ALLOGENE:
Modélisation algorithmique par actions nommées.

FOURNIER Jean-Pierre & WIRZ Jacky
Ecole d'Ingénieurs de Genève
4 rue de la praire,
CH-1202 GENEVE.
Tel CH-22 344 77 50
Fax CH-22 344 92 88
eMail: FOURNIERJP@eig.unige.ch
eMail: WIRZ@eig.unige.ch

Thèmes:

Stratégies d'enseignement et d'apprentissage
Environnements de découverte

Abrégé :

La modélisation algorithmique par actions nommées présente de multiples avantages pour un débutant. Une interface graphique évoluée permettant de "manipuler" l'action nommée -construire, tester, modifier, grouper, regrouper- favorise un taux de créativité plus élevé que dans un environnement traditionnel. De plus, la libération des contraintes syntaxiques associée à une automatisation partielle du concept de paramètrage sur des entités séquentialisables et immédiatement interprétable facilite le processus de validation. Les actions nommées favorisent donc l'attribution de noms sitôt que sont composées des entités autres que primitives; cette façon de faire est commune à toutes les méthodes aussi bien d'analyse que de développement qu'elles s'apparentent au raffinement ou à la modularisation, alors: pourquoi ne pas commencer tout de suite et avec des algorithmes élémentaires ?

Mots clés :

action, algocontexte, algorithme, algorithme paramètré, allocomposition, allodonnée, alloinstruction, alloschème, modélisation, table d'assemblage, validation.

Présentation d' ALLOGENE

ALLOGENE [FOU-92] est un environnement d'apprentissage de l'algorithmique destiné aux apprenants débutants. Conçu pour l'apprenant ainsi que pour l'enseignant, il s'articule autour de différents modules

pédagogiques -ayant un correspondant logiciel- au sein desquels une activité contrôlée est accomplie pour arriver à la résolution d'un **problème algorithmique**.
De manière brièvement décrite, les modules principaux peuvent s'énumérer selon: consulter pour la prise en charge de l'énoncé d'un problème algorithmique, reformuler pour aider à l'analyse du dit problème, modéliser pour éditer et formaliser l'algorithme et valider pour interpréter le modèle établi sur des jeux de données permettant de s'assurer de son bon fonctionnement. De plus, deux modules essentiels pour être en accord avec la démarche didactique choisie sont encore: simuler qui permet d'utiliser des **équipements virtuels** répondant à des commandes et visualiser qui permet de représenter un objet algorithmique sous différents aspects.
Le paradigme retenu -relatif au langage de modélisation- est celui de l'impérativité et les données sont regroupées dans ce qui est appelé un contexte contenant les objets du modèle algorithmique; ces derniers possédants au sein d'une collection d'attributs un particulier qui caractérise son accessibilité.

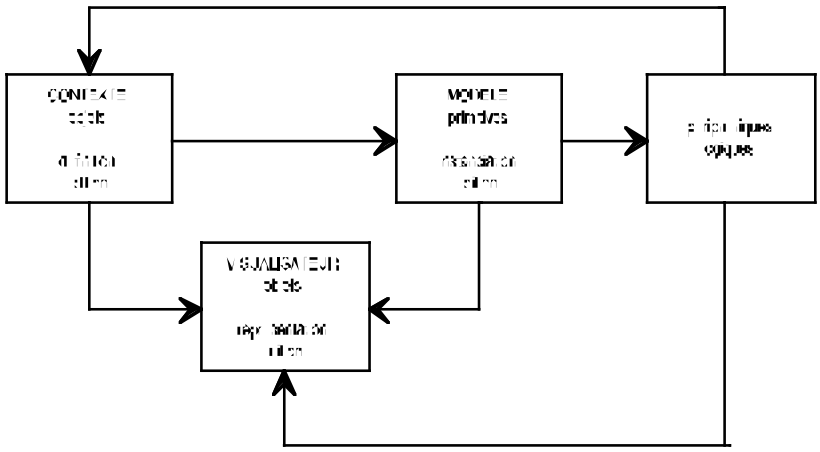

Figure 1 - L'environnement ALLOGENE

Afin de restreindre notre champ d'investigation, focalisons notre attention sur trois composants essentiels à la modélisation par actions nommées; ainsi, la contextualisation, la modélisation et la visualisation forment une triade que nous abrégerons par CMV. Chaque composant de CMV est accessible de façon permanente au moyen de la souris qui offre des possibilités de commandes/sélections au sein de la fenêtre correspondante.

La pratique de l'environnement demande à l'utilisateur de s'identifier pour débuter sa session de travail; ainsi, à chaque entité est attaché une **étiquette** ou **estampille**, qu'il s'agisse d'un objet ou d'une primitive, par le nom du propriétaire et un identificateur unique automatiquement généré par ALLOGENE.

Les difficultés de la modélisation

Les difficultés qu'éprouve un apprenant mis en situation de modélisation sont de plusieurs ordres; elles appartiennent aux domaines cognitifs que recouvrent la connaissance, la compréhension, l'application, l'analyse et la synthèse. Ces cinq domaines se retrouvent toujours dans l'acte de modélisation d'un système en général ou d'un problème algorithmique en particulier. Le résultat à obtenir est une structure formalisée dont la qualité se mesure par la capacité de représentation et l'adéquation par rapport au système réel. La modélisation algorithmique utilise plusieurs supports linguistique de syntaxe simple -purement mécanique- et aux lexèmes limités.
Une langue, un langage sont faits pour exprimer des faits et pour nous plus particulièrement pour reproduire, refléter. L'expression étant soumise à des règles, il faut en connaître les bases ainsi que les éléments grammaticaux qui vont avec. Mais, peut-être, n'est-ce pas la difficulté première d'une activité de modélisation

si l'on dispose de "patrons" et qu'un outillage d'assemblage nous est fourni. Par contre, il est toujours difficile de désigner, de nommer les entités du modèle formel qui est développé dans le but de leur donner une signification proche d'une réalité observable la moins subjective possible. C'est, paradoxe, cette difficulté qui nous a poussé à utiliser le paradigme des actions nommées comme support de modélisation algorithmique.

Présentation des entités de modélisation ALLOGÈNE

Les allodonnées

Ce sont les objets -au moins les constantes et les variables au sens classique d'un langage typé et procédural- qui seront accédés par le modèle algorithmique. Ces objets sont typés de manière traditionnelle par un attribut de nature; c'est parmi les natures que nous retrouverons les scalaires (caractère, "nombre", nombre entier, nombre relatif, ...) et les objets structurés (liste, table, dictionnaire, ensemble, ...) pour lesquels l'accès est réalisé au moyen d'une clé. Remarquons que l'objet constituant une donnée est toujours construit sur un patron et que des valeurs par défaut lui sont attribuées dès lors qu'il est créé, par exemple une valeur indéfinie sera comprise par l'interpréteur mais ne peut pas être admise pour toute instanciation demandant à être facteur d'une expression soumise à évaluation.

