

**Introduction d'une Base de
Connaissances de type
tâche/méthode pour assister la
conception de modeleurs déclaratifs**

—
Rapport de Recherche IRIN-166
—

Emmanuel DESMONTILS & Francky TRICHET

IRIN

Université de Nantes & École Centrale de Nantes

2, rue de la Houssinière - BP 92208

44322 Nantes cedex 03

FRANCE

décembre 1997

Table des matières

1	Introduction	2
2	La modélisation déclarative et le projet <i>CordiFormes</i>	5
2.1	Le processus de construction d'un modeleur déclaratif sous <i>CordiFormes</i> .	6
2.1.1	Définition des objets de la scène	6
2.1.2	Affectation des méthodes de génération	6
2.2	Nécessité de nouveaux outils à intégrer à la plate-forme	10
3	Construction de la Base de Connaissances à l'aide de DSTM	12
3.1	Justification de l'utilisation de DSTM	12
3.2	Les principes de modélisation de la Base de Connaissances	13
3.2.1	Les primitives de modélisation tâche et méthode adoptées	13
3.2.2	Le mécanisme de sélection adopté	15
3.3	Intégration au projet <i>CordiFormes</i>	18
4	Discussion	20
4.1	Bases de Connaissances de type tâche/méthode Versus Base de règles de production	20
4.2	Vers la prise en compte de nouvelles connaissances de sélection	21
4.3	Utilisation de la Base de Connaissances lors de la phase de <i>description</i>	21

Chapitre 1

Introduction

L'objectif de la *modélisation déclarative de scènes* [1] (aussi appelée *cooperative computer aided design* [2] ou *generative computer aided design* [3]) est de permettre, dans un domaine d'application spécifique, d'engendrer des scènes par la simple donnée d'un ensemble de propriétés ou de caractéristiques appelé *description*. Cette approche a donné naissance à de nombreux projets dont le contrôle spatial (projet VoluFormes [4] et NALIG [5]), la modélisation de courbes (projet CurviFormes [6]), la modélisation de sites mégalithiques (projet MégaFormes [7]), la modélisation de logements (projets «LaHave House» [8] et FLATS [2]) ou de fenêtres (GENWIN [9]), etc.

Les différents modeleurs utilisés dans ces travaux utilisent des techniques et des outils similaires lors de la phase de *génération* de la scène (c'est-à-dire lors du calcul des solutions par exploration de l'univers des scènes potentielles et sélection de celles satisfaisant la description). Cependant, aucun travail de synthèse et d'uniformisation des techniques utilisées n'a jusqu'à présent été proposé. L'objectif du projet CordiFormes [10] est de capitaliser ces différents savoirs-faire acquis en modélisation déclarative. Cette capitalisation repose sur le développement d'une bibliothèque d'objets et de méthodes de génération réutilisables caractérisant une *ontologie* du domaine de la modélisation et de la synthèse d'images¹. La finalité recherchée au travers cette capitalisation est de faciliter la tâche de conception d'un modeleur déclaratif et plus précisément de supporter le conseil sur le choix des meilleures techniques de génération à mettre en œuvre pour les différents objets manipulés.

Une première proposition d'une telle bibliothèque est actuellement intégrée dans la plate-forme de développement de modeleurs déclaratifs proposée par CordiFormes. Cette bibliothèque recense un ensemble d'objets auxquels sont associés des listes exhaustives d'algorithmes de génération. Dans ce cadre, la plate-forme permet au concepteur² (1) de

1. Une ontologie est l'ensemble des objets reconnus comme existant dans un domaine d'application particulier [11]. Dans le cadre du projet CordiFormes, l'ontologie considérée reflète les différents objets classiquement manipulés en synthèse d'images ainsi que les différentes techniques actuellement reconnues performantes à la génération de ces objets.

2. Dans ce rapport, nous utilisons le terme *concepteur* pour désigner la (ou les) personne(s) cherchant à construire un modeleur déclaratif à l'aide de la plate-forme de développement proposée par CordiFormes et le terme *utilisateur* pour désigner la (ou les) personnes cherchant à obtenir des formes à l'aide du

définir les différents objets manipulés par le modelleur (c'est-à-dire les concepts caractérisant le domaine d'application considéré) par construction ex-nihilo et/ou sélection et configuration à partir de composants réutilisables issus de la bibliothèque, (2) d'associer pour chaque objet un mode de génération (c'est-à-dire un algorithme de construction du concept) par introduction d'algorithmes spécifiques et/ou sélection et configuration d'algorithmes prédéfinis et (3) de construire un premier prototype du modelleur escompté.

Cependant, dans sa version actuelle, cette plate-forme ne permet pas de supporter le processus de *sélection* et de *configuration* des composants réutilisables issus de la bibliothèque. En effet, les objets et les algorithmes recensés ne sont pas décrits à l'aide de connaissances relatives aux conditions d'utilisation des algorithmes et/ou aux modes de représentation adoptés pour les objets³. De ce fait, la seule exploitation possible de la bibliothèque consiste à lister exhaustivement les différents algorithmes de génération associés à l'objet sélectionné par le concepteur. En d'autres termes, la description actuelle de la bibliothèque n'offre pas les moyens suffisants à la plate-forme d'aider le concepteur sur le choix de la meilleure technique de génération à utiliser, ni même de vérifier si l'algorithme adopté est techniquement applicable et pertinent pour l'objet considéré.

La solution que nous proposons consiste à représenter les différents composants réutilisables de la bibliothèque à l'aide d'une Base de Connaissances de type tâche/méthode. Les objectifs recherchés au travers cette Base de Connaissances sont (1) de représenter explicitement les caractéristiques intrinsèques des algorithmes de génération recensés dans l'ontologie (prérequis nécessaires au bon fonctionnement de l'algorithme, complexité, etc.), (2) de supporter le conseil des modes de génération des objets grâce à un mécanisme de sélection dédié et (3) de supporter la vérification de la cohérence des modes de génération adoptés pour les objets.

La notion de *tâche* est utilisée pour caractériser un objectif d'affectation d'un mode de génération d'un objet O_i . Une tâche est définie par un ensemble de pré-conditions caractérisant le type d'objet à générer (par exemple une tâche de génération d'un objet de type *couleur* ou *texture*) et par un ensemble de méthodes de génération connues a priori. La notion de *méthode* est utilisée pour caractériser une technique de génération spécifique à un objet particulier (par exemple une méthode de génération de couleurs pâles) ou générale et réutilisable pour tout objet (par exemple une méthode de génération aléatoire suivant une loi de distribution normale). Une méthode est définie par un contexte de sélection et un contexte favorable caractérisant respectivement quand il est *possible* et *préférable* de l'utiliser. Les connaissances associées au contexte de sélection traduisent des contraintes techniques du type "*une génération par énumération n'est possible que si le domaine de valeurs possibles de l'objet O_i considéré est borné et discret*". Les connaissances associées au contexte favorable caractérisent l'ensemble des solutions qu'il est possible d'obtenir par la méthode et permettent de prendre en compte les choix adoptés par le concepteur. Par exemple, si le concepteur désire que le futur modelleur délivre toutes les solutions possibles

modelleur déclaratif construit.

3. Dans sa version originelle, la bibliothèque est composée d'objets uniquement décrits par leurs noms supposés suffisamment explicite pour dénoter la fonctionnalité de l'algorithme sous-jacent et/ou la sémantique de l'objet représenté.

pour le concept C_i , il est préférable d'utiliser une méthode de type énumération plutôt qu'une méthode de type aléatoire, car une méthode de génération aléatoire ne garantit pas l'exhaustivité des solutions.

Un mécanisme de sélection dédié exploite ces connaissances pour guider le concepteur dans ses choix de modes de génération pour les objets de la scène. Ce mécanisme consiste tout d'abord à identifier un ensemble de méthodes candidates à la génération de l'objet O_i considéré (utilisation du *contexte de sélection* des méthodes comme critère de sélection), puis à identifier d'éventuelles méthodes favorables parmi les méthodes candidates (utilisation du *contexte favorable* comme critère de sélection). Une méthode est alors proposée au concepteur et justifiée (le système met en évidence les connaissances qui lui ont permis (i) de confirmer la méthode sélectionnée et (ii) d'écarter les autres méthodes possibles). Le concepteur peut suivre le conseil ainsi donné ou opter pour une autre méthode de génération. Dans ce dernier cas, le système vérifie si la méthode M_i proposée par le concepteur est cohérente, c'est-à-dire (1) si M_i est applicable aux propriétés de l'objet O_i considéré (domaine borné, domaine discret, etc.) et (2) si M_i n'introduit pas d'incompatibilité avec des méthodes déjà affectées pour d'autres objets de la scène en relation avec O_i .

