Ingénierie de l'Interaction Homme-Machine Dirigée par les Modèles

Jean-Sébastien Sottet[1], Gaëlle Calvary[1], Jean-Marie Favre[2]

1. Université de Grenoble, CLIPS-IMAG

2. Université de Grenoble, LSR-IMAG

Résumé

Alors qu'en Ingénierie de l'Interaction Homme-Machine les modèles étaient restés jusqu'ici associés à la décevante génération automatique des Interfaces Homme-Machine (IHM), ils reviennent aujourd'hui, investis d'espoir, comme solution à la diversité des contextes d'usage en informatique ambiante. Par contexte d'usage, on entend un triplet <utilisateur, plate-forme, environnement> définissant les conditions humaines, matérielles, logicielles, lumineuses, sonores, sociales, etc. hébergeant l'interaction. Récemment, un cadre de référence en Plasticité des IHM articulait la méthode de conception des IHM autour d'un ensemble de modèles et de transformations. Ce cadre de référence est ici formalisé par la définition de métamodèles explicites. Cet article montre comment la notion de plate-forme, et plus généralement les concepts de l'Ingénierie Dirigée par les Modèles, s'appliquent au domaine des IHM. La conception d'une IHM y est vue comme une série de correspondances entre cinq métamodèles : les tâches utilisateur, concepts du domaine, espaces de travail, interacteurs et finalement programme. Ce cadre de référence, ici exprimé en termes de l'IDM, pourra, en perspective, servir de grille d'analyse à l'examen de l'état de l'art en matière de langages et outils d'ingénierie d'IHM.

1 Introduction

Avec l'informatique ambiante, c'est-à-dire diffuse et mobile [2], les contextes d'usage se diversifient : l'utilisateur évolue dans un environnement varié (la rue, son bureau, sa maison) et recourt, de façon opportuniste, à des plates-formes d'interaction diverses (PC, PDA, téléphone, console de jeu, montre, etc.). Si cette gamme de plates-formes semblait jusqu'ici maîtrisable, elle ne l'est plus dès lors qu'on envisage l'assemblage et le désassemblage dynamiques de plates-formes : le rapprochement de deux écrans donne, par exemple, l'opportunité d'une plus grande surface d'affichage (Figure 1).

Figure 1 : Déconfinement des IHM.
Jusqu'ici étriquées à la boîte grise qu'était l'ordinateur, les IHM se déploient désormais dans des espaces interactifs hétérogènes et dynamiques.

Désormais, les IHM se déploient dans des espaces interactifs hétérogènes et dynamiques : elles

s'étalent sur un ensemble de plates-formes, migrent de l'une à l'autre ou, au contraire, se reconcentrent en fonction de l'arrivée ou du départ de ressources. Ces migrations partielles ou totales s'accompagnent souvent de remodelages, l'IHM ne pouvant, à l'évidence, être la même sur grand et petit écran. Si les méthodes "au cas par cas" ont fait leur preuve pour des contextes d'usage stables, figés à la conception, elles se heurtent aujourd'hui à la diversité et variabilité de ces contextes. La Plasticité des Interfaces étudie, en réponse, la capacité d'une IHM à s'adapter à son contexte d'usage dans le respect de son utilisabilité [14]. Les approches basées modèles sont reconsidérées dans ce cadre [1] : il ne s'agit plus, comme dans le passé [3], de générer automatiquement une IHM à partir d'une description plus abstraite, mais d'unifier la conception et l'exécution d'IHM autour de modèles, l'adaptation de l'IHM étant vue comme une transformation de ces modèles. Le rapprochement des communautés IHM et IDM est alors prometteur : il s'agit, d'une part, d'expliciter les métamodèles impliqués dans la méthode de conception, d'autre part, d'en décrire et cataloguer les transformations. Ce travail est précieux pour la capitalisation et la dissémination du savoir-faire.

De façon plus générale, l'objectif de l'article est de montrer qu'IHM et IDM sont vouées à être intégrées et qu'une telle intégration est bénéfique pour chaque domaine. L'ancrage se situe dans l'adaptation des systèmes (interactifs en IHM) à différentes plates-formes. Ce point est revendiqué aussi bien en plasticité des IHM qu'en IDM.

Après une première section qui pose les concepts de base en IHM et IDM, la section 3 présente brièvement une démarche de conception d'IHM plastiques. Cette section introduit un cas d'étude très simple qui sera utilisé à titre d'illustration tout au long de l'article. La section 4 introduit brièvement les cinq métamodèles à la base de la conception d'IHM. La section 5 se concentre sur les correspondances entre ces métamodèles et donne un exemple de transformation entre deux d'entre eux. L'article énonce en conclusion un ensemble de perspectives qu'ouvre ce travail.

2 Concepts de base

Cette section pose les bases respectives de l'IDM et de l'ingénierie de l'Interaction Homme-Machine (IIHM). Elle justifie, en synthèse, le décloisonnement des recherches.

2.1 IDM

Bien qu'initialement ce soit le standard MDA (Model Driven Architecture) de l'OMG qui ait donné lieu à l'Ingénierie Dirigée par les Modèles, cette approche en dépasse largement le cadre. Le "MDA" est un standard industriel [4] ayant pour point de départ la séparation des modèles indépendants des *plates-formes* (Platform Independent Model ou PIM) et des modèles spécifiques aux plates-formes (Platform Specific Model ou PSM). Dans le contexte de l'OMG, la notion de plate-forme fait principalement référence à des intergiciels comme par exemple EJB ou DotNet. L'OMG a abandonné l'idée de la plate-forme unique ou unifiable. Au contraire, il s'agit désormais d'être capable de capitaliser le savoir-faire indépendamment des plates-formes. Selon la vision de l'OMG, c'est la raison d'être des PIM. Ces modèles abstraits seront ensuite transformés en PSM via des transformations explicites. Ces transformations pourront être décrites, dans le futur, en utilisant le standard émergeant QVT (Query View Transform) [4].

Bien que l'approche préconisée par l'OMG soit séduisante, les standards produits par cette organisation sont souvent complexes, parfois centrés vers des domaines d'applications ou des technologies particulières, comme les intergiciels ; de plus les concepts sous-jacents sont souvent mal définis. En fait, il est important de distinguer dans le reste de l'article le MDA, qui est un standard industriel particulier, de l'Ingénierie Dirigée par les Modèles (IDM) ou Model Driven Engineering (MDE) en anglais [6]. Cette distinction a été décrite plus précisément dans le rapport de

l'Action Spécifique MDA du CNRS [5].

Dans le cadre de cet article, nous nous bornerons à présenter les concepts essentiels de l'IDM, sans faire référence aux technologies sous-jacentes qui sont pourtant en pleine expansion [6]. Notre objectif se borne ici à montrer l'adéquation de ces concepts au développement des IHMs. Le lecteur se rapportera à [7][8][10][11] et [12] pour une discussion approfondie portant sur chaque concept ainsi que pour la description de mégamodèles rendant explicites les relations entre concepts.

En fait, trois concepts essentiels sont à la base de l'IDM : les modèles, les métamodèles et les transformations. Chacun de ces concepts correspond à une relation différente :

- **Les modèles et la relation *Représente* (μ)**. Il n'existe pas de définition universelle pour le concept de modèle, cependant on peut trouver un consensus relatif autour du fait que *modèle* et *système étudié* sont deux rôles complémentaires. La relation qui lie un modèle à un système étudié est noté μ. De manière simplifiée, on peut dire qu'un modèle est une représentation simplifiée d'un système, représentation utilisée pour répondre à des questions à la place de ce système. Pour une discussion plus complète, le lecteur pourra se reporter à [7], article dédié à l'étude de la relation μ.
- **Les métamodèles et la relation *EstConformeA* (χ)**. Un métamodèle est un modèle d'un langage de modélisation. Le concept de métamodèle mène à la relation notée χ souvent appelée *EstConformeA*. Cette relation permet de décrire le lien entre un modèle et le métamodèle modélisant le langage dans lequel il est décrit. Dans le cas particulier de la technologie orientée objet, la relation *InstanceDe* correspond à cette notion de conformité, mais ce n'est là qu'un cas particulier. Par exemple une phrase peut être conforme (ou non) à une grammaire. L'article [8] est dédié à l'étude de cette relation χ.
- **Les transformations et la relation *EstTransforméEn* (τ).** Mettre l'accent sur la relation χ permet d'envisager des transformations automatisées d'un modèle vers un autre. La relation τ introduite dans [12] et [13] relie deux modèles. Le coeur de l'IDM consiste à représenter de manière explicite les transformations donnant lieu à la notion de modèle de transformation. Ces modèles se basent sur les métamodèles des modèles que l'on désire transformer [13]. La relation *EstTransforméEn* consiste donc à prendre des éléments d'un ou plusieurs modèles et de les mettre en correspondance avec d'autres éléments de modèle(s). Cette mise en correspondance provient d'un savoir-faire métier. Les transformations de modèles sont une manière de capitaliser le savoir-faire.