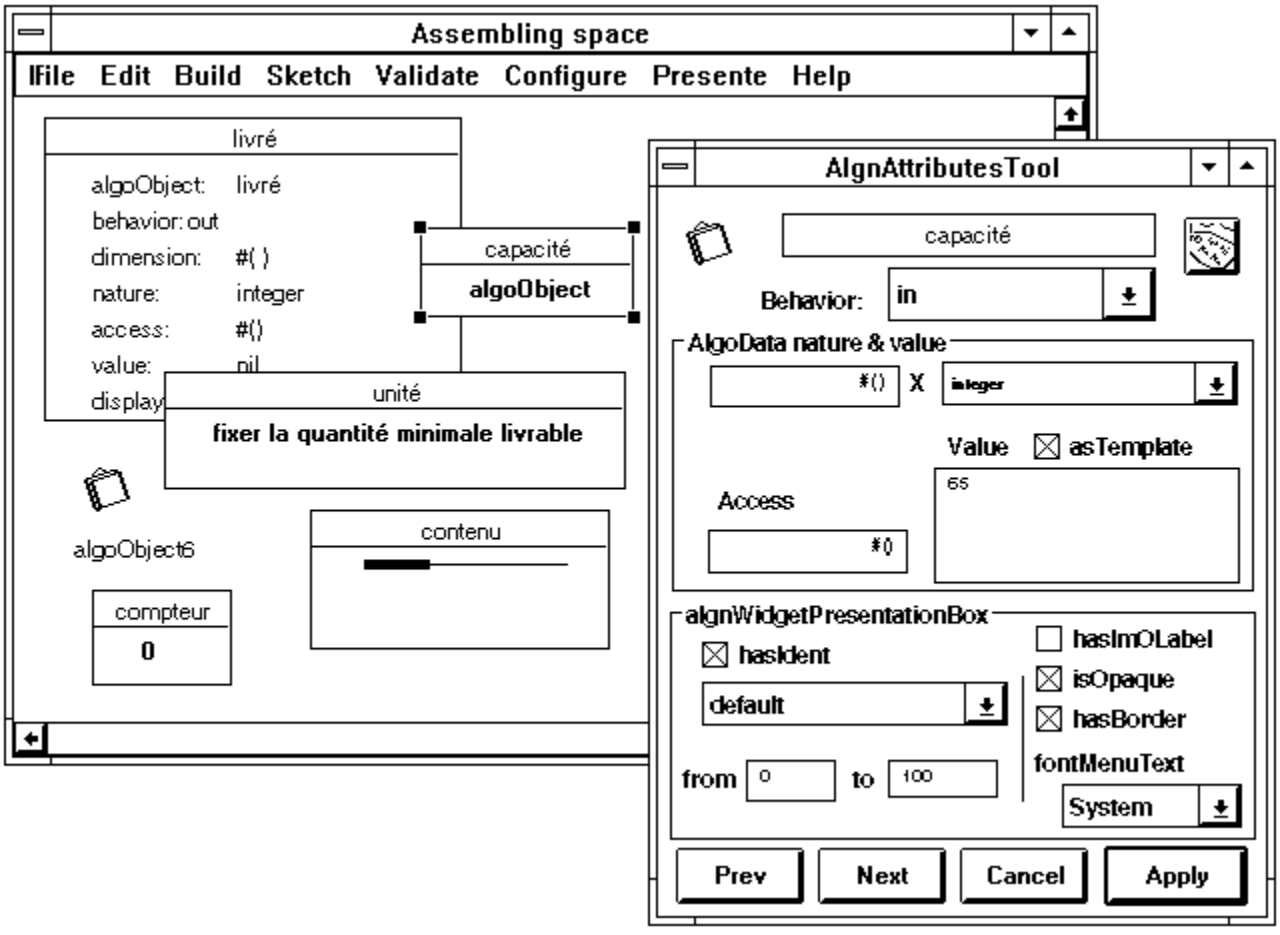

Figure 2 - Table de montage et attributs d'une sélection

La nature d'une allodonnée est un attribut d'une collection privée. D'autres attributs sont attachés à une allodonnée: son désignateur ou signifiant, sa valeur d'instance ou signifié dénotatif; de plus un rôle ou signifié connotatif est également proposé comme attribut prenant la forme d'un texte.
Ainsi, l'acquisition d'une allodonnée sur la table d'assemblage implique immédiatement l'existence de tout l'ensemble des attributs recevant des valeurs par défaut. Afin de donner un bref aperçu d'une situation, la figure 2 présente une table d'assemblage sur laquelle ont été déposées plusieurs allodonnées.

Sur cette figure 2, nous pouvons observer:

- la fenêtre table d'assemblage à proprement parler;
- la fenêtre de présentation et d'édition des attributs.

Les allodonnées sont toujours visualisables et sont capables de se présenter de différentes manières qu'il est possible de choisir -toujours à partir d'une présentation par défaut- et de changer à loisir par l'intermédiaire de la fenêtre des attributs. Sur la figure 2, l'objet visible tout à droite de la table d'assemblage est l'allodonnée sélectionnée et les attributs qu'elle possède sont affichés dans la fenêtre de présentation et d'édition; c'est donc la sélection qui détermine les attributs affichés par le choix de l'allodonnée.

Les différentes façons de présenter les allodonnées -icône, texte, rôle, graphique- et l'existence d'une unique fenêtre d'attributs ne permet pas toujours d'avoir une vue synthétique des allodonnées d'autant plus qu'il peut y avoir plusieurs représentations d'un même objet. Il est pourtant indispensable d'avoir cette vision "globale" des allodonnées et c'est pour cette raison que nous offrons une fenêtre supplémentaire que nous appelons contexte et qui regroupe tous les objets de la table d'assemblage; c'est-à-dire le contexte des données de la modélisation. La figure 3 présente cette fenêtre pour la situation déjà présentée en figure 2. Nous pouvons remarquer que l'allodonnée peut également être éditée par la voie de la fenêtre du contexte.
Le contexte est un concept important pour ALLOGÈNE qui met en oeuvre un paradigme de modélisation par actions, nous le développerons plus loin et en particulier vis-à-vis de la modélisation par actions nommées.

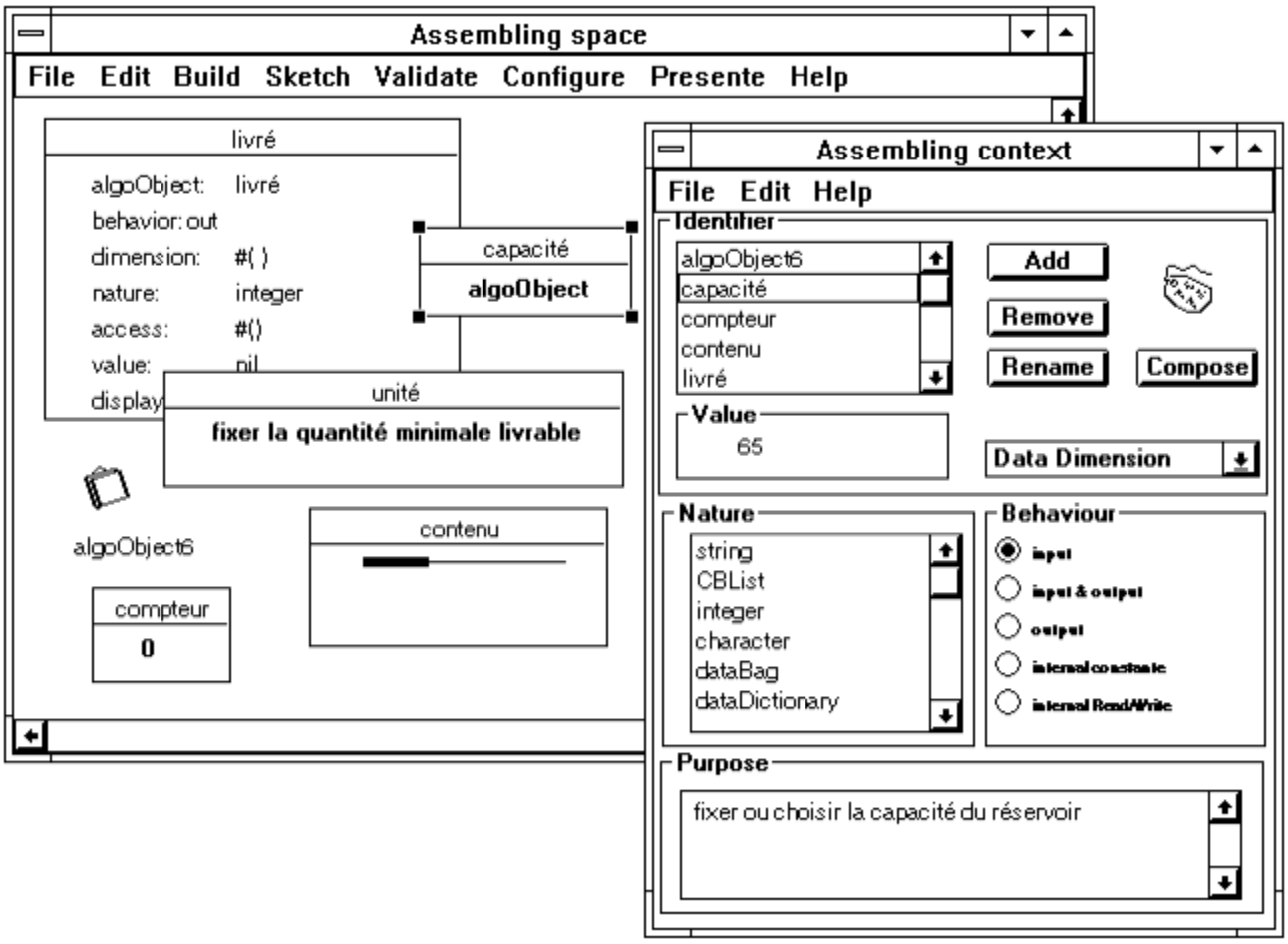

Figure 3 - Le contexte d'assemblage associé à une table de montage

Les alloinstructions

Ce sont les instructions primitives ou actions de base qui permettront d'élaborer le modèle algorithmique. Regroupées au sein de quatre catégories, elles peuvent être énumérées selon: l'affectation, les primitives conditionnelles, les primitives d'itérations et les auxiliaires ou commodités (délai à l'exécution pour la visualisation dynamique, stop ou arrêt pour disposer de points d'observation).
Les alloinstructions sont accessibles également via la table d'assemblage. Afin de donner un bref aperçu d'une situation, la figure ci-dessous présente une table d'assemblage sur laquelle ont été déposées plusieurs alloinstructions.

