# Gestion opérationnelle des connaissances sur les codes

Sabine Moisan, Jean-Louis Ermine

# Gestion opérationnelle des connaissances sur les codes

*Sabine Moisan, Jean-Louis Ermine*

---

Actes du congrès IC'2000 (Ingénierie des Connaissances 2000), Toulouse, 10, 12 mai 2000
et
Ingénierie des Connaissances, (Teulier, R., Charlet, J. et Tchounikine P. éditeurs), Communications, Médias, L'Harmattan, 2005

# Gestion opérationnelle des connaissances sur les codes

**Sabine. Moisan* et Jean-Louis Ermine†**

** INRIA Sophia Antipolis, Sabine.Moisan@sophia.inria.fr*
*† Commissariat à l'Energie Atomique, ermine@cea.fr*

**RESUME** : Cet article propose une vision large de la gestion et de l'utilisation des codes en tant que systèmes, qui prend en compte tout le « continuum » qui existe entre les connaissances des experts et des utilisateurs de codes, les connaissances de leurs concepteurs informatiques et les connaissances permettant l'utilisation opérationnelle de ces codes. Un système de codes synthétise en effet un vaste ensemble de connaissances scientifiques, métier, expérimentales, etc. Elles sont fournies tout au long de la vie du système de codes par divers acteurs, ce qui implique des passages de connaissances nécessaires mais peu formalisés. Le génie logiciel offre des solutions à certaines difficultés dans la gestion des codes, mais ses outils sont limités aux seuls programmes informatiques. Or, les programmes ne permettent pas un accès à toutes les connaissances importantes liées aux codes. Nous définissons un référentiel pour la gestion des codes, depuis leur conception et leur développement jusqu'à leur utilisation, qui couvre toutes ces connaissances. Dans la chaîne de moyens mis en place pour la gestion des codes, nous avons identifié deux points faibles : le passage de la documentation scientifique et technique aux documents de développement (informatiques) et le passage de la documentation utilisateur à l'utilisation opérationnelle. Notre travail porte sur ces deux points et propose de nouveaux moyens, complémentaires à ceux du génie logiciel, pour pallier ces faiblesses. D'une part, les livres de connaissances, contiennent les connaissances et savoir-faire essentiels à la compréhension et la réalisation des tâches supportées par les codes. C'est un nouveau type de documentation associé aux codes qui en facilite la gestion. D'autre part, les outils de pilotage de codes, intègrent des connaissances sur l'utilisation des codes et aident à la réalisation des tâches métiers supportés par ceux-ci. Nous étudions aussi comment des passerelles entre ces types d'outils peuvent faciliter l'automatisation des passages de connaissances délicats entre ces deux points.

**MOTS CLES :** Ingénierie des connaissances, gestion des connaissances, codes de calcul

**ABSTRACT :** This paper proposes an approach to the management and utilization of codes viewed as systems. This approach takes into account the whole "continuum" that exists from the knowledge of experts, computer scientists to the users of the codes: i.e., scientific, business, experimental, etc. knowledge. We define a referential for code management, from their design and development up to their use which encompasses all this knowledge. Inside the chain of possible means for code management, we propose two new techniques, that complement software engineering ones. On the one hand, knowledge books contain the essential knowledge and know-how to understand and achieve the tasks supported by the codes. On the other hand, program supervision tools encapsulate knowledge on code utilization and help achieve these business tasks.

**KEYWORDS :** Knowledge Management on codes, code systems, MASK method, program supervision.

# 1. INTRODUCTION

La gestion de gros codes de calcul est actuellement une source de difficulté pour de plus en plus d'entreprises. Cette gestion est problématique à toutes les étapes de la vie d'un code, depuis sa conception, jusqu'à son utilisation, en passant par ses évolutions. On parlera donc d'une gestion « opérationnelle », car les aspects liés à l'utilisation concrète des codes sont abordés.

On désigne par « codes » des programmes informatiques de taille importante (jusqu'à plusieurs centaines de milliers de lignes) qui sont des ressources essentielles dans des activités de recherche, de conception, de production etc. Cependant, la difficulté évoquée ci-dessus dépasse largement le simple programme informatique; elle provient essentiellement de la complexité du système centré sur ce programme, qui comprend une multiplicité de composants logiciels, d'acteurs, d'outils, d'organisations etc. On parlera donc de « systèmes de codes » pour désigner cet ensemble hétérogène et complexe.

Un système de codes synthétise un vaste ensemble de connaissances : lois, équations, modèles, données de qualification, connaissance des limites de pertinence, expérience d'utilisation, composants logiciels, machines, etc. mais aussi connaissances métiers attachées, en liaison avec l'intégration des codes dans une activité de production ou de recherche. Ces connaissances sont créées et nourries pendant une durée de vie souvent longue - jusqu'à vingt ans - par des acteurs multiples : théoriciens, concepteurs, programmeurs, utilisateurs. Ceci implique des passages de connaissances rarement formalisés. L'absence ou la mauvaise qualité de ces passages peut entraîner des surcoûts en temps et en argent (modification difficile, effet de bord imprévu, mauvaise utilisation).

Le génie logiciel offre des solutions à certaines difficultés, mais ses outils sont limités aux seuls programmes informatiques. Or, les programmes ne permettent pas un accès à toutes les connaissances importantes liées aux codes. De même, les méthodes traditionnelles d'ingénierie des connaissances (comme KADS ou KOD) ne se sont pas penchées, à notre connaissance, sur les tâches expertes et les connaissances propres aux systèmes de codes tels que nous les avons définis. Dans cet article nous proposons une vision large de la gestion et de l'utilisation des codes en tant que systèmes, qui prend en compte tout le « continuum » qui existe entre les connaissances des experts, des concepteurs informatiques et des utilisateurs de codes. De plus, nous proposons des outils, complémentaires à ceux du génie logiciel, pour la gestion des différents aspects de ce continuum, via les livres de connaissances et le pilotage de codes.

# 2. REFERENTIEL POUR LE DEVELOPPEMENT ET L'UTILISATION DES CODES

Les programmes informatiques ne reflètent pas l'intégralité des connaissances mises en jeu pour le développement et l'utilisation de codes de calcul. La documentation qui leur est attachée, les lois physiques ou les équations mathématiques qu'ils implantent, mais aussi le savoir-faire d'utilisation et leurs

objectifs de traitement dans le contexte d'une application particulière, sont autant de facteurs importants pour une gestion efficace des codes.

Les codes, considérés comme des systèmes, peuvent se positionner dans un référentiel plus large que celui proposé en génie logiciel avec le cycle de vie traditionnel des programmes. Nous définissons figure 1 un tel référentiel, proche du référentiel de Sargeant pour les codes de simulation numérique (Gustafson [1998], Larzelere [1998], Picard, Ermine & Scheurer [1999]).