Il serait possible de discuter longuement sur chacune des relations identifiées ci-dessus, et a fortiori sur la combinaison de ces relations. Ce n'est cependant pas du tout l'objectif de cet article. Ces relations sont introduites ici uniquement pour donner un cadre de référence utile à l'étude des IHM.

Tout au long de ce papier, nous utiliserons également l'architecture pyramidale de l'OMG qui distingue, entre autres, le niveau **M1**, correspondant aux modèles, du niveau **M2**, correspondant aux métamodèles. La meta-pyramide de l'OMG et les critiques associées sont décrites en détail dans l'article [8]. Nous indiquerons systématiquement pour chaque figure présente dans l'article à quel niveau elle se situe (même si cette notion de niveau n'est que relative).

Les modèles (**M1)** seront dans un cadre gris clair et les métamodèles (**M2**) auxquels ils sont conformes (χ) dans un cadre gris foncé (voir par exemple les figures 2 à 9). De plus, il est nécessaire parfois de distinguer syntaxe **C**oncrète d'un modèle (celle visible par l'utilisateur) et syntaxe **A**bstraite (celle manipulée par les outils). Par mesure de simplification, au niveau M1, seule la forme concrète des modèles a été donnée dans une notation "métier". Par contre au niveau M2, nous avons distingué les métamodèles correspondant à la syntaxe abstraite du langage décrit (**M2A**), des métamodèles correspondant à leur syntaxe concrète **M2C** (voir les Figures 4, 5, 6 et 7). Bien qu'il soit possible de définir rigoureusement les métamodèles "concrets" (M2C) ainsi que leur liaison

avec les métamodèles "abstraits" (M2A), ceci n'a pas été fait pour alléger le discours centré ici sur le métier des IHM.

2.2 IHM

Dans la vie d'une IHM, on distingue classiquement deux phases : sa construction et son exécution. La construction est dirigée par un contrat qualité en termes d'*utilité* et d'*utilisabilité* :

- *L'utilité* spécifie les services utilisateur attendus. Par exemple, pouvoir régler à distance la température de sa maison ;
- *L'utilisabilité* pose des requis en termes de facilité d'apprentissage, d'efficacité, etc. On exigera, par exemple, que la *tâche* soit réalisable en moins de trois clics. La spécification peut s'appuyer sur des critères d'ergonomie tels que Bastien&Scapin [23] ou Nielsen [24] par exemple.

L'utilité et l'utilisabilité sont définies pour des contextes d'usage donnés. On spécifie :

- le profil utilisateur par des données générales (âge, taille, etc.), ses compétences métier et informatiques ;
- la plate-forme d'interaction (PC, PDA, PC couplé à PDA, etc.) ;
- l'environnement physique (lumineux, sonore, etc.), social, etc. accueillant l'interaction.

La construction s'articule ensuite classiquement en différents niveaux d'abstraction allant de la spécification du domaine (concepts et tâches) à l'IHM finale programmée.

Tout d'abord la spécification du domaine spécifie les tâches utilisateur et les concepts manipulés dans les tâches. Par exemple, "*régler la température de la maison"* manipule les concepts de "*température"* et de "*maison"*. Trois types de modèles relèvent de ce niveau (nous utilisons dans cette partie le terme "modèle" au sens large tel qu'utilisé en IHM. Le reste de l'article fera bien évidemment la distinction très explicite entre le niveau modèle et métamodèle).

- **(C) le modèle des Concepts** décrivant les concepts du domaine. Les modèles de concepts peuvent être, par exemple, décrits par des diagrammes de classes UML ;
- **(T) le modèle des Tâches** décrivant les tâches utilisateur en termes d'objectifs et de procédures. Les procédures décomposent récursivement les tâches en sous-tâches jusqu'à atteindre les tâches élémentaires, c'est-à-dire des tâches qui ne seraient décomposables qu'en actions physiques ("*bouger la souris*", "*saisir au clavier*"). Des notations telles que ConcurrTaskTree (CTT outillée en CTTE) [26] ont été proposées à ces fins ;
- **(C-T) le modèle concepts-tâches** liant ces deux modèles (C et T) par le référencement, dans les tâches, des concepts manipulés dans ces tâches.

Le raisonnement porte ensuite sur les espaces de travail, donnant lieu au **modèle des Espaces (E)**. Un espace de travail est une zone d'interaction permettant la réalisation de tâches utilisateur. Les espaces sont identifiés sur la base des tâches utilisateur. Ils contiennent les concepts du domaine (C) manipulés dans ces tâches (T) et s'enchaînent normalement conformément aux tâches utilisateur qu'ils représentent.

L'IHM est ensuite matérialisée en termes d'interacteurs (aussi dits "widgets" en anglais). Le **modèle des Interacteurs (I)** détermine quelle sera l'incarnation des espaces de travail (fenêtres ou canevas), mais aussi celle des concepts du domaine (libellés, listes, champs de texte, etc.) et des enchaînements entre espaces (simples séparateurs ou boutons, onglets, liens hypertexte, index de défilement, etc.).

La maquette est ensuite programmée dans un ou plusieurs langages de programmation ou d'implémentation (php, java, html, javascript, etc.) en faisant appel à des librairies spécifiques (swing, awt, etc.). A ce niveau, on peut donc considérer différents **modèles de Programmes** (**P**).

Dans cet article, nous ne donnerons qu'une vision très simplifiée de ce dernier niveau, l'accent étant résolument porté sur la démarche de conception.

Le défi est d'étendre, formaliser et outiller la méthode actuelle pour répondre aux besoins de l'informatique ambiante. La vision est la suivante :

- Capitaliser des IHM méta-décrites : une IHM se décrit à tout niveau d'abstraction (C,T,E,I,P), en connaît la raison d'être et les hypothèses, notamment le contexte d'usage qu'elle suppose. En cas de changement de contexte d'usage, ces IHM préfabriquées seront des ressources potentielles à utiliser telles quelles, assembler ou scinder pour s'adapter au nouveau contexte d'usage ;
- Dynamiser ces modèles pour, par exemple, à l'exécution, connaître l'état d'interaction et en assurer la continuité lors de migrations ou de remodelages.

2.3 Synthèse

Alors que le standard MDA a été conçu initialement pour assurer une indépendance des plates-formes intergicielles telles que EJB, CCM ou DotNet, la problématique des interfaces plastiques consiste à gérer la multiplicité des plates-formes d'interaction telle qu'un ordinateur de bureau, un PDA, une montre, etc. Il est clair que cela ne peut se faire à un coût raisonnable que si l'on dispose de modèles d'IHM indépendants des plates-formes, modèles à partir desquels on peut dériver, par transformations successives et contrôlées, des modèles spécifiques. De telles transformations représentent un savoir-IHM qu'il est important de capitaliser et d'organiser. Cela mène naturellement à l'IDM. Ce rapprochement des deux communautés, IDM et IHM, fait l'objet de l'article.

3. Démarche et cas d'étude

Appréhender à la fois un nouveau domaine d'application et une approche dirigée par les modèles est souvent difficile dans le contexte d'un article. Aussi, nous appuyons-nous sur un cas d'étude simple. Ce cas d'étude est décrit dans une première section avant d'énoncer la vision globale de la démarche.

3.1 Cas d'étude

Le cas d'étude ici retenu se veut pédagogique et à seule visée d'illustration. Les modèles et métamodèles présentés dans cet article sont volontairement simplifiés, l'objectif n'étant pas de définir un cadre normatif mais plutôt de montrer la vision globale de la démarche. Le cas d'étude porte sur la programmation du chauffage à distance. La maison est limitée à deux pièces : salon et cellier. La figure 2 présente un spectre d'IHM motivées par des tailles d'écran différentes. Ces IHM sont des maquettes programmées. Elles relèvent du dernier niveau d'abstraction : programmation (P). Les divergences portent sur les tâches utilisateur (T) et les interacteurs (I) :

- Tâches (T): alors que les versions a/ à e/ permettent le réglage de la température dans le salon et le cellier, la version f/ se limite au salon. La tâche "*régler cellier*" est trop peu fréquente pour apparaître sur une montre dont la surface limitée impose des économies de pixels ;
- Interacteurs (I): des différences apparaissent dans la présentation de la navigation et des concepts. En a/, la navigation entre pièces est nulle (les deux pièces salon et cellier sont directement observables) alors qu'elle se fait par bouton en b/ (bouton Cellier), liste déroulante en c/ et d/, lien hypertexte en e/. En termes de concepts, le réglage de la température se fait par potentiomètre en a/, b/ et c/, par liste déroulante en d/ et champs de texte en e/ et f/.