Une première version de cette Base de Connaissances a été développée à l'aide de l'outil DSTM. DSTM (Dynamic Selection of Tasks and Methods) est un environnement de construction de Systèmes à Base de Connaissances (SBC) fondés sur des architectures de type Tâche/Méthode [12] [13]. Le choix de DSTM pour implanter cette Base de Connaissances se justifie de part sa flexibilité et son principe de représentation explicite de toutes les connaissances (c'est-à-dire des tâches, des méthodes et des mécanismes de sélection associés). DSTM permet de respecter un principe de *correspondance structurelle* entre les connaissances modélisées (au niveau conceptuel) et leur représentation informatique (au niveau symbolique). En d'autres termes, DSTM permet de *modéliser* et d'*opérationnaliser* la Base de Connaissances de façon simultanée.

Le principal objectif de ce rapport est de montrer l'intérêt d'introduire une Base de Connaissances de type tâche/méthode (classiquement utilisée dans les travaux réalisés en Acquisition des Connaissances [14]) pour assister la tâche de conception d'un modèleur déclaratif.

En Section 2, nous présentons (succinctement) le principe de construction d'un modèleur déclaratif proposé par le projet CordiFormes et posons les définitions de base nécessaires à la mise en œuvre de notre approche de *sélection* et de *vérification de la cohérence* des modes de génération des objets de la scène. En section 3, nous présentons les principes de modélisation adoptés pour la Base de Connaissances de type tâche/méthode et justifions l'intérêt d'utiliser l'outil DSTM comme support à l'implantation de cette dernière. En Section 4, nous discutons sur le choix d'une Base de Connaissances de type tâche/méthode plutôt qu'une base de règles de production et nous envisageons la prise en compte de nouvelles connaissances dans les critères de sélection des méthodes de génération. Nous concluons cet article en mettant en évidence les intérêts potentiels de cette même Base de Connaissances non plus lors de la phase de *conception* mais lors de la phase d'*utilisation* d'un modèleur déclaratif.

Chapitre 2

La modélisation déclarative et le projet *CordiFormes*

Dans une approche traditionnelle de la modélisation en synthèse d'image, l'utilisateur doit, à partir de spécifications abstraites mises à sa disposition (propriétés géométriques, physiques ou topologiques), concevoir et traduire, sous forme de modèles numériques compréhensibles par le système, tous les objets qu'il souhaite manipuler. L'intérêt de la modélisation déclarative est dégager l'utilisateur d'une telle charge de programmation et de calculs longs et complexes. Libéré de toutes contraintes techniques, l'utilisateur peut alors focaliser son travail sur la phase de création.

L'utilisation d'un modèleur déclaratif comporte trois phases [1, 15, 16, 17]:

1. *la description de la scène* où l'utilisateur décrit ce qu'il souhaite obtenir¹;
2. *la génération*, c'est-à-dire le calcul des solutions par le modèleur qui est chargé d'explorer l'univers des scènes potentielles afin de sélectionner celles correspondant à la description;
3. *la prise de connaissance* des solutions obtenues où l'utilisateur choisit, à l'aide d'outils appropriés, la ou les scènes générées satisfaisant sa description².

Ces phases et leurs éléments constitutifs ont été étudiés lors de nombreux travaux. Une étude exhaustive sur les modèleurs déclaratifs est proposée dans [16]. Une synthèse du point de vue des utilisateurs ainsi qu'une étude des processus de conception à l'aide de modèleurs déclaratifs sont proposées dans [17].

1. Les moyens utilisés pour décrire la scène dépendent du domaine d'application considéré. Une description peut être énoncée en langage naturel parlé ou écrit, formulée à l'aide de schémas ou dessins, etc.

2. Si aucune solution n'est jugée satisfaisante, l'utilisateur peut raffiner la description initiale et recommencer la génération.

2.1 Le processus de construction d'un modeleur déclaratif sous *CordiFormes*

Le processus de construction d'un modeleur déclaratif proposé par le projet CordiFormes repose essentiellement sur l'enchaînement des deux phases suivantes : (1) définition des objets de la scène et (2) affectation des modes de génération pour chacun des objets de la scène.

2.1.1 Définition des objets de la scène

La définition des objets de la scène repose sur la notion de *concept*. On distingue deux types de concepts : les concepts *terminaux* et les concepts *non-terminaux*. Un *concept terminal* est un concept ne faisant référence qu'à un *domaine*. Le domaine d'un concept terminal, noté $[Bm, BM]_u$, se caractérise par une liste de valeurs possibles, une unité et un ensemble de *propriétés de base* (par exemple $\{\}$, $\{\text{vérifié}\}$, $\{\text{faible, moyen, important}\}$ ou $\{\text{petit, moyen, grand}\}$). Un *concept non-terminal* est un concept défini à partir d'un ensemble de concepts appelés *concepts composants*. Un concept composant peut être soit un concept terminal, soit un concept non-terminal. On appelle *composition par définition* le processus de construction d'un concept à partir d'autres concepts.

La Figure 2.1 présente un exemple d'une hiérarchie de concepts terminaux et non-terminaux. Le concept «Couleur», défini selon un modèle RVB, est un concept non-terminal défini à partir des concepts terminaux «Rouge», «Vert» et «Bleu» dont les domaines sont $[0,255]_1$. Le concept «Maison» est défini à partir des concepts «Toit» et «Murs». Le concept «Mur» est défini à l'aide des concepts «Hauteur», «Largeur», «Texture» (ayant pour domaine $\{\text{pierre, bois, brique}\}$) et «Couleur».

La version originelle de la bibliothèque de CordiFormes propose un certain nombre de concepts prédéfinis classiquement utilisés en synthèse d'images (les couleurs, les boîtes englobantes, les courbes, etc.). Cependant, le domaine d'application considéré peut nécessiter la création de concepts spécifiques non recensés dans la bibliothèque. Pour créer un concept terminal, le concepteur doit renseigner les attributs caractérisant le concept (c'est-à-dire principalement le domaine et les propriétés de base). Pour créer un concept non-terminal, le concepteur peut soit le définir de toute pièce en spécifiant ses concepts composants et leurs relations, soit le définir par composition de concepts déjà définis, soit le définir par *Spécialisation/Généralisation* de concepts déjà définis.

2.1.2 Affectation des méthodes de génération

Proposition d'une nouvelle classification

Les différents travaux réalisés en modélisation déclarative sont généralement classés suivant *les techniques de génération utilisées* :

- Arbres d'exploration explicites (arbres d'énumération et autres algorithmes de parcours d'arbres) [7, 9, 18];

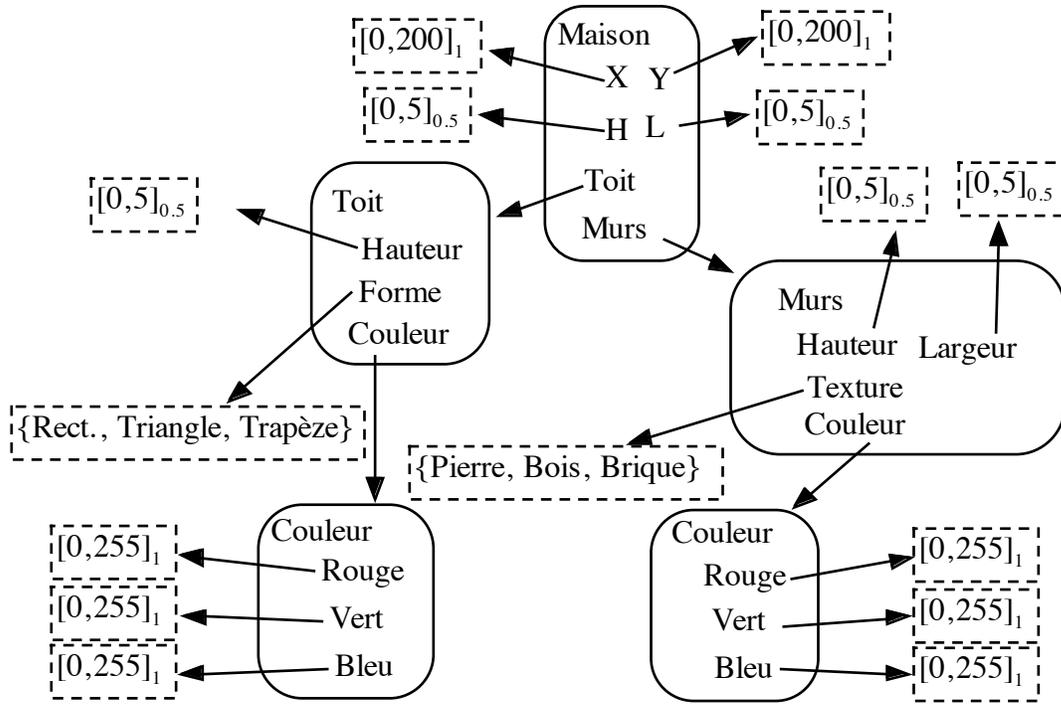


FIG. 2.1 – *Un exemple d'une hiérarchie de concepts*

- Arbres d'exploration implicites (moteurs d'inférences) [19, 20];
- Grammaires génératives [2, 3, 8];
- Systèmes à base de résolution de contraintes et d'arithmétique des intervalles [4, 5];
- Calcul de la fonction inverse [21], etc.