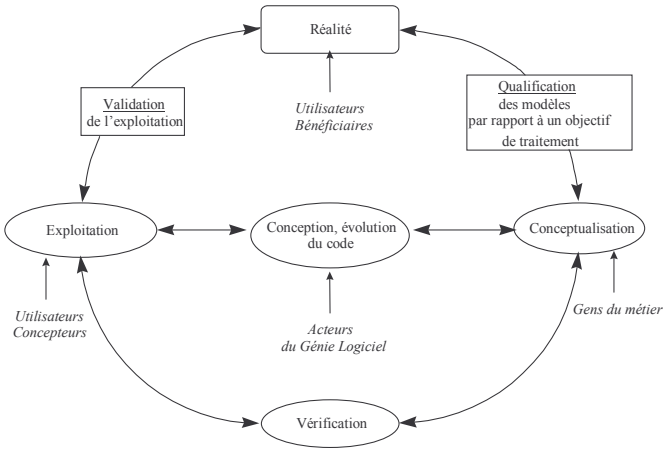


**Figure 1 — Référentiel « à la Sargeant » pour un système de codes**

Le référentiel explicite toutes les phases par lesquelles passe un système de codes, de sa conception jusqu'à son exploitation Un tel référentiel replace le code dans un environnement systémique (le « système de codes ») et permet d'identifier les différent acteurs de ce système. En effet, les codes peuvent être successivement conçus par des ingénieurs, codés par des programmeurs, étudiés par des chercheurs et utilisés par des techniciens. L'activité de ces divers acteurs tout au long de la vie des codes construit un patrimoine de connaissances considérable qui est actuellement peu ou mal exploité.

La *conceptualisation* est l'apanage des gens du métier concerné (physiciens, mécaniciens, ingénieurs, etc.). Elle utilise des modèles qui peuvent être mathématiques (équations différentielles), physiques (lois de comportement), automatiques (équations d'états, lois de commandes) ou digitaux (tableaux de pixels). Ces éléments abstraits permettent de comprendre le comportement du logiciel. La conceptualisation est confrontée au réel (à l'expérimentation dans l'activité), via l'exploitation du code qui permet de qualifier les modèles, à savoir qu'ils représentent bien ce qui intéresse l'activité dans la réalité.

L'*exploitation* est du ressort des utilisateurs du code, mais aussi des concepteurs. L'exploitation se confronte au monde réel dans l'activité concernée et valide qu'elle satisfait les utilisateurs et les bénéficiaires de l'activité afférente au code.

La *vérification* est une liaison entre l'exploitation et la conceptualisation, qui ne passe pas par l'expérimentation réelle. Elle vérifie que les modèles sont bien implantés dans le code.

Ce référentiel montre que le cycle de vie habituel du génie logiciel est essentiellement contenu dans la partie conception et évolution et qu'il ne répond que partiellement à la problématique de la gestion des systèmes de codes.

## 3. POSITIONNEMENT DES OUTILS ET METHODES

Le référentiel précédent permet de positionner les outils et les méthodes dont on dispose actuellement pour la gestion des codes.

Le génie logiciel se positionne essentiellement sur l'activité de conception et la phase de validation. C'est un ensemble de méthodes et d'outils qui couvre le cycle complet d'un logiciel, depuis sa conception jusqu'à la maintenance, essentiellement en tant que programme informatique. A notre connaissance, cette discipline n'aborde pas la problématique générale des systèmes de codes. C'est cependant un point d'entrée incontournable.

La gestion de configuration et le « reverse engineering » sont des approches génie logiciel qui touchent aux phases d'évolution. Ces techniques abordent peu le problème des connaissances mises en jeu dans les codes, en terme de compétences métier utiles dans le développement et l'exploitation des codes.

Le « design rationale » a pour but de décrire le processus de conception d'un artefact, à travers les argumentations logiques qui ont abouti aux choix retenus pour l'artefact (Moran & Caroll [1996]. C'est un élément important pour les connaissances sur les systèmes de codes, car il permet de mettre en perspective et d'argumenter les différents choix de conception effectués pour les différentes parties du logiciel. Il se rattache donc à la phase d'évolution dans le référentiel.

Les techniques d'intelligence artificielle peuvent aussi intervenir à certains points de ce référentiel. La physique qualitative (Bobrow [1984], Trave-Masuyes, Dague & Guerrin [1997]) est par exemple typique de la conceptualisation de problèmes métier (basés sur la physique) qui peuvent être liés à des codes numériques, notamment de simulation. Plus généralement, la représentation des connaissances, utilisée pour des systèmes à base de connaissances liés à des codes de calcul (Troussier, Pourroy, Tollenaere & Trébucq [1999]) est une conceptualisation des métiers qui utilisent des codes.

D'un certain point de vue, un système à base de connaissances classique peut être considéré comme un « système de codes » tel que nous l'avons défini. En ce sens, les méthodes d'ingénierie des connaissances basées sur la modélisation (KADS, KOD, etc.) réalisent un lien entre les méthodes d'acquisition (comme peut être considérée MASK) et ces systèmes de codes, par ce qu'on appelle « l'opérationnalisation des modèles ». Cependant leur problématique est beaucoup plus restreinte, car il s'agit d'un couplage fort entre les modèles et le système. Les modèles ne répondent pas, et ce n'est pas leur rôle, à des objectifs stratégiques de gestion des connaissances, comme la capitalisation et le partage des connaissances en général, et les systèmes de codes fournis n'ont pas d'autres buts que d'opérationnaliser la connaissance formalisée. Cette problématique ne s'inscrit donc pas dans l'intégration d'une chaîne de valeur globale dans l'entreprise. Elle fournit cependant des résultats partiels intéressants.

Enfin, les techniques de pilotage de codes en intelligence artificielle (Thonnat, Moisan & Crubézy [1999], Moisan & Ziébelin [2000]) et les techniques de livre de connaissances venues de la gestion et de l'ingénierie des connaissances (Picard *et al* [1999]) se rattachent aux phases d'exploitation et de conception.

Outre des techniques spécifiques de génie logiciel ou d'intelligence artificielle, il ne faut pas oublier toutes les connaissances de type scientifiques, spécifiques aux applications, qui sont la clé de voûte de la phase de vérification et de l'activité de qualification dans le référentiel.

L'existant offre donc une vision assez morcelée des différents points de vue sur les codes, avec une certaine discontinuité entre des techniques logicielles et/ou d'intelligence artificielle et les points de vue spécifiques aux applications, propres aux métiers concernés par l'utilisation des codes. Il apparaît nécessaire d'avoir une approche plus globale, plus systémique pour faire le lien entre les activités liées aux codes et faciliter les transferts de connaissances entre acteurs. Ceci dans l'optique de faciliter la gestion des codes dans l'ensemble de son système décrit par le référentiel de la figure 1.

