MASTER EN SCIENCES INFORMATIQUES

Contribution à l'enseignement assisté par ordinateur dans le cadre des télécommunications. Etude du transfert, de l'accès et de la gestion de fichiers par FTAM

Corbugy, Dominique; Denis, Joël

Award date:
1991

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX
NAMUR

Institut d'informatique

Contribution à l'enseignement assisté par ordinateur dans le cadre des télécommunications. Etude du transfert, de l'accès et de la gestion de fichiers par FTAM.

Mémoire présenté par Dominique CORBUGY et Joël DENIS en vue de l'obtention du grade de Licencié et Maître en Informatique

Promoteur - Professeur Philippe Van Bastelaer

Année académique 1990-1991

Contribution à l'enseignement assisté par ordinateur dans le cadre des télécommunications. Etude du transfert, de l'accès et de la gestion de fichiers par FTAM.

Mémoire présenté par
D. CORBUGY et J. DENIS

Année Académique 1990-1991

Qu'il nous soit permis de remercier vivement notre promoteur, le professeur Philippe Van Bastelaer, pour ses conseils tout au long de la réalisation de ce travail.

Nous remercions également Madame Véronique Nachtergaele et Monsieur Dominique de Paul respectivement assistante et étudiant aux FNDP pour leur assistance lors de l'élaboration de ce travail.

Nous tenons également à remercier Monsieur Serge Simonet de la société SPAG pour ces précieuses remarques concernant la norme FTAM.

Enfin, nous remercions toutes les personnes qui ont participé de près ou de loin à l'achèvement de ce travail.

Résumé

Dans ce travail, nous présentons les principaux concepts de la norme d'accès, de transfert et de gestion de fichier (FTAM) réalisée par l'Organisation de Standardisation Internationale (OSI). Sur base de critères pédagogiques, nous élaborons un scénario destiné à l'enseignement assisté par ordinateur du modèle étudié. Nous réalisons une partie du scénario en langage Objective-C dans l'environnement Real time Measurement Graphics (RMG). Nous complétons ce travail par des extensions au scénario et par des considérations sur la conception et la réalisation de logiciels destinés à l'enseignement.

Abstract

In this work, we present the fondamental aspects of the International Standardization Organization (ISO) File Access, Transfer and Management model (FTAM). We use pedagogical criteria for the construction of a script which is dedicated for the Computer Aided Learning of the studied model. We realize a part of this script in Objective-C language in a Real-time Measurement Graphics (RMG) environment. We complete this work by giving possible extensions of the script and some considerations about the conception and the implementation of software studied for learning.

Table des matières

Introduction

Dans ce mémoire, nous avons tenté d'apporter une contribution à l'enseignement assisté par ordinateur (EAO) dans le domaine des télécommunications. Nous nous sommes intéressés à un sujet particulier qui concerne l'enseignement de FTAM (File Access, Transfer and Management), une norme concernant le transfert, l'accès et la gestion de fichiers réalisés par l'OSI (Organisation de Standardisation Internationale). Nous avons réalisé une partie d'un logiciel qui permet l'apprentissage du fonctionnement du modèle.

L'enseignement assisté par ordinateur est relativement nouveau dans le cadre de l'équipe de Namur. C'est pourquoi, ce mémoire constitue avant tout un rassemblement de connaissances dans les divers domaines concernés par l'apprentissage ; à savoir, les connaissances théoriques relatives aux télécommunications et à la pédagogie mais aussi, plus concrètement, des connaissances acquises au cours de notre travail. Nous pensons que notre travail est important car il aide à mieux cerner les problèmes qui peuvent se poser tout au long du développement des logiciels d'EAO dans notre équipe.

Ensuite, nous allons introduire le cadre de notre travail tant du point de vue matériel (chapitre 1) que du point de vue théorique (chapitre 2 et 3). Le chapitre 2 est consacré à l'exposé des principaux concepts relatifs à la norme FTAM. Le chapitre 3 reprend une méthodologie de conception de logiciels destinés à l'enseignement ainsi que des critères pédagogiques relatifs à la conception. Nous disposons alors des bases nécessaires pour l'élaboration d'un scénario (chapitre 4) ayant pour objectif de faciliter l'apprentissage de notions relatives à FTAM. L'étape suivante consiste à implémenter une partie de notre scénario dans un environnement RMG (chapitre 5). Certains concepts de FTAM jugés importants et non repris dans le scénario sont évoqués dans le chapitre 6. Suite à l'expérience acquise et aux problèmes rencontrés, nous sommes à même d'apporter une critique constructive de notre mémoire (chapitre 7). Nous terminons par une synthèse de notre travail et par les prolongements qu'il suppose.

L'enseignement assisté par ordinateur

Dans le chapitre 1, nous décrivons le projet d'enseignement assisté par ordinateur. Nous situons l'équipe de Namur dans le cadre du projet universitaire COLOS (COnceptual Learning Of Science). Nous spécifions les buts ainsi que les moyens techniques et logiciels disponibles. Nous développons, notamment, le langage orienté objets et l'environnement de travail RMG (Real time

Measurement Graphics) qui constituent le cadre de développement de notre application. Plus particulièrement, nous présentons les travaux de l'équipe de Namur, notre projet et les relations avec les autres activités.

FTAM

Le chapitre 2 est consacré à une description générale du fonctionnement de FTAM. Nous nous arrêtons sur les concepts fondamentaux de FTAM. Puis, nous traitons dans de plus amples détails les sujets auxquels nous ferons référence par la suite. Ces sujets concernent la définition des primitives et leurs enchaînements.

Méthodologie de conception d'un didacticiel

Le chapitre 3 définit une méthodologie de conception de logiciels destinés à l'enseignement (on parle alors de didacticiels). Nous agrémentons ce chapitre de conseils pédagogiques généraux sur la manière de réaliser un didacticiel. Nous abordons également les intérêts et les inconvénients de l'enseignement assisté par ordinateur.

Scénario d'une application FTAM

Les chapitres précédents nous ont fourni tous les outils nécessaires pour nous attaquer au développement d'un scénario dans le détail. Nous avons choisi d'expliquer le fonctionnement de FTAM du point de vue d'un utilisateur travaillant avec un service fiable de fichiers (Reliable File Service). Après l'introduction du cadre du scénario et la justification de son utilité, nous en faisons une description détaillée. Elle comprend plusieurs concepts fondamentaux de FTAM comme les différents changements de régime, l'enchaînement des primitives, les effets de celles-ci. L'état dans lequel se trouve une entité et les transitions d'états possibles seront également présentés. La description s'appuie sur un certain nombre de critères pédagogiques. En outre, nous soulignons les problèmes rencontrés et la manière de les résoudre.

Médiatisation

A ce stade, les choix ou les restrictions que nous effectuons sont justifiés sur base d'aspects pratiques posés par l'environnement RMG. Nous élaborons nos classes en utilisant au maximum les propriétés des langages de programmation orientés objets, à savoir la réutilisation des classes préexistantes. Les critères de choix qui nous ont guidés lors de la définition des classes sont mis

en évidence. Nous abordons la manière dont nous avons conçu le programme. Nous soulevons un certain nombre de problèmes posés par l'environnement RMG.

Extensions du scénario

Dans le chapitre 6, nous essayons de fournir une liste de concepts dont l'illustration nous semble intéressante parce qu'elle aborde un aspect particulièrement difficile à comprendre pour l'étudiant ou parce qu'elle constitue un aspect fondamental de FTAM. Nous accompagnons les propositions par des idées de solutions. Ce chapitre nous offre également la possibilité d'aborder certains aspects fondamentaux de l'enseignement du modèle ISO.

Critique

Nous critiquons notre travail sur base des problèmes rencontrés à chaque étape de l'élaboration du mémoire, que ces problèmes soient relatifs à FTAM, à RMG ou autres. La critique tente de se montrer volontairement constructive. Nous espérons avoir réalisé la synthèse de l'expérience que nous avons acquise afin d'en faire bénéficier le projet.

Conclusion

La conclusion présente une synthèse du travail réalisé. Elle nous offre également l'occasion de proposer des développements futurs.

Chapitre 1 : L'enseignement assisté par ordinateur

Les enseignants recherchent depuis de nombreuses années d'autres méthodes pour clarifier leurs exposés et veulent profiter des possibilités offertes par les nouvelles technologies. Une simulation s'avère être une excellente solution dans le sens où elle sert de support à un travail intellectuel de la part de l'apprenant, excitant son imagination, lui permettant d'atteindre une meilleure compréhension du problème abordé. Nous appelons "simulation" l'exécution sur ordinateur d'un logiciel traduisant le comportement d'un phénomène. La valeur pédagogique de cette simulation s'accroît si l'application est interactive. En effet, l'apprenant visualise les conséquences de ses actions sur les paramètres du modèle. La compréhension de la simulation amène à une meilleure connaissance du phénomène illustré. Les simulations scientifiques se sont vues décerner de hautes qualités pédagogiques [BONN90].

1. Le projet COLOS

Le projet COLOS (**COnceptual Learning Of Science**) regroupe dix équipes universitaires européennes travaillant au développement de simulations pédagogiques dans un environnement orienté objets. Les recherches sont dirigées pour fournir des applications graphiques de haut niveau, fortement interactives, utilisables dans l'enseignement des concepts fondamentaux des sciences physiques et informatiques. L'initiative de ce projet revient à la société Hewlett-Packard qui a doté chaque équipe d'une station de travail HP 9000/360 et d'un logiciel adapté à la réalisation de simulations pédagogiques, RMG (**Real time Measurement Graphics**), développé dans ses laboratoires par Charles Young [FAZA89].

2. Le logiciel RMG

La station de travail fonctionne sous le système d'exploitation UNIX. Le logiciel RMG est écrit dans le langage Objective-C préprocesseur objet du langage C. Ce logiciel rassemble un important nombre de classes que nous décrirons en le décomposant en :

- une "boîte à outils" de classes servant de point de départ pour la réalisation d'applications et de classes fondamentales adaptées à l'interface utilisateur graphique ;
- un ensemble d'outils d'aide à la conception et à la mise au point de programmes (scénarii);
- un grand nombre d'applications prototypes réalisées par les laboratoires de Hewlett-Packard et par les partenaires européens du projet COLOS.

L'aspect graphique en temps réel de RMG se réfère à sa capacité à recalculer la mémoire d'affichage et redessiner l'écran en un temps inférieur au temps de perception de l'utilisateur. Cela rend le déroulement de l'animation agréable à suivre bien que dans le monde réel, la vitesse du phénomène peut s'avérer très différente. L'intérêt particulier de l'apprenant est d'observer l'évolution du modèle suite aux actions qu'il active, au moment où il les active. La relation entre le logiciel et l'apprenant fait essentiellement appel à la vision, ce qui situe l'importance de l'interface graphique du produit.

Les propriétés d'héritage de l'environnement orienté objets, dans lequel s'insère RMG, permettent au concepteur de logiciels de modifier une partie d'une application préexistante. Il doit se préoccuper uniquement de la partie qu'il écrit. Les propriétés du produit résultat sont celles du logiciel original agrémentées par celles découlant des modifications effectuées. La programmation orientée objets simplifie la construction d'interfaces graphiques.

L'environnement RMG offre à l'utilisateur des moyens pour définir facilement des classes, des méthodes, des instances, des messages. La boîte à outils comprend également une palette impressionnante de couleurs, un éditeur de texte, la possibilité de définir des icônes. Des outils pour la création de menus sont disponibles. RMG offre une méthode de travail dénommée "prototypage" qui permet de créer des scénarii sous forme de diagramme de flux dans lesquels les objets sont contrôlés par un ensemble de lois ou d'équations attachées à chaque scénario. Les scénarii et les fenêtres contenant les objets sont créés par des manipulations graphiques.

La documentation de RMG **[FAZA89]** se compose de trois parties. La première est un guide de l'utilisateur novice (**RMG User's First Aid Kit**). Elle dirige les premiers pas de l'utilisateur à travers les mécanismes de génération d'applications simples comprenant la notion d'héritage. La deuxième partie de cette documentation (**RMG Reference Manual**) contient les références à toutes les classes de base de RMG et à quelques applications qui font partie de RMG. Elle est dédiée à un concepteur plus expérimenté. La troisième partie (**Behind the Scenes of RMG**) s'adresse à des programmeurs avertis qui trouveront là des mécanismes de RMG plus fins. La documentation ultime, bien sûr, est contenue dans le code source des classes qui sont distribuées comme partie du bagage RMG.

Les fonctionnalités multi-fenêtrages de RMG sont, dans des contextes plus modernes, réalisées par des outils du type X-WINDOW. Mais, c'est la rapidité

des affichages graphiques qui fait la force de RMG. Ceux-ci s'effectuent à des vitesses nettement supérieures. Il semble qu'actuellement il soit difficile de faire cohabiter rapidité d'affichage et utilisation du multi-fenêtrage [MARI91].

3. Premières évaluations liées à la conception et à l'utilisation

Les applications développées jusqu'à aujourd'hui ont permis de dégager quelques observations intéressantes en considérant, tour à tour, l'usage du produit par un utilisateur apprenant et par un enseignant-réalisateur. Cette analyse s'est efforcée de préciser les attentes des uns et des autres [BONN90].

3.1. L'apprenant dans l'environnement RMG

L'apprenant utilise des produits développés à son intention afin de l'aider à comprendre des notions difficiles à appréhender avec les méthodes traditionnelles d'enseignement. Il s'agit d'un complément à l'enseignement théorique et pratique habituellement utilisé dans les sciences fondamentales. L'approche pédagogique est double : soit l'enseignant donne cours en illustrant son exposé par une démonstration devant plusieurs apprenants, soit un utilisateur seul lance une simulation. Dans ce dernier cas, il importe de posséder une interface conviviale afin de pouvoir mettre à profit la simulation sans frustration.

3.2. Le concepteur-réalisateur dans le contexte RMG

L'application RMG peut être écrite via deux grandes voies. La première est une programmation traditionnelle faite sous environnement UNIX en utilisant la chaîne classique : édition de texte source (vi, emacs, editeur RMG), compilation-édition de liens automatisée (appels du préprocesseur, du compilateur et de l'éditeur de liens) conduisant à l'intégration de la nouvelle application dans l'environnement.

La seconde manière d'écrire une application est une programmation "graphique" réalisée par l'emploi d'outils spécifiques à RMG. Cette méthode consiste à écrire des scénarii sous forme de diagramme de flux. Ce diagramme contrôle les objets définissant la scène. Le contrôle se réalise par un ensemble de lois ou d'équations spécifiant l'action sur l'objet. Nous pouvons illustrer très schématiquement cette programmation par la figure 1.1. Le concepteur veut dessiner un rectangle. Grâce aux outils mis à sa disposition par RMG, il écrit les trois messages qui apparaissent dans les rectangles arrondis. Il relie

les trois messages et le résultat apparaît (rectangle noir). Si nous analysons le deuxième message, nous constatons que l'utilisateur a simplement donné :

- les coordonnées du coin inférieur gauche (**origin**) et la taille (**extent**) du rectangle,
- la fenêtre (**superview**) dans laquelle il s'insère (la fenêtre est identifiée par son adresse),
- la couleur de fond (**bkgd**) du rectangle.

Ceci constitue un exemple simple d'utilisation **[FAZA89]**. Néanmoins, par cette méthode, le concepteur se familiarise avec l'outil.

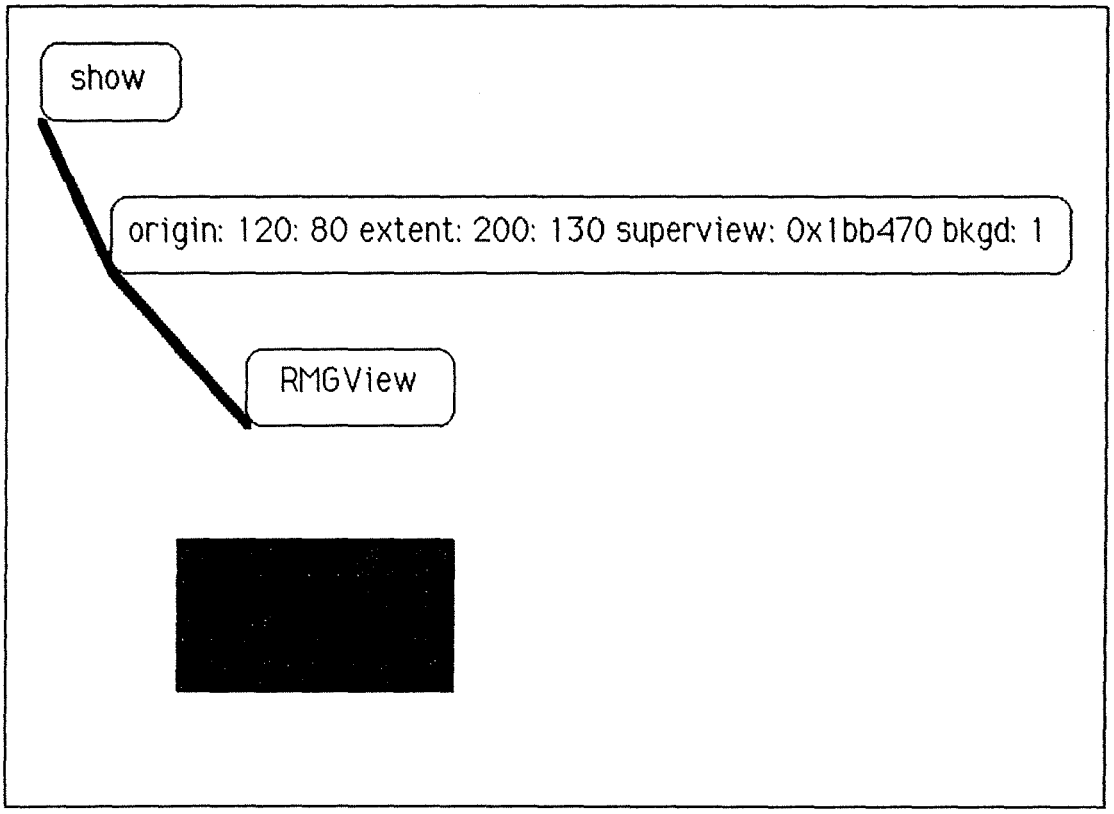


Figure 1.1 : programmation graphique sous RMG.

Le concepteur manipule directement les objets de la scène. Il peut donc valider progressivement son programme, ce qui constitue un des intérêts majeurs des outils graphiques. Pour obtenir le produit final, cette méthode de programmation offre la possibilité de générer du code source Objective-C, recompilable, qui peut être édité et complété dans un schéma de programmation traditionnel. On peut voir cette facilité offerte par RMG comme un atout de la programmation "graphique". Néanmoins, la programmation "graphique" n'est pas suffisamment puissante que pour construire totalement une application de simulation scientifique.

3.3. Conclusion tirée de l'évaluation de RMG

Les travaux réalisés par les différentes équipes du projet COLOS permettent de tirer au stade actuel, les conclusions suivantes, mettant l'accent sur les points forts de la plate-forme de développement RMG :

- l'utilisateur, qu'il soit enseignant ou apprenant, a la possibilité de constituer des "mini-mondes" associant de manière fonctionnelle des simulations différentes ;
- il est aisé de réaliser des interfaces interactives et conviviales entre l'utilisateur et la machine supportant la simulation ;
- les domaines de variation des paramètres manipulés par l'utilisateur sont contrôlés à priori pour éviter l'introduction de données aberrantes ;
- l'apprenant n'est soumis à aucune contrainte par la simulation, il est libre d'activer les actions quels qu'en soient les effets visibles ou invisibles ;
- la mise en oeuvre efficace des mécanismes d'héritage assure une réutilisabilité aisée des objets logiciels créés.

Les points faibles de RMG qui ont pu être cernés sont les suivants :

- malgré la facilité potentielle de réaliser une petite application par le biais du "prototypage", la non disponibilité de certaines structures de données et de certains outils mathématiques limite encore les possibilités de RMG dans son état actuel. Le "prototypage" permet de familiariser le concepteur avec les outils disponibles et lui permet de réaliser une application simple. Lorsqu'il s'attaque à un problème de plus grande envergure, il n'est plus possible de se limiter au "prototypage".
- la connaissance de la bibliothèque des classes est longue à acquérir, comme dans toute programmation orientée objets, et pourtant elle est indispensable au concepteur-réalisateur.

4. Activités de l'équipe des FUNDP dans le cadre du projet COLOS

Les activités de l'équipe universitaire des facultés namuroises, attachée au projet COLOS, ont essentiellement porté sur l'enseignement des télécommunications dans le cadre du modèle ISO (**Interconnexion de Systèmes Ouverts**) de l'OSI (**Organisation de Standardisation Internationale**).

4.1. Développements actuels

Les applications en cours de conception sont relatives aux notions de connexion et de déconnexion de systèmes ouverts. Le but pédagogique poursuivi est la simulation d'une ouverture (respectivement fermeture) d'une connexion

entre deux entités paires au sein d'une couche du modèle ISO [HENS88] [TANN81] [TOLH88].L'apprenant visualise les étapes nécessaires à l'accomplissement du mécanisme invoqué (connexion ou déconnexion).

Les cours destinés à l'approche du modèle ISO gagnent à s'appuyer sur des simulations lorsque les phénomènes sont difficiles à percevoir. A titre d'exemple, nous pouvons dire qu'un étudiant perçoit mieux le phénomène d'enchaînement des primitives sur un logiciel de simulation que sur un transparent ou un tableau encombré par le nombre de flèches, de commentaires,... De plus, une simulation apporte une autre dimension à l'exposé. La variable temps prend toute son importance, la dynamique du phénomène apparaît alors qu'elle était gelée sur un transparent.

Dans l'état actuel de nos travaux, nous veillons essentiellement à acquérir la maîtrise de l'outil RMG. D'autre part l'acquisition de connaissances d'un niveau scientifique dans le domaine des télécommunications et de l'enseignement assisté par ordinateur s'avère essentielle avant d'envisager la construction d'applications réellement valables.

4.2. Développements futurs

L'équipe COLOS de Namur a défini un développement modulaire qui devrait conduire à la conception d'un logiciel ou d'un ensemble de logiciels adaptés à l'enseignement des télécommunications. Nous décrivons ce programme ci-dessous.

Comme nous l'avons déjà dit, la première étape concernant la connexion et la déconnexion de deux entités paires au sein d'un système ouvert est en cours de réalisation.

L'étape suivante consistera à illustrer le multiplexage de plusieurs connexions de niveau "n" sur une connexion de niveau "n-1".

La notion d'unité de protocole de données (PDU : **Protocol Data Unit**) sera abordée en deux étapes. La première consistera à illustrer le passage d'un PDU d'une entité émettrice vers l'entité paire réceptrice. Ici, l'unité de protocole de données sera vue comme une boîte noire. La seconde étape se voudra plus détaillée, elle ôtera le voile précédemment placé sur le PDU. A ce stade, on s'attachera à expliquer le contenu des unités de protocole de données dans le détail.

Par la suite, l'équipe COLOS se propose d'étudier le fonctionnement dans le détail des couches 2, 3 et 4 du modèle ISO. Les notions d'**Abort**, de **Restart** et de **Reset** pourront être mises en évidence dans le cadre particulier de l'étude de ces couches. L'étude du niveau 3 et la notion de routage seront certainement largement développées. Les techniques de commutation par paquets (**X25**, **FAST PACKET**) seront illustrées.

Lorsqu'une bonne compréhension des couches basses du modèle d'un point de vue scientifique et pédagogique sera atteinte, les couches session et présentation seront abordées respectivement. Les logiciels de simulation seront orientés vers l'explication des services offerts par ces couches. Jusqu'à ce stade de développement, les paramètres véhiculés par les primitives ne figurent pas encore explicitement. L'association des paramètres des primitives lors de la demande d'un service d'une couche à une autre représente une étape supplémentaire dans l'évolution d'un programme de simulation orienté sur le modèle ISO.

A plus ou moins long terme, l'équipe namuroise s'est fixé pour objectif le développement d'outils pédagogiques destinés à l'enseignement des sujets suivants :

- CASE (**Common Application Service Elements**) : ces éléments de service offrent des standards utilisés par la plupart des applications. Citons l'ACSE (**Association Control Service Elements**) qui permet de créer une association et de la rompre (proprement ou pas), le RTSE (**Reliable Transfer Service Elements**) qui offrent à son utilisateur un transfert fiable des données, le ROSE (**Remote Operation Service Elements**) qui permet la définition et l'envoi d'unités de protocole de données propres à une application et le CCR (**Commitment, Concurrency and Recovery**). Le service réalisé par le CCR assure la terminaison avec succès d'activités étendues sur des systèmes ouverts. Le concept d'action atomique est central dans le service CCR. Une action atomique est composée d'une série d'opérations qui sont effectuées sur le système ouvert. Si une erreur survient durant l'exécution de l'action atomique, la partie de l'action atomique qui a échouée peut être réexécutée. Si l'erreur ne peut être corrigée, toute l'action atomique échoue et on retourne dans l'état qui précédait le lancement de l'action. Donc, une action atomique est effectuée en tout avec succès ou pas du tout [TOLH88].
- X400 : X400 est une norme du CCITT (Comité Consultatif International pour la Télégraphie et la Téléphonie) consacrée à la messagerie électronique. Cette norme régit l'échange de textes entre deux systèmes éloignés (MOTIS : **Message Oriented Text Interchange System**).

- LAN (**Local Area Network**) : ce qui importe dans les réseaux locaux d'un point de vue pédagogique, c'est la manière dont les couches 2 et 3 sont réalisées. C'est dans cette direction que porteront les recherches.
- l'interconnexion de réseaux.
- la gestion de réseaux.
- RNIS : le RNIS (Réseau Numérique à Intégration de Services) est une nouvelle facilité offerte par la RTT (Régie des Télégraphes et Téléphones).
- les terminaux virtuels (**virtual terminals**) [HENS88] : actuellement, il existe un large choix de terminaux disponibles, et parmi eux, différentes méthodes sont utilisées pour contrôler l'écran. Si un terminal "virtuel" standard était disponible sous ISO, le problème de l'accès à partir de terminaux éloignés à des applications ou des processus sur un gros système serait largement simplifié. C'est ce que fait le terminal virtuel : il présente à une extrémité d'un système ouvert un type de terminal simple. De l'autre côté (terminal éloigné), il fait correspondre les caractéristiques de ce terminal virtuel aux caractéristiques du terminal réel impliqué.
- JTM (**Job Tansfer and Manipulation** [HENS88] [TOLH88]) : ce service supporte l'initiation d'un "job" et le transfert d'information associé à ce "job" quand et où c'est nécessaire. Cela se réalise sans perte ni duplication. JTM fournit le contrôle des fonctions associées à un "job" et permet de modifier l'exécution d'un "job". Notons que le terme "job" utilisé dans ISO n'est pas nécessairement équivalent à celui traditionnellement utilisé. JTM permet le passage d'informations vers un système destinataire pour lui permettre d'exécuter un processus mais JTM ne s'intéresse pas au processus actuel. Un "job" traditionnel consiste en données et instructions à exécuter. JTM supporte le transport des données et instructions, mais pas leur exécution.

Pour notre part, nous avons examiné l'enseignement de FTAM (**File Transfer, Access and Management**). Au départ la réalisation de cette application devait se faire après l'étude du CASE. Cependant, nous avons estimé qu'aborder un sujet plus vaste permettrait d'amener de nouvelles idées sur la manière d'appréhender l'élaboration d'un logiciel d'EAO. Ceci nous a amené à parcourir toutes les étapes de la conception d'un logiciel pédagogique. Il a fallu éclaircir le large domaine FTAM afin de trouver ce qui ferait l'objet d'un enseignement. Ensuite, nous avons précisé par quels moyens visuels nous allions faire passer le message. L'étape suivante nous a conduits à la spécification du scénario FTAM. Le moment de l'implémentation avec RMG est venu. Et pour terminer, nous avons analysé le travail dans son ensemble. Tout au long du travail, nous avons dû sans cesse nous poser des questions quant à la qualité pédagogique, à la pertinence vis-à-vis des normes. Les discussions avec les autres membres de l'équipe COLOS nous ont été fort profitables.

Nous nous proposons de donner une liste non exhaustive de problèmes, de situations, qui pourraient être à la base du développement de logiciels à usage

pédagogique. Cette liste est le résultat de discussions avec des personnes concernées (étudiants et enseignants) et de notre expérience personnelle.

La notion de PDU est un point central du modèle ISO. Qui fabrique le PDU? Comment arrive-t-il à son entité paire? Que signifie 'encapsulage' de PDU? Ce sont là quelques exemples des questions qui se posent. Sur base de deux systèmes ouverts décomposés en couches et reliés par un réseau, il serait possible d'illustrer l'échange de protocole de données. Sans rentrer dans le détail du contenu du PDU, mais en le considérant comme une donnée globale, il est possible de montrer son passage par le chemin physique (on retrouve ici la notion de connexion) emprunté via les couches inférieures.

Les principes du modèle ISO eux-mêmes méritent une étude. Comment une couche communique-t-elle avec son entité paire ? Il faut envisager la notion de services offerts par une couche à sa couche supérieure. Comment l'enchaînement de ses services permet-il d'assurer les demandes de l'utilisateur ? Il faut considérer une vision globale du modèle et examiner l'enchaînement des primitives de services et la constitution des PDU. Deux aspects se dégagent. Une vision globale mais sommaire du modèle ou une vision limitée mais détaillée d'une couche particulière.

Le comportement de la couche session n'est pas évident au premier abord. Cela est dû en partie aux nombreuses unités fonctionnelles qu'elle est susceptible d'offrir. De plus, les services qu'elle rend aux couches supérieures sont fortement orientés sur les problèmes liés à l'activité d'une application. Les requêtes de la couche présentation sont généralement des primitives initialisées par la couche application et qu'elle se contente de transmettre à l'entité session. Dans ce cas aussi, il nous semble opportun de mettre au point une simulation du travail effectué par cette entité. En outre, les notions de point de synchronisation et de jeton pourraient être éclaircies à la lumière d'un outil de simulation.

Les notions de multiplexage et de segmentation sont particulièrement propices à une démonstration graphique. Le multiplexage peut être simulé par plusieurs tuyaux aboutissant tous dans le même canal de communication. Nous pouvons citer, à titre d'exemple, le multiplexage de plusieurs connexions réseau qui utilisent la même liaison de données. En X25, chaque connexion utilise une voie d'accès logique identifiée par un numéro d'accès logique. La connaissance de ce numéro permet d'aiguiller correctement les données. En ce qui concerne la segmentation, il faut représenter la décomposition d'un PDU en plusieurs morceaux. Ceux-ci seront transmis un par un grâce aux services offerts par la

couche inférieure. Les morceaux seront ensuite réassemblés dans l'entité réceptrice.

Chapitre 2 : FTAM

1. Objectif du chapitre

Ce chapitre est consacré à la description de la norme FTAM (File Transfer, Access and Management) de l'ISO. Après avoir considéré les principes fondamentaux, nous aborderons plus en détails certains problèmes que nous envisagerons d'un point de vue pédagogique dans les chapitres ultérieurs. Cette description de FTAM se base essentiellement sur les ouvrages [HENS88] [ISO8571] [STAL87].

2. Introduction

La norme FTAM est décrite dans un document en 4 parties, ISO 8571. Ces quatres parties concernent la description de la norme (ISO 8571/1) , la définition de la notion de "virtual filestore" que nous traduirons par "système de fichiers virtuels" (ISO 8571/2), la définition du service de fichiers (ISO 8571/3) et la spécification du protocole de fichier (ISO 8571/4). Nous faisons allusion ici à la norme FTAM publiée en août 1986.

Le File Transfer, Access and Management s'occupe du transfert, de l'accès et de la gestion de fichiers entre deux systèmes ouverts. Le transfert de fichier concerne l'écriture, la lecture ou la modification en tout ou en partie d'un fichier éloigné. L'accès au fichier permet la lecture, l'insertion, le remplacement, l'extension, la suppression des articles au sein de celui-ci. Enfin, la gestion s'occupe de l'écriture, de la lecture ou de la modification des attributs d'un fichier. La norme ne fait pas référence à la gestion de la localisation des fichiers (gestion des répertoires).

3. Le système de fichiers virtuels

Avant d'aborder le fonctionnement de FTAM, examinons la notion de système de fichiers virtuels (virtual filestore).

3.1. L'utilité du système de fichiers virtuels

La manière de stocker les fichiers varie considérablement de machine à machine mais nous pouvons dégager un certain nombre de caractéristiques communes à tous les fichiers. Si nous ignorons la manière dont le fichier est physiquement organisé sur disque, nous savons par contre que ce fichier contient des articles ainsi que des mécanismes pour accéder à ces articles

(clés d'accès). C'est ce niveau d'abstraction qui est utilisé dans FTAM de manière à proposer une vision commune d'un fichier à tout type de machine. Le service de fichiers travaillera donc sur une représentation virtuelle du fichier. C'est le gérant de fichiers qui, recevant les actions à entreprendre, établira le parallèle entre la représentation virtuelle et la représentation réelle du fichier. La description du système de fichiers donnée par la norme est très large de manière à englober un maximum de représentations possibles de structures de données. Dans l'état actuel des choses, seules les structures de type hiérarchique (stricte) sont représentables. Il est prévu des extensions/révisions pour couvrir des modèles tels que les organisations en réseaux, les schémas relationnels,... Nous allons examiner les caractéristiques du fichier dans ce contexte.

3.2. Les caractéristiques du fichier virtuel

Le fichier contient une structure et des données. La structure la plus générale considérée par FTAM est une structure d'accès hiérarchique possédant les propriétés suivantes :

- la structure d'accès est un arbre.
- chaque noeud de l'arbre est lié à zéro ou une unité de données contenant les données stockées dans le fichier (DU - **data unit**).
- chaque noeud permet d'accéder à son sous-arbre, l'ensemble constitue un FADU (file access data unit).
- chaque noeud peut avoir un nom.
- le nombre de niveaux, le nombre d'arcs partant d'un noeud et la longueur des arcs ne sont pas limités. La longueur est un nombre entier attribué à l'arc de manière non stipulée par la norme. La longueur permet de localiser le niveau d'un noeud dans la structure de façon relative au niveau de son noeud père.

Les unités de données peuvent contenir un ou plusieurs éléments de données (DE - **data element**) qui représentent les articles.

Ces concepts sont repris dans la figure suivante (fig. 2.1). R, A, B , C, D, E, F sont des noeuds. A et D n'ont pas d'unité de données associée. Chaque noeud définit un sous-arbre ou FADU. De plus nous reprendrons les définitions courantes d'un arbre. Nous dirons par exemple que B et C sont les noeuds fils de A ; A est le noeud père de B et C ; B et C sont des noeuds terminaux (c-à-d qu'ils n'ont pas de noeud fils) ; R est le noeud racine (c-à-d qu'il n'a pas de père).

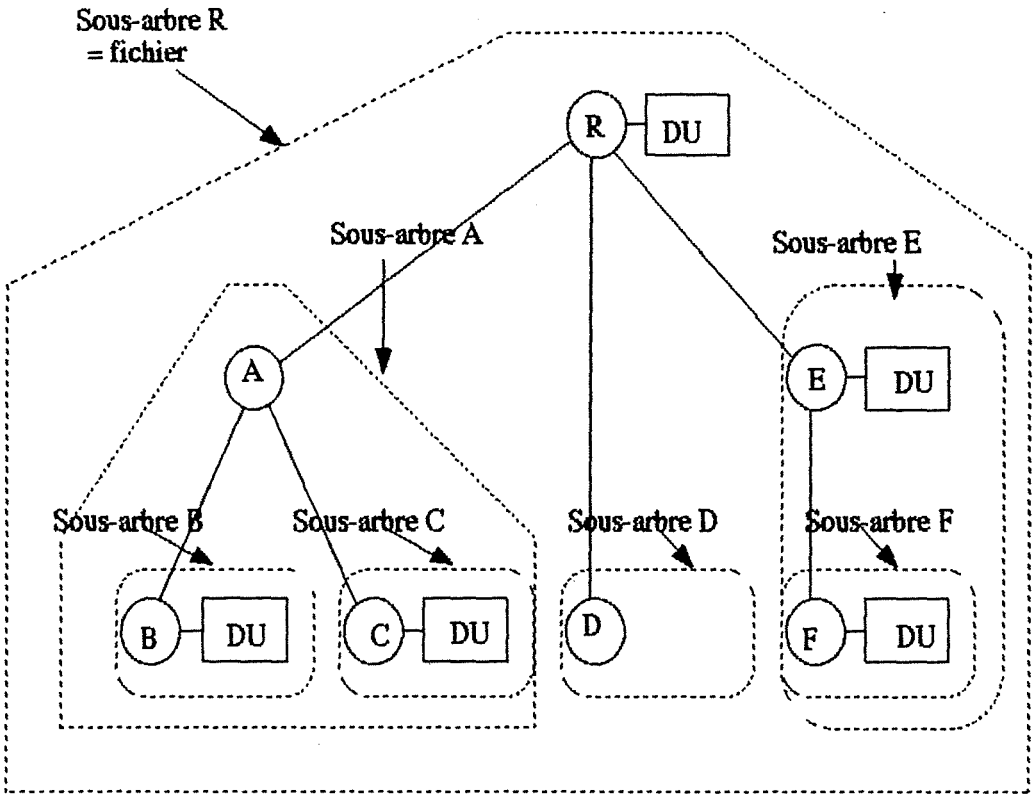


Figure 2.1 : la structure du fichier virtuel.

Les informations sur un noeud sont :

- le nom du noeud;
- la présence ou non d'une unité de données attachée au noeud;
- la manière dont il est relié à ces noeuds fils.

FTAM regroupe plusieurs ensembles de contraintes qui définissent un sous ensemble de structures de représentation valables pour le fichier. Par exemple un fichier séquentiel ne nécessite pas la définition d'une structure aussi complexe que celle de la figure 2.1. On limitera la structure à un seul niveau. Les différentes structures envisagées sont :

- non structuré (unstructured) : cette structure modélise les fichiers pouvant être accédés comme un tout. Ces fichiers peuvent également être allongés en bloc. La figure 2.2 modélise cette structure. Tous les éléments de données sont enregistrés dans l'unique unité de données attachée au noeud racine.

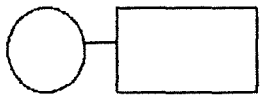


Figure 2.2: représentation du fichier non structuré.

