Conclusion

Dans notre étude, nous avons traité trois thèmes de recherche encore très peu abordés :

- la représentation d'indices pour le RàPC dans le cadre d'une classe de problèmes,
- la modélisation d'outils ouverts pour le RàPC,
- la réutilisation de comportements pour l'assistance à la navigation sur le Web.

Tout d'abord, nous présentons notre contribution pour chaque thème en faisant le bilan de la démarche suivie. Nous ébauchons alors les trois principales perspectives qui montrent la portée de nos travaux.

1 Bilan

1.1 Modèle d'indexation par situations comportementales

Durant toute notre étude, nous avons été motivés par une classe de problèmes dont la principale caractéristique est de nécessiter la prise en considération d'historiques dans la formulation du problème à résoudre. Nous avons alors étudié le problème de l'indexation qui est grandement affecté par cette spécificité au sein du cycle de raisonnement à partir de cas. Nous avons présenté dans le chapitre II, l'état de l'art des approches d'indexation par historiques et nous avons proposé le *modèle d'indexation par situations comportementales*. Aucun des travaux existants n'aborde de manière générale l'indexation par historiques dans le cadre d'un cycle de raisonnement à partir de cas. Notre modèle d'indexation isole cette problématique en deux parties : modèle de représentation des indices et des cas, guide d'utilisation. Dans ce modèle, nous offrons une modélisation générique pour faciliter l'analyse du RàPC dans le cadre de cette classe de problèmes, et pour constituer une compétence métier nécessaire pour aborder des problèmes complexes.

Notre modèle d'indexation permet également de dépasser les deux principales limites des approches existantes afin de permettre une meilleure indexation des cas dans la classe de problèmes visée. Premièrement, nous offrons la possibilité d'exprimer des indices à la fois sur des historiques échantillonnés et à événements. Cette possibilité est importante d'une part pour gérer des processus hybrides, et d'autre part, pour représenter des indices calculés sous forme d'événements, alors que d'autres variables restent échantillonnées par manque de connaissances.

Deuxièmement, nous proposons des solutions originales pour permettre une meilleure gestion du processus de découverte des connaissances. Ce processus est important dans notre classe de problèmes car il n'est pas possible de représenter de manière explicite tous les cas utiles issus des observations, étant donné le manque de connaissances et la quantité importante des données disponibles. L'introduction du concept de cas potentiel permet alors d'extraire des cas à partir des données d'observation. Les cas potentiels sont définis en compréhension au travers de patrons qui peuvent être instanciés. Ainsi, contrairement aux travaux existants, nous permettons l'utilisation d'hypothèses préétablies, représentées dans les patrons de cas potentiels, pour structurer cette extraction. Puis, pour pallier le manque d'apprentissage de la plupart des travaux existants, nous permettons de sauver ces instanciations sous forme de cas concrets. La nature explicite des cas concrets permet de faire évoluer leur représentation en fonction des raisonnements : par exemple, association d'informations sur leur utilisation ou modification de leurs indices. Les indices des cas concrets peuvent en effet être ajustés au cours des raisonnements par la sélection de comportements élémentaires et par l'expression de contraintes temporelles. Nous introduisons ainsi une

représentation qui permet l'expression de situations simples ou complexes, proches de scénarios temporels. Enfin, les cas concrets référencent toujours les données d'observation structurées en variables et en enregistrements. Cette séparation en deux niveaux permet la création de nombreux cas et leur modification, tout en facilitant leur interprétation.

1.2 Plate-forme à objets pour le RàPC

Dans les chapitres III et IV, nous avons détaillé notre approche qui a mené à la réalisation d'un outil ouvert permettant de faciliter l'utilisation du modèle d'indexation par situations comportementales, que nous avons proposé. Plus précisément, cet outil permet la spécialisation du modèle, son intégration cohérente dans un système complet de RàPC, et la capitalisation d'objets-métier. Aucun des outils existants en RàPC ne présente une ouverture comparable. Ces outils se limitent en effet à des boîtes noires dans lesquelles uniquement certains aspects sont modifiables, et ne facilitent pas l'intégration de nouveaux composants. C'est pourquoi nous avons proposé une nouvelle approche mettant en œuvre le concept de *plate-forme à objets*. La plate-forme à objets réalisée repose sur une architecture abstraite de collaboration entre les concepts du RàPC et de notre modèle d'indexation (index, similarité, base de cas, indice, situation comportementale). Cette architecture intègre un ensemble de *points d'ouverture* qui peuvent être configurés pour obtenir les comportements adaptés à une application particulière. La conception d'une telle architecture est réputée difficile pour garantir l'objectif de réutilisabilité systématique. Nous proposons alors une structuration des points d'ouverture identifiés en trois axes de variabilité et en deux niveaux de spécificité (RàPC et indexation par situations comportementales). Cette structuration est alors détaillée sous la forme de *cas d'utilisation*. Nous proposons également des modèles à objets ouverts utilisant des *patrons de conception*. Les patrons de conception présentent des solutions typiques à des problèmes récurrents de conception de logiciels à objets, et permettent de justifier les nombreuses décisions de conception.

Sur un plan théorique, nous apportons ainsi une modélisation à objets flexible du RàPC et de l'indexation par situations comportementales, pour faciliter la réalisation de différents systèmes. Sur un plan plus pratique, nous avons réalisé la plate-forme à objets CBR*Tools dans le langage à objets Java. CBR*Tools a été utilisée dans la réalisation d'une application classique en RàPC (cf. annexe C), dans un prototype de recherche des cas pour la nutrition des plantes, et dans Broadway. Cette plate-forme offre un ensemble d'objets-métier et permet l'intégration de nouveaux composants que l'on peut assembler : indice, index, similarité, phase de raisonnement… L'utilisation de cet outil est alors facilitée par la structuration des points d'ouverture en cas d'utilisation et par l'utilisation du vocabulaire formé par les patrons de conception utilisés. Sur le plan du génie logiciel, nous donnons l'exemple de la conception d'une plate-forme modélisant les fondements du raisonnement à partir de cas. La recherche d'architectures ouvertes et réutilisables est une démarche d'actualité, étant donné le besoin de rendre interopérables les logiciels pour assembler des composants spécialisés provenant d'origines diverses. Toutefois, la conception de plates-formes à objets est une tâche difficile pour laquelle il n'existe pas encore de méthode dédiée. Cependant, nous montrons concrètement comment la notation UML ainsi que les concepts de cas d'utilisation et de patrons de conception sont utilisés à la fois pour une meilleure conception mais aussi pour une aide à l'utilisation.

1.3 Broadway : un nouveau type d'assistants à la navigation sur le Web

Dans le chapitre V, nous avons premièrement proposé une nouvelle approche d'assistance à la navigation dans un hypermédia. Nous proposons en effet d'aider un utilisateur en réutilisant, à partir d'un groupe d'utilisateurs, des *comportements observés par variables*. Cette observation par

variables, combinée avec notre modèle d'indexation, permet la gestion de comportements détaillés dans un cadre flexible et générique. Le prototype Broadway est un exemple de cette approche et permet le calcul de recommandations en fonction d'observations sur quatre variables (adresse et contenu des pages visitées, évaluations explicite et implicite). Aucun des assistants de navigation existants fondés sur les comportements des utilisateurs ne prend en compte un ensemble de variables, ni ne permet l'ajout et le traitement unifié de nouvelles variables. Notre modèle d'indexation et notre plate-forme à objets sont ainsi mis à profit dans notre approche.

Deuxièmement, la réalisation concrète de Broadway ainsi que son évaluation expérimentale montrent la faisabilité et l'intérêt de notre approche. Broadway effectue un cycle de raisonnement à partir de cas en suivant les quatre phases classiques : recherche, réutilisation, révision et apprentissage. A partir des navigations observées, Broadway extrait des cas utiles grâce aux cas potentiels et permet de les sauvegarder sous la forme de cas concrets. Un cas représente une association issue d'une navigation, entre un comportement et un ensemble de pages proposées. A partir de la navigation courante d'un utilisateur, Broadway identifie tout d'abord un ensemble de cas qui présentent des comportements similaires. La réutilisation de ces cas consiste alors à faire la synthèse des pages proposées pour former la liste des recommandations présentées à l'utilisateur. Puis, l'utilisateur révise la solution suggérée au cours de sa navigation en évaluant, à son tour, les pages visitées. Enfin, Broadway apprend de cette interaction une nouvelle navigation et met à jour son indexation.

Broadway est également le résultat d'un travail important pour la réalisation des modules annexes au moteur de raisonnement à partir de cas. En effet, la construction d'un prototype réaliste a principalement nécessité : l'intégration de l'interface graphique dans un navigateur Web, la répartition des fonctions en huit processus communiquant par RMI (*Remote Method Invocation*), la spécialisation d'un proxy HTTP (serveur Jigsaw du W3C), et l'analyse de pages au format HTML pour l'extraction des mots clefs.

2 Perspectives

A partir de nos travaux, nous avons mis en évidence trois principales perspectives portant sur :

- l'extension et la généralisation de notre modèle d'indexation,
- l'intégration des caractéristiques d'une plate-forme à objets dans des méthodes de gestion de projet RàPC,
- la conception de systèmes d'aide basés sur notre approche de réutilisation de comportements utilisateurs.

2.1 Vers l'extension et la généralisation de notre modèle d'indexation

Nous avons appliqué notre modèle d'indexation dans le cadre de la régulation de la nutrition des plantes et celui de l'assistance à la navigation sur le Web. Pour poursuivre l'évaluation du modèle, d'autres applications sont envisageables dans cette classe de problèmes. Par exemple, une étude a déjà été menée pour l'aide à l'argumentation sur Internet dans un contexte de décision collective (Karacapilidis *et al.*, 1997). Il serait de plus intéressant d'étudier l'utilisation de notre modèle pour des systèmes existants comme REBECAS pour les feux de forêts ou RADIX pour la recherche d'information.

Aux deux types de cas proposés (concrets et potentiels), il serait également intéressant de combiner des *cas abstraits* (Bergman & Wilke, 1996) et de définir des techniques pour gérer le cycle

de vie des cas : extraction de cas potentiels, transformation en cas concrets et abstraits, élimination des cas. Notamment, des mécanismes de création de cas abstraits permettraient de généraliser et de remplacer les cas concrets : la taille de la base de cas serait alors plus réduite. Des méthodes d'abstraction de chroniques (Shahar, 1997) et de modification automatique des indices des cas sont alors nécessaires. Dans le cadre plus précis de la navigation sur le Web, des approches de recherche de séquences récurrentes de pages (Chen & Park, 1998 ; Schechter *et al.*, 1998) sont envisageables en s'appuyant sur les pages sélectionnées comme indices dans les cas concrets.

Enfin, avec notre modèle d'indexation, nous rejoignons les préoccupations du modèle d'indexation UIF adapté à une autre classe de problèmes. Selon Janet Kolodner (Kolodner, 1993, pages 219–220), cette démarche est prometteuse et permettra à terme de réutiliser des cas issus de domaines différents puisqu'ils utilisent une description basée sur les mêmes principes. Cette réutilisation pourra alors mener à de meilleurs raisonnements et à l'introduction d'un facteur de créativité. Nous pensons également que notre effort doit être poursuivi sur d'autres modèles d'indexation pour aboutir à une méthodologie visant à faciliter la définition de tels modèles. Le modèle d'indexation par situations comportementales et sa structuration présentent alors un point de départ novateur. Notre approche met également en valeur l'importance de l'indexation au sein du cycle de raisonnement à partir de cas, et nous pensons que les principes suivants, proposés dans notre modèle, pourraient être généralisés à d'autres classes de problèmes :

- séparation des données brutes et des cas dont la représentation est dépendante de la tâche du système et doit nécessairement intégrer des informations supplémentaires (sélection des indices, poids d'importance, qualité du cas).
- extraction de cas potentiels guidée par les raisonnements et leur sauvegarde sous la forme de cas concrets pour modifier leurs indices et leurs informations.

2.2 Vers des méthodes d'utilisation d'objets-métier pour le RàPC

Notre approche basée sur le concept de plate-forme à objets est une première étape vers la constitution d'un ensemble de modèles et d'objets-métier, voire des patrons-métier, facilitant la conception de systèmes de RàPC. De manière plus générale, nous pensons que ce type d'outils devrait permettre d'améliorer le processus de mise en place de systèmes RàPC. Tout d'abord, une plate-forme offre déjà un ensemble de classes abstraites et peut ainsi permettre de faciliter le passage des modèles au niveau connaissance jusqu'au niveau symbolique. Puis, une plate-forme amène des atouts qui pourraient être utilisés dans la gestion d'un projet de RàPC : prototypage rapide grâce à la réutilisation directe des composants de base, bonne stabilité aux modifications des exigences et configuration suivant les besoins des points d'ouverture. En définitive, un tel outil permet de concentrer les efforts sur les points importants et spécifiques à une application : identification des indices, acquisition des cas et des connaissances nécessaires à l'organisation de la mémoire, à l'adaptation et à l'apprentissage. C'est pourquoi l'étude de méthodes de gestion de projet RàPC (Bergmann *et al.*, 1997 ; Bergmann *et al.*, 1998) devrait s'enrichir d'une telle approche.

Puis, nous pensons améliorer notre plate-forme à objets en enrichissant les composants que cet outil contient. L'ajout de composants (schémas de phase de raisonnement, index) serait intéressant, comme par exemple : index par MOP (*Memory Organisation Package*, Kolodner, 1983 ; Schank, 1982) permettant d'indexer les cas dans une structure discriminante évolutive, index spécifiques pour les situations comportementales prenant en compte des techniques de mise en correspondance optimisées des historiques (Faloutsos *et al.*, 1994), schémas de recherche et d'adaptation par classification hiérarchique (Lieber, 1997). Le couplage effectif avec l'outil TEMPOS permettrait finalement de stocker les historiques dans une base de données à objets. Enfin, il s'agit de poursuivre notre effort pour la description de nos composants dans le modèle standard des JavaBeans (Beans,

1997). Ce modèle permet un couplage faible entre les objets (communication par événements) et amène ainsi une plus grande flexibilité. De plus, les composants peuvent être manipulés au sein d'environnements graphiques évolués, comme dans l'outil VisualAge[62] de IBM. Ces composants peuvent alors être combinés avec d'autres objets qui ne sont pas liés au RàPC (base de données, interface graphique, autres objets-métier).

2.3 Vers des assistants basés sur les comportements

Une première évaluation de Broadway a été menée dans un cadre précis et montre l'intérêt de notre approche. Toutefois, les expérimentations doivent continuer pour généraliser les résultats obtenus. Dans ce but, l'ensemble des paramètres identifiés (cf. annexe D) ainsi que les outils construits (Broadway*Tools) sont des apports importants. Nous avons également commencé l'étude de l'intégration de robots de recherche dans notre architecture. Ces robots peuvent remplacer totalement ou partiellement les utilisateurs dans leurs navigations sur le Web pour leur éviter des parcours fastidieux. La réutilisation des navigations se ferait alors au niveau des robots (Golinelli, 1998 ; Taheri, 1998). Notre approche de coopération indirecte peut également être enrichie de mécanismes de coopération directe dans le cadre d'une tâche de recherche d'information (Trousse *et al.,* 1998), et nécessite une gestion des notions de session et de profil utilisateur (Elhaddad, 1998).

Plus généralement, nous sommes à l'étude de deux applications importantes de notre approche visant des systèmes d'aide basés sur la réutilisation de comportements observés sur différentes variables :

- Dans le cadre d'une collaboration entre l'INRIA et le centre de recherche européen de Xerox (Meylan, France), nous étudions la réutilisation de comportements utilisateurs alliant des étapes d'exécution de requêtes dans un moteur de recherche, et des étapes de navigation sur le Web. Les objectifs sont de guider l'utilisateur dans son processus de reformulation de requête et de lui proposer des documents ou des bases documentaires à consulter en se basant sur des comportements passés.
- Dans le cadre d'un contrat de recherche avec le CNET (centre de recherche de France Télécom), nous visons l'assistance à un important groupe d'utilisateurs au sein d'un hypermédia délimité.

Dans ces deux applications, il s'agit d'identifier les variables d'observation des utilisateurs et de construire, grâce à notre modèle d'indexation et à CBR*Tools, des systèmes d'aides adaptés. Enfin, nous pensons que notre approche serait intéressante dans le cadre du commerce électronique pour guider des consommateurs vers les produits souhaités mais difficiles à exprimer *a priori* (configuration d'un micro-ordinateur par exemple).

[62] http://www.software.ibm.com/ad/vajava/.

Références bibliographiques

1 Liste des références

(Aamodt & Plaza, 1994) A. Aamodt et E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and Systems. *AI Communications*, vol. 7, n°1, pages 36–59, 1994.

(Allen, 1984) J. F. Allen. Towards a General Theory of Action and Time. *Artificial Intelligence*, vol. 23, n°2, pages 123–154, 1984.

(Althoff *et al.,* 1995) K. D. Althoff, E. Auriol, R. Barletta et M. Manago. *A Review of Industrial Case-Based Reasoning Tools*. AI Perspectives Report, AI Intelligence, 148 pages, 1995.

(Andreoli *et al.,* 1996) J. M. Andreoli, U. Borghoff et R. Pareschi. Constraint-Based Knowledge Broker Model: Semantics, Implementation and Analysis. *Journal of Symbolic Computation*, vol. 21, n°4, pages 635–667, 1996.

(Armstrong *et al.,* 1997) R. Armstrong, T. Joachims et T. Mitchell. WebWatcher: A Tour Guide for the World Wide Web. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'97),* pages 770–775, Morgan Kaufmann, 1997.

(Auriol *et al.,* 1994) E. Auriol, M. Manago, K. D. Althoff, S. Wess et S. Dittrich. Integrating Induction and Case-Based Reasoning: Methodological Approach and First Evaluations. In *Proceedings of the 2nd European Workshop on CBR (EWCBR'94)*, pages 145–155, AcknoSoft press, 1994.

(Bachiochi *et al.,* 1997) D. Bachiochi *et al*. Usability Studies and Designing Navigational Aids for the World Wide Web. In *Proceedings of the 6th International World Wide Web Conference (WWW6)*, volume 29 de *Computer Network and ISDN Systems*, pages 1489–1496, Elsevier, 1997.

(Baille *et al.,* 1990) A. Baille, T. Boulard et C. Gary. Les critères d'optimisation dans la gestion du climat et de la production sous serre. *Séances de l'Académie d'Agriculture*, vol. 76, n°5, 12 pages, 1990.

(Baille, 1991) A. Baille. L'aide à la decision dans la gestion du climat sous serre : de nouvelles perspectives. *P. H. M. Revue Horticole*, n°315, pages 51–57, 1991.

(Baird-Smith *et al.,* 1997) A. Baird-Smith, Y. Lafon et B. Mahé. Jigsaw: An Object Oriented Server. W3C, http://www.w3.org/Jigsaw/User/Introduction/wp.html, 1997.

(Barletta, 1994) R. Barletta. A Hybrid Indexing and Retrieval Strategy for Advisor CBR Systems Built with REMIND. In *Proceedings of the 2nd European Workshop on CBR (EWCBR'94)*, pages 49–58, AcknoSoft press, 1994.

(Barrett *et al.*, 1997) R. Barrett, P. P. Maglio et D. C. Kellem. How to Personalize the Web. In *Proceedings of the Conference on Human-computer Interaction (CHI'97)*, pages 75–82, Addison-Wesley, 1997.

(Bassaville, 1988) M. Basseville. *Distance Measures for Signal Processing and Pattern Recognition*. Rapport de recherche n°899, INRIA, 47 pages, 1988.

(Beans, 1997) *Java Beans*. Sun Microsystems, http://java.sun.beans, 114 pages, 1997.

(Bergman & Wilke, 1996) R. Bergmann, W. Wilke. On the Role of Abstraction in Case-Based Reasoning. In *Advances in Case-Based Reasoning, Proceedings of the 3rd European Workshop on Case-Based Reasoning (EWCBR'96)*, volume 1168 de *Lecture Notes in Artificial Intelligence*, pages 28–43, Springer, 1996.

(Bergmann *et al.,* 1997) R. Bergmann, W. Wilke et J. Schumacher. Using Software Process Modeling for Building a Case-Based Reasoning Methodology: Basic Approach and Case Study. In *Case-Based Reasoning Research and Development, Proceedings of the 2nd International Conference on Case-Based Reasoning (ICCBR'97)*, volume 1266 de *Lecture Notes in Artificial Intelligence*, pages 509–518, Springer, 1997.

(Bergmann *et al.*, 1998). R. Bergmann *et al*. Collecting Experience on the Systematic Development of CBR Applications Using the INRECA Methodology. In *Proceedings of 4th European Workshop on Case-Based Reasoning (EWCBR'98)*, volume 1488 de *Lecture Notes in Artificial Intelligence*, pages 460–470, Springer, 1998.

(Berners-Lee *et al.,* 1994) T.B. Lee, R. Cailliau, A. Luotonen, H. F. Nielsen et A. Secret. The World Wide Web. *Communications of the ACM*, vol. 37, n°8, pages 76–82, 1994.

(Bird & Wadler, 1988) R. Bird et P. Wadler. *Introduction to Functional Programming*. Prentice Hall, 293 pages 1988.

(Box & Jenkins, 1976) G. E. P. Box et G. M. Jenkins. *Time Series Analysis: Forcasting and Control*. Holden Day, 575 pages, 1976.

(Brandau *et al.,* 1991) R. Brandau, A. Lemmon, et C. Lafond. Experience with Extended Episodes: Cases with Complex Temporal Structure. In *Proceedings of the 3rd DARPA Workshop on Case-Based Reasoning*, pages 1–12, Morgan Kaufmann, 1991.

(Bray, 1996) T. Bray. Measuring the Web. In *Proceedings of the 5th International World Wide Web Conference,* volume 28 de *Computer Network and ISDN Systems*, pages 993–1005, Elsevier, 1996.

(Breiman *et al.,* 1984) L. Breiman, J. H. Freidman, R. A. Olshen et C. J. Stone. *Classification and Regression Tree*. Wadsworth and Brooks, 358 pages, 1984.

(Brooks *et al.,* 1995) C. Brooks, M. S. Mazer, S. Meeks et J. Miller. Application-Specific Proxy Servers as HTTP Stream Transducers. In *Proceedings of the 4th International World Wide Web Conference*, numéro 1 de *World wide web journal*, pages 539–548, 1995.

(Brown *et al.,* 1995) M. Brown, I. Watson et N. Filer. Separating the Cases from the Data: Towards More Flexible Case-Based Reasoning. In *Case-Based Reasoning Research and Development, Proceedings of the 1st International Conference on Case-Based Reasoning (ICCBR'95)*, volume 1010 de *Lecture Notes in Artificial Intelligence*, pages 157–168, Springer, 1995.

(Brun & Settenbrino, 1993) R. Brun et A. Settenbrino. Roses hors-sol sous serre : gestion de la nutrition minérale et hydrique. *Revue horticole*, nº345, pages 25–29, 1993.

(Brun *et al.,* 1993) R. Brun, B. Paris et I. Hammelin. Fertigation Management of Rose Plants Grown in Greenhouse on Rockwool. *Advanced Horticulture Science*, vol. 7, pages 109–111, 1993.

(Bull *et al.,* 1997) M. Bull, G. Kundt et L. Gierl. Case-Based Risk Detection and Forecasting in a Geographic-Medical System. In *Proceedings of the 5th German Workshop on Case-Based Reasoning (GWCBR'97)*, rapport LSA-97-01E, Centre for Learning Systems and Applications (LSA), Université de Kaiserslautern, pages 59–64, 1997.

(Burke *et al*., 1997) D. Burke, K. J. Hammond et B. C. Young. The FindMe Approach to Assisted Browsing. *IEEE Expert*, vol. 12, n°4, pages 32–39, 1997

(Bushmann *et al.,* 1996) F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad et M. Stal. *Pattern-Oriented Software Architecture*: *A System of Patterns*. Wiley & Sons, 457 pages, 1996.

(Canavaggio, 1997) J. F. Canavaggio. *TEMPOS : un modèle d'historiques pour un SGBD temporel*. Thèse de Doctorat, Université Joseph Fourier, Grenoble, 176 pages, 1997.

(Carbonell, 1986) J. G. Carbonell. Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. *Machine Learning*, vol. 2, chapitre 14, pages 317–392, 1986.

