre items, un tel type d'approche est tout à fait adapté à la problématique dans la mesure où l'automate probabiliste obtenu lors de l'analyse de la base reflète, non seulement, le contenu de la base mais aussi la possibilité d'extraire des itemsets fréquents sans nécessiter de génération de candidats. En effet, l'automate probabiliste ne nécessite de connaître que les différents états de la base, i.e. les différents items, et les différentes probabilités de transitions d'un état à un autre, i.e. les valeurs de probabilités d'un item à un autre. Un mécanisme d'extraction basé sur un tel automate offre la possibilité d'éviter la phase de génération de candidats et offre ainsi une solution adaptée.

Nous avons donc commencé à nous intéresser à la définition d'une approche de recherche d'itemsets basée sur les chaînes de Markov cachées. Dans ce contexte, une première étude a été menée et a montré qu'elle était particulièrement bien adaptée lors de la prise en compte d'itemsets de grandes tailles ou lors d'un support minimal faible. Les principes généraux de l'approche proposée sont les suivants. Lors d'une première passe sur la base, l'automate de transition d'états est généré et ceci de manière indépendante du support. Lors de la définition du support, les items non fréquents sont éliminés et les items fréquents sont triés de manière croissante. A partir de ces derniers, l'ensemble des transactions de la base est

réécrit de manière à ne conserver que les itemsets éventuellement fréquents. Une comparaison de la nouvelle base en éliminant les items fréquents dont le support est le plus faible et de l'automate permet d'obtenir l'ensemble des fréquents ainsi que le support associé via l'automate. Le processus se répète tant qu'il existe des items fréquents. Le premier avantage d'une telle approche est qu'elle est tout à fait indépendante de la phase classique de génération des candidats des techniques générer-élager. Le second avantage est lié à la génération de l'automate. En effet, il est tout à fait possible de définir celui-ci sans tenir compte de la notion de support en considérant éventuellement que celui-ci peut être stocké sur disque. En proposant cette génération, le processus général d'extraction de connaissance peut alors être optimisé dans la mesure où, pour connaître quels sont les itemsets fréquents, il ne suffit que de modifier la structure pointant vers l'automate. L'extraction des itemsets fréquents ne consiste alors qu'à parcourir l'automate en fonction des itemsets fréquents obtenus lors de la spécification du support. Enfin, un dernier avantage est celui de la prise en compte des mises à jours sur la base qui ne consiste alors qu'à modifier les états ou les transitions entre états dans l'automate. Les premieres expériences menées dans ce contexte ont montré que l'approche retenue était particulièrement bien adaptée lors de la prise en comtpe d'itemsets de grandes tailles. Cependant, associée à cette approche, un certains nombres de perspectives existent. Quid des motifs séquentiels ? Même si l'automate est bien adapté à la notion de changement de transactions, qu'en est-il de la notion de changement de date ? Il semble, aux travers des premières expériences, que l'approche proposée permet de répondre de manière efficace à la problématique, i.e. la recherche de motifs séquentiels de grande taille ou l'utilisation d'un support faible. Il existe cependant des applications dans lesquelles nous pouvons trouver à la fois des itemsets de petites et de grandes tailles, une perspective particulièrement intéressante est de déterminer une approche qui soit suffisamment générale pour rechercher efficacement les itemsets quelle que soit leur taille.

Vers une approche efficace pour le Text Mining

L'application de techniques de recherche de motifs séquentiels sur des documents a montré qu'il était envisageable de déterminer quels sont les éléments d'un texte qui sont fortement corrélés. En outre, la prise en compte des contraintes de temps adaptées aux composants d'un texte (chapitre, section, paragraphe, mot, etc.) offre de nouvelles possibilités d'analyse de texte sur la globalité des documents. Par exemple, la contrainte de minGap peut être utilisée pour préciser que nous ne sommes intéressés que par des régularités entre phrases, alors que la contrainte de windowSize va permettre de regrouper les phrases entre elles. Les expériences menées sur des pages Web extraites ainsi que sur un très grand nombre de fichiers pdf ont montré que les résultats obtenus pouvaient permettre par exemple de classifier les textes en fonction des occurrences des termes. Une autre application possible de ces analyses est la possibilité de rechercher des tendances dans des documents et a été proposée dans [7].