- structure séquentielle plate (sequential flat) : la racine est un noeud sans DU, tous les noeuds fils sont terminaux et comportent un DU. On peut accéder à un DU particulier grâce à sa position (relatif cobol, accès calculé). Sur la figure 2.3 nous avons représenté 3 noeuds fils avec leur DU associé. Il n'y a qu'un seul niveau ; tous les arcs sont de longueur 1.

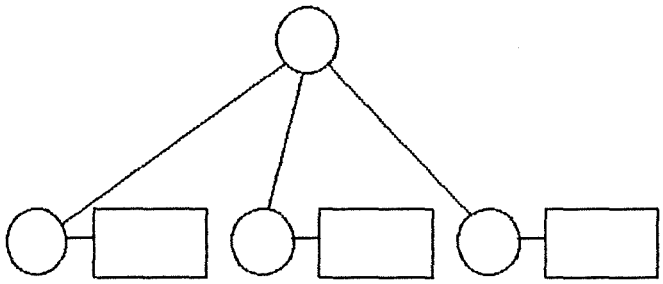


Figure 2.3 : structure séquentielle plate.

- structure ordonnée plate (ordered flat) : elle ressemble à la précédente mais ici, on peut accéder à un noeud sur base d'une valeur de clé (nom attribué au noeud). Plusieurs noeuds peuvent avoir la même valeur de clé, ils sont rangés séquentiellement. Nous avons représenté cette structure sur la figure 2.4. On constate que la Clé2 apparaît deux fois. Nous supposons bien sûr qu'une relation d'ordre est définie sur les valeurs de clés. Les noeuds sont ordonnés sur leur valeur de clé.

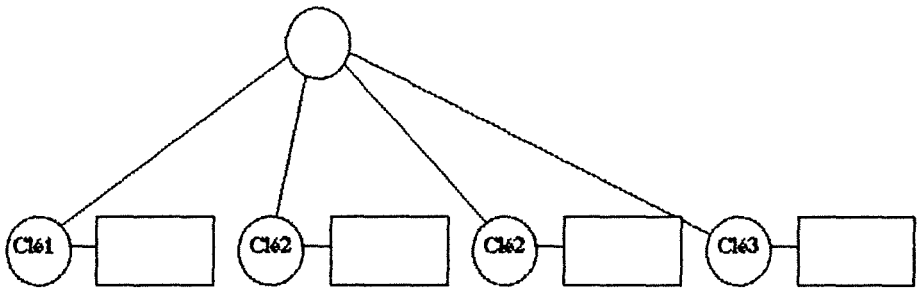


Figure 2.4 : structure ordonnée plate.

- structure ordonnée plate avec noms uniques (ordered flat with unique names) : elle possède les mêmes caractéristiques que l'ordonnée plate à l'exception que deux noeuds ne peuvent avoir le même nom. La structure présentée à la figure 2.5 est équivalente à la précédente (fig. 2.4) à l'exception que deux valeurs de clés sont obligatoirement distinctes.

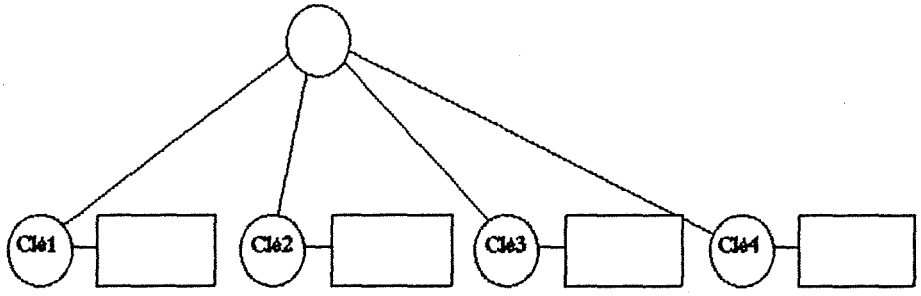


Figure 2.5 : structure ordonnée plate avec noms uniques.

- structure ordonnée hiérarchique (ordered hierarchical) : cette structure reprend les caractéristiques de l'organisation hiérarchique. Les noeuds fils sont organisés sur base de leur nom. Sur la figure 2.6, nous pouvons constater que ce type de structure comprend plusieurs niveaux. Les noeuds fils sont ordonnés sur leur valeur de clé. Plusieurs noeuds fils d'un même père peuvent avoir la même valeur de clé.

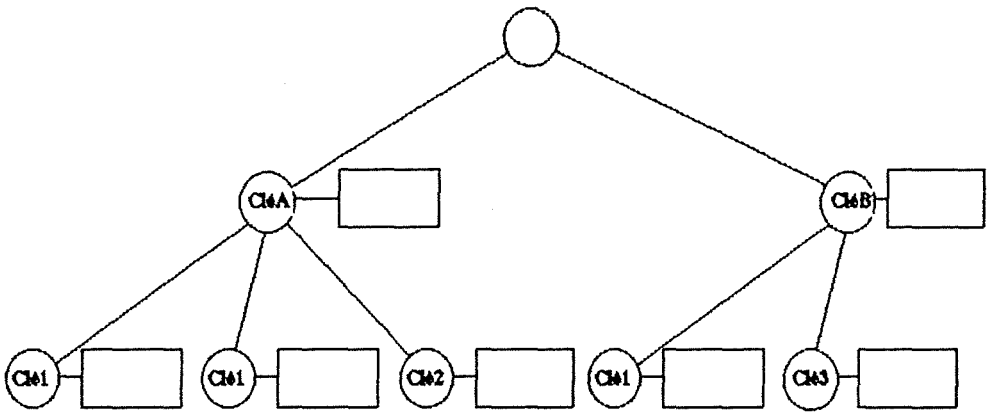


Figure 2.6 : structure ordonnée hiérarchique.

- structure hiérarchique générale (general hierarchical) : elle utilise la définition de fichier virtuel sans contrainte. La figure 2.7 donne un exemple de cette structure. Aucune contrainte n'est imposée sur les noms des noeuds ; de ce fait, nous les avons volontairement omis.

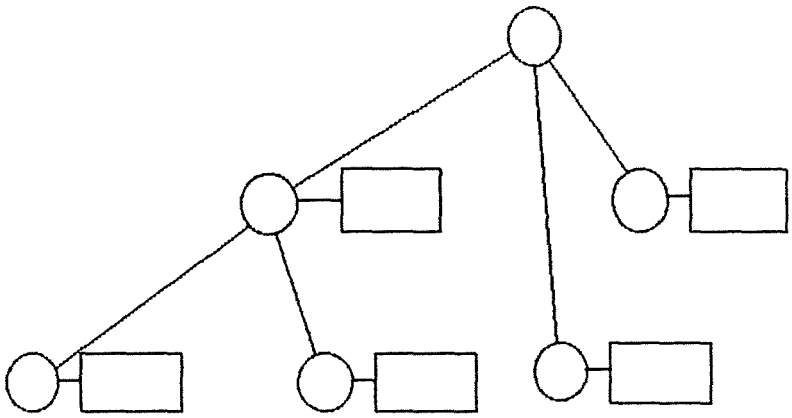


Figure 2.7 : structure hiérarchique générale.

- structure hiérarchique générale avec noms uniques (general hierarchical with unique names) : la seule contrainte concerne le nom des noeuds. Les noms des noeuds fils de tout parent donné sont uniques. Nous donnons un exemple de cette structure à la figure 2.8. Nous constatons que plusieurs noeuds peuvent avoir le même nom (en l'occurrence "Clé1") mais ils sont fils de pères différents.

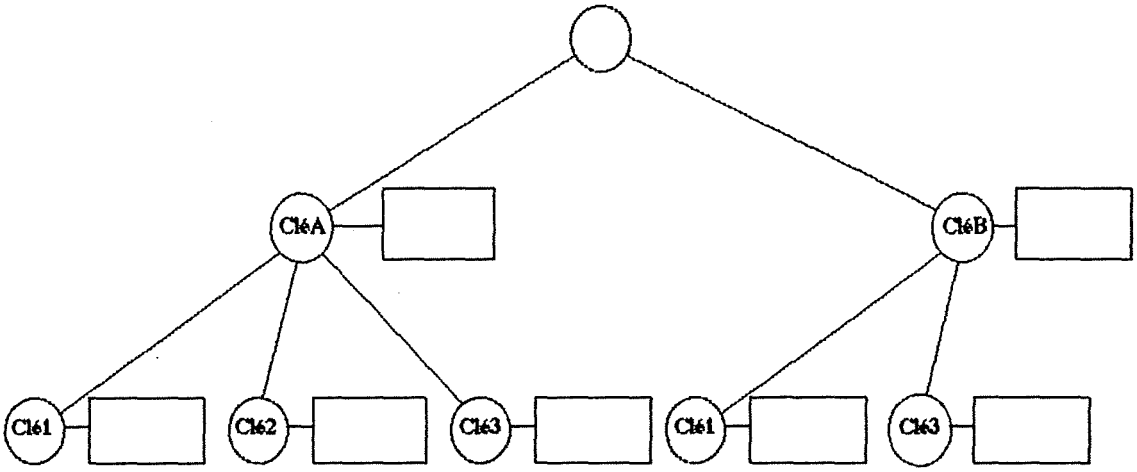


Figure 2.8 : structure hiérarchique générale avec noms uniques.

A titre d'exemple, la structure de la figure 2.10 décrit ce que pourrait être la modélisation du fichier séquentiel indexé de la figure 2.9. Les pages où sont stockés les articles se trouvent au niveau des unités de données. Les niveaux supérieurs regroupent les différents niveaux d'index. Les noms des noeuds représentent les valeurs d'index. Il s'agit d'une structure hiérarchique générale avec noms uniques. Remarquons que le standard de base impose relativement peu de contraintes et le choix de la représentation d'un fichier apparaît comme arbitraire. Pour pouvoir travailler sur le système éloigné, il faut être au

courant des conventions de représentation adoptées pour un fichier séquentiel indexé.

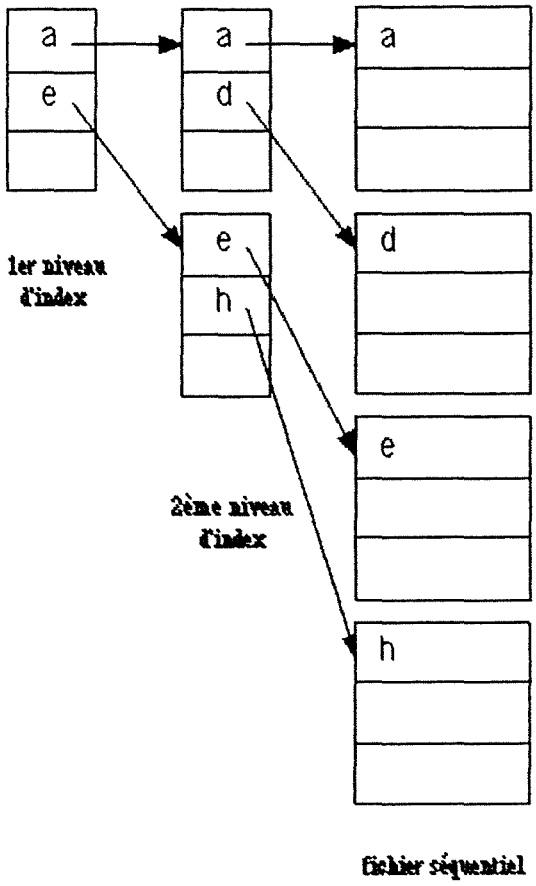


Figure 2.9 : exemple de fichier séquentiel indexé.

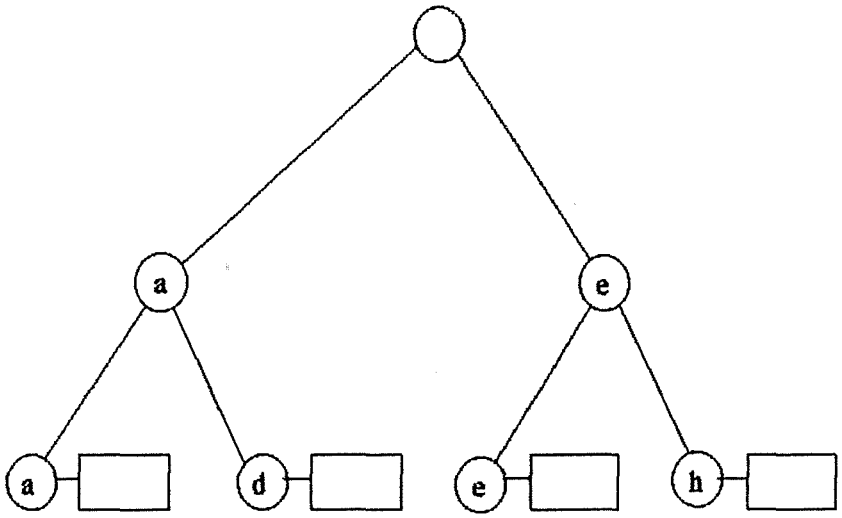


Figure 2.10 : modélisation du fichier séquentiel indexé.

Un autre exemple illustre comment nous imaginons la représentation d'une base de données. La figure 2.11 décrit un fichier comportant des articles client sur lesquels existent deux clés d'accès : l'une qui identifie l'article univoquement (l'item nr_cli qui représente le numéro du client) l'autre qui identifie une séquence d'articles (l'item région qui permet d'accéder aux clients d'une région déterminée). Nous ne pouvons pas modéliser directement un type d'article sur lequel il existe plusieurs clés d'accès. En effet, un noeud fils serait rattaché à plusieurs noeuds pères jouant les rôles des différentes clés d'accès. Néanmoins après la première transformation de schéma (fig. 2.12), cela devient possible **[HAIN86]**. Nous avons transformé l'item région en un type d'article de manière à transformer la clé d'accès en chemin d'accès. Par une deuxième transformation, nous enlevons le chemin d'accès de manière à isoler les deux types d'articles (fig 2.13).

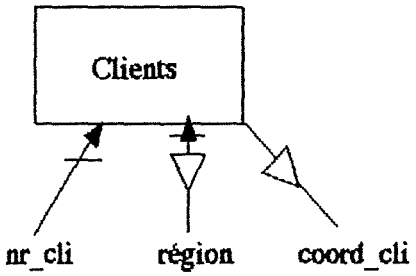


Figure 2.11 : exemple de base de données.

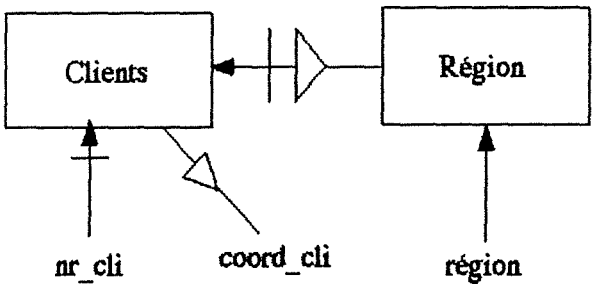


Figure 2.12 : premier schéma transformé.

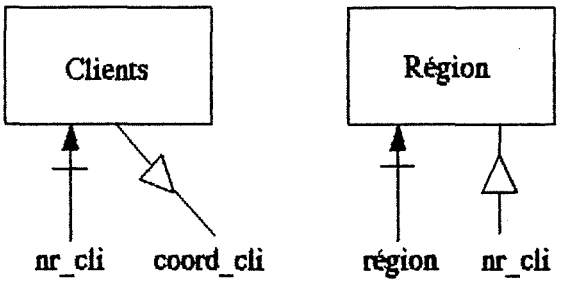


Figure 2.13 : deuxième schéma transformé.

Nous avons choisi de représenter trois régions identifiées par R1,R2,R3 et quatres clients identifiés par 1,2,3 et 4. Les clients 1 et 2 appartiennent à la région R1. Les clients 3 et 4 appartiennent à la région R2. Aucun client n'est enregistré dans la région R3.

Une représentation inspirée (-inspirée- et **non** systématiquement déduite) du premier schéma transformé est proposée à la figure 2.14. La clé d'accès sur les articles région est représentée par l'identifiant des noeuds R1, R2, R3. Aucun élément de données n'est associé car l'article région n'a pas d'autre item que son identifiant. La clé d'accès aux articles clients est représentée par les noeuds numérotés 1, 2, 3 et 4. Le chemin d'accès entre Région et Clients est représenté par les arcs liant les noeuds R1, R2, R3 aux noeuds 1,2,3 et 4. Les DU associés aux clients représentent les renseignements divers enregistrés sur les clients (coord_cli).

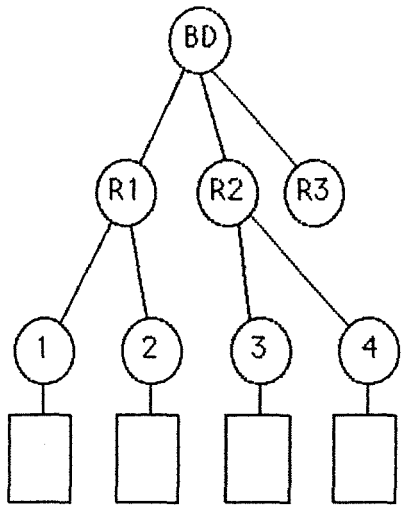


Figure 2.14 : modélisation du premier schéma transformé.

Pour le second schéma transformé nous imaginons la représentation suivante. Les types d'articles Clients et Région sont séparés (sous arbre Cli et sous arbre Rég). Chaque article région est identifié par les noeuds R1, R2, ou R3. Chaque région comporte l'ensemble des identifiants primaires (nr_cli) des articles Clients (fig. 2.15). Ces exemples illustrent bien les potentialités de modélisation de FTAM. Faisons remarquer que nous sommes parvenus à modéliser une structure relationnelle(au travers d'une structure hiérarchique générale avec noms uniques). Si la norme FTAM ne permet pas de modéliser une structure relationnelle existante, rien ne nous empêche en tant que concepteur de l'application d'adopter des conventions de représentation qui achèvent ce but. La limitation réside dans le fait que la représentation est fixée par et pour l'application et pas par la norme dans un contexte plus général.

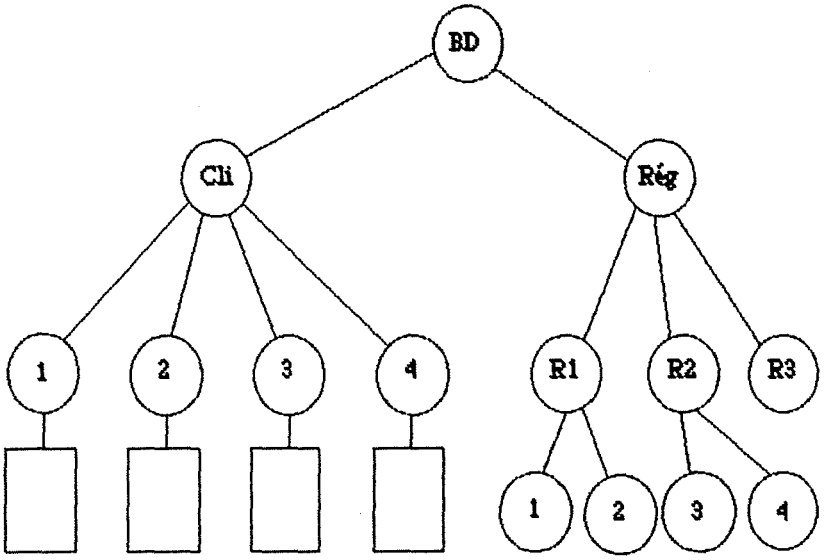


Figure 2.15 : modélisation du second schéma transformé.

Le fichier est en outre caractérisé par une série d'attributs regroupés logiquement. Un certain nombre d'attributs décrivent les caractéristiques du fichier dans l'organisation du gestionnaire de fichier. La liste de ces attributs est la suivante (liste des attributs de fichier) :

- le nom du fichier.
- le type du contenu : ce type décrit les syntaxes abstraites utilisées pour représenter la structure comme le contenu. Ceci peut évidemment être très complexe, aussi permet-on la définition d'un "**document type**" qui regroupe toutes les définitions (exemple : définition des syntaxes abstraites, définition de "record" qui utilisent plusieurs types de syntaxes abstraites pour définir une syntaxe propre au document). Le type du contenu fait alors simplement allusion à ce document plutôt que de lister toutes les syntaxes

abstraites. Les "**document types**" jouent un rôle important dans FTAM. Les principales organisations de fichiers sont reprises dans ces documents. Ils contribuent à la définition de profils FTAM au niveau européen et mondial.

- la comptabilité de stockage : elle identifie l'autorité responsable de la comptabilité.
- la date et l'heure de création.
- la date et l'heure de dernière modification.
- la date et l'heure de dernière lecture.
- la date et l'heure du dernier changement d'attribut.
- l'identité du créateur.
- l'identité du dernier modificateur.
- l'identité du dernier lecteur.
- l'identité du dernier modificateur d'attribut.
- la disponibilité du fichier : elle indique si le fichier est immédiatement disponible après création ou si cette disponibilité est différée.
- les actions permises : elles reprennent la liste des actions permises sur le fichier (lire, insérer, remplacer, étendre, effacer un FADU, lire les attributs, changer les attributs, effacer le fichier). Elles concernent également la manière dont les FADU vont pouvoir être accédés (voir le point 4.1.3).
- la taille du fichier.
- la taille future du fichier : elle détermine la taille maximale d'extension du fichier en octets lors de sa création. Si on essaye d'étendre le fichier au-delà de la taille fixée, le système peut autoriser ou refuser une modification de la taille maximale.
- le contrôle d'accès : il reprend la liste des accès autorisés pour chaque utilisateur, éventuellement avec mot de passe.
- nom de cryptage : il permet de définir un transfert crypté du fichier. Ceci n'est possible que sur les fichiers binaires non structurés.
- qualifications légales : elles indiquent le statut légal du fichier et de son utilisation.
- usage privé : cet attribut est réservé pour une utilisation éventuelle non précisée par la norme.

D'autres attributs permettent de gérer le fichier lorsqu'il est effectivement accédé par un ou plusieurs utilisateurs (liste des attributs d'activité). Il faut tout d'abord considérer les attributs dupliqués pour chaque utilisateur qui décrivent l'état du fichier pour chacun d'eux :

- demande d'accès courante : elle reprend une liste des actions permises actuellement (étendre un DU, effacer un FADU, ...).
- localisation courante : elle identifie le FADU sur lequel l'utilisateur est positionné (noeud racine, deuxième noeud dans le parcours de préordre, ...).
- contexte d'accès courant : il désigne la manière dont la structure est actuellement accédée (lecture, écriture, localisation d'un FADU) (voir le point 4.1).

- mode de traitement courant : il reprend la liste des actions permises dans le contexte d'accès courant.
- libellé de l'entité application courante.
- compte courant : il fait référence à l'autorité responsable de la comptabilité de l'action en cours.
- contrôle de concurrence courant : il désigne le mode d'accès au fichier lorsque plusieurs utilisateurs travaillent simultanément sur celui-ci (les modes permis sont : partagé, exclusif, non requis, pas d'accès).
- identité de l'initiateur courant.
- mot de passe courant.

Ces attributs sont mis à jour au long de l'association FTAM.

Des attributs uniques sont attachés au fichier lorsque celui-ci est accédé :

- type du contenu actif.
- qualification légale active.

4. Le service de fichiers

Maintenant que nous savons sur quelles structures FTAM peut travailler, nous pouvons aborder le problème de l'accès, du transfert et de la manipulation de fichiers.

4.1. Les primitives du service de fichiers et les régimes

Dans cette partie nous allons décrire de manière conceptuelle comment les actions sont réalisées. Ensuite, nous examinerons comment cela se passe plus concrètement. Une liste des primitives FTAM complète est donnée en annexe 1.

Pour accéder à une donnée dans un fichier, l'utilisateur a une marche à suivre qui lui permettra d'accéder à des possibilités de manipulation de plus en plus étendues. Ces étapes le placent successivement dans des régimes où certaines actions seulement sont permises.

4.1.1. Le régime d'association FTAM

La première chose à faire est d'entrer en communication avec l'entité FTAM répondante. Pour y parvenir, l'utilisateur utilise la primitive F-INITIALIZE request qui, lorsqu'elle est confirmée positivement, le fait entrer dans le régime d'association FTAM (FTAM association regime). Il existe une association entre les deux entités paires qui peuvent dès lors commencer leur collaboration. En principe, dans ce mode, l'utilisateur devrait pouvoir

s'occuper, entre autre, de la gestion de l'ensemble du système de fichiers (la création, le retrait, le déplacement de répertoires,...), mais la norme ne précise rien actuellement à ce sujet.

4.1.2. Le régime de sélection de fichier

L'utilisateur peut ensuite passer dans un régime qui lui permet de travailler sur un fichier particulier. Il peut soit sélectionner un fichier existant par la primitive F-SELECT request, soit créer un nouveau fichier via la primitive F-CREATE request. Ces deux primitives sont confirmées avec possibilité de refus. Après acceptation, le régime de sélection de fichier (file selection regime) est établi. Dans ce mode, l'utilisateur peut gérer les attributs du fichier. Il peut lire les attributs via la primitive F-READ-ATTRIB request ou les modifier par la demande F-CHANGE-ATTRIB request. Le F-READ-ATTRIB confirmation contient la liste des attributs lus ainsi que leurs valeurs. Pour l'écriture des attributs, une liste similaire est contenue dans le F-CHANGE-ATTRIB request. Signalons, pour être complet, que tous les attributs ne sont pas modifiables ni accessibles, certains peuvent être protégés par mot de passe. L'utilisateur peut en outre décider de l'effacement complet du fichier par le biais de la primitive F-DELETE request. Après effacement, le régime de sélection de fichier se termine et le régime d'association FTAM est rétabli.

4.1.3. Le régime d'ouverture de fichier

S'il souhaite accéder aux données, l'utilisateur doit entrer dans un nouveau régime. Suite à l'utilisation de la primitive F-OPEN request qui est confirmée, l'utilisateur entre dans le régime d'ouverture de fichier (file open regime). La structure des données est alors accessible. L'utililisateur peut se positionner sur un noeud via l'utilisation de la primitive F-LOCATE request. Il peut aussi effacer un FADU par la primitive F-ERASE request. Précisons que l'utilisateur dispose, pour chaque structure définie par un ensemble de contraintes, de plusieurs moyens d'accéder au fichier. Il s'agit de la définition d'un contexte d'accès (voir ci-après). Ce contexte d'accès est négocié comme paramètre du F-OPEN. Le contexte d'accès permet de particulariser la connexion et les relations entre les entités FTAM communiquantes. FTAM fixe également un algorithme de parcours des noeuds appartenant au FADU sélectionné. Il s'agit d'un parcours de préordre de l'arbre . Si nous reprenons l'exemple du fichier séquentiel indexé de la figure 2.10, nous pouvons définir un contexte d'accès qui nous permet de parcourir séquentiellement les articles sans considérer l'index. Nous préciserons lequel après avoir examiné les différents contextes d'accès possibles :

- hiérarchique sur toutes les unités de données (Hierarchical all data units - HA) : tous les éléments de la structure et des données du FADU

sélectionné seront transférés. La figure 2.16 donne l'exemple du fichier sur lequel nous allons faire différents accès. Dans le contexte hiérarchique sur toutes les unités de données, toutes les informations concernant le fichier seront transférées. La figure 2.16 représente donc également le fichier transféré dans ce contexte pour autant que l'on se place sur le noeud 1 avant le transfert.

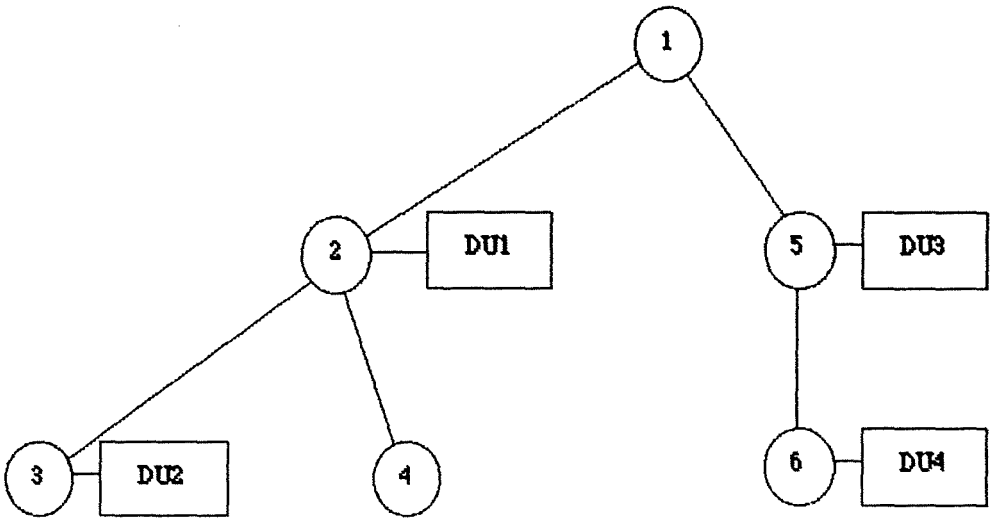


Figure 2.16 : structure transférée dans le contexte HA.

- hiérarchique sans les unités de données (Hierarchical no data units - HN) : tous les éléments descriptifs de la structure du FADU sélectionné seront transférés. La figure 2.17 montre ce que devient notre fichier exemple (fig. 2.16). Seule la structure est transférée, sans les unités de données. Nous considérons que la demande de transfert se fait à partir du noeud 1.

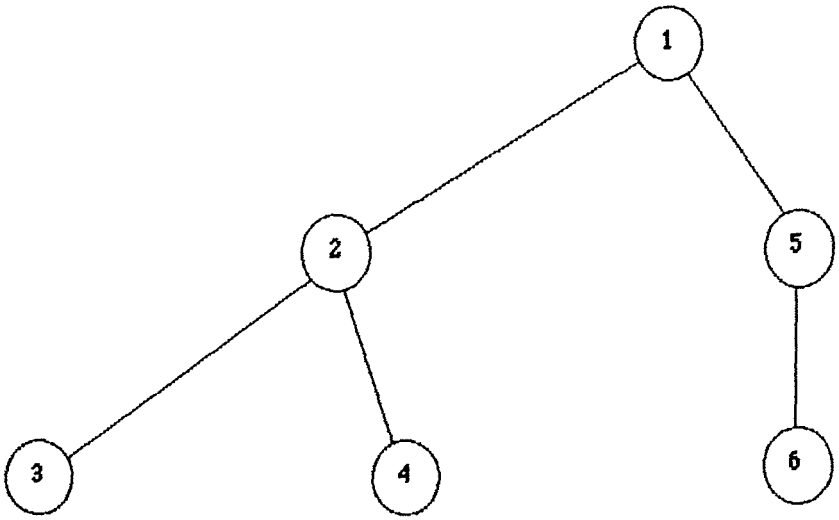


Figure 2.17 : structure transférée dans le contexte HN.

- plat sur toutes les unités de données (Flat all data units - FA) : les unités de données et les informations sur les noeuds où il existe une unité de données seront transférées pour le FADU référencé. Ces renseignements sont insuffisants pour reconstruire une structure hiérarchique. Dans ce contexte (fig. 2.18), on a perdu l'information sur la structure hiérarchique. Nous disposons des informations sur les noeuds possédant une unité de données ainsi que du contenu des unités de données. Le noeud sélectionné avant le transfert est le noeud 1. Remarquons que ce noeud n'a pas d'unité de données associée et donc les informations le concernant ne sont pas transférées. Il n'apparaît pas dans la figure. Remarquons encore que les unités de données sont rangées dans l'ordre du parcours en préordre du FADU sélectionné.

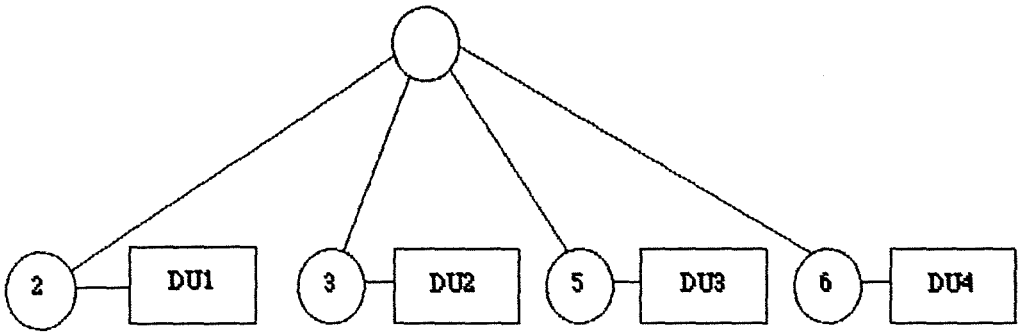


Figure 2.18 : structure transférée dans le contexte FA.

- plat sur les unités de données d'un niveau (Flat one level data units - FL) : les informations transférées seront identiques à l'accès FA mais seulement pour les noeuds appartenant au niveau choisi du FADU. Pour illustrer ce contexte d'accès, nous avons considéré l'accès aux noeuds fils de niveau 1. La figure 2.19 reprend les informations ainsi transférées.

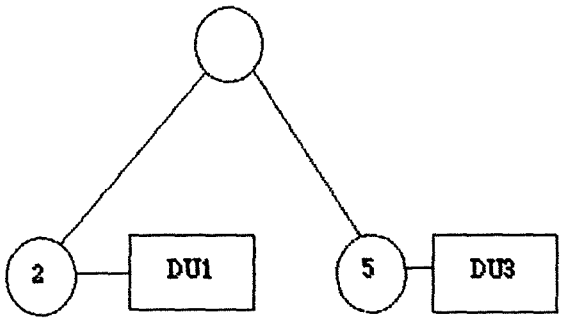


Figure 2.19 : structure transférée dans le contexte FL.

- plat sur une unité de données (Flat single data unit - FS) : les informations sur le noeud et les données du DU attaché au noeud racine du FADU sélectionné seront transférées. Pour illustrer ce cas, nous avons choisi le FADU ayant le noeud 2 pour racine (fig. 2.20).

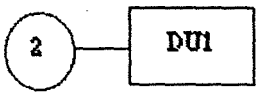


Figure 2.20 : structure transférée dans le contexte FS.

- non structuré sur toutes les unités de données (Unstructured all data units - UA) : seules les unités de données du FADU choisi seront transférées. La figure 2.21 reprend les informations transférées lorsqu'on se positionne sur le noeud 2 avant le transfert. Remarquons que les informations relatives au nom des noeuds ainsi qu'à leur situation dans la structure de départ n'apparaissent plus.

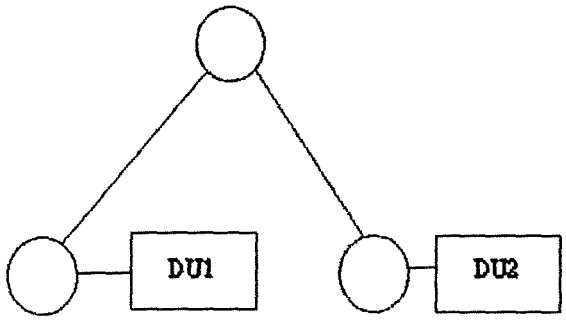


Figure 2.21 : structure transférée dans le contexte UA.

- non structuré sur une unité de données (Unstructured single data unit - US) : les données relatives au DU du noeud racine du FADU référencé seront transférées (fig. 2.22).

Figure 2.22 : structure transférée dans le contexte US.

Pour chaque structure de fichier un certain nombre de contextes d'accès sont autorisés.

- non structuré : US.
- séquentiel plat : FA, HA et UA.
- ordonné plat : HA, FA et UA.
- ordonné plat avec noms uniques : HA, FA et UA.
- ordonné hiérarchique :HA, HN, FA, FL, FS, UA et US.
- hiérarchique général : HA, HN, FA, FL, FS, UA et US.
- hiérarchique général avec noms uniques : HA, HN, FA, FL, FS, UA et US.

Si nous reprenons l'exemple du fichier séquentiel indexé (fig. 2.10-structure hiérarchique générale avec noms uniques), la lecture séquentielle du fichier peut être assurée en se positionnant sur le noeud racine et en demandant un contexte d'accès non structuré sur toutes les unités de données (contexte UA). Dans ce mode, seules les unités de données seront transférées les unes après les autres.

Considérons maintenant l'exemple de la figure 2.15. Si nous désirons reconstruire un fichier de clients, on se positionne sur le noeud "Cli". On peut alors se contenter du contexte d'accès plat sur toutes les unités de données. Un résultat similaire est obtenu en se plaçant sur le noeud "BD" et en demandant un contexte d'accès plat sur les unités de données d'un niveau (en l'occurrence, le niveau 2).

4.1.4. Le régime de transfert de données

L'utilisateur peut maintenant lire ou écrire des données. Il précise son intention dans un F-READ request ou un F-WRITE request, tous deux non confirmés. On entre dans le régime de transfert de données. Le FADU sélectionné est alors transféré selon le contexte d'accès négocié par une succession de F-DATA request réalisés par l'envoyeur. Ces primitives ne sont pas confirmées. La fin du transfert est signalée par l'envoyeur via un F-DATA-END request lui aussi non confirmé. Si, à un moment ou à un autre, l'envoyeur ou le récepteur souhaite pour une raison quelconque interrompre le transfert, il le signale par l'envoi d'un F-CANCEL request. L'initiateur du transfert décide de la sortie du régime via l'utilisation de la primitive F-TRANSFER-END. Nous réintégrons alors le régime précédent (à savoir, le régime d'ouverture de fichier).

4.1.5. La terminaison des régimes

Les actions disponibles dans le régime d'ouverture de fichier peuvent de nouveau être entreprises. L'utilisateur peut quitter ce régime par un F-CLOSE request. Cette primitive confirmée lui permet de réintégrer le régime de sélection de fichier. Pour quitter le régime de sélection de fichier, l'utilisateur recourt à la primitive F-DESELECT. Enfin, après la récupération du régime d'association FTAM, l'utilisateur peut mettre fin à l'association en utilisant la primitive F-TERMINATE request.

En guise de résumé, la figure 2.23 reprend les différents régimes FTAM ainsi que les primitives disponibles pour chaque régime.