Dans le cadre du projet CordiFormes, la classification adoptée repose non pas sur les techniques de génération utilisées mais sur les *objectifs recherchés et les effets escomptés pour l'utilisateur*. Dans ce contexte, seules deux classes de génération sont retenues à savoir l'*énumération* et le *tirage aléatoire sous contraintes*.

Une génération est dite *par énumération* si, étant donné un objet k , il est possible de prédire avec exactitude ce que sera l'objet $k+1$. Le passage d'un objet (ou d'une valeur) à un autre est totalement contrôlé. Une génération est dite *aléatoire* si elle n'est pas par énumération, c'est-à-dire s'il n'est pas possible de prédire avec exactitude ce que sera l'objet $k+1$ connaissant l'objet k . Le passage d'un objet à un autre suit une certaine loi de probabilité.

Toutes les méthodes de génération développées en modélisation déclarative peuvent être réparties selon ces deux classes.

La génération récursive

La *génération récursive* repose sur la représentation des objets de la scène sous forme de hiérarchies de concepts (composition par définition). Cette technique de génération consiste à construire un concept C_i à partir de ces concepts composants générateurs³ : on génère dans un premier temps chaque concept composant générateur du concept C_i , puis on construit le concept C_i en fonction des générations des concepts composants.

Dans ce cadre, la classe de génération d'un concept non-terminal C_i dépend des classes de génération de ces concepts composants. Intuitivement, un concept C_i est généré de façon aléatoire si un au moins un de ses concepts composants est généré de façon aléatoire ; dans le cas contraire, la méthode de génération de C_i est de type énumération.

La génération d'un concept terminal (ou d'un concept non-terminal d'un point de vue *représentation* mais considéré comme terminal d'un point de vue de *génération*) peut être de type énumération ou de type aléatoire. Dans les deux cas, différentes méthodes de parcours sont possibles suivant l'ordre d'énumération utilisé pour une génération de classe énumération (croissant, décroissant, etc.) et les lois de probabilité adoptées pour une génération de classe aléatoire.

Problèmes sous-jacents à la génération récursive

Le choix d'une méthode de génération est généralement influencé par la classe de génération recherchée pour l'objet considéré. En effet, lorsque le concepteur affecte une méthode particulière pour un concept C_i , il se pose, de façon implicite, des questions du type "le modeleur doit-il est capable de générer toutes les solutions possibles du concept C_i ?" ou "le modeleur doit-il générer deux solutions consécutives relativement différentes?". Les réponses à ces questions orientent ainsi le concepteur vers le choix d'une méthode de classe aléatoire ou de classe énumération.

Cependant, lors du processus d'affectation des modes de génération des concepts, le concepteur peut introduire des incohérences dans la hiérarchie de concepts en ce qui concerne les classes de génération souhaitées et les méthodes de génération choisies. Par exemple, supposons que la classe de génération souhaitée pour le concept C_i , défini à partir des concepts C_m et C_n , soit l'énumération et que pour le concept C_m , la méthode de génération adoptée est de classe aléatoire. Dans ce cadre, une incohérence est détectée car la classe de génération escomptée pour C_i est incompatible avec la méthode de génération adoptée pour C_m (Cf. Section 2.1.2). Ces erreurs ont pour origine le fait que les concepts manipulés peuvent se révéler très complexes (hiérarchie profonde, nombreux concepts composants, etc.).

3. Tous les concepts composants d'un concept non-terminal C_i ne sont pas nécessairement indispensables à la génération de C_i . Un concept composant participant à la construction d'un concept est appelé *concept générateur*. Un concept non-terminal peut posséder plusieurs ensembles de concepts générateurs. Dans ce cadre, le concepteur doit choisir l'ensemble le plus efficace en termes de rapidité et contrôle des solutions. Dans ce rapport, nous considérons que tout concept C_i possède un seul groupe de concepts générateurs composé de tous les concepts composants de C_i .

Concrètement, dans la version actuelle de la plate-forme, l'affectation d'une méthode de génération à un concept C_i peut être réalisée de trois façons :

1. *Affectation implicite* suite à la sélection de C_i dans la bibliothèque ou à la création de C_i sans préférence a priori sur un mode de génération particulier. La méthode par défaut associée à C_i est alors utilisée⁴.
2. *Affectation explicite par modification* suite à la sélection de C_i dans la bibliothèque et modification de la méthode de génération par défaut. Cette modification consiste soit à affecter une nouvelle méthode pour C_i (autre que la méthode proposée par défaut), soit à modifier les méthodes de génération des concepts composants de C_i , soit à modifier l'ensemble des concepts composants générateurs de C_i .
3. *Affectation explicite par création* suite à la création ex-nihilo de C_i . Dans le cas où C_i est un concept non-terminal, le concepteur peut soit directement affecter une méthode de génération spécifique (sans se préoccuper des modes de génération des concepts composants), soit associer à chaque concept composant de C_i une méthode de génération (le mode de génération adopté pour C_i étant alors défini par application de la génération récursive).

Quelque soit l'approche d'affectation utilisée, le concepteur est toujours susceptible d'introduire des incohérences dans une hiérarchie de concepts. Par exemple, dans le cadre d'une affectation explicite par modification (cas 2), le concepteur peut modifier certaines des méthodes de génération des concepts composants générateurs de C_i sans pour autant modifier la classe de génération de C_i . Ceci peut entraîner une incohérence qui aura pour effet de rendre impossible la génération de la scène.

On peut noter que dans un contexte de méthode de conception plus évoluée (comme par exemple la conception par ébauches), certaines incohérences n'ont aucune influence sur la génération de la scène car les différents objets ne sont pas générés en même temps. Par exemple, dans le cadre de la construction d'un dolmen [7] par ébauches (Cf. Figure 2.2), on détermine d'abord (à l'aide d'une méthode de génération de type énumération) la forme du dolmen. Puis, dès qu'une solution satisfaisante est retenue, on génère de façon aléatoire (c'est-à-dire avec une classe de génération concurrente) les concepts composants sans pour autant engendrer une incohérence car l'objet général n'est pas généré en même temps que les objets spécifiques (du fait de la méthode de conception utilisée).

4. Chaque concept de la bibliothèque possède une méthode de génération par défaut. De plus, pour chaque concept nouvellement créé par le concepteur est affectée, de façon automatique, une méthode de génération par défaut.

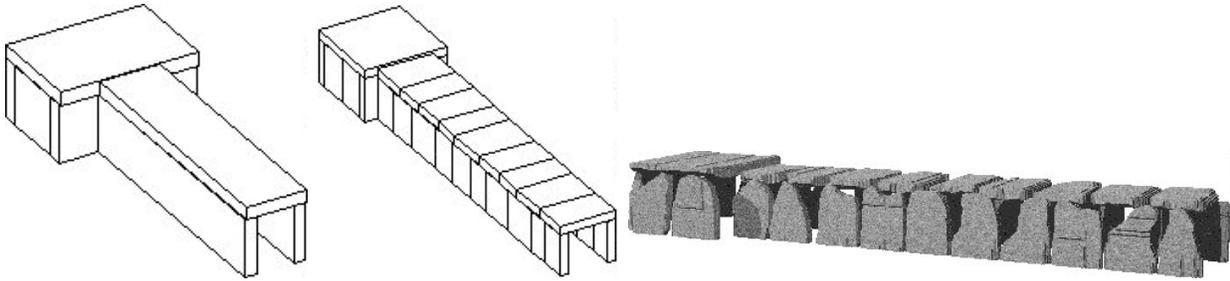


FIG. 2.2 – Les trois phases de conception d'un dolmen [7]

2.2 Nécessité de nouveaux outils à intégrer à la plate-forme

Dans sa version originelle, l'architecture de la plate-forme CordiFormes est une architecture 3-niveaux :

1. *Le niveau noyau.* Cette couche est composée d'algorithmes et de structures classiques que le concepteur peut éventuellement redéfinir et/ou paramétrer pour créer un modèleur déclaratif. Elle contient également la bibliothèque d'objets et de méthodes réutilisables que le concepteur peut enrichir. Cette couche a été développée à partir du langage *Java*.
2. *Le niveau interface.* Cette couche regroupe un ensemble de dialogues standard permettant au concepteur de définir la gestion de l'interaction avec l'utilisateur final du modèleur (i.e., la description, la génération et la prise de connaissance).
3. *Le niveau prototype.* Cette couche permet de produire un premier prototype du modèleur escompté. Elle utilise les outils des couches interface et noyau. Elle correspond à un modèleur déclaratif minimal permettant de faire une saisie de description, de générer les scènes correspondantes et d'en prendre connaissance.