Le principe général suivi pour l'analyse de fichier texte repose tout d'abord sur un prétraitement particulier des données. En effet, lors de ce pré-traitement il est indispensable de supprimer les termes jugés non significatifs comme les articles tout en tenant compte de leur position dans le texte. La gestion d'un thésaurus est également nécessaire pour faciliter

la découverte de corrélation. Même si les motifs séquentiels sont adaptés ils sont malheureusement limités à une analyse syntaxique des documents et une perspective intéressante est alors de compléter cette analyse par une analyse sémantique. Une collaboration avec l'équipe Langage Naturel du Lirmm débute sur cette perspective. Le principe général est d'étendre la recherche de motifs à la prise en compte de vecteurs sémantiques associés aux paragraphe ou à d'autres composants du document. En combinant ces deux aspects nous espérons être capable de déterminer non seulement les occurrences répétées mais également les éventuelles répétitions de parties de documents de sémantiques proches.
Intégrer une composante vecteur sémantique aux motifs séquentiels nécessite cependant de prendre en compte pour chaque item participant, i.e. pour chaque mot du document, quel est son vecteur sémantique le plus proche. Ce paramètre complique alors l'extraction des motifs fréquents dans la mesure où les éléments à analyser ne sont pas de même type.

Associée à la sémantique du document, une composante complémentaire d'analyse peut être envisagée en considérant les travaux que nous avons menés sur la recherche de régularités de structures complexes. En se basant sur une analyse de la structure syntaxique du document et en recherchant au travers des documents les structures analogues nous pouvons alors obtenir une information supplémentaire qu'il semble intéressant de combiner avec les résultats de l'analyse précédente de manière à affiner la phase de recherche de connaissances.

La définition de cette composante efficace de recherche de connaissances dans des documents entre dans le cadre du projet RNTL "CONTEXTE Bourse". En effet, au travers d'une analyse fine des différentes dépêches, l'idée principale de la composante Text Mining du projet consiste à extraire toute information pertinente permettant d'évaluer les conséquences que peut avoir une ou plusieurs dépêches sur les cours de la bourse.

Vers une temporalité accrue des résultats

Les résultats obtenus par les algorithmes d'extraction classiques ne sont pas suffisamment précis si l'on souhaite analyser de manière détaillée le comportement des utilisateurs ou des clients au cours du temps. Par exemple, il est possible de connaître en analysant une base de données de transactions que *62% des clients achètent un plat cuisiné puis un livre de recettes*. De la même manière, en tenant compte des contraintes de temps, nous pouvons déterminer que *60 % des clients qui ont visité* `/jdk1.1.6/docs/api/Package-java.io.html` *et* `/jdk1.1.6/docs/api/java.io.BufferedWriter.html`*, ont également visité, dans les 30 jours suivants, l'URL* `/jdk1.1.6/docs/relnotes/deprecatedlist.html` (C.f. Chapitre 2). Même si ces informations sont pertinentes elles ne précisent pas la répartition dans le temps des clients qui participent aux support. Il est beaucoup plus intéressant de savoir que sur les 62% des clients, *3% des clients achètent un plat cuisiné puis un livre de recettes dans les 3 jours et 59% des clients achètent un plat cuisiné puis un livre de recettes au moins 20 jours après*. En effet, en utilisant uniquement le support minimal nous ne pouvons déterminer qu'un comportement très général. Une première solution triviale serait dans notre exemple de spécifier un minGap de 20 jours. Malheureusement cette solution a plusieurs inconvénients majeurs : (i) les 3% des clients ne vérifiant pas la contrainte de minGap ne sont pas extraits et si le support minimal a été spécifié à 62%, nous n'obtenons

pas non plus les 59% des clients de la base ; (ii) La spécification des contraintes de temps est délicate et nécessite généralement une première extraction qui pourra être affiné par la suite.

Nous avons mené une première étude qui a montré qu'il était possible de générer à l'aide d'une nouvelle passe sur la base les différentes répartitions possibles des clients par rapport aux motifs fréquents trouvés. Le principe général est de stocker à la fois le motif séquentiel et un arbre préfixé. Lors du parcours sur la base de données, si la séquence du client est incluse dans le motif séquentiel, celui-ci est inséré dans l'arbre préfixé en précisant les délais entre chaque itemset ainsi que le nombre de client vérifiant ce motif. Si une nouvelle séquence est incluse dans le motif mais avec un délai entre itemset différent, celui-ci est inséré via un nouveau lien pour préciser qu'il s'agit d'un autre délai et le compteur associé au nouveau nœud cible de ce lien est incrémenté. Un simple parcours des arbres générés offre alors la possibilité d'offrir à l'utilisateur la répartition des différents clients par rapport aux temps séparant les itemsets. Si cette approche semble bien adaptée, l'une des conséquences immédiate est la génération d'un trop grand nombre de répartitions rendant inexploitables les résultats de l'extraction. Dans notre exemple, une telle approche aurait pour conséquence de répartir les 59% des clients de la base en autant de délais entre itemsets supérieurs à 20.
Une étude plus approfondie est donc indispensable pour générer les intervalles de temps les plus significatifs éventuellement en les intégrant directement dans l'algorithme de recherche de motifs. Les travaux menés dans le cadre de la recherche de règles d'associations quantitatives [11], même s'ils sont définis dans un autre contexte, pourrait être un point de départ à la définition de tels intervalles. En outre, adaptés aux données de la base, les principes issus d'une telle recherche semblent offrir une solution plus générale à la définition d'une approche de recherche de motifs séquentiels quantitatifs. En effet, les algorithmes de règles quantitatives ne pouvant pas être adaptés aux motifs une approche spécifique est indispensable [10] et il n'existe pas aujourd'hui de solution à la recherche de tels motifs qui reste toujours un domaine de recherche ouvert [10, 15].