(Charles, 1975) G. Charles. *Reconnaissance automatique de forme d'un signal transitoire rapide*. Thèse de Doctorat, Université de Paris sud, 130 pages, 1975.

(Chartron, 1996) G. Chartron. Recherche d'information sur Internet. In *la recherche d'information sur les réseaux*, cours INRIA, pages 43–101, ADBS editions, 1996.

(Chen & Park, 1998) M-S. Chen et J. S. Park. Efficient Data Mining for Path Traversal Patterns. *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, n°2, pages 209–221, 1988.

(Chen & Sycara, 1998) L. Chen et K. Sycara. WebMate: A Personal Agent for Browsing and Searching. In *Proceedings of of the 2nd International Conference on Autonomous Agents*, pages 132–139, ACM press, 1998.

(Chiriatti & Plant, 1993) K. C. Chiriatti et R. E. Plant. Case-Based Reasoning: Application to the Agricultural Domain, a Prototype. In *Proceedings of the 1st European Workshop on CBR (EWCBR'93)*, rapport SEKI SR-93-12, pages 174–177, Université de Kaiserslautern, 1993.

(Codenie *et al.,* 1997) W. Codenie, K. De Hondt, P. Steyaert et A. Vercammen. From Custom Applications to Domain-Specific Frameworks. *Communications of the ACM*, vol. 40, n°10, pages 71–77, 1997.

(Corvaisier *et al.,* 1997) F. Corvaisier, A. Mille et J.M. Pinon. Information Retrieval on the World Wide Web Using a Decison Making System. In *Actes de la 5ème conférence sur la Recherche d'Information Assistée par Ordinateur sur Internet (RIAO'97)*, Centre des hautes études internationales d'Informatique, Montréal, pages 285–295, 1997.

(Corvaisier *et al.,* 1998) F. Corvaisier, A. Mille et JM. Pinon. Recherche Assistée de documents Indexés sur l'eXpérience (RADIX) : mesures de similarités des épisodes de recherche sur le Web. In *Actes de Ingénierie des Connaissances (IC'98)*, pages 153–163, INRIA, 1998.

(Cutler *et al.,* 1997) M. Cutler, Y. Shih et W. Meng. Using the Structure of HTML Documents to Improve Retrieval. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (ITS'97)*, pages 241–252, USENIX Association, 1997.

(Demeyer *et al.,* 1997) S. Demeyer, T.D. Meijler, O. Nierstrasz et P. Steyaert. Design Guidelines for 'Tailorable' Framework. *Communications of the ACM*, vol. 40, n°10, pages 60–64, 1997.

(Deter, 1994) R. Deter. CBR for Maintenance of Telecommunication Network. In *Proceedings of the 2nd European Workshop on Case-Based Reasoning (EWCBR'94)*, pages 23–31, AcknoSoft press, 1994.

(Dillon *et al.,* 1990) A. Dillon, C. McKnight et J. Richardson. Navigation in Hypertext: A critical Review of the Concept. In *Proceedings of the 13rd International Conference on Human-Computer Interaction (INTERACT'90)*, pages 587–592, Elsevier, 1990.

(Dousson & Ghallab, 1994) C. Dousson et M. Ghallab. Suivi et reconnaissance de chroniques. *Revue d'Intelligence Artificielle*, vol. 8, n°1, pages 29–61, 1994.

(Dreyer *et al.,* 1994) W. Dreyer, A. K. Dittrich et D. Schimdt. Research Perspectives for Time Series Management Systems. *SIGMOD Record*, vol. 23, n°1, pages 10–15, 1994.

(Elhaddad, 1998) M. Elhaddad. *Conception et réalisation d'un serveur de profils utilisateurs, utilisation dans Broadway*. Mémoire de projet de fin d'études, Ecole Nationale Supérieure d'Informatique et d'Analyse des Systemes, Maroc, et INRIA Sophia Antipolis, Action AID, 63 pages, 1998.

(Elkassar & Charlet, 1997) S. Elkassar et J. Charlet. Représentation de connaissances et aide à la navigation hypertextuelle à partir de cas : application au dossier médical. In *Actes de la journée Ingénierie des Connaissances et Apprentissage Automatique (JICAA'97)*, pages 387–401, INRIA, 1997.

(Faloutsos *et al.,* 1994) C. Faloutsos, M. Ranganathan et Y. Manolopoulos. Fast Subsequence Matching in Time-Series Database. *SIGMOD Record*, vol. 23, n°1, pages 419–429, 1994.

(Faurre & Robin, 1986) P. Faurre et M. Robin. *Elément d'automatique*. Dunod, 443 pages, 1986.

(Fauvet *et al.,* 1997) M. C. Fauvet, J. F. Canavaggio et P. C. Scholl. TEMPOS : un modèle d'historiques pour un SGBD temporel à objets. In *Actes des 13eme journées Bases de Données Avancées (BDA'97)*, IMAG, pages 67–87, 1997.

(Fayad & Schmidt, 1997) M. E. Fayad et D. C. Schmidt. Object-Oriented Application Frameworks. *Communications of the ACM*, vol. 40, No 10, pages 32–38, 1997.

(Fontaine, 1996) D. Fontaine. Une approche par graphes pour la reconnaissance de scénarios temporels. *Revue d'Intelligence Artificielle*, vol. 10, n°4, pages 439–468, 1996.

(Foss, 1989) C. L. Foss. *Detecting Lost Users: Empirical Studies on Browsing Hypertext*. Rapport de Recherche n°972, INRIA, 39 pages, 1989.

(Froedlish *et al.,* 1997) G. Froedlish, H. J. Hoover, L. Liu et P. Sorenson. Hooking into Object-Oriented Application Frameworks. In *Proceedings of International Conference on Software Engineering (ICSE'97)*, pages 491–501, IEEE Computer Society press, 1997.

(Fuchs *et al.,* 1995) B. Fuchs, A. Mille et B. Chiron. Operator Decision Aiding by Adaptation of Supervision Strategies. In *Case-Based Reasoning Research and Development, Proceedings of the 1st International Conference on Case-Based Reasoning (ICCBR'95),* volume 1010 de *Lecture Notes in Artificial Intelligence*, pages 23–32, Springer, 1995.

(Fuchs, 1997) B. Fuchs. *Représentation des connaissances pour le raisonnement à partir de cas – Le système ROCADE*. Thèse de Doctorat, Université Jean Monnet de Saint-Etienne, 196 pages, 1997.

(Fynn *et al.,* 1989) R. P. Fynn, W. L. Roller et H. M. Keener. A Decision Model for Nutrition Management in Controlled Environment Argiculture. *Agricultural Systems*, n°31, pages 35–53, 1989.

(Gamma *et al.,* 1995) E. Gamma, R. Helm, R. Johnson et J. Vlissides. *Design Patterns : Elements of Reusable Object Oriented Software*. Addison-Wesley, 395 pages, 1995.

(Gebhardt *et al.,* 1997) F. Gebhardt, A. Voss, W. Grather et B. Schmidt-Belz. *Reasoning with Complex Cases*. Kluwer Academic Publisher, 250 pages, 1997.

(Geib *et al.,* 1997) J. M. Geib, C. Gransart et P. Merle. *CORBA, des concepts à la pratique*. InterEditions, 293 pages, 1997.

(Ghallab & Mounir-Alaoui, 1989) M. Ghallab et A. Mounir-Alaoui. Managing Efficiently Temporal Relations through Indexed Spanning Tree. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI'89)*, pages 1297–1303, Morgan Kaufmann, 1989.

(Golinelli, 1997) M. Golinelli. Etude d'une implémentation objet dans CBR*Tools de l'algorithme de recherche des cas similaires à base d'arbre k-d appliqué au raisonnement par cas. Rapport interne, INRIA, Action AID, 35 pages, 1997.

(Golinelli, 1998) M. Golinelli. *Persistance et robots d'évaluation pour Broadway*. Rapport de projet de DESS Informatique, Université de Nice-Sophia Antipolis et INRIA Sophia Antipolis, Action AID, 30 pages, 1998.

(Gonzalez, 1978) R.C. Gonzalez et M.G. Thomason. *Syntactic Pattern Recognition*. Addison-Wesley, 377 pages, 1978.

(Haimowitz & Kohane, 1993a) I. J. Haimowitz et I. S. Kohane. Automated Trend Detection with Temporal Hypotheses. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, pages 146–151, Morgan Kaufmann, 1993.

(Haimowitz & Kohane, 1993b) I. J. Haimowitz et I. S. Kohane. An Epistemology for Clinically Significant Trends. In *Proceedings of the AAAI National Conference on Artificial Intelligence (AAAI'93)*, pages 176–181, MIT press, 1993.

(Hall & Dowling, 1980) P. A. V. Hall et G. R. Dowling. Approximate String Matching. *Computing Survey*, vol. 12, n°4, pages 381–402, 1980.

(Hallam-Baker & Connolly, 1996) P. M. Hallam-Baker et D. Connolly. *Session Identification URI*. Document de travail du W3C, WD-session-id-960221, http://www.w3.org/pub/WWW/TR/WD-session-id.html, 1996.

(Hammond, 1986) K. Hammond. CHEF: A Model of Case-Based Planning. In *Proceedings of the AAAI National Conference on Artificial Intelligence (AAAI'86)*, pages 267–271, Morgan Kaufmann, 1986.

(Harmon, 1992) P. Harmon. Case-Based Reasoning, Part III: A Review of the CBR Products that are Currently being Marketed. *Intelligent Software Strategies*, vol. III, n°1, pages 1–12, 1992.

(Haton *et al.,* 1991) J.P. Haton *et al*. Raisonnement distribué et modèle de tableau noir. *Le raisonnement en Intelligence Artificielle*, chapitre 10, pages 387–429, InterEditions, 1991.

(Haton *et al.,* 1991) J.P. Haton *et al*. Raisonnement par analogie. *Le raisonnement en Intelligence Artificielle*, chapitre 9, pages 365–385, InterEditions, 1991.

(Hébraud *et al.,* 1998) C. Hébraud, M. Rech et A. Thébault. *Test de l'utilité de l'outil d'aide à la navigation Broadway*. Rapport interne, Université de Provence, UFR de psychologie, Aix en Provence, et INRIA Sophia Antipolis, Action AID, 19 pages, 1998.

(Hennessy & Hinkle, 1992) D. Hennessy et D. Hinkle. Applying Case-Based Reasoning to Autoclave Loading. *IEEE Experts*, vol. 7, n°5, pages 21–26, 1992.

(Hinrichs, 1989) T.R. Hinrichs. Strategies for Adaptation and Recovery in a Design Problem Solver. In *Proceedings of the 2nd DARPA Workshop on Case-Based Reasoning*, pages 115–118, Morgan Kaufmann, 1989.

(Huitema, 1995) C. Huitema. *Et Dieu créa l'Internet...* Eyrolles, 201 pages, 1995.

(Iyengar, 1997) A. Iyengar. Dynamic Argument Embedding: Preserving State on the World Wide Web. *IEEE Internet Computing*, pages 50–56, 1997.

(Jaczynski & Trousse, 1994) M. Jaczynski et B. Trousse. Fuzzy Logic for the Retrieval Step of a Case-Based Reasoner. In *Proceedings of the 2nd European Workshop on Case-Based Reasoning* (*EWCBR'94)*, pages 313–321, AcknoSoft press, 1994.

(Jaczynski & Trousse, 1997a) M. Jaczynski et B. Trousse. *CBR*Tools: une bibliothèque objet pour l'indexation des cas par situation comportementale*. Rapport de recherche n°3215, INRIA, 38 pages, 1997.

(Jaczynski & Trousse, 1997b) M. Jaczynski et B. Trousse. Broadway: a World Wide Web Advisor Reusing Past Navigations from a Group of Users. In *Proceedings of 3rd UK Workshop on Case-Based Reasoning (UKCBR3)*, 17 pages, Université de Manchester, 1997.

(Jaczynski & Trousse, 1998a) M. Jaczynski et B. Trousse. An Object-Oriented Framework for the Design and the Implementation of Case-Based Reasoners. In *Proceedings of 6th German Workshop on Case-Based Reasoning (GWCBR'98)*, IMIB Series vol. 7, pages 33–42, Université de Rostock, 1998.

(Jaczynski & Trousse, 1998b) M. Jaczynski et B. Trousse. WWW Assisted Browsing by Reusing Past Navigations from a Group of Users. In *Proceedings of 4th European Workshop on Case-Based Reasoning (EWCBR'98)*, volume 1488 de *Lecture Notes in Artificial Intelligence*, pages 160–171, Springer, 1998.

(Jaczynski & Trousse, 1998c) M. Jaczynski et B. Trousse. Broadway, a Case-Based Browsing Advisor for the Web. In *Research and Advanced Technology for Digital Libraries, Proceedings of the 2nd European Conference on Digital Libraries (ECDL'98)- poster*, volume 1513 de *Lecture Notes in Computer Science*, pages 697–698, Springer, 1998.

(Jaczynski & Trousse, 1999) M. Jaczynski et B. Trousse. Patrons de conception dans la modélisation d'une plate-forme à objets pour le raisonnement à partir de cas. *L'objet*, numéro spécial : les Patterns, vol. 5, n°2, 27 pages, Hermes, 1999. A paraître.

(Jaczynski, 1993) M. Jaczynski. *Recherche des cas similaires et utilisation des ensembles flous*. Rapport de DEA, Université de Nice Sophia-Antipolis et INRIA Sophia-Antipolis, Projet SECOIA, 123 pages, 1993.

(Jaczynski, 1997) M. Jaczynski. A Framework for the Management of Past Experiences with Time-Extended Situations. In *Proceedings of the 6th International Conference on Information and Knowledge Management (CIKM'97)*, pages 32–39, ACM press, 1997.

(Jakob & Suslenschi, 1990) F. Jakob et P. Suslenschi. Situation Assessment for Process Control. *IEEE Expert*, vol. 5, n°2, pages 49–59, 1990.

(Jerke *et al.,* 1990) K-H. Jerke, P. Szabo, A Lesh et H. Rossler. Combining Hypermedia Browsing with Formal Queries. In *Proceedings of the 13rd International Conference on Human-Computer Interaction (Interact'90)*, pages 593–598, Elsevier, 1990.

(Johnson & Foote, 1988) R. E. Johnson et B. Foote. Designing Reusable Classes. *Journal of Object-Oriented Programmin*g, vol. 1, n°2, pages 22–35, 1988.

(Johnson, 1992) R. E. Johnson. Documenting Frameworks using Patterns. In *Proceedings of Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'92)*, volume 27 de *ACM SIGPLAN Notices*, n°10, pages 63–76, 1992.

(Johnson, 1997) R. E. Johnson. Frameworks = (Components + Patterns). *Communications of the ACM*, vol. 40, n°10, pages 39–42, 1997.

(Karacapilidis *et al.,* 1997) N.I. Karacapilidis, B. Trousse et D. Papadias. Using Case-Based Reasoning for Argumentation with Multiple Viewpoints. In *Case-Based Reasoning Research and Development, Proceedings of the 2nd International Conference on Case-Based Reasoning (ICCBR'97)*, volume 1266 de *Lecture Notes in Artificial Intelligence*, pages 541–552, Springer, 1997.

(Keravnou, 1995) E. T. Keravnou. Modelling Medical Concepts as Time-Objects. In *Artificial Intelligence in Medicine*, volume 934 de *Lecture Notes in Artificial Intelligence*, pages 67–90, Springer-Verlag, 1995.

(Kolodner & Leake, 1996) J. L. Kolodner et D. B. Leake. A Tutorial Introduction to Case-Based Reasoning. *Case-Based Reasoning: Experiences, Lessons & Future Directions*, AAAI press, MIT press, pages 31–65, 1996.

(Kolodner, 1983) J. Kolodner. Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science*, vol. 7, n°4, pages 243–280, 1983.

(Kolodner, 1992) J. Kolodner. An Introduction to Case-Based Reasoning. *Artificial Intelligence Review*, vol. 6, n°1, pages 3–34, 1992.

(Kolodner, 1993) J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 668 pages, 1993.

(Kolodner, 1996) J. L. Kolodner. Making the Implicit Explicit: Clarifying the Principles of Case-Based Reasoning. *Case-Based Reasoning: Experiences, Lessons & Future Directions*, AAAI press, MIT press, pages 349–370, 1996.

(Kopeikina *et al.,* 1988) L. Kopeikina, R. Brandau, et A. Lemmon. Case-Based Reasoning for Continous Control. In *Proceedings of the 1st DARPA Workshop on Case-Based Reasoning*, pages 250–259, Morgan Kaufmann, 1988.

(Korfhage, 1997) R. R Korfhage. *Information Storage and Retrieval*. Wiley & Sons, 349 pages, 1997.

(Koton, 1989) P. Koton. *Using Experience in Learning and Problem Solving*. PhD thesis, Departement of Computer Science, MIT, 1989.

(Kurata, 1988) K. Kurata. Greenhouse Control by Machine Learning. *Acta Horticulturae*, n°230, pages 195–200, 1988.

(Lajoie & Keller, 1994) R. Lajoie et R. K. Keller. Design and Resuse in Object-Oriented Framework: Patterns, Contracts, and Motifs in Concert. In *Actes du 62ème Congrès de l'Association Française pour l'avancement des sciences*, 1994.

(Landau, 1988) I. D. Landau. *Identification et commande des systèmes*. Hermes, 306 pages, 1988.

(Lassila & Swick, 1998) O. Lassila et R. R. Swick. *Resource Description Framework (RDF) Model and Syntax*. Document de travail du W3C, WD-rdf-syntax-19980216, http://www.w3.org/TR/ WD-rdf-syntax, 1998.

(Leake *et al.,* 1997) D. B. Leake, A. Kinley et D. Wilson. Case-Based Similarity Assessment: Estimating Adaptability from Experience. In *Proceedings of the 14th AAAI National Conference on Artificial Intelligence (AAAI'97)*, pages 674–679, AAAI press, 1997.

(Leake, 1996) D. B. Leake. CBR in Context: The Present and Future. *Case-Based Reasoning: Experiences, Lessons & Future Directions*, AAAI press, MIT press, pages 2–30, 1996.

(Lieber, 1997) J. Lieber. *Raisonnement à partir de cas et classification hiérarchique – Application à la planification de synthèse en chimie organique*. Thèse de Doctorat, Université Henri Poincaré, Nancy 1, 163 pages, 1997.

(Lieberman, 1995) H. Lieberman. Letizia: An Agent that Assists Web Browsing. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 924–929, Morgan Kaufmann, 1995.

(Lissandre, 1990) M. Lissandre. *Maîtriser SADT*. Armand Colin, 219 pages, 1990.

(Luotonen & Altis, 1994) A. Luotonen et K. Altis. World-Wide Web Proxies. In *Proceedings of the 1st International World Wide Web Conference*, volume 27 de *Computer Network and ISDN Systems*, pages 147–154, Elsevier, 1994.

(Manago *et al.,* 1994) M. Manago *et al*. *CASUEL : A Common Case Representation Language*. Délivrable D1, tâche 1.1, projet Esprit 6322, 36 pages, 1994.

(Mark *et al.,* 1996) W. Mark, E. Simoudis et D. Hinkle. Case-Based Reasoning: Expectations and Results. *Case-Based Reasoning: Experiences, Lessons & Future Directions*, AAAI press, MIT press, pages 269–294, 1996.

(Mattsson & Bosch, 1997) M. Mattsson et J. Bosch. Framework Composition: Problems, Causes and Solutions. In *Proceedings of the 23rd International Conference in Technology of Object-Oriented Languages and Systems (TOOLS '97 USA)*, 1997.

(Meeks *et al.,* 1996) W. S Meeks, C. Brooks et M. S. Mazer. Transducers and Associates: Circumventing Limitations on the World Wide Web. In *Proceedings of the 1st Conference on Emerging Technologies and Applications in Communications (etaCOM'96)*, 1996.

(Meusel *et al.,* 1997) M. Meusel, K. Czarnecki et W. Köpf. A Model for Structuring User Documentation of Object-Oriented Frameworks Using Patterns and Hypertext. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP'97)*, volume 1241 de *Lecture Notes in Computer Science*, pages 496–510, Springer, 1997.

(Micarelli & Sciarrone, 1996) A. Micarelli et F. Sciarrone. A Case-Based System for Adaptive Hypermedia Navigation. In *Advances in Case-Based Reasoning, Proceedings of the 3rd European Workshop on Case-Based Reasoning (EWCBR'96),* volume 1168 de *Lecture Notes in Artificial Intelligence*, pages 266–279, Springer, 1996.

(Mladenic, 1996) D. Mladenic. *Personal WebWatcher: Design and Implementation*. Rapport technique IJS-DP-7472, School of Computer Science, Carnegie Mellon University, 18 pages, 1996.

(Moorman & Ram, 1992) K. Moorman et A. Ram. A Case-Based Approach to Reactive Control for Autonomous Robot. In *Proceedings of the AAAI Fall Symposium on AI for Real-World Autonomous Mobile Robots*, 1992.

(Morrill, 1998) J. P. Morrill. Distributed Recognition of Patterns in Time Series Data. *Communications of the ACM*, vol. 41, n°5, pages 45–51, 1998.

(Mounir-Alaoui, 1990) A. Mounir-Alaoui. *Raisonnement temporel pour la planification et la reconnaissance de situations*. Thèse de Doctorat, Université Paul Sabatier de Toulouse, 138 pages, 1990.

(Muller, 1997) P.A. Muller. *Modélisation objet avec UML*. Eyrolles, 421 pages, 1997.

(Nakhaeizadeh, 1993) G. Nakhaeizadeh. Learning Prediction of Time Series: A Theoretical and Empirical Comparison of CBR with some other Approaches. In *Proceedings of the 1st European Workshop on Case-Based Reasoning (EWCBR'93)*, rapport SEKI SR-93-12, pages 149–153, Université de Kaiserslautern, 1993.

(Nakhaeizadeh, 1994) G. Nakhaeizadeh. Learning Prediction of Time Series: A Theoretical and Empirical Comparison of CBR with some other Approaches. In *Topics in Case-Based Reasoning*, volume 837 de *Lecture Notes in Artificial Intelligence*, pages 65–76, Springer-Verlag, 1994.

(Odenthal & Quibeldey-Cirkel, 1997) G. Odenthal et K. Quibeldey-Cirkel. Using Patterns for Design and Documentation. In *Proceeding of the European Conference on Object-Oriented Programming (ECOOP'97)*, volume 1241 de *Lecture Notes in Computer Science*, pages 511–529, Springer, 1997.

(Oracle, 1992) ORACLE7 Server, SQL Language Reference Manual. Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065, USA, 1992.

(Porter, 1980) M. F. Porter. An Algorithm for Suffix Stripping. *Program*, vol. 14, pages 130–137, 1980.

(Pree, 1994) W. Pree. *Design Patterns for Object-Oriented Software Development*. Addison-Wesley, 268 pages, 1994.

(Quatrani, 1998) T. Quatrani. *Visual Modeling with Rational Rose and UML*. Addison-Wesley, 222 pages, 1998.

(Quinlan, 1979) J. R. Quinlan. Discovering Rules by Induction form Large Collections of Examples. *Expert Systems in the Micro Electronic Age*, pages 168–201, Edinburgh University press, 1979.

(Raggett *et al.,* 1998) D. Raggett, A. Le Hors et I. Jacobs. HTML 4.0 Specification. Recommandations du W3C, REC-HTML40, http://www.w3.org/TR/REC-html40/, 1998.

(Raggett, 1997) D. Raggett. HTML 3.2 Reference Specification. Recommandations du W3C, REC-HTML32, http://www.w3.org/TR/REC-html32, 1997.

(Ram & Santamaria, 1993) A. Ram et J.C. Santamaria. Continous Case-Based Reasoning. In *Proceedings of the AAAI Case-Based Reasoning Workshop*, pages 86–93, AAAI press, 1993.

(Ram & Santamaria, 1997) A. Ram et J.C. Santamaria. Continuous Case-Based Reasoning. *Artificial Intelligence*, vol. 90, n°1-2, pages 25-77, 1997.

(Redmond, 1990) M. A. Redmond. Distributed Cases for Case-based Reasoning: Facilitating Use of Multiple Cases. In *Proccedings of the AAAI National Conference on Artificial Intelligence (AAAI'90)*, pages 304–309, MIT press, 1990.