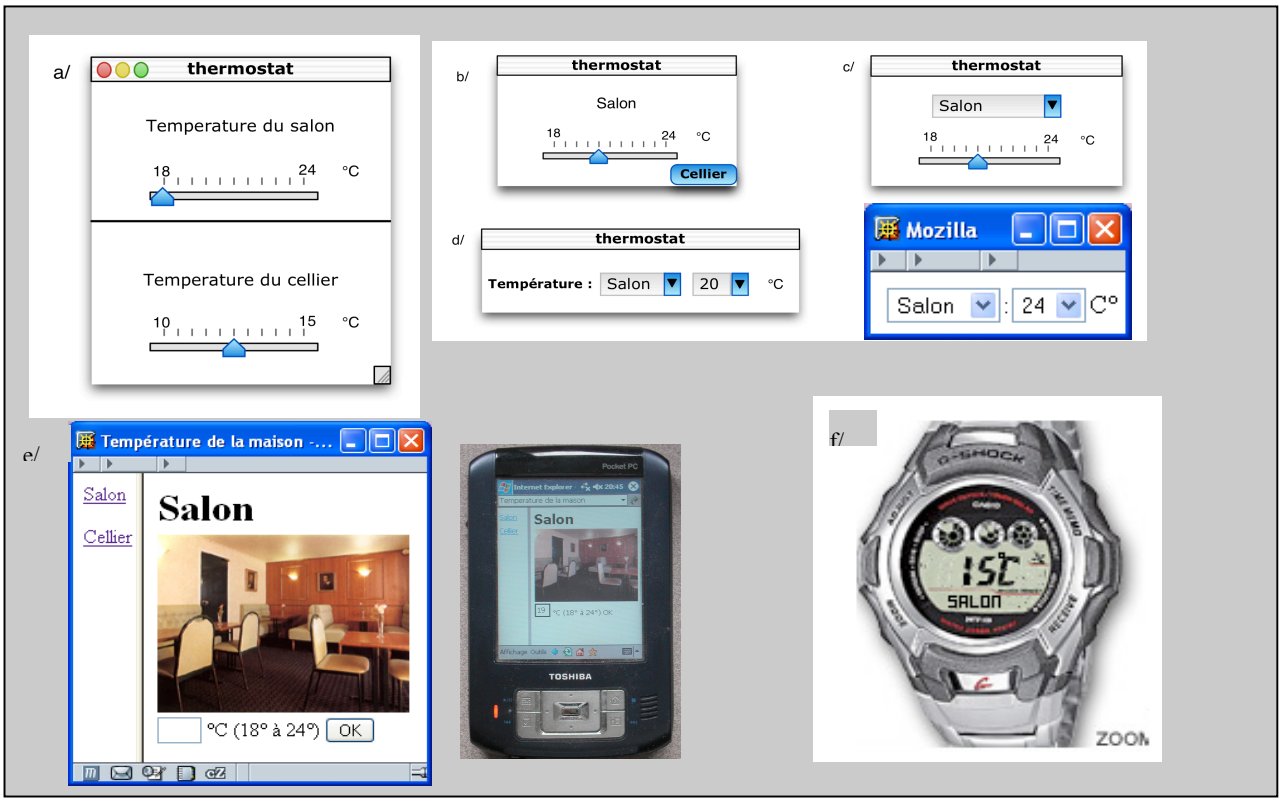

Figure2 : Différentes IHM variant pour l'essentiel par la taille d'écran qu'elles requièrent.

3.2 Méthode de conception

Comme évoqué en section 2.1, l'ingénierie d'une IHM s'appuie sur cinq types de modèles : Tâches utilisateur (T), Concepts du domaine (C), Espaces de travail (E), Interacteurs (I) et Programmes (P). La figure 3 illustre le caractère incrémental de la méthode de conception dans un esprit IDM en distinguant, en particulier, les modèles (M1) de leurs métamodèles (M2). Elle précise aussi dans quelles sections/sous-sections chaque modèle/métamodèle sera développé. Les correspondances (à gauche) feront l'objet de la section 5.

4. Modèles et Métamodèles

Dans cette section sont introduits les différents types de modèles utilisés en IHM. **Cette liste ne se veut en aucun cas exhaustive**. Rappelons que l'objectif ici n'est pas de définir des langages mais de poser les bases pour une rationalisation et opérationnalisation de la méthode de conception via l'IDM. Les (méta)modèles T, C, E, I et P sont traités dans cet ordre.

4.1 (Méta)-Modèle de Tâches (T)

Une tâche est un couple <but, procédure> [15]. Le but définit l'objectif que l'utilisateur souhaite atteindre (par exemple, *"gérer la température de la maison"*). La procédure précise la façon selon laquelle il souhaite (ou peut) l'atteindre (par exemple, *"sélectionner la pièce* PUIS *gérer la température de cette pièce"*). La procédure est une décomposition récursive en sous-tâches (*"sélectionner la pièce"* et *"gérer la température de cette pièce")*, les tâches étant liées entre elles par des *opérateurs*. On distingue deux types d'opérateurs selon la nature logique (ou, et) ou temporelle (séquence, parallélisme) qu'ils représentent. Dans l'exemple, l'opérateur est la séquence (PUIS).

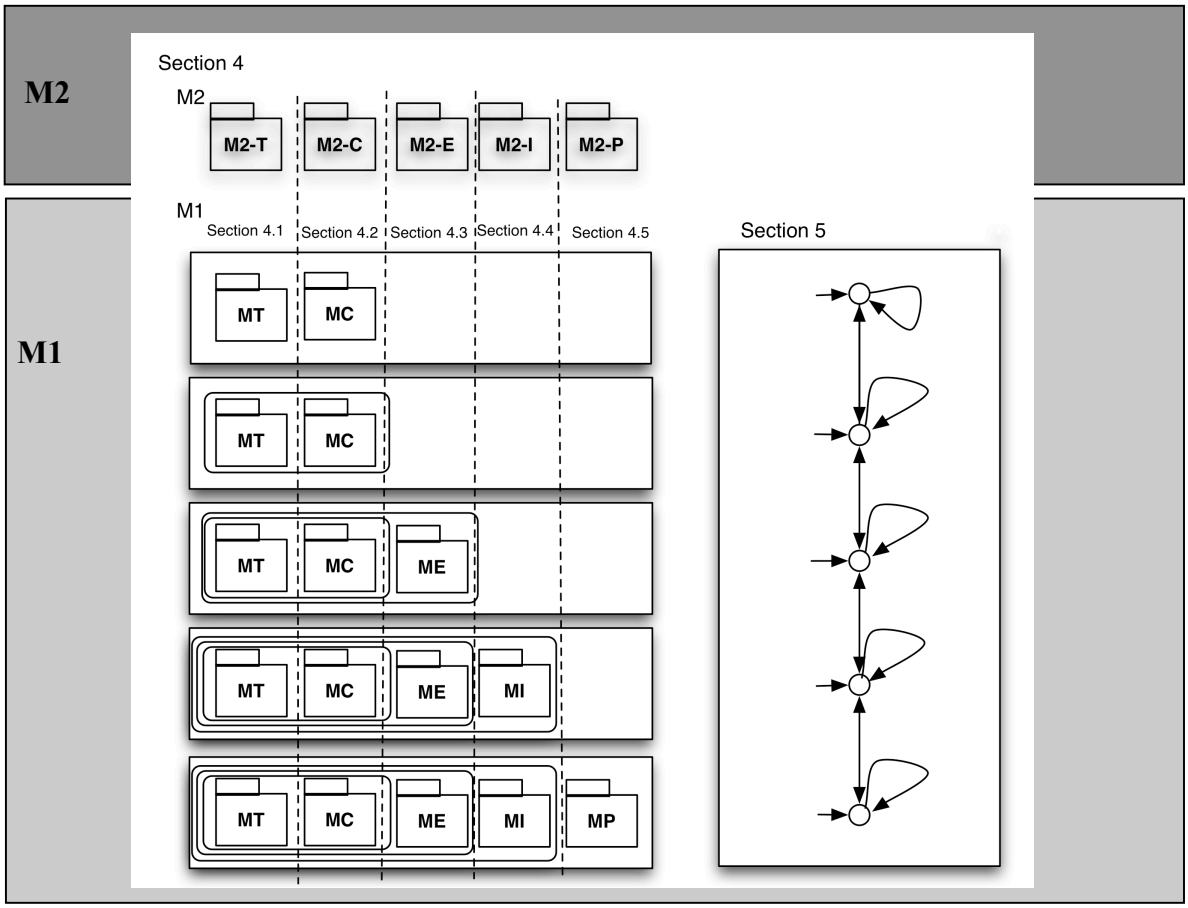

Figure 3 : Méthode de conception des IHM.

