VERS UNE SPECIFICATION FORMELLE DES TACHES UTILISATEUR

Mourad Abed[(1)], Adel Mahfoudhi[(2)], Dimitri Tabary [(1)]

[(1)] *LAMIH (UMR CNRS 8530), Université de Valenciennes et du hainaut Cambresis*
BP : 311 – 59304 Valenciennes cedex9 (France)
Tél : +(33) 3.27.51.14.66 Fax : +(33) 4.27.51.13.16
Email : mourad.abed/dimitri.tabary@univ-valenciennes.fr

[(2)] *Institut Préparatoire aux Etudes d'Ingénieurs de Sfax*
Rte Menzel Chker km 0,5 BP : 805 -- 3018 Sfax (TUNISIE)
Tél : +(216) 4.241.403
Email : adel.mahfoudhi@ipeis.rnu.tn

Résumé :

L'apparition de l'interactivité a fait naître une somme importante et variée de compétences, de techniques et d'outils issus dans leur majorité du génie logiciel, de l'ergonomie et de la psychologie cognitive. Seulement, il reste souvent difficile, dans un processus de développement, d'utiliser conjointement des modèles fondés sur les sciences cognitives et des méthodes de génie logiciel, étant donné que leurs approches des problèmes sont différentes. De ce fait, nous proposons une méthode, nommée TOOD (Task Oriented Objet Description) qui s'impose comme une démarche fédératrice permettant la prise en compte des facteurs humains et la spécification formelle des systèmes interactifs. Elle utilise l'orienté objet et les réseaux de Petri objet dans toutes ses étapes de manière à maintenir la sémantique tout au long du cycle de développement. Ses modèles formels permettent un passage aisé à la génération d'interface et assurent la vérification des propriétés intéressantes du système à partir de ses spécifications.

Pour faciliter la compréhension de la méthode un exemple de procédure de tir d'une bombe sera présenté (conduite de tir).

Mots Clés :

Spécification formelle, analyse de la tâche, orienté objet, réseaux de Petri.

I INTRODUCTION

Plusieurs travaux ont été dédiés à la modélisation des tâches utilisateurs, dans le domaine de la conception des systèmes interactifs, par exemple les travaux consacrés aux méthodes: MAD [1], DIANE [2], GOMS [3]. Cependant l'utilisation effective de ces outils est loin d'être une pratique très répandue. L'une des raisons possibles est leur manque d'utilisation de méthodes véritablement formelles qui permettent d'apporter aux modèles de tâches la concision, la cohérence et la non-ambiguïté. De plus, ces travaux souffrent de leur manque d'intégration dans un processus global de conception couvrant l'ensemble du cycle de vie de l'Interface Homme-Machine (IHM). Afin de surmonter ces problèmes, les recherches actuelles s'orientent vers un cadre méthodologique qui s'étend de la phase amont de l'analyse de l'activité jusqu'à la phase de spécification détaillée de l'IHM. Les méthodes MAD* [4], DIANE+ [5], GLADIS ++ [6], ADEPT [7], TRIDENT [8] vont dans ce sens. Elles sont basées sur plusieurs modèles (modèle de la tâche, modèle de l'utilisateur, modèle de l'interface) et assistée par des outils d'implémentation de ces modèles. Nos travaux s'inscrivent dans cette orientation tout en mettant l'accent sur les aspects formels de représentation des modèles et leur transformation tout au long des étapes du processus de conception. La méthode proposée, TOOD (Task Object Oriented Description), est fondée sur la représentation que l'utilisateur a de la tâche en dehors des considérations de traitement informatique. Elle est basée sur l'utilisation conjointe de l'approche objet et des réseaux de Petri objets pour décrire, d'une part les aspects fonctionnels et la dynamique des tâches utilisateur, et d'autre part les aspects comportementaux de l'IHM et de l'utilisateur pour spécifier comment s'effectuent les tâches. Bien entendu, cette utilisation de l'approche objet et des Réseaux de Petri a fait l'objet d'autres travaux, comme UML/PNO [9], HOOD/PNO [10], ICO [11]. L'apport principal de TOOD est l'intégration directe de l'approche objet et des RdP dans un même formalisme, sans aucune translation, ce qui permet de préserver la sémantique des modèles. De plus, son formalisme vise à couvrir la totalité du cycle de développement de l'analyse de l'existant jusqu'à la conception détaillée et l'implémentation comme le montre la Figure1.

Dans cet article, nous nous limitons à présenter l'étape de modélisation des tâches avec TOOD, le lecteur trouvera une description plus détaillée de la méthode, de son support informatique d'édition et de simulation et la présentation d'une application complexe (Contrôle du Trafic Aérien) dans nos travaux [12].