Cependant, les différents outils proposés par ces trois niveaux ne sont pas suffisants pour assister la conception d'un modèleur déclaratif. En particulier, la plate-forme n'offre aucune aide lors de la *sélection* et la *configuration* des objets et des méthodes de la bibliothèque. De plus, elle ne propose pas de processus de *vérification de la cohérence* des modes de génération associés aux objets. Aussi, la plate-forme doit être enrichie d'outils d'aide à la conception. L'objectif est de proposer des outils de haut niveau permettant d'aider le concepteur à comprendre et à préciser son problème, à concevoir son application et à valider ses choix. La finalité recherchée est de guider le concepteur dans les choix des techniques à utiliser et des éléments essentiels à programmer.

La solution que nous proposons consiste d'une part à construire une Base de Connaissances permettant d'*indexer* la bibliothèque actuelle et d'autre part à définir des mécanismes de raisonnement dédiés à la *sélection* et la *vérification* de la cohérence des modes

de génération adoptés pour les objets manipulés par le futur modelleur. Cette proposition conduit à une extension de l'architecture actuelle de la plate-forme vers une architecture 4-niveaux (Cf. Figure 2.3).

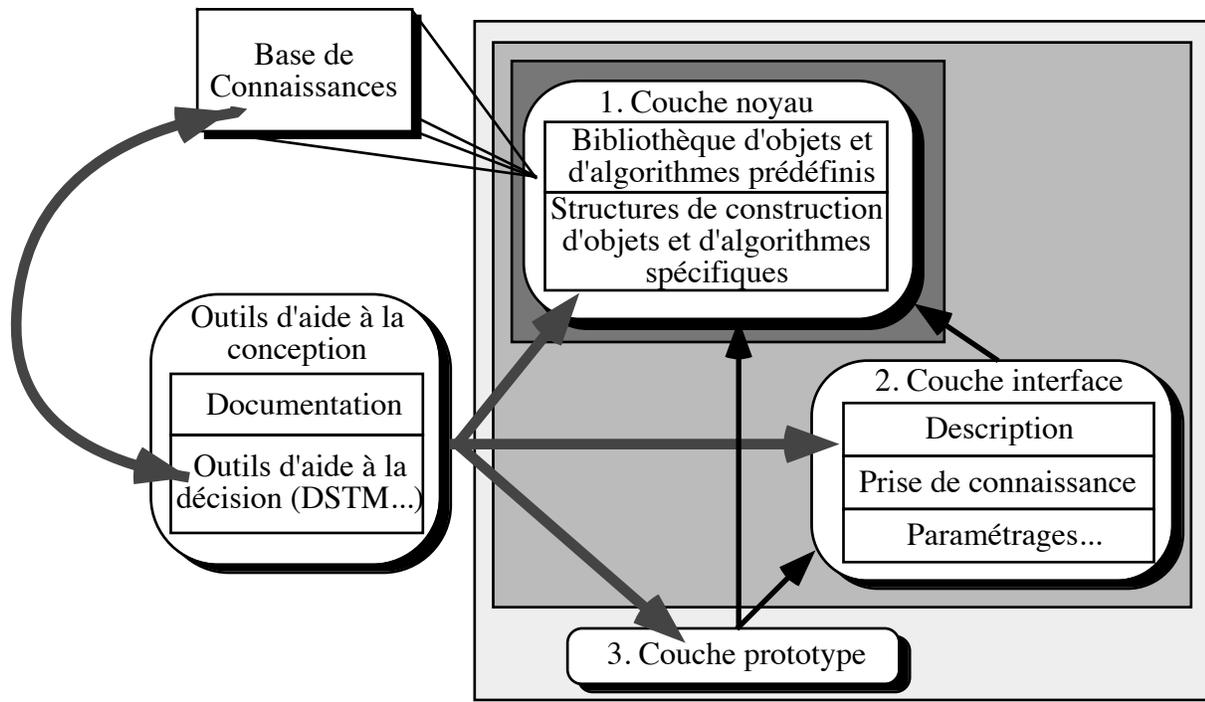


FIG. 2.3 – D'une architecture 3-niveaux à une architecture 4-niveaux

Chapitre 3

Construction de la Base de Connaissances à l'aide de DSTM

3.1 Justification de l'utilisation de DSTM

DSTM (Dynamic Selection of Tasks and Methods) est un environnement de construction de Systèmes à Base de Connaissances (SBC) fondés sur des architectures de type Tâche/Méthode [12] [13]. Un SBC construit à partir de DSTM est composé (1) d'un ensemble de tâches et de méthodes (identifiant respectivement les différents problèmes à résoudre et les différents moyens possibles de résoudre ces derniers) et (2) d'un ensemble de mécanismes d'inférence permettant de mettre en œuvre des raisonnements opportunistes (sélection dynamique de la tâche la plus pertinente à réaliser en fonction du contexte de résolution, puis de la méthode la plus favorable à sa réalisation).

L'originalité de DSTM par rapport aux travaux similaires réalisés en Intelligence Artificielle (e.g., Lisa [22] ou MML [23]) est qu'il ne force pas une représentation particulière de ce que doit être une tâche et/ou une méthode. Il offre la possibilité de définir des structures de représentation (sous-jacentes aux primitives de modélisation *tâche* et *méthode*) ainsi que des mécanismes de sélection (c'est-à-dire comment sont sélectionnées les tâches et les méthodes, comment est réalisé le choix entre plusieurs méthodes candidates à la réalisation d'une tâche, etc.) permettant de représenter au mieux l'expertise étudiée.

Dans le contexte du projet CordiFormes, l'expertise que l'on cherche à représenter est relative au *savoir-faire* des spécialistes du domaine de l'image quant au modes de génération habituellement utilisés pour chaque objet recensé dans l'ontologie. Dans ce cadre, nous avons utilisé DSTM pour définir une première ébauche de la Base de Connaissances afin de valider, sur la base d'un premier prototype, notre approche d'aide à la recherche du mode de génération des objets manipulés par le modèleur déclaratif en cours de construction.

Le choix de DSTM pour implanter cette Base de Connaissances se justifie de part la flexibilité offerte par un tel outil. DSTM permet de respecter un principe de *correspondance structurelle* entre les connaissances modélisées (au niveau conceptuel) et leur représentation informatique (au niveau symbolique). Une telle représentation explicite (des

tâches, des méthodes et des mécanismes de sélection associés) facilite les modifications et permet de tester (à moindre frais) différentes versions de la Base de Connaissances.