Les tâches peuvent être décorées pour exprimer des particularités, par exemple, leur fréquence, criticité, complexité, caractère optionnel ou itératif, valeurs par défaut, etc. Dans le cas d'étude, la tâche *"Gérer température de la maison"* est itérative (notation *). La tâche *"Sélectionner pièce"* a le salon comme valeur par défaut.

Les modèles des tâches sont traditionnellement représentés graphiquement par des arbres. Il existe plusieurs notations (par exemple, CTT [26]). La figure 4 sur la page suivante propose des métamodèles de langages abstrait (M2A-T) et concret (M2C-T) illustrés sur le cas d'étude (M1-T). Il est intéressant de noter que les variantes d'IHM (Figure 2) correspondent en fait à des modèles de tâches potentiellement différents : alors que dans les versions c/, d/ et e/, l'utilisateur spécifie explicitement la pièce dont il veut gérer la température, cette spécification est indirecte en a/ : la pièce se déduit du potentiomètre manipulé. En b/, une contrainte temporelle de type séquence est fixée entre le salon et le cellier. Cette contrainte se répercute dans le bouton de navigation *Cellier* imposant un ordre dans la spécification des températures.
En pratique, l'énumération des concepts manipulés dans les tâches se fait par le biais de décorations. La section suivante traite des concepts du domaine.

4.2 (Méta)-Modèle des Concepts (C)

Le modèle des concepts décrit les entités du domaine (nommées concepts) manipulées dans les tâches. Il explicite, par exemple, le fait qu'une maison soit composée de pièces (salon ou cellier) et que la température y soit appréciée localement (pièce) ou globalement (maison). Une température évolue entre des valeurs minimale et maximale et s'exprime dans une unité. Le modèle des concepts s'appuie sur les relations d'héritage et d'association telles que définies dans le diagramme des classes UML. La figure 5 en propose des métamodèles abstrait (M2A-C) et concret (M2C-C) (simplifiés !) illustrés sur le cas d'étude (M1-C).

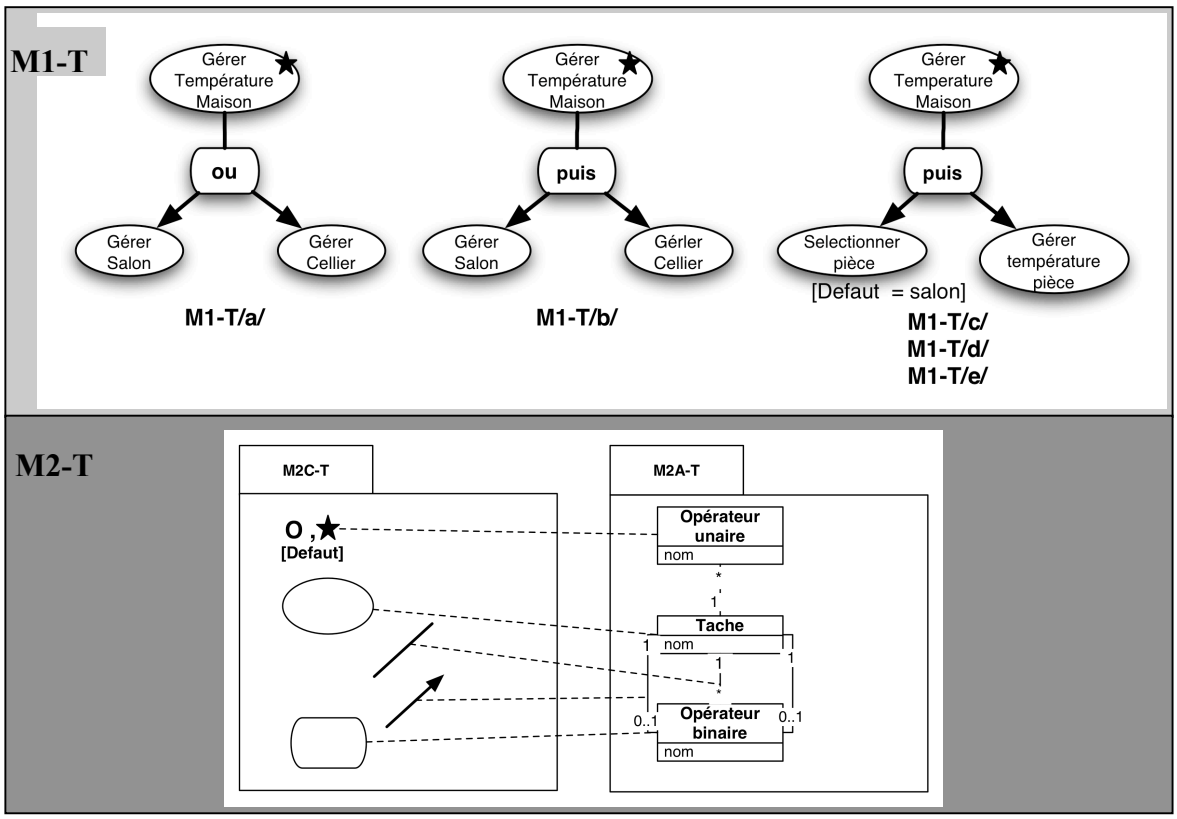

Figure 4 : Modèle et métamodèles de Tâches (T)

Les décorations sont traitées en opérateurs unaires.
Les opérateurs logiques et temporels sont considérés comme binaires.

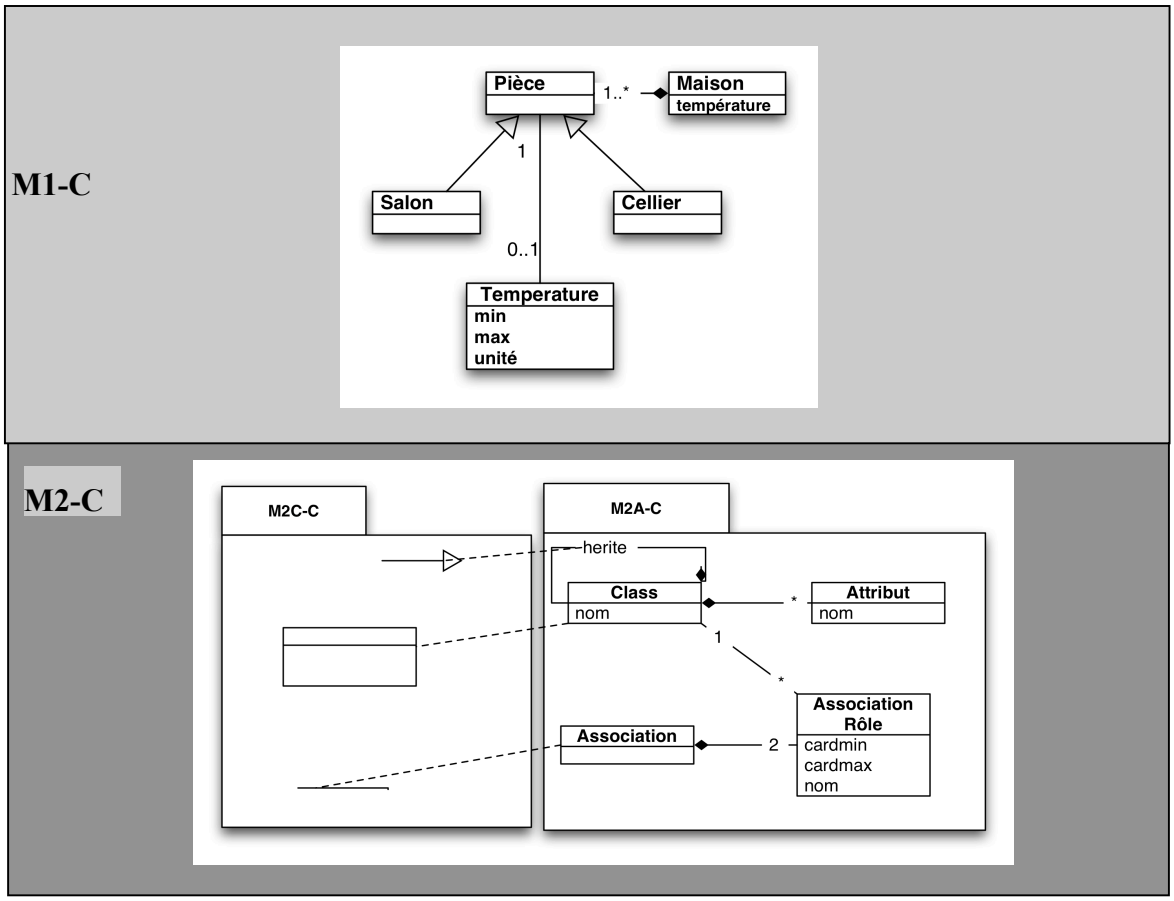

Figure 5 : Modèle et métamodèles de Concepts (C).