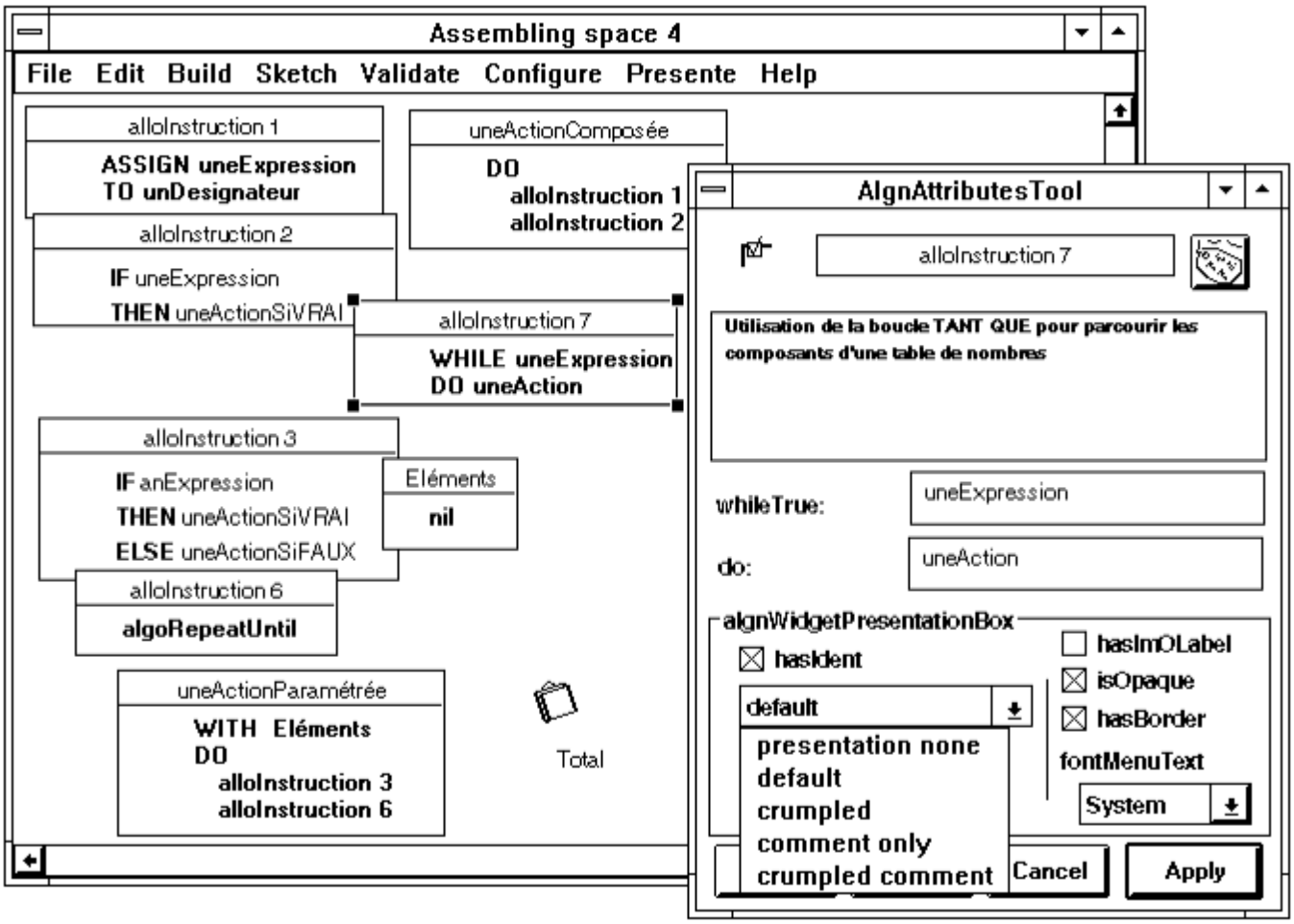

Figure 4 - Table d'assemblage et alloinstructions

Comme pour les allodonnées, la fenêtre des attributs est également présente; en effet, elle est capable de s'adapter à la spécificité de l'entité sélectionnée. C'est donc ici que l'on retrouve les attributs de l'alloinstruction qui sont exhaustivement: un identificateur d'alloinstruction, une forme littérale pour décrire une constante, une expression ou une fonction et un ou plusieurs champs identifiant d'action, donc présent en nombre variable et en adéquation avec la catégorie et l'élément choisi dans cette dernière (si ... alors ... , si ... alors ... sinon ... , etc.).
Les alloinstructions sont toutes, et immédiatement, des actions nommées par défaut lorsqu'elles sont déposées sur la table d'assemblage.

Avant de développer les action nommées, nous constaterons simplement, en ce point, qu'elles sont toutes interprétables et donc capables d'agir sur l'attribut d'instance ou valeur d'une allodonnée. Il est possible d'interpréter une alloinstruction, une sélection d'allointsructions, une séquence ou encore toutes celles présentes sur la table d'assemblage; cette interprétation est faite sur ce que nous appellerons le plan d'exécution qui est élaboré par des manipulations de sélection au moyen d'un périphérique de pointage.
Pour terminer, vous avez sans doute remarqué que la fenêtre d'édition des attributs pour les alloinstructions est la même que celle présentée pour les allodonnées; cette interface est donc partagée pour les deux entités décrites.

Les actions nommées et la composabilité

Nous avons vu auparavant que les deux catégories d'entités fondamentales -les allodonnées et les alloinstructions- sont nommées par défaut par le logiciel de l'environnement d'apprentissage ALLOGENE.
Cette dénomination est toutefois triviale, elle n'est en fait qu'un numéro d'ordre dont l'unicité est assurée; elle ne possède donc aucune sémantique rapportée à la modélisation en cours. Cette seule observation suffit à montrer la nécessité d'une possibilité de renommer de l'action. La nomination d'une action a donc deux aspects: facultatif s'il s'agit de clarifier ou impératif s'il s'agit d'invoquer cette action par le support d'une -ou plusieurs- alloinstruction. Pour la clarté de l'exposé, nous allons distinguer quatre types d'actions nommées:

- l'action nommée par défaut;
- l'action nommée par l'apprenant;
- l'action nommée composée signifiante ou sibylline;
- l'action nommée composée signifiée.

Remarquons tout de suite qu'une action nommée par l'apprenant n'est rien d'autre et obligatoirement une action nommée par défaut qui a été renommée par l'utilisateur. L'action nommée composée signifiée se différencie de l'action nommée composée sibylline par le fait qu'elle contient des allodonnées et que celles-ci sont désormais liées à cette action nommée; nous obtenons ainsi la possibilité d'introduire et de manipuler le concept de procédure d'une façon cohérente avec la philosophie de l'assemblage qui est la base de la méthodologie de la pratique que nous désirons instaurer.

La composabilité est une notion importante du paradigme de la modélisation par actions. Cette notion est par ailleurs suffisante si l'élaboration du modèle conduit toujours à un plan d'exécution séquentielle basé sur la forme linéaire d'écriture des alloinstructions et l'arborescence qu'elle sous-tend est donnée ci-après:

ACTION:
- Elémentaire
  - -affectation
- Composée
  - -en séquence
  - -en une conditionnelle
  - -en une itérative

Table 1 - Les actions de base

Les éléments de base sont donc particulièrement peu nombreux et simples. Toutefois, cette simplicité est limitative: l'analyse descendante est pratiquement exclue due à l'absence de l'action paramètrée et, par voie de conséquence, l'impossibilité de définir et d'inclure un contexte local. De plus, le raisonnement par actions, par rapport à un raisonnement par "instructions", est imposé. Si cette contrainte est garante de cohérence, elle nous a paru trop élémentaire et trop restrictive face aux approches de modélisation qui sont mises en pratique lors d'une réalisation d'un modèle algorithmique. Avant d'en expliciter l'usage, nous donnons ci-après une représentation arborescente de l'action telle qu'elle est implémentée par ALLOGENE.

ACTION:
- Elémentaire simple
  - -affectation
  - -FAIRE <un nom>
  - -FAIRE <un nom> AVEC <des paramètres>
- Elémentaire structurée
  - -une conditionnelle
  - -une itérative
- Composée
  - -en séquence
  - *-en une conditionnelle*
  - *-en une itérative*

Table 2 - Les actions ALLOGENE

Comparativement à la table 1, nous constatons une certaine complexité ajoutée qui n'a d'avantage que l'introduction d'une approche double pour la structuration lors de l'élaboration du modèle. Les deux actions "FAIRE" sont introduites au niveau de l'action élémentaire simple en raison de l'abstraction qu'elles introduisent au sein d'un espace d'assemblage.
Les composées conditionnelles et itératives ont été portées en italique sur l'arborescence de la table 2 ci-dessus afin de mettre en évidence qu'elles sont obtenues par une opération de sélection-composabilité. Remarquons qu'une fois établie de cette façon, l'action composée ne se distingue pas de l'action élémentaire structurée lui correspondant. De plus, la composabilité est applicable à toute action, c'est-à-dire à toute feuille de l'arborescence et ce dans un processus récurrent.

Nous donnons ci-après quelques exemples d'actions nommées.

	ACTION	EXEMPLE	Nommée par
1	affectation	contenu + 1 é contenu	augmenter
2	FAIRE		diminuer
3	affectation	contenu - 1 é contenu	diminuer
4	struct. conditionnelle	SI contenu > 0 ALORS diminuer	
5	struct. itérative	TANTQUE non plein FAIRE augmenter	remplir
6	affectation	80 é capacité	initialiser capacité
7	affectation	contenu = capacité é plein	
8	affectation	correction é x	
9	séquence	initialiser la capacité remplir	faire le plein
10	conditionnelle	SI capacité-contenu>5 ALORS faire le plein	
11	FAIRE	faire le plein	

Table 3 - Exemples d'actions

Cette table appelle quelques commentaires. Remarquons tout d'abord qu'une case vide de nomination signifie que l'action n'a pas été nommée par l'apprenant; c'est donc ce que nous avons appelé une action nommée par défaut. Des allodonnées -contenu, plein, capacité, x- et des alloinstructions -affectation, structurée conditionnelle et structurée itérative- sont présentes dans la table qui contient également deux actions composées. La colonne exemple -qui présente une forme syntaxique d'un langage- ne révèle pas de différence entre une alloinstruction structurée et la composée qui lui correspond; par contre, la démarche de modélisation a été chronologiquement différente. C'est donc dans ce sens chronologique que les lignes 2 et 3 sont relatées; en effet, il est tout à fait possible de disposer de l'action FAIRE nommée par diminuer bien que sa séquence soit primitivement vide. La ligne 3, établie ultérieurement, défini alors l'action auparavant vide qui résultait d'une abstraction constructive de la modélisation.

Afin de concentrer notre attention sur l'action et sa composabilité, nous avons jusqu'à présent omis de préciser un point fondamental relatif aux entrées et aux sorties. En fait, ALLOGENE gérera celles-ci au travers du concept de l'équipement virtuel qui peut être un terminal, un traceur ou encore tout autre appareil simulé commandé par des actions paramètrées primitives -lire() et écrire() pour le terminal par exemple- spécifiquement dédiées à cet appareil.

Rappelons que les allodonnées sont regroupées au sein d'un contexte dont les désignateurs constituent le lexique des données; des cas d'homonymie entre un lexème du contexte et un nom d'action sont supportés; mais pas souhaitables sans nul doute.

En supposant que la table 3 précédente est une synthèse au terme d'une activité d'un apprenant, sa table d'assemblage peut avoir l'allure présentée par la figure 5.