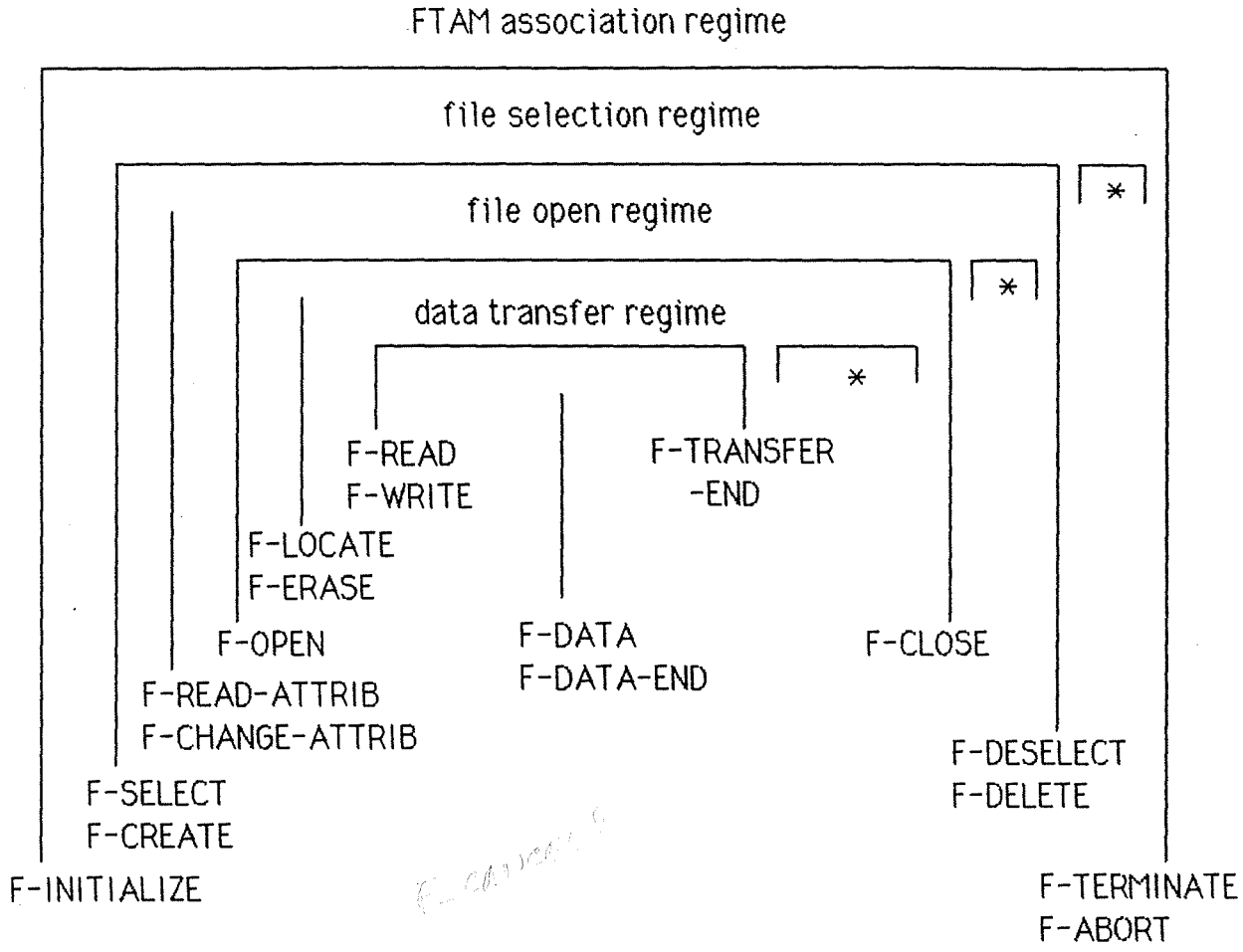


Figure 2.23 : les régimes FTAM.

Outre les primitives décrites ci-dessus FTAM offre quelques possibilités supplémentaires comme nous allons le voir.

4.1.6. Les classes proposées par FTAM

FTAM permet à l'utilisateur de travailler dans une classe prédéfinie donnée. L'intérêt résulte du fait que l'utilisateur ne souhaite pas forcément disposer de toutes les possibiltés offertes par FTAM ; cela dépend de l'application réalisée. Les classes permises sont :

- la classe de transfert de fichier (classe 1).
- la classe d'accès au fichier (classe 2).
- la classe de gestion du fichier (classe 3).
- la classe de transfert et de gestion de fichier (classe 4).
- la classe libre (classe 5).

Afin d'accéder uniquement aux primitives permises par la classe (celles-ci correspondent aux primitives accessibles dans un régime avec une limitation supplémentaire éventuelle), il faut enchaîner les primitives pour se retrouver directement dans le bon régime. Deux primitives obligatoires lorsqu'une classe est sélectionnée sont disponibles pour grouper des primitives. Il s'agit des primitives F-BEGIN-GROUP et F-END-GROUP. Ces primitives sont également utilisées pour quitter la classe négociée. Elles encadrent la succession des primitives et permettent une réponse groupée de ces primitives. Le groupement n'a pas de justification conceptuelle mais bien une raison d'efficacité d'implémentation.

4.1.7. Le transfert fiable des données

FTAM peut assurer un transfert fiable des données. La primitive F-CHECK request permet d'insérer des points de synchronisation dans le flot des F-DATA request. La primitive F-RESTART autorise la négociation d'un point de synchronisation commun à partir duquel l'utilisateur désire reprendre l'envoi des données. La primitive F-RECOVER request permet la récupération du régime en cas d'erreurs plus graves que celles survenues sur le flot de données, comme par exemple une rupture momentanée de l'association . FTAM fonctionne selon deux modes : soit le service de fichier corrigible par l'utilisateur (UCFS - user correctable file service) soit le service fiable de fichier (RFS - reliable file service). Le mode UCFS permet la gestion par l'utilisateur des primitives précitées. Dans le mode RFS, l'entité FTAM prend en charge la synchronisation et la gestion des éventuelles reprises sur erreur.

4.1.8. La fin abrupte de l'association

Nous terminons la revue des primitives en signalant la possibilité donnée à l'utilisateur de terminer à tout moment l'association par l'utilisation de la primitive F-U-ABORT request. Cette terminaison brutale provoque l'abandon pur et simple du travail en cours par les deux entités FTAM correspondantes. Enfin, en cas de problèmes provenant du fournisseur, l'utilisateur reçoit un F-P-ABORT indication qui met lui aussi fin à la session de manière abrupte.

4.1.9. Les unités fonctionnelles

FTAM regroupe les primitives dans des unités fonctionnelles (FU). L'intérêt résulte du fait que pour chaque classe, seules certaines unités fonctionnelles sont nécessaires. Ainsi, une machine qui n'implémente qu'un petit nombre d'unités fonctionnelles, permet-elle l'utilisation d'un nombre limité de classes.

Les unités fonctionnelles disponibles sont reprises dans le tableau ci-dessous (tableau 2.1). La première colonne reprend le nom des différentes unités fonctionnnelles. La deuxième colonne décrit quelles sont les unités fonctionnelles indispensables pour chaque classe (voir légende). La troisième colonne reprend la liste des services disponibles dans l'unité fonctionnelle.

Functionnal unit	1	2	3	4	5	Service element
Kernel	M	M	M	M	M	F-INITIALIZE F-TERMINATE F-ABORT F-SELECT F-DESELECT
Read	*	M	-	*	0	F-READ F-DATA F-DATA-END F-TRANSFER-END F-CANCEL F-OPEN F-CLOSE
Write	*	M	-	*	0	F-WRITE F-DATA F-DATA-END F-TRANSFER-END F-CANCEL F-OPEN F-CLOSE
File access	-	M	-	-	0	F-LOCATE F-ERASE
Limited file management	0	0	M	M	0	F-CREATE F-DELETE F-READ-ATTRIB
Enhanced file management	0	0	M	M	0	F-CHANGE-ATTRIB
Grouping	M	0	M	M	0	F-BEGIN-GROUP F-END-GROUP
Recovery	0	0	-	0	0	F-RECOVERY F-CHECK F-CANCEL
Restart data transfer	0	0	-	0	0	F-RESTART F-CHECK F-CANCEL

M : Mandatory (obligatoire) O : Optional (optionnel) - : not available (non disponible) * : au moins une FU read ou write doit être incluse.

Tableau 2.1 : les unités fonctionnelles.

4.2. Organisation du modèle FTAM dans le cadre du modèle ISO

Dans cette partie, nous allons examiner comment FTAM s'intègre dans le modèle ISO. Nous étudierons comment les services offerts sont remplis et comment les services des autres couches du modèle sont utilisés.

Au niveau application, on distingue l'agent et l'entité. L'agent est l'utilisateur de l'entité FTAM. C'est lui qui envoie et reçoit les primitives FTAM et qui s'occupe de la gestion des fichiers réels à partir des informations fournies ou reçues dans les paramètres des primitives. En aucun cas, cette gestion n'est standardisée par ISO étant donné qu'il s'agit d'une organisation purement locale à partir des informations transférées. ISO s'occupe de définir précisement la collaboration entre entités paires FTAM qui s'échangent des informations sous forme de PDU (en FTAM on parle de FPDU ou FTAM Protocole Data Unit) afin de satisfaire les services demandés par l'agent. Signalons que dans le cas de FTAM tous les FPDU sont codés dans la syntaxe ASN.1 spécifiée par les normes [ISO8824] et [ISO8825].

FTAM regroupe deux types de services. Premièrement le service de fichier corrigible par l'utilisateur (User Correctable File Service) qui permet à l'agent d'utiliser les primitives F-CHECK, F-RESTART, F-RECOVER afin d'implémenter sa propre gestion de correction des erreurs. Deuxièment le service de transfert fiable de fichier (Reliable File Service) où les primitives précitées ne sont pas disponibles pour l'utilisateur mais sont gérées par l'entité FTAM selon un protocole spécifié dans la norme (fig. 2.24). La Basic FPM (Basic File Protocole Machine) ou machine de protocole de fichier de base assure le service des différentes primitives FTAM. Pour ce faire elle utilise les services de l'ACSE et de la couche présentation (lower service provider). L'Error Recovery FPM ou machine de protocole de fichier avec recouvrement d'erreurs permet la gestion automatique du transfert de fichier, elle assure l'insertion des points de synchronisation et la reprise du transfert en cas d'erreur. L'utilisateur de l'un ou l'autre de ces services se situe au dessus de la machine correspondante (Reliable file service user ou user correctable file service user).

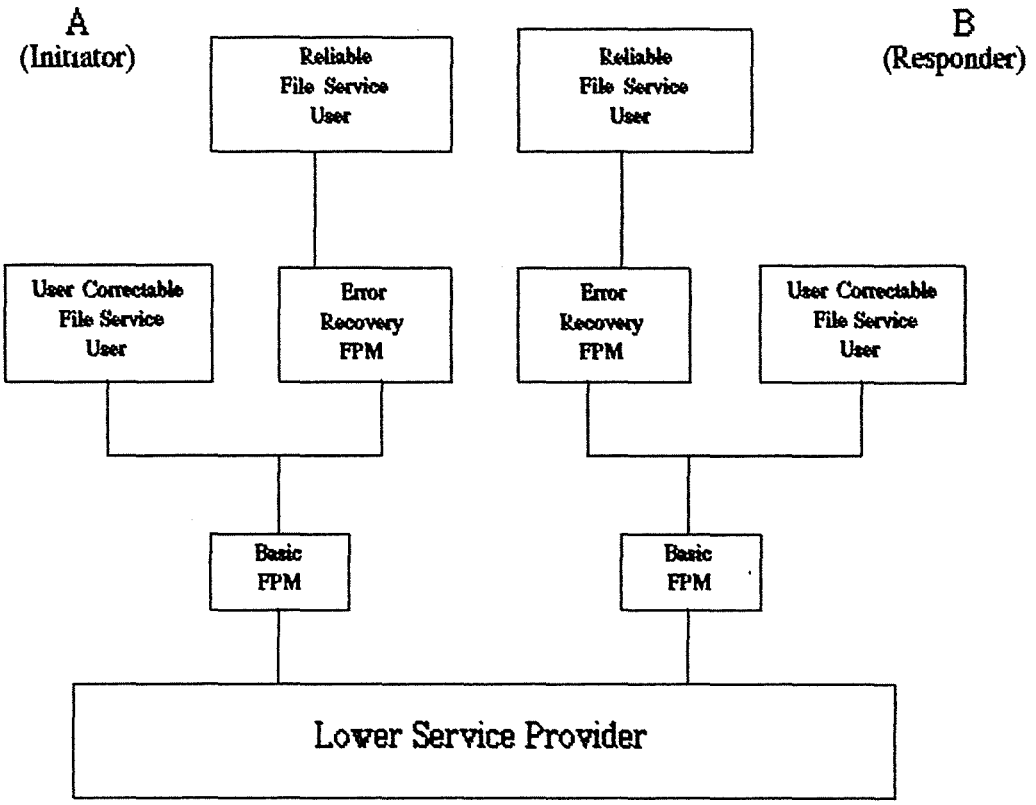


Figure 2.24 : modèle étendu de la machine de protocole de fichier.

Précisons en outre que par utilisateur ("user" dans Reliable File Service user ou dans User Correctable File Service user), on désigne un agent de la couche application qui utilise un enchaînement des primitives FTAM dans un programme de manière à offrir un service de plus haut niveau. L'application pourrait par exemple offrir la possiblité à un opérateur situé à son terminal de lire un fichier sur une machine en tapant une seule commande. Sur la figure 2.25, cela veut dire que le "FTAM user" sélectionne une commande "Lire le fichier". Si l'application FTAM tourne en mode RFS, la demande de lecture va passer par le "Error Recovery FPM" qui se chargera du transfert de données fiable. Le programme conçu au niveau de l'agent FTAM devra donc demander l'ouverture de l'association avec la machine qui stocke le fichier (consultation de tables, ...), la sélection du fichier, son ouverture et sa lecture. L'entité "Error Recovery FPM" se charge de placer des F-CHECK et de redemander l'envoi de données en cas d'erreurs (F-RESTART). Le résultat de la lecture parvient à l'entité FTAM via des F-DATA indication, le résultat est contrôlé. Ensuite, la donnée est transmise au programme d'application. Quand celui-ci a recueilli toute l'information, il la donne au "FTAM user" sous une forme propre à l'application. Dans le cas d'un agent utilisant l'UCFS, le programme réalisant la lecture est plus compliqué : il doit placer les points de synchronisation et réaliser le contrôle lors de la réception des données. Il doit recommencer éventuellement la procédure en cas d'erreur.

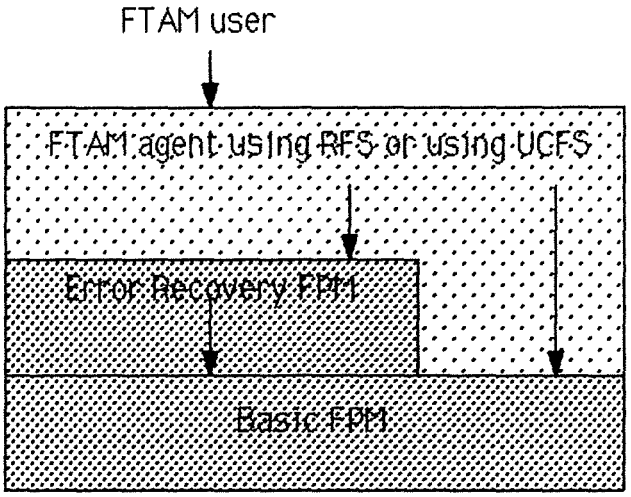


Figure 2.25 : l'utilisateur et l'agent dans le modèle FTAM.

4.2.1. L'utilisation de l'ACSE

Pour communiquer avec son entité paire, l'entité utilise les services offerts par la couche présentation et par l'entité ACSE de la couche application. L'utilisation du service ACSE permet l'établissement d'une association entre deux entités FTAM. L'entité FTAM qui reçoit une demande de connexion via un F-INITIALIZE request construit un F-INITIALIZE request FPDU qu'elle place dans le champ de données (user data) du A-ASSOCIATE request. A la réception du A-ASSOCIATE indication, l'entité ACSE paire passe le champ de données utilisateur à l'entité FTAM identifiée. Celle-ci décode le FPDU, le reconnaît et communique la demande à son agent via un F-INITIALIZE indication. La réponse et la confirmation suivent le chemin en sens inverse. D'autres primitives sont transportées par le champ de données utilisateur des APDU, il s'agit des primitives F-TERMINATE et F-U-ABORT.

4.2.2. L'utilisation de la couche présentation

Les autres FPDU sont communiqués par l'utilisation directe des services de la couche présentation. La plupart sont envoyés via la primitive de P-DATA request. Les F-CANCEL request FPDU et F-CANCEL response FPDU sont transmis dans le champ de données d'un P-RESYNCHRONISE si l'unité fonctionnelle existe ou, à défaut, par un P-DATA normal. L'utilisation du P-RESYNCHRONIZE aura un effet sur la gestion des points de synchronisation au niveau de la couche session.

Le F-DATA ne donne pas lieu à la construction d'un FPDU. L'entité FTAM qui reçoit un F-DATA request passe les données (DU) et la syntaxe abstraite sous forme de paramètres d'un P-DATA request.

Aucun FPDU ne correspond à la primitive F-CHECK qui donne lieu à l'utilisation d'un P-SYNC-MINOR.

Le tableau 2.2 reprend, en guise de résumé, les différentes primitives FTAM utilisées par l'agent FTAM (colonne 1), les FPDU correspondants construits par la machine protocolaire FTAM (colonne 2), les primitives ACSE ou Présentation utilisées par la machine protocolaire FTAM pour assurer le service FTAM demandé (colonne 3), les primitives reçues par la machine protocolaire FTAM paire (colonne 4) et les primitives FTAM envoyées à l'agent FTAM pair (colonne 5).

Primitive FTAM	FPDU	Primitive ACSE ou PRESENTATION	Primitive ACSE ou PRESENTATION	Primitive FTAM
F_INI.req	F_INI_RQ	A_ASS.req	A_ASS.ind	F_INI.ind
F_INI.resp	F_INI_RS	A_ASS.resp	A_ASS.conf	F_INI.conf
F_TERM.req	F_TERM_RQ	A_REL.req	A_REL.ind	F_TERM.ind
F_TERM.resp	F_TERM_RS	A_REL.resp	A_REL.conf	F_TERM.conf
F_UABT.req	F_UABT_RQ	A_UABT.req	A_UABT.ind	F_UABT.ind
-	F_PABT_RQ	A_ABT.req	A_ABT.ind	F_PABT.ind
F_SEL.req	F_SEL_RQ	P_DATA.req	P_DATA.ind	F_SEL.ind
F_SEL.resp	F_SEL_RS	P_DATA.req	P_DATA.ind	F_SEL.conf
F_CRE.req	F_CRE_RQ	P_DATA.req	P_DATA.ind	F_CRE.ind
F_CRE.resp	F_CRE_RS	P_DATA.req	P_DATA.ind	F_CRE.conf
F_DESEL.req	F_DESEL_RQ	P_DATA.req	P_DATA.ind	F_DESEL.ind
F_DESEL.resp	F_DESEL_RS	P_DATA.req	P_DATA.ind	F_DESEL.conf
F_DEL.req	F_DEL_RQ	P_DATA.req	P_DATA.ind	F_DEL.ind
F_DEL.resp	F_DEL_RS	P_DATA.req	P_DATA.ind	F_DEL.conf
F_RD_ATT.req	F_RDATT_RQ	P_DATA.req	P_DATA.ind	F_RD_ATT.ind
F_RD_ATT.resp	F_RDATT_RS	P_DATA.req	P_DATA.ind	F_RD_ATT.conf
F_CH_ATT.req	F_CHATT_RQ	P_DATA.req	P_DATA.ind	F_CH_ATT.ind
F_CH_ATT.resp	F_CHATT_RS	P_DATA.req	P_DATA.ind	F_CH_ATT.conf
F_OPN.req	F_OPN_RQ	P_DATA.req	P_DATA.ind	F_OPN.ind
F_OPN.resp	F_OPN_RS	P_DATA.req	P_DATA.ind	F_OPN.conf
F_CLO.req	F_CLO_RQ	P_DATA.req	P_DATA.ind	F_CLO.ind
F_CLO.resp	F_CLO_RS	P_DATA.req	P_DATA.ind	F_CLO.conf
F_BGR.req	F_BGR_RQ	P_DATA.req	P_DATA.ind	F_BGR.ind
F_BGR.resp	F_BGR_RS	P_DATA.req	P_DATA.ind	F_BGR.conf
F_EGR.req	F_EGR_RQ	P_DATA.req	P_DATA.ind	F_EGR.ind
F_EGR.resp	F_EGR_RS	P_DATA.req	P_DATA.ind	F_EGR.conf
F_REC.req	F_REC_RQ	P_DATA.req	P_DATA.ind	F_REC.ind
F_REC.resp	F_REC_RS	P_DATA.req	P_DATA.ind	F_REC.conf
F_LOC.req	F_LOC_RQ	P_DATA.req	P_DATA.ind	F_LOC.ind

F_LOC.resp	F_LOC_RS	P_DATA.req	P_DATA.ind	F_LOC.conf
F_ERA.req	F_ERA_RQ	P_DATA.req	P_DATA.ind	F_ERA.ind
F_ERA.resp	F_ERA_RS	P_DATA.req	P_DATA.ind	F_ERA.conf
F_READ.req	F_RD_RQ	P_DATA.req	P_DATA.ind	F_READ.ind
F_WRI.req	F_WR_RQ	P_DATA.req	P_DATA.ind	F_WRI.ind
F_DATA.req	-	P_DATA.req	P_DATA.ind	F_DATA.ind
F_DEND.req	F_EDAT_RQ	P_DATA.req	P_DATA.ind	F_DEND.ind
F_TSFEND.req	F_ETRSF_RQ	P_DATA.req	P_DATA.ind	F_TSFEND.ind
F_TSFEND.resp	F_ETRSF_RS	P_DATA.req	P_DATA.ind	F_TSFEND.conf
F_CAN.req	F_CAN_RQ	P_RESYNC.req	P_RESYNC.ind	F_CAN.ind
F_CAN.resp	F_CAN_RS	P_RESYNC.resp	P_RESYNC.conf	F_CAN.conf
F_CHK.req	-	P_SMIN.req	P_SMIN.ind	F_CHK.ind
F_CHK.resp	-	P_SMIN.resp	P_SMIN.conf	F_CHK.conf
F_RSTART.req	F_RSTRT_RQ	P_RESYNC.req	P_RESYNC.ind	F_RSTART.ind
F_RSTART.resp	F_RSTRT_RS	P_RESYNC.resp	P_RESYNC.conf	F_RSTART.conf

Tableau 2.2 : FTAM et l'utilisation de son fournisseur.

4.2.3. L'utilisation de la couche session

La couche session est utilisée suite à l'appel via la présentation de S-SYNC-MINOR et S-RESYNCHRONIZE. L'absence de l'unité fonctionnelle implémentant ces primitives n'empêche pas le fonctionnement normal de deux entités FTAM paires. Celles-ci utiliseront uniquement le P-DATA de la présentation et donc le S-DATA de la session et remédieront aux manques de la couche session.

4.2.4. Exemples de fonctionnement de FTAM dans le cadre du modèle ISO

Pour terminer, nous illustrons le fonctionnement de FTAM sur deux exemples.

Un premier exemple concerne l'établissement d'une association entre deux agents FTAM pairs (fig. 2.26). L'agent initiateur exécute une demande d'association par un F-INITIALIZE request (1). L'entité FTAM construit un FPDU de demande d'association et le place dans le champ de données utilisateur du A-ASSOCIATE request (2). L'élément de service ACSE construit son propre PDU d'association et le place dans le champ de données utilisateur du P-CONNECT request (3). La couche Présentation en fait autant avant de demander un S-CONNECT request (4). La couche session commence par demander une connexion transport par l'utilisation de la primitive T-CONNECT request (5). Supposons qu'une connexion réseau soit déjà établie. La couche transport construit un TPDU de demande de connexion qu'elle envoie à son entité paire via le service de transfert des données de la couche réseau (N-DATA req (6)). Les données sont communiquées à l'entité transport de la machine répondante (7) qui décode le

TPDU et signale une demande de connexion à sa couche session par le biais de T-CONNECT indication (8). Supposons que la couche session accepte la connexion, elle communique sa réponse dans un T-CONNECT response (9). La couche transport utilise le service de transfert du réseau pour véhiculer la réponse (10,11). L'établissement de la connexion est confirmé par la couche transport à la couche session via la primitive T-CONNECT confirm (12). La couche session peut maintenant utiliser le service de transfert de données du transport pour satisfaire la demande de S-CONNECT request qui lui avait été faite (cfr. (4)). Elle envoie les données (SPDU) dans un T-DATA request (13). La couche transport utilise le fournisseur réseau pour envoyer les données (14,15). Les données reçues sont communiquées à l'entité session répondante grâce au T-DATA indication (16). La couche session décode le SPDU et signale à la couche présentation la demande de connexion par un S-CONNECT indication (17). Le champ de données utilisateur est décodé par la couche présentation qui trouve un PPDU de demande d'association et des données (le APDU et le FPDU qu'elle n'est pas en mesure de comprendre). La couche présentation signale la demande de connexion à la couche application (en fait à l'ACSE) par un P-CONNECT indication. Cette primitive contient dans son champ de données utilisateur les données qui n'ont pas pu être décodées par la couche présentation (18). L'élément de service ACSE décode les données et y trouve une demande d'association entre deux entités applications (il s'agit des deux entités FTAM dans notre cas). Le service signale la demande d'association à l'entité concernée. Le champ de données utilisateur s'est réduit au FPDU (19). Finalement, l'entité FTAM répondante décode le FPDU et communique la demande d'association à son agent FTAM (20). Supposons que la réponse soit favorable, un processus d'encapsulage équivalent aux demandes de connexion a lieu pour les réponses (21 - 26). Le décapsulage se déroule de manière similaire (27 - 32). Finalement, le régime d'association FTAM est établi.

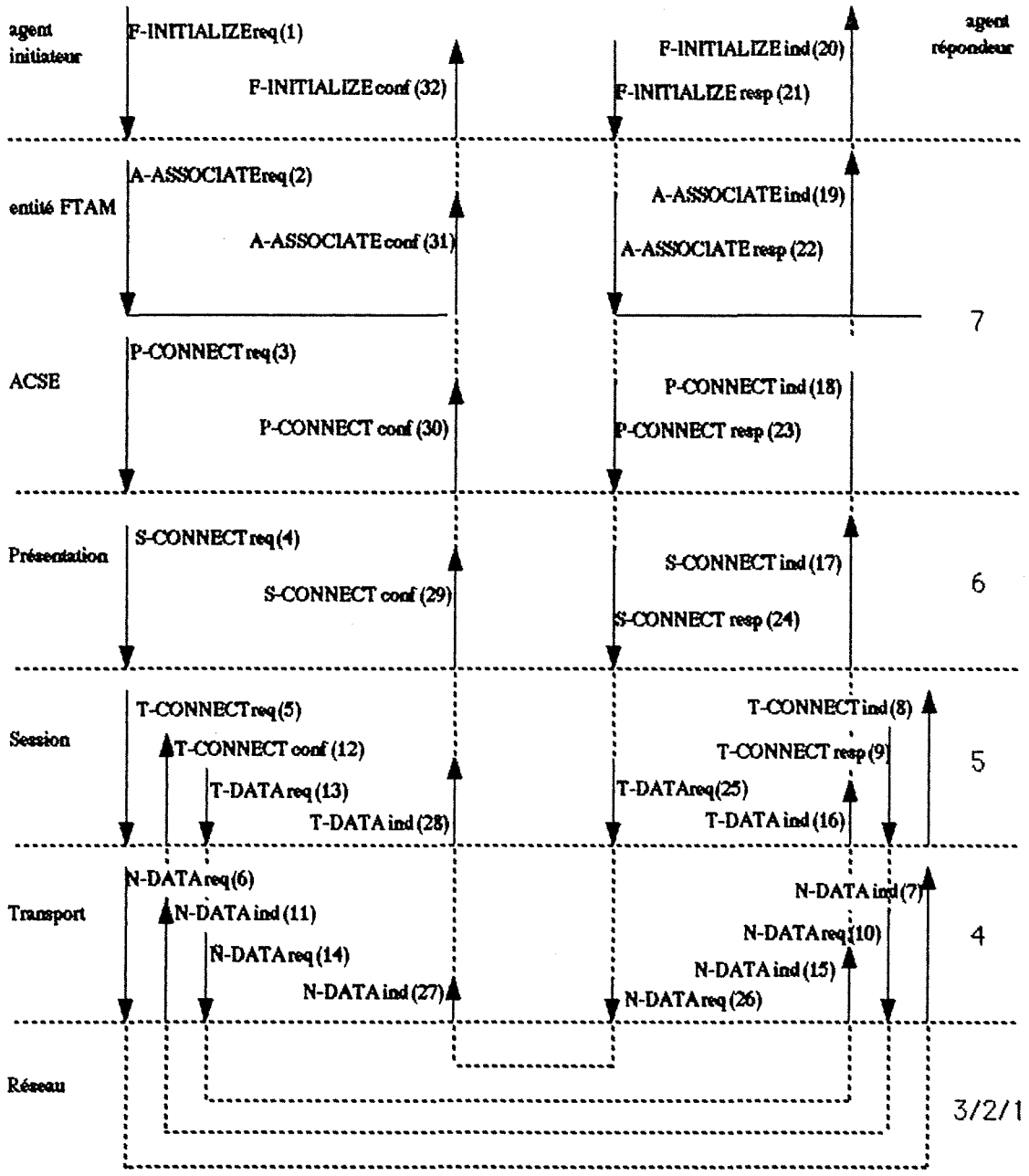


Figure 2.26 : l'établissement d'une association FTAM dans le modèle ISO.

Le deuxième exemple (fig. 2.27) concerne l'utilisation des primitives de transfert de données des couches inférieures pour réaliser le transfert de la plupart des FPDU. Nous avons choisi d'illlustrer la primitive F-SELECT. L'agent initiateur demande la sélection d'un fichier par la primitive F-SELECT request (1). L'entité FTAM construit un FPDU et l'envoie comme donnée à son entité paire en utilisant le service de P-DATA request de la couche présentation (2). Les données sont reportées à la couche session par un S-DATA request (3) puis au niveau transport par un T-DATA request (4) et enfin à la couche réseau

par un N-DATA request (5) ; les données reçues sont communiquées aux couches supérieures par une série d'indications (6 - 9). L'entité FTAM décode le FPDU et communique la demande à son agent par la primitive F-SELECT indication (10). La réponse suit un schéma similaire (11 - 20).

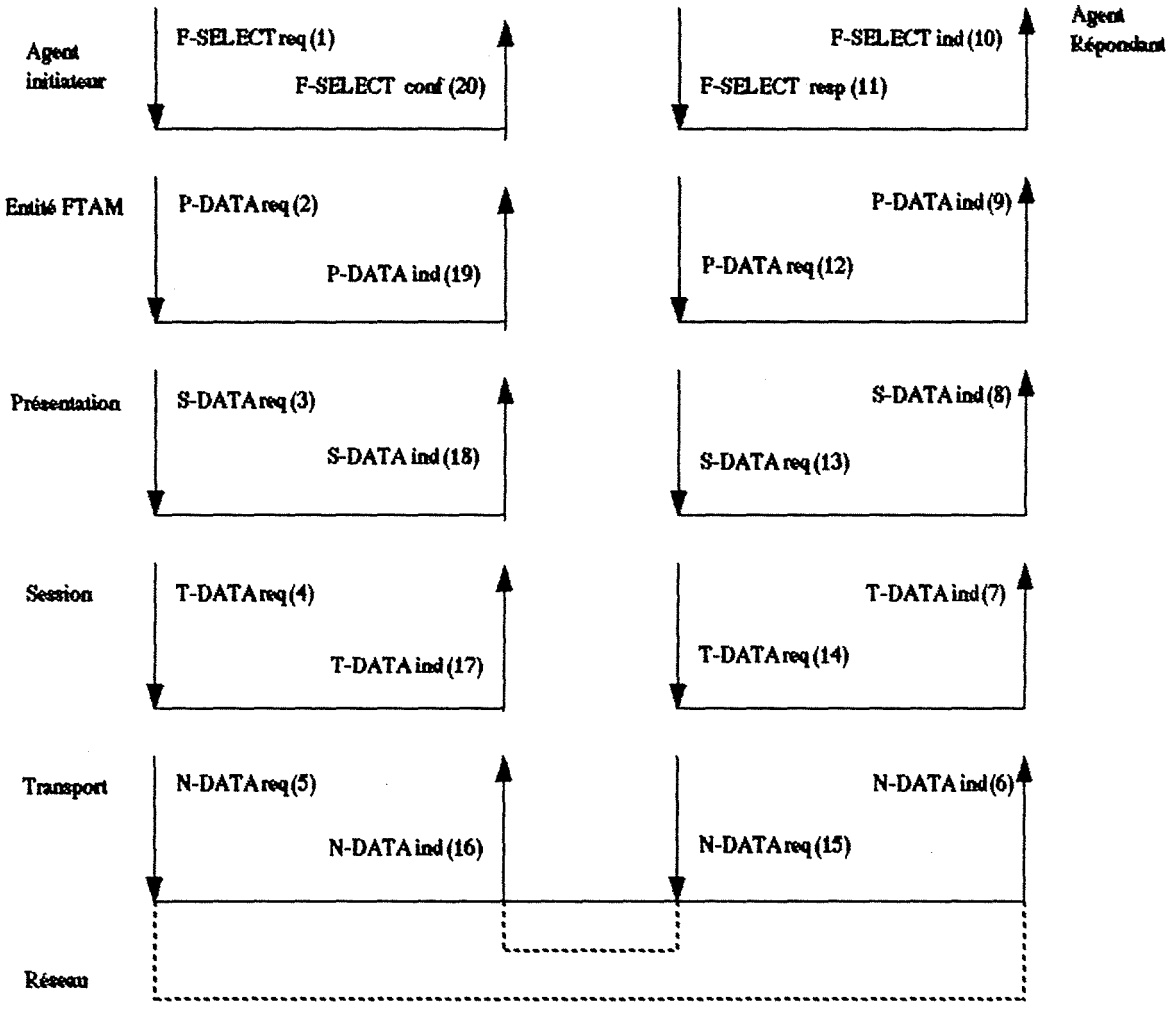


Figure 2.27 : la primitive F-SELECT dans le cadre du modèle ISO.

Chapitre 3 : Méthodologie de conception d'un didacticiel

La description d'une méthodologie de conception d'un didacticiel n'est pas chose fréquente dans la littérature. Le didacticiel est un logiciel destiné à l'apprentissage. La méthodologie décrite ci-dessous est basée sur l'ouvrage CONCEVOIR ET UTILISER UN DIDACTICIEL **[BESN88]**. Nous ferons une description succincte de la méthode proposée. Nous examinerons également les différents points pour en faire la critique au vu de notre situation. Nous reprendrons plusieurs critères intervenant dans l'élaboration d'un didacticiel.

1. Description de la méthode

1.1. Première étape : l'analyse pédagogique

Cette analyse a pour but de préciser l'objectif général et les objectifs spécifiques du didacticiel. Il s'agit également de déterminer la population cible et l'environnement d'utilisation du didacticiel. D'autre part, nous élaborons un plan de mise en oeuvre. Nous organisons le contenu, précisons la démarche pédagogique et décidons des formes d'interaction.

L'objectif général d'un didacticiel, tel qu'on le conçoit en pédagogie, doit être décrit en terme d'évaluation. C'est-à-dire qu'il faut pouvoir définir une méthode qui permette de tester la valeur de l'apprentissage chez l'étudiant afin de vérifier que l'objectif fixé est atteint. Un exemple d'objectif général pourrait être : "A la fin de l'apprentissage, l'étudiant sera capable d'expliquer l'enchaînement des primitives dans le modèle ISO".

Pour atteindre ce but, plusieurs objectifs spécifiques doivent être satisfaits par l'apprenant, par exemple : "L'étudiant connaîtra les couches du modèle ISO, le fonctionnement général du modèle, les différentes primitives utilisées". Le didacticiel est découpé en unités d'apprentissage qui visent à l'acquisition de ces objectifs intermédiaires.

Les caractéristiques de la population cible doivent être prises en considération lors de l'élaboration du logiciel. Il faut évidemment fournir un produit adapté aux apprenants. Précisons les critères à prendre en considération :

- la structure cognitive. Il s'agit de l'organisation de l'ensemble des connaissances d'une personne. Le concepteur ne dispose pas d'information

complète sur la structure cognitive des apprenants. De ce fait, il ne peut prendre aucune décision à ce niveau.

- le stade de développement cognitif. Il s'agit du niveau de maturation atteint par l'étudiant tant au niveau de l'équipement que du mode de fonctionnement cognitif. La connaissance du stade de développement cognitif par le concepteur est très aléatoire. Il postulera que le stade nécessaire est atteint, ce qui bien sûr ne sera pas toujours vrai.
- la capacité intellectuelle. Il s'agit du niveau général d'intelligence telle qu'elle est mesurée par les tests généraux. Le concepteur peut parfois avoir connaissance de ce type de document. Pour produire un travail efficace, il vaut mieux postuler que le niveau requis est atteint.
- les facteurs motivationnels et les attitudes. Il s'agit des facteurs qui influencent le désir de faire quelque chose, d'atteindre un objectif. Des entretiens avec la population cible permettent de se rendre compte du niveau de motivation. Le concepteur tentera de susciter cette motivation lors de l'apprentissage.
- les facteurs socio-psychologiques. Il s'agit des interactions entre étudiants ou entre étudiants et enseignants. Dans une situation d'autoformation, ces facteurs sont peu pertinents.
- les facteurs didactiques. Il s'agit de la démarche et de l'environnement pédagogique à mettre en oeuvre. La question fondamentale à résoudre est celle-ci : quelles activités intellectuelles faut-il chercher à susciter chez l'apprenant pour atteindre l'objectif pédagogique de façon durable et avec un maximum de chance d'en favoriser le transfert ?
- les circonstances réelles de l'environnement. Il s'agit de la réalité de la situation dans laquelle se déroulera l'apprentissage. Par exemple, la conception du didacticiel doit prendre en compte les plages de temps disponibles pour l'apprentissage. Il faut également éviter les effets sonores si l'apprentissage se déroule dans une salle commune, ...