4.3 (Méta)-Modèle des Espaces de travail (E)

Le modèle des espaces de travail dérive du modèle des tâches. Il associe des espaces de travail aux tâches utilisateur, spécifie l'enchaînement entre espaces et le contenu conceptuel de ces espaces. Nous nous focalisons ici sur l'identification des espaces et leur enchaînement. La figure 6 en propose des métamodèles abstrait et concret, illustrés sur le cas d'étude. Dans l'illustration, un espace de travail est associé par tâche et l'enchaînement entre espaces calque fidèlement les opérateurs entre tâches. Ainsi :

- Dans l'exemple a/, l'espace associé à la tâche *"Gérer la température de la maison"* donne accès aux deux espaces de travail fils *"Gérer salon"* et *"Gérer cellier"*. Ces sous-espaces redonnent accès à l'espace mère une fois la sous-tâche accomplie ;
- Dans l'exemple b/, l'espace associé à la tâche *"Gérer la température de la maison"* donne accès au premier fils *"Gérer salon"* qui donne accès à *"Gérer cellier"* qui redonne accès à la mère ;
- Dans les exemples c/, d/ et e/, le principe de navigation est le même qu'en b/. La différence provient de la nature des tâches hébergées dans les espaces. Il s'agit ici de *"Sélectionner une pièce"* pour pouvoir en gérer la température (*"Gérer la température de cette pièce*").

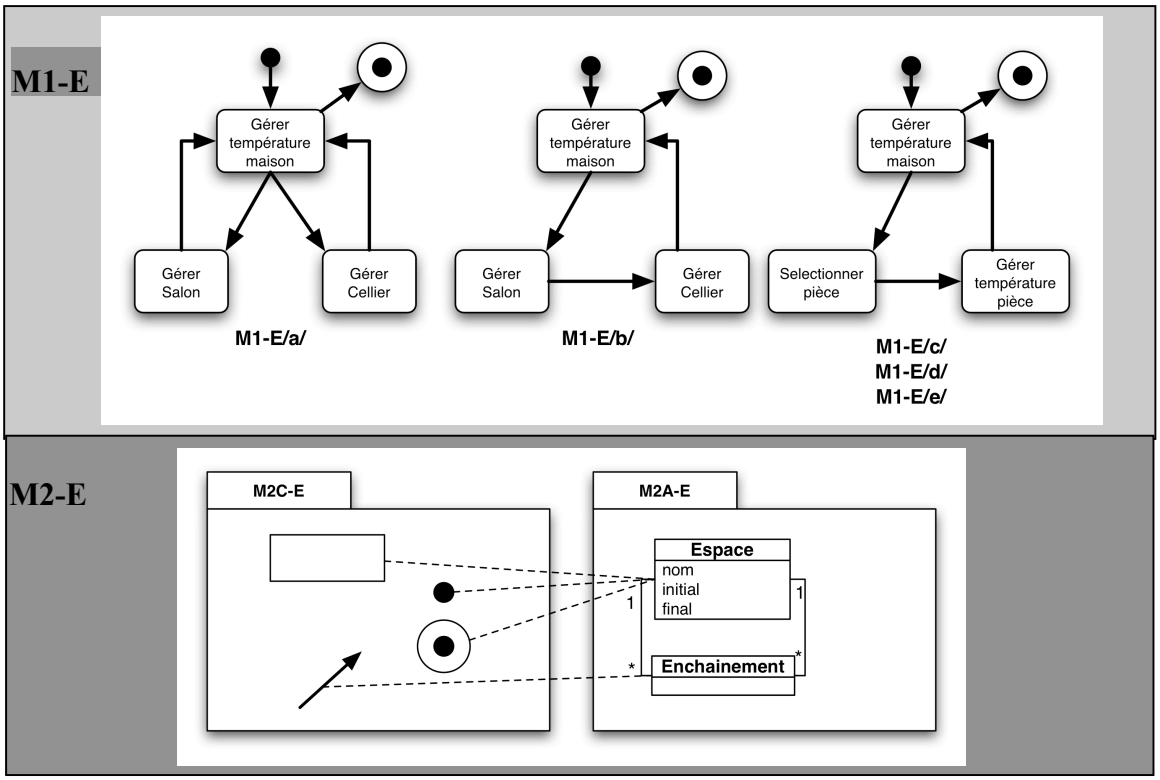

Figure 6 : Modèle et métamodèles des espaces de travail (E).

4.4 (Méta)-Modèle des Interacteurs (I)

Les choix de présentation en termes d'interacteurs et d'agencement sont faits à ce niveau :

- les espaces de travail deviennent conteneurs (fenêtres ou canevas en graphique) ;
- l'enchaînement entre espaces devient interacteurs de navigation (le bouton Cellier par exemple) ;
- les concepts du domaine et opérations applicables deviennent interacteurs (par exemple, le potentiomètre, la liste déroulante ou le champ texte pour la spécification de la température).

Le niveau de précision du modèle est variable. La figure 7 sur la page suivante met, par exemple, l'accent sur l'agencement spatial. Les libellés, couleurs, etc. ne sont pas spécifiés.

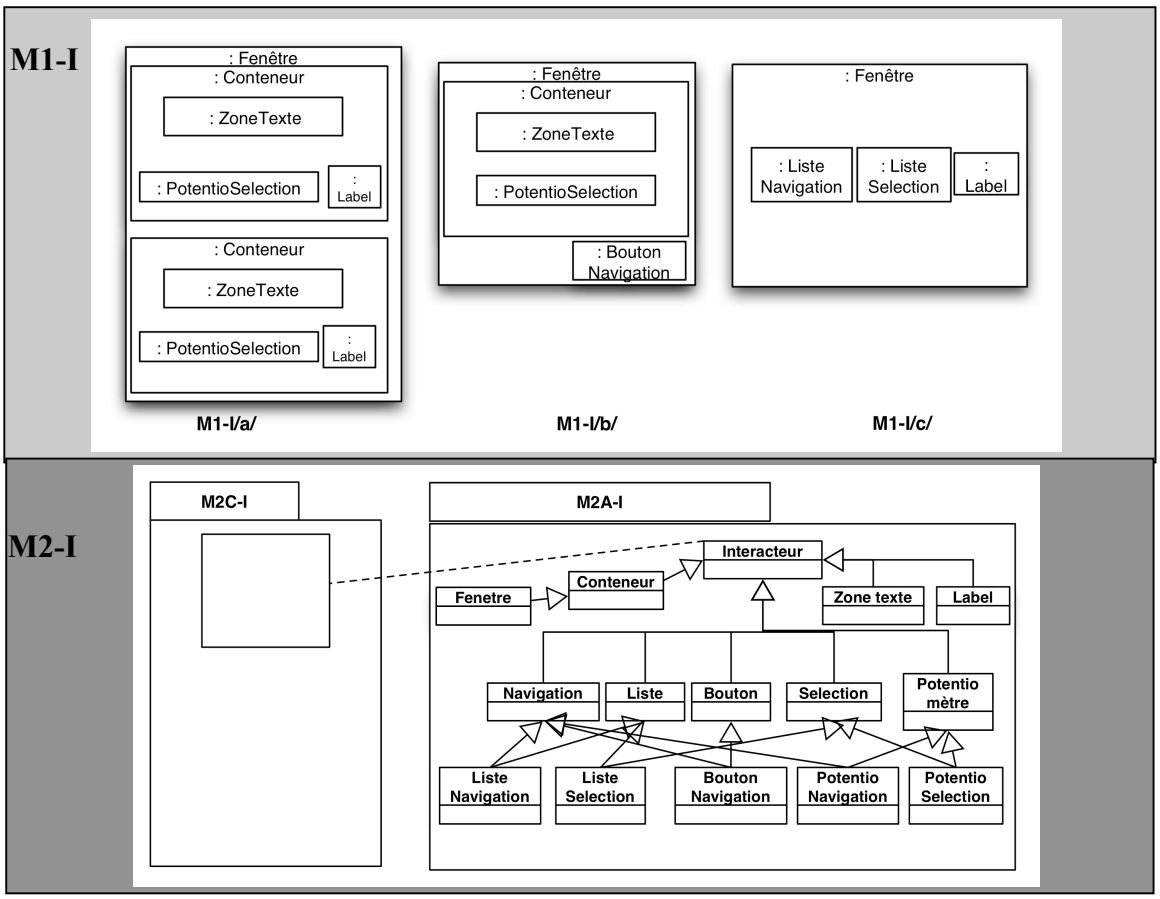

Figure 7 : Modèle et métamodèles d'espaces de travail (E).