(REMIND, 1992) *REMIND: Developer's Reference Manual*. Cognitive System, 200-230, Commercial St., Boston MA 02109, 562 pages, 1992.

(RFC 1738, 1994) T. Berners-Lee, L. Masinter et M. McCahill. *Uniform Resource Locators (URL)*. Request For Comments (RFC) 1738, 1994.

(RFC 1808, 1995) R. Fielding. *Relative Uniform Resource Locator*. Request For Comments (RFC) 1808, 1995.

(RFC 2046, 1996) N. Freed et N. Borenstein. *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. Request For Comments (RFC) 2046, 1996.

(RFC 2068, 1997) R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. Request For Comments (RFC) 2068, 1997.

(RFC 2109, 1997) D. Kristol et L. Montulli. *HTTP State Management Mechanism*. Request For Comments (RFC) 2109, 1997.

(Riehle, 1997) D. Riehle. Composite Design Patterns. In *Proceedings of Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'97)*, volume 32 de *SIGPLAN Notices*, n°10, pages 218–228, 1997.

(Riesbeck & Schank, 1989) C. K. Riesbeck et R. C. Schank. *Inside Case-Based Reasoning*. Erlbaum, 423 pages, 1989.

(Ritter & Larkin, 1994) F. E. Ritter et J. H. Larkin. Developing Process Models as Summaries of HCI Action Sequences. *Human-Computer Interaction*, vol. 9, pages 345–383, 1994.

(RMI, 1997) Java Remote Method Invocation Specification. Sun Microsystem.s Inc, http://java.sun.com/products/jdk/1.1/download-pdf-ps.html, 96 pages, 1997.

(Rose, 1998) Rational Rose 98 - Using Rational Rose. Rational Software Corporation, http://www.rational.com, 267 pages, 1996.

(Rougegrez, 1994a) S. Rougegrez. Prédiction de processus à partir de comportement observés : le système REBECAS. Thèse de Doctorat, Institut Blaise Pascal, 174 pages, 1994.

(Rougegrez, 1994b) S. Rougegrez. Similarity Evaluation between Observed Behaviours for the Prediction of Processes. In *Topics in Case-Based Reasoning*, volume 837 de *Lecture Notes in Artificial Intelligence*, pages 155–166, Springer-Verlag, 1994.

(Rüping, 1996) A. Rüping. Framework Patterns. In *Proceedings of the European Conference on Pattern Langages of Programming (EuroPLoP'96)*, http://www.cs.wustl.edu/~schmidt/europlop-96/EuroPLoP-96.html, 10 pages, 1996.

(Sakagami & Kamba, 1997) H. Sakagami et T. Kamba. Learning Personal Preferences on line Newspaper Articles from User Behaviors. In *Proceedings of the 6th Internatinal World Wide Web Conference*, volume 29 de *Computer Network and ISDN Systems*, pages 1447–1455, Elsevier, 1997.

(Schank, 1982) R. C. Schank. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University press, 234 pages, 1982.

(Schechter *et al*., 1998) S. Schechter, M. Krishman et M. D. Smith. Using Path Profiles to Predict HTTP Requests. In *Proceedings of the 7th International World Wide Web Conference (WWW7)*, volume 30 de *Computer Network and ISDN Systems*, pages 457–467, Elsevier, 1997.

(Schicker *et al.,* 1996) M. A. Schicker, M. S Mazer et C. Brooks. Pan-Browser Support for Annotations and other Meta-Information on the World Wide Web. In *Proceedings of the 5th International World Wide Web Conference*, volume 28 de *Computer Network and ISDN Systems*, pages 1063–1074, Elsevier, 1996.

(Schmid, 1997) H. A. Schmid. Systematic Framework Design by Generalization. *Communications of the ACM*, vol. 40, n°10, pages 48–51, 1997.

(Schmidt *et al.,* 1996) R. Schmidt, B. Heindl, B. Pollwein, et L. Gierl. Abstraction of Data and Time for Multiparametric Time Course Prognoses. In *Advances in Case-Based Reasoning, Proceedings of the 3rd European Workshop on Case-Based Reasoning (EWCBR'96),* volume 1168 de *Lecture Notes in Artificial Intelligence*, pages 377–391, Springer, 1996.

(Schmidt, 1995) D. C. Schmidt. Experiences Using Design Patterns to Develop Reusable Object-Oriented Communication Software. *Communications of the ACM,* vol. 38, n°10, pages 65-74, 1995.

(Scholl *et al.,* 1993) P.C. Scholl, M.C. Fauvet, F. Lagnier, F. Maraninchi. Cours d'informatiques: langages et programmation. Masson, 418 pages, 1993.

(Segev & Chandra, 1993) A. Segev et R. Chandra. A Data Model for Time-Series Analysis. In *Advanced Database Systems,* volume 759 de *Lecture Notes in Computer Science,* chapitre 10, pages 191–212, Springer, 1993.

(Segev & Shoshani, 1993) A. Segev et A. Shoshani. A Temporal Data Model based on Time Sequences. *Temporal Databases: Theory, Design and Implementation*, Benjamin/Cummings, chapitre 11, pages 248–270, Benjamin-Cummings, 1993.

(Segev *et al.,* 1995) A. Segev, C. S. Jensen et R. T. Snodgrass. Report on the 1995 International Workshop on Temporal Databases. *SIGMOD Record*, vol 24, n°4, pages 46–52, 1995.

(Shahabi *et al.,* 1997) C. Shahabi, A. M. Zarkesh, J. Adibi et V. Shah. Knowledge Discovery from Users Web-Page Navigation. In *Proceedings of the 7th International Workshop on Research Issue in Data Engineering, High Performance Database Management for Large-Scale Applications*, pages 20–29, IEEE Computer Society press, 1997.

(Shahar, 1997) Y. Shahar. A Framework for Knowledge-Based Temporal Abstraction. *Artificial Intelligence*, vol. 90, n°1-2, pages 79–133, 1997.

(Slade, 1991) S. Slade. Case-Based Reasoning: A Research Paradigm. *AI magazine*, vol. 12, n°1, pages 42–55, 1991.

(Smyth & Keane, 1995) B. Smyth et M. T. Keane. Remembering to Forget. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 377–382, Morgan Kaufmann, 1995.

(Taheri, 1998) Y. Taheri. *Modèle et composants logiciels pour un moteur de recherche sur le Web fondé sur des robots intelligents*. Mémoire de projet de fin d'études, Ecole Mohammadia d'Ingénieurs, Maroc, et INRIA Sophia Antipolis, Action AID, 37 pages, 1998.

(Trousse & Visser, 1993) B. Trousse et W. Visser. Use of Case-Based Reasoning Techniques for Intelligent Computer-Aided-Design Systems. In *Proccedings of IEEE International Conference on Systems, Man and Cybernetics - Systems Engineering in the Service of Humans*, pages 513–518, 1993.

(Trousse *et al.,* 1998) B. Trousse, M. Jaczynski et R. Kanawati. Towards a Framework for Building Collaborative Information Searching Systems. In *Research and Advanced Technology for Digital Libraries, Proceedings of the 2nd European Conference on Digital Libraries (ECDL'98)- poster*, volume 1513 de *Lecture Notes in Computer Science*, pages 659–660, Springer, 1998.

(UML, 1997) *UML Notation Guide*. Rational Software, 142 pages, 1997.

(Vanoirbeek, 1996) C. Vanoirbeek. Avantages et limites des hypertextes pour la gestion de l'information. *La recherche d'information sur les réseaux*, cours INRIA, pages 103–126, ADBS éditions, 1996.

(Veloso, 1996) M. Veloso. Flexible Strategy Learning: Analogical Replay of Problem Solving Episodes. *Case-Based Reasoning: Experiences, Lessons & Future Directions*, AAAI press, MIT press, pages 137–149, 1996.

(Visser & Trousse, 1993) W. Visser et B. Trousse. Reuse of Designs: An Interdisciplinary Cognitive Approach. *Proceedings of the IJCAI'93 Workshop on Reuse of Designs*, 171 pages, 1993.

(Watson, 1997) I. Watson. *Applying Case-Based Reasoning: Techniques for Enterprise Systems*. Morgan Kaufmann, 289 pages, 1997.

(Wess *et al.,* 1994) S. Wess, K.D. Althoff, et G. Derwand. Using K-d Trees to Improve the Retrieval Step in Case-Based Reasoning. In *Topics in Case-Based Reasoning*, volume 837 de *Lecture Notes in Artificial Intelligence*, pages 167–181, Springer-Verlag, 1994.

(Wexelblat & Maes, 1997) A. Wexelblat et P. Maes. Footprints: Visualizing Histories for Web Browsing. In *Actes de la 5ème conférence sur la Recherche d'Information Assistée par Ordinateur sur Internet (RIAO'97)*, Centre des hautes études internationales d'Informatique, Montréal, pages 75–84, 1997.

(Williams, 1986) B. C. Williams. Doing Time: Putting Qualitative Reasoning on Firmer Ground. In *Proceedings of the AAAI National Conference on Artificial Intelligence (AAAI'86)*, pages 105–112, Morgan Kaufmann, 1986.

(Yan *et al.,* 1996) T.W. Yan, M. Jacobsen, H.Garcia-Molina et U. Dayal. From User Access Patterns to Dynamic Hypertext Linking. In *Proceedings of the 5th International World Wide Web Conference*, volume 28 de *Computer Network and ISDN Systems*, pages 1007–1014, Elsevier, 1996.

2 Index des références

Aamodt & Plaza, 1994 .....18, 32, 33, 35, 107, 213
Allen, 1984 .....75, 213
Althoff *et al*., 1995 .....37, 89, 213, 257
Amstrong *et al*., 1995 .....31, 197
Andreoli *et al*., 1996 .....30, 213
Armstrong *et al*., 1997 .....31, 197, 199, 213
Auriol *et al*., 1994 .....94, 213
Bachiochi *et al*., 1997 .....31, 213
Baille *et al*., 1990 .....25, 213
Baille, 1991 .....25, 213
Baird-Smith *et al*., 1997 .....167, 213
Barletta, 1994 .....90, 213
Barrett *et al.*, 1997 .....197, 199, 213
Bassaville, 1988 .....56, 213
Beans, 1997 .....151, 209, 213
Bergman & Wilke, 1996 .....32, 207, 214
Bergmann *et al*., 1997 .....89, 208, 214
Bergmann *et al.*, 1998 .....89, 208, 214
Berners-Lee *et al*., 1994 .....28, 214
Bird & Wadler, 1988 .....70, 214
Box & Jenkins, 1976 .....27, 214
Brandau *et al*., 1991 .....45, 214
Bray, 1996 .....30, 171, 214
Breiman *et al*., 1984 .....91, 214
Brooks *et al*., 1995 .....165, 214
Brown *et al*., 1995 .....109, 214
Brun & Settenbrino, 1993 .....17, 25, 26, 27, 214
Brun *et al*., 1993 .....17, 25, 26, 27, 214
Bull *et al*., 1997 .....18, 49, 50, 54, 214
Burke *et al.*, 1997 .....197, 199, 214
Bushmann *et al*., 1996 .....119, 215, 243
Canavaggio, 1997 .....62, 63, 143, 215
Carbonell, 1986 .....36, 215
Charles, 1975 .....56, 215
Chartron, 1996 .....28, 30, 215
Chen & Park, 1998 .....162, 208, 215
Chen & Sycara, 1998 .....197, 199, 215
Chiriatti & Plant, 1993 .....27, 215
Codenie *et al*., 1997 .....102, 215
Corvaisier *et al*., 1997 .....51, 162, 197, 199, 215
Corvaisier *et al*., 1998 .....18, 49, 51, 54, 77, 215
Cutler *et al*., 1997 .....173, 215
Demeyer *et al*., 1997 .....103, 106, 215
Deter, 1994 .....45, 215
Dillon *et al*., 1990 .....31, 215
Dousson & Ghallab, 1994 .....59, 215
Dreyer *et al*., 1994 .....62, 215
Elhaddad, 1998 .....167, 209, 216
Elkassar & Charlet, 1997 .....197, 199, 216
Faloutsos *et al*., 1994 .....208, 216
Faurre & Robin, 1986 .....27, 216
Fauvet *et al*., 1997 .....63, 64, 216
Fayad & Schmidt, 1997 .....100, 216
Fontaine, 1996 .....59, 216
Foss, 1989 .....31, 216
Froedlish *et al*., 1997 .....105, 156, 216
Fuchs *et al*., 1995 .....18, 49, 50, 54, 77, 216
Fuchs, 1997 .....152, 216
Fynn *et al*., 1989 .....27, 216
Gamma *et al*., 1995 .....101, 103, 105, 117, 216, 243, 244
Gebhardt *et al*., 1997 .....32, 94, 95, 96, 100, 107, 216
Geib *et al*., 1997 .....166, 168, 216
Ghallab & Mounir-Alaoui, 1989 .....60, 216
Golinelli, 1997 .....135, 216
Golinelli, 1998 .....169, 209, 217
Gonzalez, 1978 .....57, 58, 217
Haimowitz & Kohane, 1993a 59, 60, 217
Haimowitz & Kohane, 1993b 59, 60, 61, 217
Hall & Dowling, 1980 .....57, 217
Hallam-Baker & Connolly, 1996 .....165, 217
Hammond, 1986 .....40, 217
Harmon, 1992 .....89, 217
Haton *et al*., 1991 .....119, 217
Hébraud *et al*., 1998 .....20, 193, 194, 217
Hennessy & Hinkle, 1992 .....37, 217
Hinrichs, 1989 .....40, 217
Huitema, 1995 .....29, 217
Iyengar, 1997 .....165, 217
Jaczynski & Trousse, 1994 .....107, 110, 138, 217
Jaczynski & Trousse, 1997a 19, 20, 217
Jaczynski & Trousse, 1997b 20, 218
Jaczynski & Trousse, 1998a 20, 218
Jaczynski & Trousse, 1998b 20, 218
Jaczynski & Trousse, 1998c 20, 218
Jaczynski & Trousse, 1999 .....19, 218
Jaczynski, 1993 .....138, 152, 218
Jaczynski, 1997 .....19, 218
Jakob & Suslenschi, 1990 .....39, 218
Jerke *et al*., 1990 .....30, 218
Johnson & Foote, 1988 .....100, 101, 118, 218
Johnson, 1992 .....105, 156, 218
Johnson, 1997 .....100, 101, 105, 218
Karacapilidis *et al*., 1997 .....207, 218
Keravnou, 1995 .....38, 218
Kolodner & Leake, 1996 .....32, 218
Kolodner, 1983 .....32, 208, 219
Kolodner, 1992 .....35, 219
Kolodner, 1993 .....18, 32, 40, 41, 68, 208, 219
Kolodner, 1996 .....32, 34, 36, 219
Kopeikina *et al*., 1988 .....45, 219
Korfhage, 1997 .....173, 219
Koton, 1989 .....40, 219
Kurata, 1988 .....27, 219
Lajoie & Keller, 1994 .....105, 219
Landau, 1988 .....27, 45, 219
Lassila & Swick, 1998 .....172, 219
Leake *et al*., 1997 .....33, 219
Leake, 1996 .....32, 35, 41, 218, 219
Lieber, 1997 .....208, 219
Lieberman, 1995 .....31, 197, 199, 219
Lissandre, 1990 .....41, 219
Luotonen & Altis, 1994 .....165, 219
Manago *et al*., 1994 .....94, 219
Mark *et al*., 1996 .....35, 36, 37, 93, 219
Mattsson & Bosch, 1997 .....105, 220
Meeks *et al*., 1996 .....164, 220

Meusel *et al.*, 1997 .....104, 105, 156, 220
Micarelli & Sciarrone, 1996 .....197, 199, 220
Mladenic, 1996 .....165, 197, 199, 220
Moorman & Ram, 1992 .....48, 220
Morrill, 1998 .....57, 220
Mounir-Alaoui, 1990 .....59, 220
Muller, 1997 .....117, 220, 235, 236, 239
Nakhaeizadeh, 1993 .....49, 220
Nakhaeizadeh, 1994 .....18, 49, 54, 220
Odenthal & Quibeldey-Cirkel, 1997.....104, 105, 220
Oracle, 1992 .....132, 220
Porter, 1980 .....173, 220, 281
Pree, 1994 .....220, 243
Quatrani, 1998 .....152, 221, 235
Quinlan, 1979 .....91, 221
Raggett *et al*., 1998 .....29, 221
Raggett, 1997 .....29, 172, 221
Ram & Santamaria, 1993 .....18, 47, 221
Ram & Santamaria, 1997 .....18, 45, 47, 52, 54, 221
Redmond, 1990 .....68, 221
Remind, 1992 .....90, 179, 221
RFC 1738, 1994 .....29, 171, 221
RFC 1808, 1995 .....171, 221
RFC 2046, 1996 .....171, 221
RFC 2068, 1997 .....29, 171, 221
RFC 2109, 1997 .....165, 221
Riehle, 1997 .....103, 221
Riesbeck & Schank, 1989 .....32, 221
Ritter & Larkin, 1994 .....38, 221
RMI, 1997 .....166, 221
Rose, 1998 .....20, 152, 221
Rougegrez, 1994a 18, 45, 46, 52, 54, 68, 77, 221
Rougegrez, 1994b 46, 222
Rüping, 1996 .....104, 222
Sakagami & Kamba, 1997 .....174, 222
Schank, 1982 .....32, 208, 222
Schechter *et al.*, 1998 .....162, 208, 222
Schicker *et al*., 1996 .....165, 222
Schmid, 1997 .....102, 103, 222
Schmidt *et al*., 1996 .....18, 49, 50, 54, 77, 222
Schmidt, 1995 .....104, 222
Scholl *et al.*, 1993 .....70, 222
Segev & Chandra, 1993 .....62, 222
Segev & Shoshani, 1993 .....62, 63, 222
Segev *et al*., 1995 .....62, 222
Shahabi *et al*., 1997 .....162, 166, 222
Shahar, 1997 .....208, 222
Slade, 1991 .....32, 35, 223
Smyth & Keane, 1995 .....36, 223
Taheri, 1998 .....209, 223
Trousse & Visser, 1993 .....138, 223
Trousse *et al*., 1998 .....209, 223
UML, 1997 .....117, 223, 235
Vanoirbeek, 1996 .....28, 223
Veloso, 1996 .....36, 223
Visser & Trousse, 1993 .....138, 223
Watson, 1997 .....32, 89, 98, 223
Wess *et al*., 1994 .....83, 135, 223, 257
Wexelblat & Maes, 1997 .....197, 199, 223
Williams, 1986 .....69, 223
Yan *et al*., 1996 .....197, 199, 223

Table des illustrations

Chapitre I : le RàPC pour une classe de problèmes

Figure I-1 : régulateur de concentration 26
Figure I-2 : exemple simple d'hypertexte (Vanoirbeek, 1996) 28
Figure I-3 : affichage d'une page Web avec le navigateur Netscape 29
Figure I-4 : l'espace virtuel du Web pour Lycos 30
Figure I-5 : principe d'un assistant à la navigation 31
Figure I-6 : cycle du raisonnement à partir de cas 33
Figure I-7 : chargement de l'autoclave 37
Figure I-8 : tâche de réalisation d'un système de RàPC 41

Chapitre II : modèle d'indexation par situations comportementales

Figure II-1 : historiques d'un cas suivant une direction de propagation 46
Figure II-2 : choix du cas minimisant le décalage 47
Figure II-3 : architecture de SINS 48
Figure II-4 : principes de la recherche dans SINS 48
Figure II-5 : mise en correspondance dans SINS 49
Figure II-6 : mise en correspondance de deux séquences dans PADIM 51
Figure II-7 : exemple de cas avec une situation non explicite dans SINS 53
Figure II-8 : électrocardiogramme humain normal (Gonzalez, 1978) 57
Figure II-9 : automate fini pour la reconnaissance d'un électrocardiogramme normal (Gonzalez, 1978) 58
Figure II-10 : déplacement d'un convoyeur dans une usine automatisée (Mounir-Alaoui, 1990) 59
Figure II-11 : définition d'une situation dans IxTeT 59
Figure II-12 : extrait d'un patron de tendances (Haimowitz & Kohane, 1993b) 61
Figure II-13 : types de chroniques (Segev & Shoshani, 1993) 63
Figure II-14 : différentes unités de temps dans TEMPOS (Canavaggio, 1997, page 48) 63
Figure II-15 : hiérarchie de types d'historiques dans TEMPOS (Fauvet *et al.,* 1997) 64
Figure II-16 : exemple de situation complexe 68
Figure II-17 : exemple d'historique en indices potentiels 70
Figure II-18 : exemples de chroniques échantillonnées et à événements 71
Figure II-19 : exemples de contraintes de localisation 73
Figure II-20 : relations entre intervalles 75
Figure II-21 : données d'observation pour la nutrition des plantes 81
Figure II-22 : exemple d'enregistrement pour la nutrition des plantes 81
Figure II-23 : exemple de cas concret pour la nutrition des plantes 83

Tableau II-1 : gestion des cas et découverte de connaissances dans les approches existantes 54
Tableau II-2 : synthèse des approches d'indexation 65

Chapitre III : plate-forme à objets pour la réalisation de systèmes de RàPC

Figure III-1 : exemple d'une hiérarchie de prototypes 90
Figure III-2 : arbre d'induction 91
Figure III-3 : index hybride dans REMIND 92
Figure III-4 : architecture de REPRO (Mark *et al.,* 1996) 93
Figure III-5 : extrait de la représentation d'un cas en CASUEL 94
Figure III-6 : contexte du calcul d'une similarité dans Rabit (Gebhardt *et al.,* 1997, page 54) 95
Figure III-7 : exemple de configuration distribuée du prototype FABEL 96
Figure III-8 : résolution d'un problème technique avec CasePoint WebServer 97
Figure III-9 : représentation d'un cas dans Help!CPR 98
Figure III-10 : extrait de l'interface de programmation de CPR 99
Figure III-11 : cycle de développement d'une plate-forme 102
Figure III-12 : types de points d'ouverture (d'après Schmid, 1997) 103
Figure III-13 : représentation d'un point d'ouverture 103
Figure III-14 : patron de conception *Composite* et son application pour des objets graphiques 104

Figure III-15 : décomposition des points d'ouverture en trois axes de variabilité 107
Figure III-16 : points d'ouverture pour la gestion du raisonnement 108
Figure III-17 : points d'ouverture pour la représentation des cas 108
Figure III-18 : points d'ouverture pour l'organisation de la mémoire 109
Figure III-19 : remplacement du point d'ouverture ReasonerFactory 110
Figure III-20 : points d'ouverture pour la représentation des cas avec des indices comportementaux 111
Figure III-21 : points d'ouverture pour l'organisation de la mémoire dans l'indexation par situations comportementales 112

Tableau III-1 : structuration et dénombrement des points d'ouverture 106

Chapitre IV : patrons de conception et modèles à objets ouverts pour le RàPC

Figure IV-1 : représentation du contrôleur de raisonnement 118
Figure IV-2 : représentation de l'état d'un raisonnement 119
Figure IV-3 : protocole standard de raisonnement 120
Figure IV-4 : interfaces de base pour les résultats des phases de raisonnement 121
Figure IV-5 : adaptation d'interfaces pour l'assemblage 122
Figure IV-6 : la fabrique comme composant du contrôleur du raisonnement 123
Figure IV-7 : classes d'objets du schéma de base pour la phase de recherche 124
Figure IV-8 : protocole de configuration et d'exécution de la recherche 124
Figure IV-9 : schéma d'adaptation d'un cas par analyse des indices 125
Figure IV-10 : protocole de réutilisation par analyse d'indices 126
Figure IV-11 : schéma de réutilisation de plusieurs cas par formule 127
Figure IV-12 : schéma d'apprentissage par chaîne de responsabilité 127
Figure IV-13 : protocole d'apprentissage par chaîne de responsabilité 128
Figure IV-14 : cas et situation 129
Figure IV-15 : indices structurés et types d'indices 129
Figure IV-16 : hiérarchie de poids d'importance pour les indices 130
Figure IV-17 : extrait du diagramme d'objets représentant les indices d'un cas 130
Figure IV-18 : délégation du stockage et de l'indexation 131
Figure IV-19 : hiérarchie de bases de cas 132
Figure IV-20 : exemple d'une structuration de bases de cas 132
Figure IV-21 : index et filtre d'indices 133
Figure IV-22 : hiérarchie extensible d'index 134
Figure IV-23 : exemple de composition d'index 135
Figure IV-24 : hiérarchie d'interfaces des résultats d'un index (recherche et indexation) 136
Figure IV-25 : transmission du résultat courant par les index composés 137
Figure IV-26 : indication des profils associés à un index 137
Figure IV-27 : exemple d'assemblage valide des index 138
Figure IV-28 : valeurs des similarités et relations d'ordre 139
Figure IV-29 : mesures de similarité structurées et modifiables 139
Figure IV-30 : extrait de la hiérarchie des fonctions élémentaires de similarité 141
Figure IV-31 : extrait de la hiérarchie des fonctions d'agrégation 142
Figure IV-32 : hiérarchie des chroniques 144
Figure IV-33 : représentation des enregistrements 144
Figure IV-34 : représentation des cas et structuration des indices 145
Figure IV-35 : décomposition de la partie comportementale 146
Figure IV-36 : hiérarchie des contraintes temporelles 146
Figure IV-37 : hiérarchie des contraintes de localisation 147
Figure IV-38 : représentation des patrons de cas potentiels 147
Figure IV-39 : hiérarchie d'algorithmes d'itération sur les enregistrements 147
Figure IV-40 : extension de la mémoire 148
Figure IV-41 : structuration typique des bases de cas concrets et potentiels 148
Figure IV-42 : structuration typique des index pour l'instanciation des patrons de cas potentiels 149
Figure IV-43 : sous-systèmes principaux de CBR*Tools 150
Figure IV-44 : organisation du sous-système core 150