Pour illustrer notre propos, nous avons choisi de décrire un exemple simple, pour monter la faisabilité de la méthode, d'une procédure de tir d'une bombe (conduite de tir) à partir d'un bombardier.

II LE CYCLE DE DEVELOPPEMENT TOOD

Le processus de conception TOOD peut être divisé en quatre grandes étapes, figure 1:

L'analyse du système existant et du besoin s'appuie sur l'activité de son utilisateur, et constitue le point d'entrée et la base de toute nouvelle conception.

Le modèle structurel de la tâche concerne la description des tâches utilisateurs du système à concevoir. Il permet de décrire, de façon cohérente et complète, la tâche utilisateur. Deux modèles sont élaborés à ce niveau: le modèle statique et le modèle dynamique de la tâche, afin de l'exploiter pour la spécification de l'IHM.

Le modèle opérationnel permet de spécifier les objets IHM ainsi que les procédures opérateur du système à concevoir. Il exploite les besoins et les caractéristiques du modèle structurel de la tâche pour aboutir à une interface compatible avec les objectifs et les procédures des opérateurs.

La réalisation de l'IHM concerne l'implémentation informatique des spécifications issues de l'étape précédente.

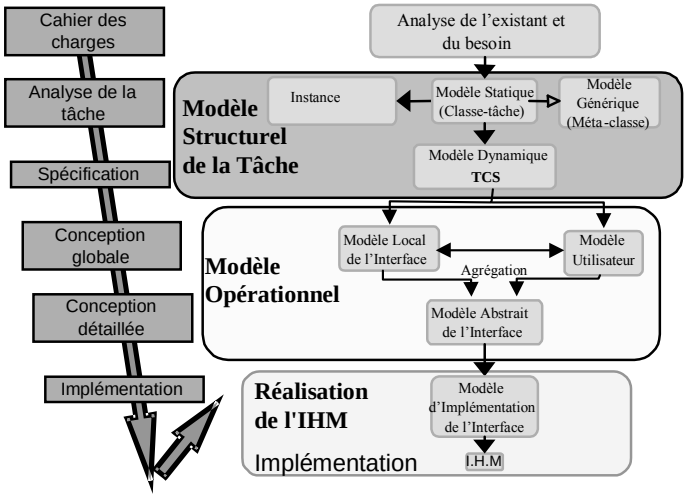


Figure 1: Cycle de développement de TOOD.

III MODELE STRUCTUREL DE LA TACHE (MST)

Le MST a pour objectif principal d'établir une description, cohérente et complète, de la tâche "future" de l'utilisateur. Pour ce faire, deux types de modèle sont élaborés: un modèle statique (MSST) et un modèle dynamique (MSDT)

Tout comme MAD* [4], Diane+ [5], le MST est conçu comme un moyen de prendre en compte l'utilisateur et sa tâche dans le cycle de conception. L'objectif est de fournir un outil méthodologique aux équipes de développement permettant d'abstraire l'information de la tâche utilisateur nécessaire à la conception des interfaces de manière formelle pour permettre son intégration dans les cycles de conception informatique. Il s'agit d'un langage de travail facilitant le dialogue et les échanges d'information entre les participants d'une équipe de conception.

La construction du modèle structurel passe par quatre étapes itératives :

1. Décomposition hiérarchique des tâches.
2. Identification des objets descripteurs et des objets du monde.
3. Définition de la dynamique des tâches élémentaires et de contrôle.
4. Intégration de la concurrence de tâches (interruptions)

III.1 Modèle Structurel Statique de la Tâche (MSST)

Le MSST permet une décomposition du travail prescrit de l'utilisateur avec le système interactif en éléments significatifs, appelés tâches. Chaque tâche dans TOOD est considérée comme une instance d'une classe-tâche. Cette dernière définie la structure générique qui permet de décrire et d'organiser les informations nécessaires à l'identification et à la réalisation de chaque tâche. (Figure 2).

Ce rapprochement au génie logiciel assure un lien fort entre une spécification basée sur un modèle de tâche et la conception logicielle basée sur le modèle objet

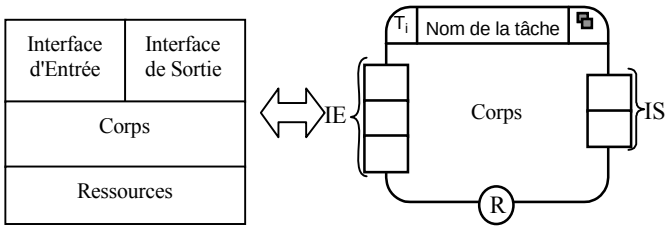


Figure 2: Structure générique de la classe-tâche

La classe-tâche est composée d'une **Interface d'Entrée**, d'une **Interface de Sortie**, de **Ressources** et d'un **Corps**. A ces composants, des identificateurs sont associés. Ils permettent de distinguer une instance de la classe-tâche parmi les autres :