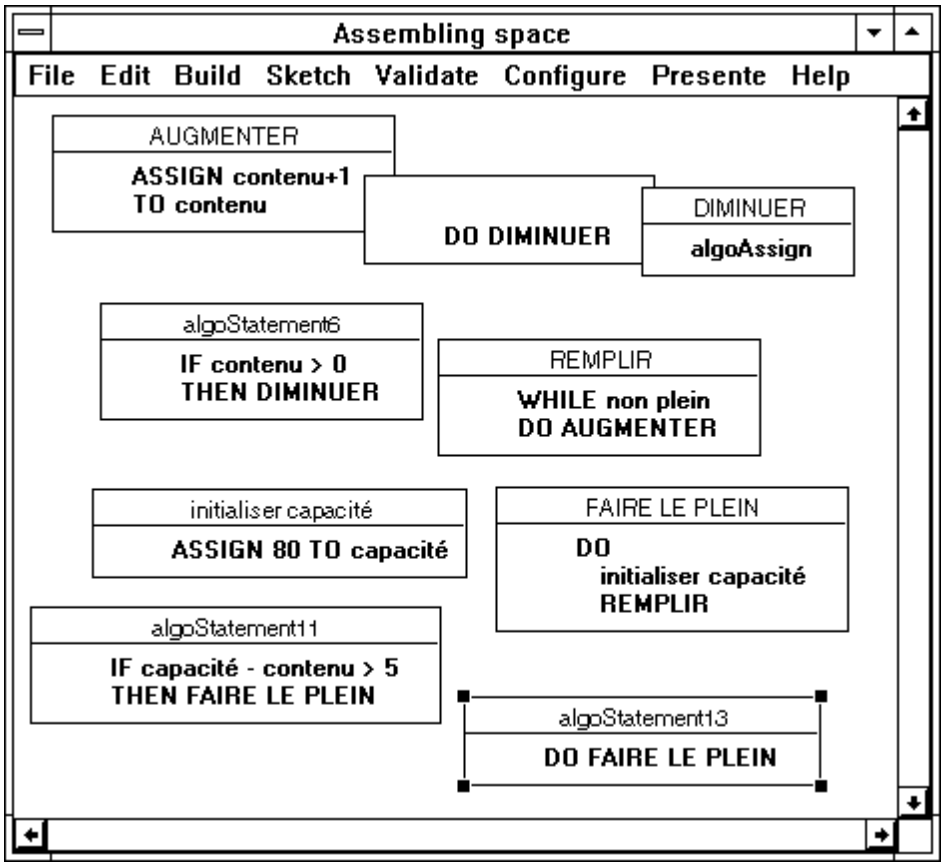

Figure 5 - Des actions sur une table de montage

Une première modélisation

Nous prendrons un exemple fort élémentaire qui ne met en jeu que quelques entités de modélisation ALLOGENE tant pour les allodonnées que pour les allointsructions. Considérons maintenant l'énoncé adapté suivant:

> Procéder au remplissage d'un réservoir dont on connaît la capacité et qui possède un contenu. Au terme du remplissage, nous devons posséder la quantité délivrée; pour effectuer une facturation par exemple. La livraison du liquide se fait par unités discrétisées et constitue ainsi, en quelque sorte, une résolution minimale.

Notre objectif n'est pas d'en présenter ici l'analyse de reformulation de ce problème ni d'expliciter différentes contraintes qui pourraient y être attachées.

Les allodonnées retenues par l'apprenant ont été celles-ci :

désignateur	**rôle**
unité	fixer la quantité minimale livrable
capacité	fixer la capacité du réservoir
contenu	mesurer le contenu du réservoir
compteur	compter les unités pour une livraison
livré	calculer ce qui a été livré

Le modèle algorithmique a été ébauché ainsi:

choisir une capacité	(1)
choisir un contenu	(2)
préparer le comptage des unités livrées	(3)
procéder au remplissage	(4)
calculer ce qui a été livré	(5)

Le modèle jusqu'à présent établi est un modèle d'intention. A priori, il ne comporte aucune alloinstruction et a fortiori aucune composition. Nous appellerons cette forme de modélisation la prime intention et sur l'espace d'assemblage, elle est représentée de la façon illustrée par la figure 6.

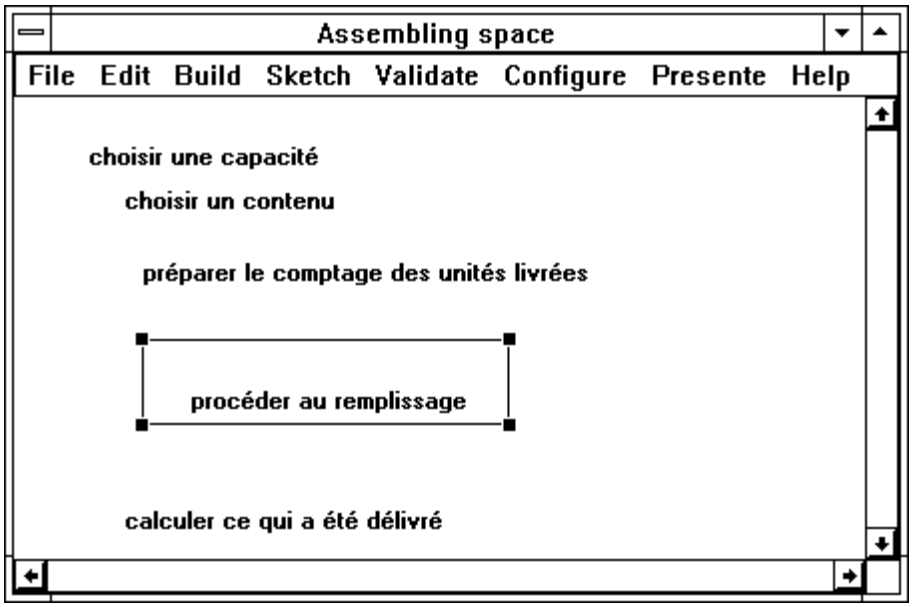

Figure 6 - Une modélisation de prime intention

Une première modification consiste a transformer l'intention énoncée en (4) par une action itérative. Ce choix conduit a utiliser le texte de l'intention (4) comme attribut de commentaire pour le TANTQUE qui a été choisi et qui résulte d'une conversion demandée par l'utilisateur.
La présentation de défaut permet d'afficher les champs qui ont été complétés selon:

choisir une capacité	(1)
choisir un contenu	(2)
préparer le comptage des unités livrées	(3)
|TANTQUE la place est suffisante pour une unité de livraison	(4)
|FAIRE ajouter au contenu et comptabiliser l'unité	
calculer ce qui a été livré	(5)

Cette approche a conduit l'apprenant à réaliser l'ébauche de modélisation sur la table d'assemblage en utilisant cinq actions nommées par défaut telle que présentées sur la copie de la table d'assemblage en figure 7:

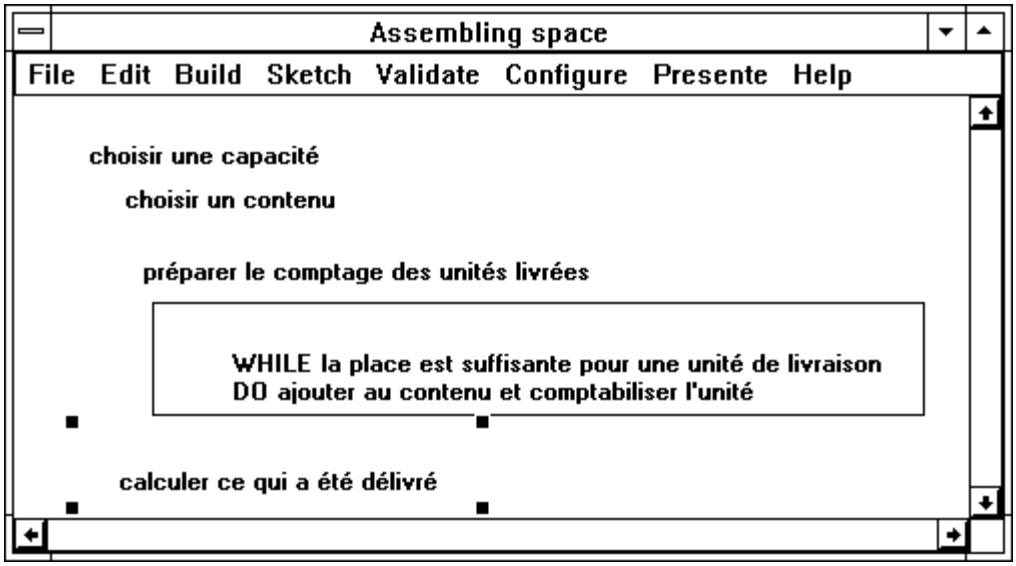

Figure 7 - Choix d'une action sur un modèle de prime intention

C'est ce que nous appelons un commentogramme; celui-ci décrit - avec plus ou moins d'acuité suivant l'application- le rôle ou l'objectif des actions qui ont été choisies. La table d'assemblage montre également les allodonnées.

A ce stade de développement, une exécution du modèle -bien qu'elle puisse être demandée- n'a pas d'effet aucune action nommée par défaut n'étant établie au niveau de l'instruction; c'est-à-dire dans une forme syntaxique reconnue comme interprétable utilisant les couples <information;valeur> du contexte algorithmique ou algocontexte.
Nous présentons ci-après un graphe des activités conduisant à la réalisation complète du modèle qui doit être validé par des exécutions pour différents contextes choisis.