4.5 (Meta)-Modèle des Programmes (P)

Le niveau "programme" correspond à la représentation interprétable/compilable de l'IHM. Elle est écrite dans un (ou plusieurs) langage(s) de programmation et/ou dans tout autre langage d'implémentation utilisé par un "programmeur". La figure 8 en donne un exemple HTML. Son exécution produit, sous Mozilla, la page de droite. Ces deux exemples relèvent du même niveau M1. On note dans cet exemple la balise <select> pour le choix de la liste déroulante. Cet interacteur fait partie du métamodèle (M2-P) d'HTML. Le métamodèle n'est pas donné ici par manque de place. Il existe, par ailleurs, une multitude de techniques d'implémentation. Cette partie basse du cycle de développement est, de toute façon, largement étudiée en IDM.

M1-P

```
<table border="0" cellpadding="0">
    <select>
        <option>Salon</option>
        <option>Cellier</option>
      </select>
  </td>
    <td>:</td>
    <td>
      <select>
        <option>24</option>
        <option>23</option>
        ...
        <option>18</option>
      </select></td>
    <td>°C</td>
</table>
```

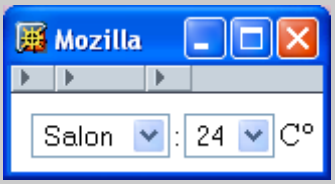

Figure 8 : Modèles de programmes (P).

5. Correspondances et transformations

Les métamodèles présentés en section 4 ne sont évidemment pas indépendants. L'idée sous-jacente de l'IDM est de rendre explicites les correspondances entre ces métamodèles. Cette section en présente une amorce. Un traitement complet nécessiterait un espace bien plus important que celui dont nous disposons dans cet article, d'autant plus que chaque transformation doit être adaptée aux métamodèles précis retenus pour chaque niveau. Nous nous limitons ici aux transformations verticales, c'est-à-dire concrétisant les concepts-tâches (C-T) en espaces de travail (E), les espaces en interacteurs (I), ou les interacteurs en programmes (P). A titre d'exemple, la première section traite du cas particulier de la transformation des tâches en espaces (T->E). La deuxième section donne une vision plus globale, couvrant l'ensemble des correspondances mises en œuvre dans la méthode de conception, mais de manière très simplifiée.

5.1 Exemple de correspondance : Tâches-Espaces (T->E)

Dans la transformation Tâches-Espaces, les tâches sont transformées en espaces ; les opérateurs entre tâches en enchaînements entre espaces. Dans la figure 9 (M1), on voit par exemple qu'à chaque tâche est associé un espace et que l'opérateur OU donne lieu aux enchaînements entre espaces : l'espace mère donne accès aux deux filles. La figure 9 en propose un métamodèle abstrait.

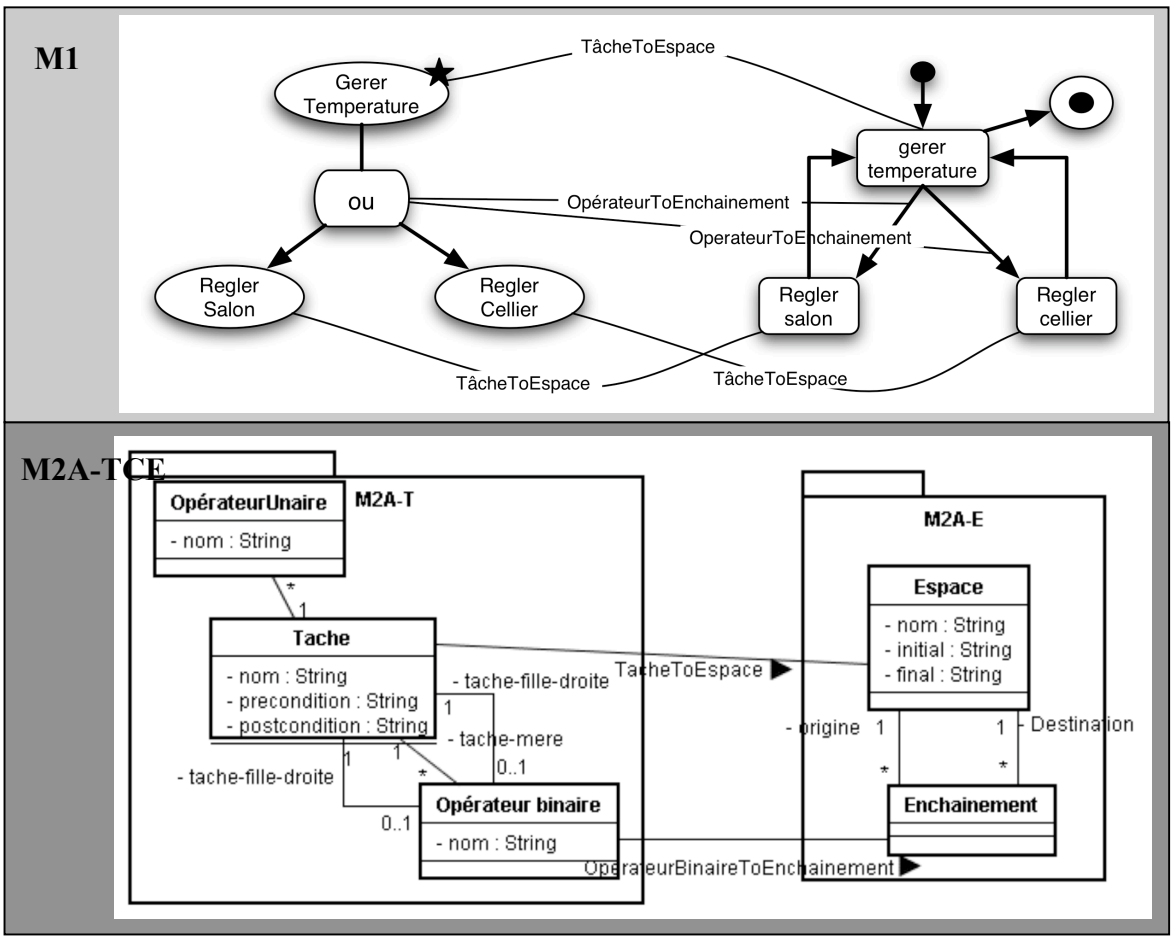

Figure 9 : Modèle et métamodèle abstrait de transformation des tâches en espaces de travail (T->E).

Bien évidemment, mettre les différents éléments en correspondance n'est pas suffisant et il ne s'agit pas là d'une vision opérationnelle. Il est intéressant d'utiliser un langage de transformation pour modéliser / implanter cette transformation. Les règles de transformation sont écrites sur la base du métamodèle (M2A-TCE). Par exemple, ci-après, la transformation est exprimée en langage ATL développé à l'Université de Nantes [25]. La première règle de cette transformation illustre le lien TacheToEspace de la figure 9. Ici simplifiée, elle consiste à créer un espace pour chaque tâche et lui

allouer le nom de la tâche. La règle 2 illustre le lien OperateurBinaireToEnchainement. Pour des raisons de concision, nous ne considérons que l'opérateur « OU ». La règle est écrite en deux parties : la première consiste en une sélection des opérateurs binaires de type « ou » ; la deuxième décrit l'accès donné par l'espace représentant la tâche « mère » aux espaces représentant ses deux filles.

```
-- Transformation des tâches en espaces
module M2A-T to M2A-E {
from M1A-T : M2A-T to M1A-E : M2A-E

-- Regle 1 : L'espace prend le nom de la tâche
rule TacheToEspace {
    from t : M2-T!Tache
    to e : M2-E!Espace (
        nom <- t.nom

    )
}
-- Règle 2 : Traitement des opérateurs de type ou
rule OperateurOu ToEnchainement{
    from o : M2-T!OperateurBinaire (
                o.name = "ou"
           )
-- la mère donne accès à ses deux filles (gauche et droite)
enchainementGauche : M2-E!Enchainement (
             origine <- [Tache2Espace.e]o.tache-mere,
             destination <-  [Tache2Espace.e]o.tache-fille-droite,
       ),
to enchainementDroit : M2-E!Enchainement (
             origine <- [Tache2Espace.e]o.tache-mere,
             destination <-  [Tache2Espace.e]o.tache-fille-Gauche,
       ),
}
```

5.2 Vision globale sur les correspondances

Au delà de l'exemple précédent, l'approche peut être vue de manière plus globale. La figure 10 présente une cartographie globale mais simplifiée des transformations entre modèles, sous la forme d'une carte au niveau M2 :

- Les tâches manipulent des concepts (lien M2A-T vers M2A-C) ;
- Les tâches sont hébergées dans des espaces de travail, s'enchaînant conformément aux opérateurs entre tâches (liens M2A-T vers M2A-E) ;
- Les interacteurs matérialisent les espaces de travail et enchaînements entre espaces (liens M2A-I vers M2A-E) ; ils incarnent des tâches (liens M2A-I vers M2A-T) et des concepts du domaine (liens M2A-I vers M2A-C).