- Un nom, verbe d'action suivi d'un complément (objet traité par la tâche), reflète le traitement à réaliser par la tâche. Il est souhaitable que le nom reprenne le vocabulaire utilisé par les opérateurs de manière à respecter la terminologie lors du développement de l'interface.
- Un but donne une explication en langage naturel du but que l'opérateur ou l'application veut atteindre au travers de la tâche.
- Un indice identifie formellement de la tâche. Il est formé à partir du numéro de la tâche mère, à laquelle on ajoute le numéro séquentiel correspondant à la dite tâche.
- Un type indique la nature de la tâche (humaine, automatique, interactive ou coopérative)
- Une hiérarchie, symbolisé par des petits carrés, définit le nombre de classes-tâches composantes.

Le **Corps** représente l'unité centrale de la classe-tâche. Il peut être décomposé récursivement en sous-tâches jusqu'aux tâches élémentaires. Les tâches élémentaires sont des tâches procédurales qui définissent les procédures d'actions utilisateur et le comportement de l'interface correspondant à ces procédures. La spécification de ces procédures est réalisée au niveau du modèle opérationnel. Les tâches

intermédiaires, appelées de contrôle, décrivent les relations d'assemblage par des constructeurs logico-temporels (ET, OU, Exclusivité, …) sous-tâches. Ces relations reflètent, d'une certaine manière, l'organisation du travail de l'utilisateur.

Les **Ressources** représentent les opérateurs humains et/ou les entités du système d'interaction impliquées dans l'exécution de la tâche. Les tâches sont réalisées par une collaboration de ces ressources. Par conséquent, quatre types de tâche sont définis: Manuelle, Automatique, Interactive ou Coopérative. (Tableau 1)

Type de tâche	Ressource Humaine	Ressource Système
Manuelle	1..1	0
Automatique	0	1..N
Interactive	1..1	1..N
Coopérative	2..N	0..N

Tableau 1: Type de tâche

Ainsi une tâche manuelle est accomplie par un et un seul utilisateur. Une tâche automatique peut être effectuée par une ou plusieurs ressources système. Une tâche interactive est accomplie par l'interaction d'un utilisateur avec un ensemble de ressources système. Enfin, une tâche coopérative requiert l'activité de deux ou plusieurs utilisateurs qui interagissent entre eux (coopération humaine) ou sur un ensemble de ressources système (coopération interactive).

L'**Interface d'Entrée** spécifie l'état initial de la tâche. Elle définit les données nécessaires à l'exécution de la tâche. Ces données sont considérées comme les conditions initiales à satisfaire obligatoirement au début de la tâche. Elle est composée de trois catégories d'information, Tableau 2: les *Déclencheurs*, les *Conditions contextuelles* et les *Données d'Entrée.*

L'**Interface de Sortie** spécifie l'état final de la tâche. Elle est composée de deux types de données, Tableau 2: Les *comptes-rendus* et *les Données de Sortie*.

		Description
Interface d'Entrée	***Déclencheurs***	Evénements qui provoquent l'exécution de la tâche. Ils sont classés en deux catégories : • Les événements d'enclenchement formel ou explicite correspondent à des déclencheurs extérieurs. Ils apparaissent de façon observable dans l'environnement du travail (information sur écran, appui sur un bouton, communication, …). Les tâches déclenchées par ce type d'événement sont considérées comme obligatoires; c. à. d leur exécution est indispensable. • Les événements d'enclenchement informel ou implicite sont non visibles pour un observateur.
	Conditions contextuelles	Informations qui influencent le comportement de l'exécution de la tâche. Ces informations définissent, par exemple, les consignes, le temps,….
	Données d'Entrée	Informations sur lesquelles s'effectue l'activité de la tâche.
Interface de Sortie	***Comptes-rendus***	Résultats produits par l'exécution de la tâche. Leur contenu indique les modifications d'ordre : • physique et dans ce cas, il désigne la modification de l'environnement (appel applicatif, changement d'état….). • mental, désignant la modification ou la nouvelle représentation de la situation par l'opérateur. Ces comptes-rendus déterminent ainsi si les objectifs sont atteints ou non et, dans tel cas, la tâche sera reprise après une éventuelle évolution de la situation.
	Données de Sortie	Données d'entrée transformées ou données créées par la réalisation de la tâche.

Tableau 2 : Données de l'interface d'entrée et de sortie.

Les ressources et les informations des interfaces d'entrée et de sortie sont modélisées par des objets, appelés "objets descripteurs", qui ont une structure et un comportement.

D'un point de vue informatique, ces objets descripteurs représentent les composants de la Classe-Tâche et spécifient l'utilisation des objets du monde. De cette manière, il reflète une relation de correspondance entre les objets du monde et les objets descripteurs.