Figure IV-45 : interface d'utilisation de CBR*Tools avec Rational Rose 152
Figure IV-46 : dépendances de l'axe « gestion du raisonnement » 155
Figure IV-47 : dépendances de l'axe « représentation des cas » 155
Figure IV-48 : dépendances de l'axe « organisation de la mémoire » 156

Tableau IV-1 : exemples de contrôles d'assemblages 121
Tableau IV-2 : fonctions élémentaires de similarité dans CBR*Tools 141
Tableau IV-3 : fonctions d'agrégation de CBR*tools 142
Tableau IV-4 : points d'ouverture à spécialiser obligatoirement pour une application au niveau « RàPC » 153
Tableau IV-5 : points d'ouverture à configurer obligatoirement pour une application au niveau « Indexation par situations comportementales » 154

Chapitre V : application pour l'assistance à la navigation sur le Web

Figure V-1 : réutilisation d'expériences pour déterminer des recommandations 161
Figure V-2 : barre d'outils de Broadway 163
Figure V-3 : contrôleur de Broadway 164
Figure V-4 : architecture globale de Broadway 165
Figure V-5 : architecture détaillée de Broadway 166
Figure V-6 : abonnement d'un module à des événements 168
Figure V-7 : cycle du raisonnement dans Broadway 169
Figure V-8 : exemple de structuration d'une page Web 172
Figure V-9 : extrait d'une navigation 175
Figure V-10 : exemple d'un cas concret 178
Figure V-11 : décomposition hiérarchique des adresses 179
Figure V-12 : utilisation des points d'ouverture pour le raisonnement 184
Figure V-13 : schéma de réutilisation de recommandations 185
Figure V-14 : protocole du schéma de réutilisation de recommandations 186
Figure V-15 : utilisation des points d'ouverture pour les données d'observation 187
Figure V-16 : utilisation des points d'ouverture pour les indices 187
Figure V-17 : utilisation des points d'ouverture pour les cas 188
Figure V-18 : utilisation des index lors d'une recherche des cas sources 188
Figure V-19 : utilisation des points d'ouverture pour l'organisation de l'indexation 189
Figure V-20 : utilisation des points d'ouverture pour l'organisation des cas et de la mémoire 189
Figure V-21 : extrait de l'enregistrement des événements 196
Figure V-22 : extrait du fichier généré par l'enregistreur pour les événements de la Figure V-21 196

Tableau V-1 : URI de pages Web 171
Tableau V-2 : utilisation des concepts pour la représentation des données d'observation 190
Tableau V-3 : utilisation des concepts pour la représentation des cas 190
Tableau V-4 : utilisation des points d'ouverture de CBR*Tools 192
Tableau V-5 : types de classes utilisées dans Broadway 193
Tableau V-6 : synthèse des résultats pour les trois premières hypothèses 195
Tableau V-7 : comparatif des assistants de navigation existants avec Broadway 199

Annexes

Figure A-1 : cas d'utilisation 235
Figure A-2 : types de relations entre cas d'utilisation 235
Figure A-3 : exemple de cas d'utilisation (Muller, 1997, page 127) 236
Figure A-4 : points d'ouverture et cas d'utilisation 236
Figure A-5 : utilisation des points d'ouverture 236
Figure A-6 : attributs et opérations d'une classe 237
Figure A-7 : classe abstraite et interface 237
Figure A-8 : association binaire unidirectionnelle 237
Figure A-9 : type d'associations 238

Figure A-10 : contraintes et relations 238
Figure A-11 : relation d'héritage 238
Figure A-12 : représentations d'un objet 238
Figure A-13 : diagramme d'objets (Muller, 1997, page 136) 239
Figure A-14 : diagramme de séquence 239

Figure C-1 : cycle de raisonnement 257
Figure C-2 : utilisation des points d'ouverture pour la représentation des cas 259
Figure C-3 : représentation des cas 259
Figure C-4 : utilisation des points d'ouverture pour l'organisation de la mémoire 260
Figure C-5 : hiérarchie de prototypes de la deuxième stratégie d'indexation 262
Figure C-6 : structure de la deuxième stratégie d'indexation 264
Figure C-7 : utilisation des points d'ouverture pour le contrôle du raisonnement 264

Tableau C-1 : attributs d'une voiture 258
Tableau C-2 : rapport d'exécution du premier raisonnement 267
Tableau C-3 : rapport d'exécution du deuxième raisonnement 268

Tableau D-1 : paramètres d'observation 281
Tableau D-2 : paramètres de représentation 282
Tableau D-3 : paramètres de recherche 283
Tableau D-4 : paramètres de réutilisation 283
Tableau D-5 : paramètres d'apprentissage et de maintenance 284

Annexe A : éléments de notation UML

Dans cette annexe, nous donnons les éléments de la notation UML (*Unified Modelling Language*) nécessaires pour la compréhension des diagrammes présentés dans ce document. Pour une description exhaustive de la notation, le lecteur pourra se référer aux ouvrages spécialisés (UML, 1997 ; Muller, 1997 ; Quatrani, 1998). La notation UML est une notation conçue pour la modélisation objet d'applications et fait suite notamment aux notations des méthodes OMT et Booch. Nous décrivons plus précisément les principes des diagrammes de cas d'utilisation, de classes, d'objets et de séquence.

1 Diagramme de cas d'utilisation

Les cas d'utilisation permettent de modéliser et de structurer les interactions entre les utilisateurs au sens large, appelés acteurs, et un système. Les cas d'utilisation représentent un moyen d'analyse des besoins utilisateurs et permettent de relier les actions faites par un utilisateur avec les réactions attendues d'un système. Plus précisément, un cas d'utilisation unitaire est une abstraction d'un ensemble de scénarios concrets effectués sur l'initiative d'un type d'utilisateurs (cf. Figure A-1).

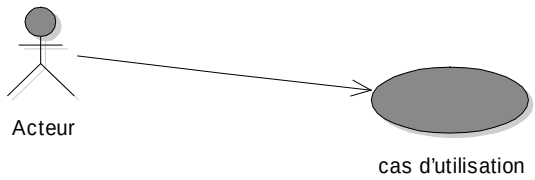


Figure A-1 : cas d'utilisation

Les cas d'utilisation peuvent être structurés et reliés par deux types principaux de relations : les relations d'utilisation (*uses*) et les relations d'extension (*extends*). Une relation d'utilisation permet de décomposer un cas d'utilisation en sous-cas d'utilisation. Une relation d'extension indique une spécialisation. Dans la Figure A-2, l'interaction définie par le cas d'utilisation A comprend celle du cas d'utilisation B, et le cas d'utilisation C est un cas particulier du cas d'utilisation B.

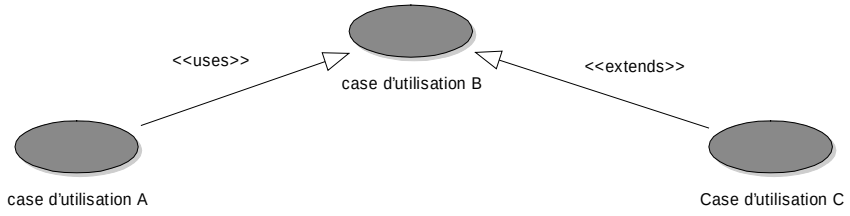


Figure A-2 : types de relations entre cas d'utilisation

Par exemple (cf. Figure A-3), si l'on désire modéliser les opérations de virements bancaires, on peut identifier deux types d'acteurs : les clients distants et les clients locaux. Les deux types de clients effectuent un virement, mais les clients distants utilisent un Minitel qui engendre une interaction spécifique (relation d'extension). Dans tous les cas, l'opération de virement comprend une phase d'identification (relation d'utilisation).

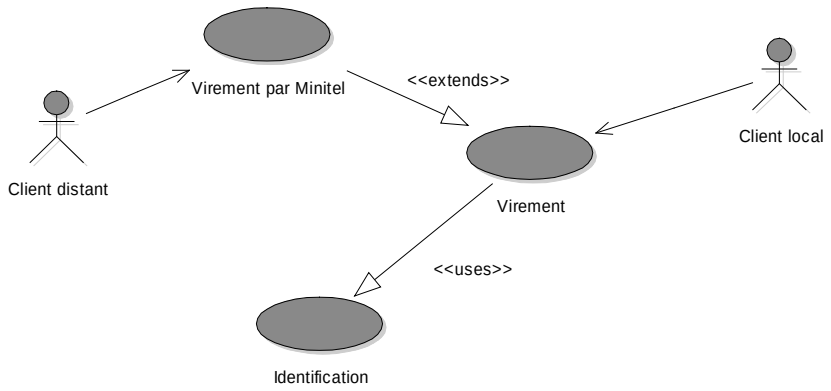


Figure A-3 : exemple de cas d'utilisation (Muller, 1997, page 127)

Dans le cadre de la plate-forme à objet CBR*Tools, nous avons utilisé les cas d'utilisation pour exprimer les interactions entre un utilisateur et le système formé par la plate-forme elle-même complétée de son environnement de manipulation. Un cas d'utilisation peut alors être relié à des points d'ouverture qui participent dans la réalisation du cas d'utilisation (cf. Figure A-4).

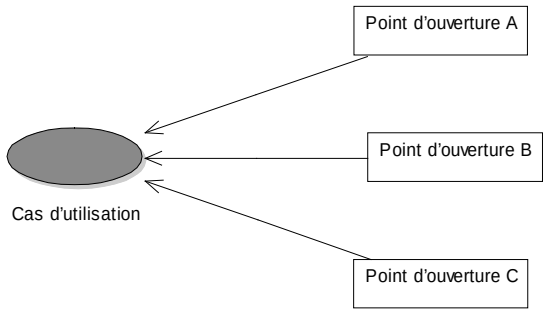


Figure A-4 : points d'ouverture et cas d'utilisation

Les cas d'utilisation nous permettent également de préciser le type de la configuration d'un point d'ouverture dans la réalisation d'une application faite avec CBR*Tools, tout en gardant la structure d'analyse initiale (cf. Figure A-5). Nous avons identifié trois types de configuration : spécialisation (point d'ouverture à boîte transparente), instanciation (point d'ouverture à boîte noire), ou utilisation du comportement par défaut. Si un même point d'ouverture est utilisé plusieurs fois avec différents types de configuration, nous ne retenons que le plus spécifique suivant cet ordre. Nous utilisons alors les stéréotypes de UML pour indiquer le type de la configuration sur les relations entre les points d'ouverture et les cas d'utilisation.

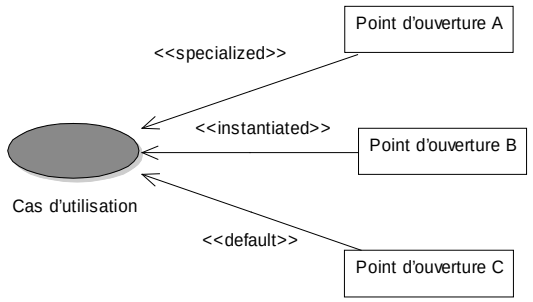


Figure A-5 : utilisation des points d'ouverture

2 Diagramme de classes

Une classe d'objets est représentée par un rectangle comprenant trois parties (cf. Figure A-6) : nom de la classe, attributs et opérations (ou méthodes). Les listes des attributs et des opérations sont toutefois optionnelles suivant le degré de détail recherché dans un diagramme : ces parties peuvent être vides ou même absentes. Les attributs et les opérations possèdent une visibilité (notamment publique ou protégée) qui est indiquée par un symbole précédent leur nom : si la forme d'une clef est dessinée, l'accès est dit protégé car il est réduit à la classe courante et à ses sous-classes.

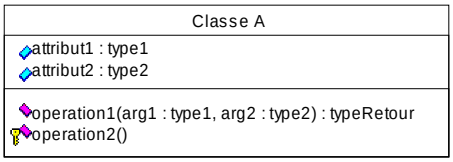


Figure A-6 : attributs et opérations d'une classe

Trois types de classes sont utilisés : les classes concrètes, les classes abstraites et les interfaces. Contrairement à une classe concrète, une classe abstraite dont la définition est en italique (cf. Figure A-7), ne peut pas être instanciée. Une interface est une classe ne définissant que des opérations et ne contient aucun code. Une interface est identifiée par l'indication du stéréotype « Interface » accolé à son nom.

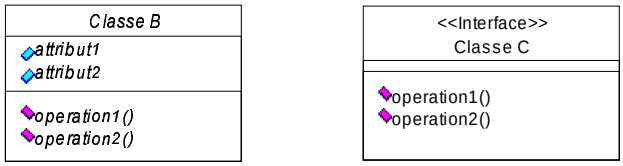


Figure A-7 : classe abstraite et interface

Une association représente une relation structurelle entre différentes classes. Une association est généralement bidirectionnelle mais nous précisons à chaque fois le sens de la navigation. L'association est alors dite unidirectionnelle (cf. Figure A-8) et indique également le sens de lecture de l'association. Chaque classe joue un rôle dans l'association qui porte une indication de multiplicité (1 pour un et un seul objet, et une étoile pour zéro à plusieurs objets). Par exemple, la Figure A-8 indique qu'un objet de classe C1 est associé à un ensemble d'objets de classe C2, sachant qu'un même objet de classe C2 est associé à un unique objet de classe C1.

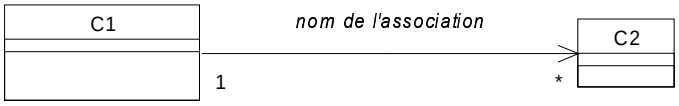


Figure A-8 : association binaire unidirectionnelle

Différents types d'associations sont également définis (cf. Figure A-9) : associations qualifiées, associations dérivées, associations avec classe et agrégations. Une association qualifiée est une association qui permet de restreindre les objets référencés dans une association grâce à une clef. Une association dérivée est une association calculée à partir d'une autre association. Une association avec classe est une association elle-même gérée par une classe qui peut ainsi fournir des attributs ou des opérations spécifiques pour l'association. Enfin, une agrégation est une association fortement asymétrique.

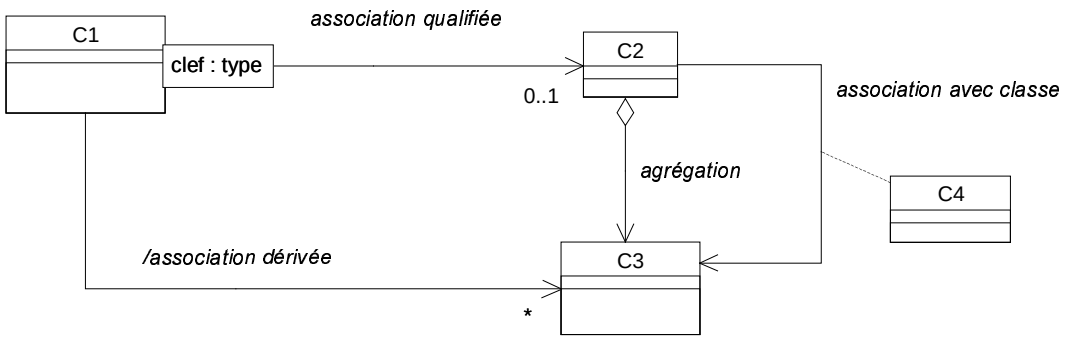


Figure A-9 : type d'associations

Des contraintes peuvent être posées sur les associations (cf. Figure A-10) : contrainte sur le rôle d'une association (par exemple, les objets sont ordonnés), ou entre associations (par exemple, une association est un sous-ensemble d'une autre).

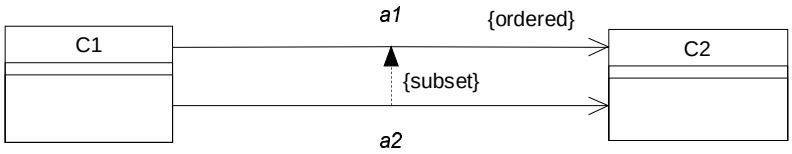


Figure A-10 : contraintes et relations

Enfin, les classes peuvent être reliées par la relation d'héritage simple ou multiple (cf. Figure A-11).

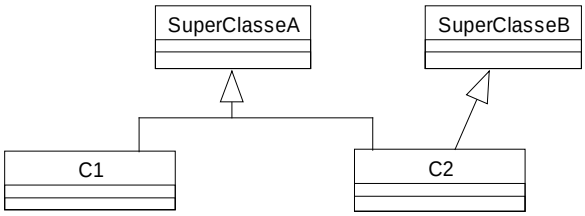


Figure A-11 : relation d'héritage

3 Diagramme d'objets

Un diagramme d'objets permet de montrer concrètement un assemblage entre objets. Un objet (ou instance) est représenté par un rectangle qui comporte généralement le nom de l'objet et sa classe (cf. Figure A-12). Toutefois, la classe ou le nom de l'objet peuvent être omis.

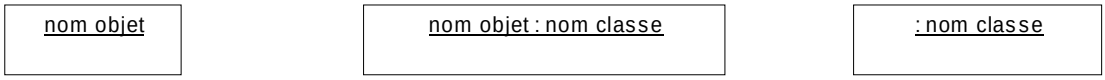


Figure A-12 : représentations d'un objet

La Figure A-13 donne l'exemple d'un diagramme d'objets montrant les relations d'agrégation entre une voiture, ses roues et son moteur.

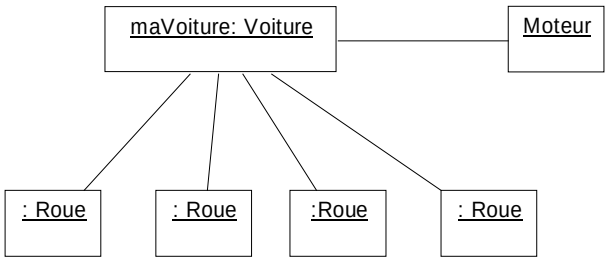


Figure A-13 : diagramme d'objets (Muller, 1997, page 136)

4 Diagramme de séquence

Un diagramme de séquence montre chronologiquement (de haut en bas) les interactions entre un ensemble d'objets (cf. Figure A-14). Chaque objet dispose d'une ligne de vie (ligne verticale). Sur ces lignes de vie, des périodes d'activité sont indiquées par des rectangles fins qui sont superposés en cas d'appel récursif.

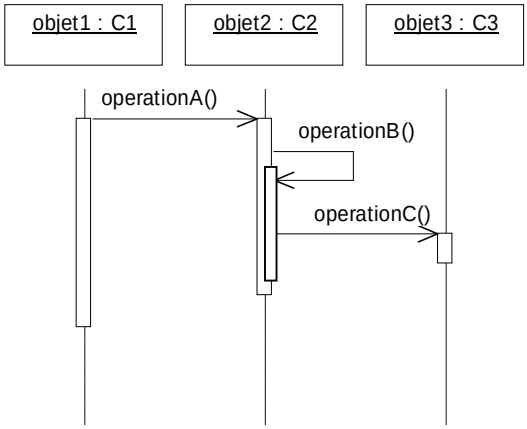


Figure A-14 : diagramme de séquence

Annexe B : patrons de conception et CBR*tools

Dans cette annexe, nous présentons un mémento des patrons de conception utilisés dans CBR*Tools. Notre objectif n'est donc pas de décrire chaque patron utilisé de manière exhaustive, mais seulement de faciliter la lecture du document en rappelant les caractéristiques fondamentales de ces patrons. Le lecteur pourra se référer à des ouvrages spécialisés pour de plus amples descriptions (Gamma *et al.*, 1995 ; Pree, 1994 ; Bushmann *et al.*, 1996). Chaque patron est présenté suivant une fiche qui comporte six parties :

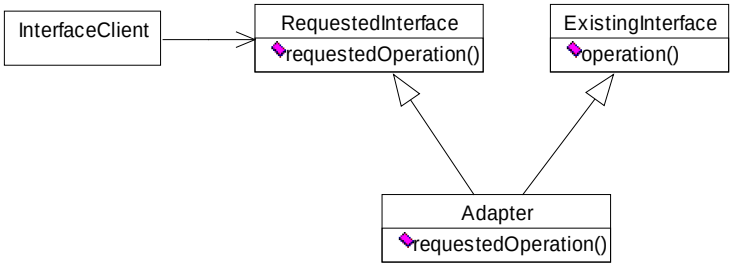
- *nom* : donne le nom du patron en français et en anglais tel qu'il est défini dans (Gamma *et al.*, 1995).
- *description* : décrit très brièvement le contexte, le problème et la solution abordés par ce patron,
- *structure et constituants* : liste les éléments typiques intervenant dans le patron avec des illustrations graphiques (diagramme de classes et si nécessaire diagramme d'objets).
- *collaborations* : définit les principales interactions entre les constituants avec, si nécessaire, la représentation graphique d'un diagramme de séquence.
- *forces et faiblesses* : présente les principales forces (avantages) et faiblesses (inconvénients) du patron, soit dans sa mise en œuvre, soit dans son fonctionnement.
- *utilisation dans CBR*Tools* : référence les utilisations principales du patron dans CBR*Tools.

Les fiches sont classées dans l'ordre alphabétique : Adaptateur, Chaîne de responsabilité, Composite, Fabrication, Fabrique abstraite, Façade, Itérateur, Patron de méthode, Prototype, Stratégie.

Adaptateur (*Adapter*)

Description : ce patron est utile lorsque l'on désire utiliser une classe existante mais que l'interface proposée n'est pas satisfaisante ; il propose alors de définir par héritage multiple une classe effectuant les actions d'adaptation nécessaires.

Structure et constituants :



- RequestedInterface définit l'interface désirée par le client.
- ExistingInterface définit l'interface existante des objets à manipuler.
- Adpater effectue les actions nécessaires pour la réalisation des opérations suivant l'interface désirée en se basant sur les opérations existantes.
- InterfaceClient manipule l'objet suivant l'interface désirée.

Collaborations :

- le client manipule l'objet par l'interface désirée.
- l'adaptateur effectue les actions d'adaptation et utilise l'interface existante.

Forces et faiblesses :

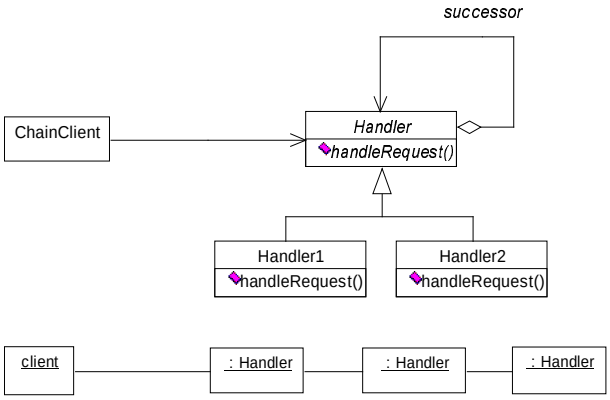
- permet la réutilisation d'une classe d'objets existante dans un autre contexte.
- il n'est pas toujours possible d'effectuer des actions d'adaptation pour réaliser ce patron.