3.2 Les principes de modélisation de la Base de Connaissances

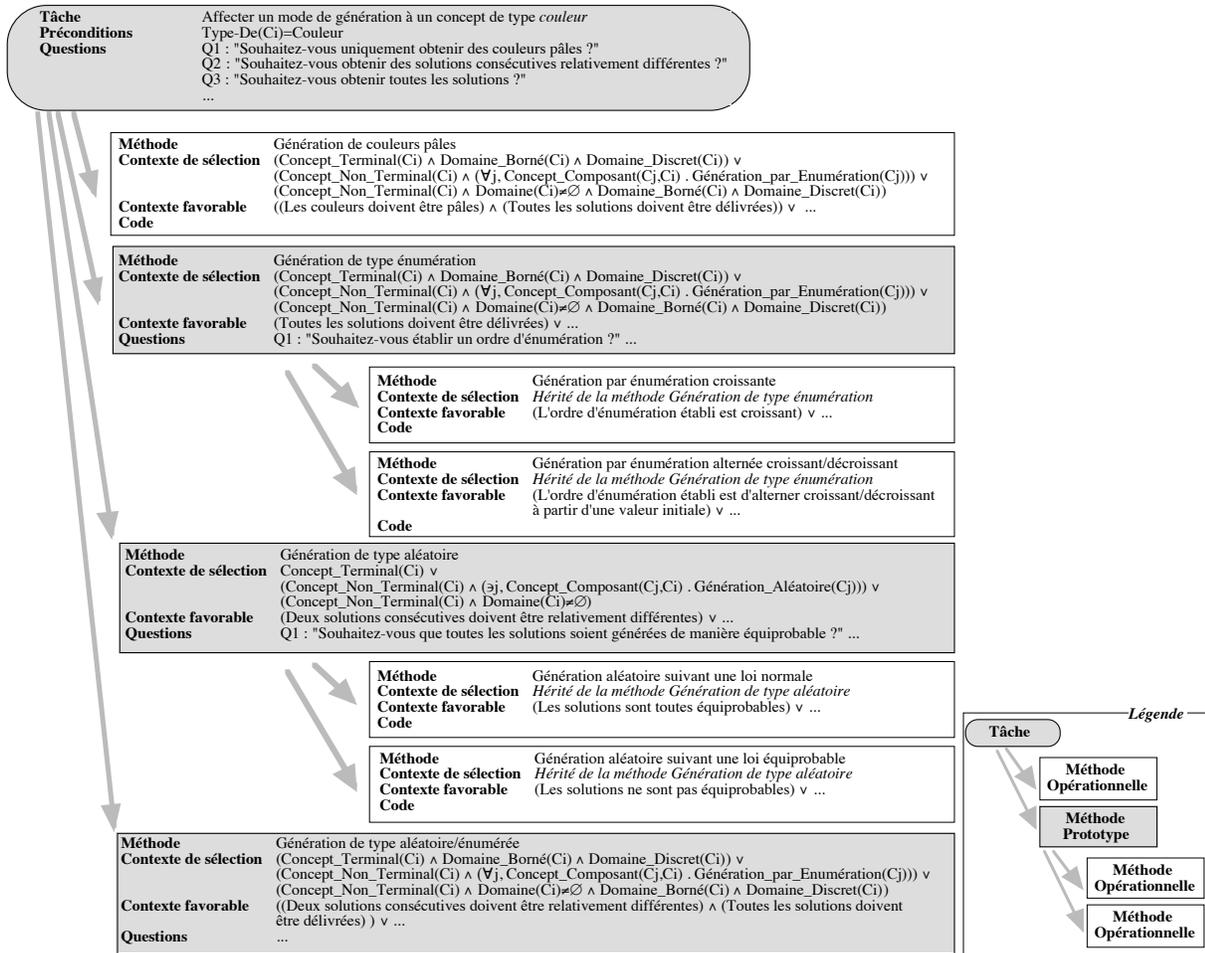
3.2.1 Les primitives de modélisation tâche et méthode adoptées

Les choix adoptés pour les primitives de modélisation tâche et méthode ont été conditionnés (1) par les contraintes techniques intrinsèques aux méthodes de génération actuelles (par exemple “*une méthode de type énumération ne peut être utilisée que pour des concepts dont le domaine est borné et discret*”) et (2) par notre volonté de prendre en compte les choix du concepteur quant aux objectifs attribués au modèleur déclaratif en cours de construction (par exemple “*le modèleur doit être capable de générer toutes les solutions possibles d’un concept*” ou “*deux solutions consécutives demandées par l’utilisateur doivent être relativement différentes*”).

La notion de tâche est utilisée pour caractériser un objectif d’affectation d’un mode de génération à un concept C_i . Une tâche est définie par un ensemble de pré-conditions caractérisant le type de concept à générer (par exemple une tâche de génération d’un concept de type *couleur* ou *texture*) et par un ensemble de méthodes de génération (connues a priori) du concept C_i . A chaque concept recensé dans l’ontologie correspond une tâche.

La notion de méthode est utilisée pour caractériser une technique de génération spécifique à un concept particulier (par exemple une méthode de génération de couleurs pâles) ou générale et réutilisable pour tout concept (par exemple une méthode de génération aléatoire suivant une loi de distribution normale). Une méthode est définie par un contexte de sélection et un contexte favorable caractérisant respectivement quand il est *possible* et *préférable* de l’utiliser. Les connaissances associées au contexte de sélection traduisent des contraintes techniques du type “*une génération par énumération n’est possible que si le domaine de valeurs possibles du concept C_i considéré est borné et discret*” ou “*un concept non terminal est nécessairement généré de façon aléatoire lorsque un de ses concepts composants est généré de façon aléatoire*”. Les connaissances associées au contexte favorable permettent de prendre en compte les choix du concepteur. Par exemple, si le concepteur désire que le futur modèleur délivre toutes les solutions possibles pour le concept C_i , il est préférable d’utiliser une méthode de type énumération plutôt qu’une méthode de type aléatoire.

La Figure 3.1 présente la tâche permettant d’affecter un mode de génération à un concept de type *couleur* ainsi que les différentes méthodes actuellement recensées pour un tel objectif. La méthode *Génération de couleurs pâles* est spécifique à la tâche *Affecter un mode de génération à un concept de type couleur*. Les méthodes *Génération de type énumération*, *Génération de type aléatoire* et *Génération de type aléatoire/énumérée* sont des techniques utilisables quelque soit le concept considéré (ces méthodes sont donc associées à toutes les tâches de la Base de Connaissances).



Concept_Terminal, *Domaine_Borné*, *Concept_Composant*, etc. sont des primitives qui s'appuient sur la description des concepts obtenue au terme de la phase 1 (construction par définition des concepts) du processus de construction d'un modèleur déclaratif présenté en Section 2.

FIG. 3.1 – La tâche d'affectation d'un mode de génération à un concept de type couleur

Un ensemble de questions est associé à chaque tâche. Ces questions sont posées (dynamiquement) au concepteur dès que la tâche est sélectionnée (c'est-à-dire lorsque le concepteur est en cours de recherche d'un mode de génération pour le concept C_i sous-jacent à la tâche). Ces questions permettent de prendre en compte les préférences du concepteur et enrichissent le contexte conditionnant le choix entre les différentes méthodes de génération. Ce contexte est initialement composé de connaissances techniques relatives aux caractéristiques du concept C_i étudié (par exemple " C_i est un concept terminal dont le domaine est borné et discret" ou " C_i est un concept non-terminal défini à partir des concepts C_j et C_k ", etc.). Les réponses fournies à ces questions vont permettre d'enrichir ce contexte de connaissances plus générales relatives aux choix du concepteur (par exemple "*Je souhaite offrir la possibilité au futur utilisateur de prendre connaissance de toutes les solutions possibles*"). Certaines questions sont communes à toutes les tâches car non liées à la sémantique du concept considéré mais à la façon de générer ce

dernier (par exemple “*Souhaitez-vous obtenir toutes les solutions?*”). D’autres sont plus spécifiques et permettent de particulariser les propriétés d’un concept. Par exemple, la question “*Souhaitez-vous uniquement obtenir des couleurs pâles?*” est spécifique à la tâche d’affectation dédiée au concept de *couleur* et permet d’orienter le conseil vers une méthode particularisant la propriété *saturation* (en l’occurrence la méthode *Génération de couleurs pâles*).

On distingue deux types de méthodes : les méthodes *prototypes* qui caractérisent des classes de génération et les méthodes *opérationnelles* qui caractérisent des algorithmes de génération effectifs et opérationnels. Une méthode prototype est la généralisation d’un ensemble de méthodes opérationnelles (mécanisme de *Spécialisation/Généralisation*). Par exemple, les méthodes *Génération aléatoire suivant une loi normale* et *Génération aléatoire suivant une loi équiprobable* sont toutes deux des spécialisations de la méthode prototype *Génération de type aléatoire*. Elles ont pour point commun des algorithmes de génération reposant sur un tirage aléatoire des solutions potentielles parmi l’ensemble des solutions possibles.

A chaque méthode prototype est associé un ensemble de questions. Ces questions sont soumises au concepteur dès que la méthode (prototype) est *candidate*¹ (ou *favorable*) à la génération du concept C_i considéré. Celles-ci permettent de préciser le contexte de sélection conditionnant le choix entre différentes méthodes appartenant à la même classe de génération. Par exemple, la question “*Souhaitez-vous établir un ordre d’énumération?*”, associée à la méthode *Génération de type énumération*, offre la possibilité au concepteur de préciser un ordre de génération des solutions potentielles.