D'un point de vue opérateur, les objets représentent aussi les données par lesquelles il comprend son environnement de travail et élabore une image mentale pour effectuer sa tâche. Par conséquent, ces objets pour les tâches interactives auront une existence dans l'image finale du système.

Les classes d'objets descripteurs sont organisées selon une relation d'héritage (généralisation spécialisation) afin de traduire la similitude de structure qui existe entre elles, Figure 3. La structure de Généralisation Spécialisation est représentée par une Super-Classe "DESCRIPTEUR" et les classes spécialisées par des classes correspondantes à la nature de l'information utilisée dans l'Interface d'entrée, l'Interface de sortie et les ressources.

Ces classes sont définies par :

- Nom : identification de la classe. Chaque classe spécialisée porte le nom qui définit la nature de sa spécialisation.
- Indice : référence l'objet de manière formelle.
- Description : explication en langage naturel du rôle de l'objet.

- Attribut : indique les caractéristiques que l'on souhaite modéliser en référence aux objets du monde réel.

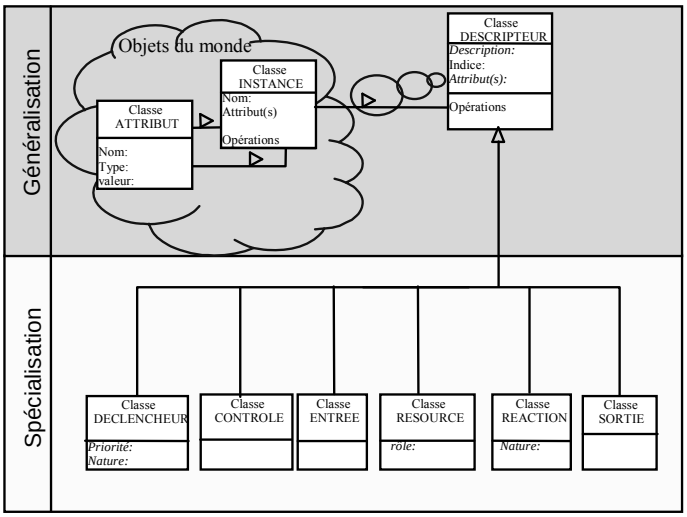
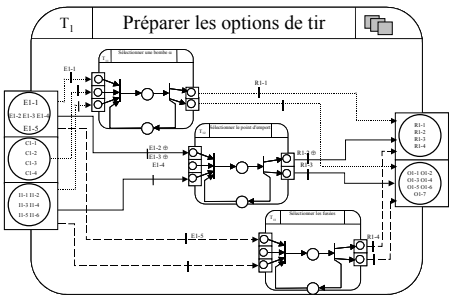


Figure 3 : Hiérarchie des classes.

Dans notre exemple de conduite de tir, l'analyse de l'existant et du besoin ont fait apparaître que le pilote de l'avion doit pouvoir sélectionner une bombe, un point d'emport (PE), point où la bombe est située sur l'avion, et enfin le mode d'explosion de cette bombe (fusée). Il doit également choisir les options opérationnelles de réalisation de son tir. Il n'y a pas d'ordre préférentiel pour la réalisation des tâches, elles sont réalisées une première fois et peuvent être exécutées à nouveau afin d'obtenir un résultat différent (exemple modifier une option de conduite).

A partir de cette analyse, la construction du modèle structurel commence par une décomposition hiérarchique dans un graphe de tâches. Il est constitué d'une tâche racine "Effectuer une conduite de tir" décomposée en deux sous-tâches parallèles "Préparer les options de tir" et "Préparer les options de conduite de tir" associées aux objectifs de l'utilisateur. La tâche "préparer les options de tir", étant une tâche de contrôle, est composée en trois sous-tâches "Sélectionner une bombe α", "Sélectionner un point d'emport" et "Sélectionner les fusées". Cette décomposition est traduite dans des diagrammes qui permettent de définir les liens inter-tâches. La Figure 4 montre la décomposition de la tâche "Préparer les options de tir", accompagné de sa fiche textuelle.