***Utilisation dans CBR*Tool*s :** ce patron est utilisé pour connecter deux phases de raisonnement dont la première fournit un type de résultats qui n'est pas conforme à l'attente de la seconde. Ce patron est également utilisé de la même manière pour connecter des index. Toutefois, dans CBR*Tools, l'application de ce patron est facilitée par la définition de hiérarchies de classes interfaces (i.e. ne définissant aucun code). L'adaptation évite ainsi l'appel à du code existant (qui n'est pas toujours adaptable) et se présente de manière symétrique entre l'interface d'origine et celle qui est désirée. Ce type d'adaptateur est dit *bidirectionnel* (Gamma *et al.*, 1995, p. 168).

Chaîne de Responsabilité (*Chain of Responsibility*)

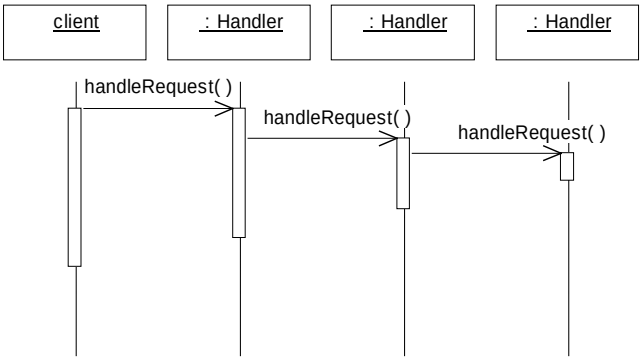
Description : lorsque plusieurs objets sont susceptibles de pouvoir traiter une requête suivant le contexte, ce patron permet d'éviter le couplage direct entre l'émetteur de la requête et son traitement par l'un des récepteurs potentiels. Une chaîne d'objets récepteurs est alors constituée et parcourue jusqu'à ce que l'un d'entre eux traite effectivement la requête.

Structure et constituants :



- Handler définit l'interface d'un récepteur de requêtes.
- Handler1 et Handler2 traitent concrètement une requête ou la passent à leur successeur.
- ChainClient transmet une requête.

Collaborations : la requête émise par le client est transmise le long de la chaîne jusqu'à son traitement effectif.



Forces et faiblesses :

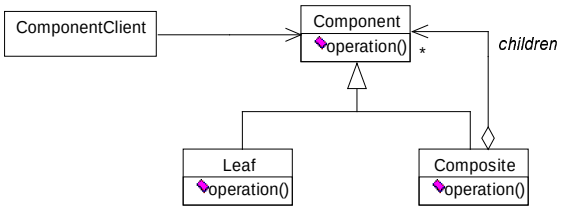
- 👍 réduction du couplage entre le client et le traitement d'une requête par un ensemble d'objets dont on ne sait pas au préalable lequel pourra effectivement traiter la requête.
- 👍 possibilité de modifier dynamiquement la façon de traiter une requête.
- 👍 souplesse dans l'attribution des responsabilités de traitement aux objets de traitement.
- 👎 suivant les contraintes de conception, le traitement peut ne pas être garanti.

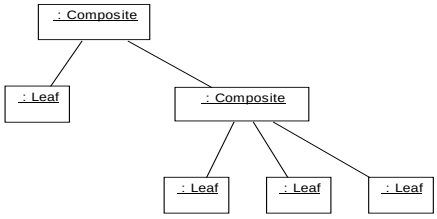
***Utilisation dans CBR*Tool*s :** un schéma d'apprentissage utilise ce patron pour prendre la décision de l'ajout d'un cas et de son indexation dans la mémoire. Une chaîne de décideurs est alors formée, dans laquelle chaque élément étudie suivant un critère la réponse à cette requête.

Composite (*Composite*)

Description : lorsque l'on souhaite représenter une arborescence ou masquer les différences entre des objets composites et simples de natures similaires, le patron composite propose la définition d'une interface unifiée d'objets simples et composites.

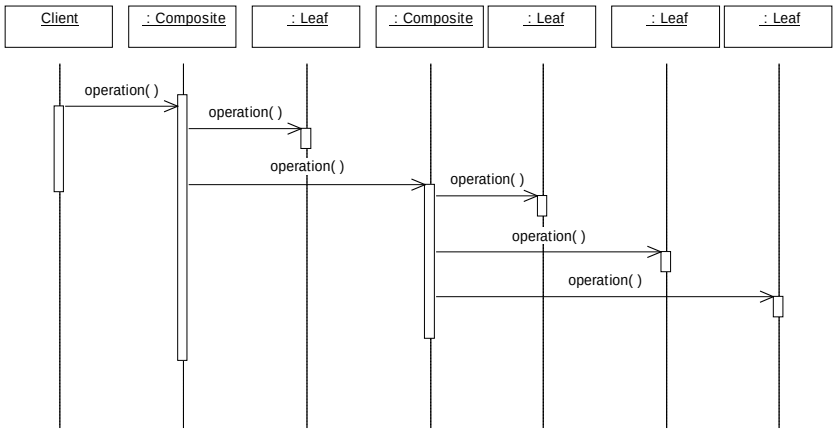
Structure et constituants :





- Component définit l'interface abstraite d'un composant simple ou composite.
- Leaf définit le comportement d'un composant feuille, c'est-à-dire qui ne peut pas avoir de fils.
- Composite définit le comportement d'un composite en stockant ses fils et en répercutant le traitement des méthodes à ses fils de manière appropriée.
- ComponentClient manipule des composants.

Collaborations : les clients manipulent un composant au travers de son interface générale. Une feuille exécute directement les opérations demandées alors qu'un composite transfère les requêtes à ses fils en opérant des actions supplémentaires avant ou après ce transfert.



Forces et faiblesses :

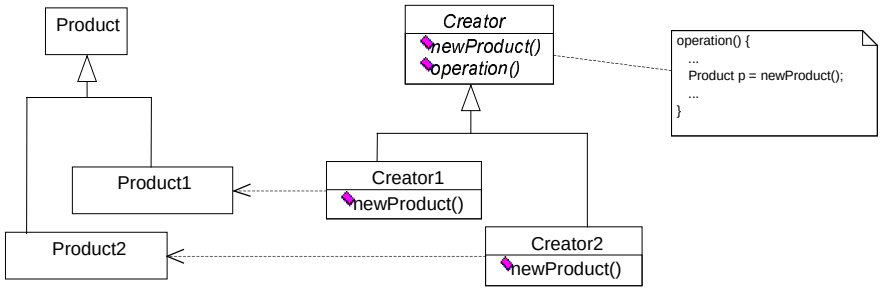
- structuration arborescente d'objets : un composite qui réunit des sous-composants peut être à son tour intégré dans un composite.
- encapsulation de l'hétérogénéité d'objets simples et composites : un objet composite peut être utilisé à la place d'un objet simple et inversement.
- intégration facilitée de nouvelles classes d'objets composites ou simples qui pourront être combinés de manière stable.
- nécessité de contrôler dynamiquement des contraintes particulières dans la composition d'objets au sein d'un composite.
- la maximisation de l'interface générale d'un composant (nécessaire pour l'encapsulation) peut amener à déclarer des méthodes qui n'ont pas de sens pour des objets simples.

Utilisation dans CBR*Tool*s : ce patron est utilisé pour représenter de manière structurée les indices, les similarités, les bases de cas et les index.

Fabrication (*Factory Method*)

Description : lorsqu'un algorithme a besoin de créer un objet dont sa classe concrète est variable suivant le contexte, ce patron propose la délégation à une sous-classe de l'instanciation de cet objet.

Structure et constituants :



- Creator définit une méthode dans laquelle un objet doit être créé et délègue cette instanciation à une de ses sous-classes par la définition de méthode à spécialiser (méthode newProduct).
- Creator1 et Creator2 implantent de manière concrète l'instanciation de l'objet requis.
- Product définit l'interface abstraite de l'objet qui est manipulé par Creator.
- Product1 et Product2 implantent les comportements des objets à utiliser.

Collaborations :

- Creator délègue à ses sous-classes l'instanciation de l'objet requis.

Forces et faiblesses :

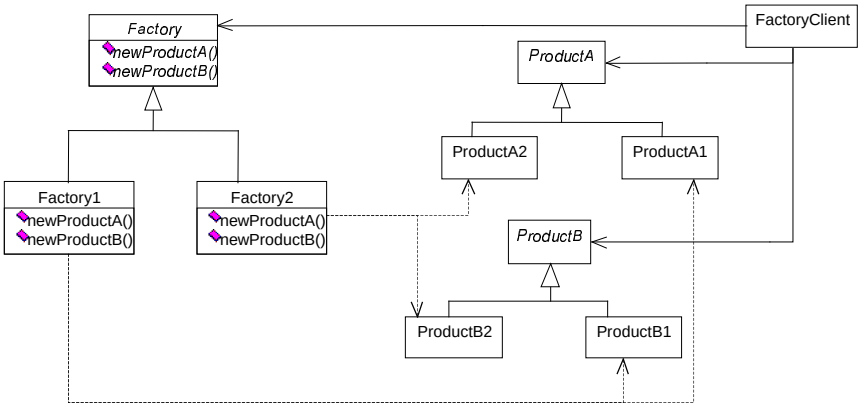
- 👍 dispense d'avoir à spécifier directement l'objet à instancier et permet à une sous-classe de choisir l'objet concret voulu.
- 👎 dans certains cas, la spécialisation de la classe mère est effectuée uniquement pour spécifier l'instanciation (opération plus lourde par exemple que de passer en paramètre l'objet requis). Il est ainsi préférable d'utiliser ce patron lorsqu'il faut également pour d'autres raisons spécialiser la classe mère.

***Utilisation dans CBR*Tool*s :** ce patron intervient dans différents algorithmes de CBR*Tools et notamment dans les protocoles internes des index pour la création des objets sauvegardant le résultat d'une recherche (de classe IndexResult) de manière à instancier les objets adéquats.

Fabrique Abstraite (*Abstract Factory*)

Description : lorsque différentes familles d'objets apparentés doivent être représentées tout en masquant leurs différences aux clients, ce patron propose la définition d'une interface abstraite centralisée de création d'un ensemble d'objets qui peut être spécialisée.

Structure et constituants :



- Factory définit l'interface abstraite de création d'une famille d'objets.
- Factory1 et Factory2 réalisent la création des objets en encapsulant le choix des classes concrètes.
- ProductA et ProductB définissent les interfaces de manipulation des objets à manipuler dans une famille.
- ProductA1, ProductA2, ProductB1 et ProductB2 définissent les comportements concrets des objets.
- FactoryClient crée les objets d'une famille et les manipule au travers de leur interface abstraite.

Collaborations :

- une fabrique délègue à ses sous-classes la création effective des objets d'une famille.
- le client n'accède aux objets ainsi créés que par leur interface abstraite.
- les objets d'une même famille peuvent coopérer directement entre eux si nécessaire.

Forces et faiblesses :

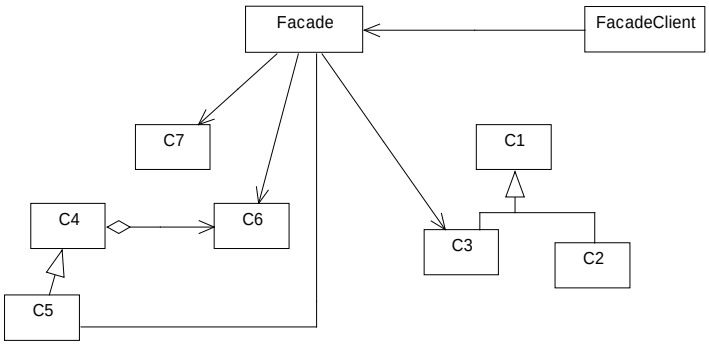
- 👍 le maintien de la cohérence entre objets est favorisé.
- 👍 la substitution d'une classe concrète par une autre est facilitée et reste masquée pour le client.
- 👎 si la coopération directe entre les objets concrets est nécessaire, ces objets sont rendus moins réutilisables et interchangeables.

Utilisation dans CBR*Tools : les objets nécessaires au raisonnement (phase de raisonnement, cas cible, objets sauvegardant l'état du raisonnement, etc.) représentent une famille dont la cohérence doit être assurée. C'est pourquoi, une fabrique abstraite (classe ReasonerFactory) est utilisée pour réunir les opérations de création de ces objets. Cette fabrique est spécialisée (classe TimeExtendedReasonerFactory) de manière à déclarer les opérations de création d'objets spécifiques pour l'indexation par situations comportementales.

Façade (*Facade*)

Description : lorsque l'on désire faciliter l'utilisation d'un sous-système, ce patron permet de fournir une interface simplifiée qui évite le couplage direct entre les clients et les éléments du sous-système.

Structure et constituants :



- Facade définit une interface unifiée pour le sous-système et donne accès à certains composants publics.
- C1 à C7 implantent des fonctions du sous-système.
- FacadeClient manipule le sous-système en passant par la façade.

Collaborations :

- le client manipule le sous-système en s'adressant à la façade ou aux éléments du sous-système rendus publics par la façade.
- la façade transmet les requêtes au sous-système après transformation si nécessaire.

Forces et faiblesses :

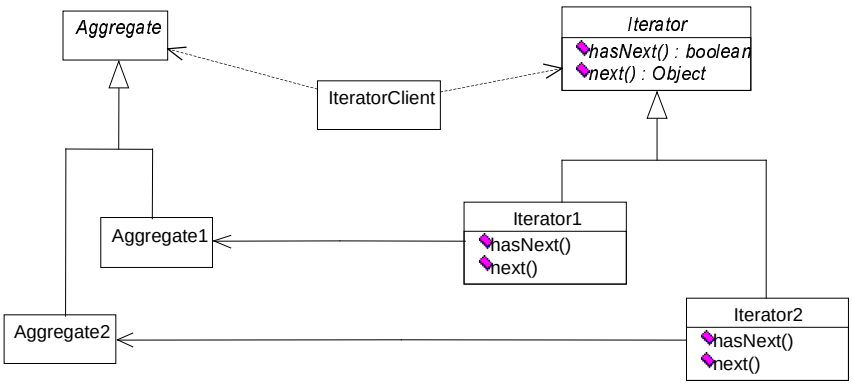
- diminue le couplage entre le client et le sous-système.
- masque des éléments privés du sous-système.
- l'interface unifiée présentée par la façade peut être trop restrictive pour utiliser l'ensemble des fonctions du sous-système.

***Utilisation dans CBR*Tool*s :** le contrôleur du raisonnement (classe Reasoner) est une façade pour l'ensemble du système de raisonnement, la mémoire (classe Memory) est une façade simplifiant la gestion de la base des cas et les index, enfin la partie comportementale (classe BehaviouralSituation) d'une situation comportementale est une façade des indices comportementaux pour simplifier leur gestion.

Itérateur (*Iterator*)

Description : lorsque l'on désire masquer au client la mécanique de parcours d'un agrégat, ce patron permet l'encapsulation de l'itération dans une hiérarchie de classe, soit pour des itérations polymorphes, soit pour offrir différents types de parcours pour un même type d'agrégat.

Structure et constituants :



- Iterator définit l'interface d'itération.
- Iterator1 et Iterator2 implantent deux mécaniques de parcours.
- Aggregate définit l'interface d'un agrégat.
- Aggregate1 et Aggregate2 implantent deux types d'agrégats concrets.
- IteratorClient lance l'itération sur l'agrégat.

Collaborations :

- le client demande la création d'un objet d'itération à l'agrégat ou à un autre objet responsable de l'instanciation de l'itérateur.
- le client contrôle l'itération (itérateur externe) en manipulant l'itérateur.

Forces et faiblesses :

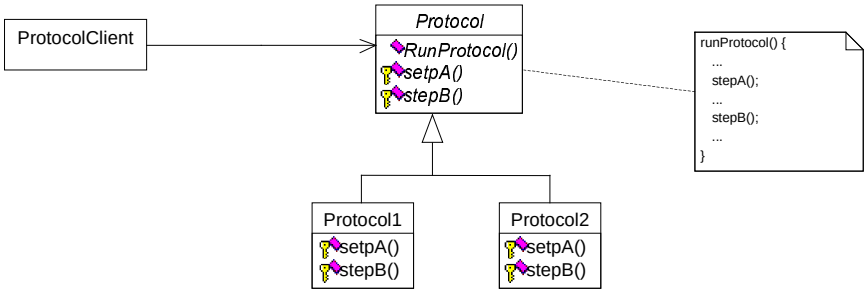
- 👍 permet de parcourir un agrégat sans connaître sa structure interne.
- 👍 permet de parcourir de différentes manières un même agrégat.
- 👎 l'agrégat peut être obligé d'exporter des méthodes uniquement pour que les itérateurs puissent fonctionner.

***Utilisation dans CBR*Tool*s :** le patron Itérateur est utilisé pour parcourir les enregistrements (classe Record) dans l'indexation par situations comportementales. Chaque type d'enregistrements définit un parcours par défaut adapté à sa représentation interne (propriétés des chroniques dont il est composé). De plus, les patrons de cas potentiels (classe PotentialCaseTemplate) définissent leur propre parcours d'un enregistrement suivant leurs conditions d'instanciation.

Patron de méthode (*Template Method*)

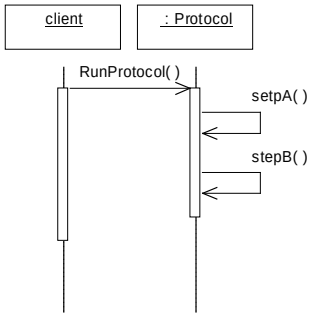
Description : lorsqu'un algorithme possède des étapes variables suivant le contexte, ce patron propose de déléguer à des sous-classes la réalisation concrète de ces étapes.

Structure et constituants :



- Protocol définit l'interface de l'algorithme et appelle des méthodes pour la réalisation de certaines étapes variables.
- Protocol1 et Protocol2 implantent les étapes manquantes dans l'algorithme.
- ProtocolClient lance l'exécution de l'algorithme.

Collaborations : le client lance l'algorithme dont la partie invariante est exécutée directement alors que les étapes variables sont réalisées de manière spécifique par les opérations définies dans une classe concrète.



Forces et faiblesses :

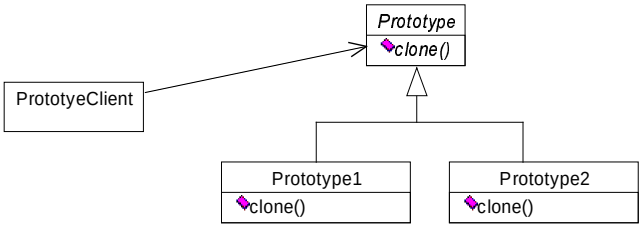
- permet d'implanter une fois pour toutes la structure d'un algorithme et des parties invariables qui seront partagées.
- permet de personnaliser certaines parties d'un algorithme dans des sous-classes.
- les étapes déléguées aux sous-classes doivent être choisies avec soin pour être vraiment utiles.
- problème de vérification des hypothèses faites par la classe mère qui peuvent être violées par l'implantation dans une sous-classe.

Utilisation dans CBR*Tools : ce patron est utilisé dans les phases de raisonnement et dans le protocole d'indexation, pour offrir des algorithmes et des protocoles ouverts.

Prototype (*Prototype*)

Description : lorsque l'on désire créer un objet à partir d'un objet existant, ce patron propose la définition d'une méthode de clonage qui masque au client les opérations effectives de duplication.

Structure et constituants :



- Prototype définit l'interface de l'opération de clonage avec si besoin une implantation par défaut.
- Prototype1 et Prototype2 implantent l'opération de clonage appropriée.
- PrototypeClient crée de nouveaux objets en demandant à un objet primaire de se cloner.

Collaborations :

- un client demande à un objet de se cloner.
- l'opération de clonage est déléguée aux sous-classes.

Forces et faiblesses :

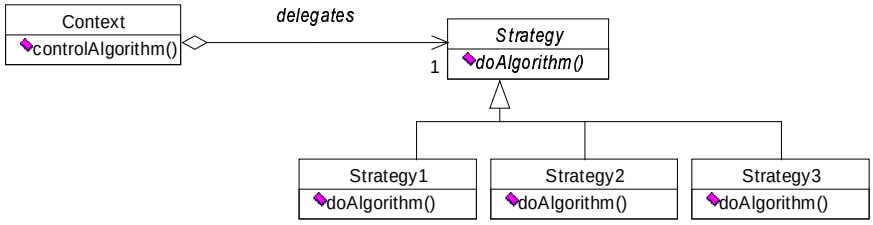
- encapsule la création d'un objet et le client n'a pas de connaissance explicite sur la classe de cet objet.
- permet la définition dynamique de classes d'objets plutôt qu'une définition statique faite à la compilation.
- permet de constituer des objets qui ne diffèrent que par des modifications faites incrémentalement.
- l'implantation d'une opération de clonage peut être difficile s'il existe des composants qui ne peuvent être clonés (opération de clonage non préalablement définie) ou s'il y a des références circulaires.
- la profondeur du clonage des composants à effectuer peut dépendre du contexte d'utilisation.

***Utilisation dans CBR*Tool*s :** ce patron est utilisé pour la constitution par clonage d'une bibliothèque de mesures de similarité, pour la copie de vecteurs de poids et pour la configuration du parcours des index. Pour réaliser ces clonages, tous les objets accessibles depuis une similarité implantent eux-mêmes une opération de clonage. Cette opération retourne un nouvel objet identique à l'objet primaire ou le même objet, si celui-ci est non modifiable (il sera donc partagé par tous les clones).

Stratégie (*Strategy*)

Description : lorsqu'il existe plusieurs implantations possibles d'un algorithme et qu'il peut être nécessaire d'en choisir une dynamiquement, ce patron propose la délégation de l'exécution de c*et al*gorithme à des objets interchangeables qui respectent une interface commune.

Structure et constituants :



- Strategy définit l'interface d'appel d'un algorithme.
- Strategy1, Strategy2 et Strategy3 implantent des variantes d'un même algorithme.
- Context détient le contexte d'exécution.

Collaborations :

- Context lance l'exécution de l'algorithme.
- Strategy peut avoir besoin de données contextuelles présentes dans Context.

Forces et faiblesses :

- 👍 permet la définition de famille d'algorithmes.
- 👍 permet le choix de l'algorithme de manière dynamique et adaptée au contexte.
- 👎 il peut être difficile de prévoir la représentation des paramètres issus du contexte d'exécution qui sont transmis de manière identique à tous les algorithmes concrets.

***Utilisation dans CBR*Tool*s :** ce patron est utilisé pour chacune des phases du raisonnement (classes Retrieve, Resuse, Revise et Retain) ainsi que la réalisation des phases elles-mêmes (classes CaseSetAdpatationFormula et AnalysisAdpatationFormula). Il est également utilisé pour les filtres d'indices (classe IndiceFilter) et dans les similarités (classes ValueSimilarityFct et AggregationFct).

Annexe C : exemple d'utilisation de CBR*Tools

Pour illustrer l'aide apportée par CBR*Tools dans la réalisation d'un système de RàPC, nous donnons l'exemple de notre prototype dans le cadre du calcul du facteur de risque d'une voiture pour un assureur. L'implantation de ce système nécessite uniquement la création de dix classes dont le code reste très déclaratif. Nous présentons tout d'abord le cadre de l'application, puis nous expliquons la construction du système de raisonnement avec CBR*Tools. Enfin, nous donnons des exemples d'exécution ainsi que l'intégralité du code source.

1 Présentation de l'application

L'objectif de cette application est de déterminer le facteur de risque (nombre entier compris entre -3 et 3) pour une voiture donnée (cas cible) en fonction des assignations déjà effectuées sur d'autres voitures (cas sources). L'approche RàPC repose sur l'utilisation d'une base de cas dans laquelle un cas associe la description des caractéristiques d'une voiture (problème formé d'indices) au facteur de risque (solution). Nous proposons alors d'effectuer un raisonnement en trois étapes dans lequel la phase de révision n'est pas utilisée (cf. Figure C-1). La phase de recherche permet d'identifier un ensemble de cas dont la description est proche de celle de la voiture courante pour laquelle il s'agit de déterminer le facteur de risque. Deux stratégies de recherche sont mises en œuvre pour les évaluer : recherche par plus proches voisins, et recherche combinée d'un filtrage par hiérarchie de prototypes et sélection par plus proches voisins. La phase de réutilisation prend en compte l'ensemble des cas retrouvés pour calculer le facteur de risque, en effectuant une moyenne des facteurs de risque assignés aux cas sources pondérés par leur similarité. Enfin, la phase d'apprentissage ajoute le nouveau cas à la mémoire, s'il n'existe pas déjà un cas trop proche du cas cible courant, et si la base de cas ne dépasse pas une taille donnée.