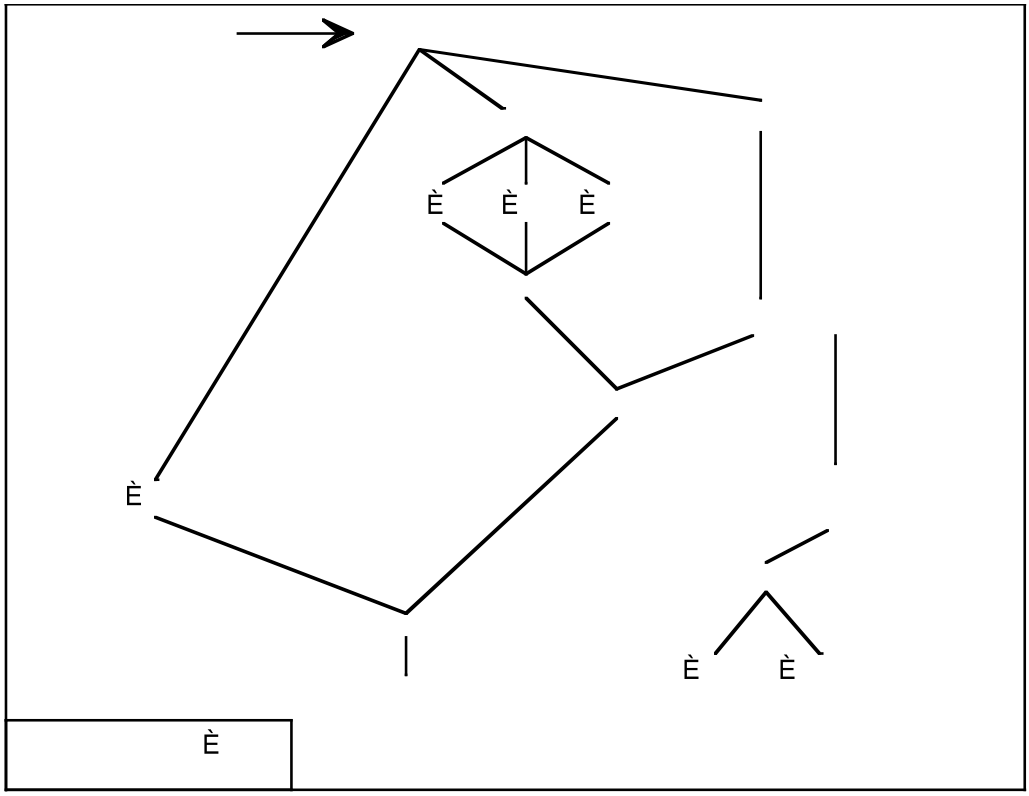

Figure 8 - Composition d'actions

Cette figure 8 présente une démarche possible parmi d'autres. Certaines opérations de composition sont superfétatoires; en particulier, l'action nommée REMPLIR qui résulte d'une composition d'une séquence et de l'alloinstruction (5) aurait également pu être obtenu par une composition séquentielle multiple à partir de la sélection des entités (1) à (5). En ce qui concerne l'action (4) nommée par "procéder au remplissage", nous remarquons qu'elle a été ensuite obtenue par une conversion, que son champ d'appel d'action a reçu le texte "ajouter au contenu et comptabiliser l'unité" et que cette action a été développée ensuite seulement à l'aide de deux affectations (4.1) et (4.2) composées en une séquence.

La figure 9a présente le travail terminé tel qu'il est obtenu sur la table au terme des compositions d'assemblage.

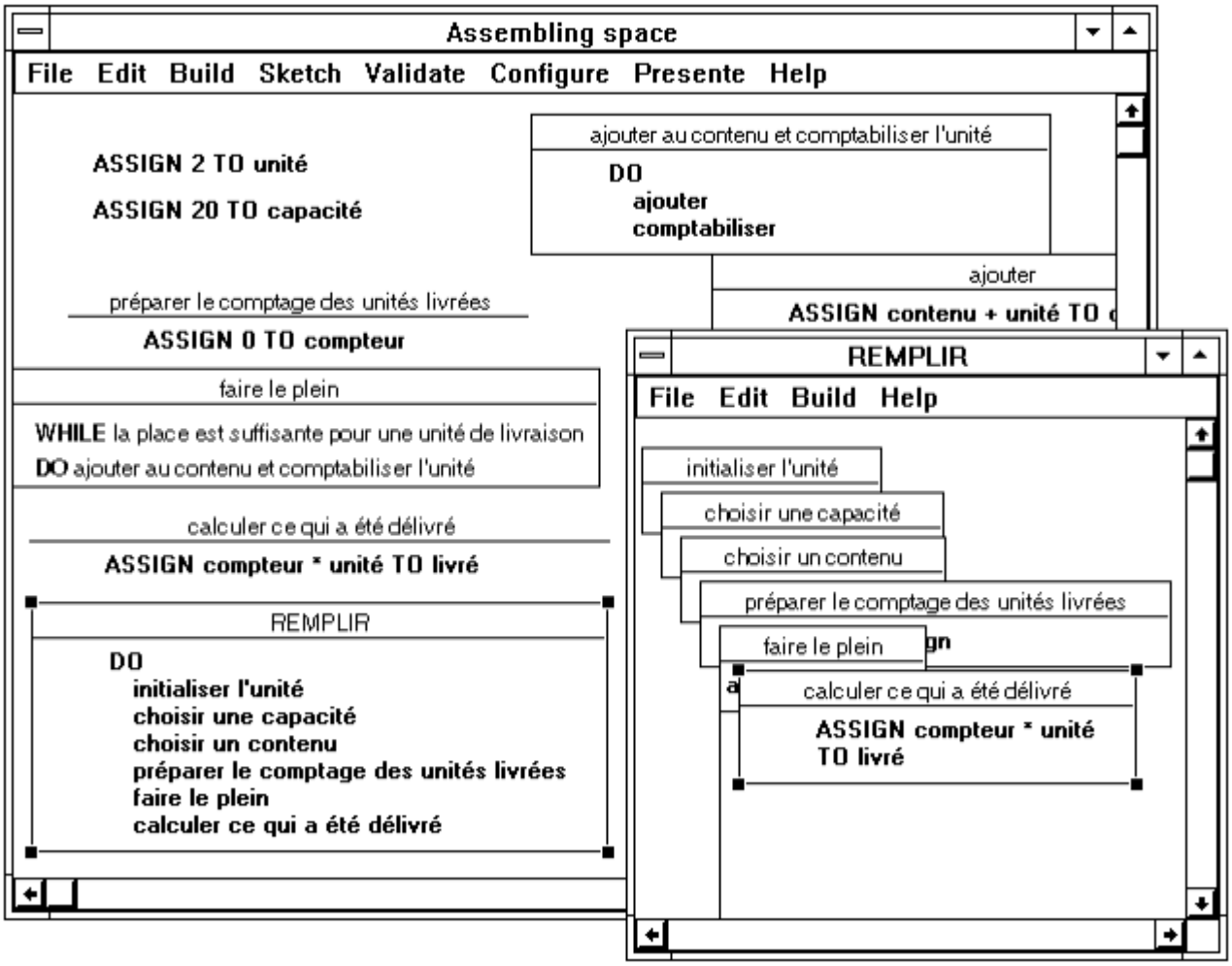

Figure 9a - Modélisation et zoom sur une action nommée

Des actions à l'algorithme paramètré

Les actions nommées précomposées ou composées constituent, une fois validées, ce que nous appellerons une allocomposition qui décrit un algorithme simple. En ce qui concerne la réutilisabilité assimilée à une bibliothèque, la gestion des allocompositions est faite selon un procédé comparable à une gestion de fichier; ainsi le rappel d'un algorithme simple est possible en tout temps et permet donc d'apporter des modifications ou compléments sur la base du travail mémorisé. Cette possibilité est insuffisante pour résoudre des problèmes algorithmiques plus étoffés et ceci, pour plusieurs raisons. En effet, à partir d'un certain niveau, le raisonnement de modélisation ayant comme seul support les instructions précomposées, la composition -ou encore un mélange des deux- doit pouvoir se répercuter à différents niveaux d'abstraction lors de la conception de programme.

Dans une optique d'analyse de typologie descendante les besoins élémentaires sont au moins les algorithmes paramètrés -terme préféré à action paramètrée par la connotation de finalité qui lui est associée- et l'exploitation d'un algocontexte local à l'entité paramètrée.

L'algocontexte d'une allocomposition regroupe exhaustivement les allodonnées. Auparavant nous n'avons pas développé un attribut qui va maintenant jouer un rôle fondamental dans le processus de paramétrage. Cet attribut est celui du flux auquel toute allodonnée est impérativement soumise. L'attribut est choisi au sein d'une liste de cinq comportements réunis dans deux groupes:

Interne
- lecture
- lecture/écriture

Interfaçable
- entrée
- entrée/sortie
- sortie

Tous les attributs internes constituent et établissent un algocontexte local à l'algorithme paramètré dont la portée lexicale est limitée à l'espace implicitement associé avec le désignateur de cet algorithme.
Un algorithme paramètré est "installable" dans l'environnement privé -mais aussi exportable- de l'apprenant. Une fois cette installation faite, l'algorithme paramètré devient invocable au même titre que toute action nommée; cette explication justifie maintenant le choix comme action élémentaire simple tel que nous l'avons présenté au paragraphe "Les actions nommées et la composabilité". Ainsi, nous ne voulons pas distinguer des actions primitives paramètrées -celles associées à un équipement virtuel par exemple- de celles construites par l'apprenant.

Ainsi, l'action FAIRE <un nom> AVEC <des paramètres> est toujours utilisable comme une modélisation algorithmique désignée et répertoriée. Le paramètrage proprement dit s'effectue également au sein d'un espace de modélisation et à l'aide d'une interface automatique prenant les allodonnées des groupes interne et interfaçable.