## 4. IDENTIFICATION DU PATRIMOINE DE CONNAISSANCES ASSOCIE A UN CODE

La complexité d'un système de codes, et donc la difficulté à le gérer provient de la multiplicité d'acteurs, et de leurs savoir-faire très divers qui ont souvent du mal à interagir (informaticiens, numériciens, physiciens, ingénieurs, etc.). L'hypothèse formulée ici est que la bonne gestion d'un système de codes passe par la gestion des connaissances liées à ce système, au sens où on l'entend maintenant sous le terme général de gestion des connaissances (Knowledge Management).

Dans la méthode MASK, dédiée à la gestion des connaissances, une étape préliminaire est d'identifier le plus précisément possible le patrimoine de connaissances d'une organisation, afin de rendre compte de l'ensemble des acteurs et de leurs connaissances qui doivent être mobilisés dans une gestion globale de ce patrimoine. On utilise pour cela un modèle, dit OIDC (Opération, Information, Décision, Connaissances), qui est le modèle systémique classique de toute organisation, en y ajoutant son « système de connaissances » (Ermine[2000]).

Une telle analyse a été faite pour les systèmes de codes dans le cadre de la simulation numérique (Picard *et al* [1999]). Elle peut se faire dans d'autres domaines : par exemple, un système de codes de traitement d'images peut être associé à un patrimoine de connaissances comportant des connaissances sur :

- les prétraitements d'images,
- l'interprétation d'images,
- le métier concerné,
- l'informatique,
- le traitement d'images proprement dit : (stratégies de traitement, techniques de traitement, algorithmes),
- etc.

Cet exemple montre que l'ensemble des connaissances nécessaires pour des codes de traitement d'image est vaste, très diversifié et dépasse largement celles que l'on prend en compte habituellement : par exemple les stratégies de traitement d'images sont laissées à l'intuition ou au savoir-faire du spécialiste. De même, les connaissances métiers (qui concernent tout système qui a des utilisateurs) sont rarement intégrées dans les systèmes de codes eux-mêmes.

La gestion de ces connaissances sur les systèmes de codes consiste donc à intégrer des méthodes et des outils dans une démarche globale qui permette de prendre en compte l'ensemble du patrimoine de connaissances liées au système afin d'optimiser les performances, à la fois en terme d'efficacité et d'ergonomie cognitive, les produits finaux, qu'ils soient logiciels ou autres (documentation, par exemple).

L'objectif de nos travaux est donc d'assurer par différents moyens la compréhension et la maîtrise d'un ensemble de codes d'un point de vue global, en suivant les codes depuis leur conception jusqu'à leurs utilisations. Chaque aspect particulier doit être pris en compte, sans cloisonnement entre les points de vues reliés entre eux qui nécessitent des passages de connaissances.

## 5. CHAINE DE MOYENS POUR LA GESTION DES SYSTEMES DE CODES

La gestion des connaissances autour d'un système de codes s'organise autour de son cycle de vie élargi (au sens du référentiel décrit dans la figure 1). On y rencontre différents acteurs qui correspondent à différentes problématiques (formation, aide à la conception, aide à l'utilisation). L'objectif étant de considérer la gestion de cette chaîne dans son ensemble, il faut fournir des outils pour les différentes problématiques. Certains manques ayant été identifiés dans cette optique lors des projets que nous avons menés; nous proposons de compléter les outils existants, grâce aux livres de connaissances (hérités de la méthode MASK (Ermine [2001]) et aux techniques de pilotage de codes (Thonnat & Moisan [2000], Moisan & Ziébelin [2000]). Nous détaillons ci-après les différents moyens disponibles pour gérer ces connaissances.

- Les documents scientifiques et techniques, les publications, etc. fournissent la base de la connaissance nécessaire à l'utilisation des systèmes de codes. Ce sont les connaissances les plus « objectives », dans le sens où elles peuvent être déconnectées du contexte d'utilisation ou des métiers considérés. Ce sont souvent les plus disponibles et les plus naturelles (mais pas les plus faciles) à écrire.

- Les livres de connaissances contiennent les connaissances et les savoir-faire essentiels à la compréhension et la réalisation des tâches supportées par les codes. Ils font le lien entre documentation scientifique et technique et documentation de développement ou utilisateur. C'est un nouveau type de documentation associé aux codes qui en facilite la gestion (*cf.* section 6.1).

- La documentation de développement est régie par les procédures et les règles du génie logiciel.

- Les outils généraux de développement. C'est une immense panoplie d'outils de toutes sortes que fournit actuellement l'industrie du logiciel.

- Les outils de pilotage de codes (voir section 6.2) intègrent des connaissances sur l'utilisation des codes et sont des aides à la réalisation des tâches métiers supportés par ceux-ci. Dans ce sens, ils réalisent une gestion opérationnelle des connaissances (d'utilisation, si tant est qu'on peut séparer cette catégorie des autres) des systèmes de codes.

- Les documentations utilisateur et les interfaces homme-machine, qui aident de manière indirecte la réalisation des tâches attachées aux codes.

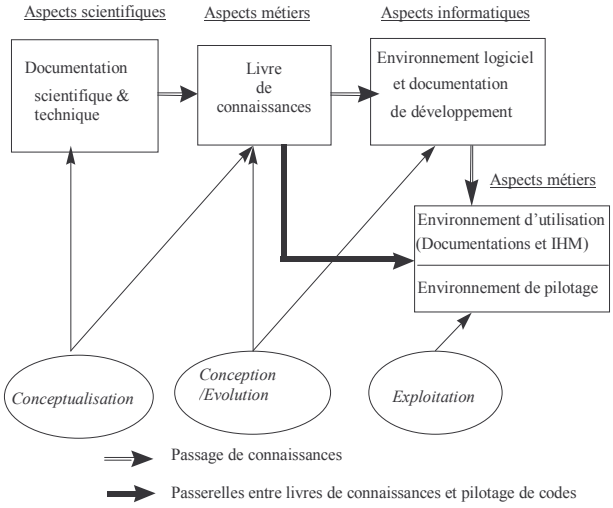


**Figure 2 — Chaîne de moyens pour la gestion des connaissances sur un système de codes**

C'est donc toute une chaîne de moyens qui se met en place, relativement au référentiel des systèmes de codes de la figure 1. Cette chaîne, résumée figure 2, ne peut pas consister en une simple juxtaposition de moyens. La cohérence, la complémentarité et même une certaine redondance doit être assurée pour répondre à l'objectif de la maîtrise globale d'un système de codes. Dans cette chaîne, nous avons identifié deux points faibles : le passage de la documentation scientifique et technique aux documents de développement (informatiques) et le passage de la documentation utilisateur à l'utilisation opérationnelle. Notre travail porte sur ces deux points et propose de nouveaux moyens pour pallier ces faiblesses : les livres de connaissances d'une part, et les outils de pilotage d'autre part. De plus, des passerelles entre ces types d'outils peuvent faciliter l'automatisation des passages de connaissances délicats entre ces deux points.