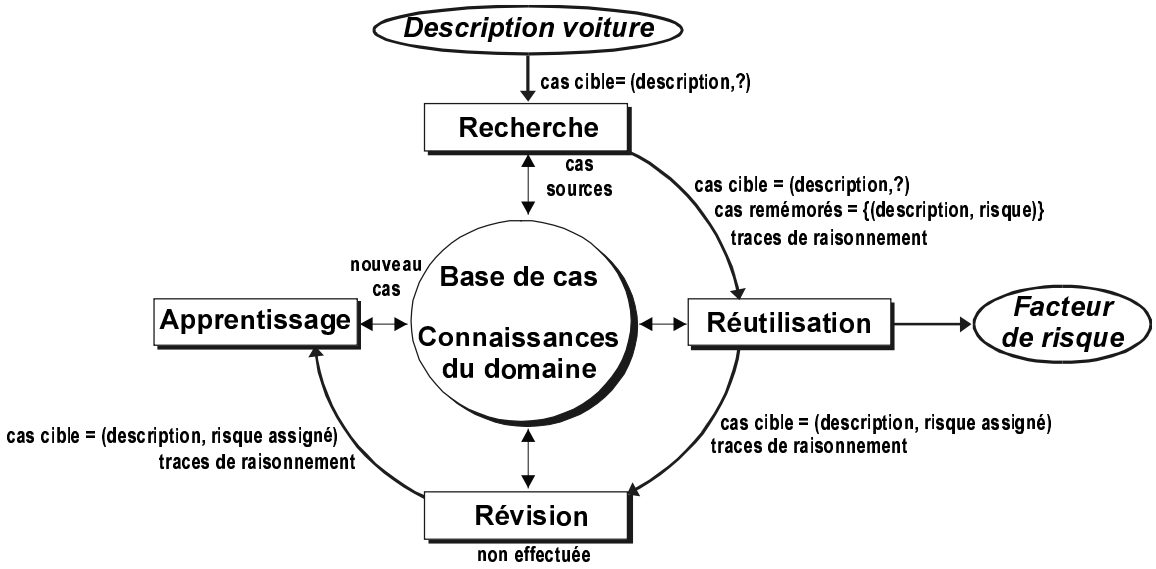


Figure C-1 : cycle de raisonnement

Bien que le raisonnement proposé soit réaliste, il ne s'agit pas de valider les résultats obtenus. Le prototype réalisé a pour but uniquement de montrer comment CBR*Tools facilite le développement d'un tel cycle de raisonnement. La base de cas utilisée dans ce prototype a déjà été utilisée pour effectuer des comparaisons et des expérimentations en RàPC (Althoff *et al.*, 1995 ; Wess *et al.*, 1994). Elle est librement accessible sur le serveur FTP :

ftp://ftp.ics.uci.edu/pub/ml-repos/machine-learning-databases/autos

2 Construction du système de raisonnement

Nous expliquons la construction de l'application et l'utilisation des ressources de CBR*Tools suivant les trois axes de variabilité : représentation des cas (2 classes), organisation de la mémoire (3 classes), gestion du raisonnement (4 classes).

2.1 Représentation des cas

La représentation des cas repose sur des données brutes décrivant 26 attributs pour chaque voiture selon le Tableau C-1.

Attribut	*Description*	*Domaine de valeur*
risk	Facteur de risque	entier de -3 à 3
normalizedLosses	Perte moyenne due aux sinistres	entier de 65 à 256
make	Marque de la voiture	symbole (exemple alfa-romero, audi, bmw, chevrolet…)
fuelType	Type de carburant	entier {1=diesel, 2=gas}.
aspiration	Turbo	entier {1=std, 2=turbo}
numOfDoors	Nombre de portes	entier {4=four, 2=two}
bodyStyle	Style	symbole {hardtop, wagon, sedan, hatchback, convertible}
driveWheels	Motorisation	entier {1=4wd, 2=fwd, 3=rwd}
engineLocation	Propulsion	entier {1=front, 2=rear}
wheelBase	Empattement	réel de 86.6 à 120.9
length	Longueur	réel de 141.1 à 208.1
width	Largeur	réel de 60.3 à 72.3
height	Hauteur	réel de 47.8 à 59.8
curbWeight	Poids carrosserie	réel de 1488 à 4066
engineType	Type de moteur	symbole {dohc, dohcv, l, ohc, ohcf, ohcv, rotor}
numOfCylinders	Nombre de cylindres	entier {2,3,4,5,6,8,12}
engineSize	Taille du moteur	réel de 61 to 326
fuelSystem	Type d'alimentation	symbole {bbl, 2bbl, 4bbl, idi, mfi, mpfi, spdi, spfi}
bore	Largeur des pistons	réel de 2.54 à 3.94
stroke	Temps du moteur	réel de 2.07 à 4.17
compressionRatio	Compression	réel de 7 à 23
horsepower	Puissance	réel de 48 à 288
peakRpm	Vitesse maximum	réel de 4150 à 6600
cityMpg	Consommation en ville	réel de 13 à 49
highwayMpg	Consommation sur autoroute	réel de 16 à 54
price	Prix	réel de 5118 à 45400

Tableau C-1 : attributs d'une voiture

Ces attributs de base sont alors séparés en deux groupes : la solution (attribut risk) et les indices potentiels (tous les autres attributs). Suivant les points d'ouverture définis dans CBR*Tools (cf. Figure C-2), nous définissons deux classes pour la représentation des cas : la classe CarCase (spécialisation du point d'ouverture CbrCase) et la classe CarCaseSituation (spécialisation du point d'ouverture CompoundIndice).

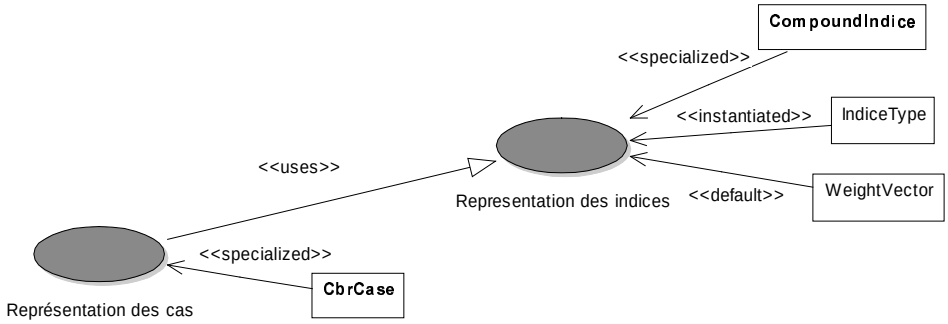


Figure C-2 : utilisation des points d'ouverture pour la représentation des cas

La classe CarCase hérite indirectement de CbrCase par la classe abstraire AbstractCbrCase. Cette classe utilitaire permet d'alléger le codage d'un cas en gérant les opérations de base (affectation d'un identificateur au cas, lien avec la base de cas). De même, la classe CarCaseSituation hérite de CompoundIndice au travers de la classe abstraite JavaClassIndice. Cette dernière implante l'interface d'un indice composite en utilisant une classe Java comme conteneur d'indices grâce aux mécanismes de réflexivité que possède le langage Java.

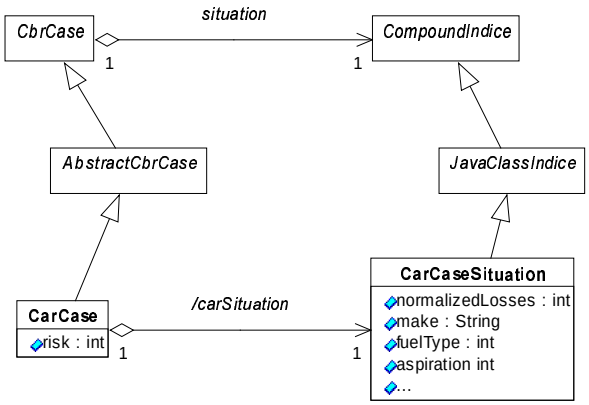


Figure C-3 : représentation des cas

Le code de la classe CarCase comprend la déclaration de deux attributs : l'un sauvegardant le facteur de risque et l'autre donnant les indices.

```
public class CarCase extends AbstractCbrCase {
   public int risk = 0;
   public CarCaseSituation indices = new CarCaseSituation();
...
}
```

Grâce à l'utilisation de la classe JavaClassIndice de CBR*Tools, la représentation des indices est grandement facilitée et s'intègre dans le langage Java. Les sous-indices d'un indice composite sont alors identifiés par des attributs d'une classe Java. Ces attributs doivent être publics pour qu'ils puissent être manipulés directement et efficacement. Toutefois, l'utilisation d'un accesseur (méthode

publique d'accès avec un attribut privé ou virtuel) est tout à fait possible et même nécessaire pour représenter des indices calculés (non utilisés dans cet exemple).

```
public class CarCaseSituation extends JavaClassIndice {
   public int normalizedLosses;
   public String make;
   public float height;
   public int numOfCylinders;
...
}
```

De plus, l'initialisation de la classe permet de déclarer effectivement les indices et de préciser les informations de typage : type (entier, réel, symbole), intervalle pour les attributs entier et réel, domaine de valeurs pour les types liste d'entiers.

```
static private  int _cylinders []  = {2,3,4,5,6,8,12};
static {
      ArrayIndiceDescriptor desc =
        new ArrayIndiceDescriptor("aid.cbr.samples.cars.CarCaseSituation");
     desc.addIndiceDescriptor(new IndiceDescriptor("normalizedLosses", new RangeIntType(65,256)));
     desc.addIndiceDescriptor(new IndiceDescriptor("make", StringType.STRING));
     desc.addIndiceDescriptor(new IndiceDescriptor("height", new RangeFloatType(60,75)));
     desc.addIndiceDescriptor(new IndiceDescriptor("numOfCylinders",new ListIntType(_cylinders)) );
     ...
 }
```

2.2 Organisation de la mémoire

L'organisation de la mémoire consiste à stocker les cas dans une base de cas et à les indexer. Dans ce but, trois points d'ouverture sont spécialisés (cf. Figure C-4) : base de cas (avec la classe CarSimpleFileCaseBase), définition d'une mesure de similarité (classe CarSimilarity) et constitution de la base d'index (classe CarIndexBase).

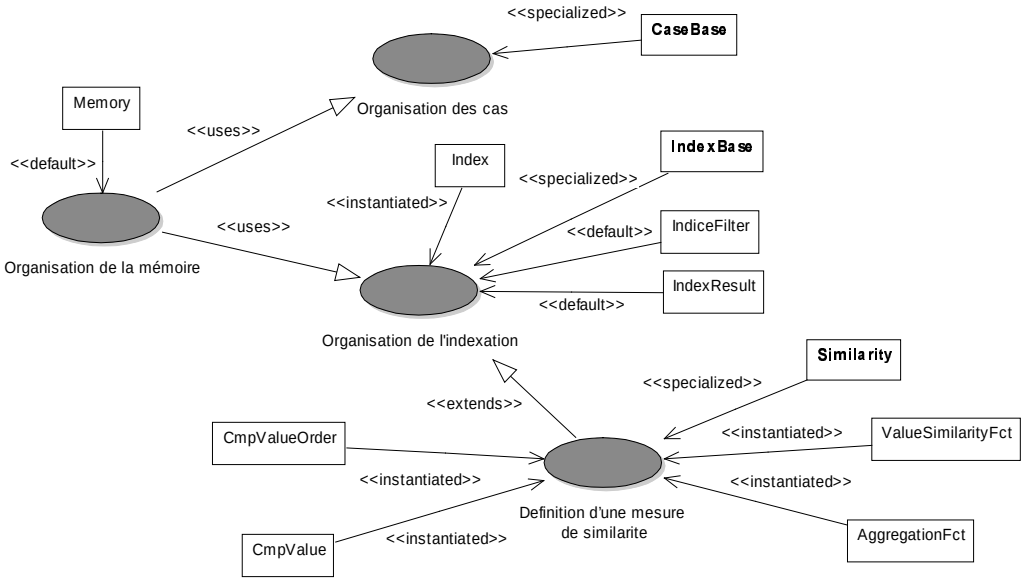


Figure C-4 : utilisation des points d'ouverture pour l'organisation de la mémoire

2.2.1 Base de cas

Dans cet exemple simple, nous ne cherchons pas à structurer la base de cas en sous-bases. Nous utilisons directement la classe abstraite FileSimpleCaseBase de CBR*Tools pour stocker les 205 cas disponibles dans un fichier. Le fichier de la base de cas est alors de la forme suivante :

```
CSV, Jeffrey C. Schlimmer, 19.05.1987, 1985 Auto Imports Database
2,1,168,toyota,2,1,2,hatchback,3,1,94.5,168.7,64.0,52.6,2204.0,ohc,4,98.0,2bbl,3.19,3.03,9.0,70.0,4800.0,29.0,34.0,8238.0
3,1,168,toyota,2,1,2,sedan,3,1,94.5,168.7,64.0,52.6,2169.0,ohc,4,98.0,2bbl,3.19,3.03,9.0,70.0,4800.0,29.0,34.0,8058.0
37,1,?,porsche,2,1,2,hatchback,3,1,98.4,175.7,72.3,50.5,3366.0,dohcv,8,203.0,mpfi,3.94,3.11,10.0,288.0,5750.0,17.0,28.0,?
```

La première ligne du fichier décrit les métadonnées de la base de cas (format, nom du créateur de la base, date de création, description). Ensuite, chaque ligne représente un cas suivant des données séparées par des virgules. La première donnée représente l'identificateur unique du cas, et les suivantes donnent les valeurs des attributs dans l'ordre du Tableau C-1. Les attributs dont les valeurs sont inconnues dans un cas sont marqués par le signe "?". La gestion de cette base de cas se fait très simplement en définissant deux méthodes déclarées abstraites dans la classe FileSimpleCaseBase : transformation d'une ligne en un cas (méthode stringToCase) pour la lecture, et transformation d'un cas en une ligne (méthode caseToString) pour l'écriture. Les autres opérations (ouverture, écriture, accès aux cas) sont prises en charge automatiquement par les super-classes.

2.2.2 Mesure de similarité

La mesure de similarité qui sera utilisée dans l'indexation est définie par la classe CarSimilarity qui hérite indirectement de CompoundIndiceSimilarity par la classe abstraite CompoundArraySimilarity. Cette dernière implante les opérations de base d'une similarité composite en utilisant un tableau de sous-similarités. Le code de la classe CarSimilarity est complètement déclaratif et consiste en l'instanciation de différents objets associant aux indices les fonctions élémentaires de similarité.

Tout d'abord, le constructeur de cette classe permet l'initialisation de la similarité avec une fonction d'agrégation par moyenne pondérée.

```
public class CarSimilarity extends CompoundIndiceArraySimilarity {
   public CarSimilarity () {
      super(new MeanAggregationFct(),25);
```

Ensuite, la structure de la mesure de similarité ainsi que les fonctions élémentaires de similarité sont instanciées et associées aux indices.

```
Class c = JavaClassIndice.getClass("aid.cbr.samples.cars.CarCaseSituation");
SimpleIndiceSimilarity floatAtt = new SimpleIndiceSimilarity(new FloatDomainSimilarityFct());
setSubSimilarity(CarCaseSituation.indexOf(c,"height"),floatAtt);
...
SimpleIndiceSimilarity intAtt = new SimpleIndiceSimilarity(new IntDomainSimilarityFct());
setSubSimilarity(CarCaseSituation.indexOf(c,"normalizedLosses"),intAtt);
setSubSimilarity(CarCaseSituation.indexOf(c,"numOfCylinders"),intAtt);
...
SimpleIndiceSimilarity eqsim = new SimpleIndiceSimilarity(new IdentitySimilarityFct(    new CmpFactorValue(1),
                                                                                         new CmpFactorValue(0)));
setSubSimilarity(CarCaseSituation.indexOf(c,"make"),eqsim);
```

Par exemple, une fonction élémentaire de similarité par matrice symétrique est associée à l'indice bodyStyle. Cette fonction est définie pour retourner la valeur 1 lorsque les valeurs des indices à comparer sont identiques, 0 si l'une d'entre elles n'est pas spécifiée. Puis, chaque couple de valeurs est explicitement défini avec la valeur de similarité correspondante.

```
SymetricExtensionSimilarityFct bodyStyleSim = new SymetricExtensionSimilarityFct(    new CmpFactorValue(0),
                                                                                     new CmpFactorValue(1));
bodyStyleSim.define("hardtop","convertible", new CmpFactorValue((float)0.9));
bodyStyleSim.define("hardtop","hatchback", new CmpFactorValue((float)0.5));
```

```
        bodyStyleSim.define("hardtop","sedan", new CmpFactorValue((float)0.5));
        setSubSimilarity(CarCaseSituation.indexOf(c,"bodyStyle"),new SimpleIndiceSimilarity(bodyStyleSim));
      }
}
```

2.2.3 Base d'index

Pour l'indexation, nous désirons mettre en place deux stratégies : la première utilise un index par plus proches voisins et la deuxième combine un index par hiérarchie de prototypes (filtrage) avec un index par plus proches voisins (sélection). La réalisation de ces deux stratégies est effectuée dans la classe CarIndexBase héritant indirectement de IndexBase par la classe abstraite BasicIndexBase qui implante la gestion de base des différentes stratégies. Le code de la classe CarIndexBase est alors réduit à la déclaration et à la configuration des index nécessaires.

```
public class CarIndexBase extends BasicIndexBase {
   CarIndexBase() {
        // First strategy: knn
        buildAndDeclareKnn();

        // Second strategy: prototype + knn
        declarePrototype();
        declareKnnForPrototype();
        buildProtoKnn();
   }
   ....
}
```

Plus précisément, la construction de la première stratégie est effectuée par l'instanciation de l'index LinearKnnIndex avec en paramètres la fonction d'ordre, la mesure de similarité, et le nombre de cas à retourner lors d'une recherche. D'autres paramètres sont également spécifiés : nom de l'index, commentaires, et valeur à utiliser en cas d'indices inconnus. Enfin, l'index est ajouté à la base.

```
protected void buildAndDeclareKnn() {
        LinearKnnIndex knnIndex = new LinearKnnIndex(new CmpFactorValueOrder(),new CarSimilarity(),5);
        knnIndex.setName("CarCaseKnn");
        knnIndex.setComments("This index uses Knn method to retrieve car cases");
        knnIndex.setUnknownCmpValue(new CmpFactorValue((float)0.5));
        addTopLevelIndex(knnIndex);
   }
```

La deuxième stratégie, plus complexe, s'exprime encore par de simples déclarations. Tout d'abord, l'index par hiérarchie de prototypes est construit par instanciation de l'index PrototypeIndex en indiquant que chaque prototype de la hiérarchie (feuilles et nœuds) doit sauvegarder les cas qui lui sont compatibles lors de l'indexation (classification).

```
protected void declarePrototype() {
        PrototypeIndex protoIndex = new PrototypeIndex();
        protoIndex.setName("ProtoIndex");
        protoIndex.setAddCaseToAcceptedPrototypes(true);
```

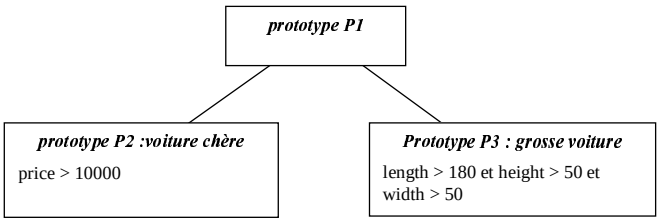


Figure C-5 : hiérarchie de prototypes de la deuxième stratégie d'indexation

Puis les prototypes (cf. Figure C-5) sont déclarés en utilisant les classes anonymes (déclaration de la classe au niveau de l'instanciation) de Java qui sont dans ce cas très pratiques.

```
Prototype p1 = new BooleanPrototype () { // root prototype
   public CmpValue computeMatching(CompoundIndice indices) {
            return CmpBoolValue.TRUE;
   }
};
p1.setName("Root");
protoIndex.addPrototype(p1);

Prototype p2 = new BooleanPrototype () { // expensive cars prototype
   public CmpValue computeMatching(CompoundIndice indices) {
            int priceIndex = ((JavaClassIndice)indices).indexOf("price");
            if (!indices.isUnknown(priceIndex)) {
               float price = indices.getFloatIndice(priceIndex);
               return (price > 10000)?CmpBoolValue.TRUE:CmpBoolValue.FALSE;
            }
            return CmpBoolValue.TRUE;
   }
};
p2.setName("ExpensiveCars");
protoIndex.addPrototype(p2);

Prototype p3 = new BooleanPrototype () { // big cars prototype
   public CmpValue computeMatching(CompoundIndice indices) {
            int lengthIndex = ((JavaClassIndice)indices).indexOf("length");
            int widthIndex = ((JavaClassIndice)indices).indexOf("width");
            int heightIndex = ((JavaClassIndice)indices).indexOf("height");
            if ( !indices.isUnknown(lengthIndex) &&
               !indices.isUnknown(widthIndex) &&
               !indices.isUnknown(heightIndex)) {
                     float length = indices.getFloatIndice(lengthIndex);
                     float width = indices.getFloatIndice(widthIndex);
                     float height = indices.getFloatIndice(heightIndex);
                     return (length > 180 && height > 50 && width > 50)?CmpBoolValue.TRUE:CmpBoolValue.FALSE;
               }
            return CmpBoolValue.TRUE;
   }
};
p3.setName("BigCars");
protoIndex.addPrototype(p3);

// build the tree
p1.addSon(p2);
p1.addSon(p3);
protoIndex.setRoot(p1);

addIndex(protoIndex);
}
```

De manière identique à la première stratégie, un autre index par plus proches voisins est déclaré.

```
protected void declareKnnForPrototype() {
   LinearKnnIndex knnIndex = new LinearKnnIndex(new CmpFactorValueOrder(),new CarSimilarity(),5);
   knnIndex.setName("CarCaseKnnAfterProto");
   knnIndex.setComments("This index uses Knn method to retrieve car cases");
   knnIndex.setUnknownCmpValue(new CmpFactorValue((float)0.5));
   addIndex(knnIndex);
}
```

Enfin, l'index par hiérarchie de prototypes et l'index par plus proches voisins sont combinés grâce à un index composite instance de ToDynamicKnnIndex (cf. Figure C-6)

```
protected void buildProtoKnn() {
    ToDynamicKnnIndex index = new ToDynamicKnnIndex();
    index.setName("Proto+KNN");
    index.addIndex(getIndex("ProtoIndex"));
    index.addIndex(getIndex("CarCaseKnnAfterProto"));
    addTopLevelIndex(index);
}
```

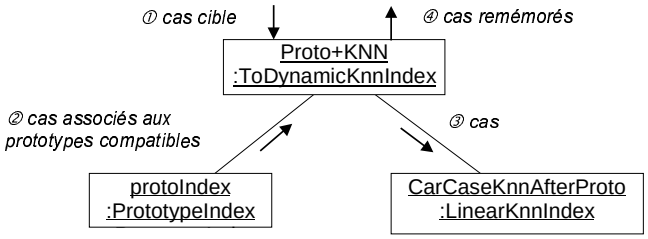


Figure C-6 : structure de la deuxième stratégie d'indexation

2.3 Gestion du raisonnement

La gestion du raisonnement nécessite la spécialisation de quatre points d'ouverture (cf. Figure C-7) : phase du raisonnement (recherche, réutilisation et apprentissage) et la fabrique chargée d'instancier les objets spécifiques de l'application.