Classe Préparer les options de
{
//Nom
Nom = Préparer les options de
//Bat
Bat = préparation d'une
//indice
Tc T1
//Type
Type:
//Hiérarchi
Classe mère:Effectuer une conduite de
Classes filles: T11: Sélectionner une α
T12: Sélectionner le point
T13: Sélectionner les
//DECLENCHEU
E1-1: E0-1: Sélecteur
E1-2: E0-2: Init
E1-3: E0-3: Sélecteur
E1-4: E0-4: Sélecteur
E1-5: E0-5: Sélecteur
//CONTROL
C1-1: C0-1: Point d'emport
C1-2: C0-2: Capteur présent
C1-3: C0-3: Capteur présent
C1-4: C0-4: Mode de sélection
//ENTREE
I1-1: I0-1: Tension
I1-2: I0-2: Tension
I1-3: I0-3: Mode de sélection
I1-4: I0-4: Point d'emport
I1-5: I0-5: Fusée
//REACTION
R1-1: R0-1: Sélection d'une
R1-2: R0-2: Nouveau point emport sélectionné
R1-3: R0-3: Nouveau point d'emport sélectionné
R1-4: R0-4: Nouvelle fusée
//SORTIE
O1-1: O0-1: Bombe
O1-2: O0-2: Tension
O1-3: O0-3: Tension
O1-4: O0-4: Mode de sélection
O1-5: O0-5: Point d'emport
O1-6: O0-6: Fusée
//MOYEN
M1-1: M0-1: Pilote
M1-2: M0-2: Zone sélection α
M1-3: M0-3: Applicatio
M1-4: M0-4: Zone sélection point
M1-5: M0-5: Zone de sélection
}

Figure 4 : Diagramme de la classe tâche "préparer les options de tir"

La construction du modèle se poursuit en définissant les objets descripteurs et les objets du monde de chaque tâche. La Figure 5 spécifie les classes descripteurs composant la classe-tâche "sélectionner une bombe α". Dans cet exemple, la classe déclencheur E11-1 caractérise un événement de sélection manuelle d'une bombe. A l'attribut de cette classe correspond un objet du monde "Sélecteur α" de type booléen, sélecteur on/off.

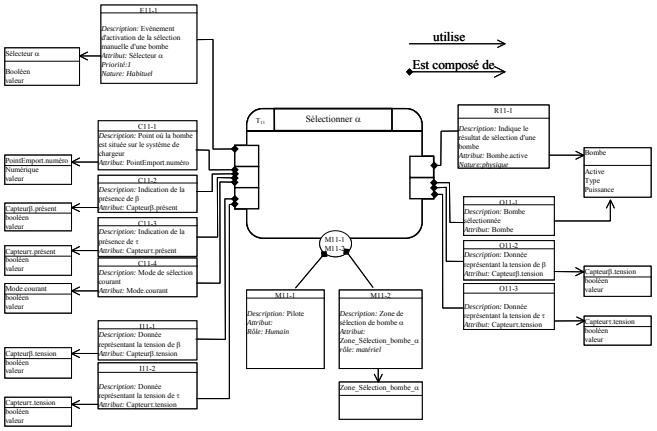


Figure 5 : Identification des objets descripteurs de la classe tâche "Sélectionner une bombe α"

III.2 Modèle Structurel Dynamique de la Tâche (MSDT)

Le Modèle Structurel Dynamique de la Tâche (MSDT) a pour objectif d'intégrer la dimension temporelle (séquencement, synchronisation, concurrence) en complétant le modèle statique. Le comportement dynamique de tâches est défini par une structure de contrôle, appelée TCS (Task Control Structure), modélisée par réseau de Petri à objets (RPO) [13],
Figure **6**. Cette TCS décrit la consommation des objets descripteurs de l'interface d'entrée, l'activité de la tâche, la libération des objets descripteurs de l'interface de sortie ainsi que l'occupation des ressources.

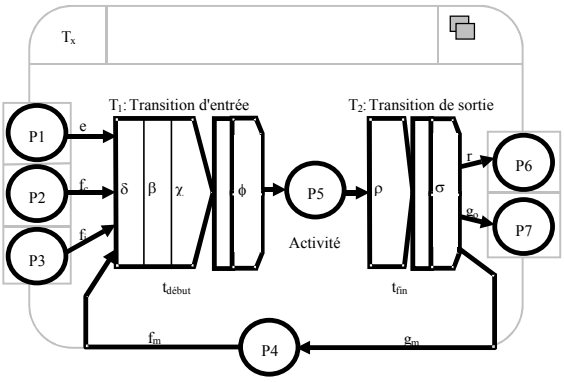


Figure 6 : Structure de Contrôle de la classe-tâche.

La TCS est typée par l'ensemble C constitué des six classes d'objets descripteurs C= {DECLENCHEMENT, CONDITION, DONNEE_ENTREE, RESSOURCE, DONNEE_SORTIE, REACTION}.

Chaque TCS possède une transition d'entrée T1 et une transition de sortie T2. Les fonctions associées à chaque transition permettent la sélection des objets et définissent leurs répartitions par rapport à l'activité de la tâche.