## 6. DEUX NOUVEAUX MOYENS POUR LA GESTION DES CONNAISSANCES SUR LES CODES

Nous proposons d'utiliser deux types de techniques bien maîtrisées et arrivées à maturité et d'étudier les possibilités de passage de connaissances automatiques

entre ces techniques, pour éviter la perte ou la distorsion d'informations entre la documentation et l'exploitation opérationnelle.

## 6.1. Livres de connaissance sur les codes

Les livres de connaissances sont des produits réalisés avec la méthode de gestion des connaissances MASK (Ermine [2001]). MASK est une méthode d'analyse préalable à la mise en place d'un système opérationnel de gestion des connaissances, c'est-à-dire un système, informatisé ou pas, qui contient des connaissances, des savoir-faire et dont le but est soit de faire partager, soit de capitaliser, soit de servir à la création de la connaissance dans l'organisation. MASK procède par recueil des connaissances auprès des « sources de connaissances » de l'entreprise. Il s'agit essentiellement des détenteurs du savoir : experts, spécialistes, ou de documents de références (avec des personnes capables de les expliquer). C'est un processus difficile à organiser à une échelle industrielle, puisqu'il nécessite, outre des interviews nombreux, des actions de mobilisation, de mise en cohésion, de consensus etc. MASK est basée sur la modélisation des connaissances. MASK n'utilise pas *un* mais *des* modèles, qui correspondent à des points de vue différents sur un domaine, considéré comme un *système de connaissances* à part entière (comme les systèmes de codes).

Le premier résultat obtenu dans un projet MASK est un ensemble de modèles formalisant la connaissance, élaborés notamment pendant les entrevues avec les détenteurs de cette connaissance. Les modèles graphiques sont agrémentés de fiches descriptives synthétiques et de divers documents de synthèse : scientifiques, conseils, retour d'expérience, références bibliographiques, documentaires ou logicielle, etc. rédigés par les experts, les documentalistes, les équipes concernées. A cet ensemble sont rattachés les documents essentiels du domaine (fiches types, procédures, publications de référence etc.). Le résultat final est appelé *livre de connaissances*. C'est une synthèse structurée des connaissances sur un domaine, avec les pointeurs adéquats vers les sources d'information détaillées s'y rattachant, c'est une sorte d'« encyclopédie métier ». En ce sens, MASK et le livre de connaissances fournissent une explicitation (partielle) et une structuration d'un sous-ensemble d'un patrimoine de connaissances, avec des liens forts au système d'information. Autour d'un livre de connaissances peuvent s'organiser un grand nombre d'actions de gestion de connaissances : formation, valorisation de fonds documentaires, veille technologique, outils spécifiques, etc. Sa diffusion, son évolution (et sa confidentialité) peuvent être gérées par des outils informatisés (on parlera alors de « livre de connaissances électronique » diffusé par intranet, lié aux sources d'informations existantes). L'évolution des livres de connaissances est un problème crucial, actuellement à l'étude pour formaliser les premiers retours d'expérience.

La méthode MASK de conception de livre de connaissances a été adaptée aux systèmes de codes par S. Picard (Picard *et al* [1999]) et appliquée à des grands codes de simulation de centrales électriques classiques pour EDF ou à des codes de simulation numérique du CEA.

Cette méthode est complémentaire des activités de développement sur les codes déjà existantes. Elle a pour but de fournir aux utilisateurs et concepteurs des

documents plus élaborés. Ces documents visent une description du logiciel selon différents niveaux de granularité (du code en général, jusqu'aux algorithmes particuliers), différents points de vue « métiers » (théoriciens, expérimentateurs, numériciens) et différentes échelles de temps pour appréhender l'évolution du code. Elle organise des modèles pour décrire un patrimoine de connaissances lié à un système de codes. Ces modèles sont élaborés à partir soit de documents techniques, soit d'entretiens auprès de différents experts (physiciens, numériciens, informaticiens, etc.). Ils ont servi de référentiel de travail pour élaborer une structure documentaire (diagrammes, fiches, références aux documents techniques et scientifiques, etc.) qui est le livre de connaissances du système de codes. Les modèles qui permettent de représenter les différents points de vue suffisant à rendre compte de la complexité du patrimoine de connaissances autour d'un système de codes sont structurés par un outil appelé le macroscope de la connaissance (Picard *et al* [1999]).

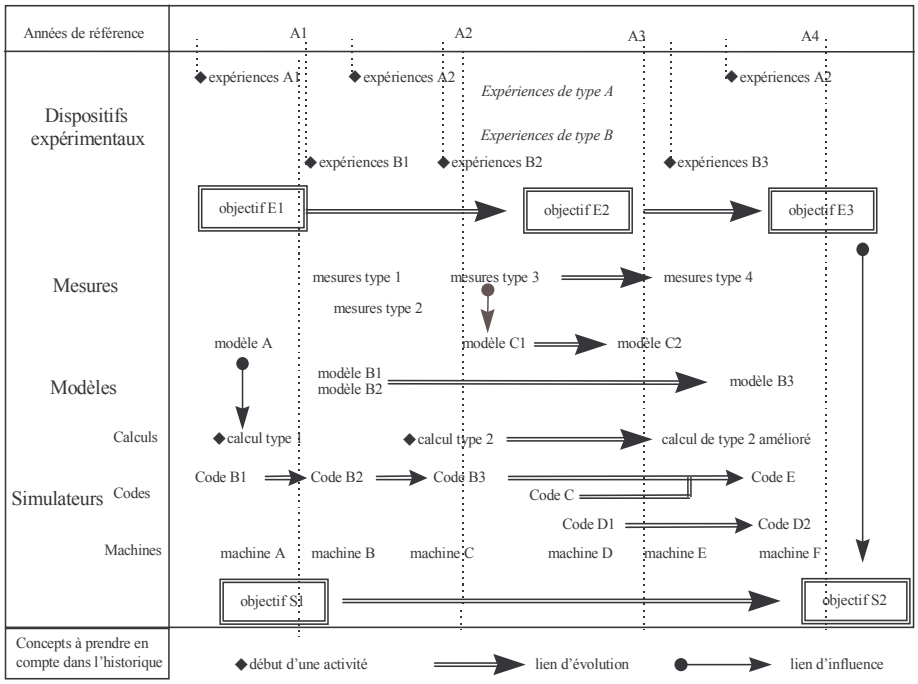


**Figure 3 — Modèle d'historique pour l'évolution d'un système de codes**

Deux points de vue sont dominants. Le *point de vue du contexte*, permet de décrire le contexte des codes : les activités autour des codes dans l'organisation (qui résultent du référentiel de la figure 1), différentes visions métier des phénomènes (physiques, physico-chimiques etc.) qui sont pris en compte dans la conception ou l'utilisation des codes, et l'*historique* décrivant dans les grandes lignes l'évolution du système dans le temps, en fonction des évolutions des matériels, des techniques, des expérimentations, des objectifs etc. ainsi qu'un historique de la conception d'un code de calcul. Les deux premiers modèles sont les modèles classiques de la méthode MASK. La figure 3 donne un exemple de modèle de contexte utilisé pour décrire l'historique.