Un premier algorithme paramètré

Nous reprenons l'exemple développé précédemment et désigné par REMPLIR.
Les allodonnées retenues par l'apprenant sont ici reportées avec leur attribut de flux:

désignateur	**attribut de flux**	**rôle**
unité	interne - lecture	fixer la quantité minimale livrable
capacité	interfaçable - entrée	fixer la capacité du réservoir
contenu	interfaçable - entrée/sortie	mesurer le contenu du réservoir
compteur	interne - lecture/écriture	compter les unités pour une livraison
livré	interfaçable - sortie	calculer ce qui a été livré

Cet algorithme peut donc être invoqué par:

FAIRE REMPLIR **AVEC**	capacité	60
	contenu	disponible
	livré	valeur

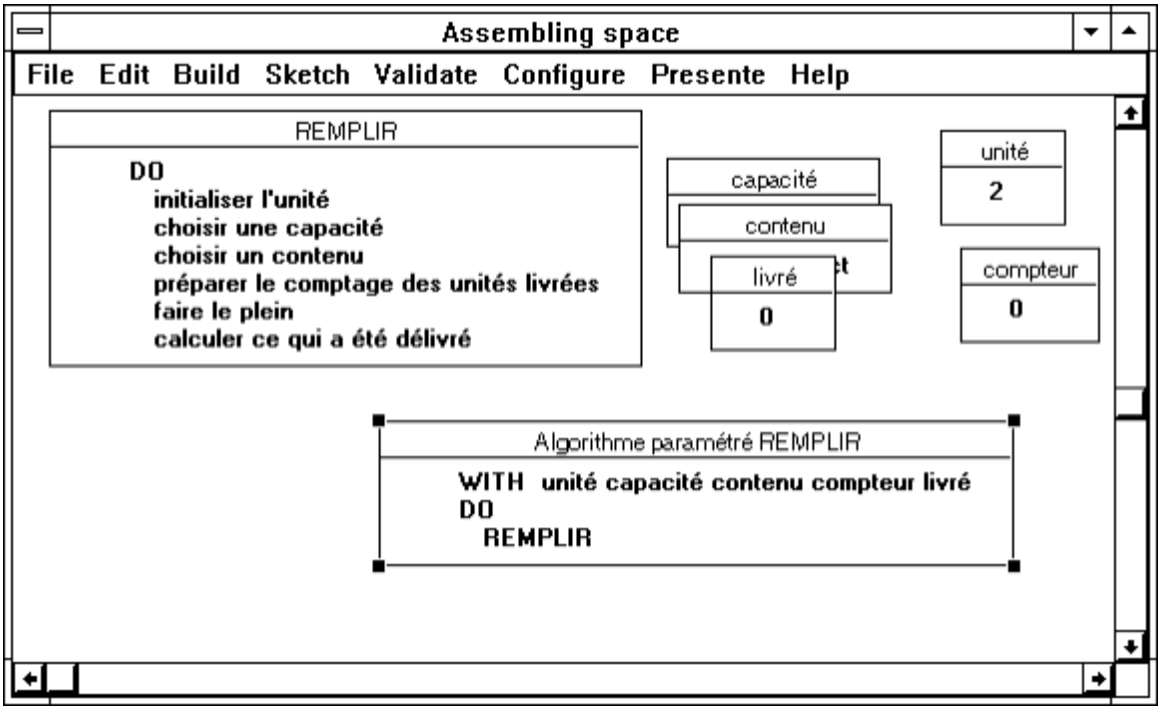

Figure 9b - Un action nommée devient un algorithme paramètré

Les allodonnées disponible et valeur sont bien sûr déclarées dans l'algocontexte de l'espace de modélisation qui procède à l'appel de l'algorithme paramètré REMPLIR. Remarquons que tous les paramètres interfaçables ont été utilisés dans l'appel ci-dessus mais que cela n'est pas impératif.

La compilation de l'algorithme paramètré étant faite au sein de l'espace de modélisation-validation courant, un autre modélisation peut faire appel à REMPLIR avec une autre formalisation des paramètres. De ce fait, les effets de bord involontaires sont donc exclus puisque les allodonnées sont toujours référencées par rapport à un contexte: l'algocontexte de l'algorithme.

Les unités d'exécution

L'espace d'assemblage réunit des allodonnées et des actions. Les actions sont exécutables ou interprétables au niveau de l'entité de modélisation déjà puis au niveau de l'action composée par la suite. Cette possibilité de validation incrémentale est toutefois insuffisante d'une part en considérant l'algorithme paramètré et d'autre part sur le plan des variantes possibles de formalisation algorithmique. Par exemple, le tri à bulles possède de nombreuses variantes intéressantes et une forme optimale de modélisation n'est probablement pas obtenue immédiatement.
A cet effet, un espace de modélisation peut contenir une ou plusieurs unités d'exécution à titre d'illustration de variantes. Une unité d'exécution est aussi un plan d'exécution qui regroupe allodonnées et actions. Les allodonnées y sont intégrées et les attributs de flux de chacune pris en compte. La construction d'une unité d'exécution se fait par une opération de sélection-groupage qui produit celle-ci et lui attribue un désignateur. Ce désignateur peut ensuite être utilisé comme désignateur d'algorithme paramètré dans le cadre d'une autre modélisation; en quelque sorte, il s'agit d'une procédure précompilée et dûment paramètrée.

Les textes ou listages d'une modélisation

Tel que nous l'avons présentée, la modélisation n'est concrétisée que par le contenu de la table d'assemblage. Bien que nous ne poursuivions pas le but d'associer un langage de modélisation à notre environnement, nous

sommes persuadés qu'un texte est indispensable pour mémoriser d'une manière plus formelle la construction de la modélisation. Le listage dont peut disposer l'apprenant est alors comparable au code d'un programme et présenté selon la méthode de beaucoup de langages impératifs: une section de déclarations pour les allodonnées et une section algorithmique pour les actions nommées dont la désignation est écartée et par conséquent les compositions également; les commentaires peuvent également être obtenus.

ALGORITHME REMPLIR

Allodonnées
  Internes
    -lecture: unité dans N
    -lecture/écriture: compteur dans N
  Interfaçables
    -entrée: capacité dans N
    -entrée/sortie: contenu dans N
    -sortie: livré dans N

Début de l'algorithme
  2 é unité
  20 é capacité
  13 é contenu
  0 é compteur
  TANTQUE capacité - contenu > unité FAIRE
    contenu + unité é contenu
    compteur + 1 é compteur
  FIN TANTQUE
  compteur * unité é livré
Fin de l'algorithme

Nous offrons également la possibilité d'obtenir un texte tel que ci-dessous et que nous appelons un alloschème; il donne l'état à l'instant final des actions après leur exécution pour un algocontexte déterminé.

Alloschème REMPLIR

1#a AlgoAffectation **2** à **unité** ; {unité} = {2}
1#b AlgoAffectation **20** à **capacité** ; {capacité} = {20}
1#c AlgoAffectation **13** à **contenu** ; {contenu} = {13}
1#d AlgoAffectation **0** à **compteur** ; {compteur} = {0}
1#e AlgoTantque **capacité - contenu** > **unité**
{capacité - contenu > unité} = {VRAI}
1 1#a AlgoAffectation **contenu + unité** à **contenu** ; {contenu, unité} = {15, 2}
1 1#b AlgoAffectation **compteur + 1** à **compteur** ; {compteur} = {1}
{capacité - contenu > unité} = {VRAI}
1 1#a AlgoAffectation **contenu + unité** à **contenu** ; {contenu, unité} = {17, 2}
1 1#b AlgoAffectation **compteur + 1** à **compteur** ; {compteur} = {2}
{capacité - contenu > unité} = {VRAI}
1 1#a AlgoAffectation **contenu + unité** à **contenu** ; {contenu, unité} = {19, 2}
1 1#b AlgoAffectation **compteur + 1** à **compteur** ; {compteur} = {3}
{capacité - contenu > unité} = {FAUX}
1#fAlgoAffectation **compteur * unité** à **livré** ; {compteur, unité, livré} = {3, 2, 6}

Dans une phase d'apprentissage de premières modélisations, l'apprenant ne perçoit pas toujours comment sont exécutées les actions et les instructions. Si la séquence d'affectations ne pose guère de difficultés, il n'en est pas de même des actions structurées présentant des conditions d'entrée ou de terminaison qu'il soit fait référence à des formes itératives et/ou des formes conditionnelles. Ainsi, lors d'une validation partielle ou complète, un commutateur permet d'obtenir le schéma de l'exécution ou alloschème. Celui-ci est le texte de toutes les instructions élémentaires exécutées pour accomplir l'algorithme avec le jeu des allodonnées retenues pour la validation. La numérotation des instructions est confiée à un séquenceur qui procède selon le principe de la numérotation des paragraphes et réalise l'indentation d'imbrication également.

Supporter les approches différenciées

Les méthodologies de modélisation sont nombreuses et il est courant d'en distinguer deux typologies pour le paradigme impératif que nous avons choisi: l'une dite ascendante et l'autre dite descendante. Avec ce que nous avons décrit jusqu'ici, il semble que l'environnement de modélisation que nous proposons s'instaure typiquement de la philosophie ascendante. C'est-à-dire que l'apprenant n'aurait pas d'autre possibilité que de réaliser complètement son modèle avant de procéder à une validation. Le modèle s'établissant en échafaudant des compositions sur des actions élémentaires simples ou structurées. Ce serait une limitation importante dont la rigidité contraignante n'est pas souhaitable en regard des différents modes de pensées d'une part et des tâches qui peuvent être remises à plus tard d'autre part; certaines actions du modèle pouvant être non opérationnelles dans une première phase tout en ayant une nécessité d'existence. Il est à noter encore qu'un développement n'est que rarement inscrit dans une typologie strictement ascendante ou descendante; nous avons d'ailleurs utilisé cette mixité dans la présentation de notre premier exemple de modélisation.

Ainsi, notre espace d'assemblage permet de disposer d'actions nommées dont l'édition (c'est-à-dire un processus de modélisation local) peut se faire en marge de la modélisation proprement dite. Cela implique bien sûr que cette édition locale informe de ses changements l'espace d'assemblage de l'algorithme.

Afin d'illustrer notre propos, prenons le listage partiel d'un programme.

Début de l'algorithme
.....
SI C1 ALORS
    <expr> é a
    <expr> é b
    TANTQUE C2 FAIRE
        <expr> é x
        <expr> é y
    FIN TANTQUE
    <expr> é c
FINSI
.....
Fin de l'algorithme

Nous donnons ci-après les deux schémas de modélisation dans une typologie choisie et aussi stricte que possible pour respecter continûment ce choix.