La transition T1, est composée de trois fonctions: **δ, β, χ**

- La **fonction de priorité δ** permet de sélectionner le déclencheur le plus prioritaire pour la tâche.

  $$\delta : E \rightarrow Z$$
  $$\delta(E_{x,j}) = \sup(E_{x,1}, E_{x,2}, \ldots, E_{x,n}) \quad \text{pour une tâche Tx}$$

  Cette fonction est à la base du système d'interruption. Elle permet d'initier l'exécution d'une tâche, même si une autre tâche moins prioritaire est en cours d'exécution. Cependant l'exécution de la tâche par rapport à ce déclencheur reste conditionnée à la vérification des fonctions de complétude et de cohérence.

- La **fonction de complétude β** vérifie la présence de tous les objets descripteurs relatifs à l'événement observé, c.à.d les données d'entrée, de contrôle et les ressources utilisées pour activer la classe tâche par rapport à un événement de déclenchement donné.

  $$\beta : E \rightarrow < E, I, C, M >$$
  $$E_{x,j} \mapsto \beta(E_{x,j}) = < E_{x,j}, f_c, f_i, f_m >$$
  $$\text{avec:} \begin{cases} E_{x,j} \in E(E_{x,j} : \text{le jéme événement de la tâche } T_x) \\ f_c \subset C \\ f_i \subset I \\ f_m \subset M \end{cases}$$

- La **fonction de cohérence χ** évalue la recevabilité de ces descripteurs par rapport aux conditions envisagées pour la tâche. Comme [4], cette fonction définit un ensemble de règles de vérification qui utilisent des opérateurs simples de type logique et mathématique et obéissent à une syntaxe unique permettant leurs formulations. Cette syntaxe est donnée par la règle récursive suivante:

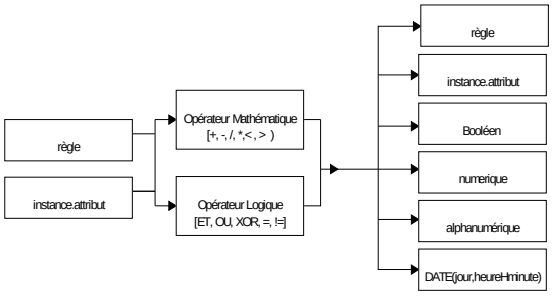


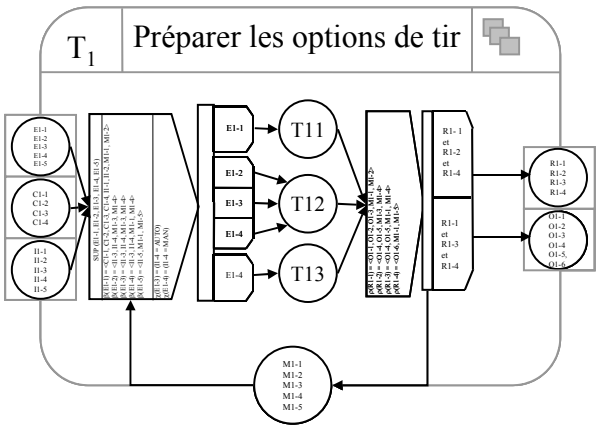
Figure 7 : Syntaxe des règles récursives adaptée de [4]

La transition de sortie T2 possède une fonction de **complétude ρ** qui vérifie la présence des données de sortie et des ressources associées aux réactions libérées par le corps de la tâche.

$$\rho : R \rightarrow < R, O, M >$$
$$R_{k,j} \mapsto \rho(R_{x,j}) = < R_{x,j}, g_o, g_m >$$
$$\text{avec :} \begin{cases} R_{x,j} \in R(R_{x,j} : \text{la jéme réaction de la tâche Tx}) \\ g_o \subset O \\ g_m \subset M \end{cases}$$

Les tâches hiérarchiques sont considérées comme des **tâches de contrôle** de leurs tâches composantes. Par conséquent, les transitions d'entrée et de sortie de leurs TCS possèdent respectivement une fonction **ϕ** d'émission et une fonction **σ** de synchronisation.

La fonction **ϕ** définit, pour la transition T1, les **règles d'émission** (constructeurs de la transition d'entrée) d'activation des sous-tâches, ainsi que la répartition des données consommées par ces dernières. (Tableau 3)



Transition d 'Entrée	Constructeur	Symbole	Transition	Ordre de priorité	Partage ressource	Description
	Jonction et Distribution (simultanéité)			Cst	non	n tâches sont réalisées en même temps par m ressources différentes. Ces tâches sont enclenchées par le même déclencheur ou bien par des déclencheurs différents.
	Transfert (OU)			oui	oui	n tâches sont exécutées dans l'ordre de priorité de leurs déclencheurs. Les tâches partagent les données et les ressources. Ces tâches sont interruptibles.
	Transfert Avec condition			oui	-	n tâches sont exécutées dans l'ordre de priorité de leurs déclencheurs qui satisfont certaines conditions. Les tâches partagent les données et les ressources. Ces tâches sont interruptibles.
	Transfert Alternatif			non	-	Une et seulement une tâche est déclenchée. Les déclencheurs sont similaires, mais dont seulement un est pris en fonction du contexte.