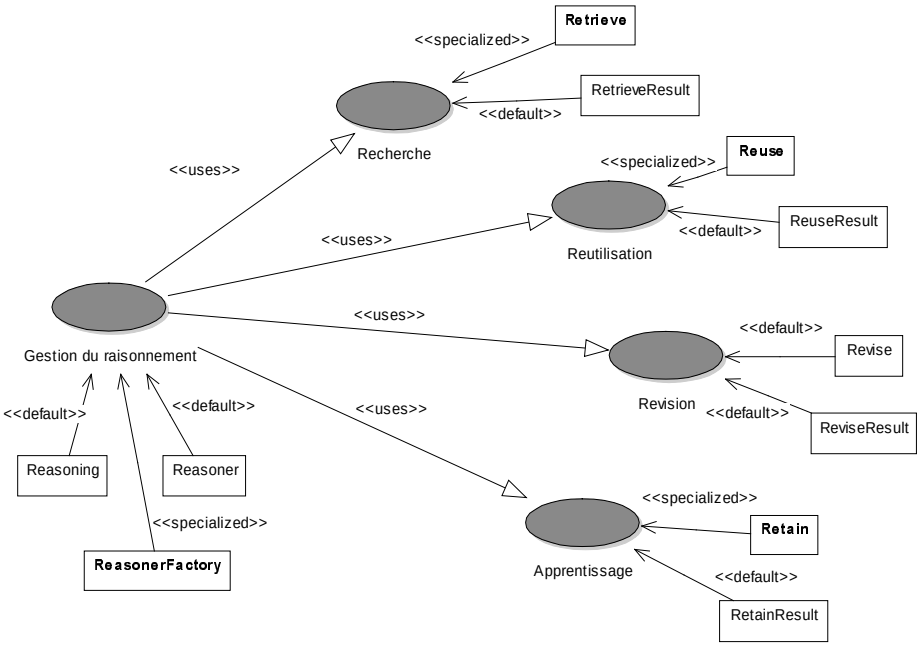


Figure C-7 : utilisation des points d'ouverture pour le contrôle du raisonnement

2.3.1 Recherche

Le protocole de recherche est spécialisé avec la classe CarRetrieve qui redéfinit la méthode updatedRetrieveResult pour transformer les résultats du parcours des index en résultats de la phase de recherche : récupération des identificateurs de cas et transformation de la similarité des cas retrouvés en degré d'utilité pour la phase de réutilisation.

```
public class CarRetrieve extends BasicRetrievePattern {

  protected  void updateRetrieveResult(IndexResult result, RetrieveResult stepResult){
      List res= Assoc.getFactory().newListAssoc();
      res.addAll(((CaseSetIndexResult)result).retrievedCases());
      stepResult.setCases(res);

      for (Iterator it = ((CaseSetIndexResult)result).retrievedCases().iterator();
          it.hasNext(); ) {
         Object id = it.next();
         float f = ((CmpFactorValue)((CaseSimIndexResult)result).getCaseMatchingResults(id)).getFactor();
         ((UsefulnessRetrieveResult)stepResult).setCaseUsefulness(id,f);
      }
  }
}
```

2.3.2 Réutilisation

La phase de réutilisation se base sur le schéma d'adaptation par formule prenant en compte l'ensemble des cas retrouvés. Le protocole de ce schéma (classe abstraite CaseSetFormulaReuse) est spécialisé dans la classe CarReuse. Tout d'abord, les méthodes de lecture et d'écriture de la solution d'un cas sont spécifiées, puis le constructeur initialise l'objet avec une formule d'adaptation par moyenne pondérée tout en prenant soin d'arrondir le résultat qui doit être un entier compris entre -3 et 3 (à noter, encore une fois, l'utilisation d'une classe anonyme de Java).

```
public  class CarReuse extends CaseSetFormulaReusePattern  {

protected  float getSourceCaseSolution(CbrCase c){
      return ((CarCase)c).risk;
  }

protected  void setTargetCaseSolution(CbrCase c, float sol){
      ((CarCase)c).risk=(int)sol;
  }

CarReuse () {
      setFormula( new MeanAdaptationFormula() {
         public float compute(float[] solutions, float[] usefullness) {
                float f = super.compute(solutions,usefullness);
                return Math.round(f);
         }
      });
  }
}
```

2.3.3 Apprentissage

La phase d'apprentissage est réalisée suivant le schéma par chaîne de responsabilité. La classe CarRetain est alors réduite à une simple déclaration de deux analyseurs : le premier examine si la base de cas comporte plus de 250 cas, et le deuxième vérifie s'il existe déjà un cas dont la similarité est très proche (au-dessus de 0,95) du cas cible courant. Ainsi, de manière très flexible et simple, le cas cible courant sera ajouté et indexé dans la mémoire si et seulement si ces deux conditions ne sont pas satisfaites.

```
public  class CarRetain extends ResponsibilityChainRetainPattern {
CarRetain() {
        RetainAnalyser a1 = new CaseBaseSizeRetainAnalyser(250);
        RetainAnalyser a2 = new SimilarityRetainAnalyser(new CmpFactorValue((float)0.95));
        a1.setNextAnalyser(a2);
        setAnalyser(a1);
   }
}
```

2.3.4 Fabrique des objets

Enfin, la fabrique d'objets est spécialisée pour référencer les classes d'objets spécifiquement créées pour l'application.

```
public class CarReasonerFactory extends AbstractReasonerFactory {
   public CbrCase newTagetCase() {
        return new CarCase();
   }
   public CaseBase newCaseBase() {
        return new CarFileSimpleCaseBase("CarCaseBase","cars.txt");
   }
   public Retrieve newRetrieveStep() {
        return new CarRetrieve();
   }
   public Reuse newReuseStep() {
        return new CarReuse();
   }
   public Retain newRetainStep() {
        return new CarRetain();
   }
   public IndexBase newIndexBase() {
        return new CarIndexBase();
   }
}
```

3 Exemples d'exécution

Nous présentons deux exemples d'exécution de raisonnements réalisés dans la classe CarsApp. Ces raisonnements montrent l'utilisation de deux configurations différentes.

3.1 Premier raisonnement

Dans le premier raisonnement, nous lançons l'exécution avec les paramètres par défaut. Tout d'abord, nous instancions un nouveau contrôleur de raisonnement (variable reasoner) depuis la fabrique.

```
ReasonerFactory factory = new CarReasonerFactory();
Reasoner reasoner = factory.getReasoner();
```

Nous pouvons alors ouvrir la base de cas, et l'indexation automatique de l'ensemble des cas issus de la base est effectuée suivant les deux stratégies d'indexation présentes.

```
reasoner.getMemory().getCaseBase().open();
reasoner.getMemory().buildFromCaseBase(new GlobalIndexParams());
```

Puis, un nouveau raisonnement est lancé après avoir instancié et initialisé un cas cible.

```
CbrCase target = factory.newTagetCase();
initialize(target);
```

```
Reasoning reasoning = reasoner.startReasoning(target);
```

L'utilisation de l'objet sauvant l'état du raisonnement est alors très pratique pour effectuer un rapport d'exécution (cf. Tableau C-2).

```
printReport(reasoning);
```

```
started at Mon Jul 13 17:34:05 CEST 1998
done in 460 ms

1/ retrieve done in 240 ms
found 5 cases
caseid=201 usefulness=0.86 risk=2.0
caseid=197 usefulness=0.86 risk=2.0
caseid=184 usefulness=0.86 risk=2.0
caseid=3 usefulness=0.87 risk=1.0
caseid=205 usefulness=0.96 risk=1.0

2/ reuse done in 20 ms
assigned risk by adaptation is 2.0

3/ revise not done

4/ retain done in 140 ms
 target case not added
```

Tableau C-2 : rapport d'exécution du premier raisonnement

3.2 Deuxième raisonnement

Dans le deuxième raisonnement, nous nous proposons d'utiliser la deuxième stratégie d'indexation (hiérarchie de prototypes et plus proches voisins). Nous passons également le nombre maximum de cas à retrouver de 5 initialement à 10. Le rapport (cf. Tableau C-3) montre alors un temps d'exécution divisé par deux par rapport au premier raisonnement pour l'obtention du même résultat.

```
((BasicRetrievePattern)reasoner.getRetrieveStep()).getDefaultParams().
        setSelectedIndex(reasoner.getMemory().getIndexBase().getIndex("Proto+KNN"));
((KnnIndex)reasoner.getMemory().getIndexBase().getIndex("CarCaseKnnAfterProto")).setK(10);
target = factory.newTagetCase();
initialize(target);
reasoning = reasoner.startReasoning(target);
```

```
started at Mon Jul 13 17:34:06 CEST 1998
done in 230 ms

1/ retrieve done in 70 ms
found 10 cases
caseid=201 usefulness=0.86 risk=2.0
caseid=199 usefulness=0.84 risk=2.0
caseid=197 usefulness=0.86 risk=2.0
caseid=184 usefulness=0.86 risk=2.0
caseid=181 usefulness=0.84 risk=3.0
caseid=179 usefulness=0.84 risk=3.0
caseid=3 usefulness=0.87 risk=1.0
caseid=205 usefulness=0.96 risk=1.0
caseid=2 usefulness=0.83 risk=1.0
caseid=203 usefulness=0.86 risk=2.0

2/ reuse done in 10 ms
assigned risk by adaptation is 2.0

3/ revise not done

4/ retain done in 130 ms
 target case not added
```

Tableau C-3 : rapport d'exécution du deuxième raisonnement

4 Code source complet de l'exemple

L'exemple représente la définition de dix classes dont nous donnons l'intégralité du code Java correspondant dans l'ordre alphabétique : CarCase, CarCaseSituation, CarFileSimpleCaseBase, CarIndexBase, CarReasonerFactory, CarRetain, CarRetrieve, CarReuse, CarsApp, CarSimilarity.

4.1 Classe CarCase

```
package aid.cbr.samples.cars;

import aid.cbr.tools.core.cbrcase.CompoundIndice;
import aid.cbr.tools.core.cbrcase.AbstractCbrCase;

public class CarCase extends AbstractCbrCase {

   public int risk = 0;
   public CarCaseSituation indices = new CarCaseSituation();
   public CompoundIndice getPotentialIndices() {
     return indices;
   }
}
```

4.2 Classe CarCaseSituation

```
package aid.cbr.samples.cars;

import aid.cbr.tools.core.cbrcase.*;

public class CarCaseSituation extends JavaClassIndice {

   // these fields are used to store indices
   public int normalizedLosses;
```

```
    public String make;
    public int fuelType;
    public int aspiration;
    public int numOfDoors;
    public String bodyStyle;
    public int driveWheels;
    public int engineLocation;
    public float wheelBase;
    public float length;
    public float height;
    public float width;
    public float curbWeight;
    public String engineType;
    public int numOfCylinders;
    public float engineSize;
    public String fuelSystem;
    public float bore;
    public float stroke;
    public float compressionRatio;
    public float horsePower;
    public float peakRpm;
    public float cityMpg;
    public float highwayMpg;
    public float price;

    static private  int _fuel []  = {1,2};
    static private  int _doors []  = {2,4};
    static private  int _asp [] = {1,2};
    static private  int _cylinders []  = {2,3,4,5,6,8,12};
    static private  int _wheels [] = {1,2,3};
    static private  int _loc [] = {1,2};

    static {
        ArrayIndiceDescriptor desc =
         new ArrayIndiceDescriptor("aid.cbr.samples.cars.CarCaseSituation");
         desc.addIndiceDescriptor(new IndiceDescriptor("normalizedLosses", new RangeIntType(65,256)));
       desc.addIndiceDescriptor(new IndiceDescriptor("make", StringType.STRING));
       desc.addIndiceDescriptor(new IndiceDescriptor("fuelType", new ListIntType(_fuel)));
       desc.addIndiceDescriptor(new IndiceDescriptor("aspiration", new ListIntType(_asp)));
       desc.addIndiceDescriptor(new IndiceDescriptor("numOfDoors", new ListIntType(_doors)));
       desc.addIndiceDescriptor(new IndiceDescriptor("bodyStyle", StringType.STRING ));
       desc.addIndiceDescriptor(new IndiceDescriptor("driveWheels", new ListIntType(_wheels)));
       desc.addIndiceDescriptor(new IndiceDescriptor("engineLocation", new ListIntType(_loc) ));
       desc.addIndiceDescriptor(new IndiceDescriptor("wheelBase", new RangeFloatType(86,121)));
       desc.addIndiceDescriptor(new IndiceDescriptor("length", new RangeFloatType(140,210)));
       desc.addIndiceDescriptor(new IndiceDescriptor("height", new RangeFloatType(60,75)));
       desc.addIndiceDescriptor(new IndiceDescriptor("width", new RangeFloatType(47,60)));
       desc.addIndiceDescriptor(new IndiceDescriptor("curbWeight", new RangeFloatType(1488,4066)));
       desc.addIndiceDescriptor(new IndiceDescriptor("engineType", StringType.STRING));
       desc.addIndiceDescriptor(new IndiceDescriptor("numOfCylinders",new ListIntType(_cylinders)) );
       desc.addIndiceDescriptor( new IndiceDescriptor("engineSize", new RangeFloatType(60,326)));
       desc.addIndiceDescriptor( new IndiceDescriptor("fuelSystem",StringType.STRING));
       desc.addIndiceDescriptor( new IndiceDescriptor("bore", new RangeFloatType(2,4)));
       desc.addIndiceDescriptor( new IndiceDescriptor("stroke", new RangeFloatType(2,5)));
       desc.addIndiceDescriptor( new IndiceDescriptor("compressionRatio", new RangeFloatType(7,23)));
       desc.addIndiceDescriptor( new IndiceDescriptor("horsePower", new RangeFloatType(48,288)));
       desc.addIndiceDescriptor( new IndiceDescriptor("peakRpm", new RangeFloatType(4150,6600)));
       desc.addIndiceDescriptor( new IndiceDescriptor("cityMpg", new RangeFloatType(13,49)));
       desc.addIndiceDescriptor( new IndiceDescriptor("highwayMpg", new RangeFloatType(16,54)));
       desc.addIndiceDescriptor( new IndiceDescriptor("price", new RangeFloatType(5000,50000)));
    }
}
```

4.3 Classe CarFileSimpleCaseBase

```
package aid.cbr.samples.cars;
import aid.cbr.tools.core.memory.casebase.*;
import aid.cbr.tools.core.cbrcase.*;
import java.util.StringTokenizer;
import java.util.NoSuchElementException;

public class CarFileSimpleCaseBase extends FileSimpleCaseBase {
  public CarFileSimpleCaseBase (String name,String filename ) {
      super (name,filename);
  }

  protected void setFloat(String name, JavaClassIndice s, String token){
      int i = s.indexOf(name);
      if (token.equals("?")) {
       s.setUnknown(i);
      } else {
        s.setFloatIndice(Float.valueOf(token).floatValue(),i);
      }
  }
  protected void setInt(String name, JavaClassIndice s, String token){
      int i = s.indexOf(name);
      if (token.equals("?")) {
        s.setUnknown(i);
      } else {
        s.setIntIndice(Integer.parseInt(token),i);
      }
  }
  protected void setSymbol(String name, JavaClassIndice s, String token){
      int i = s.indexOf(name);
      if (token.equals("?")) {
        s.setUnknown(i);
      } else {
        s.setIndice(token,i);
      }
  }

  protected CbrCase stringToCase(String  l) throws CaseBaseAccessException{
      CarCase c= new CarCase();
      CarCaseSituation s = (CarCaseSituation)c.getPotentialIndices();

      StringTokenizer t = new StringTokenizer(l,",");
      try {
        //caseid
        c.setCaseId(new Integer(Integer.parseInt(t.nextToken())));
        // risk
        String token = t.nextToken().trim();
        if (token.equals("?")) throw new CaseBaseAccessException ("Token cannot be unknown");
        c.risk= Integer.parseInt(token);

        // potential indices
        setInt("normalizedLosses",s,t.nextToken().trim());
        setSymbol("make",s,t.nextToken().trim());
        setInt("fuelType",s,t.nextToken().trim());
        setInt("aspiration",s,t.nextToken().trim());
        setInt("numOfDoors",s,t.nextToken().trim());
        setSymbol("bodyStyle",s,t.nextToken().trim());
        setInt("driveWheels",s,t.nextToken().trim());
        setInt("engineLocation",s,t.nextToken().trim());
        setFloat("wheelBase",s,t.nextToken().trim());
        setFloat("length",s,t.nextToken().trim());
        setFloat("height",s,t.nextToken().trim());
        setFloat("width",s,t.nextToken().trim());
        setFloat("curbWeight",s,t.nextToken().trim());
```

```
            setSymbol("engineType",s,t.nextToken().trim());
            setInt("numOfCylinders",s,t.nextToken().trim());
            setFloat("engineSize",s,t.nextToken().trim());
            setSymbol("fuelSystem",s,t.nextToken().trim());
            setFloat("bore",s,t.nextToken().trim());
            setFloat("stroke",s,t.nextToken().trim());
            setFloat("compressionRatio",s,t.nextToken().trim());
            setFloat("horsePower",s,t.nextToken().trim());
            setFloat("peakRpm",s,t.nextToken().trim());
            setFloat("cityMpg",s,t.nextToken().trim());
            setFloat("highwayMpg",s,t.nextToken().trim());
            setFloat("price",s,t.nextToken().trim());

            return c;
        } catch (NoSuchElementException e) {
            throw new CaseBaseAccessException("Token missing");
        }
    }
    protected String caseToString(CbrCase c){
        CarCaseSituation s = (CarCaseSituation)c.getPotentialIndices();
        StringBuffer buff = new StringBuffer(255);
        buff.append(c.getCaseId()); buff.append(",");
        buff.append(((CarCase)c).risk);            buff.append(",");
        if (s.isUnknown(s.indexOf("normalizedLosses"))) buff.append("?"); else  buff.append(s.normalizedLosses);  buff.append(",");
        if (s.isUnknown(s.indexOf("make"))) buff.append("?"); else buff.append(s.make);          buff.append(",");
        if (s.isUnknown(s.indexOf("fuelType"))) buff.append("?"); else buff.append(s.fuelType);       buff.append(",");
        if (s.isUnknown(s.indexOf("aspiration"))) buff.append("?"); else buff.append(s.aspiration);     buff.append(",");
        if (s.isUnknown(s.indexOf("numOfDoors"))) buff.append("?"); else buff.append(s.numOfDoors);     buff.append(",");
        if (s.isUnknown(s.indexOf("bodyStyle"))) buff.append("?"); else buff.append(s.bodyStyle);      buff.append(",");
        if (s.isUnknown(s.indexOf("driveWheels"))) buff.append("?"); else buff.append(s.driveWheels);    buff.append(",");
        if (s.isUnknown(s.indexOf("engineLocation"))) buff.append("?"); else buff.append(s.engineLocation); buff.append(",");
        if (s.isUnknown(s.indexOf("wheelBase"))) buff.append("?"); else buff.append(s.wheelBase);       buff.append(",");
        if (s.isUnknown(s.indexOf("length"))) buff.append("?"); else buff.append(s.length);          buff.append(",");
        if (s.isUnknown(s.indexOf("height"))) buff.append("?"); else buff.append(s.height);          buff.append(",");
        if (s.isUnknown(s.indexOf("width"))) buff.append("?"); else buff.append(s.width);           buff.append(",");
        if (s.isUnknown(s.indexOf("curbWeight"))) buff.append("?"); else buff.append(s.curbWeight);      buff.append(",");
        if (s.isUnknown(s.indexOf("engineType"))) buff.append("?"); else buff.append(s.engineType);     buff.append(",");
        if (s.isUnknown(s.indexOf("numOfCylinders"))) buff.append("?"); else buff.append(s.numOfCylinders); buff.append(",");
        if (s.isUnknown(s.indexOf("engineSize"))) buff.append("?"); else buff.append(s.engineSize);      buff.append(",");
        if (s.isUnknown(s.indexOf("fuelSystem"))) buff.append("?"); else buff.append(s.fuelSystem);     buff.append(",");
        if (s.isUnknown(s.indexOf("bore"))) buff.append("?"); else buff.append(s.bore);            buff.append(",");
        if (s.isUnknown(s.indexOf("stroke"))) buff.append("?"); else buff.append(s.stroke);          buff.append(",");
        if (s.isUnknown(s.indexOf("compressionRatio"))) buff.append("?"); else buff.append(s.compressionRatio);buff.append(",");
        if (s.isUnknown(s.indexOf("horsePower"))) buff.append("?"); else buff.append(s.horsePower);      buff.append(",");
        if (s.isUnknown(s.indexOf("peakRpm"))) buff.append("?"); else buff.append(s.peakRpm);         buff.append(",");
        if (s.isUnknown(s.indexOf("cityMpg"))) buff.append("?"); else buff.append(s.cityMpg);         buff.append(",");
        if (s.isUnknown(s.indexOf("highwayMpg"))) buff.append("?"); else buff.append(s.highwayMpg);      buff.append(",");
        if (s.isUnknown(s.indexOf("price"))) buff.append("?"); else buff.append(s.price);
        return new String(buff);
    }
}
```

4.4 Classe CarIndexBase

```
package aid.cbr.samples.cars;

import aid.cbr.tools.core.cbrcase.*;
import aid.cbr.tools.core.memory.similarity.*;
import aid.cbr.tools.core.memory.index.*;

public class CarIndexBase extends BasicIndexBase {

    CarIndexBase() {
        // First strategy: knn
```

```
        buildAndDeclareKnn();

        // Second strategy: prototype + knn
        declarePrototype();
        declareKnnForPrototype();
        buildProtoKnn();
    }
    protected void buildAndDeclareKnn() {
        LinearKnnIndex knnIndex = new LinearKnnIndex(new CmpFactorValueOrder(),new CarSimilarity(),5);
        knnIndex.setName("CarCaseKnn");
        knnIndex.setComments("This index uses Knn method to retrieve car cases");
        knnIndex.setUnknownCmpValue(new CmpFactorValue((float)0.5));
        addTopLevelIndex(knnIndex);
    }

    protected void declarePrototype() {
        PrototypeIndex protoIndex = new PrototypeIndex();
        protoIndex.setName("ProtoIndex");
        protoIndex.setAddCaseToAcceptedPrototypes(true);

        Prototype p1 = new BooleanPrototype () { // root prototype
          public CmpValue computeMatching(CompoundIndice indices) {
                return CmpBoolValue.TRUE;
          }
        };
        p1.setName("Root");
        protoIndex.addPrototype(p1);

        Prototype p2 = new BooleanPrototype () { // expensive cars prototype
          public CmpValue computeMatching(CompoundIndice indices) {
                int priceIndex = ((JavaClassIndice)indices).indexOf("price");
                if (!indices.isUnknown(priceIndex)) {
                    float price = indices.getFloatIndice(priceIndex);
                    return (price > 10000)?CmpBoolValue.TRUE:CmpBoolValue.FALSE;
                }
                return CmpBoolValue.TRUE;
          }
        };
        p2.setName("ExpensiveCars");
        protoIndex.addPrototype(p2);

        Prototype p3 = new BooleanPrototype () { // big cars prototype
          public CmpValue computeMatching(CompoundIndice indices) {
                int lengthIndex = ((JavaClassIndice)indices).indexOf("length");
                int widthIndex = ((JavaClassIndice)indices).indexOf("width");
                int heightIndex = ((JavaClassIndice)indices).indexOf("height");
                if ( !indices.isUnknown(lengthIndex) &&
                    !indices.isUnknown(widthIndex) &&
                    !indices.isUnknown(heightIndex)) {
                            float length = indices.getFloatIndice(lengthIndex);
                            float width = indices.getFloatIndice(widthIndex);
                            float height = indices.getFloatIndice(heightIndex);
                            return (length > 180 && height > 50 && width > 50)?CmpBoolValue.TRUE:CmpBoolValue.FALSE;
                    }
                return CmpBoolValue.TRUE;
          }
        };
        p3.setName("BigCars");
        protoIndex.addPrototype(p3);
```

```
        // build the tree
        p1.addSon(p2);
        p1.addSon(p3);
        protoIndex.setRoot(p1);

        addIndex(protoIndex);
    }

    protected void declareKnnForPrototype() {
        LinearKnnIndex knnIndex = new LinearKnnIndex(new CmpFactorValueOrder(),new CarSimilarity(),5);
        knnIndex.setName("CarCaseKnnAfterProto");
        knnIndex.setComments("This index uses Knn method to retrieve car cases");
        knnIndex.setUnknownCmpValue(new CmpFactorValue((float)0.5));
        addIndex(knnIndex);
    }

    protected void buildProtoKnn() {
        // connect the indexes
        ToDynamicKnnIndex index = new ToDynamicKnnIndex();
        index.setName("Proto+KNN");
        index.addIndex(getIndex("ProtoIndex"));
        index.addIndex(getIndex("CarCaseKnnAfterProto"));
        addTopLevelIndex(index);
    }
}
```