Les questions associées aux tâches et aux méthodes prototypes permettent (1) d’acquérir dynamiquement les connaissances nécessaires à la sélection de la méthode de génération la plus favorable pour le concept considéré et (2) de construire dynamiquement l’interaction avec le concepteur. L’objectif recherché est de mobiliser les ressources uniquement lorsque celles-ci sont nécessaires. Ces deux aspects sont mis en œuvre par le mécanisme de sélection présenté en Section 3.2.2.

3.2.2 Le mécanisme de sélection adopté

Le mécanisme de sélection des tâches et des méthodes repose sur l’enchaînement des primitives suivantes :

- (1) sélection d’une tâche d’affectation de mode de génération pour le concept C_i (utilisation des *Préconditions* des tâches comme critère de sélection) ;
- (2) raffinement du contexte de sélection par interaction avec le concepteur (mise en œuvre des *Questions* associées à la tâche sélectionnée) ;

1. Une méthode M_i est *candidate* à la génération d’un concept C_i si et seulement si les connaissances associées à son *Contexte de sélection* sont vérifiées. Une méthode M_i est *favorable* à la génération d’un concept C_i si et seulement si elle est candidate et les connaissances associées à son *Contexte favorable* sont vérifiées.

- (3) identification d'un ensemble de méthodes candidates (utilisation du *Contexte de sélection* des méthodes comme critère de sélection) ;
- (4) identification d'un ensemble de méthodes favorables (utilisation du *Contexte favorable* des méthodes comme critère de sélection) ;
- (5) raffinement du contexte de sélection par interaction avec le concepteur (mise en œuvre des *Questions* associées aux méthodes prototypes sélectionnées) ;
- (6) sélection d'une méthode de génération (utilisation du *Contexte favorable* des méthodes opérationnelles comme critère de sélection) ;
- (7) justification de la méthode de génération choisie pour C_i (présentation des différentes méthodes candidates et explicitation du choix en fonction du *Contexte favorable* et/ou *Contexte de sélection*).

Suite à l'étape (4), plusieurs cas de figures sont possibles :

- Cas 1. Une seule méthode est favorable et cette méthode est opérationnelle *ou* une seule méthode est candidate (aucune n'est favorable) et cette méthode est opérationnelle. Les étapes (5) et (6) sont omises et le système justifie la méthode sélectionnée (étape 7).
- Cas 2. Une seule méthode est favorable et cette méthode est de type prototype *ou* une seule méthode est candidate (aucune n'est favorable) et cette méthode est de type prototype. Les étapes (5), (6) et (7) sont réalisées séquentiellement.
- Cas 3. Plusieurs méthodes sont favorables et toutes sont opérationnelles *ou* plusieurs méthodes sont candidates (aucune n'est favorable) et toutes sont opérationnelles. Les étapes (5) et (6) sont omises. Le système présente (et justifie) les différentes méthodes candidates et/ou favorables et demande au concepteur de choisir. Dans le cas où il existe des méthodes favorables et des méthodes candidates, le système propose en priorité les méthodes favorables mais offre également la possibilité au concepteur de choisir parmi les méthodes uniquement candidates.
- Cas 4. Plusieurs méthodes sont favorables et certaines sont de type prototype et d'autres opérationnelles *ou* plusieurs méthodes sont candidates (aucune n'est favorable) et certaines sont de type prototype et d'autres opérationnelles. Le système poursuit le mécanisme de sélection uniquement à partir des méthodes opérationnelles (Cf. Cas 1 et Cas 3). Cette heuristique a pour origine notre volonté de privilégier les choix du concepteur particularisant les propriétés du concept étudié. Par exemple, dans un contexte où le concepteur souhaite obtenir toutes les solutions de couleur pâles, les méthodes *Génération de couleurs pâles* et *Génération de type énumération* sont toutes les deux favorables (les connaissances associées à

leur *Contexte favorable* respectif sont vérifiées). Cependant, la priorité doit être accordée à la méthode *Génération de couleurs pâles* qui satisfait totalement (et pas seulement partiellement comme c'est le cas pour la méthode *Génération de type énumération*) les volontés du concepteur (Cf. Figure 3.1).

Cas 5. Plusieurs méthodes sont favorables et toutes sont de type prototype ou plusieurs méthodes sont candidates (aucune n'est favorable) et toutes sont de type prototype. Le système sélectionne la méthode prototype satisfaisant "*au mieux*" les choix du concepteur. Les étapes (5), (6) et (7) sont ensuite réalisées séquentiellement. Par exemple, dans un contexte où le concepteur souhaite obtenir, pour un concept de type *couleur* dont le domaine est borné et discret, toutes les solutions avec pour contrainte que "*deux solutions consécutives doivent être relativement différentes*", les méthodes *Génération de type aléatoire* et *Génération de type aléatoire/énumérée* sont toutes deux favorables (Cf. Figure 3.1). Cependant, la priorité doit être accordée à la méthode *Génération de type aléatoire/énumérée* qui satisfait totalement les souhaits du concepteur (la méthode *Génération de type aléatoire* n'assurant pas l'obtention de toutes les solutions).

Il est intéressant de noter que la prise en compte des désirs du concepteur peut entraîner des incompatibilités. Par exemple, supposons un concept terminal C_i pour lequel le domaine de valeurs possibles est non-borné et qu'un des choix du concepteur est d'offrir la possibilité à l'utilisateur de visualiser toutes les solutions possibles de C_i . Dans ce cas de figure, les caractéristiques intrinsèques de C_i imposent une méthode de type génération aléatoire alors que la volonté du concepteur oriente le choix vers une méthode de type énumération. Lors de tels cas de figures, le parti pris retenu est (1) d'informer le concepteur de l'incompatibilité soulevée par sa demande et (2) de privilégier le respect des caractéristiques techniques (le respect du *Contexte de sélection* est prioritaire au respect du *Contexte favorable*).

Étant donné un concept C_i , le concepteur peut souhaiter utiliser une méthode de génération personnelle. Pour détecter cette situation, la question "*Souhaitez-vous utiliser une méthode de génération personnelle?*" est posée en priorité lors de la réalisation de toute tâche (étape 2) et le mécanisme de sélection est modifié à partir de l'étape (3) de la façon suivante :

- (3 bis) recueil d'informations sur la nouvelle méthode de génération proposée pour C_i ;
- (4 bis) vérification de la cohérence de la méthode de génération proposée pour C_i .

L'étape (3 bis) a pour objectif de collecter les informations nécessaires à l'insertion de la méthode dans la Base de Connaissances. Ces informations portent sur les prérequis nécessaires au bon fonctionnement de la méthode ("*ensemble borné ou non-borné?*", "*ensemble continu ou discret?*", etc.) et sur l'ensemble des solutions possibles générés par la méthode ("*votre méthode permet-elle d'engendrer toutes les solutions?*", "*deux solutions consécutives sont-elles proches?*", etc.). L'objectif de cette session *questions/réponses* est

de renseigner le *contexte de sélection* et le *contexte favorable* de la méthode. Parallèlement à ce recueil d'informations nécessaires, il est également possible de formuler des questions spécifiques traduisant les (éventuelles) propriétés particularisant la méthode proposée (par exemple, “*Souhaitez-vous uniquement obtenir des couleurs pâles ?*” qui traduit la spécificité de la méthode *Génération de couleurs pâles*). Ces questions sont associées à la tâche sélectionnée.

L'étape (4 bis) a pour objectif de vérifier si la méthode proposée n'introduit pas d'incohérences avec des méthodes de génération précédemment affectées à des concepts en relation avec C_i .

Pour résumer, au terme d'un cycle de sélection pour un concept C_i , différents cas de figures sont possibles:

1. la méthode proposée par DSTM satisfait les souhaits du concepteur qui confirme simplement le choix;
2. une incohérence a été détectée (par exemple les choix du concepteur ne peuvent être satisfaits et/ou la méthode personnelle du concepteur ne peut pas être utilisée au vu des contraintes techniques imposées par les caractéristiques du concept C_i) et explicitée au concepteur qui suit les conseils de DSTM en confirmant la méthode proposée;
3. une incohérence a été détectée et explicitée au concepteur qui ne souhaite pas relâcher ces contraintes et/ou opter pour une méthode autre que la sienne. Une phase de négociation est alors proposée durant laquelle le concepteur est autorisé à remettre en cause ses décisions antérieures concernant les méthodes de génération choisies et/ou remettre en cause les modes de représentation des concepts.