Par manque de place, seuls quatre métamodèles sont présentés, mais bien évidemment il faut y ajouter le métamodèle des programmes pour la liaison avec les artefacts informatiques utilisés par le programmeur. Quoi qu'il en soit, la figure 10 présente l'avantage de rassembler, en une seule image, les différents niveaux décrits précédemment.

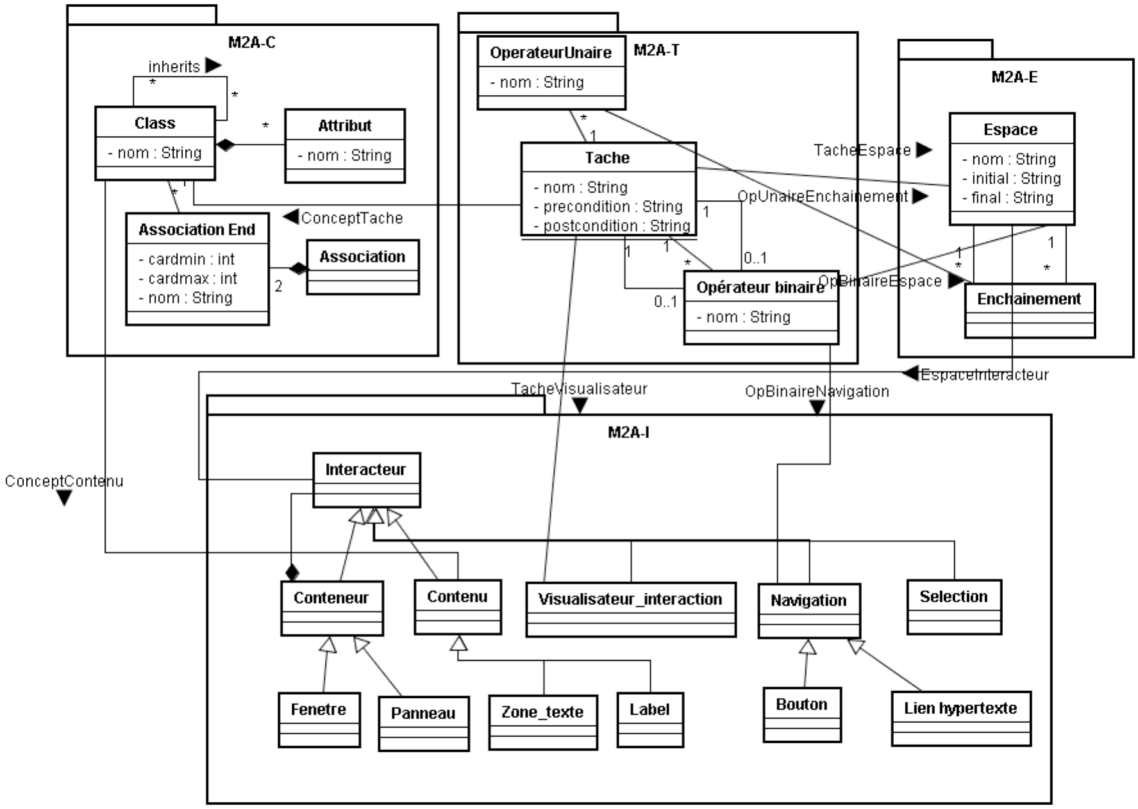

Figure 10 : Cartographie des transformations entre métamodèles.

Alors que le diagramme ci-dessus correspond au niveau métamodèle (M2), les correspondances sont brièvement illustrées au niveau modèle (M1) dans la figure 11. Au niveau M2, les correspondances sont décrites via des associations UML ; au niveau M1 il s'agit bien évidemment de liens. Ainsi le modèle de correspondance au niveau M1 doit être conforme au métamodèle du niveau M2.

L'exemple de la figure 11 correspond au cas d'étude. Il s'agit d'une version simplifiée, étant impossible de dessiner tous les liens. Elle est obtenue comme suit :

Le modèle Concept-Tâche (en haut de la figure) est le premier obtenu. Il s'obtient en référençant les concepts du domaine dans les tâches utilisateur (association ConceptTâche au niveau M2, liens correspondant au niveau M1). Sur l'exemple, la tâche racine de l'application manipule le concept de température tandis que les tâches de sélection et de réglage des pièces manipulent le même concept de pièce.

Aux tâches sont ensuite associés des espaces de travail. Cette transformation d'espaces vers tâches a déjà été illustrée.

Une réification (ou projection) des espaces de travail conduit au modèle des interacteurs. Les transformations EspaceInteracteur correspondent à cette réification. Selon la nature des enchaînements entre espaces, l'organisation des interacteurs peut être spatiale (au travers de conteneurs) ou temporelle (par le biais d'interacteurs de navigation). Ici, nous avons choisi de réifier les enchaînements entre espaces de manière spatiale uniquement. Ceci conduit à créer un conteneur de type fenêtre pour représenter l'espace principal et y intégrer les éléments d'interaction symbolisant les espaces auxquels ils donnent accès. Cependant toutes les informations permettant l'obtention du modèle des interacteurs ne sont pas portées directement dans le modèle des espaces. Il est nécessaire d'avoir un ensemble de mises en correspondance de ce modèle d'interacteurs vers les autres modèles ayant servi à la conception. Dans cet exemple, nous illustrons, une mise en correspondance entre l'unité de la température et l'interacteur (label) symbolisant cette valeur "°C"

(ConceptContenu).

Enfin, après d'autres réifications du modèle, il est possible finalement, pour chacun des interacteurs, de trouver une implémentation correspondante dans la boite à outils considérée. Dans cette dernière étape, seul est montré le lien entre un "frame" (HTML) et un interacteur fenêtre (conteneur). Bien évidemment, en pratique, tous les liens sont conservés.