Les dispositifs expérimentaux conçus et (ou) exploités au cours du temps rendent compte de la réalité physique (dans le sens du référentiel de la figure 1). Les différentes mesures effectuées ont été interprétées à partir du type d'instrumentation mis en œuvre sur les dispositifs, et des modèles physiques pris en compte à cette période. En général, plusieurs modèles physiques sont considérés pour étudier des phénomènes physiques (ils constituent le modèle conceptuel au niveau du référentiel). Ces modèles sont mis en œuvre dans des codes de calcul et interviennent dans la compréhension des résultats expérimentaux

Par exemple, de nouveaux types d'instrumentation dans les dispositifs peuvent amener à reconsidérer l'emploi d'un modèle utilisé dans les codes de calcul et à effectuer des modifications au niveau de ce modèle (sur la figure, les mesures de type 3 ont entraîné une évolution du modèle physique C). De même, l'augmentation de la puissance de calcul des machines, permettent de mettre en œuvre différemment certains modèles physiques (par exemple, les modèles B1 et B2 ont été remplacés par le modèle B3). Les éléments pris en compte comprennent les codes de calcul, les machines sur lesquelles ils sont exploités, et les types de calculs qui sont réalisés à partir des codes. L'acquisition de nouvelles machines permettent de développer et d'exploiter des codes de calculs plus puissants et mettent en œuvre de nouveaux modèles physiques (par exemple, différentes versions du code B se succèdent, puis ce code et le code C sont remplacés par un nouveau code: le code E).

Le *point de vue cognitif* cherche à décrire le savoir-faire spécifique d'un acteur autour du code, notamment les tâches d'utilisation du code dans la résolution d'un problème particulier (cf. fig.7) et *les concepts* correspondant à sa vision métier (classification des concepts utiles à la compréhension du code dans la classe de problème donnée). Ces modèles sont classiques dans la méthode MASK. On décrit aussi, en terme d'évolution des codes ou des ses composants logiciels, les lignées successives qui se sont succédées dans le temps, l'évolution des versions étant, dans le modèle, justifiée par des critères, des avantages et des inconvénients. Un exemple de ce dernier modèle est donné en figure 4. Ce diagramme présente, sur plusieurs années, les différentes versions du code qui ont fait date. On peut indiquer à partir de quel(s) code(s), la première version d'un nouveau code a été élaborée (par exemple, sur la figure, la première version du code Y est une évolution du code X) et les codes qui ont été intégrés (par exemple, un code d'étude à été intégré à la deuxième version du code Y). Entre deux versions du code, on rappelle les principales améliorations qui ont été apportées au code, ces améliorations peuvent être détaillées par chaque option du code sur un autre diagramme. Pour les choix de conception, on présente (a posteriori) les choix de conception effectués ainsi qu'une argumentation de ces choix. On indique en premier lieu quel était l'objectif recherché et quel a été le but atteint lors de cette phase de conception de la version. Ces choix sont décomposés en trois niveaux :

- Modèle physique (choix du modèle physique à mettre en œuvre, dans quelle option du code).
- Traitement numérique utilisé pour discrétiser le modèle physique (choix d'une technique ou méthode, maillage, schéma...).

- Développement informatique (choix et/ou contraintes de programmation: traitement des données, fonctionnalités spécifiques utilisées...).

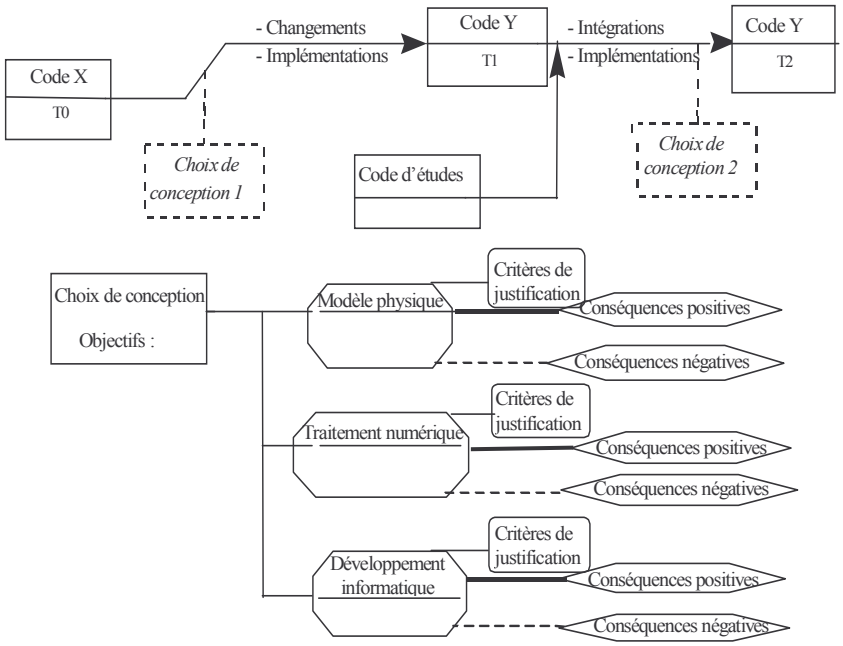


**Figure 4 — Les lignées d'évolution d'un système de codes**

Le diagramme permet d'argumenter les différents choix de conception, il reprend des techniques élaborées dans le cadre du design rationale (Moran & Caroll [1996]) pour décrire la conception d'un artefact. On représente donc sur ce diagramme, les critères justifiant ces choix et les conséquences liées à ces choix. Les flèches (trait plein ou pointillé) indiquent si la conséquence est un avantage ou un inconvénient que l'on peut attribuer à ce choix.

## 6.2. Le pilotage de codes

Les techniques de pilotage de codes (Thonnat & Moisan [2000], Chien & Mortensen [1996], Parmentier, Ziébelin & Rechenman [1998]) ont pour objectif d'automatiser (totalement ou en partie) l'utilisation optimale d'un ensemble codes préexistants, indépendamment d'un domaine applicatif particulier. En effet, quel que soit le domaine, la sélection et l'assemblage des codes les mieux adaptés à un problème et leur application correcte correspond à la même forme de résolution de problème, bien qu'il existe des variantes. Les systèmes de pilotage prennent en charge les étapes nécessaires à la construction d'une chaîne de traitement qui résout l'objectif d'un utilisateur. Le but du pilotage n'est pas d'optimiser les codes eux-mêmes, mais leur *utilisation* qui peut être délicate pour un utilisateur novice. En effet, les codes ne s'appliquent souvent que sous certaines conditions, qui dépendent par exemple de la nature ou du contenu de leurs données d'entrée. Leur syntaxe d'appel peut être complexe et leur

exécution nécessite souvent la détermination préalable de valeurs de paramètres d'entrée ou le formatage de certaines données. Les systèmes de pilotage permettent de stocker ce type de connaissances sur l'utilisation des codes et de les appliquer afin de produire le meilleur enchaînement de programmes possible, en s'adaptant aux données de l'utilisateur. Ce savoir-faire intégré dans une base de connaissances permet ensuite d'appliquer les programmes dans des systèmes (semi-)automatiques robustes. La motivation des systèmes de pilotage est double : gestion de l'utilisation opérationnelle de codes et capitalisation des compétences sur cette utilisation.