3.3 Intégration au projet CordiFormes

Une première version de la Base de Connaissances est actuellement en cours de développement à l'aide de DSTM. La flexibilité offerte par cet outil ainsi que le principe de représentation explicite de toutes les connaissances (c'est-à-dire des tâches, des méthodes et des mécanismes de sélection associés) nous permet de tester et raffiner progressivement la Base de Connaissances sans problèmes d'implantation de bas niveau. En d'autres termes, DSTM nous permet de *modéliser* et d'*opérationnaliser* la base de façon simultanée.

DSTM a été développé à partir de la couche orientée objet *Ceyx* du langage *LeLisp*. Aussi, la version actuelle de la Base de Connaissances est uniquement utilisable à partir d'un environnement *LeLisp*. Cependant, un projet en cours consiste à re-développer DSTM dans le langage *Java*. Ceci offre une perspective intéressante d'uniformisation de nos travaux et nous permet d'envisager une intégration aisée de la Base de Connaissances dans la plate-forme CordiFormes (également développée en *Java*). L'architecture que nous proposons est présentée Figure 3.2.

Suite à la définition des hiérarchies de concepts définies sous CordiFormes (et représentées à l'aide d'objets *Java*), une phase de recherche du mode de génération pour chacun

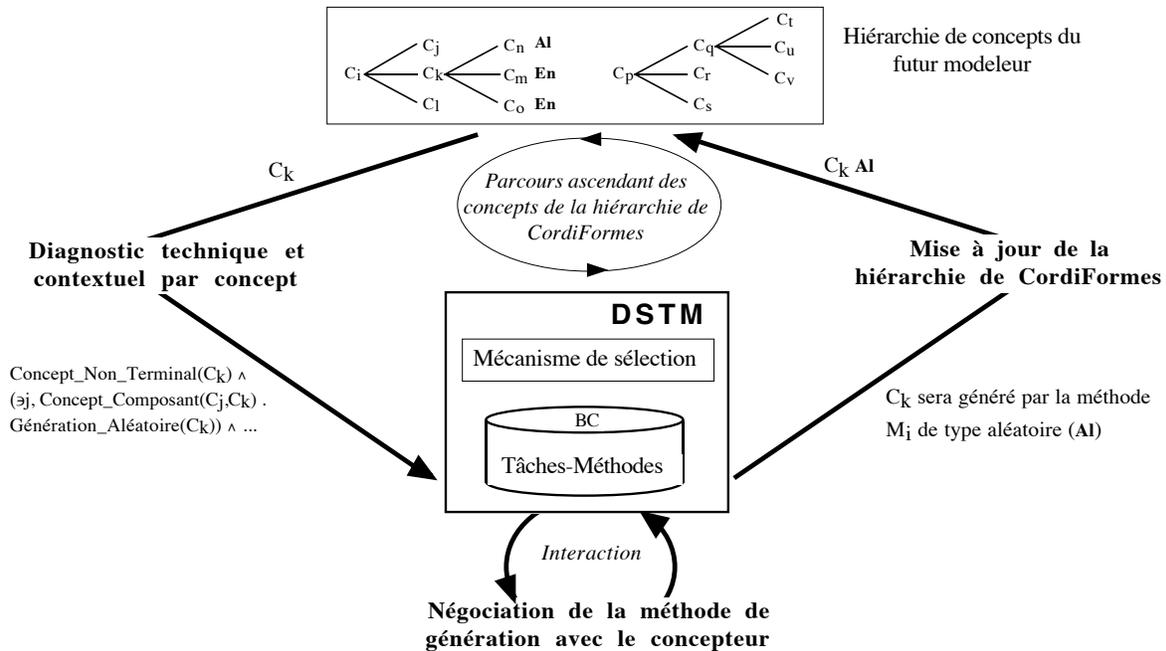


FIG. 3.2 – Introduction de l’architecture DSTM dans la plate-forme CordiFormes

des concepts est effectuée. Chaque hiérarchie est parcourue de façon ascendante (on étudie en premier lieu les concepts terminaux puis les concepts non-terminaux). Étant donné un concept C_i , un premier diagnostic est réalisé afin de déterminer le contexte technique de sélection. Ce diagnostic est qualifié de contextuel car il prend en compte les éventuelles affectations déjà réalisées. Par exemple, pour le concept C_k (Cf. Figure 3.2), ce diagnostic technique peut être interprété comme suit : “ C_k est un concept non-terminal et un de ses concepts composants est généré de façon aléatoire”. Ensuite, le mécanisme de sélection dynamique de tâches et de méthodes défini à l’aide de DSTM (et implanté en *Java*) est mis en œuvre avec pour point d’entrée le concept C_i . Suivant la situation engendrée (c’est-à-dire consensus ou conflit), DSTM affecte directement une méthode de génération au concept C_k ou propose une phase de négociation. Cette négociation peut remettre en cause les modes de génération précédemment affectés aux concepts composant C_k et/ou les modes de représentation de ces concepts (c’est-à-dire la hiérarchie d’objets *Java* définie sous CordiFormes).

Concrètement, la Base de Connaissances actuelle (définie à partir de la version originale de DSTM) ne permet pas de mettre en œuvre ce processus de négociation².

2. On peut également noter que cette Base de Connaissances ne permet pas de gérer la *configuration* des différents éléments réutilisables de la bibliothèque.

Chapitre 4

Discussion

4.1 Bases de Connaissances de type tâche/méthode Versus Base de règles de production

Le choix d'un formalisme de représentation de type tâche/méthode se justifie de part les deux objectifs recherchés au travers la Base de Connaissances qui sont (1) d'offrir la possibilité de justifier le choix de la méthode proposée (ce qui nécessite l'explicitation de toutes les connaissances conditionnant l'utilisation des méthodes de génération) et (2) de faciliter l'extension de la base.

L'utilisation d'un formalisme de type "règles de production" ne permet pas de répondre à ces deux objectifs. En effet, une règle de production comporte un ensemble de prémisses P_i où chaque prémisses identifie une connaissance conditionnant la sélection de la méthode de génération sous-jacente à la règle. Cependant, ces prémisses ne sont pas structurées suivant la nature des connaissances qu'elles représentent. Par exemple, la règle suivante, utilisée pour représenter les conditions d'utilisation de la méthode «Génération de couleurs pâles», ne permet pas de différencier, de façon explicite, que les 4 premières prémisses (P1, P2, P3 et P4) sont liées à des contraintes techniques alors que les deux dernières (P5 et P6) sont relatives aux choix du concepteur.

Si	Type-De(Ci)=Couleur	P1
	Concept_Terminal(Ci)	P2
	Domaine_Borné(Ci)	P3
	Domaine_Discret(Ci)	P4
	Les couleurs doivent être pâles	P5
	Toutes les solutions doivent être délivrées	P6
Alors	La méthode "Génération de couleurs pâles" est une méthode candidate et favorable	

Le système n'ayant pas les moyens de différencier des prémisses de nature différentes, il est impossible de mettre en œuvre un mécanisme de sélection spécifique comme celui présenté en Section 3.2.2 (en particulier, de différencier les méthodes candidates des méthodes favorables) ni même de justifier, de façon détaillée, la méthode sélectionnée.

De plus, il est reconnu que la maintenance et l'extension d'une base de règles pose de nombreux problèmes [14] comme par exemple la difficulté de conserver une compréhension

synthétique du comportement général du système à partir de règles isolées.

L'utilisation du formalisme de représentation tâche/méthode permet de résoudre ces problèmes et en particulier permet (1) de représenter explicitement toutes les connaissances conditionnant l'utilisation d'une méthode de génération et (2) de faciliter la modularité et la flexibilité de la base. De plus, DSTM propose des outils de *vérification* et de *validation* qui facilitent la mise au point de la Base de Connaissances [12].

4.2 Vers la prise en compte de nouvelles connaissances de sélection

Comme nous l'avons introduit en Section 2.1.2, un concept C_i peut posséder plusieurs ensembles de concepts générateurs caractérisant différents modèles de représentation de C_i . Or, chacun de ces modèles de représentation peut posséder des caractéristiques spécifiques influençant les solutions potentielles de C_i . Par exemple, le concept «couleur» peut être représenté à l'aide d'un grand nombre de modèles [24] dont, en particulier, les modèles RVB, CMJN et TLS. Cependant, ces modèles ne sont pas totalement équivalents du point de vue des couleurs produites [24]. Ces différents points de vues sur un même concept C_i doivent donc être pris en compte lors du processus d'affectation des méthodes de génération. Pour se faire, nous envisageons (1) d'associer à chaque méthode des connaissances caractérisant le mode de représentation adopté pour C_i et (2) d'associer à chaque tâche des questions spécifiques permettant au concepteur de préciser le mode de représentation qu'il souhaite adopté pour C_i .