La prise en compte de ces différents critères permet de mettre en oeuvre une stratégie pédagogique tant au niveau du matériel que du contenu. Il faut organiser les écrans, choisir les couleurs,... Il faut aussi structurer le contenu de manière telle que l'apprentissage en soit facilité. Les échanges d'informations entre l'apprenant et la machine sont divisés en unités d'interaction. Il faut prévoir le rôle de chaque unité d'interaction dans le scénario ainsi que l'enchaînement des unités d'interaction entre elles. La figure 3.1 reprend la structure d'une unité d'interaction. L'unité débute par une sollicitation de l'apprenant par la machine. L'étudiant entre les données. La machine réagit aux données entrées par l'étudiant. La machine décide du branchement vers une autre unité d'interaction. Lorsque la machine sollicite l'apprenant, celui-ci doit pouvoir refuser la sollicitation, demander de l'aide ou revenir en arrière.

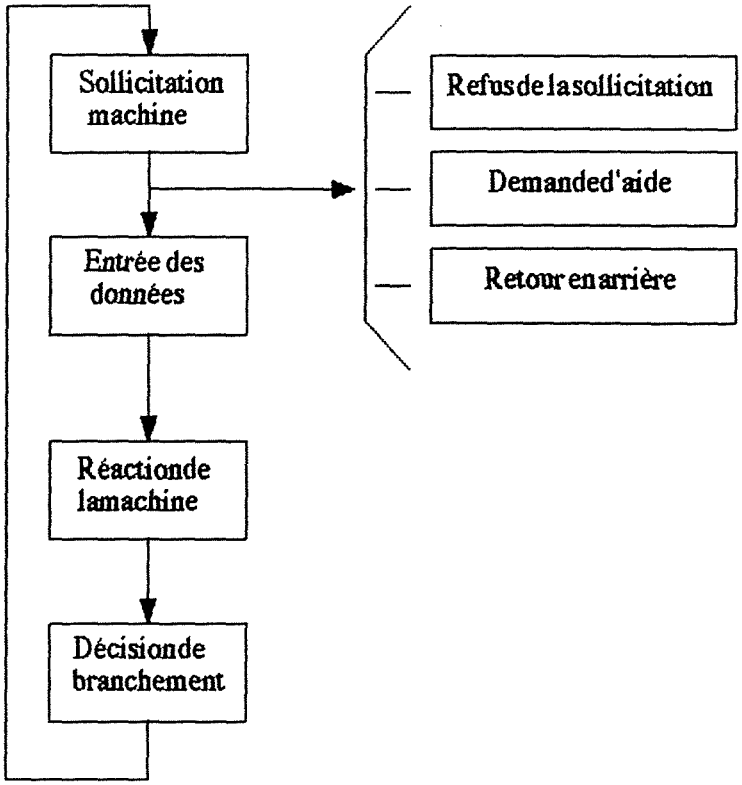


Figure 3.1 : structure de l'unité d'interaction.

Il faut envisager les différentes formes d'interaction offertes par l'ordinateur pour permettre l'apprentissage. Nous ne considérerons dans le cadre du projet COLOS que les formes à simulation mais citons pour mémoire les autres formes d'interaction possibles :

- les formes à choix (choix unique, choix multiples). Un exemple peut être la question suivante : "La couche présentation est le client de la couche session (vrai ou faux)?".
- les formes à corrélations (mise en rapport d'éléments). Citons à titre d'exemple : "Attachez les primitives : **N-DATA**, **S-ACTIVITY-DISCARD**, **P-SYNCH-MINOR** à leurs couches respectives : *réseau, présentation, session*".
- les formes ouvertes (réponses à compléter par exemple). Un exemple de forme ouverte en rapport avec le monde ISO pourrait être la phrase suivante : "Pour communiquer avec son entité paire, une entité envoie des ...". La forme est dite ouverte car l'on peut admettre plusieurs bonnes réponses Ici, les réponses possibles sont "PDU" ou "unités de protocole de données" par exemple.

Les formes à simulation se caractérisent par le fait que les réponses de l'apprenant sont de simples données (paramètres) qui permettent de faire évoluer la situation. Tenant compte des paramètres et du modèle, l'ordinateur simule l'évolution du phénomène. L'apprentissage se réalise lorsque l'étudiant

est amené à vérifier l'effet de l'entrée des paramètres sur l'évolution de la simulation. Peu à peu, il reconstitue le modèle sous-jacent à la simulation. La simulation permet donc, en principe, d'induire des connaissances chez l'apprenant mais aucune garantie n'est présente puisqu'il n'y a pas de vérification des progressions de l'apprenant. De plus une simulation à vocation pédagogique est généralement plus réductrice que la réalité. Trop de paramètres à gérer dérouteraient l'étudiant. L'intérêt pédagogique d'une simulation provient du fait que l'apprenant procède par expérimentation. La valeur pédagogique de l'erreur est plus forte que dans le mode tutoriel, l'activité de recherche personnelle plus importante. Elle permet une approche globale de la réalité traitée.

On peut s'étonner de devoir considérer les formes avec lesquelles l'apprenant va interagir dès la première étape de la méthode. Cela tient au fait que les formes d'interaction jouent un rôle fondamental dans le processus d'apprentissage. Elles font partie intégrante du comportement d'apprentissage.

En ce qui concerne le nombre de paramètres, ils sont très nombreux dans FTAM et nous pensons qu'il faut concevoir des unités d'apprentissage différentes si nous voulons arriver à un certain niveau de maîtrise chez l'apprenant. D'autant que d'une primitive à l'autre, les paramètres sont très différents. Plusieurs petits scénarios valent mieux qu'un grand scénario.

1.2. Deuxième étape : la validation de l'analyse pédagogique

Le projet pédagogique établi, il doit recevoir l'aval du commanditaire du didacticiel. Sauter cette étape revient à s'exposer à une remise en cause de tout le travail beaucoup plus tard dans la conception. De toute façon, la conception d'un didacticiel est avant tout une question de collaboration entre spécialistes de différents domaines. Informaticiens, pédagogues,spécialistes du domaine étudié travaillent ensemble pour obtenir le produit final. L'étape de validation marque la fin de l'étape d'analyse pédagogique lorsque l'ensemble des personnes concernées tombent d'accord entre elles.

1.3. Troisième étape : réalisation de la maquette papier

La conception sur papier du didacticiel constitue un gage de la qualité du travail. La maquette papier apporte une aide précieuse au concepteur et au commanditaire du didacticiel. Il faut se montrer très rigoureux. Cette méthode permet de tester la pertinence et la cohérence de la conception.

Généralement le spécialiste pédagogique n'est pas forcément un spécialiste en informatique, il faut donc trouver une forme de spécification du scénario qui soit compréhensible par tous. Il est recommandé de présenter les choses de la manière la plus proche possible de la forme finale à l'écran. Ceci permet d'éviter tout malentendu. Remarquons toutefois que, dans le cas d'une simulation, la spécification est beaucoup plus complexe étant donné la multiplicité des situations possibles en fonction des paramètres. La spécification que l'on peut faire est le modèle mathématique qui sous-tend la simulation. Il n'est pas possible d'envisager tous les écrans pour toutes les situations. De même les animations sont difficiles à spécifier autrement que sous un style "bande dessinée".

Dans notre cas, la matière est informatique et les personnes concernées sont des informaticiens. Nous pouvons envisager l'écriture de spécifications plus rigoureuses et plus concises qu'une forme schématique. Les objets graphiques et leurs comportements seront définis de manière schématique. Par la suite, nous pourrons décrire l'ensemble de l'évolution de la simulation en faisant référence aux comportements définis précédemment. Cette manière de travailler est plus concise, plus précise. Elle a l'inconvénient d'être plus abstraite. Le langage employé pourrait éventuellement servir à la définition d'outils de prototypage.

1.4. Quatrième étape : validation de la maquette papier

Cette étape concerne la validation de la maquette papier sur un échantillon de la population cible afin de vérifier l'adéquation de la méthode pour l'apprentissage.

Pour notre part nous considérons que ce genre de test est très difficile à réaliser dans le cas de l'élaboration d'un didacticiel de simulation. Dans une simulation, la multiplicité des possibilités à envisager est telle, qu'une maquette papier exhaustive de part sa taille n'est pas pertinente. Le test d'un prototype semble plus approprié. A cet égard une approche incrémentale est tout à fait envisageable dans le cadre d'une approche orientée objets.

1.5. Cinquième étape : médiatisation

Il s'agit maintenant de réaliser le scénario sur une machine. Cette étape est facilitée par les outils à notre disposition et par l'existence d'un scénario clair et détaillé.

1.6. Sixième étape : validation informatique

Il s'agit d'une étape non spécifique à l'EAO où il s'agit de vérifier que le programme ne contient pas d'erreur.

1.7. Septième étape : tests

Le produit fini doit être testé sur un échantillon de la population cible. Il s'agit maintenant de vérifier que la forme d'interaction offerte par le logiciel est adéquate.

1.8. Huitième étape : correction

Sur base des tests effectués, des corrections appropriées seront apportées au programme.

1.9. Neuvième étape : diffusion

Il s'agit d'implanter le produit fini dans l'environnement d'apprentissage et de le mettre en fonction.

1.10. Dixième étape : évaluation

Il s'agit d'évaluer le produit en situation de formation en vraie grandeur.

2. Application de la méthode

2.1. Première étape : l'analyse pédagogique

Le projet d'EAO entrepris par l'équipe de Namur est très vaste puisqu'il se propose à plus ou moins long terme d'étudier l'ensemble de la matière des télécommunications. En guise de première approche nous avons réduit notre objectif pédagogique à l'apprentissage de FTAM et plus particulièrement encore nous définissons dans ce cadre les objectifs suivants :

- l'apprentissage de la structure du modèle FTAM dans le cadre du modèle ISO. L'apprenant devra pouvoir distinguer entre agent et entité. Il devra également se rendre compte de "qui" fait "quoi". Par exemple, l'agent s'occupe de la gestion du fichier réel, l'entité s'occupe de transmettre à son entité paire les demandes de services faites par l'agent.

- l'apprentissage du fonctionnement du modèle. L'étudiant sera capable d'accomplir une tâche spécifique dans le cadre de FTAM. C'est-à-dire qu'il sera capable d'enchaîner les primitives de manière à accomplir la tâche.

En ce qui concerne la population cible il s'agit de considérer principalement les étudiants de deuxième licence en informatique qui suivent le cours de télécommunications matières approfondies. Dans le cadre de ce cours, la compréhension d'ISO en général et de FTAM en particulier n'est pas toujours évidente. Bien souvent, on retrouve les mêmes questions. La population est généralement motivée parce que tous les étudiants qui suivent ce cours à option l'ont choisi délibérément.

Les facteurs didactiques résultent du fait que dans l'environnement de travail RMG, nous utilisons une simulation. La question pertinente à se poser dans ce cadre est de savoir si une simulation est adaptée à notre cas? La réponse est affirmative si l'on considère qu'il s'agit d'étudier l'utilisation que l'on peut faire d'un modèle tel que FTAM. Aucune autre forme d'interaction ne semble convenir mieux.

Les objets qui font partie de la simulation sont les agents FTAM, le fournisseur FTAM, les fichiers. Nous envisageons de représenter l'échange de primitives entre agents pairs et de visualiser leurs effets sur le système de fichiers. En vue de fournir une aide constante à l'apprenant, nous envisageons d'afficher le régime FTAM courant et les états des machines de protocole de fichier.

2.2. Deuxième étape : la validation de l'analyse pédagogique

C'est l'équipe COLOS qui, en accord avec le professeur Ph. Van Bastelaer, a pris la décision de valider les objectifs pédagogiques. L'utilisation d'une simulation ne se discutait même pas étant donné le cadre du projet COLOS et la nécessité de représenter le comportement d'un modèle.

2.3. Troisième étape : réalisation de la maquette papier

Cette étape fait l'objet du chapitre 4. Signalons que la spécification présentée n'a pas la même forme que celle réalisée sur papier. Nous avons dans un premier temps envisagé la présentation de l'enchaînement des primitives sous forme graphique (cfr. annexe 4). Nous avons ensuite choisi de présenter une forme plus textuelle. Les raisons de ce choix s'expliquent par le fait qu'une telle spécification est plus précise et plus rigoureuse. De plus, cette forme est plus concise. Elle permet déjà d'envisager une implémentation. Elle ne pose pas de problème de compréhension pour des informaticiens. Il est bien évident

que si nous devions travailler avec un pédagogue n'ayant pas de connaissances informatiques, cette forme de spécification sera tout à fait inadéquate. Remarquons cependant que le langage utilisé (sorte de pseudo PASCAL) pourrait se prêter à un prototypage. Nous diposerions alors d'un moyen commode de générer les écrans de la simulation pour éditer un scénario complet. Eventuellement, si le contexte s'y prête, nous parviendrions à la réalisation non optimale de l'implémentation du produit. Les possibilités de prototypage offertes par RMG ne permettent pas une spécification du scénario. Elles permettent plutôt une implémentation du scénario par manipulation directe des objets de la scène. Dans une simulation un peu complexe, les possibilités deviennent tellement nombreuses que le recours à une spécification précise du scénario est indispensable. De même, la programmation en langage Objective-C est inévitable. Les possibilités offertes par RMG n'évitent pas le recours à un spécialiste informatique.

2.4. Quatrième étape : validation de la maquette papier

La validation s'est réalisée sur la représentation graphique. La difficulté de représenter avec précision l'enchaînement des différents écrans rend cette représentation difficile à suivre pour une personne non impliquée directement dans la conception. Par rapport à la spécification textuelle, la représentation graphique manque de rigueur mais donne une assez bonne idée du résultat final Nous pensons qu'il est difficile de se comprendre parfaitement, que l'on choisisse l'une ou l'autre forme de représentation. La réalisation de l'étape de médiatisation en travaillant par prototypage permet de se comprendre beaucoup mieux ; simplement par la démonstration du résultat final des unités d'interaction déjà réalisées.

2.5. Cinquième étape : médiatisation

Cette étape fait l'objet du chapitre 5. Nous avons choisi d'introduire des critères ergonomiques ainsi que des considérations pédagogiques plus matériels de manière à réaliser concrètement le scénario élaboré plus haut. La conception informatique semble être une étape très lourde dans l'environnement RMG. Comment concevoir des classes réutilisables dans le cadre du modèle ISO? RMG est-il l'outil approprié pour réaliser ce genre de choses? Nous pensons qu'il est idéal si pour les besoins du scénario, les mêmes objets sont multipliés. C'est notamment le cas dans bon nombre de simulations destinées à l'EAO des sciences physiques. Par exemple, une application simulant des particules dotées de masses et de vitesses différentes utilise plusieurs instances d'une même classe d'objet "particule". Dans le cas de FTAM, nous aurons généralement dans les didacticiels deux machines en relation. Leurs comportements sont différents car elles travaillent en utilisant un dialogue "maître-esclave". Les deux machines peuvent tour à tour jouer le même

rôle. Nous pensons que cette possibilité n'apporte rien d'un point de vue pédagogique. Il suffit de faire remarquer aux étudiants que "cela marche dans les deux sens".

2.6. Sixième étape : validation informatique

Nous ne ferons pas de remarque à ce sujet. Cette étape bien qu'indispensable n'apporte rien d'intéressant dans le cadre de ce mémoire.

2.7. Septième étape : tests

Nous avons testé notre logiciel sur quelques étudiants. Nous reprenons ici, les remarques le plus généralement faites. Dans l'ensemble, la critique est plutôt positive. Comme les étudiants connaissaient quelque peu le sujet, la séance d'apprentissage consiste surtout à vérifier que le logiciel se comporte de manière correcte par rapport aux connaissances que les étudiants possèdent. Le guidage semble bon vu que personne n'est resté bloqué. Assez curieusement, certains éléments ne sont pas perçus comme faisant partie de l'apprentissage. Citons le cas de l'objet Fournisseur FTAM qui apparaît ou disparaît lors de l'établissement ou lors de la terminaison d'une association (cfr. chapitre 4). Ce fait semble évident pour les étudiants et ils ne le remarquent même pas. Par contre certains éléments sont perçus comme faisant partie de la simulation alors que ce n'est pas le cas. L'exemple de l'avance pas à pas est édifiant à ce sujet. Nous avons décidé de découper la simulation en pas élémentaires. A chaque pas, l'utilisateur doit cliquer sur la souris pour poursuivre. Ce fait est parfois compris comme une action interne au modèle. La question posée fut la suivante : "Pourquoi, l'entité répondante doit-elle aller chercher l'indication? Celle-ci n'est-elle pas automatiquement délivrée?". D'autres remarques concernent l'avance pas à pas qui est ennuyeuse à la longue mais qui se justifie pleinement dans le cadre de l'apprentissage. Enfin, une remarque souligne que le fait de suivre l'enchaînement des actions à différents endroits de l'écran nécessite un temps d'adaptation. Il est vrai que la simulation ne se déroule pas en continu et que constamment, l'apprenant voyage de la représentation du modèle au centre de l'écran à l'explication du phénomène en bas de l'écran. Le fait que les objets Etats-Initiateur et Etats-Répondeur (voir chapitre 4) clignotent lorsqu'un changement leur est appliqué attire le regard et facilite le suivi de la simulation.

2.8. Huitième étape : correction

Bien évidemment, l'implémentation réalisée est trop peu conséquente pour vérifier qu'un véritable apprentissage a lieu et donc pour nous permettre de

modifier notre approche en conséquence. Cependant, les critiques positives que nous avons reçues encouragent à poursuivre dans cette voie.

2.9. Neuvième étape : diffusion

La diffusion des applications ne s'envisage pas pour le moment. Etant donné le peu de machines disponibles et leur inaccessibilité pour les étudiants, l'apprentissage ne peut se réaliser de manière acceptable. Dans le cadre actuel, seule une démonstration magistrale semble envisageable. Dès lors, l'ordinateur perd de sa puissance : il ne s'agit plus d'une manipulation du modèle par l'étudiant mais bien d'une démonstration réalisée par le professeur. Il n'y a plus d' "autoapprentissage". Dans cette optique, une implémentation différente, moins pédagogique et donc plus simple, pourrait être réalisée. La présence du professeur qui se sert du logiciel pour soutenir son cours, permet de remédier aux manques du logiciel. Dans ce cas, la présentation d'une vidéo plutôt que d'une simulation pourrait être tout aussi efficace bien qu'offrant moins de souplesse.

Le cas de RMG est assez spécial concernant la diffusion. Le prix du matériel limite le nombre de stations et donc freine la diffusion des applications développées sous RMG. A titre d'exemple, nous pouvons dire qu'il existe des simulations intéressantes dans le domaine de la chimie et de la physique. Cependant les étudiants des facultés des sciences ne peuvent actuellement pas profiter de ces programmes, le matériel étant utilisé pour la recherche en EAO à l'institut d'informatique.

2.10. Dixième étape : évaluation

Il est impossible, étant donné l'état d'avancement du projet, d'évaluer l'impact du logiciel sur la population. Tout au plus pouvons nous présenter en annexe une analyse du logiciel basée sur la grille d'évaluation **[DONN84]** et réalisée par nos soins [Annexe 2].

3. Règles intervenant dans la conception d'un didacticiel

Dans cette partie, nous considérons un certain nombre de critères pédagogiques qui nous ont servi de guide tout au long de la réalisation de notre didacticiel.

3.1. Simplicité et convivialité pour l'apprenant

La manipulation des informations par l'apprenant doit être facilitée par l'exécution rapide d'actions simples. Le nombre d'actions stériles doit être restreint au minimum. Seules les fonctionnalités qui produisent un effet doivent pouvoir être activées à un moment donné. Une attention particulière doit être portée à la rédaction des messages. Ceux-ci doivent être clairs et compréhensibles pour l'apprenant [BESN88]. Il ne faut pas perdre de vue que l'apprenant se trouve en situation d'autoformation. A tout moment les actions disponibles pour l'apprenant doivent être claires. Il est très rare qu'un apprenant qui reste "en panne" devant la machine prenne une initiative. Il a tendance à se méfier des réactions de l'ordinateur.

3.2. Le contrôle sur le dialogue

Le contrôle doit être le plus possible laissé à l'utilisateur. Dans une simulation, seules certaines actions seront disponibles en fonction de l'état du système simulé. Le champ d'action est donc restreint. Le rythme de lecture et de décodage de l'affichage est dépendant de chacun et doit être laissé à l'apprenant. Les temporisations sont limitées à l'affichage de graphiques ou à des animations. Même à ce niveau, on recommande d'offrir à l'utilisateur la possibilité de définir la vitesse des animations. A n'importe quel moment du dialogue, il est souhaitable que l'utilisateur puisse accéder à l'aide. Il doit avoir la possibilité d'interrompre son activité lorsqu'il le désire.

3.3. La présentation visuelle

L'écran se divise en zones. On peut travailler séparément sur chaque zone. Par exemple citons une zone boîte à outils, une zone pour les commentaires, une zone pour les consignes.

L'image qui apparaît à l'écran doit être structurée de manière à ce que l'utilisateur reconnaisse sans effort ses articulations et ses points forts. Les informations doivent être structurées visuellement pour faciliter la mémorisation. L'enchaînement des écrans doit aussi être parfaitement structuré. S'il s'agit de la même unité d'interaction, on évitera au maximum de changer d'écran. Dans le cas d'une simulation, il est évident que tout doit se passer sur le même écran. A partir du moment ou la simulation évolue vers une plus grande complexité, il faudra faire abstraction de la réalité et éliminer les choses accessoires pour se concentrer sur l'essentiel. La clé de la mémorisation devant l'écran réside dans le fait que toutes les informations nécessaires à la compréhension du sujet sont disponibles sur l'écran en même temps. En effet, l'utilisateur moyen peut mémoriser trois à sept éléments

d'information (loi de Miller [BESN88]). On voit combien la structuration des informations est importante si celles-ci sont nombreuses pour l'assimilation de la matière. Des expériences de psychologie qui portent sur le traitement de l'information montrent également que le contenu est mieux retenu lorsqu'il est intégré dans une histoire, relié par un fil narratif, et hiérarchisé.

3.4. L'écriture sur l'écran

Les polices de caractères sont importantes car elles n'ont pas toutes le même pouvoir d'expression. De même, les caractères gras ou italiques sont perçus différemment par le lecteur. Le texte ne doit pas prendre plus des deux tiers de l'écran. On ne lit pas un écran comme on lit une feuille de papier. Les lignes doivent comporter entre 50 et 65 caractères de manière à ne pas heurter le rythme de lecture.

3.5. Le graphisme

Dans un didacticiel, il est important d'associer visuel et texte de manière à faire intervenir les deux catégories d'interactions cérébrales (hémisphère gauche : lecture de textes, hémisphère droit : reconnaissance des symboles, des formes [BESN88]). La communication graphique peut prendre quatres formes. La représentation figurative donne une image semblable à l'objet. La représentation schématique n'a pas une signification immédiatement perceptible et nécessite une certaine réflexion. La représentation symbolique utilise des pictogrammes. Il s'agit de la représentation stylisée de l'objet. La représentation abstraite, enfin, nécessite un apprentissage car la signification donnée est de pure convention. Ce type de représentation ne doit être utilisée que lorsqu'on est sûr qu'il peut être compris par la population visée par le didacticiel.

3.6. La couleur

L'utilisation de la couleur suscite généralement de prime abord un enthousiasme non dissimulé. Mais, à la longue, elle révèle ses limites basées sur la physiologie et la psychologie. Mal utilisée, elle peut pervertir la représentation de l'information. Les études réalisées en pédagogie permettent d'énoncer un certains nombre de critères [BESN88] :

- les couleurs doivent être peu nombreuses. Cinq à sept couleurs sur le même écran constituent un maximum. Une exception à cette règle est faite si les objets codés par la couleur sont de grande taille car alors le décodage des couleurs par l'apprenant n'est pas compliqué.
- le contenu nécessite une hiérarchisation. L'utilisation de différentes couleurs n'est pas suffisant pour hiérarchiser le texte à lui seul.

- les couleurs choisies doivent bien se différencier entre elles.
- le codage social et culturel commun de la couleur doit être respecté. Le rouge et le vert sont peu perçus à la périphérie visuelle. Il ne faut pas utiliser ces couleurs pour coder un message. Si on ne peut faire autrement, on utilisera le clignotement pour attirer l'attention.
- les formes et les couleurs doivent être associées dans le codage. Il ne faut pas oublier que 6 à 10% de la population masculine ne peut pas percevoir certaines couleurs. Le daltonisme touche également 0,5% de la population féminine.

Les résultats de ces études montrent que :

- les couleurs attirent l'attention. Les adultes reconnaissent les images en couleur plus rapidement que les images en noir et blanc. Apparemment la couleur permet à la personne de structurer son travail. L'interprétation de l'information présentée ne diffère pas selon que l'on utilise la couleur ou pas.
- la couleur n'améliore pas les performances d'apprentissage des jeunes enfants. Il semble bien que ceux-ci ne décodent pas l'information couleur aussi rapidement que les adultes.
- la couleur aide à mémoriser certains détails mais a peu d'influence sur le contenu principal.
- la couleur a surtout un impact émotionnel sur l'apprenant. L'utilisation de la couleur ne permet pas d'apprendre mieux. Par contre, les sujets testés apprennent plus agréablement lorsqu'ils disposent de la couleur. Celle-ci joue donc un rôle motivationnel important.
- les écrans très contrastés sont mieux perçus. La présentation d'un texte de couleur sombre sur fond clair facilite la reconnaissance et la mémorisation de l'information.

4. Les avantages communs aux utilisations pédagogiques de l'ordinateur

4.1. La motivation

Chacun connaît l'engouement qu'éprouvent les jeunes pour l'ordinateur. Mais peut-être ne s'agit-il que d'un simple attrait pour la nouveauté ou du plaisir de manipuler des périphériques sophistiqués. Il faut bien avouer que l'on manque de recul pour pouvoir en juger. Mais après tout, que la fascination exercée par la machine s'explique par de bonnes ou de mauvaises raisons, l'important n'est-il pas qu'elle entraîne un regain de motivation pour l'étude, même lorsqu'elle s'avère exigeante ?

4.2. La participation à l'apprentissage

Des études scientifiques ont montré que le rendement des individus est fortement influencé par leur participation au processus d'apprentissage [BESN88]. Les étudiants savent bien d'ailleurs que, s'ils font partie d'un effectif réduit, leurs performances s'améliorent souvent en proportion du taux élevé de leurs interventions. Les professeurs, de leur côté, s'échinent à susciter l'activité de leurs classes. Mais ils ont beau faire : on évalue généralement le temps d'activité réelle de l'étudiant à quelques minutes par heure de cours. L'ordinateur, n'entrant en action que sous l'effet d'une sollicitation, accule en quelque sorte l'utilisateur à intervenir et l'implique d'office dans son propre apprentissage. Le temps passé à la tâche se trouve du même coup optimalisée.

4.3. L'évaluation immédiate

Il a également été démontré qu'une évaluation est d'autant plus efficace qu'elle suit immédiatement l'apprentissage jugé. Au bout de quelques jours, elle peut devenir inopérante. Mais réagir sans délai n'est pas toujours à la portée du professeur. L'allure vertigineuse à laquelle l'ordinateur exécute des instructions et réagit à l'introduction d'une réponse permet à l'étudiant de juger immédiatement de la qualité de son intervention.

4.4. Une aseptisation affective

Le tête-à-tête avec une machine a nécessairement des répercussions affectives. Pour les uns, l'absence du professeur (en cas d'utilisation de programmes d'enseignement proprement dit) obligerait l'étudiant à se débrouiller tout seul et l'éduquerait à l'autonomie; pour les autres, le fait d'avoir à commander, ne serait-ce qu'à une machine, permettrait aux plus timides de s'affirmer. Bref, celui qui éprouverait quelques difficultés à nouer des relations avec ses professeurs ou avec ses camarades se trouverait bien de l'emploi de l'ordinateur et verrait ses apprentissages rentabilisés d'autant. A l'inverse, d'autres encore font remarquer que, placé au centre de son apprentissage, l'étudiant n'est pas nécessairement incité à développer sa capacité d'écoute des opinions et des sentiments d'autrui, ni à se définir par rapport à l'autre. Les méthodes individualisantes développant l'individualisme, la formation par et dans la classe est indispensable pour en réduire les effets.

4.5. Le droit à l'erreur

Généralement, un professeur qui enseigne de nouvelles notions ou de nouvelles compétences à ses étudiants, s'arrange pour leur éviter les causes

d'erreur; si celles-ci se produisent tout de même, le plus souvent, il les sanctionne. A l'inverse, procédant par d'incessantes interrogations ou attendant une intervention, l'ordinateur provoque presque nécessairement les tâtonnements, les faux-pas. Mais il s'agit moins, dans ce cas, d'erreurs que de tentatives pour progresser ; tentatives parfois malhabiles mais révélatrices du raisonnement poursuivi et donc utiles. L'emploi de l'ordinateur pourrait ainsi entraîner une modification du statut de l'erreur.

Chapitre 4 : Scénario d'une application FTAM

1. Description et objectifs du scénario

Le scénario FTAM que nous proposons permettra à l'utilisateur de se familiariser avec la plupart des primitives FTAM. Nous n'aborderons pas les paramètres des primitives dans leur détail car cet aspect du problème n'est pas fondamental pour l'enseignement que nous voulons donner. Notre scénario s'attachera également à montrer le fonctionnement d'une machine à états simple. Chaque entité du modèle ISO possède à tout moment un état qui la caractérise (nous l'appellerons "état courant"). Les primitives ont pour effet de modifier cet état. En fonction de l'état courant, seuls certains états caractérisant l'entité FTAM sont accessibles, et donc seules certaines primitives sont acceptables. C'est en vérifiant l'état courant (et les états accessibles à partir de celui-ci) que nous pourrons contrôler le bien-fondé d'une demande de l'utilisateur du logiciel.

Nous faisons remarquer le traitement spécial que nous avons décidé pour la primitive **F-USER-ABORT request**. Cette primitive peut être utilisée à n'importe quel moment par l'utilisateur dès qu'une association est établie ou est en cours d'établissement. Il aurait donc fallu, pour être complet, ajouter l'état suivant **Idle** comme étant toujours acceptable pour l'état courant (excepté s'il est lui-même **Idle**) simplement par utilisation de la primitive **F-USER-ABORT request**. Nous avons décidé de l'omettre pour des raisons de clarté essentiellement.

Pour expliquer la simulation que nous voulons réaliser, nous avons adopté une approche "objets". En effet, lors de la simulation, différents objets seront manipulés. Ils apparaîtront et disparaîtront en fonction des requêtes demandées par l'utilisateur du didacticiel. Nous avons donc décrit dans un premier temps les objets dont nous avons besoin ainsi que les méthodes associées à ces objets. Par méthode associée à un objet, nous entendons "moyen d'interaction" sur cet objet. Par exemple, une méthode associée à l'objet Messages (décrit au point 5.1) s'intitule "Affiche". Lorsque nous voulons qu'un message à destination de l'apprenant lui parvienne, nous spécifions dans le scénario

Messages → Affiche (donnée du message).

Cela signifie "le message *donnée du message* doit apparaître dans l'objet Messages". Ce formalisme s'inspire du cours "METHODOLOGIE DE DEVELOPPEMENT DE LOGICIEL" **[DUB090]**. Les méthodes sont spécifiées en précondition et postcondition. Il n'est pas toujours possible de réaliser la

spécification dans un langage abstrait de type *mathématique*. Ceci est dû au caractère graphique de l'application et des méthodes que nous décrivons. Nous avons donc spécifier en *français* les méthodes relevant du graphisme pur.

La seconde étape de la spécification du scénario consiste à se servir des objets et des méthodes que nous aurons décrits pour construire le scénario proprement dit. Cette étape est réalisée au point 7.

Afin d'éclaircir la lecture de ce chapitre, nous avons utilisé différentes polices de caractère associées à des intentions différentes. Outre le style standard (dans lequel ce paragraphe est écrit), nous avons utilisé un style différent pour les spécifications et un autre encore pour les remarques pédagogiques que nous souhaitions formuler en accompagnement de la spécification des méthodes. Nous espérons que cela facilitera la lecture et la compréhension de ce chapitre.

2. Description des objets composants la scène

La figure 4.1 représente les différents objets qui peuvent apparaître à l'écran au cours de la simulation.

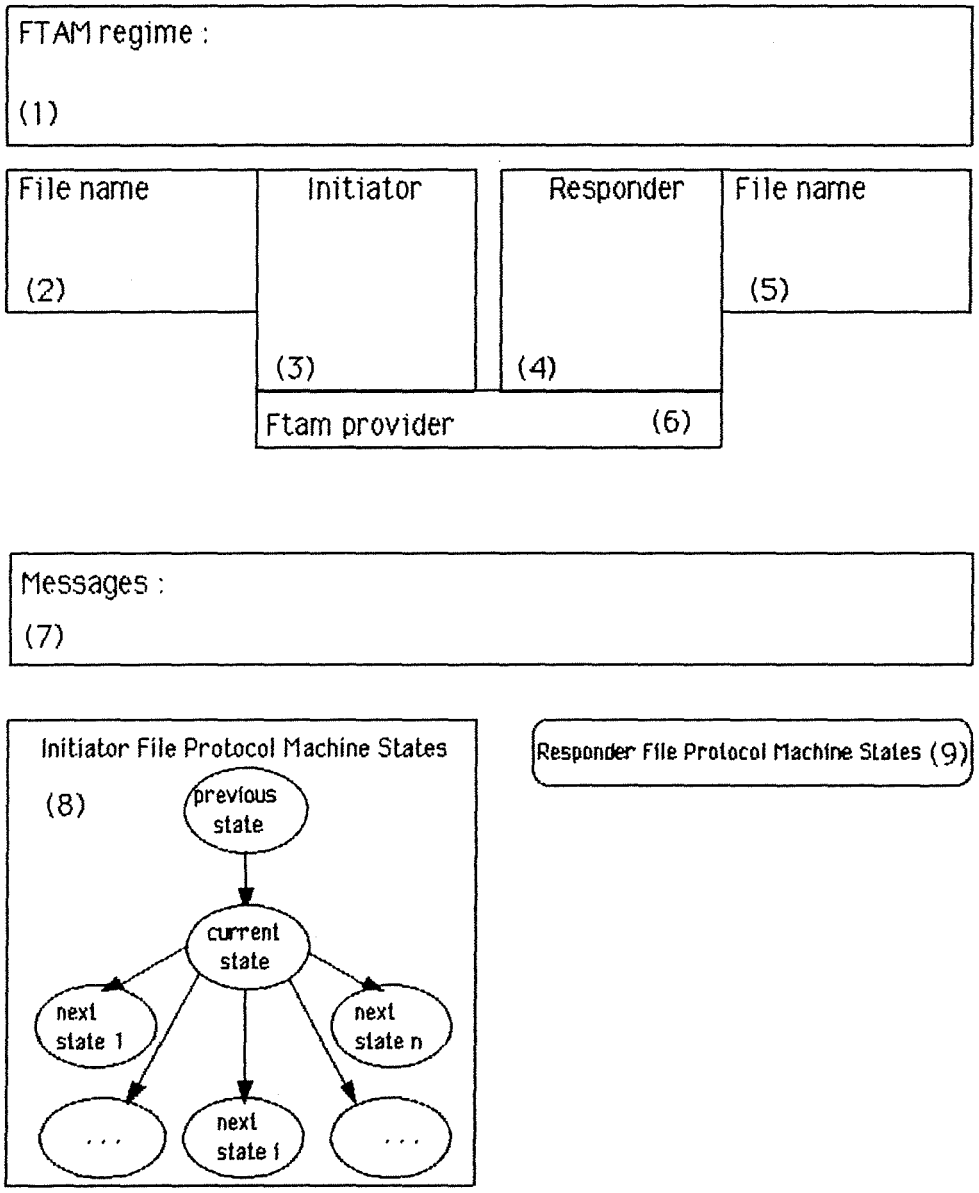


Figure 4.1 : représentation schématique des objets de la scène.

Nous distinguons les objets suivants :

(1) Objet Régime : cet objet affiche le régime FTAM en cours (**FTAM regime**);

(2) Objet Fichier-Initiateur : cet objet permet d'afficher le fichier du côté initiateur, les attributs et la structure du fichier;

(3) Objet Agent-Initiateur : cet objet affiche les primitives FTAM et permet à l'utilisateur de choisir les primitives (**Initiator**);

(4) Objet Agent-Répondeur : cet objet affiche les primitives FTAM et permet à l'utilisateur de renvoyer une réponse positive ou négative à certaines primitives (**Responder**);

(5) Objet Fichier-Répondeur : cet objet permet d'afficher le fichier du côté répondeur, ses attributs et sa structure;

(6) Objet Fournisseur : cet objet affiche l'envoi des primitives d'un agent FTAM à l'autre. Il symbolise le transfert des PDU (**FTAM provider**);

(7) Objet Messages : cet objet affiche les messages à destination de l'utilisateur (**Messages**);

(8) Objet Etat-Initiateur : cet objet affiche les états de la machine de protocole de fichier du côté initiateur. Plus particulièrement, l'état précédent (**previous state**), l'état courant (**current state**) et les états suivants (**next states**) de la machine sont affichés. Par état précédent, nous entendons l'état qu'occupait la machine avant de passer à l'état courant. Par états suivants, nous désignons les états accessibles à partir de l'état courant. Sur la figure 4.1 l'objet est représenté sous sa forme développée (les états sont visibles) (**Initiator File Protocol Machine States**);

(9) Objet Etat-Répondeur : cet objet affiche les états de la machine de protocole de fichier du côté répondeur. Sur la figure 4.1 l'objet est représenté sous forme d'icône (les états sont cachés) (**Responder File Protocol Machine States**).

3. Spécification des structures de données

Nous aurons besoin d'un certain nombre de structures de données afin de spécifier les paramètres utilisés par les méthodes ou renvoyés par les méthodes.

Type énuméré.

Le type énuméré définit un ensemble exhaustif de valeurs qui peuvent être prises par les variables de ce type.

Exemple : type_énum=(valeur1,valeur2). Si une variable est de type *type_énum* alors elle peut uniquement prendre les valeurs *valeur1* ou *valeur2*.

Type séquence.

Le type séquence définit un ensemble ordonné de valeurs d'un type quelconque.