Vers des mises à jour de schéma

Lié aux études menées dans le cadre de la conception et de l'extraction de connaissances, il serait intéressant de voir si les techniques mises en œuvre sont adaptées au contexte de mises à jour de schéma des sources d'information dans un entrepôt de données.
De manière générale, dans l'architecture d'un entrepôt, le schéma de la source doit être obtenu et incorporé dans le moniteur et dans l'adaptateur. Cependant, dès que le schéma évolue, ces composants doivent aussi être modifiés de manière à répercuter les modifications au sein de l'entrepôt. Ainsi, pour prendre en compte les modifications de schémas des sources dans l'entrepôt il est nécessaire d'obtenir le nouveau schéma et manuellement de modifier les composants de l'entrepôt de manière à refléter les modifications. Cette approche peut être envisagée dans le cas de modifications peu fréquentes mais n'est plus réaliste dans le cas de changements fréquents. Ainsi, dans [2], les auteurs précisent qu'il y a en moyenne 2 à 3 changements des schémas sources par an et, dans [9], une étude a montré que dans le cas d'un système de gestion d'hopital, il y a eu une augmentation de 139% des relations et de 274% d'attributs et que chaque relation dans le schéma a été changée au

moins une fois au cours de 19 mois de l'étude. Tenir compte des modifications des sources d'information est un problème relativement récent [13] et plusieurs propositions ont été faites pour prendre en compte les modifications des données selon les types de sources. Nous pouvons citer par exemple :

- les techniques basées sur les règles actives qui peuvent déterminer automatiquement des changements d'états ;
- la maintenance d'un fichier log qui peut être interrogé et ainsi les changements intéressants peuvent être automatiquement appliqués à l'entrepôt ;
- la comparaison de "clichées" des sources d'information avec des "clichés" de l'entrepôt pour rechercher des changements de valeurs.

Par contre, il n'existe que très peu de travaux concernant la prise en compte à la fois des données et des schémas sources. Ainsi dans [6], les auteurs proposent une approche de gestion des modifications via une base de méta-connaissance de données et une gestion des mises à jours dans les méta-données. Basée sur la notion de vues, le principe consiste à définir via une extension de SQL les préférences d'évolution des vues au niveau des métaconnaissances (quelles sont les informations indispensables, quelle information peut être remplacée par une autre similaire mais venant d'une autre sources, etc). A l'aide de ces informations une vue peut être réécrite en une vue non-équivalente mais qui respecte quand même la sémantique précisée par l'utilisateur. Lors de l'arrivée d'une mise à jour des sources, le système recherche alors les alternatives possibles de réécriture de la vue en utilisant les composants disponibles au niveau méta-connaissance. Un module supplémentaire permet de choisir via un modèle de coût la meilleure solution possible. Enfin les mises à jour sont reportées sur l'entrepôt. La limite principale de cette proposition est que toutes les sources doivent être spécifiées en relationnel et dans le cas de sources non structurées cette approche est difficile à mettre en œuvre.

De manière générale, repérer des modifications dans un schéma requiert de faire une comparaison entre deux versions du même document. Aussi récemment, il y a eu un certain nombre d'approches pour détecter les changements dans des documents semi-structurés. Cependant, la plupart des approches diffèrent principalement de la manière dont ils représentent les documents. Ainsi, par exemple, dans [4] et [12], les vues semi-structurées des documents sont représentés par des arbres. Pour réaliser la comparaison entre les schémas, les documents sont d'abord convertis en arbre et les nœuds similaires sont recherchés. Toutes les séquences d'opérations (insertion, suppression, modification, déplacement) qui transforment l'ancien arbre en nouvel arbre sont alors recherchées et les séquences de transformation minimale sont conservées. Dans [3], les auteurs utilisent un graphe OEM pour représenter les documents et un modèle général, DOEM, pour représenter les changement dans les données semi-structurées. Afin de déterminer quels sont les changements opérés sur un document, un langage de requête, Chorel, est proposé. Même si cette approche est intéressante car elle peut s'adapter à d'autres types de représentation, le fait de modéliser les documents via un graphe OEM nécessite d'évaluer en détail le contenu des documents semi-structurés afin de rechercher les modifications apportées [1]. En outre, la prise en compte d'un réordonnancement des objets du graphe n'est pas pris en compte dans l'approche proposée.
Dans [1], une nouvelle approche est proposée pour détecter les changements et ne nécessite pas de nouvelle transformation des documents. Au cours de l'examen d'un document par