Nous envisageons également d'intégrer des connaissances sur la complexité des algorithmes sous-jacents aux méthodes de génération (temps de calcul, mobilisation de ressources, etc.) et sur la fiabilité, l'exhaustivité sémantique, la validité, l'unicité et l'abstraction [3, 18] de ces dernières.

Il est important de souligner que grâce à la flexibilité de l'outil DSTM, ces nouvelles connaissances seront facilement et rapidement prise en compte dans la Base de Connaissances.

4.3 Utilisation de la Base de Connaissances lors de la phase de *description*

Suite à la conception d'un modeleur déclaratif à l'aide de la plate-forme CordiFormes, un concept est associé à une et une seule méthode de génération et ce choix est définitif pour toute utilisation du modeleur. Autrement dit, le concepteur impose ses choix à l'utilisateur. Une telle rigidité peut être levée de deux façons.

Une première solution consiste à offrir la possibilité au concepteur d'associer, pour chaque concept, plusieurs méthodes potentielles de génération tout en restant dans un domaine borné par les prérequis techniques. Le choix entre les méthodes est alors réalisé

lors de la description en fonction de critères techniques définis par l'utilisateur. Cependant, cette approche n'est pas vraiment satisfaisante du point de vue des préceptes de la modélisation déclarative où l'objectif est justement de dégager l'utilisateur de toutes contingences techniques liées au domaine d'application.

Une deuxième approche, plus intéressante, consiste à offrir la possibilité au concepteur d'associer des méthodes se différenciant d'une part selon des critères techniques et d'autre part selon les souhaits et les exigences du futur utilisateur. Par exemple, pour le concept «maison», il peut être intéressant de définir une première méthode qui génère de façon détaillée une solution (génération précise mais coûteuse en temps de calcul et en ressources matérielles) et une seconde qui génère uniquement une solution grossière (peu précise mais rapide et économique). Ces deux méthodes, identiques en terme d'objectif, permettent au modelleur de s'adapter aux souhaits éventuels de l'utilisateur exprimés soit de façon explicite dans la description (par exemple, «*Je veux une ébauche de maison rapidement*» qui oriente le choix vers la méthode de génération peu précise mais rapide), soit de façon implicite et, par conséquent, interprétés par le modelleur (par exemple, «*Au premier plan, je veux la maison que je viens de détailler précédemment et, au second plan, je veux une maison blanche*»). Dans ce dernier cas, le modelleur interprète le fait que l'utilisateur attache peu d'importance à la maison du second plan et, par conséquent, décide d'utiliser la méthode de génération peu précise mais rapide alors que pour la maison du premier plan (détaillée lors d'une description antérieure), le modelleur choisit la méthode de génération précise mais coûteuse en temps de calcul et en ressources matérielles. On peut noter que ces choix peuvent être remis en cause suivant le point de vue adopté pour la visualisation de la scène. En effet, dans notre exemple, si l'utilisateur utilise des outils de réalité virtuelle (comme par exemple VRML) à partir desquels il est possible de se déplacer dans la scène générée (c'est-à-dire d'adopter différents points de vue de visualisation de la même scène), la solution retenue par le modelleur n'est pas satisfaisante car l'utilisateur peut focaliser la visualisation sur tous les objets de la scène. Aussi, la maison située au second plan doit être générée de façon détaillée au même titre que celle du premier plan.

Cet exemple met en évidence l'intérêt potentiel d'utiliser la Base de Connaissances lors de la phase d'*utilisation* d'un modelleur déclaratif. En effet, le fait de mettre à la disposition du modelleur plusieurs méthodes de génération potentielles pour un même objet offre la possibilité de sélectionner dynamiquement le mode de génération permettant d'obtenir des solutions satisfaisant "*au mieux*" les souhaits de l'utilisateur final.

Bibliographie

- [1] M. Lucas, P. Martin, D. Martin, and D. Plemenos. Le projet ExploFormes : quelques pas vers la modélisation déclarative de formes. In BIGRE, editor, *Acte des journées AFCET-GROPLAN*, 67, pages 35–49, France, 1989.
- [2] S. Kochhar. CCAD: A Paradigm for Human-Computer Cooperation in Design. *IEEE Comp. Graph. and Appl.*, 14(3):54–65, 1994.
- [3] R. F. Woodbury. Searching for Designs: Paradigm and Practice. *Building and Environment*, 26(1):61–73, 1991.
- [4] D. Chauvat. *Le projet VoluFormes: un exemple de modélisation déclarative avec contrôle spatial*. PhD thesis, Univ. de Nantes, France, 1994.
- [5] E. Giunchiglia, A. Armando, P. Traverso, and A. Cimatti. Visual Representation of Natural Language Scene Description. *IEEE Trans. on Syst., Man and Cyber.*, 26(4):575–589, 1996.
- [6] M. Daniel and M. Lucas. Towards Declarative Geometric Modeling in Mechanics. In *IDMME'96*, pages 455–464, Nantes, France, 15-17 avril 1996.
- [7] F. Poulet and M. Lucas. Modelling Megalithic Sites. In *Eurographics'96*, pages 279–288, France, 1996.
- [8] A. Rau-Chaplin, B. MacKay-Lyons, and P. F. Spierenburg. The LaHave House Project: Towards an Automated Architectural Design Service. In *Cadex'96*, pages 24–31, 1996.
- [9] L. Khemlani. GENWIN: A Generative Computer Tool For Window Design in Energy-Conscious Architecture. *Building and Environment*, 30(1):73–81, 1995.
- [10] E. Desmontils. *Le projet CordiFormes: une plateforme pour la construction de modèles déclaratifs*. PhD thesis, Univ. de Nantes, France, 1998.
- [11] J. Charlet, B. Bachimont, J. Bouaud, and P. Zweigenbaum. Ontologie et réutilisabilité : expérience et discussion. In Cépadues-Editions, editor, *Acquisition et ingénierie des connaissances*, pages 69–87, Toulouse, 1996.

- [12] F. Trichet and P. Tchounikine. Reusing a Flexible Task-Method Framework to Prototype a Knowledge Based System. In *the 9th International Conference on Software Engineering and Knowledge Engineering (SEKE'97)*, pages 192–199, Madrid, Espagne, 1997.
- [13] F. Trichet and P. Tchounikine. DSTM: une approche de l'opérationnalisation fondée sur la réutilisation d'un noyau opérationnel. In *Actes des Journées Ingénierie des Connaissances (IC'97)*, pages 317–330, Roscoff, France, 1997.
- [14] J.M. David, J.P. Krivine, and R. Simmons. *Second Generation Expert Systems*. Springer-Verlag, 1993.
- [15] M. Lucas. Equivalence Classes in Object Shape Modeling. In *IFIP TC5/WG 5.10 Working Conference on Modeling in Computer Graphics*, pages 35–49, Japan, 1991.
- [16] M. Lucas and E. Desmontils. Les modeleurs déclaratifs. *Revue Int. de CFAO et d'infographie*, 10(6):559–585, 1995.
- [17] C. Colin, E. Desmontils, J.-Y. Martin, and J.-P. Mounier. Working with Declarative Modeler. In *Compugraphics'97*, pages 117–126, Portugal, 1997.
- [18] C. Colin. *Modélisation déclarative de scènes à base de polyèdres élémentaires*. PhD thesis, Univ. de Rennes, France, 1990.
- [19] D. Martin and P. Martin. Declarative Generation of a Family of Polyhedra. In *GraphiCon'93*, Russia, 1993.
- [20] D. Plemenos. *Contribution à l'étude et au développement des techniques de modélisation, génération et visualisation de scènes: le projet MultiFormes*. Prof. thesis, Univ. de Nantes, France, 1991.
- [21] C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, and D. Greenberg. Painting with Light. In ACM SIGGRAPH, editor, *Siggraph'93, Computer Graphics Proceedings, Annual Conference Series 1993*, pages 143–146, Anaheim, California, USA, 1–6 août 1993.
- [22] I. Jacob-Delouis and J.P. Krivine. LISA: un langage réflexif pour opérationnaliser les modules d'expertise. *Revue d'Intelligence Artificielle*, 9(1):53–88, 1995.
- [23] V. Guerrero-Rojo. MML, a modelling language with dynamic selection of methods. In *The Knowledge Acquisition for Knowledge-Based Systems Workshop (Banff'95)*, Banff, Canada, 1995.
- [24] J.-P. Couwenberg. *L'indispensable pour maîtriser la couleur*. Marabout, Allier, Belgique, 1992.