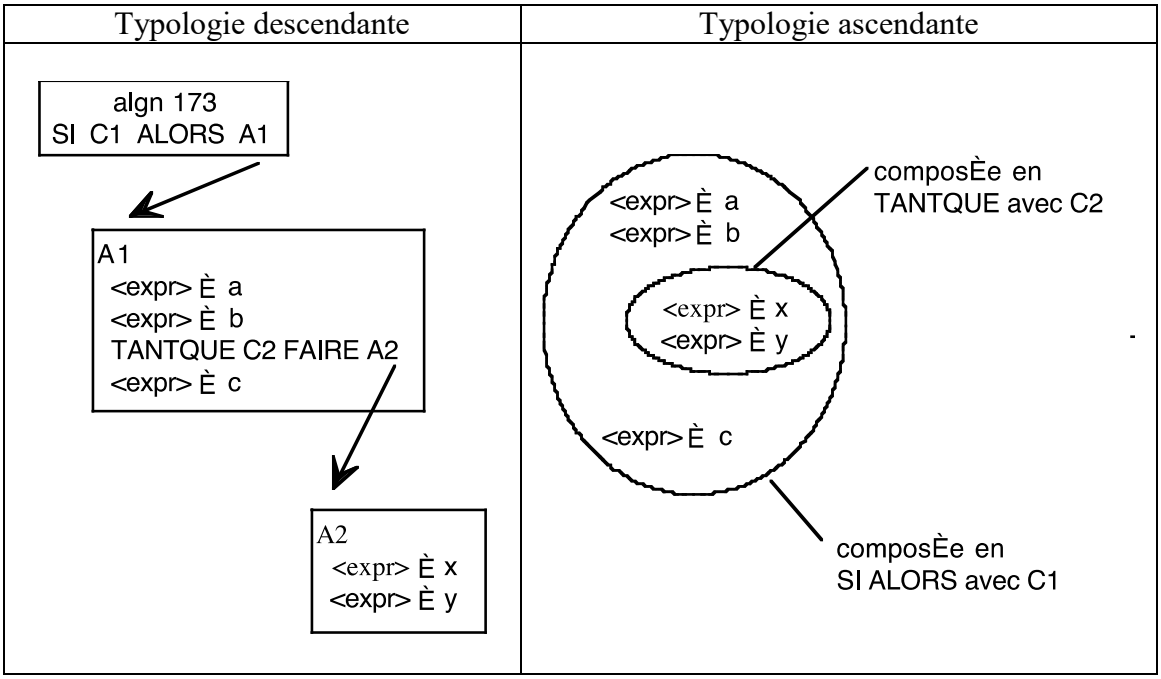

Figure 10 - Deux approches de la composition d'actions

D'après cette illustration, il est clair que la réalisation selon la typologie ascendante est plus économique quant au nombre des manipulations nécessaires sur la table d'assemblage. Mais cette économie ne peut être obtenue que si le raffinement a été suffisamment développé avant la phase de réalisation proprement dite; c'est d'ailleurs là une bonne suggestion.

Une modélisation complète: le tri à bulles

Le tri à bulles est un algorithme destiné à un débutant qui a déjà exercé l'affectation, le parcours des suites avec test des éléments contenus et qui sait procéder à l'échange réciproque des valeurs de deux variables. Le tri à bulles présente de multiples variantes et nous en présenterons une pour laquelle l'analyse est guidée par des indications plutôt que d'utiliser l'anecdote qu'il suffit d'observer une bouteille de soda débouchée pour découvrir cet algorithme. Tout d'abord, pour trier une suite bornée dont les composants sont accessibles par une clé qui est un élément pris dans un intervalle avec relation d'ordre, nous pouvons commencer par nous demander si il est utile de procéder au tri; c'est-à-dire de déterminer si la suite est déjà triée. Prenons une suite désignée par TABLE avec M éléments; la table est une suite triée si TABLE[j-1] <= TABLE [j] que nous pouvons réduire en particularisant selon TABLE[1] <= TABLE [j]; cette dernière écriture nous incite à considérer le premier élément de la suite comme un témoin. Ainsi, prouver qu'une table est triée, consiste à remettre en cause le prédicat relationnel au cours d'une itération et en utilisant le témoin.
La figure ci-après présente la table d'assemblage sur laquelle est réalisée cette modélisation; quatre allodonnées sont utilisées: la suite TABLE et sa taille M, le témoin témoin et une clé cle d'accès à la suite.

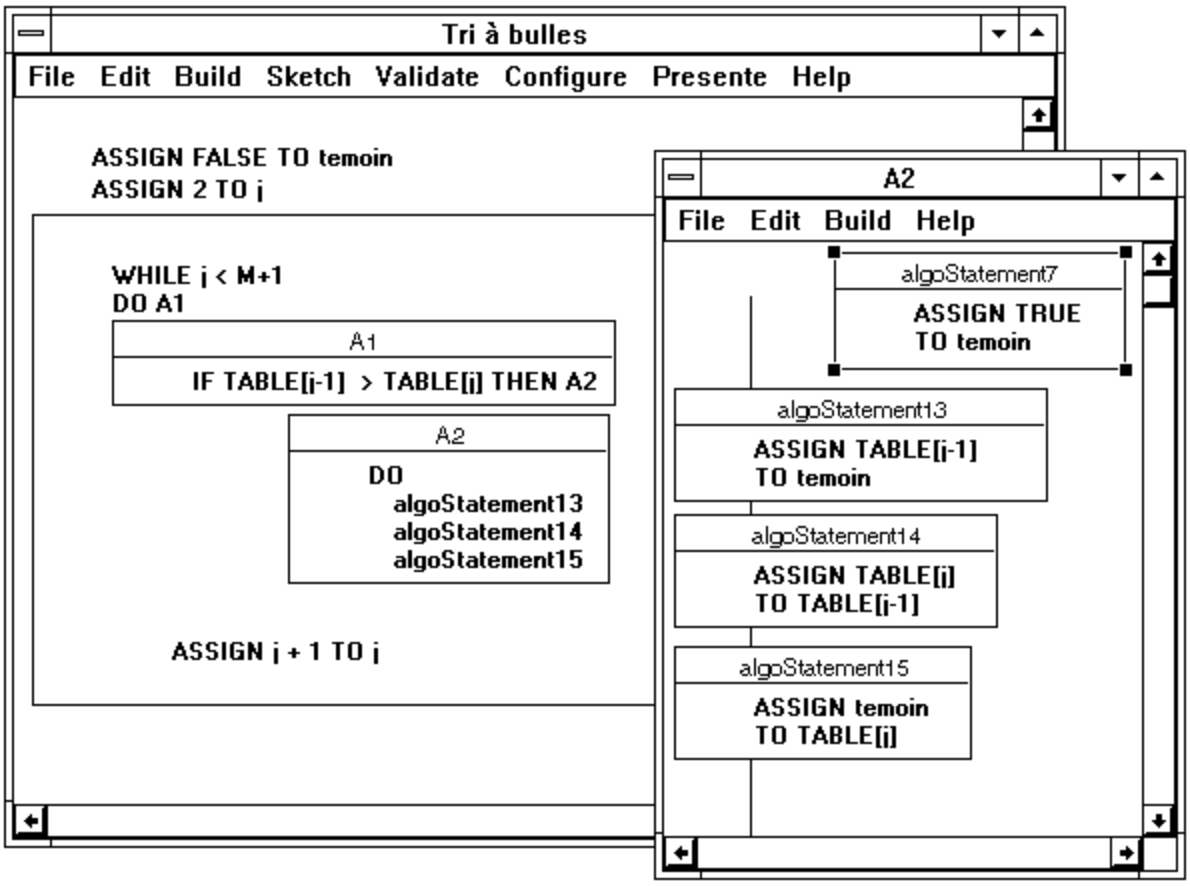

Figure 11 - Un tri à bulles en cours de modélisation

L'action A1 a pour but d'évaluer le prédicat et relève de la suspicion prédicative. L'action A2 a pour fonction de mémoriser un éventuel changement de l'instance du témoin qui est repris ensuite afin d'obtenir une manifestation du "résultat".
Un point qu'il n'est pas difficile de remarquer est la possibilité d'une remise en ordre partielle lorsque le prédicat relationnel contenant deux composants adjacents de la suite est évalué à faux; il suffit alors de procéder à une permutation de ces composants. L'activité de l'apprenant consiste alors à éditer l'action nommée A2. Remarquons que l'allodonnée témoin convient comme variable auxiliaire d'échange et que deux actions élémentaires simple d'affectation suffisent à compléter l'action A2 qui demeure une séquence. Comme autre procédure, il eut été également possible de composer les actions élémentaires simples en séquence et de conserver le nom de A2 comme identifiant d'action. Quelle que soit l'approche, il résulte pour A2 une séquence décrite par: { TABLE[j-1] é temoin ; TABLE[j] é TABLE[j-1] ; temoin é TABLE[j] }. A ce stade, il ne devrait pas être difficile de conclure l'algorithme; les étapes intermédiaires ont été validées et la TABLE étant visualisée pour observer les effets dynamiques de l'application de l'algorithme. C'est soit l'observation visuelle soit une réflexion théorique qui permet de découvrir que l'objectif du tri ne peut être atteint que si l'itération conduit à conserver le temoin instancié de sa valeur initiale. Autrement, il est nécessaire de procéder à une nouvelle itération et de répéter celle-ci jusqu'à que le témoin ne soit pas modifier. Il est alors temps d'utiliser la composabilité et de l'appliquer sur les deux alloinstructions - une séquence implicite - déjà déposées sur la table d'assemblage: { TABLE[1] é temoin ; TANTQUE j<M+1 FAIRE A1 } . La figure ci-dessous présente une modélisation en cours de validation avec une visualisation de l'allodonnée TABLE.