Exemple : type_séq=SEQ[INTEGER] désigne un ensemble ordonné de valeurs entières. Une variable de type *type_séq* peut prendre la valeur *[1,5,7]* par exemple.

Type produit cartésien.

Le type produit cartésien désigne un ensemble structuré de valeurs de types quelconques.

Exemple : type_cp=CP[STRING,INTEGER] désigne un couple constitué d'une chaîne de caractères et d'un entier. Une variable de type *type_cp* peut prendre la valeur *["TOTO",5]* par exemple.

4. Spécification des types de données

Les différents types de données dont nous aurons besoin pour décrire le scénario sont les suivants :

- BOOL : le type BOOL désigne une valeur booléenne. Il s'agit en fait d'un type énuméré. Les valeurs admises sont

BOOL={TRUE, FALSE}.

- STRING : le type STRING désigne une chaîne de caractères de longueur quelconque.

- FP : le type FP est un type énuméré désignant une primitive FTAM. Les valeurs admises (voir tableau 4.1 pour la signification) sont

FP={F_INI_RQ, F_INI_RS+, F_INI_RS-, F_TERM_RQ, F_TERM_RS, F_UABT_RQ, F_SEL_RQ, F_SEL_RS+, F_SEL_RS-, F_CRE_RQ, F_CRE_RS+, F_CRE_RS-, F_DESEL_RQ, F_DESEL_RS, F_DEL_RQ, F_DEL_RS, F_RDATT_RQ, F_RDATT_RS, F_CHATT_RQ, F_CHATT_RS, F_OPN_RQ, F_OPN_RS+, F_OPN_RS-, F_CLO_RQ, F_CLO_RS, F_LOC_RQ, F_LOC_RS, F_ERA_RQ, F_ERA_RS, F_RD_RQ, F_WR_RQ, F_DATA_RQ, F_EDAT_RQ, F_ETRSF_RQ, F_ETRSF_RS}.

- ETAT : le type ETAT désigne un état de la machine de protocole de fichier. Il s'agit d'un type énuméré. Les valeurs admises sont

ETAT={Idle, Initialize pending, Initialized, Terminate pending, Select pending, Selected, Deselect pending, Create pending, Delete pending, Read attrib pending, Change attrib pending, Open pending, Data Transfer Idle, Close pending, Read, Write, Read ending, Write ending, Transfer ending read, Transfer ending write, Locate pending, Erase pending}

Cette terminologie fait partie de la norme FTAM ; le lecteur interessé trouvera en annexe les schémas de transitions d'états qui nous préoccupent (annexe 3).

Noms abrégés	Noms des primitives de service	Rôle des primitives
F_INI_RQ	F_INITIALIZE request	L'initiateur demande la création d'un régime FTAM en reliant les utilisateurs aux fournisseurs du service de fichiers dans une association de niveau application.
F_INI_RS+	F_INITIALIZE response positive	Le répondeur accepte la création du régime d'association FTAM.
F_INI_RS-	F_INITIALIZE response negative	Le répondeur refuse la création du régime d'association FTAM.
F_TERM_RQ	F_TERMINATE request	L'initiateur supprime le régime d'association FTAM.
F_TERM_RS	F_TERMINATE response	Le répondeur confirme la fin de l'association entre les deux partenaires.
F_UABT_RQ	F_U_ABORT request	L'utilisateur du service FTAM détruit inconditionnellement l'association FTAM.
F_SEL_RQ	F_SELECT request	L'initiateur sélectionne un fichier existant.
F_SEL_RS+	F_SELECT response positive	Le répondeur accepte la sélection du fichier.
F_SEL_RS-	F_SELECT response negative	Le répondeur refuse la sélection. Le fichier est indisponible pour l'initiateur.
F_CRE_RQ	F_CREATE request	L'initiateur demande la création d'un fichier.
F_CRE_RS+	F_CREATE response positive	Le répondeur accepte la création du fichier. Il confirme cette création. Le fichier est sélectionné.
F_CRE_RS-	F_CREATE response negative	Le répondeur refuse la création du fichier.
F_DESEL_RQ	F_DESELECT request	L'initiateur désélectionne un fichier existant.
F_DESEL_RS	F_DESELECT response	Le répondeur confirme la désélection du fichier.
F_DEL_RQ	F_DELETE request	L'initiateur abandonne une sélection existante, et demande la destruction du fichier sélectionné.
F_DEL_RS	F_DELETE response	Le répondeur confirme la destruction du fichier.
F_RDATT_RQ	F_READ_ATTRIB request	L'initiateur demande une consultation des attributs du fichier sélectionné.
F_RDATT_RS	F_READ_ATTRIB response	Le répondeur renvoie la liste des attributs du fichier demandé en lecture.
F_CHATT_RQ	F_CHANGE_ATTRIB request	L'initiateur souhaite modifier les attributs du fichier sélectionné.
F_CHATT_RS	F_CHANGE_ATTRIB response	Le répondeur transmet les attributs ayant subi une modification.
F_OPN_RQ	F_OPEN request	L'initiateur établit le contexte de présentation. Il demande l'ouverture du fichier.
P_OPN_RS+	F_OPEN response positive	Le répondeur accepte l'ouverture du fichier sélectionné.
F_OPN_RS-	F_OPEN response negative	Le répondeur refuse l'ouverture du fichier sélectionné.
F_CLO_RQ	F_CLOSE request	L'initiateur demande la libération du contexte établi par le service d'ouverture du fichier.
F_CLO_RS	F_CLOSE response	Le répondeur confirme la fermeture du fichier. Le fichier est toujours sélectionné.

F_LOC_RQ	F_LOCATE request	L'initiateur spécifie l'identité d'un FADU qui doit être localisé par le répondeur.
F_LOC_RS	F_LOCATE response	Le répondeur confirme l'action de localisation d'un FADU entreprise sur le fichier.
F_ERA_RQ	F_ERASE request	L'initiateur efface un FADU.
F_ERA_RS	F_ERASE response	Le répondeur confirme l'effacement d'un FADU.
F_RD_RQ	F_READ request	L'initiateur demande au répondeur de lire certaines données du fichier et de les lui transmettre.
F_WR_RQ	F_WRITE request	L'initiateur demande au répondeur d'écrire les données qui seront envoyées, dans le fichier.
F_DATA_RQ	F_DATA request	L'expéditeur envoie des données à lire ou à écrire.
F_EDAT_RQ	F_END_DATA request	L'expéditeur marque la fin de l'envoi des données.
F_ETRSF_RQ	F_TRANSFER_END request	L'initiateur déclare que le transfert de données est terminé.
F_ETRSF_RS	F_TRANSFER_END response	Le répondeur confirme la fin du transfert de données.

Tableau 4.1 : primitives FTAM utilisées dans le scénario.

5. Spécification des objets de la scène

5.1. Objet Messages

L'objet Messages est une simple zone d'écran où tous les messages vont apparaître.

5.1.1. Structure de l'objet

Nous avons distingué trois types de messages. D'une part, nous trouvons les messages d'accompagnement (figure 4.2) qui sont destinés à commenter les changements qui s'opèrent lors de la simulation. Ils accompagnent l'exécution d'une animation pour renseigner l'utilisateur sur son évolution. D'autre part, nous avons les messages d'aide (figure 4.3). Ils sont utilisés lorsque l'utilisateur doit entreprendre une action. Par exemple, le message "*Sélectionnez une primitive*" invite l'utilisateur à choisir une primitive de requête FTAM. Enfin, il nous reste les messages d'erreur (figure 4.4). Ceux-ci apparaissent lorsque l'utilisateur entreprend une action incorrecte. Le message indique la raison de l'échec et la manière d'y remédier.

Les messages bien que de type différent doivent apparaître au même endroit sur l'écran de manière à ne pas perturber l'apprenant. Nous avons choisi pour les distinguer d'utiliser trois couleurs de fond différentes et des polices de caractères distinctes. Les messages d'erreur sont accompagnés

d'un signal sonore. La simulation étant bloquée lorsqu'un message d'aide ou d'erreur est affiché, il est primordial d'attirer l'attention de l'apprenant pour qu'il lise le message et puisse remédier à la situation en connaissance de cause.

5.1.2. Méthodes associées

- Affiche(message_d'accompagnement)

Cette méthode affiche un message d'accompagnement.

message_d'accompagnement = STRING

Précondition : /.

Postcondition :

L'objet Messages est dans l'état repris à la figure 4.2. Nous constatons que le texte *message_d'accompagnement* est affiché.

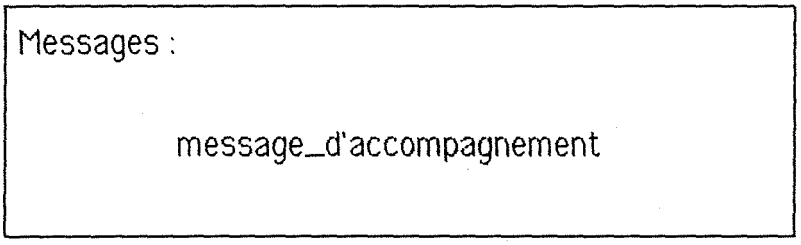


Figure 4.2 : affichage d'un message d'accompagnement.

- Affiche-Aide(message_d'aide)

Cette méthode affiche un message d'aide dans la zone de message.

message_d'aide = STRING

Précondition : /.

Postcondition :

L'objet Messages est dans l'état repris à la figure 4.3. Le texte *message_d'aide* est affiché.

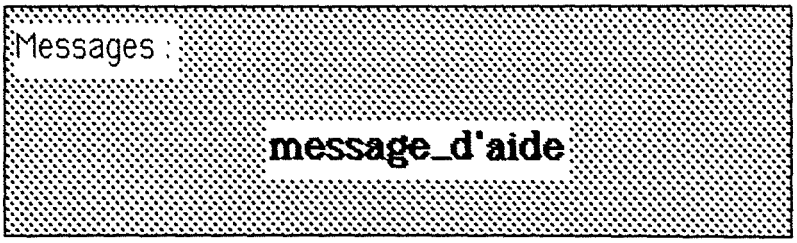


Figure 4.3 : affichage d'un message d'aide.

- **Affiche-Erreur(message_d'erreur)**

Cette méthode affiche un message d'erreur.

message_d'erreur = STRING

Précondition : /.

Postcondition :

L'objet Messages est dans l'état repris à la figure 4.4. Nous constatons que le texte *message_d'erreur* est affiché.

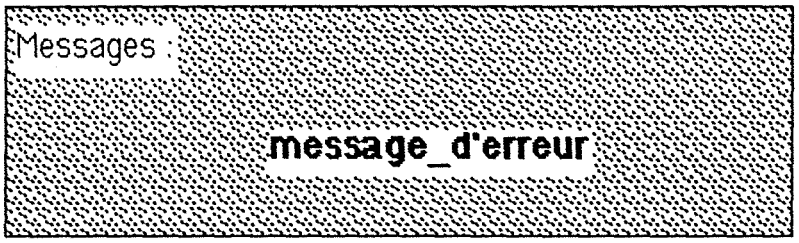


Figure 4.4 : affichage d'un message d'erreur.

5.2. Objet Régime

L'objet Régime est une zone d'écran où les différents régimes FTAM se succèdent. Cela permet de montrer l'enchaînement des régimes et d'identifier le régime courant.

5.2.1. Structure de l'objet

Les différents états possibles de l'objet Régime sont représentés sur les figures 4.5 à 4.9. Décrivons par exemple la figure 4.8 (Etat_3). Nous remarquons que le régime courant est affiché en surimpression sur les régimes précédents. Cette représentation permet de symboliser l'encapsulage des régimes.

L'objet Régime est à tout moment dans un des états spécifiés.

Régime → Etat = {Etat_0, Etat_1, Etat_2, Etat_3, Etat_4}

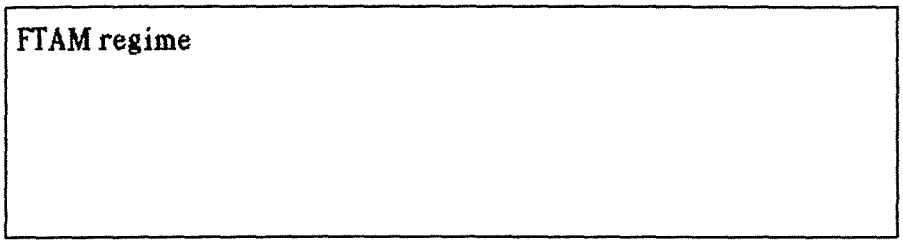


Figure 4.5 : Etat_0 : pas de régime FTAM.

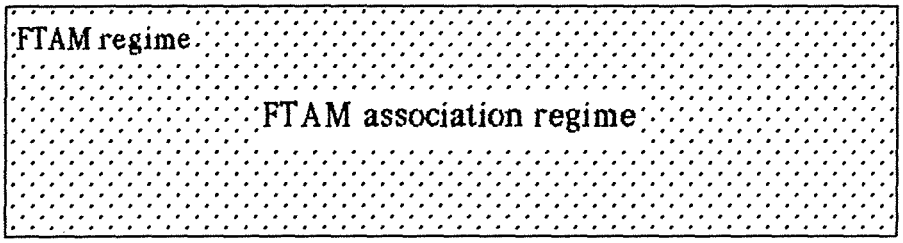


Figure 4.6 : Etat_1 : régime d'association FTAM.

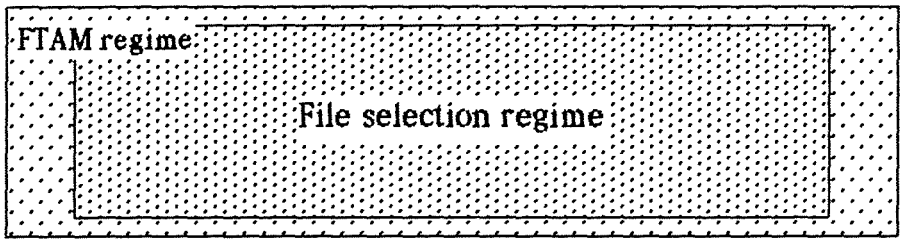


Figure 4.7 : Etat_2 : régime de sélection de fichier.

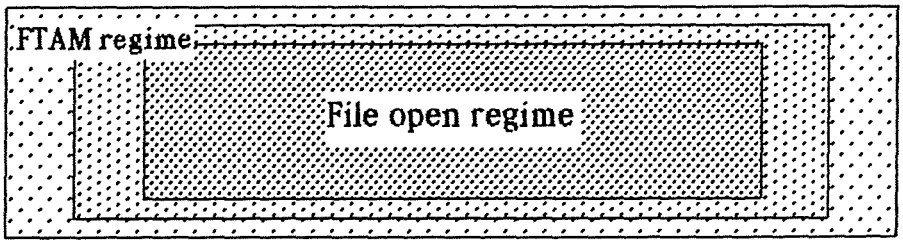


Figure 4.8 : Etat_3 : régime d'ouverture de fichier.

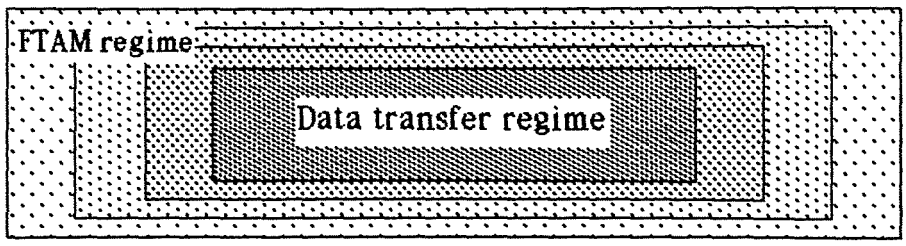


Figure 4.9 : Etat_4 : régime de transfert de données.

Cette méthode de représentation modélise l'enchaînement des régimes tel qu'il est spécifié dans les normes FTAM. Nous faisons l'hypothèse que ce symbolisme permet à l'apprenant de se rendre compte de cet état de fait. Le régime courant est seul affiché. L'utilisateur visualise un seul régime, il n'y a pas de confusion possible. De plus, afficher uniquement ce régime évite la surcharge de l'écran.

Pour montrer l'empilement des régimes, nous avons associé une couleur de fond à chaque régime. Cela permet de se rendre compte que pour entrer dans le régime de sélection de fichier par exemple, il faut être auparavant dans le régime d'association FTAM (cfr. figure 4.7).

5.2.2. Méthodes associées

- Entrer-Régime

Cette méthode permet d'entrer dans le régime suivant s'il existe.

Précondition :
Régime → Etat not = Etat_4.

Postcondition :
Case Régime → $Etat_{PRE}$ of
  Etat_0 : Régime → Etat=Etat_1
  Etat_1 : Régime → Etat=Etat_2
  Etat_2 : Régime → Etat=Etat_3
  Etat_3 : Régime → Etat=Etat_4
Endcase
and Messages→Affiche("Un nouveau régime est établi.")

- Sortir-Régime

Cette méthode permet de revenir au régime précédent lorsque l'activité liée au régime en cours se termine.

Précondition :

Régime → Etat not = Etat_0.

Postcondition :

Case Régime → $Etat_{PRE}$ of

Etat_1 : Régime → Etat=Etat_0

Etat_2 : Régime → Etat=Etat_1

Etat_3 : Régime → Etat=Etat_2

Etat_4 : Régime → Etat=Etat_3

Endcase

and Messages→Affiche("L'ancien régime est rétabli.")

- Hauteur = Etat

Cette méthode renvoie l'état de l'objet régime.

Etat : INTEGER.

Précondition : /

Postcondition :

Case Régime→Etat of

Etat_0 : Etat=0

Etat_1 : Etat=1

Etat_2 : Etat=2

Etat_3 : Etat=3

Etat_4 : Etat=4

Endcase

5.3. Objet Fichier-Initiateur

L'objet Fichier-Initiateur est caractérisé par un nom, des attributs, une structure de données et des données.

5.3.1. Structure de l'objet

L'apparition des différents objets symbolise l'information disponible pour l'agent initiateur. La figure 4.10.a donne l'information disponible après sélection d'un fichier. La lecture des attributs du fichier côté répondeur se traduit par l'apparition de la boîte *Attributes* (figure 4.10.b). Lors du changement des attributs du fichier, une partie de la boîte *Attributes* clignote (figure 4.10.b et 4.11 alternativement). Ce clignotement signifie qu'une action est entreprise sur les attributs (modification de ceux-ci). L'ouverture du

fichier se traduit par la présentation de la zone d'affichage du contenu. Aucune information concernant le contenu n'est disponible après l'ouverture, c'est pourquoi cette zone est vide. Dans cette zone, la structure et le contenu du fichier seront affichés au fur et à mesure du transfert de données (figure 4.12).

Le fichier sera identifié par son nom :

Fichier-Initiateur → nom = STRING.

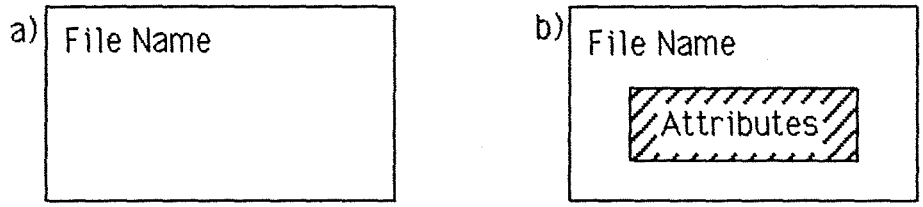


Figure 4.10 : les deux représentations possibles de l'objet Fichier-Initiateur :
a) sans ses attributs;
b) avec ses attributs.

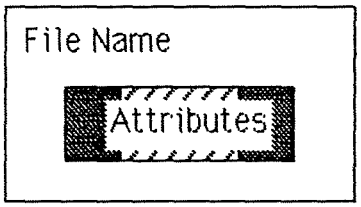


Figure 4.11: la boîte d'attributs clignote.

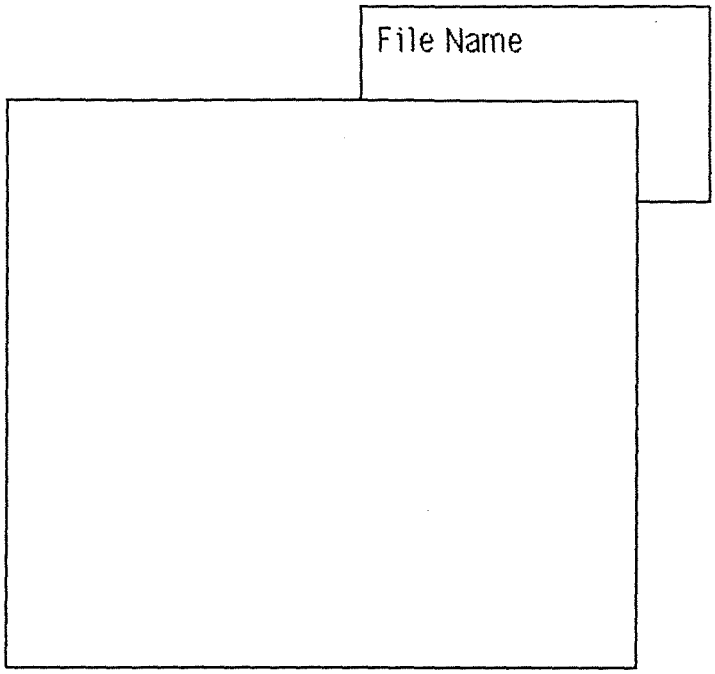


Figure 4.12 : la zone d'affichage du contenu côté Fichier-Initiateur est présente.

Les attributs du fichiers sont représentés par une boîte de couleur. Lorsque des changements sont effectués sur ces attributs, nous faisons clignoter la zone relative à ces changements. Nous attirons l'attention sur l'action qui se déroule. Un message d'accompagnement précise à l'apprenant le symbolisme du clignotement.

Pendant le régime d'ouverture d'un fichier, les attributs de ce dernier ne peuvent plus être changés ou lus. C'est pourquoi nous avons décidé de ne plus les montrer. De cette manière, l'utilisateur n'a à l'écran que des objets sur lesquels il est susceptible d'apporter une modification.

5.3.2. Méthodes associées

- Affiche-Fichier(nom)

Cette méthode permet d'afficher l'objet Fichier-Initiateur lorsque celui-ci est sélectionné par l'utilisateur. Le paramètre *nom* contient le nom du fichier qui sera affiché.

Précondition : /

Postcondition :

On affiche l'objet Fichier-Initiateur représenté à la figure 4.10.a. Le champ **File name** contient la valeur *nom*. Fichier-Initiateur → nom = *nom*.

- Affiche-Attributs

Cette méthode permet d'afficher les attributs lorsqu'ils ont été lus du côté répondeur et envoyés du côté initiateur.

Précondition :

L'objet Fichier-Initiateur est affiché (figure 4.10.a)

Postcondition :

On affiche les attributs du fichier (figure 4.10.b).

- Clignote-Attributs

Cette méthode montre qu'une action est entreprise sur les attributs.

Précondition :

Les attributs de l'objet Fichier-Initiateur sont affichés.

Postcondition :

Les attributs de l'objet Fichier-Initiateur sont affichés.

Règles de traitement :

Une animation fait clignoter une partie des attributs pendant quelques secondes, alternant ainsi les figures 4.10.b et 4.11.

- Affiche-Contenu

Cette méthode affiche la zone qui permettra de visualiser les données reçues par l'agent initiateur (cas du **F-READ request**).

Précondition :

L'objet Fichier-Initiateur est affiché.

Postcondition :

On affiche la zone où l'on représentera la structure des données (figure 4.12).

- **Efface**

Cette méthode efface l'objet Fichier-Initiateur ainsi que les attributs de cet objet si ceux-ci étaient affichés.

Précondition :

L'objet Fichier-initiateur est affiché. Ses attributs peuvent l'être.

Postcondition :

L'objet Fichier-initiateur est effacé, les attributs sont effacés (s'ils étaient présents avant l'application de la méthode).

- **Efface-Contenu**

Cette méthode permet d'effacer la zone d'affichage de la structure du fichier et son contenu. La méthode rétablit l'affichage de l'objet Fichier-Initiateur dans l'état où il se présentait en régime de sélection de fichier (c'est-à-dire avant l'apparition de la zone d'affichage du contenu).

Précondition :

La structure des données de l'objet Fichier-initiateur est affichée (figure 4.12 avec une structure de fichier éventuelle).

Postcondition :

La structure des données de l'objet Fichier-initiateur est effacée, l'objet Fichier-Initiateur est affiché (figure 4.10.a). Les attributs sont affichés (figure 4.10.b) s'ils l'étaient avant d'être masqués par la fenêtre de représentation du contenu.

Les méthodes relatives au transfert de fichier font l'objet d'un paragraphe spécial (cfr. point 6).

5.4. Objet Agent-Initiateur

L'objet Agent-Initiateur (figure 4.13 et figure 4.1 (3) **Initiator**) modélise l'agent FTAM initiateur. Son rôle est de demander des services à l'objet Fournisseur. Ces services constituent les primitives FTAM de requêtes, elles sont initialisées par l'utilisateur. Les résultats des services sont communiqués à l'agent par des primitives FTAM d'indications ou de confirmations.

5.4.1. Structure de l'objet

L'agent initiateur permet d'afficher une primitive FTAM. Il peut s'agir d'une primitive de requête sélectionnée par l'utilisateur, d'une primitive de confirmation ou d'une primitive d'indication (**F-DATA indication** et **F-DATA-END indication** dans le cas d'une lecture du fichier). La figure 4.13.a montre l'objet initiateur lorsqu'aucune primitive FTAM n'est affichée. La figure 4.13.b illustre le cas contraire.

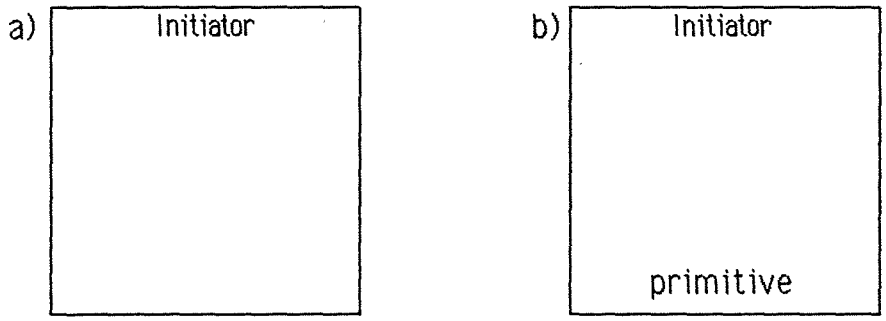


Figure 4.13 : a) l'agent initiateur sans message;
b)l'agent initiateur avec un message affiché.

5.4.2. Méthodes associées

- Affiche(primitive)

Cette méthode affiche le message *primitive* passé en paramètre dans l'objet Agent-Initiateur.

primitive = STRING.

Précondition : /.

Postcondition :

Après l'exécution de la méthode, l'objet se trouve dans l'état repris à la figure 4.13. b. Nous constatons que le message *primitive* est affiché.

- Efface

Cette méthode efface le message affiché dans l'objet Agent-Initiateur.

Précondition : /.

Postcondition :

Après l'exécution de la méthode, l'objet se trouve dans l'état repris à la figure 4.13.a . Nous constatons que tous les messages ont été effacés.

- **Accepte-Choix=choix**

Cette méthode renvoie la primitive choisie par l'utilisateur.

choix =FP.

Précondition : /.

Postcondition :

Le paramètre *choix* contient la primitive de demande de service FTAM sélectionnée par l'utilisateur.

Nous ne précisons pas comment le choix est fait à ce stade de la spécification. Le mode de représentation pour effectuer le choix (menu déroulant, menu fixe, ligne de commande, ...) relève de l'implémentation. Nous aborderons ce point au chapitre suivant.

- **Lire-nom=nom_transmis**

Cette méthode renvoie le nom choisi par l'utilisateur. Par exemple cette méthode permet la saisie du nom du fichier sur lequel l'utilisateur souhaite travailler.

nom-transmis = STRING.

Précondition : /.

Postcondition :

Le paramètre *nom_transmis* contient le nom choisi par l'utilisateur.

5.5. Objet Agent-Répondeur

L'objet Agent-Répondeur (figure 4.1 (4) **Responder**) modélise l'agent FTAM répondeur. Son rôle est d'exécuter les demandes de services acheminées par l'objet Fournisseur. Ces services, demandés par l'agent FTAM initiateur, sont communiqués sous forme de primitives FTAM d'indications. L'exécution du service peut nécessiter une action sur le système de fichiers. Le résultat de l'exécution est communiqué à l'entité paire par des primitives FTAM de réponses ou d'indications.

5.5.1. Structure de l'objet

Une représentation similaire à celle de l'objet Agent-Initiateur est adoptée.

5.5.2. Méthodes associées

- Affiche(primitive)

Cette méthode affiche le message *primitive* passé en paramètre.

primitive = STRING.

Précondition : /.

Postcondition :

Après l'exécution de cette méthode, le message *primitive* est affiché dans l'objet de manière similaire à la figure 4.13.b

- Efface

Cette méthode efface le message affiché dans l'objet Agent-Répondeur.

Précondition : /.

Postcondition :

Les messages affichés dans l'agent répondeur ont été effacés.

- Accepte-Choix=choix

Cette méthode renvoie la réponse *choix* de l'utilisateur.

choix ={"Y","N"}.

Précondition : /.

Postcondition :

le paramètre *choix* contient la réponse de l'utilisateur. Le sens attribué à cette réponse dépend du contexte.

5.6. Objet Fichier-Répondeur

L'objet Fichier-Répondeur est caractérisé par un nom, des attributs, une structure de données et des données.

5.6.1. Structure de l'objet

En ce qui concerne les attributs de l'objet Fichier-Répondeur rien ne diffère de la représentation adoptée pour l'objet Fichier-Initiateur. Par contre, la zone d'affichage du contenu du fichier représente la structure et les données du fichier sous la forme adoptée dans la norme FTAM (représentation

hiérarchique stricte). Dès l'ouverture d'un fichier la structure est accessible (figure 4.14).

Fichier-Répondeur → Nom = STRING.

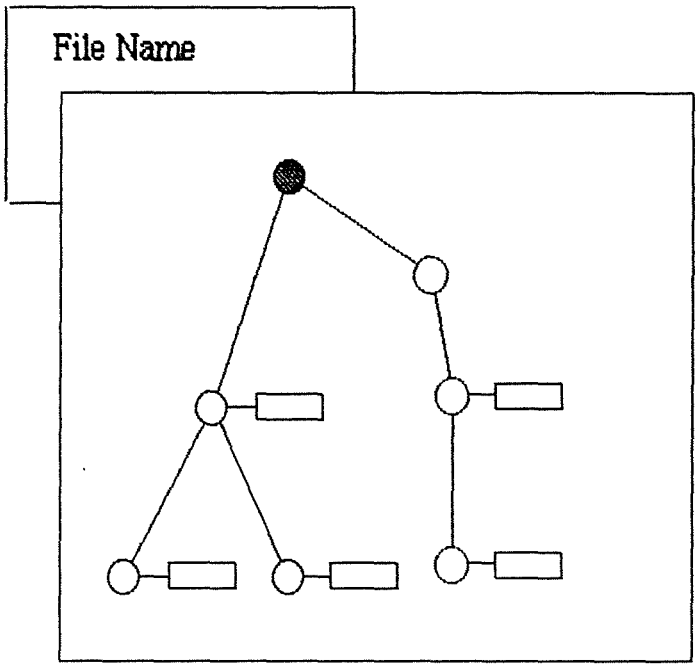


Figure 4.14 : la zone d'affichage du contenu du côté répondeur.

Notons qu'après l'ouverture du fichier, le noeud racine est mis en évidence par l'utilisation d'une couleur différente. Le noeud en évidence renseigne l'apprenant sur le FADU sélectionné. Les actions que l'apprenant entreprendra porteront sur ce FADU. Nous poursuivons notre politique de représenter à tout instant les objets manipulables et seulement ceux-là.

5.6.2. Méthodes associées

- Affiche-Fichier(nom_fichier)

Cette méthode affiche l'objet Fichier-Répondeur.

nom_fichier = STRING.

Précondition : /

Postcondition :

On affiche l'objet Fichier-Répondeur comme représenté à la figure 4.1 (5). Le champ **File name** contient la valeur *nom_fichier*. Fichier-Répondeur → Nom = *nom_fichier*.

- Affiche-Attributs

Cette méthode affiche les attributs du fichier côté répondeur.

Précondition :

L'objet Fichier-Répondeur est affiché.

Postcondition :

On affiche les attributs du fichier comme représenté à la figure 4.10.b.

- Clignote-Attributs

Cette méthode montre qu'une action est entreprise sur les attributs.

Précondition :

Les attributs de l'objet Fichier-Répondeur sont affichés.

Postcondition :

Les attributs de l'objet Fichier-Répondeur sont affichés.

Règles de traitement :

Une animation fait clignoter une partie des attributs pendant quelques secondes, alternant l'affichage comme sur les figures 4.10.b et 4.11.

La méthode Clignote-Attributs attire l'attention de l'apprenant sur la zone de l'écran représentant les attributs du fichier appartenant à l'agent répondeur dans une association FTAM. Par l'utilisation de cette méthode, nous voulons lui faire sentir que les attributs du fichier sont soumis à une opération (lecture ou écriture).

- Affiche-Contenu

Cette méthode affiche la zone d'affichage du contenu et la structure du fichier virtuel dans cette zone.

Précondition :

L'objet Fichier-Répondeur est affiché.

Postcondition :

On affiche la zone d'affichage du contenu puis on y insère la structure des données (figure 4.14). Le noeud racine est sélectionné.

La méthode Affiche-Contenu est appelée quand les deux partenaires de l'association entre en régime d'ouverture de fichiers. Pour l'apprenant,

l'apparition de la structure du fichier, lui indique qu'il va pouvoir réaliser des opérations sur cette structure. Il connaît le FADU sélectionné grâce à la couleur différente du noeud racine par rapport aux autres noeuds de la structure.

- **Efface-Contenu**

Cette méthode efface l'affichage de la structure du fichier et la zone d'affichage du contenu. Le résultat final obtenu après l'application de cette méthode est identique à ce que représente la figure 4.10.b.

Précondition :

La structure des données de l'objet Fichier-Répondeur est affichée.

Postcondition :

La structure des données de l'objet Fichier-Répondeur et la zone d'affichage du contenu sont effacées, l'objet Fichier-répondeur et ses attributs sont affichés.

- **Efface-Structure**

Cette méthode efface l'affichage de la structure du fichier. Nous l'utiliserons lors d'une demande de la primitive **F-ERASE request** quand le noeud sélectionné sera le noeud racine.

Précondition :

La structure des données de l'objet Fichier-Répondeur est affichée.

Postcondition :

La structure des données de l'objet Fichier-Répondeur est effacée, seule la zone d'affichage du contenu est préservée.

Les méthodes relatives au transfert de fichier sont développées dans un paragraphe spécial (cfr. point 6).

- **Efface**

Cette méthode efface l'objet Fichier-Répondeur.

Précondition :

L'objet Fichier-répondeur et ses attributs sont affichés.

Postcondition :

L'objet Fichier-répondeur et ses attributs sont effacés.

Toutes ces méthodes d'affichage et d'effacement sont utiles pour mettre en évidence les objets que l'apprenant peut manipuler à un moment donné de la simulation. Nous insistons sur ce point car nous ne voulons pas que des objets qui n'ont plus de raison d'être soient toujours à l'écran. L'avantage de cette démarche est la représentation immédiate des changements relatifs aux commandes de l'utilisateur.

5.7. Objet Fournisseur

L'objet Fournisseur a pour but de véhiculer la demande de service de l'agent FTAM vers son agent pair. Pour que ces demandes puissent être transportées, une association doit être établie entre les deux agents FTAM coopérants. Lorsque l'application se termine, l'association est rompue.

5.7.1. Structure de l'objet

Lors du lancement du programme de simulation, aucune association n'existe entre ce que nous avons appelé agent initiateur et agent répondeur. L'objet Fournisseur, qui est représenté à la figure 4.1 (6), n'est pas encore présent à l'écran. Nous avons donc une situation similaire à celle de la figure 4.15.

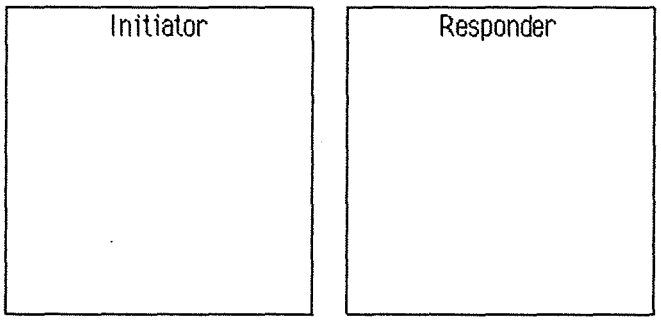


Figure 4.15 : les agents initiateur et répondeur lorsqu'il n'existe pas d'association. L'objet Fournisseur n'apparaît pas.

Dès qu'une association est établie l'objet Fournisseur apparaît (figure 4.16). Ce dernier est également présent lors de la négociation de l'association. Il symbolise alors le lien temporaire entre les deux agents.

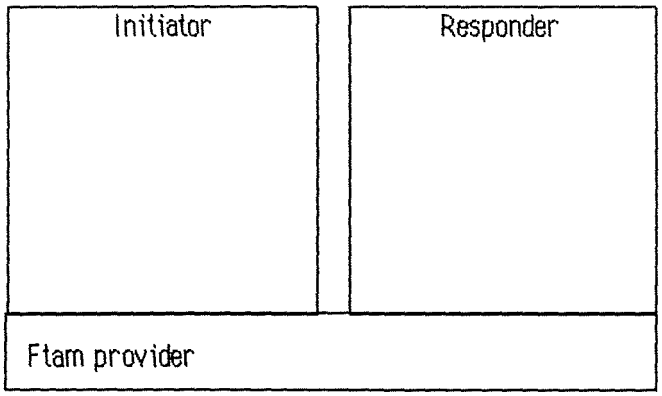


Figure 4.16 : les agents initiateur et répondeur sont associés. L'objet Fournisseur les relie.

Au cours de l'association qui les unit, les deux partenaires dialoguent et réalisent des opérations sur des fichiers. En toute généralité, ils s'échangent des "messages". Le passage de ces messages s'effectue par l'intermédiaire de PDU. La figure 4.17 symbolise le passage d'un PDU de l'agent initiateur vers l'agent récepteur. La figure 4.18, quant à elle, illustre le transfert d'un PDU en provenance du répondeur et à destination de l'initiateur.