rapport au schéma déterminé lors des étapes précédentes, il est possible de déterminer automatiquement quelles sont les modifications apportées dans les données d'un document. En outre, les auteurs proposent une approche efficace permettant de déterminer les changements opérés lors des modifications d'un schéma source. Cependant, à partir du moment où des modifications sont opérées sur le schéma d'origine quid du schéma descriptif du document ? Les auteurs préconisent d'utiliser des techniques de fouilles de données, proches de celles que nous avons définis pour la recherche de régularités d'objets complexes afin de rechercher le schéma le plus générique par rapport aux différents documents. L'avantage d'une telle approche est alors de faciliter la génération automatique d'un adaptateur qui est un problème actuel de recherche.
Il semble que la structure mise en œuvre dans le cas de l'extraction de motifs séquentiels pourrait également servir de représentation du document et ainsi, les algorithmes efficaces de détection d'une séquence dans la structure pourrait servir de base à une comparaison d'un document par rapport au schéma. Etant donnée l'importance, à l'heure actuelle, des données semi-structurées, il semble indispensable de mener une étude plus approfondie dans ce domaine en tenant compte des résultats que nous avons précédemment obtenus.

Bibliographie

[1] N. Adam, I. Adiwijaya, T. Critchlow, and R. Musick. Detecting Data and Schema Changes in Scientific Documents. Proceedings of the IEEE Advances in Digital Libraries Conference, May 2000.

[2] N.R. Adam, V. Atluri, and I. Adiwijaya. SI in Digital Libraries. *Communications of the ACM*, 43(6):64–72, 2000.

[3] S. Chawathe, S. Abiteboul, and J. Widom. Representing and Querying Changes in Semistructured Data. pages 4–13, Proceedings of the International Conference on Data Engineering, February 1998.

[4] S. Chawathe, A. Rajaraman, H. Garcia-Molina, and J. Widom. Change Detection in Hierarchically Structured Information. In *Proceedings of International Conference on Management of Data (SIGMOD'96)*, pages 493–504, 1996.

[5] J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In *Proceedings of International Conference on Management of Data (SIGMOD'2000)*, Dallas, Texas, 2000.

[6] A. Koeller and E.A. Rudensteiner. A History-Driven Approach at Evolving Views Under Meta Data Changes. Technical Report WPI-CS-TR-00-01, Worcester Polytechnic Institute, Massachusetts, January 2000.

[7] B. Lent, R. Agrawal, and R. Srikant. Discovering Trends in Text Databases. In *Proceedings of the 3rd Int'l Conference on Knowledge*, Newport Beach, California, August 1997.

[8] P. Loucopoulos and R. Zicari. *Conceptual Modeling, Databases and CASE: An Integrated View of Information Systems Development.* Wiley Professional Computing, 1992.

[9] E.A. Rudensteiner, A. Koeller, and W. Zhang. Maintaining Data Warehouses over Changing Information Sources. *Communications of the ACM*, 43(6):57–62, 2000.

[10] R. Srikant. Fast Algorithms for Mining Association Rules and Sequential Patterns. Technical Report PHD Dissertation, University of Wisconsin - Madison, 1996.

[11] R. Srikant and R. Agrawal. Mining Quantitative Association Rules in Large Relational Tables. In *Proceedings of the 1996 ACM SIGMOD Conference*, Montreal, Canada, June 1996.

[12] J. Wang, K. Zhang, K. Jeong, and D. Shasha. A System for Approximate Tree Matching. *IEEE Transaction on Knowledge and Data Engineering*, 6:559–570, August 1994.

[13] J. Widom. Research Problems in Data Warehousing. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 25–30, November 1995.

[14] Work97. The Workshop on Management of Semistructured Data. In *www.research.att.com/šuciu/workshop-papers.html*, Tucson, Arizona, May 1997.

[15] M. Zaki. Scalable Data Mining for Rules. Technical Report PHD Dissertation, University of Rochester - New York, 1998.