Tableau 3 : Constructeurs de la transition d'entrée.

La fonction **σ** définit, pour la transition T2, les **règles de synchronisation** (constructeurs de la transition de sortie) des sous-tâches. (Tableau 4)

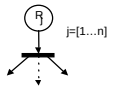
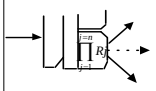
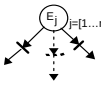
Transition de Sortie	Constructeur	Symbole	Transition	Description
	Synchronisation			n sous-tâches doivent être terminée pour que la tâche gestionnaire soit terminée. La tâche gestionnaire libère soit les Rj réactions, soit une nouvelle réaction.
	Ou			La tâche gestionnaire se termine lorsqu'au moins une de ces sous-tâches est terminée.
	Alternatif			La tâche gestionnaire se termine quand une seule de ses tâches filles est terminée

Tableau 4 : Constructeurs de la transition de sortie.

Dans notre exemple, Figure 8, la tâche de contrôle T1 "préparer les options de tir" gère les trois sous tâches T11, T12, T13. Ces tâches sont exécutées en parallèle en fonction de la priorité de leur déclencheur (constructeur OU). Le pilote peut donc "choisir une bombe α", "sélectionner un point d'emport" ou "sélectionner une fusée" dans un ordre quelconque. On remarque que la tâche T12 "sélectionner un point d'emport" peut être réalisée par rapport à trois déclencheurs alternatifs (constructeur Alternatif) en effet le choix du point d'emport est réalisé de façon exclusive à l'initialisation du système, ou pendant la préparation d'un nouveau tir par le système lors d'une sélection automatique ou par le pilote lors d'une sélection manuelle. Enfin, la tâche de contrôle T1 se termine soit quand une bombe est sélectionnée de façon manuelle, soit d'une manière automatique par le système (constructeur Alternatif et constructeur Synchronisation).

Figure 8 : TCS de la tâche de contrôle "Préparer les options de tir"

Cette gestion définit également la fonction de priorité entre les déclencheurs, δ = (E1-1, E1-2, E1-3, E1-4, E1-5), et leurs fonctions de complétude. Dans ce cas d'exemple, la fonction de complétude du déclencheur E1-1 "Sélecteur bombe α" s'exprime par :

β(E1-1) = <C1-1, C1-2, C1-3, C1-4, I1-1, I1-2, M1-1, M1-2> avec
C1-1 = Point d'emport courant
C1-2 = Capteur b présent
C1-3 = Capteur t présent
C1-4 = Mode de sélection courant
I1-1 = Tension du capteur b
I1-2 = Tension du capteur t
M1-1 = Pilote
M1-2= Zone de sélection de bombe α

De même, la règle de cohérence du déclencheur E1-3 est χ(E1-3) = (I1-1 = AUTO). Elle spécifie la contrainte que le système de conduite doit être initialement en mode automatique pour basculer vers une sélection de point d'emport manuel.

Enfin, la tâche peut se terminer par rapport au compte-rendu R1-1: "bombe sélectionnée". Ce résultat est obligatoirement accompagné des données de sortie et des ressources spécifiées par la fonction de synchronisation ρ(R1-1) =<O1-1, O1-2, O1-3, M1-1, M1-2>.

III.3 Définition de la concurrence des tâches

Afin de guider le concepteur lors des étapes de spécification, nous proposons un mécanisme de modélisation des interruptions de tâche. Peu de méthodes permettent de formaliser les interruptions de tâche; [14] présente une modélisation à base de Réseaux de Petri des interruptions en complément de MAD et de UAN. L'avantage de notre méthode est de compléter la TCS par le mécanisme d'interruption tout en gardant le même formalisme, Figure 9

Ce mécanisme permet de modéliser l'interruption d'une tâche. L'interruption se réalise quand un nouveau déclencheur plus prioritaire demande l'exécution d'une tâche dont les ressources sont utilisées par une autre tâche moins prioritaire. A ce moment la tâche en cours d'exécution passe dans un état suspendu (P8) et libère ses ressources. La tâche prioritaire est ainsi exécutée. Une fois que cette tâche est terminée et que les ressources sont de nouveaux libres la tâche suspendue reprend son exécution selon trois cas possibles: au début, au point où elle est arrêtée ou à la fin quand elle est abandonnée.