D'une part, *la gestion automatique* de l'utilisation de codes est motivée par la complexité croissante de ceux-ci pour des applications dans des domaines variés. Par exemple, des applications en médecine, biologie, astronomie, etc. utilisent souvent des codes de traitement d'images. Les systèmes de pilotage déchargent, partiellement ou totalement, un utilisateur de la gestion concrète des codes. Ils assurent la sélection des meilleurs codes, leur ordonnancement, la gestion des flots de données entre codes, la détection de problèmes et les retours en arrière nécessaires, en tenant compte des particularités des algorithmes, de leurs implantations, des formatages intermédiaires des données, des syntaxes d'appel, des paramètres à régler, etc. Ils offrent des capacités de reconfiguration et d'adaptation qui sont particulièrement cruciales lors d'utilisations dans des chaînes de traitement automatiques. Un dialogue avec l'utilisateur peut parfois s'établir sur des points relevant de sa compétence afin de guider le processus de pilotage (par exemple pour évaluer la qualité d'un résultat).

D'autre part, l'utilisation de techniques de pilotage permet de constituer une *mémoire consultable des compétences* sur l'utilisation de codes existants. Cela répond à un besoin réel des entreprises qui, en cas d'indisponibilité ou de départ des personnes possédant ce savoir faire, souhaitent le conserver sous une forme compréhensible et si possible opérationnelle. Ce type de connaissance accessible sur les programmes et leur utilisation facilite, en particulier, les re-conceptions et les modifications qui, sans cela, sont difficiles et coûteuses.

Les techniques à base de connaissances, adoptées pour réaliser les systèmes de pilotage permettent d'atteindre ces deux objectifs. Une base de connaissances en pilotage contient principalement les représentations des codes eux-mêmes et de (certaines de) leurs configurations possibles, connues des experts (Clouard, Elmoataz & Revenu [1998], Moisan & Ziébelin [2000]). Le contenu de ces représentations doit être suffisant pour permettre au moteur de sélectionner les codes, d'initialiser les valeurs de leurs paramètres, de gérer les flots de données et de produire un enchaînement de codes adapté aux données de l'utilisateur et à ses contraintes. Les représentations des codes existants (dites primitives) contiennent des informations permettant l'exécution des programmes associés. Elles sont souvent connectées à des représentations des arguments en entrée et en sortie de ces programmes, décrivant chaque argument par son type, son domaine de validité ou son format. Les représentations des configurations prédéfinies (dites composites) peuvent se décomposer de diverses façons, en sous-niveaux de plus en plus concrets (séquences, alternatives, spécialisation etc.). Elles contiennent des informations permettant de raffiner un enchaînement pour passer d'un haut niveau d'abstraction vers des niveaux inférieurs.

Le mécanisme de résolution adopté par le moteur émule le comportement d'un expert lors de l'utilisation des codes. Il est souvent à base de techniques de planification (Gong & Kulikowski [1995], Dalle & Dejean [1998]). : l'enchaînement de codes à trouver est vu comme un plan. On trouve aussi des techniques de classification (Willamowski [1994]) ou d'instanciation de hiérarchies de tâches (Bodington [1995]). Les nécessaires points de décisions dans la résolution, sont aussi représentés par l'expert dans la base de connaissances, via des critères indiquant comment décider d'un échec ou d'un problème, comment régler les valeurs des paramètres des codes, comment choisir entre des alternatives, etc. On les trouve parfois aussi sous forme de dialogues prédéfinis avec l'utilisateur (Bodington [1995]). Ces critères sont utilisés par le moteur pour diriger son raisonnement (Clément & Thonnat [1996]) de façon éventuellement totalement automatique. Ils confèrent aux systèmes de pilotage flexibilité et robustesse face à des changements de situations. Certains systèmes travaillent plus en interaction avec l'utilisateur (Willamowski [1994]), qui indique alors lui-même les réparations à faire.

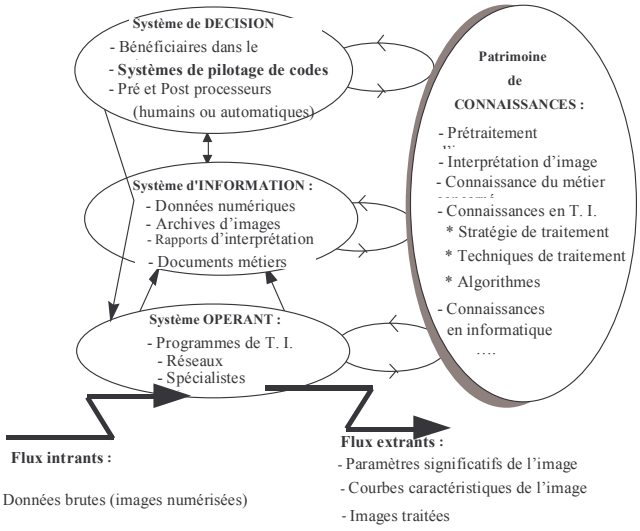


**Figure 5 — Modèle OIDC d'un système de codes de traitement d'images**

En reprenant l'analyse selon le modèle OIDC, évoquée en section 4, on voit qu'une grande partie du patrimoine de connaissances identifié est intégrable dans un système de pilotage (figure. 5). Par exemple, les prétraitements s'incluent dans les décompositions de configurations prédéfinies, les connaissances métier peuvent être prises en compte dans la mesure où elles interviennent dans l'utilisation des codes, les stratégies sont représentées par les types de décompositions, etc. De plus, les techniques de pilotage rendent ces connaissances opérationnellement utilisables.

## 6.3. Apports réciproques des livres de connaissances et du pilotage

Les livres de connaissances correspondent au point de vue « conception » du code. C'est actuellement un support papier, dont informatisation est possible, car ce document est très structuré. Cette informatisation pourra générer un format de stockage des connaissances accepté par des outils de pilotage. En effet, les livres contiennent une grande partie de la connaissance incorporée dans une base de connaissances en pilotage. Par exemple, les diagrammes de tâches correspondent aux décompositions de haut niveau des représentations composites de configurations connues, les concepts se retrouvent dans les représentations des arguments des codes et les descriptions informatiques des options des codes se retrouvent dans les représentations primitives des codes eux-mêmes.