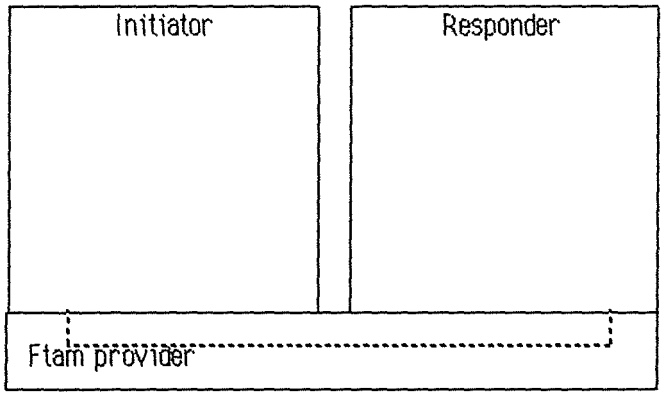


Figure 4.17 : l'objet Fournisseur transmet un PDU de l'initiateur vers le répondeur.

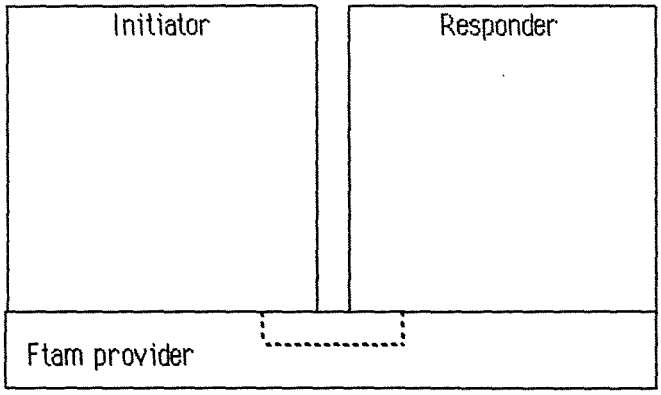


Figure 4.18 : l'objet Fournisseur transmet un PDU du répondeur vers l'initiateur.

La symbolique de l'association entre deux entités paires FTAM est entièrement contenue dans la présence à l'écran de l'objet Fournisseur. L'apprenant se rend compte de la liaison entre l'initiateur et le répondeur grâce à l'apparition de l'objet Fournisseur contre les objets Agent-Initiateur et Agent-Répondeur.

L'échange de PDU est simulé par le cheminement des lignes pointillées dans l'objet Fournisseur. L'apprenant se rend compte, grâce à ce mouvement, que des informations transitent entre les deux agents FTAM.

5.7.2. Méthodes associées

- Montre

Cette méthode affiche l'objet Fournisseur à l'écran. Elle symbolise l'établissement d'une association.

Précondition :

L'objet Fournisseur n'apparaît pas à l'écran (figure 4.15).

Postcondition :

L'objet Fournisseur est visible (figure 4.16).

- Cache

Cette méthode efface l'objet Fournisseur de l'écran. Elle symbolise la fin d'une association.

Précondition :

L'objet Fournisseur est visible à l'écran (figure 4.16).

Postcondition :

L'objet Fournisseur n'apparaît plus à l'écran (figure 4.15).

- Affiche-Envoi(sens)

Cette méthode s'occupe de l'animation du déplacement du "PDU" dans la direction *sens* spécifiée. Elle affiche aussi un message d'accompagnement.

sens = {IR,RI}.

Précondition :

L'objet Fournisseur se trouve dans l'état de la figure 4.16. Aucune animation n'est en cours. Rien n'apparaît dans l'objet.

Postcondition :

L'objet Fournisseur se trouve dans l'état de la figure 4.17 si la direction spécifiée vaut "IR" ou dans l'état de la figure 4.18 si la direction spécifiée vaut "RI". Les pointillés symbolisant le transfert sont affichés. Le message "L'unité de protocole de données (PDU) est envoyée à l'entité paire." est affiché.

Règles de traitement :

Case direction of
  IR : Animation_initiateur_répondeur.
  RI : Animation_répondeur_initiateur.
endcase
and Messages→Affiche("L'unité de protocole de données (PDU) est envoyée à l'entité paire.").

- Animation_initiateur_répondeur

Cette méthode symbolise l'envoi d'un "PDU" à partir de l'initiateur vers le répondeur.

Précondition :

L'objet Fournisseur se trouve dans l'état de la figure 4.16. Aucune animation n'est en cours. Rien n'apparaît dans l'objet.

Postcondition :

L'objet Fournisseur se trouve dans l'état de la figure 4.17. Les pointillés symbolisant le transfert sont affichés.

Règles de traitement :

Les pointillés représentés sont apparus au fur et à mesure en provenance du côté initiateur vers le côté répondeur.

- **Animation_répondeur_initiateur**

Cette méthode symbolise l'envoi d'un "PDU" à partir du répondeur vers l'initiateur.

Précondition :

L'objet Fournisseur se trouve dans l'état de la figure 4.16. Aucune animation n'est en cours. Rien n'apparaît dans l'objet.

Postcondition :

L'objet Fournisseur se trouve dans l'état de la figure 4.18. Les pointillés symbolisant le transfert sont affichés.

Règles de traitement :

Les pointillés représentés sont apparus au fur et à mesure au départ du côté répondeur vers le côté initiateur.

- **Efface-Envoi**

Cette méthode efface les pointillés.

Précondition :

Les pointillés sont affichés (figures 4.17 ou 4.18).

Postcondition :

Les pointillés ont disparu (figure 4.16).

5.8. Objet Etat-Initiateur

L'objet Etat-Initiateur reprend les états de la machine de protocole de fichier. Il permet de visualiser l'état actuel de la machine, l'état précédent et les états suivants.

5.8.1. Structure de l'objet

L'objet est caractérisé par son état précédent, son état courant et ses états suivants.

L'état précédent fait partie des états de la machine de protocole de fichier :

Etat-Initiateur→état_précédent = ETAT.

L'état courant fait partie des états de la machine de protocole de fichier :

Etat-Initiateur→état_courant = ETAT.

Le(s) état(s) suivant(s) fait (font) partie d'une séquence des états de la machine de protocole de fichier :

Etat-Initiateur→états_suivants = SEQ[ETAT].

Les transitions possibles sont rangées dans une table telle que chaque ligne de la table renseigne une transition valide. La transition est valide si pour un PDU reçu dans son état initial, la machine peut changer d'état (vers un état final) :

Etat-Initiateur→transitions = SET[CP[pdu:FP,état_ini:ETAT,état_fin:ETAT]]

L'objet est représenté sous forme d'icône (figure 4.19 - Etat-Initiateur→ icône = Y) ou sous forme développée dans laquelle les états de l'initiateur sont présentés (figure 4.20 - Etat-Initiateur→ icône = N).

Etat-Initiateur→icône = {Y,N}

D'un point de vue pédagogique, il nous semble opportun de laisser à l'utilisateur le choix de visualiser ou non la machine à états. En effet, il peut se contenter dans un premier temps d'employer les primitives de services à seule fin d'acquérir la connaissance de l'enchaînement des primitives. Une fois ce pas franchi, il peut se concentrer davantage sur les transitions d'états qui se produisent suite à des primitives dont il connaît désormais le sens et la finalité.

5.8.2. Méthodes associées

- Etat-Valide?(pdu) = BOOL

Cette méthode vérifie que le service demandé est valide dans l'état où la machine se trouve. Elle vérifie qu'il existe une ligne de la table des transitions qui comporte le PDU donné et l'état courant de la machine de protocole de fichier du côté initiateur.

pdu =FP.
état_fin : ETAT.

Précondition : /

Postcondition :

(∃ état_fin | (pdu,Etat-Initiateur → état_courant,état_fin)
∈ Etat-Initiateur → transitions)

- Modifie-Etat(pdu)

Cette méthode réalise le changement d'état de la machine de protocole de fichier. La transition se fait au départ de l'état courant vers l'état suivant en relation avec la demande de service contenue dans *pdu*

pdu : FP.

Précondition :

Etat-Valide?(pdu).

Postcondition:

(pdu,Etat-Initiateur → état_courant$_{PRE}$,état_fin) ∈ Etat-Initiateur → transitions
and Etat-Initiateur → état_courant=état_fin
and Etat-Initiateur → état_précédent=Etat-Initiateur → état_courant$_{PRE}$
and (pdu,Etat-Initiateur → état_courant,s) ∈ Etat-Initiateur → transitions
⇔ s ∈ Etat-Initiateur → états_suivants
and
if Etat-Initiateur → icône=N
then
Affiche_Etats
and Messages → Affiche("La machine de protocole de fichier côté initiateur
change d'état.").
endif

- Change-Mode-Affichage?=BOOL

Cette méthode renvoie la valeur TRUE si l'utilisateur veut changer le mode d'affichage (états/icône) de l'objet.

Précondition : /

Postcondition :

Change-Mode-Affichage? renvoie la valeur TRUE si l'utilisateur a manifesté son intention de changer le mode d'affichage de l'objet. Change-Mode-Affichage? prend la valeur FALSE sinon.

- Changer-Mode-Affichage

Cette méthode change le mode d'affichage des états. Elle fait basculer un affichage en détail des états de la machine de protocole de fichier, vers un affichage iconique.

Précondition : /

Postcondition

if Etat-Initiateur→icône=Y

then Etat-Initiateur→icône=N; Affiche_états

else Iconifie.

- Iconifie

Cette méthode affiche l'objet sous forme d'icône.

Précondition :

Etat-Initiateur→icône=N

Postcondition :

Etat-Initiateur→icône=Y

et l'objet Etat-Initiateur se présente à l'écran sous la forme reprise à figure 4.19.

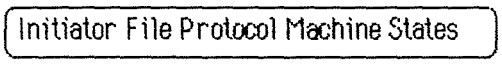


Figure 4.19 : l'objet Etat-Initiateur sous forme d'icône.

- Affiche_Etats.

Cette méthode affiche l'objet Etat-Initiateur sous le mode états détaillés.

Précondition : /

Postcondition :

Etat-Initiateur→icône=N

et l'objet Etat-Initiateur se présente à l'écran sous la forme reprise à figure 4.19. Le champ **previous state** contient la valeur de Etat-Initiateur→état_précédent. Le champ **current state** contient la valeur de Etat-Initiateur→état_courant. Il y a autant de champs **next state** qu'il y a d'éléments dans la séquence des Etat-Initiateur→états_suivants. Chaque champ contient un

des états de la séquence. Deux champs des états suivants ne contiennent pas le même état.

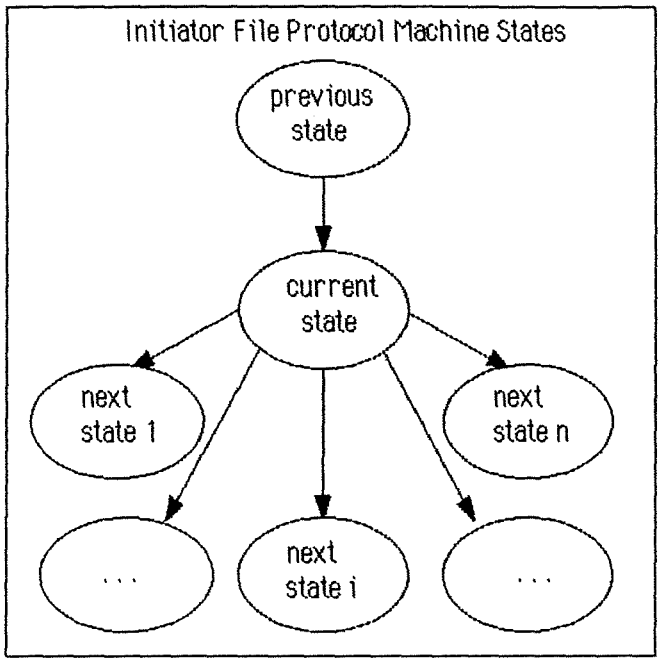


Figure 4.20 : l'objet Etat-Initiateur sous forme développée.

Nous avons choisi de représenter un nombre restreint d'états. Ce choix se justifie par le fait que d'autres éléments sont présents à l'écran et qu'en aucun cas il ne faut surcharger l'apprenant. Nous proposons une représentation sous forme d'icône qui permet simplement de masquer les changements d'états à l'utilisateur. Il peut ainsi choisir le moment où il désire prendre en compte cet aspect des choses.

Le nombre d'états possibles est élevé. Nous limiterons la représentation aux états que nous jugeons indispensable pour permettre à l'apprenant de situer le contexte de la simulation. Il s'agit de l'état courant, de l'état précédent et du ou des état(s) accessible(s) à partir de l'état courant (état(s) suivant(s)). Nous pensons qu'à elle seule, la connaissance de ces états facilite l'apprentissage de la machine de protocole de fichier. Disposer de tous les états à l'écran, alors que quelques uns seulement ont un sens pour le FPM à un moment donné de la simulation, ne nous semble pas être un service rendu à l'apprenant. De même nous avons consciemment éliminé l'état suivant **Idle** résultant du fait que la primitive **F-USER-ABORT**request peut être activée à n'importe quel moment. Cela ne faisait qu'ajouter un cas très particulier aux états suivants que la simulation traînerait avec elle jusqu'à la rupture de l'association.

5.9. Objet Etat-Répondeur

L'objet Etat-Répondeur reprend les états de la machine de protocole de fichier. Il permet de visualiser l'état actuel de la machine, l'état précédent et les états suivants.

5.9.1. Structure de l'objet

L'objet est caractérisé par son état précédent, son état courant et ses états suivants. La structure de cet objet s'établit de manière similaire à l'objet Etat-Initiateur (pour plus de détails, voir cet objet).

Etat-Répondeur → état_précédent = ETAT
Etat-Répondeur → état_courant = ETAT
Etat-Répondeur → états_suivants = SEQ[ETAT]
Etat-Répondeur → transitions = SET[CP[pdu:FP,état_ini:ETAT,état_fin:ETAT]]

L'objet est représenté sous forme d'icône (cfr. figure 4.21) ou sous forme développée où les états du répondeur sont présentés.

Etat-Répondeur → icône = {Y,N}

Les méthodes associées à cet objet que nous développons ci-dessous sont similaires à celles décrites pour l'objet Etat-Initiateur. Les considérations pédagogiques et les raisons découlant de la simulation étant les mêmes, nous ne nous attarderons pas sur leur explication mais nous nous contenterons de citer les méthodes utilisées.

5.9.2. Méthodes associées

- Change-Mode-Affichage?=BOOL

Cette méthode renvoie la valeur TRUE si l'utilisateur veut changer le mode d'affichage (états/icône) de l'objet.

Précondition : /

Postcondition :
Change-Mode-Affichage? prend la valeur TRUE si l'utilisateur a manifesté son intention de changer le mode d'affichage de l'objet ; sinon Change-Mode-Affichage? prend la valeur FALSE.

- Changer-Mode-Affichage

Cette méthode change le mode d'affichage.

Précondition : /

Postcondition :

if Etat-Répondeur→icône=Y

then Etat-Répondeur→icône=N; Affiche_états

else Iconifie.

- Modifie-Etat(pdu)

Cette méthode change l'état courant de la machine de protocole de fichier et affiche les modifications s'il y a lieu.

pdu = FP.
état_fin : ETAT.

Précondition : /

Postcondition:

(pdu ,Etat-Répondeur→état_courant$_{PRE}$, état_fin) ∈ Etat-Répondeur→transitions

and Etat-Répondeur→état_courant=état_fin

and Etat-Répondeur→état_précédent=Etat-Répondeur →état_courant$_{PRE}$

and (pdu,Etat-Répondeur→état_courant,s) ∈ Etat-Répondeur→transitions

⇔ s ∈ Etat-Répondeur→états_suivants

and

if Etat-Répondeur→icône=N

then

Affiche_Etats

Messages→Affiche("La machine de protocole de fichier côté répondeur change d'état.").

endif

- Iconifie

Cette méthode permet de passer dans le mode de représentation sous forme d'icône.

Précondition :

Etat-Répondeur→icône=N

Postcondition :

Etat-Répondeur → icône=Y

et l'objet Etat-Répondeur se présente à l'écran sous la forme reprise à figure 4.21.

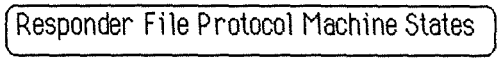


Figure 4.21 : la machine de protocole de fichier sous forme iconique.

- **Affiche_Etats**

Cette méthode affiche les états de l'objet Etat-Répondeur.

Précondition : /

Postcondition :

Etat-Répondeur → icône=N

et l'objet Etat-Répondeur se présente à l'écran sous une forme identique à la figure 4.20. Le champ **previous state** contient la valeur de Etat-Répondeur → état_précédent. Le champ **current state** contient la valeur de Etat-Répondeur → état_courant. Il y a autant de champs **next state** qu'il y a d'éléments dans la séquence des Etat-Répondeur → états_suivants. Chaque champ contient un des états de la séquence. Deux champs des états suivants ne contiennent pas le même état.

6. Spécification du transfert de fichier

Le transfert de fichier mérite un traitement en profondeur à lui tout seul et devra faire l'objet d'un autre scénario (cfr. chapitre 6). Nous limiterons, dans le cadre de ce scénario, les cas d'exploitation des possibilités offertes par FTAM en matière de transfert de fichier. Cette partie sera donc essentiellement démonstrative et servira simplement à permettre la liaison avec un autre scénario.

6.1. Objet Fichier-Répondeur

6.1.1. La structure du fichier Répondeur

Nous présentons un seul fichier. Il s'agit d'un fichier séquentiel plat. Sa structure est représentée à la figure 4.22.

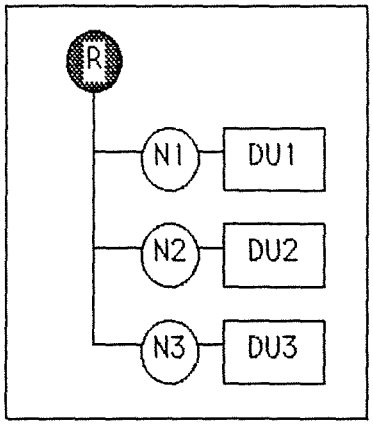


Figure 4.22 : un fichier séquentiel plat.

La structure est représentée par :

- le nom du noeud racine :
  Fichier-Répondeur →racine=STRING;
- la séquence ordonnée des noeuds fils du noeud racine identifiés par leur nom :
  Fichier-Répondeur →noeuds=SEQ[STRING];
- la séquence ordonnée des unités de données identifiées par leur nom :
  Fichier-Répondeur →DU=SEQ[STRING];
  La $i^{ème}$ unité de données de la séquence correspond au $i^{ème}$ noeud de la séquence des noeuds du fichier répondeur.

6.1.2. Méthodes associées

- Racine?(nom_noeud)

Cette méthode vérifie si la valeur du paramètre *nom_noeud* correspond au nom du noeud donné à la racine.

nom_noeud = STRING.

Précondition : /

Postcondition :
(nom_noeud=Fichier-Répondeur →racine)

- Dernier-Noeud?(nom_noeud)

Cette méthode vérifie si la valeur contenue dans le paramètre *nom_noeud* correspond au dernier noeud de la séquence Fichier-Répondeur → noeuds.

nom_noeud = STRING

Précondition : /

Postcondition :
(nom_noeud=Last(Fichier-Répondeur →noeuds)).

Remarque :
Last prend le dernier élément d'une liste [DUBO90].

- Noeud-Suivant(noeud_précédent) = noeud_suivant

Cette méthode renvoie le nom du noeud *noeud_suivant* situé dans la séquence derrière le noeud passé en paramètre *noeud_précédent*.

noeud_précédent, noeud_suivant = STRING.

Précondition :
not Dernier-Noeud?(noeud_précédent)
and $\exists i \mid$ Fichier-Répondeur→ noeuds$_i$ = noeud_précédent.

Postcondition :
noeud_suivant = Fichier-Répondeur→ noeuds$_{i+1}$.

Remarque :
noeud$_i$ désigne le $i^{ème}$ élément de la séquence des noeuds [DUBO90].

- Lire-DU(nom_noeud)=nom_DU

Cette méthode renvoie le nom de l'unité de données *nom_DU* associée au noeud identifié par le paramètre *nom_noeud*

nom_noeud, nom_DU = STRING.

Précondition :
$\exists i \mid$ Fichier-Répondeur→ noeuds$_i$ = nom_noeud.

Postcondition :
nom_DU = Fichier-Répondeur→ DU$_i$.

- Choisir-FADU=nom

Cette méthode permet à l'utilisateur de choisir un noeud. Ce noeud constitue le noeud racine du FADU sur lequel portera une opération de lecture, d'effacement, ... Le paramètre *nom* contient le nom du noeud sélectionné par l'utilisateur.

nom = STRING.

Précondition :

Le contenu de l'objet Fichier-Répondeur est affiché.

Postcondition :

Le paramètre *nom* contient le nom du noeud choisi par l'utilisateur. La valeur du paramètre correspond obligatoirement à un noeud existant dans l'objet Fichier-Répondeur.

- Localise-Noeud(noeud)

Cette méthode permet de mettre en évidence un noeud sélectionné par l'utilisateur. Par exemple, si l'utilisateur choisi le noeud N2, on obtient la représentation de la figure 4.23.

noeud = STRING.

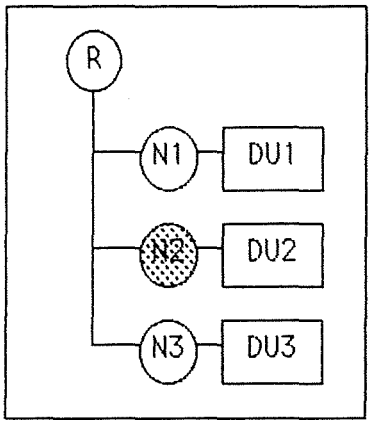


Figure 4.23 : le noeud N2 du fichier est sélectionné.

Précondition :

Le contenu de l'objet Fichier-Répondeur est affiché.

Postcondition :

Le noeud dont le nom correspond à la valeur passée dans le paramètre *noeud* est mis en évidence par rapport aux autres noeuds. Ceci suppose que si un noeud était déjà en évidence lors de l'appel de la méthode, il ne le sera plus après l'exécution de la méthode, sauf s'il correspondait au noeud à mettre en évidence.

- Transfert-Unité(noeud)

Cette méthode permet de mettre en évidence les objets de la structure qui sont concernés par le transfert actuel dans le **F-DATA request**.

noeud=STRING

Le paramètre *noeud* désigne le noeud qui est transféré. L'unité de données (DU) associée est également transférée.

Précondition :

Le contenu de l'objet Fichier-Répondeur est affiché.

Postcondition :

Le noeud désigné et son unité de données associée sont mis en évidence comme sur la figure 4.24.

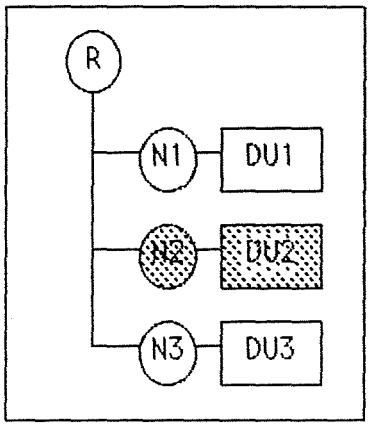


Figure 4.24 : le noeud N2 et son unité de données font l'objet d'un transfert.

- Fin-Transfert-Unité(noeud)

Cette méthode permet de terminer la mise en évidence des objets de la structure qui sont concernés par le transfert actuel dans le **F-DATA request**.

noeud=STRING.

Le paramètre *noeud* désigne le noeud qui a été transféré. L'unité de données (DU) associée a également été transférée.

Précondition :

Le noeud désigné et son unité de données associée sont mis en évidence comme sur la figure 4.24.

Postcondition :

Le noeud désigné et son unité de données associée ne sont plus mis en évidence.

- Insérer-Unité(nom_noeud,nom_DU)

Cette méthode crée un nouveau noeud et son unité de données associée. Le noeud est attaché au noeud racine. Le nom du noeud prend la valeur de *nom_noeud* passée en paramètre. Le nom de l'unité de données prend la valeur de *nom_DU* passée en paramètre. Si les paramètres contiennent des valeurs nulles ("") alors, on attribue les noms Ni et DUi ou i représente un chiffre supérieur d'une unité au nombre d'unités de données déjà présentes dans le fichier.

nom_noeud, nom_DU = STRING.

Précondition :

Le contenu de l'objet Fichier-Répondeur est affiché (cfr. figure 4.24, par exemple).

Postcondition :

Un noeud est créé avec pour nom la valeur du paramètre *nom_noeud*. Une unité de données est créée avec pour nom la valeur du paramètre *nom_DU*. L'unité de données est attachée au noeud. Le noeud est attaché au noeud racine. Pour la figure 4.22, l'exécution de la méthode avec *nom_noeud* = 'N4' et *nom_DU* = 'DU4' donne le résultat représenté à la figure 4.25.

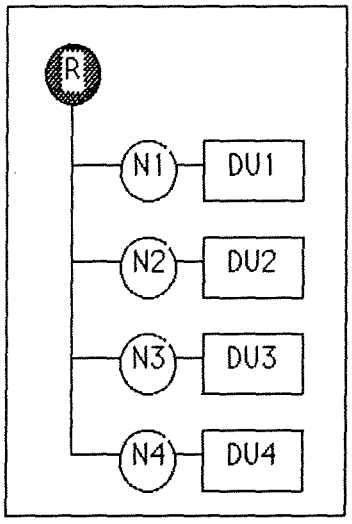


Figure 4.25 : le noeud N4 et l'unité de données DU4 ont été ajoutés au fichier côté répondeur.

- Effacer-Unité(nom_noeud)

Cette méthode permet d'effacer un noeud (cas du **F-ERASE request**).

nom_noeud = STRING.

Précondition :

Le contenu de l'objet Fichier-Répondeur est affiché.

Postcondition :

Le noeud sélectionné *nom_noeud* et l'unité de données attachée ont disparu.

6.2. Objet Fichier-Initiateur

Dans le fichier initiateur, on représentera les informations transférées (cas du **F-READ request**) au fur et à mesure de leur transfert.

6.2.1. Méthodes associées

- Créer-Racine(nom_noeud)

Cette méthode crée le noeud racine et lui attribue le nom *nom_noeud*.

nom_noeud = STRING.

Précondition :

Le contenu de l'objet Fichier-Initiateur est affiché. Aucune structure n'apparaît.

Postcondition :

Le noeud racine fait partie du contenu, le nom attribué correspond au *nom_noeud* passé en argument.

- Attacher-Unité(nom_noeud,nom_DU)

nom_noeud, nom_DU = STRING.

Cette méthode crée un nouveau noeud et son unité de données associée. Le noeud est attaché au noeud racine s'il existe. Le nom du noeud prend la valeur *nom_noeud* passée en paramètre. Le nom de l'unité de données prend la valeur *nom_DU* passée en paramètre.

Précondition :

Le contenu de l'objet Fichier-Initiateur est affiché (cfr figure 4.22, par exemple).

Postcondition :

Un noeud est créé avec pour nom la valeur du paramètre *nom_noeud*. Une unité de données est créée avec pour nom la valeur du paramètre *nom_DU*. L'unité de données est attachée à ce noeud. Si le noeud racine est présent, le noeud est attaché au noeud racine. Pour la figure 4.21, l'exécution de la méthode avec

nom_noeud = 'N4' et *nom_DU* = 'DU4' donne le résultat représenté à la figure 4.25.

Toutes les méthodes qui viennent d'être décrites concernant les opérations sur le fichier sont importantes. Toujours guidés par le souci d'amener l'utilisateur du didacticiel à se rendre compte des mécanismes de transfert et de gestion de fichiers, nous avons développé ces méthodes afin de rendre la simulation aussi visuelle que possible. Il ne faut pas oublier que nous travaillons avec des structures de fichiers virtuels.

L'analyse des différents points que nous désirons illustrer grâce au didacticiel, nous a conduit à mettre en évidence toute opération en relation avec les données ou la structure du fichier. Lorsqu'une lecture ou une écriture est en cours, le noeud et l'unité de données sont mis en évidence par un changement de couleur, voire un clignotement. Les opérations, qui d'un point de vue FTAM se déroulent en temps réel, sont simulées à une vitesse telle que l'utilisateur puisse se rendre compte de ce qui se produit.

Remarquons que cette vitesse est difficile à fixer étant donné qu'elle peut varier très fort d'un utilisateur à l'autre. Nous pourrions envisager de permettre l'interruption de l'animation pour les utilisateurs pressés de voir la suite.

7. Description du scénario

La simulation peut être décrite par l'intermédiaire d'un réseau tel que celui de la figure 4.26. L'entrée dans le didacticiel correspond à l'unité d'interaction UI0. Le parcours dans le réseau est entièrement laissé à l'utilisateur. Le programme de simulation n'impose pas explicitement un chemin plutôt qu'un autre. Toutefois, le développement des actions associées à une unité d'interaction n'est effectué qu'après une vérification sémantique de la requête. Si la requête s'avère être invalide dans le contexte où elle est évaluée, un message d'erreur est affiché et l'utilisateur est invité à sélectionner une autre primitive. Lorsque la séquence des événements à mettre en oeuvre s'achève, l'unité d'interaction UI0 reprend la main et donne à l'utilisateur la possibilité de démarrer l'exécution d'une autre primitive.

Quand nous nous trouvons dans l'unité d'interaction UI0, nous avons le choix de déclencher une requête FTAM ou de quitter le didacticiel. Remarquons qu'il ne s'agit pas de quitter l'association FTAM éventuellement créée, mais de sortir directement de la simulation quelques soient les actions entreprises jusque là.

Nous avons délibérément choisi de laisser l'apprenant libre dans la simulation. Il peut prendre des décisions et c'est en fonction de celles-ci que le didacticiel va réagir. Si l'apprenant commet des erreurs (par exemple, vouloir ouvrir un fichier sans l'avoir sélectionné), le didacticiel les lui signale et l'apprenant peut continuer à avancer dans la simulation. En effet, les erreurs d'apprentissage ou de manipulation étant détectées immédiatement, la cohérence et la fidélité vis-à-vis de FTAM sont garanties.

La possibilité de quitter la simulation à n'importe quel moment nous paraissait être primordiale (même en dehors de UI0). Nous estimons que forcer l'apprenant à quitter d'abord l'association FTAM avant de pouvoir sortir du programme de simulation n'apporte rien à l'apprenant quand son seul désir est de sortir de l'application.

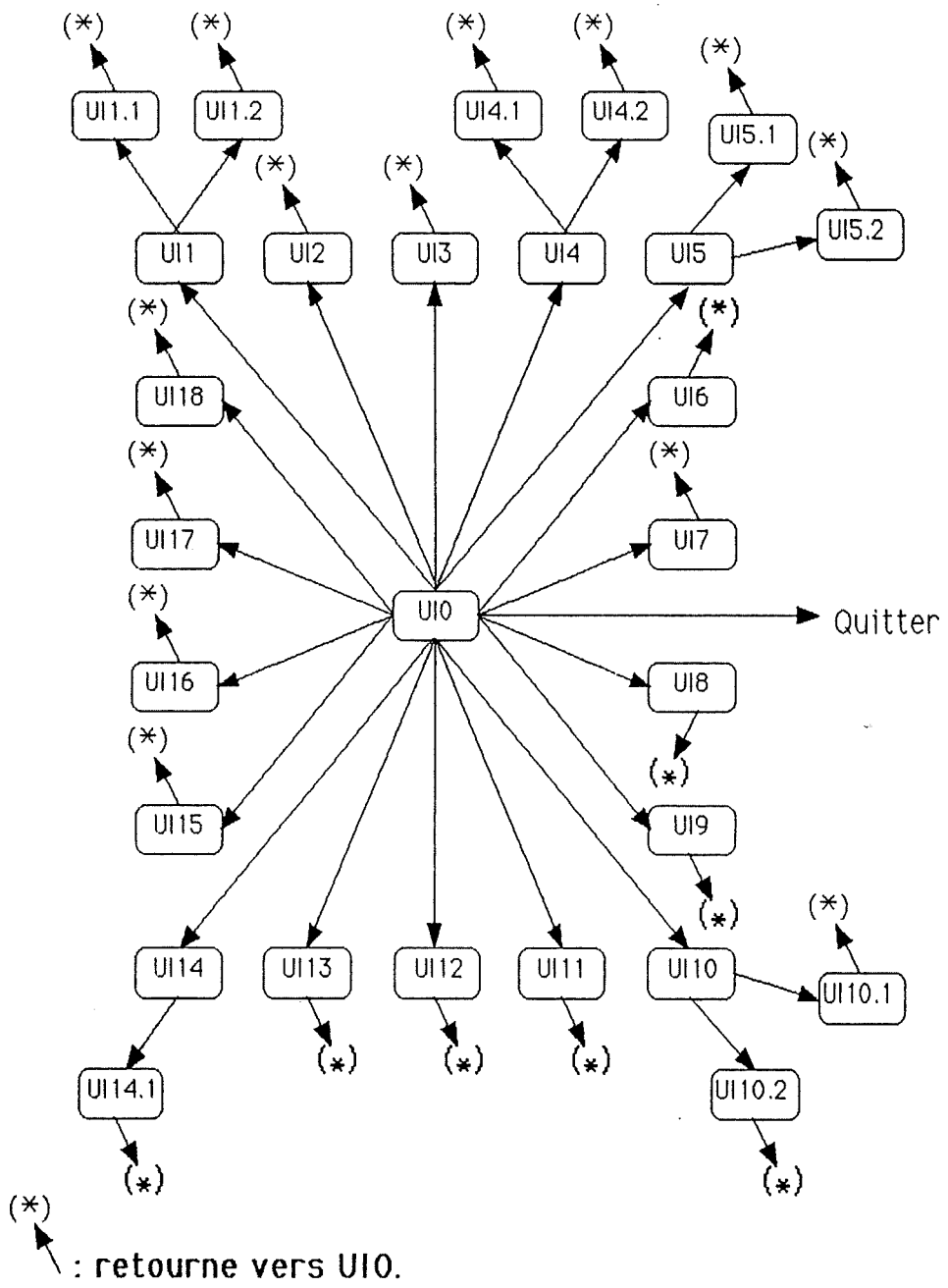


Figure 4.26 : les unités d'interaction régissant la simulation.

Dans la suite de la description du scénario, les paramètres présents dans les primitives de requêtes seront identifiés par "XXXXXX". Les paramètres se trouvant dans la réponse seront repérés par "YYYYYY". Nous avons choisi ce moyen pour exprimer qu'il existe généralement plusieurs paramètres dans la primitive et que nous ne les prenons pas en compte individuellement dans ce scénario.

7.1. Unité d'interaction UI0 : sélection d'une primitive et traitement associé

L'unité d'interaction UI0 est le coeur de la simulation, c'est en quelque sorte un coordinateur. C'est dans cette unité que l'utilisateur réalise le choix d'une primitive dont il désire l'exécution. Lorsque le choix est fait, l'unité d'interaction UIi correspondant au choix réalisé est déclenchée. Elle est responsable des séquences d'actions sur les différents objets impliqués dans la simulation de la primitive. Lorsque l'unité UIi a terminé son travail, le contrôle est rendu à l'unité UI0 (voir figure 4.26).

Nous pouvons décrire le comportement de l'unité d'interaction coordinatrice comme suit :

```
fin = FALSE
do while not fin

    if Etat-Initiateur --> Change-Mode-Affichage?
    then Etat-Initiateur --> Changer-Mode-Affichage
    endif

    if Etat-Répondeur --> Change-Mode-Affichage?
    then Etat-Répondeur --> Changer-Mode-Affichage
    endif

    Messages --> Affiche-Aide( "Sélectionner une primitive." )
    choix-utilisateur = Agent-Initiateur --> Accepte-Choix

    case choix-utilisateur of
       F_INI_RQ : goto UI1
       F_TERM_RQ : goto UI2
       F_UABT_RQ : goto UI3
       F_SEL_RQ : goto UI4
       F_CRE_RQ : goto UI5
       F_DESEL_RQ : goto UI6
       F_DEL_RQ : goto UI7
       F_RDATT_RQ : goto UI8
       F_CHATT_RQ : goto UI9
       F_OPN_RQ : goto UI10
       F_CLO_RQ : goto UI11
       F_LOC_RQ : goto UI12
```

F_ERA_RQ : goto UI13
F_RD_RQ : goto UI14
F_WR_RQ : goto UI15
F_DATA_RQ : goto UI16
F_EDAT_RQ : goto UI17
F_ETRSF_RQ : goto UI18
QUIT : fin = TRUE
endcase

enddo

Si nous analysons cette description de l'unité d'interaction UI0, nous constatons qu'elle permet à l'utilisateur de changer le mode d'affichage des états initiateur et répondeur et qu'elle permet à l'utilisateur de sélectionner une primitive FTAM. L'utilisateur débutant peut choisir une foule de primitives et le risque d'erreur est grand. Bien qu'une erreur lui sera signalée rapidement, il est certain qu'au bout de trois ou quatre essais infructueux, l'apprenant décide de quitter tout bonnement l'application. C'est d'ailleurs la seule requête dont il est certain du résultat. Nous avons donc étudié ce problème et nous allons vous proposer une solution. Nous envisageons deux modes de travail sur le didacticiel, un mode débutant et un mode expert. En mode débutant, les requêtes valides en fonction de l'état courant apparaissent dans une couleur différente des requêtes invalides. L'apprenant est donc guidé dans son choix. Néanmoins, les requêtes "invalides" sont toujours susceptibles d'être sélectionnées par l'apprenant qui recevra dès lors un message d'erreur. Le mode expert ne fait pas de nuance sur les couleurs des requêtes, il est destiné à un utilisateur averti qui sait déjà où il va et ce qu'il désire faire du didacticiel.

7.2. Unité d'interaction UI1 : F-INITIALIZE

L'établissement d'un régime FTAM est la première phase de toute activité sur un fichier. La primitive **F-INITIALIZE request** est sélectionnée par l'utilisateur du didacticiel. L'utilisateur joue donc le rôle de l'agent initiateur d'un transfert de fichier. Deux associations FTAM ne peuvent être imbriquées mais elles peuvent se succéder.