4.5 Classe CarReasonerFactory

```
package aid.cbr.samples.cars;

import aid.cbr.tools.core.reasoner.*;
import aid.cbr.tools.core.cbrcase.*;
import aid.cbr.tools.core.memory.index.*;
import aid.cbr.tools.core.memory.casebase.*;

public class CarReasonerFactory extends AbstractReasonerFactory {
    public CbrCase newTagetCase() {
        return new CarCase();
    }
    public CaseBase newCaseBase() {
        return new CarFileSimpleCaseBase("CarCaseBase","cars.txt");
    }
    public Retrieve newRetrieveStep() {
        return new CarRetrieve();
    }
    public Reuse newReuseStep() {
        return new CarReuse();
    }
    public Retain newRetainStep() {
        return new CarRetain();
    }
    public IndexBase newIndexBase() {
        return new CarIndexBase();
    }
}
```

4.6 Classe CarRetain

```
package aid.cbr.samples.cars;

import aid.cbr.tools.core.cbrcase.CbrCase;
import aid.cbr.tools.core.reasoner.*;
import aid.cbr.tools.core.memory.similarity.CmpFactorValue;

public  class CarRetain extends ResponsibilityChainRetainPattern {

   CarRetain() {
        RetainAnalyser a1 = new CaseBaseSizeRetainAnalyser(250);
        RetainAnalyser a2 = new SimilarityRetainAnalyser(new CmpFactorValue((float)0.95));
        a1.setNextAnalyser(a2);
        setAnalyser(a1);
   }
}
```

4.7 Classe CarRetrieve

```
package aid.cbr.samples.cars;

import aid.cbr.tools.core.reasoner.*;
import aid.cbr.tools.core.memory.index.*;
import aid.cbr.tools.core.memory.similarity.*;
import com.sun.java.util.collections.*;
import aid.util.assoc.*;

public class CarRetrieve extends BasicRetrievePattern {
   protected  void updateRetrieveResult(IndexResult result, RetrieveResult stepResult){
        List res= Assoc.getFactory().newListAssoc();
        res.addAll(((CaseSetIndexResult)result).retrievedCases());
        stepResult.setCases(res);
        for (Iterator it = ((CaseSetIndexResult)result).retrievedCases().iterator();
            it.hasNext(); ) {
           Object id = it.next();
           float f = ((CmpFactorValue)((CaseSimIndexResult)result).getCaseMatchingResults(id)).getFactor();
           ((UsefulnessRetrieveResult)stepResult).setCaseUsefulness(id,f);
        }
   }
}
```

4.8 Classe CarReuse

```
package aid.cbr.samples.cars;

import aid.cbr.tools.core.cbrcase.CbrCase;
import aid.cbr.tools.core.reasoner.*;

public  class CarReuse extends CaseSetFormulaReusePattern  {

   protected  float getSourceCaseSolution(CbrCase c){
        return ((CarCase)c).risk;
   }

   protected  void setTargetCaseSolution(CbrCase c, float sol){
        ((CarCase)c).risk=(int)sol;
   }
```

```
    CarReuse () {
        setFormula( new MeanAdaptationFormula() {
            public float compute(float[] solutions, float[] usefullness) {
                    float f = super.compute(solutions,usefullness);
                    return Math.round(f);
            }
        });
    }
}
```

4.9 Classe CarsApp

```
package aid.cbr.samples.cars;

import aid.cbr.tools.core.cbrcase.*;
import aid.cbr.tools.core.reasoner.*;
import aid.cbr.tools.core.memory.similarity.*;
import aid.cbr.tools.core.memory.index.*;
import aid.cbr.tools.core.memory.casebase.*;
import aid.cbr.tools.core.memory.*;
import com.sun.java.util.collections.*;
import java.util.Date;

public class CarsApp {
    public static void initialize(CbrCase c)throws  CaseBaseAccessException {
        CaseBase cb = new CarFileSimpleCaseBase("TargetCases","target.txt");
        cb.open(true);
        ((CarCase)c)._indices=(CarCaseSituation)cb.getCase(new Integer(1)).getPotentialIndices();
        cb.close();
    }

    public static void printReport(Reasoning r) throws  CaseBaseAccessException{
        StringBuffer buf = new StringBuffer(250);
        buf.append("\n\n*****Reasoning Report*****\n");
        buf.append("\nstarted at "+new Date(r.getBeginDate()));
        buf.append("\ndone in "+r.duration()+ " ms");

        appendRetrieveReport(r,buf);
        appendReuseReport(r,buf);
        appendReviseReport(r,buf);
        appendRetainReport(r,buf);

        //print the report
        System.out.print(buf);
    }
    public static void appendRetrieveReport(Reasoning r, StringBuffer buf) throws  CaseBaseAccessException{
        RetrieveResult retrieve = r.getRetrieveResult();
        if (retrieve!=null && retrieve.done()) {
            buf.append("\n\n1/ retrieve done in "+retrieve.duration()+ " ms");
            if (retrieve.isCaseRetrieved()) {
                    int nbCases = retrieve.cases().size();
                    buf.append("\nfound "+nbCases+" cases");
                    for (Iterator it = retrieve.cases().iterator();
                        it.hasNext();){
                        Object id = it.next();
                        float f =((UsefulnessRetrieveResult)retrieve).getCaseUsefulness(id);
                        float sol = ((CarCase)r.getMemory().getCaseBase().getCase(id)).risk;
                        buf.append("\ncaseid="+id+" usefulness="+f+ " risk="+sol);
                    }
            } else {
                    buf.append("\nno case found");
            }
        }else {
            buf.append("\n\n1/ retrieve not done");
```

```
        }
    }
    public static void appendReuseReport(Reasoning r, StringBuffer buf){
        ReuseResult reuse = r.getReuseResult();
        if (reuse!=null && reuse.done()) {
            buf.append("\n\n2/ reuse done in "+reuse.duration()+" ms");
            if (reuse.isAdaptationDone()) {
                    buf.append("\nassigned risk by adaptation is "+reuse.getAdaptedSolution());
            } else {
                    buf.append("\nadaptation not performed");
            }
        }else {
            buf.append("\n\n2/ reuse not done");
        }
    }

    public static void appendReviseReport(Reasoning r, StringBuffer buf) {
        ReviseResult revise = r.getReviseResult();
        if (revise!=null && revise.done()) {
            buf.append("\n\n3/ revise done in "+revise.duration()+" ms");
        }else {
            buf.append("\n\n3/ revise not done");
        }
    }
    public static void appendRetainReport(Reasoning r, StringBuffer buf) {
        RetainResult retain = r.getRetainResult();
        if (retain!=null && retain.done()) {
            buf.append("\n\n4/ retain done in "+retain.duration()+" ms");
            if (retain.isTargetCaseRetained())
                    buf.append("\n target case added");
            else
                    buf.append("\n target case not added");
        }else {
            buf.append("\n\n4/ retain not done");
        }
    }

    public static void main(String[] args) {
        try {

            // create the reasoner
            ReasonerFactory factory = new CarReasonerFactory();
            Reasoner reasoner = factory.getReasoner();

            // initialise the case base and the indexes
            reasoner.getMemory().getCaseBase().open();
            reasoner.getMemory().buildFromCaseBase(new GlobalIndexParams());

            // start a reasoning
            CbrCase target = factory.newTagetCase();
            initialize(target);
            Reasoning reasoning = reasoner.startReasoning(target);

            // print the report
            printReport(reasoning);

            // start a second reasoning
            ((BasicRetrievePattern)reasoner.getRetrieveStep()).getDefaultParams().
                    setSelectedIndex(reasoner.getMemory().getIndexBase().getIndex("Proto+KNN"));
            ((KnnIndex)reasoner.getMemory().getIndexBase().getIndex("CarCaseKnnAfterProto")).setK(10);
            target = factory.newTagetCase();
            initialize(target);
            reasoning = reasoner.startReasoning(target);

            // print the report
```

```
            printReport(reasoning);

            reasoner.getMemory().getCaseBase().close();


        } catch (CaseBaseAccessException expt) {
            expt.printStackTrace();
        } catch (MemoryAccessException expt) {
            expt.printStackTrace();
        } catch (ReasoningAbortedException expt) {
            System.out.println("Reasoner exited abormally");
            expt.printStackTrace();
        }
    }

}
```

4.10 Classe CarSimilarity

```
package aid.cbr.samples.cars;

import aid.cbr.tools.core.memory.similarity.*;
import aid.cbr.tools.core.cbrcase.*;

public class CarSimilarity extends CompoundIndiceArraySimilarity {
    public CarSimilarity () {
        super(new MeanAggregationFct(),25);
          Class c = JavaClassIndice.getClass("aid.cbr.samples.cars.CarCaseSituation");

        // indices with float similarity on float Domain
        SimpleIndiceSimilarity floatAtt = new SimpleIndiceSimilarity(new FloatDomainSimilarityFct());
        setSubSimilarity(CarCaseSituation.indexOf(c,"wheelBase"),floatAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"length"),floatAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"height"),floatAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"width"),floatAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"curbWeight"),floatAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"engineSize"),floatAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"bore"),floatAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"stroke"),floatAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"compressionRatio"),floatAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"horsePower"),floatAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"peakRpm"),floatAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"cityMpg"),floatAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"highwayMpg"),floatAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"price"),floatAtt);

          SimpleIndiceSimilarity intAtt = new SimpleIndiceSimilarity(new IntDomainSimilarityFct());
        setSubSimilarity(CarCaseSituation.indexOf(c,"normalizedLosses"),intAtt);
        setSubSimilarity(CarCaseSituation.indexOf(c,"numOfCylinders"),intAtt);

        // indices with equality Similarity
        SimpleIndiceSimilarity eqsim = new SimpleIndiceSimilarity(new IdentitySimilarityFct(    new CmpFactorValue(1),
                                                                                                 new CmpFactorValue(0)));
        setSubSimilarity(CarCaseSituation.indexOf(c,"make"),eqsim);
        setSubSimilarity(CarCaseSituation.indexOf(c,"fuelType"),eqsim);
        setSubSimilarity(CarCaseSituation.indexOf(c,"aspiration"),eqsim);
        setSubSimilarity(CarCaseSituation.indexOf(c,"numOfDoors"),eqsim);
        setSubSimilarity(CarCaseSituation.indexOf(c,"driveWheels"),eqsim);
        setSubSimilarity(CarCaseSituation.indexOf(c,"engineLocation"),eqsim);
        setSubSimilarity(CarCaseSituation.indexOf(c,"engineType"),eqsim);
        setSubSimilarity(CarCaseSituation.indexOf(c,"fuelSystem"),eqsim);

        // Body style
        SymetricExtensionSimilarityFct bodyStyleSim = new SymetricExtensionSimilarityFct(    new CmpFactorValue(0),
```

```
                                                                                    new CmpFactorValue(1));
        bodyStyleSim.define("hardtop","convertible", new CmpFactorValue((float)0.9));
        bodyStyleSim.define("hardtop","hatchback", new CmpFactorValue((float)0.5));
        bodyStyleSim.define("hardtop","sedan", new CmpFactorValue((float)0.5));
        setSubSimilarity(CarCaseSituation.indexOf(c,"bodyStyle"),new SimpleIndiceSimilarity(bodyStyleSim));

    }
}
```

Annexe D : paramètres et algorithmes de Broadway

Cette annexe décrit les différents paramètres de la modélisation de Broadway ainsi que deux algorithmes utilisés : l'instanciation du patron de cas potentiels et le filtrage sur les comportements élémentaires.

1 Paramètres de la modélisation

Pour chaque paramètre nous donnons son nom, une brève description, la valeur actuellement utilisée dans la modélisation.

Nom Paramètre	*Description*	*Valeur choisie*
V	Vecteur des variables d'observation des utilisateurs	(Adresse, Contenu, Evaluation Affichage)
$\nabla()$	Fonction d'égalité de deux URI	Cf. chapitre V
motSignificatif()	Fonction indiquant si un mot est significatif	Dictionnaire de 571 mots anglais et 347 mots français
racine()	Fonction donnant la racine d'un mot	Algorithme de Porter (Porter, 1980)
C	Vecteur de classes de mots d'une page HTML	(MotsClefs = métadonnées, MisEnValeur = titre + sous titres (1,2 et3) + description, Autre)
CP	Vecteur de poids des classes	(0.5, 0.3, 0.2)
$\Delta()$	Fonction donnant la discrimination des mots	$\forall\omega$ un mot, $\Delta(\omega)=1$
m	Nombre de mots clefs maximum pour une page	20
δ_{min}	Seuil de sélection d'un mot	0,7
E	Domaine d'évaluation explicite	{-1,0,1,2}
$a_{\max}$	Ration d'affichage maximum pour être significatif (secondes/mots)	2.4 (10 min/ 250 mots)

Tableau D-1 : paramètres d'observation

Nom Paramètre	*Description*	*Valeur choisie*
l_{min}	Longueur minimale d'une navigation pour appliquer la patron de cas potentiels	3
p	Taille de la restriction en nombre de pages	3
e_p	Ensemble des valeurs d'évaluation pour la sélection des pages passées	{-1,2}
a_p	Seuil d'affichage pour la sélection des pages passées	0.48 (2min/250 mots)
τ_{loc}	Seuil de localisation des pages passées	0,9
φ	Fonction d'agrégation des évaluations explicite (x) et implicite (y)	0 si x=-1 (y/ a_{max}) 0.66 si x=0 et y <= a_{max} 0,5 si x=0 et y > a_{max} 0,66 si x =1 1 si x=2
φ_f	Seuil de sélection des pages recommandées dans le patron de cas potentiels	0,7

Tableau D-2 : paramètres de représentation

Nom Paramètre	*Description*	*Valeur choisie*
k	Nombre de cas maximum à rechercher	20
τ	Seuil de similarité minimum	0,8
S	Vecteur de fonctions de similarité	$(s_{Adresse}, s_{Contenu}, s_{Evaluaution}, s_{Affichage})$
k_r	Nombre de cas maximum à rechercher sur la restriction	100
τ_r	Seuil de filtrage sur la restriction	0,5
MP	Matrice de poids pour la restriction	$\forall$i et $\forall$j, *MP*(i,j)=1
k_c	Nombre de cas maximum à rechercher sur les pages passées	50
τ_c	Seuil de filtrage sur les pages passées	0,5
VP	Vecteur de poids entre les variables pour une page passée	(1,1,1,1)
CP	Vecteur de poids entre la similarité des comportements et le ratio du nombre de comportements	(1, 0.2)
GP	Vecteur de poids pour l'agrégation finale	(0.5, 1, 0.5)

Tableau D-3 : paramètres de recherche

Nom Paramètre	*Description*	*Valeur choisie*
F	Vecteur de fonctions calculant les caractéristiques d'une page pour la réutilisation	cf. Chapitre V
FP	Vecteur de poids entre caractéristiques	(10,5,5,5,2,2)
r	Nombre de recommandations maximum	10
τ_r	Seuil de qualité minimum	0,2

Tableau D-4 : paramètres de réutilisation

Nom Paramètre	*Description*	*Valeur choisie*
e_n	Ensemble de valeurs d'évaluation que doit présenter au moins une page dans une navigation pour être sauvegardée	{-1, 2}
a_n	Seuil d'affichage minimum que doit présenter au moins une page dans une navigation pour être sauvegardée	0.48 (2min/250 mots)
f_{obs}	Taux minimum d'échec de chargement d'une page dans les 15 derniers jours rendant cette page obsolète	90%
τ_{obs}	Seuil de similarité minimum indiquant un changement de contenu, ne rendant pas la page obsolète	0,8
η	Taux minimum de pages obsolètes pour effacer une navigation	25%

Tableau D-5 : paramètres d'apprentissage et de maintenance

2 Algorithmes

Nous présentons les deux algorithmes importants utilisés dans Broadway : l'instanciation du patron de cas potentiels et le filtrage sur les comportements élémentaires. Notre objectif est d'améliorer la compréhension des mécanismes de Broadway et nous ne cherchons pas à décrire une version optimisée de ces deux algorithmes. L'expression de ces algorithmes utilise les fonctions sur les chroniques (cf. chapitre II, 2.1.3.3).

2.1 Instanciation du patron de cas potentiels

La fonction d'instanciation du patron de cas potentiels de Broadway prend comme argument la navigation courante et l'instant courant. De par la nature de l'unité de temps choisie, l'instant courant est un entier qui donne la position de la dernière page affichée dans la navigation. Les comportements élémentaires de la restriction ne sont pas instanciés par le patron. En effet, ces comportements sont directement gérés par les index lors de la recherche.

```
NouveauP (n : Navigation, i :  Entier) {
    CasPotentiel c = CasPotentiel(n,i) ;  // création du cas potentiel à retourner et initialise la composante instantanée
    ComposanteComportementale cc =
        c.Problème.SituationComportementale. ComposanteComportementale; // composante comportementale à construire
    Ensemble pages = ∅; // pages déjà examinées dans les itérations
    ComportementElémentaire ce = nil; // dernier comportement créé
    Entier n = 1; // numero du dernier comportement élémentaire créé

    Pour chaque k de i à 1 {
        URl p = Index(Chr(cc,Adresse),k);        // page
        MotsClefs m = Index(Chr(cc,Contenu),k); // mots clefs
        Entier e = Index(Chr(cc,Evaluation),k); // évaluation
        Réel a = Index(Chr(cc,Affichage),k); // ratio d'affichage
```

```
        si   p ∉ pages  et (e ∈ e_p ou a_max >= a > a_p)
        alors {
            pages = pages + p;
            ComportementElémentaire nce =   ComportementElémentaire(Chr(cc,Adresse), p,Ψ_B(.,Adresse,p,m));
            cc = cc + nce;
            cc= cc+  ComportementElémentaire(Chr(cc,Contenu), m, Ψ_R (.,f(.,n)));
            cc= cc+  ComportementElémentaire(Chr(cc,Evaluation), e, Ψ_R (.,f(.,n)));
            cc= cc+  ComportementElémentaire(Chr(cc,Affichage), a, Ψ_R (.,f(.,n)));
            n = n +  4;
            si ce ≠ nil alors  cc = cc + ContrainteAvant(nce, ce);
            ce=nce;
        }
    }

    pages = ∅ ;
    Pour chaque k de Len(Chr(cc,Adresse)) à i+1 {
        URI p = Index(Chr(cc,Adresse),k);
        si   p ∉ pages et φ( Index(Chr(cc,Evaluation), k), Index(Chr(cc,Affichage), k)) > φ_f
        alors {
            pages = pages + p;
            c.Solution = c.Solution + Recommandation(p, φ( Index(Chr(cc,Evaluation), k), Index(Chr(cc,Affichage), k))) ;
        }
    }
    retourne c;
}
```

2.2 Filtrage sur les comportements élémentaires

Le filtrage sur les comportements élémentaires est une étape complexe que nous précisons : identification des comportements, calcul de la similarité sur les pages puis sélection par plus proches voisins.

```
filtrage(casSources : Ensemble ; casCible : Cas) {
    Queue q = Queue(k_c, τ_c); // queue qui contient les meilleurs cas (plus proches voisins)
    Entier max = maxPages(casSources); // indique le nombre maximum de pages selectionnées dans les cas sources
    Pour chaque c dans casSources {
        // création d'une composante comportementale pour contenir les comportements homologues
        ComposantComportementale cc =
                ComposanteComportementale(casCible.Problème.Enregistrement, casCible.Problème.Instant);
        // ----- identification des comportements homologues
        Pour chaque comportement dans c.Probleme.SituationComportementale.ComportementElémentaires {
            cc=cc+ comportement.Ic(cc); // application de la contrainte de localisation
        }
        // ----- calcul de la similarité
        Réel sim = 0; // similarité
        Reel nombrePage = 0; // nombre de pages de la composante comportementale
        Entier k= 1; // indice d'itération sur les comportements élémentaires
        Tant que k <=  Card(c.Probleme.SituationComportementale.ComportementElémentaires) {
            si cc(k) ≠ [] alors {
                Reel s1= s_Adresse(cc(k), c.Probleme.SituationComportementale.ComportementElémentaires(k));
                Reel s2= s_Contenu(cc(k+1), c.Probleme.SituationComportementale.ComportementElémentaires(k+1));
                Reel s3= s_Evaluation(cc(k+2), c.Probleme.SituationComportementale.ComportementElémentaires(k+2));
                Reel s4= s_Affichage(cc(k+3), c.Probleme.SituationComportementale.ComportementElémentaires(k+3));
                sim = sim + g([s1,s2,s3,s4], VP);
            }
            nombrePage++;
            k= k+ 4;
        }
        sim = sim / nombrePage;
        // ----- agrégation avec le critère du nombre de comportements
```

```
            sim = g ([sim, nombrePage/max], CP);
            q.ajoute(c,sim);
        }
        retourne q;
    }
```

RÉSUMÉ. Dans de nombreux processus de résolution de problèmes (commande, prévision, assistance), la prise en considération des évolutions de variables d'observation est nécessaire. Dans cette thèse, nous proposons de faciliter l'analyse et l'utilisation du *raisonnement à partir de cas* (RàPC) à cette classe de problèmes. Le RàPC est un paradigme de résolution de problèmes basé sur la réutilisation par analogie d'expériences passées. Nous proposons, tout d'abord, le *modèle d'indexation par situations comportementales* permettant de réutiliser des cas suivant les comportements pertinents qui les caractérisent. Ce modèle d'indexation générique comprend un formalisme de représentation ainsi qu'un guide d'utilisation. Il permet la conception d'un meilleur processus de découverte des connaissances : extraction de cas potentiels, sauvegarde et mise à jour des cas potentiels sous la forme de cas concrets, et interprétation facilitée des cas concrets. Ensuite, pour gérer notre modèle et faciliter son utilisation au sein d'un cycle complet de raisonnement, nous proposons une *plate-forme à objets*, appelée CBR*Tools, intégrant des points d'ouverture spécialisables ainsi que des composants de base réutilisables. Notre modélisation repose sur la structuration des points d'ouverture suivant des cas d'utilisation, et sur l'utilisation systématique de patrons de conception permettant d'expliciter et de justifier les décisions de conception. Notre approche met ainsi en évidence des modèles à objets ouverts pour le RàPC et permet la capitalisation d'objets-métier RàPC. Enfin, nous avons appliqué nos travaux pour assister un groupe d'utilisateurs navigant sur le Web. Dans notre système, nommé Broadway, des expériences extraites des navigations passées sont réutilisées pour fournir des recommandations. Broadway est un exemple de notre approche originale d'assistance, dans laquelle nous proposons la réutilisation de comportements utilisateurs observés suivant un ensemble de variables.

MOTS CLEFS. Raisonnement à partir de cas (RàPC), indexation par historiques, plate-forme à objets, points d'ouverture, patrons de conception, recherche d'information sur le Web, système de recommandation.

Scheme and Object-Oriented Framework for Case Indexing by Behavioural Situations: Application to Assisted Web Browsing.

ABSTRACT. In many problem solving activities (control, forecast, assistance), it is necessary to take evolutions of observation variables into account. In this thesis, we propose to facilitate the analysis and the use of *case-based reasoning* (CBR) for this class of problems. CBR is a problem solving paradigm based on the reuse by analogy of past experiences. Firstly, we propose the *indexing scheme by behavioural situations* in order to reuse cases based on their relevant behaviours. This generic indexing scheme provides a representation formalism and guidelines. It allows the design of a better knowledge discovery process: extraction of potential cases, updating of potential cases saved as concrete cases, and facilitated interpretation of concrete cases. Secondly, in order to facilitate the management and the use of our scheme in a complete reasoning cycle, we propose an *object-oriented framework* called CBR*Tools that integrates specialisable hot spots and reusable components. The proposed model is based on use cases to structure hot spots, and on the systematic use of design patterns to comment and justify design decisions. Our approach proposes object-oriented models for CBR and allows the capitalisation of corporate objects for CBR. Finally, we have applied our work to assist a group of users browsing the Web. In our system, called Broadway, extracted experiences from past navigations are reused to provide recommendations. Broadway is an example of our original approach to aiding systems, where we propose to reuse behaviours of users observed through a set of variables.

KEYWORDS. Case-based reasoning (CBR), time series indexing, object-oriented framework, hot spots, design patterns, information retrieval on the Web, recommender system.