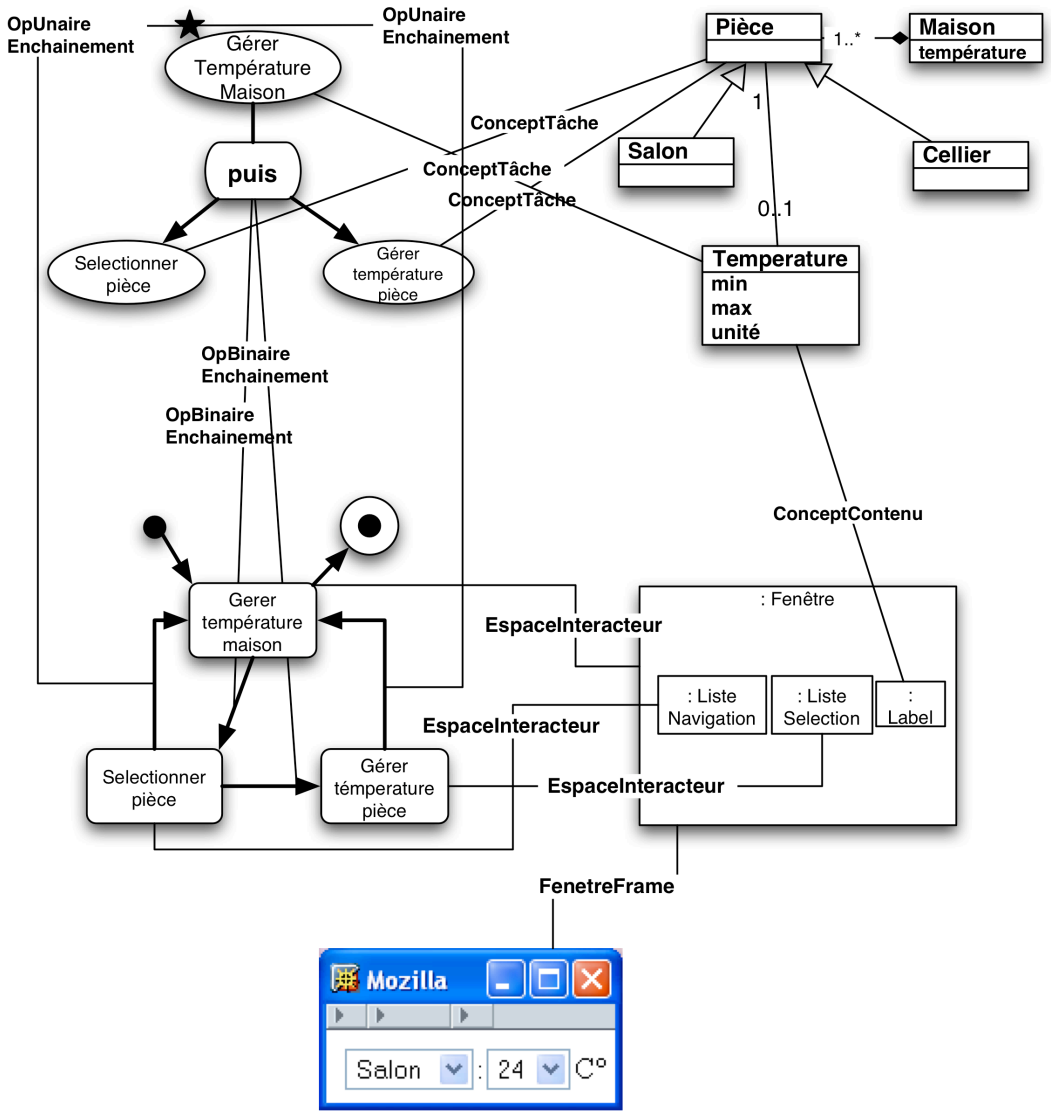

Figure11 : Illustration des transformations entre modèles sur le cas d'étude.
Modèles illustrant quelques associations entre métamodèles

6. Conclusion

Cet article visait à montrer que les concepts de l'Ingénierie Dirigée par les Modèles pouvaient être utilement appliqués au développement d'Interfaces Homme-Machine (IHM). Il a été possible de schématiser, en moins d'une page, les relations entre modèles (et métamodèles) utiles en IHM. Décrire de manière explicite les métamodèles et correspondances permet non seulement de capitaliser les connaissances nécessaires à la conception d'IHM, mais aussi de classifier les travaux existants et en outiller la démarche.

Si le décloisonnement des communautés se veut prometteur en recherche, il nous semble être également pertinent pour l'enseignement. Nous pensons que le travail présenté dans cet article forme

une base pédagogique intéressante pour l'enseignement de l'IHM et de l'IDM. Le pouvoir illustratif des IHM est, en effet, selon nous, bien supérieur aux traditionnels exemples de plates-formes EJB ou DotNet pour lesquelles il est très difficile d'exprimer clairement les projections vers les différentes plates-formes, celles-ci étant bien trop techniques et complexes pour être appréhendées simplement. Il nous apparaît chaque jour plus clairement que l'Ingénierie Dirigée par les Modèles et les Interfaces Homme-Machine sont deux disciplines vouées à être mariées. La nécessité de l'union est encore plus flagrante dès lors qu'on considère la plasticité des IHM pour lesquelles le changement de plates-formes est dynamique. C'est finalement un chemin vers de nombreuses perspectives qu'ouvre cet article.

7. Bibliographie

1. Calvary, G., Coutaz J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A unifying reference framework for multi-target user interfaces, Interacting With Computers, Vol. 15/3, pp 289-308, 2003
2. Lyytinen, K. & Yoo, Y. (2002). Issues and Challenges in Ubiquitous Computing. Communications of the ACM, 45(12), 62-65
3. Myers, B., Hudson, S.E., Pausch, R. Past, Present, and future of user interface software tools, ACM Transactions on Computer-Human Interaction (TOCHI), Volume 7, Issue 1 (March 2000), Special issue on human-computer interaction in the new millennium, Part 1, pp 3 – 28, 2000
4. OMG, Model Driven Architecture, http://www.omg.org/mda
5. AS MDA, Rapport de l'Action Spécifique CNRS MDA, décembre 2004, http://idm.imag.fr/as
6. Model Driven Engineering, Planet MDE, http://planetmde.org
7. J.M. Favre, Foundations of Model (Driven) (Reverse) Engineering: Models - Episode I, Stories of the Fidus Papyrus and of the Solarus, in [9]
8. J.M. Favre, Foundations of the Meta-pyramids: Languages and Metamodels - Episode II, Story of Thotus the Baboon, in [9]
9. Dagstuhl Seminar 04101 on Language Engineering for Model-Driven Software Development, Dagsthul, Germany, February 29-March 5, 2004, DROPS proceedings, http://drops.dagstuhl.de/portals/04101/
10. J. Bézivin, On the Unification Power of Models, Journal of Software and System Modeling, SoSym, May 2005
11. J.M. Favre, Megamodelling and Etymology: Med, Modus, Modulus, Mold, Model, Module, Modification, Mapping, Dagstuhl Seminar 05161 on "Transformation Techniques in Software Engineering", to Appear in DROPS http://drops.dagstuhl.de/portals/05161/
12. J.M. Favre, Towards a Basic Theory to Model Model Driven Engineering, Workshop on Software Model Engineering, WISME @ UML2004, Lisboa, Portugal, October 11, 2004, http://www-adele.imag.fr/~jmfavre
13. J.M. Favre, T. NGuyen, Towards a Megamodel to Model Software Evolution Through Software Transformation, Workshop on Software Evolution through Transformation, SETRA 2004, Rome, Italy, October 2, 2004, Electronic Notes in Theoritical Computer Science, Volume 127, Issue 3, ENTCS ELSVIER.
14. Thevenin, D., Coutaz, J.: Plasticity of User Interfaces: Framework and Research Agenda. In: Proc. Interact99, Edinburgh, A. Sasse & C. Johnson Eds, IFIP IOS Press Publ., (1999) 110–117
15. Balbo S., Evaluation Ergonomique des Interfaces Utilisateur : Un Pas Vers l'Automatisation,

Thèse de Doctorat Informatique, Université Joseph Fourier, Grenoble I, Septembre 1994, 287 pages

16. Mori G., Paternò F., Santoro C. Design and Development of Multidevice User Interfaces through Multiple LogicalDescriptions IEEE Transactions on Software Engineering (August 2004, pp.507-520).

17. Crowle, S. and Hole, L. (2003) ISML: An Interface Specification Meta-Language in DSV-IS 2003 : Issues in Designing New-generation Interactive Systems Proceedings of the Tenth Workshop on the Design, Specification and Verification of Interactive Systems. Eds: Joaquim Jorge, Nuno Jardim Nunes, João Falcão e Cunha. 4-6 June 2003. pp 381-396

18. Vanderdonkt J. -A MDA-Compliant Environment for Developing User Interfaces of Information Systems – CAiSE05

19. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Florins, M., Trevisan, D., UsiXML: A User Interface Description Language for Context-Sensitive User Interfaces, in Proceedings of the ACM AVI'2004 Workshop "Developing User Interfaces with XML: Advances on User Interface Description Languages" (Gallipoli, May 25, 2004), Luyten, K., M. Abrams, Limbourg, Q., Vanderdonckt, J. (Eds.), Gallipoli, 2004, pp. 55-62

20. XML User Interface Language (XUL) Project, www.mozilla.org/projects/xul/

21. Petzold C., "Code Name Avalon: Create Real Apps Using New Code and Markup Model."B. In Microsoft MSDN Magazine Volume 19 Number 1 (January 2004).

22. Azevedo, P., Merrick, R., Roberts, D. "OVID to AUIML - User Oriented Interface Modeling"

23. Scapin, D.,Bastien CH., Ergonomic Criteria for evaluating the ergonomic quality interactive systems. Behaviour & Information Technologie 16,p 220-231,1997.

24. Nielsen J., Conception de sites Web : l'art de la simplicité, Ed. CampuPress, France 2000

25. Université de Nantes, ATL Home Page, http://www.sciences.univ-nantes.fr/lina/atl

26. F.Paternò, ConcurTaskTrees: An Engineered Notation for Task Models, Chapter 24, in Diaper, D., Stanton, N. (Eds.), The Handbook of Task Analysis for Human-Computer Interaction, pp.483-503, Lawrence Erlbaum Associates, Mahwah, 2003.