La séquence d'actions suivante se déroule afin de satisfaire la demande de l'utilisateur :

Agent-Initiateur → Affiche("↓ F-INITIALIZE req (XXXXXX)")
Messages → Affiche("L'agent initiateur FTAM demande l'établissement d'une association.")

```
if not Etat-Initiateur --> Etat-Valide?( F_INI_RQ)
then
   Messages --> Affiche-Erreur( "Vous avez sélectionné une primitive invalide
                dans l'état actuel de la machine de protocole de fichier." )
   Agent-Initiateur --> Efface
   goto UI0
endif

Etat-Initiateur --> Modifie-Etat( F_INI_RQ )
Fournisseur-FTAM --> Montre
Fournisseur-FTAM --> Affiche-Envoi( IR )
Etat-Répondeur --> Modifie-Etat( F_INI_RQ )
Agent-Répondeur --> Affiche( "F-INITIALIZE ind ( XXXXXX ) ↑" )
Messages --> Affiche( "Le fournisseur FTAM signale à l'agent répondeur qu'une
                demande d'association lui est parvenue." )
Agent-Initiateur --> Efface
Fournisseur-FTAM --> Efface-Envoi
Messages --> Affiche-Aide( "Acceptez-vous l'établissement d'une association?" )
choix-utilisateur = Agent-Répondeur --> Accepte-Choix
Agent-Répondeur --> Efface

case choix-utilisateur of
   "Y" : goto UI1.1
   "N" : goto UI1.2
endcase
```

Suite au choix effectué par l'agent répondeur, une autre unité d'interaction va prendre le relais. Si l'utilisateur accepte l'association, les deux entités paires entreront en régime d'association FTAM (UI1.1). Par contre, un refus place les deux entités au même point qu'avant la demande d'association (UI1.2).

7.3. Unité d'interaction UI1.1 : F-INITIALIZE resp(+ve)

Cette unité d'interaction est déclenchée quand l'utilisateur de la simulation a opté pour l'acceptation de la primitive **F-INITIALIZE request**. Dans ce cas, l'association sera effective entre les deux entités paires lorsque l'initiateur de la primitive recevra la confirmation du choix positif provenant de l'entité répondante. Cela se traduira par l'entrée dans le régime d'association FTAM.

La réalisation des changements relatifs au déroulement de cette unité d'interaction se caractérise par la séquence d'événements suivante :

Agent-Répondeur → Affiche("F-INITIALIZE resp (YYYYYY) ↓")
Messages → Affiche("L'agent répondeur FTAM accepte l'association.")
Etat-Répondeur → Modifie-Etat(F_INI_RS+)
Fournisseur-FTAM → Affiche-Envoi(RI)
Etat-Initiateur → Modifie-Etat(F_INI_RS+)
Agent-Initiateur → Affiche("↑ F-INITIALIZE conf (YYYYYY)"
Messages → Affiche("Le fournisseur FTAM signale à l'agent initiateur qu'il a reçu une confirmation de la part de l'entité paire. La confirmation donne lieu à l'établissement d'une association.")
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Régime → Entrer-Régime
Agent-Initiateur → Efface
goto UI0

Nous pouvons noter que les paramètres de la primitive de réponse ont pu changer suite à des modifications effectuées par le répondeur (*YYYYYY* au lieu de *XXXXXX*). En effet, l'association est négociée par les deux partenaires qui expriment leurs exigences pour une association acceptable par le biais de ces paramètres.

Nous faisons remarquer que d'un point de vue pédagogique cette approche n'est pas parfaite. Jusqu'à présent l'apprenant s'identifiait à l'agent initiateur. Voilà maintenant qu'il doit prendre des décisions en lieu et place de l'agent répondeur. Par la suite cette situation se rencontrera encore. Néanmoins, ces décisions permettent d'enrichir la simulation par un plus grand nombre de possibilités offertes. De plus, l'apprenant est davantage impliqué dans le déroulement de la séquence. Ceci constitue un facteur motivationnel important. L'apprenant ne se trouve pas devant un écran de cinéma mais dans un mini-monde où ses actions sont prises en compte.

C'est pourquoi nous pensons qu'une telle attitude est justifiable d'autant que nous proposons de réaliser la sélection dans l'objet concerné, ce qui pemet une identification assez naturelle de l'apprenant avec l'objet qu'il manipule.

7.4. Unité d'interaction UI1.2 : F-INITIALIZE resp(-ve)

Cette unité d'interaction est appelée suite à un refus de la primitive **F-INITIALIZE request** de la part de l'agent répondeur de l'application FTAM. Dans ce cas, les deux entités paires n'entrent pas en association. L'objet Fournisseur qui était apparu pour véhiculer le PDU de demande d'association va disparaître après avoir transmis la réponse.

La séquence d'événements suivante s'opère en vue de satisfaire aux exigences de l'utilisateur :

Agent-Répondeur → Affiche("F-INITIALIZE resp (YYYYYY) ↓")
Messages → Affiche("L'agent répondeur refuse d'établir une association.")
Etat-Répondeur → Modifie-Etat(F_INI_RS-)
Fournisseur-FTAM → Affiche-Envoi(RI)
Etat-Initiateur → Modifie-Etat(F_INI_RS-)
Agent-Initiateur → Affiche("↑ F-INITIALIZE conf (YYYYYY)")
Messages → Affiche("La confirmation de la demande d'association parvient à
son initiateur. Celui-ci est informé de l'échec de sa requête.")
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Fournisseur-FTAM → Cache
Agent-Initiateur → Efface
goto UI0

7.5. Unité d'interaction UI2 : F-TERMINATE

L'utilisateur sélectionnant la primitive **F-TERMINATE request** va quitter le régime d'association. Par cette action, il exprime son désir de rompre toute activité avec l'entité paire avec laquelle il communiquait. La primitive **F-TERMINATE request** représente la terminaison ordonnée d'une association FTAM. Elle est utilisée quand il n'y a plus d'action en cours entre les deux entités paires. L'exécution de cette primitive est accompagnée d'un changement de régime : les deux protagonistes quittent le régime **FTAM association regime**. Les deux machines de protocole de fichier en attente d'une nouvelle association se trouve dans l'état *disponible* (**Idle**). L'objet Fournisseur n'est plus à l'écran.

Cet objectif est atteint au travers de la séquence suivante d'actions :

Agent-Initiateur → Affiche("↓ F-TERMINATE req")
Messages → Affiche("L'agent FTAM initiateur veut rompre l'association.")

```
if not Etat-Initiateur → Etat-Valide?( F_TERM_RQ )
then
   Messages → Affiche-Erreur( "Vous avez sélectionné une primitive invalide
                dans l'état actuel de la machine de protocole de fichier." )
   Agent-Initiateur → Efface
   goto UI0
endif

Etat-Initiateur → Modifie-Etat( F_TERM_RQ )
Fournisseur-FTAM → Affiche-Envoi( IR )
Etat-Répondeur → Modifie-Etat( F_TERM_RQ )
Agent-Répondeur → Affiche( "F-TERMINATE ind ↑" )
Messages → Affiche( "Le fournisseur FTAM signale à l'agent répondeur que
                l'association avec l'entité FTAM paire va être rompue." )
Agent-Initiateur → Efface
Fournisseur-FTAM → Efface-Envoi
Agent-Répondeur → Efface
Agent-Répondeur → Affiche( "F-TERMINATE resp ( YYYYYY ) ↓" )
Messages → Affiche( "L'agent répondeur envoie une réponse à la demande de
                rupture d'association." )
Etat-Répondeur → Modifie-Etat( F_TERM_RS )
Fournisseur-FTAM → Affiche-Envoi( RI )
Etat-Initiateur → Modifie-Etat( F_TERM_RS )
Agent-Initiateur → Affiche( "↑ F-TERMINATE conf ( YYYYYY )" )
Messages → Affiche( "La confirmation provenant de l'agent répondeur est portée
                à la connaissance de l'agent initiateur." )
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Fournisseur-FTAM → Cache
Régime → Sortir-Régime
Agent-Initiateur → Efface
goto UI0
```

7.6. Unité d'interaction UI3 : F-USER-ABORT

La primitive **F-USER-ABORT request** correspond à une terminaison désordonnée d'une association FTAM. Elle a pour conséquence la cessation des activités en cours. Si un fichier était ouvert, il sera fermé. Si un fichier était

sélectionné, il ne le sera plus. Si des données étaient en cours de manipulation, elles seront perdues.

La séquence d'actions suivante est mise en route :

```
Agent-Initiateur --> Affiche( "↓ F-USER-ABORT req ( XXXXXX )" )
Messages --> Affiche( "L'agent initiateur rencontre des problèmes qu'il ne peut
                surmonter. Il va quitter l'association de manière abrupte." )

if not Etat-Initiateur --> Etat-Valide?( F_UABT_RQ)
then
   Messages --> Affiche-Erreur( "Vous avez sélectionné une primitive invalide
                dans l'état actuel de la machine de protocole de fichier. Il n'y a
                pas d'association en ce moment." )
   Agent-Initiateur --> Efface
   goto UI0
endif

Etat-Initiateur --> Modifie-Etat( F_UABT_RQ )
Fournisseur-FTAM --> Affiche-Envoi( IR )
Etat-Répondeur --> Modifie-Etat( F_UABT_RQ )
Agent-Répondeur --> Affiche( "F-USER-ABORT ind ( XXXXXX ) ↑" )
Messages --> Affiche( "Le fournisseur FTAM signale une fin anormale de
                l'association à la demande de l'initiateur. Attention, cela peut
                causer la perte des données." )
Agent-Initiateur --> Efface
Fournisseur-FTAM --> Efface-Envoi

If Régime --> Hauteur = 4
then Régime --> Sortir-Régime
endif

If Régime --> Hauteur = 3
then
   Régime --> Sortir-Régime
   Fichier-Initiateur --> Efface-Contenu
   Fichier-Répondeur --> Efface-Contenu
endif
```

If Régime → Hauteur = 2
then
  Régime → Sortir-Régime
  Fichier-Initiateur → Efface
  Fichier-Répondeur → Efface
endif

If Régime → Hauteur = 1
then Régime → Sortir-Régime
endif

Fournisseur-FTAM → Cache
Agent-Répondeur → Efface
goto UI0

La notion de terminaison abrupte est un point essentiel du modèle ISO. Elle apparaît à plusieurs niveaux du modèle et n'est pas toujours simple à comprendre. Dans le cas qui nous préoccupe, la terminaison abrupte de l'association résulte d'une demande de l'agent FTAM. Les conséquences sont la cessation de l'activité en cours. Au sens FTAM, la primitive **F-USER-ABORT request**rompt l'association entre les deux agents FTAM. Si un transfert de données est en cours, les données seront perdues. Le fichier qui est ouvert et sélectionné ne le sera plus. Le scénario doit rendre le plus clairement la terminaison abrupte FTAM. Nous avons donc dû quitter le régime courant pour retourner à un mode où aucun régime FTAM n'est présent. Si la structure du fichier était affichée, il faut l'effacer côté répondeur et initiateur. Si les attributs du fichier étaient disponibles, il faut également supprimer l'affichage à l'écran. Nous avons donc dû enlever de l'écran tous les objets qui étaient apparus au cours de l'association.

7.7. Unité d'interaction UI4 : F-SELECT

L'utilisateur qui se trouve dans un régime d'association peut entrer dans un régime de sélection de fichier. Une manière d'y accéder est de sélectionner un fichier préexistant.

Ci-dessous figurent les événements découlant d'une primitive **F-SELECT request** :

Messages → Affiche("Sélectionnez un fichier.")
nom-fichier = Agent-Initiateur → Lire-Nom
Agent-Initiateur → Affiche("↓ F-SELECT req (XXXXXX)")
Messages → Affiche("L'agent initiateur demande la sélection d'un fichier.")

```
if not Etat-Initiateur → Etat-Valide?( F_SEL_RQ)
then
   Messages → Affiche-Erreur( "Vous avez sélectionné une primitive invalide
                  dans l'état actuel de la machine de protocole de fichier." )
   Agent-Initiateur → Efface
   goto UI0
endif

Etat-Initiateur → Modifie-Etat( F_SEL_RQ )
Fournisseur-FTAM → Affiche-Envoi( IR )
Etat-Répondeur → Modifie-Etat( F_SEL_RQ )
Agent-Répondeur → Affiche( "F-SELECT ind ( XXXXXX ) ↑" )
Messages → Affiche( "Le fournisseur FTAM donne à l'entité répondante une
                  indication concernant la sélection d'un fichier." )
Agent-Initiateur → Efface
Fournisseur-FTAM → Efface-Envoi
Messages → Affiche-Aide( "Acceptez-vous ou refusez-vous la sélection du
                  fichier?")
choix-utilisateur = Agent-Répondeur → Accepte-Choix
Agent-Répondeur → Efface

case choix-utilisateur of
   "Y" : goto UI4.1
   "N" : goto UI4.2
endcase
```

7.8. Unité d'interaction UI4.1 : F-SELECT resp(+ve)

Cette unité d'interaction est déclenchée dès que l'utilisateur de la simulation a opté pour l'acceptation de la primitive **F-SELECT request**. L'entité répondante confirmant la sélection, les partenaires entreront dans le régime de sélection de fichier. Les attributs du fichier sont accessibles après la fin de cette unité d'interaction.

La réalisation des changements relatifs au déroulement de cette unité d'interaction se caractérise par la séquence d'événements suivante :

```
Fichier-Répondeur → Affiche-Fichier(nom-fichier)
Fichier-Répondeur → Affiche-Attributs
```

Agent-Répondeur → Affiche("F-SELECT resp (YYYYYY) ↓")
Messages → Affiche("L'agent FTAM répondeur accepte la sélection du fichier.")
Etat-Répondeur → Modifie-Etat(F_SEL_RS+)
Fournisseur-FTAM → Affiche-Envoi(RI)
Etat-Initiateur → Modifie-Etat(F_SEL_RS+)
Agent-Initiateur → Affiche("↑ F-SELECT conf (YYYYYY)")
Messages → Affiche("Le fournisseur FTAM donne la confirmation de la sélection du fichier à l'initiateur de la requête.")
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Régime → Entrer-Régime
Fichier-Initiateur → Affiche-Fichier(nom-fichier)
Agent-Initiateur → Efface
goto UI0

Nous pouvons noter que les paramètres de la primitive de réponse ont pu changer suite à des modifications effectuées par le répondeur. Les paramètres indiquent le résultat de la sélection et le nom exact du fichier sélectionné dans le cas où un nom générique était envoyé dans la requête.

L'utilisateur reçoit une confirmation de l'action qu'il voulait exécuter quand la boîte contenant les attributs et le nom du fichier s'affiche près de l'agent répondeur. La présence des attributs lui indique qu'il pourra entreprendre une lecture ou une modification de ceux-ci. Du côté agent initiateur, seule une boîte contenant le nom du fichier sélectionné apparaît. Cela veut dire que l'agent initiateur n'a pas connaissance de la valeur de ces attributs (au sens FTAM). A la fin du processus de sélection de fichier, l'apprenant est averti d'un changement de régime. Il peu donc se rendre compte de l'empilement des régimes, à savoir qu'un régime de sélection de fichier se produit au sein d'un régime d'association FTAM.

7.9. Unité d'interaction UI4.2 : F-SELECT resp(-ve)

Cette unité d'interaction est activée par un refus de la primitive **F-SELECT request** de la part du répondeur de l'application FTAM. Dans ce cas, les deux entités paires restent en régime d'association et aucun fichier n'est sélectionné.

La séquence d'événements suivante s'opère :
Agent-Répondeur → Affiche("F-SELECT resp (YYYYYY) ↓")
Messages → Affiche("L'agent répondeur refuse la sélection du fichier qui lui était demandée. Il envoie sa réponse à l'initiateur.")

Etat-Répondeur → Modifie-Etat(F_SEL_RS-)
Fournisseur-FTAM → Affiche-Envoi(RI)
Etat-Initiateur → Modifie-Etat(F_SEL_RS-)
Agent-Initiateur → Affiche("↑ F-SELECT conf (YYYYYY)")
Messages → Affiche("Le fournisseur FTAM transmet la confirmation de la requête
de sélection de fichier à l'agent initiateur. La sélection a été
refusée par l'entité paire.")
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Agent-Initiateur → Efface
goto UI0

7.10. Unité d'interaction UI5 : F-CREATE

La seconde manière pour accéder au régime de sélection de fichier est de créer un nouveau fichier. Ci-dessous figurent les événements découlant d'une primitive **F-CREATE request**. Ils sont similaires à ceux exécutés lors d'une sélection de fichier. La différence (au sens de la simulation) réside essentiellement dans la manière de donner un nom de fichier. Dans le cas d'une sélection, le nom du fichier est relatif à un fichier préexistant. Ici, le nom du fichier est donné par l'utilisateur. Il représente l'identifiant du fichier que l'on souhaite créer.

Les actions qui suivent, illustrent les changements effectués sur les objets :

Messages → Affiche("Donnez un nom de fichier.")
nom-fichier = Agent-Initiateur → Lire-nom
Agent-Initiateur → Affiche("↓ F-CREATE req (XXXXXX)")
Messages → Affiche("L'agent FTAM initiateur demande la création d'un fichier.")

if not Etat-Initiateur → Etat-Valide?(F_CRE_RQ)
then
Messages → Affiche-Erreur("Vous avez sélectionné une primitive invalide
dans l'état actuel de la machine de protocole de fichier.")
Agent-Initiateur → Efface
goto UI0
endif

Etat-Initiateur → Modifie-Etat(F_CRE_RQ)

Fournisseur-FTAM → Affiche-Envoi(IR)
Etat-Répondeur → Modifie-Etat(F_CRE_RQ)
Agent-Répondeur → Affiche("F-CREATE ind (XXXXXX) ↑")
Messages → Affiche("Le fournisseur FTAM donne à l'agent répondeur la
demande de création de fichier envoyée par l'agent répondeur.")
Agent-Initiateur → Efface
Fournisseur-FTAM → Efface-Envoi
Messages → Affiche-Aide("En tant qu'agent répondeur vous pouvez accepter ou
refuser la création du fichier. Acceptez-vous?")
choix-utilisateur = Agent-Répondeur → Accepte-Choix
Agent-Répondeur → Efface

case choix-utilisateur of
"Y" : goto UI5.1
"N" : goto UI5.2
endcase

7.11. Unité d'interaction UI5.1 : F-CREATE resp(+ve)

Cette unité d'interaction est déclenchée dès que l'utilisateur de la simulation a opté pour l'acceptation de la primitive **F-CREATE request**. L'entité répondante confirmant la création, les partenaires FTAM entreront dans le régime de sélection de fichier (**File Selection Regime**).

La réalisation des changements relatifs au déroulement de cette unité d'interaction se caractérise par la séquence d'événements suivante :

Fichier-Répondeur → Affiche-Fichier(nom-fichier)
Fichier-Répondeur → Affiche-Attributs
Agent-Répondeur → Affiche("F-CREATE resp (YYYYYY) ↓")
Messages → Affiche("L'agent FTAM répondeur accepte la création du fichier.")
Etat-Répondeur → Modifie-Etat(F_CRE_RS+)
Fournisseur-FTAM → Affiche-Envoi(RI)
Etat-Initiateur → Modifie-Etat(F_CRE_RS+)
Agent-Initiateur → Affiche("↑ F-CREATE conf (YYYYYY)")
Messages → Affiche("Le fournisseur FTAM signale à l'agent initiateur le succès
de la création.")
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Régime → Entrer-Régime

Fichier-Initiateur → Affiche-Fichier(nom-fichier)
Agent-Initiateur → Efface
goto UI0

7.12. Unité d'interaction UI5.2 : F-CREATE resp(-ve)

Cette unité d'interaction est activée par un refus de la primitive **F-CREATE request** de la part du répondeur de l'application FTAM. Dans ce cas, les deux entités paires restent en régime d'association et aucun fichier n'est créé.

La séquence d'événements suivante s'opère :

Agent-Répondeur → Affiche("F-CREATE resp (YYYYYY) ↓")
Messages → Affiche("L'agent FTAM répondeur refuse la création du fichier.")
Etat-Répondeur → Modifie-Etat(F_CRE_RS-)
Fournisseur-FTAM → Affiche-Envoi(RI)
Etat-Initiateur → Modifie-Etat(F_CRE_RS-)
Agent-Initiateur → Affiche("↑ F-CREATE conf (YYYYYY)")
Messages → Affiche("Le fournisseur FTAM donne à l'agent initiateur une
confirmation renseignant le refus donné à la demande de
création.")
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Agent-Initiateur → Efface
goto UI0

7.13. Unité d'interaction UI6 : F-DESELECT

En sélectionnant la primitive **F-DESELECT request**, l'utilisateur quitte le régime de sélection de fichier. Le fichier en cours d'utilisation n'est plus sélectionné et bien sûr les attributs du fichier ne sont plus disponibles. Cet objectif est atteint au travers de la séquence d'actions suivante :

Agent-Initiateur → Affiche("↓ F-DESELECT req (XXXXXX)")
Messages → Affiche("L'agent initiateur demande la fin de la sélection du fichier
courant.")

if not Etat-Initiateur → Etat-Valide?(F_DESEL_RQ)
then
Messages → Affiche-Erreur("Vous avez sélectionné une primitive invalide
dans l'état actuel de la machine de protocole de fichier.")

Agent-Initiateur → Efface
goto UI0
endif

Etat-Initiateur → Modifie-Etat(F_DESEL_RQ)
Fournisseur-FTAM → Affiche-Envoi(IR)
Etat-Répondeur → Modifie-Etat(F_DESEL_RQ)
Agent-Répondeur → Affiche("F-DESELECT ind (XXXXXX) ↑")
Messages → Affiche("Le fournisseur FTAM signale à l'agent répondeur une fin de sélection du fichier courant.")
Agent-Initiateur → Efface
Fournisseur-FTAM → Efface-Envoi
Fichier-Répondeur → Efface
Agent-Répondeur → Efface
Agent-Répondeur → Affiche("F-DESELECT resp (YYYYYY) ↓")
Messages → Affiche("L'agent répondeur envoie une réponse pour signaler qu'il a reçu et exécuté la demande de fin de sélection.")
Etat-Répondeur → Modifie-Etat(F_DESEL_RS)
Fournisseur-FTAM → Affiche-Envoi(RI)
Etat-Initiateur → Modifie-Etat(F_DESEL_RS)
Agent-Initiateur → Affiche("↑ F-DESELECT conf (YYYYYY)")
Messages → Affiche("Le fournisseur FTAM confirme la fin de la sélection du fichier courant à l'entité initiatrice.")
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Fichier-Initiateur → Efface
Régime → Sortir-Régime
Agent-Initiateur → Efface
goto UI0

Après avoir décidé de désélectionner un fichier, l'apprenant constate les effets en observant le comportement du didacticiel. Visuellement, il remarque la disparition des objets Fichier-Initiateur et Fichier-Répondeur. De plus il est averti d'un changement de régime.

7.14. Unité d'interaction UI7 : F-DELETE

En sélectionnant la primitive **F-DELETE request**, l'utilisateur quitte le régime de sélection de fichier. Le fichier est supprimé du répertoire. Cet objectif est atteint au travers de la séquence d'actions suivante :

Agent-Initiateur → Affiche("↓ F-DELETE req (XXXXXX)")
Messages → Affiche("L'agent initiateur demande l'effacement du fichier courant.")

if not Etat-Initiateur → Etat-Valide?(F_DEL_RQ)
then
  Messages → Affiche-Erreur("Vous avez sélectionné une primitive invalide
    dans l'état actuel de la machine de protocole de fichier.")
  Agent-Initiateur → Efface
  goto UI0
endif

Etat-Initiateur → Modifie-Etat(F_DEL_RQ)
Fournisseur-FTAM → Affiche-Envoi(IR)
Etat-Répondeur → Modifie-Etat(F_DEL_RQ)
Agent-Répondeur → Affiche("F-DELETE ind (XXXXXX)↑")
Messages → Affiche("Le fournisseur FTAM signale à l'entité répondante
    l'effacement du fichier courant.")
Agent-Initiateur → Efface
Fournisseur-FTAM → Efface-Envoi
Fichier-Répondeur → Efface
Agent-Répondeur → Efface
Agent-Répondeur → Affiche("F-DELETE resp (YYYYYY) ↓")
Messages → Affiche("L'agent répondeur répond à la requête d'effacement de
    fichier.")
Etat-Répondeur → Modifie-Etat(F_DEL_RS)
Fournisseur-FTAM → Affiche-Envoi(RI)
Etat-Initiateur → Modifie-Etat(F_DEL_RS)
Agent-Initiateur → Affiche("↑ F-DELETE conf (YYYYYY)")
Messages → Affiche("Le fournisseur FTAM donne à l'agent initiateur la
    confirmation de l'effacement du fichier.")
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi

Fichier-Initiateur → Efface

Régime → Sortir-Régime

Agent-Initiateur → Efface

goto UI0

Du point de vue du scénario, la nuance n'est pas énorme entre une désélection et un effacement de fichier. Il faudrait s'arranger pour que l'apprenant puisse constater la différence. Un fichier effacé ne pourra plus être sélectionné. Donc, il faudra prévoir un répertoire de fichiers qui sera tenu à jour en fonction des actions sur ce répertoire.

7.15. Unité d'interaction UI8 : F-READ-ATTRIB

En sélectionnant la primitive **F-READ-ATTRIB request**, l'utilisateur interroge les attributs du fichier sélectionné. Les paramètres de la requête identifient les attributs souhaités, la réponse renvoie les attributs correspondants. Cette lecture est réalisée par la séquence d'événements qui suit :

Agent-Initiateur → Affiche("↓ F-READ-ATTRIB req (XXXXXX)")

Messages → Affiche("L'agent initiateur désire lire certains attributs du fichier.")

if not Etat-Initiateur → Etat-Valide?(F_RDATT_RQ)

then

Messages → Affiche-Erreur("Vous avez sélectionné une primitive invalide dans l'état actuel de la machine de protocole de fichier.")

Agent-Initiateur → Efface

goto UI0

endif

Etat-Initiateur → Modifie-Etat(F_RDATT_RQ)

Fournisseur-FTAM → Affiche-Envoi(IR)

Etat-Répondeur → Modifie-Etat(F_RDATT_RQ)

Agent-Répondeur → Affiche("F-READ-ATTRIB ind (XXXXXX) ↑")

Messages → Affiche("Le fournisseur FTAM indique à l'agent répondeur la demande de lecture d'attributs du fichier à l'initiative de l'agent répondeur.")

Agent-Initiateur → Efface

Fournisseur-FTAM → Efface-Envoi

Agent-Répondeur → Efface

Fichier-Répondeur → Clignote-Attributs

Agent-Répondeur → Affiche("F-READ-ATTRIB resp (YYYYYY) ↓")

Messages → Affiche("L'agent répondeur envoie une réponse contenant les attributs du fichier lu.")

Etat-Répondeur → Modifie-Etat(F_RDATT_RS)

Fournisseur-FTAM → Affiche-Envoi(RI)

Etat-Initiateur → Modifie-Etat(F_RDATT_RS)

Agent-Initiateur → Affiche("↑ F-READ-ATTRIB conf (YYYYYY)")

Messages → Affiche("Le fournisseur FTAM transmet la confirmation de lecture des attributs à l'agent FTAM initiateur.")

Agent-Répondeur → Efface

Fournisseur-FTAM → Efface-Envoi

Fichier-Initiateur → Affiche-Attributs

Agent-Initiateur → Efface

goto UI0

La lecture des attributs du fichier est rendue très visuelle par l'utilisation de changements de couleur, de clignotement, de messages d'accompagnement, .. Expliquons davantage le coeur de la spécification de cette unité d'interaction. La primitive de requête de lecture des attributs est envoyée à l'agent répondeur. L'envoi du PDU est simulé par l'apparition de pointillés dans l'objet Fournisseur. L'agent répondeur reçoit la demande de lecture. A ce moment, une partie des attributs (celle qui correspond aux attributs demandés par l'initiateur) commence à clignoter. Ce processus dure un certain temps pour permettre à l'utilisateur d'observer à son aise les réactions du didacticiel. Le **clignotement** représente la **lecture** Ensuite, le répondeur envoie une réponse contenant les attributs lus. La réponse parvient à l'initiateur et une boîte contenant les attributs lus apparaît côté initiateur. Nous ne pensons pas que ce symbolisme soit suffisant pour que l'apprenant se rende compte de ce qui se passe. C'est pourquoi chaque étape est accompagnée d'un message d'explication.

7.16. Unité d'interaction UI9 : F-CHANGE-ATTRIB

La primitive **F-CHANGE-ATTRIB request** permet à l'utilisateur de modifier les attributs du fichier sélectionné. Les attributs de la requête identifient les attributs que l'utilisateur souhaite changer, la réponse renvoie la liste des attributs réellement modifiés. Cette modification est réalisée par la séquence d'événements suivante :

Agent-Initiateur → Affiche("↓ F-CHANGE-ATTRIB req(XXXXXX)")

Messages → Affiche("L'agent FTAM initiateur désire changer quelques attributs du fichier.")

```
if not Etat-Initiateur → Etat-Valide?( F_CHATT_RQ )
then
   Messages → Affiche-Erreur( "Vous avez sélectionné une primitive invalide
                dans l'état actuel de la machine de protocole de fichier." )
   Agent-Initiateur → Efface
   goto UI0
endif

Etat-Initiateur → Modifie-Etat( F_CHATT_RQ )
Fournisseur-FTAM → Affiche-Envoi( IR )
Etat-Répondeur → Modifie-Etat( F_CHATT_RQ )
Agent-Répondeur → Affiche( "F-CHANGE-ATTRIB ind ( XXXXXX ) ↑" )
Messages → Affiche( "Le fournisseur FTAM informe l'agent répondeur du
                changement d'attributs demandés par l'agent FTAM pair." )
Agent-Initiateur → Efface
Fournisseur-FTAM → Efface-Envoi
Agent-Répondeur → Efface
Fichier-Répondeur → Clignote-Attributs
Agent-Répondeur → Affiche( "F-CHANGE-ATTRIB resp ( YYYYYY ) ↓" )
Messages → Affiche( "L'agent répondeur FTAM envoie une réponse reprenant les
                attributs du fichier qui ont été changés." )
Etat-Répondeur → Modifie-Etat( F_CHATT_RS )
Fournisseur-FTAM → Affiche-Envoi( RI )
Etat-Initiateur → Modifie-Etat( F_CHATT_RS )
Agent-Initiateur → Affiche( "↑ F-CHANGE-ATTRIB conf ( YYYYYY )" )
Messages → Affiche( "Le fournisseur FTAM transmet les changements effectués
                sur les attributs dans la primitive de confirmation." )
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Fichier-Initiateur → Clignote-Attributs
Agent-Initiateur → Efface
goto UI0
```

Lors d'une modification des attributs du fichier, l'initiateur donne en paramètre de la primitive de requête le nom des attributs à changer et la valeur de remplacement. Le répondeur va alors modifier les attributs. Cela est simulé par un changement de couleur des attributs impliqués et d'un **clignotement** de ceux-ci pour imager l'**écriture**. Ensuite la réponse de

l'agent répondeur parvient à l'initiateur. Le constat des changements réalisés s'effectue en faisant **clignoter**les attributs modifiés dans la boîte des attributs du fichier se trouvant du côté initiateur.

Nous remarquons immédiatement que nous avons utilisé le clignotement des attributs du fichier dans trois contextes différents. La première fois lors de la lecture des attributs (unité d'interaction UI8), la deuxième fois lors de l'écriture des attributs et la troisième pour indiquer les attributs modifiés. Afin de ne pas utiliser strictement le même style lors d'opérations de types distincts, nous suggérons l'utilisation de couleurs différentes pour chaque mode.

7.17. Unité d'interaction UI10 : F-OPEN

L'utilisateur qui se trouve dans un régime de sélection de fichier peut entrer dans un régime d'ouverture de fichier de manière à pouvoir accéder au contenu du fichier. Ci-dessous figurent les événements découlant d'une primitive **F-OPEN request**:

```
Agent-Initiateur → Affiche( "↓ F-OPEN req ( XXXXXX )" )
Messages → Affiche( "L'agent FTAM initiateur demande l'ouverture du fichier." )

if not Etat-Initiateur → Etat-Valide?( F_OPN_RQ)
then
   Messages → Affiche-Erreur( "Vous avez sélectionné une primitive invalide
                  dans l'état actuel de la machine de protocole de fichier." )
   Agent-Initiateur → Efface
   goto UI0
endif

Etat-Initiateur → Modifie-Etat( F_OPN_RQ )
Fournisseur-FTAM → Affiche-Envoi( IR )
Etat-Répondeur → Modifie-Etat( F_OPN_RQ )
Agent-Répondeur → Affiche( "F-OPEN ind ( XXXXXX ) ↑" )
Messages → Affiche( "Le fournisseur FTAM signale à l'entité répondante la
                  demande d'ouverture de fichier qui lui est faite." )
Agent-Initiateur → Efface
Fournisseur-FTAM → Efface-Envoi
Messages → Affiche-Aide( "En tant qu'agent répondeur, acceptez-vous ou
                  refusez-vous d'ouvrir le fichier?" )
choix-utilisateur = Agent-Répondeur → Accepte-Choix
```

Agent-Répondeur → Efface

case choix-utilisateur of
"Y" : goto UI10.1
"N" : goto UI10.2
endcase

7.18. Unité d'interaction UI10.1 : F-OPEN resp(+ve)

Cette unité d'interaction est déclenchée dès que l'utilisateur de la simulation a opté pour l'acceptation de la primitive **F-OPEN request**. L'entité répondante confirmant l'ouverture, les partenaires entreront dans le régime d'ouverture de fichier (**File Open Regime**). Notamment, le contenu du fichier est accessible.

La réalisation des changements relatifs au déroulement de cette unité d'interaction se caractérise par la séquence d'événements qui suit :

Fichier-Répondeur → Affiche-Contenu
Agent-Répondeur → Affiche("F-OPEN resp (YYYYYY) ↓")
Messages → Affiche("L'agent répondeur accepte l'ouverture du fichier. La structure du fichier est disponible.")
Etat-Répondeur → Modifie-Etat(F_OPN_RS+)
Fournisseur-FTAM → Affiche-Envoi(RI)
Etat-Initiateur → Modifie-Etat(F_OPN_RS+)
Agent-Initiateur → Affiche("↑ F-OPEN conf (YYYYYY)")
Messages → Affiche("L'agent initiateur reçoit la confirmation de l'ouverture du fichier.")
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Régime → Entrer-Régime
Agent-Initiateur → Efface
goto UI0

La structure du fichier qui est affichée côté répondeur est du type décrit dans la première partie de ce chapitre, à savoir **séquentiel plat**. Lors de l'ouverture du fichier, le noeud racine est mis en évidence comme précisé dans la norme. Cela signifie que le FADU sélectionné contient le fichier entier. L'utilisateur peut désormais modifier le noeud sélectionné.

7.19. Unité d'interaction UI10.2 : F-OPEN resp(-ve)

Cette unité d'interaction est activée par un refus de la primitive **F-OPEN request** de la part du répondeur de l'application FTAM. Dans ce cas, les deux entités paires restent en régime de sélection de fichier et aucun fichier n'est ouvert.

La séquence d'événement suivante s'opère :

Agent-Répondeur → Affiche("F-OPEN resp (YYYYYY) ↓")
Messages → Affiche("L'agent répondeur FTAM refuse l'ouverture du fichier.")
Etat-Répondeur → Modifie-Etat(F_OPN_RS-)
Fournisseur-FTAM → Affiche-Envoi(RI)
Etat-Initiateur → Modifie-Etat(F_OPN_RS-)
Agent-Initiateur → Affiche("↑ F-OPEN conf (YYYYYY)")
Messages → Affiche("Le fournisseur FTAM signale à l'agent initiateur l'échec de sa tentative d'ouverture du fichier.")
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Agent-Initiateur → Efface
goto UI0

7.20. Unité d'interaction UI11 : F-CLOSE

L'utilisateur qui sélectionne la primitive **F-CLOSE request** quitte le régime d'ouverture de fichier et revient au régime de sélection de fichier. La structure et le contenu du fichier ne sont plus disponibles pour l'utilisateur. Par contre les attributs du fichier sont à nouveau à sa disposition. Cet objectif est atteint au travers de la séquence d'actions suivante :

Agent-Initiateur → Affiche("↓ F-CLOSE req ")
Messages → Affiche("L'agent initiateur FTAM veut fermer le fichier courant.")

if not Etat-Initiateur → Etat-Valide?(F_CLO_RQ)
then
  Messages → Affiche-Erreur("Vous avez sélectionné une primitive invalide dans l'état actuel de la machine de protocole de fichier.")
  Agent-Initiateur → Efface
  goto UI0
endif

Etat-Initiateur → Modifie-Etat(F_CLO_RQ)
Fournisseur-FTAM → Affiche-Envoi(IR)
Etat-Répondeur → Modifie-Etat(F_CLO_RQ)
Agent-Répondeur → Affiche("F-CLOSE ind ↑")
Messages → Affiche("Le fournisseur FTAM envoie une indication à l'agent répondeur lui signalant la fermeture du fichier.")
Agent-Initiateur → Efface
Fournisseur-FTAM → Efface-Envoi
Agent-Répondeur → Efface
Fichier-Répondeur → Efface-Contenu
Agent-Répondeur → Affiche("F-CLOSE resp (YYYYYY) ↓")
Messages → Affiche("L'agent FTAM répondeur donne sa réponse au sujet de la fermeture du fichier à son fournisseur.")
Etat-Répondeur → Modifie-Etat(F_CLO_RS)
Fournisseur-FTAM → Affiche-Envoi(RI)
Etat-Initiateur → Modifie-Etat(F_CLO_RS)
Agent-Initiateur → Affiche("↑ F-CLOSE conf (YYYYYY)")
Messages → Affiche("Le fournisseur FTAM donne la confirmation de fermeture de fichier à l'agent initiateur.")
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Régime → Sortir-Régime
Fichier-Initiateur → Efface-Contenu
Agent-Initiateur → Efface
goto UI0

Comme toujours lorsque nous quittons un régime, un certain nombre d'objets disparaissent de l'écran. La signification que l'on doit donner à cette disparition est liée au domaine d'action de l'utilisateur. Quand, dans le cas de la fermeture de fichier, nous utilisons la méthode

Fichier-Répondeur → Efface-Contenu,

la structure de donnée disparaît. Cette action est cohérente car en mode de sélection de fichier (où l'utilisateur retourne après la fermeture), seuls les attributs du fichier sont à la disposition de l'utilisateur. C'est la raison pour laquelle la méthode Efface-Contenu restaure la boîte des attributs du fichier (cfr. point 5.6).