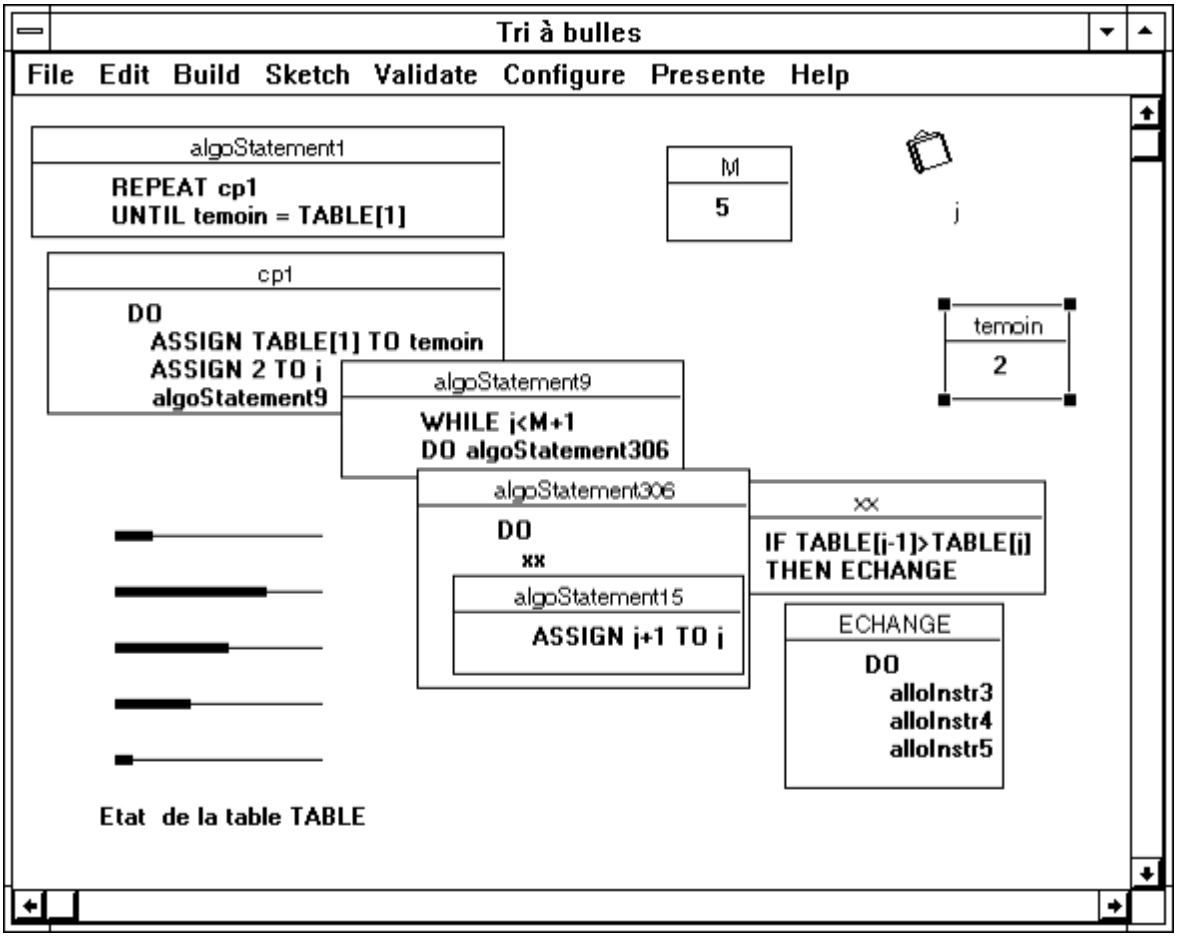

Figure 12 - Le tri à bulles en phase de validation

Le listage du programme obtenu pour cette modélisation est alors:

ALGORITHME TRI_A_BULLES

Allodonnées
Internes
-lecture: M dans N
-lecture/écriture: j, temoin dans N
Interfaçables
-entrée:
-entrée/sortie: TABLE dans N
-sortie:

```
Début de l'algorithme
    REPETER
        TABLE[1] é temoin
        2 é j
        TANTQUE j < M+1 FAIRE
            SI TABLE[j-1]>TABLE[j] ALORS
                TABLE[j-1] é temoin
                TABLE[j] é TABLE[j-1]
                temoin é TABLE[j]
            FINSI
            j+1 é j
        FIN POUR
    JUSQU'A temoin = TABLE[1]
Fin de l'algorithme
```

Récapitulatif des entités de modélisation

Nous présentons ci-après les entités de modélisation en faisant la liste de tous les attributs qui y sont attachés. Nous rappelons que toutes les entités reçoivent leurs attributs par défaut, certains attributs sont impératifs pour un modèle validable et d'autres ont un aspect plus documentaire.

Entité:	**allodonnée**	**alloinstruction**	**composition**
Attributs:			
-date de création	oui	oui	oui
-date de révision	oui	oui	oui
-propriétaire	oui	oui	oui
-droits d'accès	oui	non	non
-désignateur	oui	non	non
-nature de typage	oui	-	-
-valeur d'instance	oui	-	-
-comportement de flux	oui	-	-
-rôle	oui	non	non
-nomination	non	oui	oui
-commentaire	non	oui	oui
Autres paramètres:			
-choix de présentation	oui	oui	oui
-choix de visualisation	oui	non	non
-animation selon	automatique	signal d'exécution	signal d'exécution

Conclusion

L'environnement d'apprentissage ALLOGENE est donc une réalisation destinée à la modélisation algorithmique informatique pour un apprenant débutant. Le paradigme retenu est celui de modélisation par actions nommées au sens large puisque l'interface est conçue de telle sorte qu'un raisonnement par instructions plus traditionnel est aussi partiellement supporté. Les problèmes de syntaxe liés au langage sont pratiquement absents et devraient favoriser une progression plus rapide dans l'étude des modèles.

Un autre aspect de l'environnement est d'offrir la possibilité à l'enseignant de préparer des modèles. Ceux-ci peuvent être partiels donc destinés à être complétés ou également totalement validés afin qu'une exécution reprise permette d'observer la dynamique et le comportement des objets données en cours de tâche.

La philosophie choisie d'une visibilité totale des actions et des données dispense de la fastidieuse gestion des entrées et des sorties qui est souvent mal gérée par les débutants qui y consacrent trop de temps. Toutefois, l'équipement virtuel permet tout de même cette approche.

Mais, ce sont bien les actions nommées qui resteront au centre de l'activité de l'apprenant qui apprendra aussi à développer des modélisations plus complexes grâce à l'algorithme paramètré qui est naturellement introduit sur un ensemble d'actions réunies sur une table d'assemblage.

Glossaire

Note: suit une brève description des renforcements *in text* et néologismes

action
Entité de modification de la partie valeur d'une allodonnée résultant d'un événement propre à un exécutif.

algocontexte
Liste exhaustive des allodonnées.

algorithme
Résultat suffisant d'une modélisation.

algorithme paramètré
Une action désignée associée à des abstractions d'allodonnées caractérisées significativement par un comportement de flux; l'algorithme paramètré est réutilisable par d'autres algorithmes par invocation.

allocomposition
Opération ou résultat de celle-ci consistant à composer selon un thème des actions.

allodonnée
Objet pour un couple information-valeur.

alloinstruction
Action primitive construite toujours sur un même patron.

alloschème
Listage d'une exécution d'une modélisation signifiant les états terminaux des allodonnées à la suite de l'action sur celles-ci.

équipement virtuel
Simulation logique d'un matériel accessoire typique des périphériques informatiques.

modélisation
L'assemblage et l'ordonnancement de la suite finie des éléments d'une solution définissent cette activité; du point de vue "mécanique" celle-ci se pratique à l'aide de la métaphore d'une table d'assemblage.

modèles solution
La juxtaposition des deux termes modèle et solution signifie que le modèle, l'algorithme modélisé, est solution du problème algorithmique. Dans ce contexte solution agit comme qualificatif substantivé à modèle.

problème algorithmique
Pour l'environnement d'apprentissage ALLOGENE, les problèmes algorithmiques sont des énoncés de questions à propos desquels les apprenants devront trouver la méthode que devra suivre l'automate, cette méthode n'étant rien d'autre que l'algorithme lui même. Il est à noter qu'ALLOGENE impose, tout du moins suggère très fortement, une démarche analytique caractéristique pour résoudre les problèmes algorithmiques.

solution
Une solution viable est une suite finie de décisions et d'actes pouvant résoudre une difficulté: le problème algorithmique.

table d'assemblage
Un espace de travail adapté à l'activité de modélisation recevant les allodonnées et les actions; les commandes d'édition et l'appel à l'exécutif sont réalisées depuis cette table d'assemblage.

validation
Processus informel de vérification d'adéquation de l'algorithme pour obtenir un modèle solution.

visualisation
Ensemble des représentations visuelles des allodonnées.

Bibliographie

[CHA-92] Programmation - Cours et exercices.
Guy Chaty et Jean Vicard ; Ellipses - 1992 ; ISBN 2-7298-9288-5

[FOU-92] ALLOGÈNE: Un environnement d'apprentissage de l'algorithmique
Jean-Pierre Fournier et Jacky Wirz
in AFDI - Actes 3e rencontre francophone de didactique de l'informatique p. 101 à 113 - 1992

[MIL-92] Tuteurs informatiques pour l'apprentissage de la programmation
E. Milgrom et Monique De Wolf
in TSI Vol. 11 - N°6 p. 9 à 37 - 1992

[SCH-93] Cours d'informatique: langages et programmation
P.-C. Scholl, M.-C. Fauvet, F. Lagnier et F. Maraninchi
Masson - 1993 ; ISBN 2-225-84289-2

[VAS-89] La programmation naturelle
Jean-Pierre Vasseur ; Teknea - 1989 ; ISBN 2-87717-003-9