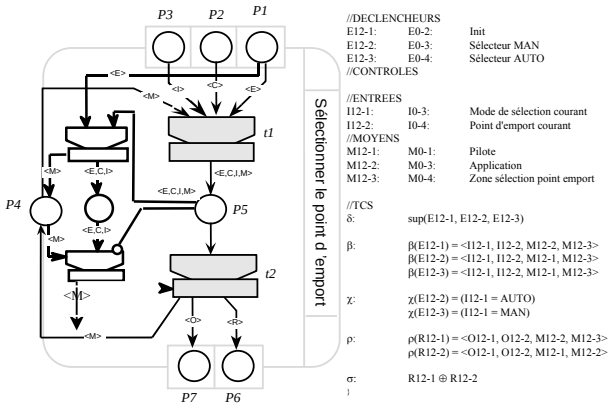


Figure 9 : Structure d'interruption de la tâche "Sélectionner le point d'emport"

Dans l'exemple de Figure 8, la tâche "Sélectionner le point d'emport" peut être réalisée manuellement ou automatiquement. Cependant le pilote peut en cours de sélection manuelle basculer vers le mode automatique. La sélection automatique est plus prioritaire que la sélection manuelle. Ceci est modélisé d'une part en donnant une priorité plus élevée au déclencheur E12-3 relatif au sélecteur auto, et d'autre part en complétant la TCS d'une structure d'interruption, Figure 9.

IV CONCLUSION

Nous avons montré dans cet article que l'utilisation de l'orienté objet et des réseaux de Petri objet présente plusieurs avantages pour modéliser la tâche de l'utilisateur. En effet, le modèle de tâches de TOOD, par sa description statique et dynamique, permet la modularité des spécifications, l'expression des interruptions et de la concurrence. L'adjonction des objets descripteurs à l'entité tâche permet un rapprochement à un langage de programmation ce qui simplifie le passage à l'implémentation.

En outre, la méthode TOOD peut contribuer à aider la communication entre les différents intervenants dans le

processus de conception par le biais de sa description formelle. Cependant, nous n'avons pas abordé le modèle opérationnel qui conduit à la spécification de l'IHM. Ce modèle est conçu dans la continuité du modèle structurel en utilisant les mêmes formalismes ce qui privilégie la stabilité sémantique de la méthode TOOD.

REFERENCES

[1] Scapin, D.L., Pierret-Golbreich, C. Towards a method for task description: MAD. Work with Display Unit'89 in Berlinguet, L & Berthelette, D. (Eds.) CAP, (1990)

[2] Barthet, M.F. : Logiciels interactifs et ergonomie : modèles et méthodes de conception, Dunod , (1988)

[3] Card, S.K, Moran, T.P, Newell A.: The Psychology of HCI. In Lawrence Erlbaum Ass (Ed.). London, (1983)

[4] Gamboa-Rodriguez, F. Spécification et implémentation d'ALACIE: Atelier Logiciel d'Aide à la Conception d'Interfaces Ergonomiques. Thèse en sciences: Université Paris XI, (1998)

[5] Tarby, J-C & Barthet, M-F. The Diane+ Method in CADUI'96, pp95-119, (1996)

[6] Ricard, E. & Buisine, A. Des tâches utilisateur au dialogue homme-machine: GLADIS++, une démarche industrielle. IHM96, pp71-76, (1996)

[7] Johnson, P., Johnson, H. Wilson, S.: Scenario-based design and task analysis. In Carroll, J.M. (Ed.). Scenario-based design: Envisioning work and technology in system development. Willey, (1995)

[8] Vanderdonckt, J.: Conception assistée de la présentation d'une interface homme-machine ergonomique pour une application de gestion hautement interactive. Thèse, Faculté Notre Dame de la Paix Louvain, Belgique, (1997)

[9] Delatour, J. & Paludeto, M. De HOOD/PNO à UML/PNO : Une méthode pour les systèmes temps réels basée sur UML et objets à réseaux de Petri. Rapport LAAS N° :98248, (1998)

[10] Paludetto, M. & Benzina, A. Une méthodologie orientée objet HOOD et réseaux de Petri. : Concepts et outils pour les systèmes de production in J-C. Hennet (ed.) Cépadués, p293-325, (1997)

[11] Palanque, P., Bastide, R & Paterno, F. Formal specification as a tool for objective assessment of safety-critical interactive systems. In proceedings of the IFIP TC13 conference on HCI, Interact'97, pp 323-330. Sydney, (1997)

[12] Mahfoudhi, A. TOOD: Une méthodologie de description orientée objet des tâches utilisateur pour la spécification et la conception des interfaces homme-machine. Thèse en automatique. Université de Valenciennes, (1997)

[13] Sibertin-Blanc, C. High-level Petri nets with Data Structure. 6th EWPNA, Espoo (Finland), (1985)

[14] Jambon, F. Erreurs et Interruptions du point de vue de l'ingénierie de l'interface homme-machine. Thèse en informatique: Université Joseph Fourier, (1996)