Le pilotage correspond au point de vue « utilisation » des codes, qui vient après la réalisation des livres de connaissances et où les connaissances de conception sont « compilées » pour devenir opérationnelles. Une connaissance plus pratique, liée à l'expérience d'utilisation est aussi nécessaire, elle peut provenir du point de vue cognitif dans les modèles MASK, aussi bien que directement des experts. En sens inverse, des évolutions dans une base de connaissances en pilotage peuvent être répercutées automatiquement dans les livres de connaissances associés. Par exemple, des évolutions de version du code, ou des ajouts de nouveaux codes, indiqués par un changement dans la base de connaissances vont enrichir le modèle d'historique.

## 6.4. Exemple

Nous allons montrer sur un exemple l'étendue et la diversité du patrimoine de connaissances associé à un code. L'exemple choisi est un code de simulation d'un phénomène physique sur les matériaux, appelé « flambage d'Euler ». Le problème physique est d'étudier les déformations mécaniques d'une structure physique soumise à une charge et à des conditions aux limites (par exemple imposées par des fixations). Le flambage permet, par approximation linéaire des équations physiques modélisant les déformations mécaniques, de déterminer les bifurcations dans ces équations (*i.e.* les points de passage d'un mode à un autre). Cela correspond à un calcul de valeurs propres qui représentent des valeurs critiques dans l'augmentation de la charge sur la structure physique.

Sur cet exemple on dispose de diverses connaissances. Il existe tout d'abord des *documents* scientifiques et techniques (publications par exemple) qui présentent la méthode. De plus, la description de ce qui est utile à connaître dans l'utilisation du code et le savoir-faire d'utilisation ont été *modélisés* à partir d'interviews d'experts.

On donne en figures 6 et 7 un exemple simplifié d'un *réseau de concepts* (modèle MASK) qui structure les données d'entrée et de sortie, et un exemple partiel de tâche d'utilisation.

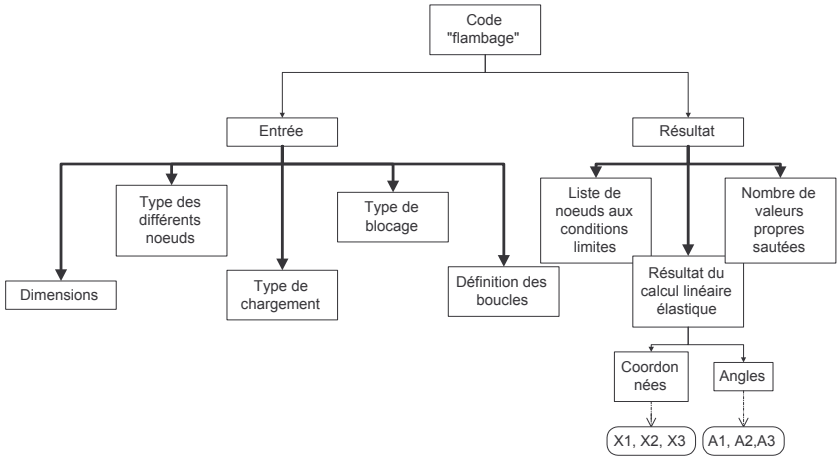


**FIG. 6 — Un réseau de concepts pour un code de flambage**

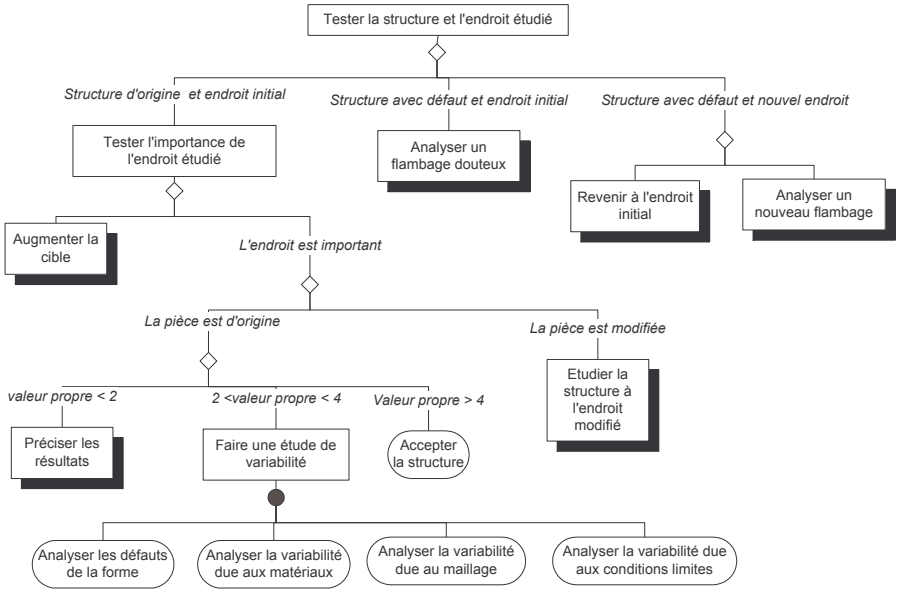


**Figure7 — Une tâche d'utilisation d'un code de flambage**

On peut trouver dans Picard *et al* [2000] des modèles de connaissances qui décrivent la physique étudiée dans de tels codes. Ces modèles, édités dans le *livre de connaissances* du code, participent à sa maîtrise de la part des utilisateurs et à l'implantation des fonctions de pilotage.

Une *base de connaissances* de pilotage pour ce problème comporte tout d'abord la description du code « flambage » lui-même. La voici sous forme textuelle, selon la syntaxe de notre système (les mots-clefs sont en gras) :

**Primitive name** flambage
**Functionality** flambage_linéaire
**Input Data**

Structure_Physique **name** struct_étudiée

**Input Parameters**

Integer **name** : v0

**comment** "charge initiale de recherche''

**Output Data**

Float **name** : vp

**comment** "coefficient critique de charge le plus proche de v0''

Integer **name** nb_vp

**comment** ''nombre de coefficients critiques de charge positifs''

Il s'agit d'une description primitive correspondant à un code réel, donnée par l'expert. L'expert doit aussi définir les arguments du code nécessaires, comme ici *Structure_Physique*, qui décrit la structure à étudier par un objet structuré possédant des attributs comme *matériau*, *géométrie*, *nombre de dimensions* ou *conditions aux limites*.

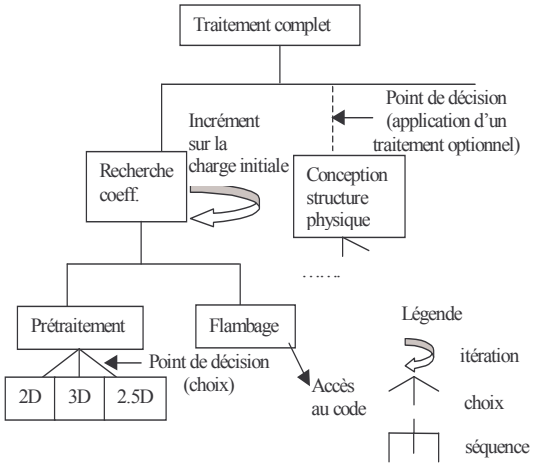


**Figure 8 — Une hiérarchie de la base de connaissances de pilotage pour le flambage.**

La base de connaissances comporte aussi des descriptions composites représentant des traitements plus abstraits. Par exemple, la structure physique étudiée qui est fournie en entrée, peut être décrite en deux ou trois dimensions (2D ou 3D) ou de manière axisymétrique (2,5D ou 2D Fourier). On retrouve ici le concept de dimension du réseau de la figure 6. Le code « flambage » admet ces trois types de dimension, mais le prétraitement à réaliser dépend de la dimension. Il existe pour cela une description composite (voir fig. 8) qui réalise le choix adéquat grâce à des règles de décision et dirige le raisonnement vers le bon prétraitement. Un traitement complet comprend ainsi plusieurs étapes, dont le code de flambage lui-même n'est qu'un élément. En simplifiant, ce traitement commence par la recherche des coefficients critiques, puis si l'un d'entre eux est inférieur à 1, une conception de la structure physique (ajouts de raidisseurs ou d'épaississeurs), etc. Ces connaissances, stratégiques, décisionnelles et structurelles sont résumées dans la hiérarchie de pilotage montrée figure 8.

Les connaissances associées à ce code, pourtant simple, sont donc très diverses et sont gérées via différents outils, qui doivent fonctionner de façon complémentaire et cohérente.

## 7. CONCLUSION ET PERSPECTIVES

La maîtrise et l'utilisation de codes complexes nécessitent la compréhension d'un grand nombre de connaissances qui vont des connaissances scientifiques et techniques jusqu'aux savoir-faire d'utilisation en passant par la conception, l'expérimentation et la programmation. Dans une organisation où un grand nombre de personnes développent et/ou utilisent des codes, cela nécessite un ambitieux programme de gestion des connaissances, ayant pour objectif de ne pas rompre la « chaîne du savoir » entre les scientifiques, les concepteurs et les utilisateurs du métier. Deux outils nouveaux peuvent aider à la maîtrise des connaissances et à l'utilisation dans les métiers : les livres de connaissance et le pilotage de codes. Ces outils se situent à des points différents de la « chaîne du savoir », et sont en continuité l'un de l'autre. Le premier structure des connaissances qui sont utilisées opérationnellement par le second.

La synergie entre les deux outils a été identifiée lors de programmes de recherche communs entre le CEA et l'INRIA. Il reste à mettre en place une méthode complète de gestion des connaissances permettant de dériver des livres de connaissance aux fonctions de pilotage et à prototyper un atelier supportant cette méthode.

## 8. RÉFÉRENCES


BODINGTON R. [1995], "A Software Environment for the Automatic Configuration of Inspection Systems", in INRIA (ed.) *Proceedings KBUP'95*, Sophia Antipolis, France, 99-108.

BOBROW D.G. [1984], *Qualitative reasoning about physical systems* ed. by Daniel G. Bobrow, North-Holland.

CHIEN S.A. & MORTENSEN H.B.[1996], "Automating Image Processing for Scientific Data Analysis of a Large Image Database", in *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 18 (8), 854-859.

CLÉMENT V. & THONNAT M. [1996], "A Knowledge-Based Approach to The Integration of Image processing Procedures". in *Computer Vision, Graphics and Image* 57(2). 854-859.

CLOUARD R, ELMOATAZ A. & REVENU M [1998], « Une modélisation explicite et opérationnelle de la connaissance de traitement d'images », in *Actes du congrès RFIA'98, 11ème congrès Reconnaissance des formes et intelligence artificielle*, Clermont-Ferrand, II-65-74.

DALLE P. & DEJEAN. P [1998], « Planification en traitement d'images : approche basée sur les données », in *Actes du congrès RFIA'98, 11ème congrès Reconnaissance des formes et intelligence artificielle*, Clermont-Ferrand , II-75-84.

ERMINE J-L. [2000], *Les systèmes de Connaissances*, Hermès, seconde édition.

ERMINE J-L. [2001], Capitaliser et partager les connaissances avec la méthode MASK, *Traité* IC2 (Information, Commande, Communication), *Volume « Ingénierie* et *Capitalisation des Connaissances »*, Hermès.

GONG L. & KULIKOWSKI C.A. [1995], "Composition of Image Analysis Processes Through Object-Centered Hierarchical Planning", IEEE Transactions on Pattern Analysis and Machine Intelligence, no 10, vol. 17, oct. 1995.

GUSTAFSON J. [1998], "Computational Verifiability and Feasibility of the ASCI Program", *IEEE Computational Science & Engineering*, 36-45, Janvier - Mars

LARZELERE A.R.98], "Creating Simulation Capabilities"*, IEEE Computational Science & Engineering*, p. 27-35, Janvier - Mars

MORAN T. P.&.J. M. CARROLL (Ed.) [1996], *Design Rationale: Concepts, Techniques and Use*, Lawrence Erlbraum Associates, Hillsdale, NJ.

MOISAN S. & ZIEBELIN D.[2000], « Résolution de problèmes en pilotage de programmes », in *RFIA 2000,* Paris France,.*III-387-396.*

PARMENTIER T., ZIEBELIN D.& RECHENMANN F.[1998], « Environnement de résolution de problèmes distribué », in *RFIA'98, 11ème congrès Reconnaissance des formes et intelligence artificielle*, Clermont-Ferrand, II-265-274.

PICARD S., ERMINE J-L & SCHEURER B.[1999], « Gestion des connaissances pour des grands logiciels de calcul scientifique », *IC'99, Ingénierie des Connaissances,* Palaiseau, France, 171-180.

THONNAT M. & MOISAN S. [2000], "What Can Program Supervision do for Software Reuse?", IEE Proceedings-Software, Special Issue on Knowledge Modeling for Software Components Reuse, oct., (147(5), 163-167..

THONNAT M., MOISAN S. & CRUBEZY M. [1999], "Experience in Integrating Image Processing Programs", in *International Conference on Computer Vision Systems*, Las Palmas, Canaries.

TRAVE-MASSUYES L. , DAGUE P. & GUERRIN F. [1997], *Le Raisonnement qualitatif pour les sciences de l'ingénieur*, Hermès.

TROUSSIER N., POURROY, F., TOLLENAERE M.& TRÉBUCQ, B.[1999], "Information Structuring for Use and Reuse of Mechanical Analysis Models", in *Engineering Design, Journal of Intelligent Manufacturing*, 10 (1).

WILLAMOWSKI J.[1994] *Modélisation de tâches pour la résolution de problèmes en coopération avec l'utilisateur*. thèse, Université J. Fourier, Grenoble.