7.21. Unité d'interaction UI12 : F-LOCATE

La primitive **F-LOCATE request** permet à l'utilisateur de se positionner sur un noeud précis de la structure du fichier. Cet objectif est atteint au travers de la séquence d'actions suivante :

Agent-Initiateur → Affiche("↓ F-LOCATE req ")
Messages → Affiche("L'agent initiateur veut sélectionner un noeud de l'arbre du fichier.")

if not Etat-Initiateur → Etat-Valide?(F_LOC_RQ)
then
  Messages → Affiche-Erreur("Vous avez sélectionné une primitive invalide dans l'état actuel de la machine de protocole de fichier.")
  Agent-Initiateur → Efface
  goto UI0
endif

Etat-Initiateur → Modifie-Etat(F_LOC_RQ)
Fournisseur-FTAM → Affiche-Envoi(IR)
Etat-Répondeur → Modifie-Etat(F_LOC_RQ)
Agent-Répondeur → Affiche("F-LOCATE ind ↑")
Messages → Affiche("Le fournisseur FTAM donne à l'agent répondeur l'indication de la sélection d'un noeud.")
Agent-Initiateur → Efface
Fournisseur-FTAM → Efface-Envoi
Messages → Affiche("Selectionnez un noeud du fichier.")
noeud = Fichier-Répondeur → Choisir-FADU
Fichier-Répondeur → Localise-Noeud(noeud)
Agent-Répondeur → Efface
Agent-Répondeur → Affiche("F-LOCATE resp (YYYYYY) ↓")
Messages → Affiche("L'agent répondeur FTAM envoie la réponse concernant le noeud choisi.")
Etat-Répondeur → Modifie-Etat(F_LOC_RS)
Fournisseur-FTAM → Affiche-Envoi(RI)
Etat-Initiateur → Modifie-Etat(F_LOC_RS)
Agent-Initiateur → Affiche("↑ F-LOCATE conf (YYYYYY)")

Messages —> Affiche("Le fournisseur FTAM envoie une confirmation à l'agent répondeur concernant le noeud sélectionné.")

Agent-Répondeur —> Efface

Fournisseur-FTAM —> Efface-Envoi

Agent-Initiateur —> Efface

goto UI0

D'un point de vue strictement "simulation", cette unité d'interaction n'a pour seul effet que de mettre en évidence un FADU sélectionné par l'utilisateur. Cela veut dire que le noeud précédemment mis en évidence (par une couleur différente des autres noeuds) retrouve une couleur correspondant à la couleur normale des noeuds. Le noeud identifiant le FADU sélectionné prend la couleur spécifiée pour effectuer la mise en évidence. Il est certain que le but avoué de cette unité d'interaction ne se limite pas à ce changement de couleur. Nous aimerions faire comprendre que l'utilisation de la primitive **F-LOCATE request**à une portée plus grande dans FTAM que la simple identification d'un FADU comme celle invoquée en paramètre d'une primitive de lecture (**F-READ**) ou d'écriture (**F-WRITE**).

Faisons remarquer que la manière de sélectionner un noeud est discutable point de vue pédagogique. La sélection du noeud se réalise via un paramètre du request et non pas comme notre scénario pourait le faire croire après réception de l'indication. Il faut trouver un compromis entre choix interactif (cliquer directement sur le noeud) et choix par entrée du nom du noeud.

7.22. Unité d'interaction UI13 : F-ERASE

L'utilisateur qui sélectionne la primitive **F-ERASE request** va pouvoir se positionner sur un noeud précis de la structure du fichier et effacer le sous-arbre associé à ce noeud. De part la forme de structure utilisée (fichier séquentiel plat), effacer un FADU revient à effacer tout le fichier (si le noeud sélectionné est le noeud racine) ou à effacer un noeud et le DU associé (si la position courante est celle d'un noeud fils du noeud racine).

Cet objectif est atteint au travers de la séquence d'actions suivante :

Agent-Initiateur —> Affiche("↓ F-ERASE req ")

Messages —> Affiche("L'agent FTAM initiateur sélectionne un noeud et veut effacer le sous-arbre associé à ce noeud.")

if not Etat-Initiateur —> Etat-Valide?(F_ERA_RQ)

then

```
   Messages → Affiche-Erreur( "Vous avez sélectionné une primitive invalide
                  dans l'état actuel de la machine de protocole de fichier." )
   Agent-Initiateur → Efface
   goto UI0
endif

Etat-Initiateur → Modifie-Etat( F_ERA_RQ )
Fournisseur-FTAM → Affiche-Envoi( IR )
Etat-Répondeur → Modifie-Etat( F_ERA_RQ )
Agent-Répondeur → Affiche( "F-ERASE ind ↑" )
Messages → Affiche( "Le fournisseur FTAM indique à l'agent répondeur
                  l'effacement du sous-arbre associé à un noeud." )
Agent-Initiateur → Efface
Fournisseur-FTAM → Efface-Envoi
Agent-Répondeur → Efface
Messages → Affiche( "Choisissez le noeud racine du FADU qui sera effacé." )
noeud-choisi = Fichier-Répondeur → Choisir-FADU
Fichier-Répondeur → Localise-Noeud( noeud-choisi )

if Fichier-Répondeur → Racine?( noeud-choisi )
then Fichier-Répondeur → Efface-Structure
else Fichier-Répondeur →  Effacer-Noeud(noeud-choisi)
endif

Agent-Répondeur → Affiche( "F-ERASE resp ( YYYYYY ) ↓" )
Messages → Affiche( "L'agent répondeur FTAM envoie une réponse concernant
                  l'effacement d'une partie du fichier." )
Etat-Répondeur → Modifie-Etat( F_ERA_RS )
Fournisseur-FTAM → Affiche-Envoi( RI )
Etat-Initiateur → Modifie-Etat( F_ERA_RS )
Agent-Initiateur → Affiche( "↑ F-ERASE conf ( YYYYYY )" )
Messages → Affiche( "Le fournisseur FTAM donne à l'agent initiateur la
                  confirmation de l'effacement du sous-arbre associé au noeud
                  choisi.")
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Agent-Initiateur → Efface
```

goto UI0

7.23. Unité d'interaction UI14 : F-READ

L'utilisateur qui sélectionne la primitive **F-READ request** va pouvoir se positionner sur un noeud précis de la structure du fichier et lire le sous-arbre attaché à ce noeud. Durant le transfert de données, le régime courant est **Data Transfer Regime**. Cet objectif est atteint au travers de la séquence d'actions suivante :

```
Agent-Initiateur —> Affiche( "↓ F-READ req " )
Messages —> Affiche( "L'agent FTAM initiateur sélectionne un noeud du fichier et
                      lit le sous-arbre relatif à ce noeud." )

if not Etat-Initiateur —> Etat-Valide?( F_RD_RQ )
then
   Messages —> Affiche-Erreur( "Vous avez sélectionné une primitive invalide
                      dans l'état actuel de la machine de protocole de fichier." )
   Agent-Initiateur —> Efface
   goto UI0
endif

Fichier-Initiateur —> Affiche-Contenu
Régime —> Entrer-Régime
Etat-Initiateur —> Modifie-Etat( F_RD_RQ )
Fournisseur-FTAM —> Affiche-Envoi( IR )
Etat-Répondeur —> Modifie-Etat( F_RD_RQ )
Agent-Répondeur —> Affiche( "F-READ ind ↑" )
Messages —> Affiche( "Le fournisseur FTAM donne à l'agent répondeur l'indication
                      d'une lecture de fichier." )
Agent-Initiateur —> Efface
Fournisseur-FTAM —> Efface-Envoi
Messages —> Affiche( "Sélectionnez un noeud de l'arbre du fichier." )
noeud-choisi = Fichier-Répondeur —> Choisir-FADU
Fichier-Répondeur —> Localise-Noeud( noeud-choisi )
Agent-Répondeur —> Efface

if Fichier-Répondeur —> Racine?( noeud-choisi )
then
```

```
    noeud-départ   =noeud-choisi
    Agent-Répondeur  → Affiche( "F-DATA req ( XXXXXX ) ↓" )
    *** XXXXXX contient le noeud-choisi, ici il correspond à la racine du fichier ***
    Messages → Affiche( "L'agent répondeur FTAM (envoyeur) transmet les
                    données relatives au sous-arbre à lire." )
    Fournisseur-FTAM → Affiche-Envoi( RI )
    Agent-Initiateur → Affiche( "↑ F-DATA ind ( XXXXXX )" )
    Messages → Affiche( "Le fournisseur FTAM donne les données à l'agent
                    initiateur (receveur)." )
    Agent-Répondeur  → Efface
    Fournisseur-FTAM → Efface-Envoi
    Agent-Initiateur → Efface
    Fichier-Initiateur → Créer-Racine( noeud-choisi )

    do while not noeud-choisi = Fichier-Répondeur → Dernier-noeud?
       noeud-choisi = Fichier-Répondeur → Noeud-Suivant( noeud-choisi )
       goto UI14.1
    enddo

    Fichier-Répondeur → Localise( noeud-départ )

else

    goto UI14.1

endif

Agent-Répondeur  → Affiche( "↓ F-DATA-END req " )
Messages → Affiche( "L'agent FTAM répondeur a terminé l'envoi de toutes les
                    données." )
Etat-Répondeur → Modifie-Etat( F_EDAT_RQ )
Fournisseur-FTAM → Affiche-Envoi( RI )
Etat-Initiateur → Modifie-Etat( F_EDAT_RQ )
Agent-Initiateur → Affiche( "F-DATA-END ind ↑" )
Messages → Affiche( "L'agent initiateur (receveur) reçoit une indication de fin de
                    lecture de fichier." )
Agent-Répondeur  → Efface
```

Fournisseur-FTAM → Efface-Envoi

Agent-Initiateur → Efface

goto UI0

Nous constatons que l'agent répondeur (**Responder**) devient l'envoyeur (**Sender**) tandis que l'agent initiateur (**Initiator**) devient le receveur (**Receiver**). C'est une notion fondamentale pour comprendre le transfert de données. Afin de mettre en évidence cet état de fait, nous proposons d'ajouter un libellé dans les objets Initiateur et Répondeur. Lors d'une lecture, nous inscrirons sous le littéral **Initiator** un libellé **Receiver**, et sous le **Responder** figurera **Sender**.

Le transfert de données utilise la primitive **F-DATA request** pour véhiculer les données. Le nombre des primitives **F-DATA request** utilisées est fonction de la structure du FADU, c'est-à-dire du nombre de noeuds constituant ce FADU. Lorsque l'envoyeur a terminé, il le signale à l'initiateur par l'entremise d'une primitive **F-DATA-END request**.

Remarquons que l'algorithme décrit ici, s'appuie sur le modèle simple de structure de fichier évoqué.

7.24. Unité d'interaction UI14.1 : F-READ(suite)

Cette unité d'interaction réalise le transfert d'un noeud et de l'unité de données attachée par la séquence d'événements suivante :

DU-choisi = Fichier-Répondeur → Lire-DU(noeud-choisi)

*** XXXXXX contient noeud-choisi et DU-choisi ***

Fichier-Répondeur → Transfert-Unité(noeud-choisi)

Agent-Répondeur → Affiche("F-DATA req (XXXXXX) ↓")

Messages → Affiche("L'agent FTAM répondeur (envoyeur) passe les données
en relation avec le sous-arbre à lire.")

Fournisseur-FTAM → Affiche-Envoi(RI)

Agent-Initiateur → Affiche("↑ F-DATA ind (XXXXXX)")

Messages → Affiche("Le fournisseur FTAM donne les données à l'agent initiateur
(receveur).")

Agent-Répondeur → Efface

Fournisseur-FTAM → Efface-Envoi

Fichier-Initiateur → Insérer-Unité(noeud-choisi, DU-choisi)

Fichier-Répondeur → Fin-Transfert-Unité(noeud-choisi)

Agent-Initiateur → Efface

7.25. Unité d'interaction UI15 : F-WRITE

L'utilisateur qui sélectionne la primitive **F-WRITE request** va pouvoir écrire des données qui s'ajouteront à l'arbre du fichier déjà existant. Durant le transfert de données, le régime courant est **Data Transfer Regime**. Cet objectif est atteint au travers de la séquence d'actions suivante :

Agent-Initiateur → Affiche("↓F-WRITE req")
Messages → Affiche("L'agent FTAM initiateur veut écrire des données dans le fichier.")

if not Etat-Initiateur → Etat-Valide?(F_WR_RQ)
then
  Messages → Affiche-Erreur("Vous avez sélectionné une primitive invalide dans l'état actuel de la machine de protocole de fichier.")
  Agent-Initiateur → Efface
  goto UI0
endif

Régime → Entrer-Régime
Etat-Initiateur → Modifie-Etat(F_WR_RQ)
Fournisseur-FTAM → Affiche-Envoi(IR)
Etat-Répondeur → Modifie-Etat(F_WR_RQ)
Agent-Répondeur → Affiche("F-WRITE ind ↑")
Messages → Affiche("Le fournisseur FTAM envoie une indication à l'agent répondeur pour lui signaler l'écriture dans le fichier.")
Agent-Initiateur → Efface
Fournisseur-FTAM → Efface-Envoi
Agent-Répondeur → Efface
goto UI0

Au sens de la simulation, cette unité d'interaction signale simplement à l'agent répondeur que l'initiateur va écrire sur le fichier. Nous pourrions adopter le même raisonnement que lors d'une lecture et proposer d'ajouter le libellé **Sender** dans l'objet Initiateur et celui de **Receiver** dans l'objet Répondeur.

L'écriture proprement dite est à la charge de l'apprenant qui doit sélectionner autant de fois la primitive **F-DATA request** qu'il souhaite ajouter d'unités de données à la structure existante. Il devra aussi indiquer

la fin du transfert de données en demandant la réalisation de la primitive **F-DATA-END request**

7.26. Unité d'interaction UI16 : F-DATA

L'utilisateur qui sélectionne la primitive **F-DATA request** va pouvoir écrire des données qui s'ajouteront à l'arbre du fichier déjà existant. Cet objectif est atteint au travers de la séquence d'actions suivante :

Agent-Initiateur → Affiche("↓ F-DATA req ")
Messages → Affiche("L'agent initiateur va ajouter des données dans le fichier.")

if not Etat-Initiateur → Etat-Valide?(F_DATA_RQ)

then
  Messages → Affiche-Erreur("Vous avez sélectionné une primitive invalide
    dans l'état actuel de la machine de protocole de fichier.")
  Agent-Initiateur → Efface
  goto UI0
endif

Etat-Initiateur → Modifie-Etat(F_DATA_RQ)
Fournisseur-FTAM → Affiche-Envoi(IR)
Etat-Répondeur → Modifie-Etat(F_DATA_RQ)
Agent-Répondeur → Affiche("F-DATA ind ↑")
Messages → Affiche("Le fournisseur FTAM donne l'indication d'arrivée de
    données à l'agent répondeur.")
Agent-Initiateur → Efface
Fournisseur-FTAM → Efface-Envoi
Agent-Répondeur → Efface
Fichier-Répondeur → Insérer-unité("", "")
Messages → Affiche("Une unité de données du fichier est ajoutée.")
goto UI0

7.27. Unité d'interaction UI17 : F-DATA-END

L'utilisateur qui sélectionne la primitive **F-DATA-END request** montre qu'il a terminé l'envoi des données(par des **F-DATA request**). Cet objectif est atteint au travers de la séquence d'actions suivante :

```
Agent-Initiateur —> Affiche( "↓ F-DATA-END req " )
Messages —> Affiche( "L'agent FTAM initiateur a terminé l'écriture de données." )

if not Etat-Initiateur —> Etat-Valide?( F_EDAT_RQ )
then
   Messages —> Affiche-Erreur( "Vous avez sélectionné une primitive invalide
                dans l'état actuel de la machine de protocole de fichier." )
   Agent-Initiateur —> Efface
   goto UI0
endif

Etat-Initiateur —> Modifie-Etat( F_EDAT_RQ )
Fournisseur-FTAM —> Affiche-Envoi( IR )
Etat-Répondeur —> Modifie-Etat( F_EDAT_RQ )
Agent-Répondeur —> Affiche( "F-DATA-END ind ↑" )
Messages —> Affiche( "Le fournisseur FTAM envoie une indication à l'agent
                répondeur signalant la fin de l'écriture." )
Agent-Initiateur —> Efface
Fournisseur-FTAM —> Efface-Envoi
Agent-Répondeur —> Efface
goto UI0
```

7.28. Unité d'interaction UI18 : F-TRANSFER-END

L'utilisateur sélectionne la primitive **F-TRANSFER-END request** pour quitter le régime de transfert de données. A la suite de cette primitive, les deux partenaires retrouvent le régime d'ouverture de fichier. Cet objectif est atteint au travers de la séquence d'actions suivante :

```
Agent-Initiateur —> Affiche( "↓ F-TRANSFER-END req ( XXXXXX ) " )
Messages —> Affiche( "L'agent initiateur signale la fin d'un transfert de données." )

if not Etat-Initiateur —> Etat-Valide?( F_ETRSF_RQ )
then
   Messages —> Affiche-Erreur( "Vous avez sélectionné une primitive invalide
                dans l'état actuel de la machine de protocole de fichier." )
   Agent-Initiateur —> Efface
   goto UI0
endif
```

Etat-Initiateur → Modifie-Etat(F_ETRSF_RQ)
Fournisseur-FTAM → Affiche-Envoi(IR)
Etat-Répondeur → Modifie-Etat(F_ETRSF_RQ)
Agent-Répondeur → Affiche("F-TRANSFER-END ind (XXXXXX) ↑")
Messages → Affiche("Le fournisseur indique à l'agent répondeur la fin d'un
transfert de données.")
Agent-Initiateur → Efface
Fournisseur-FTAM → Efface-Envoi
Agent-Répondeur → Efface
Agent-Répondeur → Affiche("F-TRANSFER-END resp (YYYYYY) ↓")
Messages → Affiche("L'agent répondeur FTAM envoie sa réponse concernant la
fin du transfert de données.")
Etat-Répondeur → Modifie-Etat(F_ETRSF_RS)
Fournisseur-FTAM → Affiche-Envoi(RI)
Etat-Initiateur → Modifie-Etat(F_ETRSF_RS)
Agent-Initiateur → Affiche("↑ F-TRANSFER-END conf (YYYYYY)")
Messages → Affiche("Le fournisseur FTAM donne la confirmation de fin de
transfert à l'agent initiateur.")
Agent-Répondeur → Efface
Fournisseur-FTAM → Efface-Envoi
Agent-Initiateur → Efface
Régime → Sortir-Régime
goto UI0

L'unité d'interaction UI18 clôture tout transfert de données. Les deux partenaires de l'association reviennent au régime d'ouverture de fichiers. De là, l'initiateur peut, s'il le désire, demander un nouveau transfert de données.

8. Conclusion concernant la spécification du scénario

Nous avons décrit les objets qui composent la scène et les méthodes qui permettent de manipuler ces objets. Le scénario est écrit sur base de ces méthodes. Nous avons maintenant une idée plus précise de ce qui doit apparaître à l'écran. Le pseudo-langage utilisé pour écrire le scénario nous a permis de séquencer les actions qui constituent la simulation. De plus, le découpage en unités d'interaction nous permet d'entrevoir plus sereinement la prochaine étape de conception : l'implémentation. En effet, nous allons pouvoir produire le logiciel pas à pas, en ajoutant progressivement les unités d'interaction

Chapitre 5 : Médiatisation

1. Introduction

Nous abordons dans ce chapitre l'étape de médiatisation de notre scénario. Cette médiatisation demande une deuxième analyse approfondie étant donné l'environnement d'implémentation. Rappelons que l'implémentation se fait dans un environnement graphique orienté objets. L'utilisation des objets de l'environnement est facilitée par la description choisie par le scénario. En effet nous avons basé le scénario sur la manipulation d'objets. Dans le cadre plus large de l'unité d'enseignement des télécommunications, il faudra envisager le développement de classes réutilisables. A titre d'exemple,nous pouvons citer l'objet "états de la machine de protocole de fichier" qui représente les états du côté initiateur et du coté répondeur comme nous l'avons largement décrit dans le chapitre consacré au scénario. Cet objet est très utile et peut être réutilisé pour exprimer les transitions d'états des autres couches du modèle ISO.

2. Analyse de l'existant

Pour développer notre application, il faut prendre en compte deux critères essentiels liés au logiciel RMG :

- d'une part, il faut considérer les classes existantes (les classes standards de RMG) qui peuvent nous aider à deux niveaux. Tout d'abord, certains objets peuvent être modélisés par l'utilisation d'instances de classes existantes. Citons par exemple un objet message qui peut être simplement la matérialisation d'une instance de la classe **RMGString**. Ensuite, certains objets devront être modélisés par l'utilisation de plusieurs classes distinctes. C'est en fait généralement le cas pour la définition d'une chaîne de caractères définie dans une instance de la classe **RMGString** mais utilisant une police de caractère que l'on trouve dans la classe **FontMngr**.
- d'autre part, il faut considérer les nouvelles classes à développer. Là aussi, les classes existantes serviront de support pour le développement de nouvelles classes. En fait, grâce au mécanisme d'héritage, il s'agira le plus souvent de spécialiser une ou plusieurs classes existantes en fonction de nos besoins. C'est de cette manière que nous avons créé une classe "Ftamétat" régissant l'affichage correct des états précédant, courant et suivant(s) d'une machine de protocole de fichier en fonction des primitives de service qui sont activées.

La question qui se pose naturellement est la suivante : comment choisir les objets de base de l'implémentation? Essayons d'apporter une réponse à cette

question. Les objets sont assez naturellement définis dans notre scénario. Nous considérons comme objets à part entière les éléments suivants :

- l'affichage des régimes FTAM;
- le fichier virtuel côté initiateur;
- l'agent initiateur FTAM;
- la machine de protocole de fichier côté initiateur FTAM;
- la machine de protocole de fichier côté répondeur FTAM;
- l'agent répondeur FTAM;
- le fichier virtuel côté répondeur;
- le fournisseur FTAM;
- l'affichage des messages.

La définition de ces objets est assez naturelle tant du point de vue de la conception du scénario que du point de vue de l'approche orientée objets. Les télécommunications concernent l'échange de messages entre machines (objets) différentes. Nous pourrions multiplier les exemples tant conceptuels que pratiques. Par exemple, au niveau conceptuel, une couche du modèle ISO requiert, en tant qu'utilisatrice, le service de son fournisseur par l'envoi d'un message. Si dans une architecture orientée objets, nous considérons l'objet couche utilisatrice et l'objet fournisseur l'échange d'un message revient à un simple appel à une méthode de l'objet fournisseur. Ainsi le concept d'échange de message est tout à fait naturel dans l'approche orientée objets. La modélisation de problèmes de télécommunication est ainsi grandement facilitée.

Néanmoins un problème capital se pose : les appels peuvent être imbriqués. Un objet en appelle un autre qui lui-même en appelle un autre, et ainsi de suite. Le nombre d'appels imbriqués est difficilement prévisible. Pour mieux comprendre la portée de ce phénomène, nous allons considérer l'exemple de la modélisation d'un transfert de fichier en approche orientée objet dans le cadre de notre scénario. L'agent initiateur envoie un message (il appelle une méthode) à son fournisseur dans lequel il lui demande de communiquer une demande de lecture de fichier (**F-READ request**). L'objet fournisseur se contente d'envoyer un message (appel de méthode) directement à l'agent répondeur (**F-READ indication**). L'objet Agent-Répondeur effectue alors une lecture sur le fichier, ce qui se traduit par l'envoi d'un message à destination de l'objet qui matérialise le fichier virtuel. L'objet fichier envoie à son tour un message à l'agent répondeur (par exemple la première unité de données à transférer) en utilisant une méthode de l'objet Répondeur. L'agent répondeur envoie un message à son fournisseur qui lui demande de transmettre l'information à l'agent initiateur et ainsi de suite. Les appels aux méthodes des objets sont imbriqués.

Sur un plan pédagogique, le problème se complique encore. En effet, l'envoi d'un message constitue en soi une information qui doit être communiquée d'une manière ou d'une autre à l'apprenant. Il nous faut exprimer l'information "envoi de message" au travers d'une animation graphique ou grâce à l'affichage d'un message par exemple. Or, l'envoi du message est un mécanisme fourni au niveau de la programmation. Il est par conséquent, complètement transparent pour l'apprenant. Pour remédier à ce problème, nous avons pensé à différentes solutions. Citons en quelques unes :

- une solution serait de réaliser la mise en évidence de l'envoi du message au niveau des objets eux-mêmes (nous pourrions, par exemple, faire clignoter les deux objets qui communiquent). Avant d'appeler une méthode d'un autre objet, l'objet considéré produirait une animation (clignotement, changement de couleurs, ...) ou afficherait un message.
- une autre solution revient à considérer le transfert du message comme effectué par un moyen de communication. Il suffit alors de transformer ce moyen de communication en objet. L'envoi se fait en trois temps. L'objet envoyeur donne son message à l'objet moyen de communication. L'animation est assurée par l'objet "moyen de communication" entre les deux objets qui souhaitent se communiquer. L'objet "moyen de communication" délivre le message à l'objet appelé.

Les deux approches sont défendables et compatibles dans le cas de la réalisation d'un didacticiel. Pour l'apprentissage, l'envoyeur d'un message devra signaler à l'apprenant l'envoi du message. Avec la première méthode, le message d'accompagnement pourra être particularisé pour chaque message envoyé. La deuxième méthode est intéressante parce qu'elle est plus élégante et somme toute modélise mieux la réalité. En effet, l'existence d'un moyen de communication est une information qu'il faut également transmettre à l'apprenant.

Il demeure le problème des appels imbriqués. Pour résoudre ce problème, nous allons adopter une politique tout à fait différente. Cette politique nous semble être très bonne en ce qui concerne la réalisation de didacticiels pour des raisons que nous développerons plus tard. Examinons d'abord cette méthode. Il s'agit de réaliser un programme qui se comporte comme le professeur. Lors d'une leçon, celui-ci utilise le matériel à sa disposition pour faciliter l'apprentissage de la matière. Notre programme est conçu de telle manière qu'il utilise les objets que nous avons définis pour parvenir à cet apprentissage. Les méthodes définies sur ces objets sont agencés pour réaliser la simulation. Le programme "professeur" envoie un message à un objet. L'objet va alors effectuer un certain nombre d'actions élémentaires (essentiellement affichage de messages et/ou petites animations) avant de rendre la main au

programme "professeur". Nous pouvons maintenant définir les avantages et les inconvénients de ce style de programmation :

- un avantage au niveau de l'implémentation est l'élimination des appels imbriqués. Tous les appels aux méthodes des objets sont effectués de manière séquentielle (Il est possible qu'un certain nombre d'appels contrôlés et limités dérogent à cette règle). L'initiative du lancement d'une partie de la simulation revient à l'utilisateur.
- un autre avantage réside dans l'extrême souplesse de ce style de programmation. Toutes les manipulations d'objets sont centralisées, ceci permet de modifier l'enchaînement de l'utilisation des objets. Par exemple d'un point de vue pédagogique, il peut être plus efficace d'afficher un message avant une animation ou l'inverse. Le but pédagogique poursuivi est différent. La présence du message avant l'animation rassure l'apprenant tandis que l'animation avant l'affichage du message induit une réflexion plus grande.
- un dernier avantage provient du fait que la manipulation des objets est relativement élémentaire. La plupart du temps, il s'agit d'activer une méthode de l'objet. Ceci permet de développer le scénario sans posséder des connaissances approfondies en programmation.
- un inconvénient réside dans la redondance du code du programme pour les différentes unités d'interaction. Bien souvent, au sein d'une même unité d'interaction, des morceaux de code sont identiques. Mais la redondance n'est pas gênante, elle n'influence pas la performance de l'exécution. Dans le cadre du développement pédagogique il nous semblait primordial de conserver la souplesse et la simplicité. De plus, le code est facilement lisible, et facilement transformable. Ceci est primordial si l'on pense qu'une partie de ce code peut être réutilisée pour les besoins d'une autre application.

3. Analyse des classes à développer

La partie difficile de la programmation concerne la définition des objets et des méthodes travaillant sur ces objets. La question essentielle à résoudre est : quand faut-il développer de nouvelles classes? La réponse n'est pas immédiate. De prime abord, nous pourrions considérer qu'il faut définir une nouvelle classe pour chaque objet de la scène. Comme nous l'avons déjà signalé, certains objets n'exigent pas la définition de nouvelles méthodes, si bien que l'on peut se contenter de les instancier à partir de classes existantes et d'utiliser les méthodes de ces classes. En ce qui concerne les autres objets, il n'est pas immédiat qu'il faille systématiquement définir de nouvelles classes. Il faut évaluer le niveau de réutilisabilité de la classe. En effet, si la classe est très spécialisée et très spécifique à l'application que l'on réalise, il vaut mieux créer l'objet et définir les méthodes au sein de l'application plutôt que passer par la définition d'une classe à part. Lorsque la réutilisabilité peut être envisagée, il faut définir une nouvelle classe. La réalisation d'une telle classe doit être relativement large pour être utilisée dans d'autres

applications. Il faut penser aux différentes méthodes qui permettront de manipuler cette classe.

Nous allons examiner l'exemple de la classe "Ftamétat" que nous avons développée en partie. Cette classe concerne la gestion des états d'une machine de protocole. Le développement de cette classe se justifie par le fait qu'elle est utilisable pour toutes les couches du modèle ISO, mais également par le fait que plusieurs instances de cette classe vont être utilisées dans notre application. Nous avons besoin de deux instances : une instance côté initiateur et une instance côté répondeur. Nous allons parler d'une classe plus générale que celle développée. Nous l'appellerons "ISOetat". Pour représenter les transitions d'état de la machine de protocole de fichier, nous devons prendre en compte les transitions possibles (pour plus de renseignements concernant les transitions possibles nous renvoyons le lecteur à l'annexe 3). Les transitions possibles sont déterminées par le fait que dans l'état actuel et pour une demande de changement d'état, il existe un nouvel état. Par exemple dans FTAM, la demande d'association FTAM (**F-INITIALIZE request**) n'aura d'effet que si la machine de protocole de fichier côté initiateur est dans l'état disponible (**IDLE**). La méthode fondamentale à définir revient à permettre l'appel de la machine avec, comme paramètre, la demande de changement d'état. La machine devra vérifier et signaler que ce changement est valide. Dans le cas où le changement d'état est autorisé, il faudra afficher le changement d'état. Pour chaque couche du modèle ISO, il faudra permettre la définition des états valides, des demandes de changement d'état valides et des transitions valides. Il faudra donc offrir des méthodes qui permettent de définir ces caractéristiques lors de la création d'une instance de la machine de protocole. Il est clair que d'autres objets, tels que les couches du modèle ISO, pourraient être définis de la même manière dans le cadre du développement complet du modèle.

Nous avons développé la classe "Ftametat". Elle ne reprend pas les méthodes d'initialisation des états de la machine. Ceux-ci ont été définis statiquement. La définition de la classe "ISOetat" est envisageable dans la suite par simple développement de la classe "Ftamétat". En fait, il faudra ajouter des méthodes permettant la définition dynamique des états.

Remarquons que la complexité de l'environnement RMG ne pousse pas à la réutilisation des classes déjà développées par d'autres personnes. Bien souvent, l'effort nécessaire pour utiliser une classe existante demande plus de temps que de développer soi-même les deux ou trois méthodes que l'on souhaite utiliser. De même, il n'est pas évident de prime abord de déterminer si une classe est appropriée pour ce que l'on veut faire. L'apprentissage des classes est long et peut se révéler peu fructueux à la fin.

4. Ajouts par rapport au scénario original

La matérialisation des interactions avec l'apprenant n'a pas été précisée lors de la spécification du scénario. Ainsi, à plusieurs reprises nous avons demandé à l'utilisateur de communiquer son choix à notre application sans préciser comment l'interaction entre l'utilisateur et la machine se réalise. Nous allons dans cette partie considérer des critères ergonomiques et pédagogiques pour choisir les formes d'interaction.

Faisons remarquer que dans un souci de cohérence avec l'environnement RMG, il est souhaitable d'utiliser les mêmes moyens d'interaction dans les applications. Ces moyens sont constitués par la souris dont le bouton gauche peut déclencher une réaction de l'objet sur lequel le curseur se trouve (si le programmeur a associé une action à cet objet) alors que le bouton droit permet la sélection dans un menu déroulant avec possibilité de sélection dans des sous-menus. De plus, certains sous-menus sont restreints à une boîte de saisie de données numériques ou alphanumériques.

4.1. La sélection des primitives FTAM par l'agent initiateur

Nous avons choisi de présenter les primitives FTAM dans un menu déroulant. Par souci de pédagogie, nous n'avons pas laissé la liberté de faire apparaître le menu n'importe où à l'écran comme c'est le cas dans l'environnement RMG. Le menu apparaît uniquement au niveau de l'agent initiateur. C'est lui en effet qui lance les primitives FTAM. Nous faisons l'hypothèse que l'apprenant s'identifie ainsi davantage à l'agent initiateur. Le menu a été entièrement réalisé par nos soins car aucune classe de RMG ne satisfaisait à nos besoins très particuliers. Nous souhaitons disposer d'un menu qui puisse être étendu au fur et à mesure que nous implémentons de nouvelles primitives. En cours d'exécution, nous souhaitons pouvoir modifier le menu de manière à afficher toutes les primitives ou seulement celles qui peuvent être exécutées dans l'état actuel de la machine de protocole de fichier. Nous souhaitons que le menu ne soit affiché que lorsque l'application le décide. Nous souhaitons que le menu soit affiché uniquement dans l'objet agent initiateur pour bien montrer que le choix des primitives se réalise à ce niveau. Dans l'état d'avancement de notre implémentation, nous n'avons pas construit une classe pour le menu déroulant. Nous nous sommes contentés d'inclure le menu directement dans la scène en le construisant à partir d'instances de la classe **RMGString**. Il va sans dire que l'étape suivante de programmation devrait faire de ce menu une classe à part entière. C'est l'exemple type de classe réutilisable par bon nombre d'applications quelqu'en soient les domaines d'activités.

Observons la figure 5.1. Elle donne un exemple de menu pour la sélection de fichier FTAM, mais peut être facilement généralisée pour tout menu. Il existe des choix simples dans le menu et des choix plus fins. Un choix fin est illustré par la sélection du fichier FICHIER_1. Les choix fins se distinguent des choix simples par la présence d'un sigle ">" derrière le libellé d'une ligne de menu. Dans le cas du **F-SELECT request**, l'utilisateur se positionne dans un premier temps sur la case correspondante et un déplacement vers la droite de la souris, en maintenant le bouton de droite enfoncé, lui permet d'accéder à une liste des fichiers disponibles. Il peut, dès lors, effectuer un choix sur le fichier qu'il désire. Le processus associé au mécanisme de sélection de fichier débute quand l'utilisateur relache le bouton droit de la souris. Nous avons simulé sur la figure 5.1, la sélection du fichier FICHIER_1.

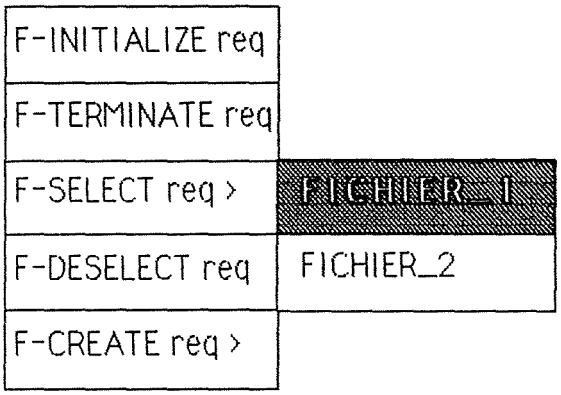


Figure 5.1 : exemple de menu pour l'application FTAM.

4.2. Le choix de la réponse par l'agent répondeur

Nous avons utilisé des boutons de commande (figure 5.2) lorsque l'apprenant à le choix entre deux réponses exclusives (généralement une réponse positive ou négative). Ce choix se justifie d'un point de vue ergonomique mais aussi d'un point de vue pédagogique. En effet, le fait d'adopter une représentation différente pour les choix à réaliser du côté répondeur permet de bien différencier les rôles que joue l'apprenant. Pour l'implémentation des boutons de contrôle, nous avons utilisé la classe **RMGString**.

Figure 5.2 : exemples de boutons de commande.

Ce mode d'interaction est relativement proche de l'utilisateur, surtout s'il est familiarisé avec l'environnement WINDOWS. Le libellé qui doit figurer dans les boutons de contrôle doit être une des réponses possibles à la question posée. Par exemple, si un message lui dit "Acceptez-vous l'association FTAM?", le libellé devra être "OUI" pour un bouton et "NON" pour l'autre et pas "OK" et "CANCEL". Il faut être cohérent au niveau du langage utilisé et cohérent au niveau des termes utilisés.

4.3. L'étude de la simulation en pas-à-pas

Pour nous adapter au rythme de travail de chaque individu, il est nécessaire de laisser celui-ci décider du moment où se poursuivra l'étude de la simulation. Après chaque événement, nous arrêtons la simulation et attendons une action de l'apprenant pour continuer. Il a donc le loisir de lire les messages et d'analyser les effets des actions précédentes. Nous demandons à l'utilisateur de cliquer sur un bouton de la souris. La méthode qui détecte si l'utilisateur a cliqué sur la souris est empruntée à la classe **RMGMouse**.

Comme nous l'avons déjà signalé précédemment, plusieurs niveaux d'utilisation sont envisageables. Dans le cas où un mode expert serait utilisé, les retours d'informations sur les actions qu'entreprend l'utilisateur seraient minimaux. Cet utilisateur n'aurait donc pas à cliquer pour observer la poursuite de la simulation. Ses interventions serviraient à aiguiller la suite des opérations sur base de son choix.

5. Implémentation

Dans cette partie, nous décrivons l'implémentation que nous avons réalisée. Nous caractériserons les différents objets en terme des classes qui les composent et des méthodes qu'ils utilisent.

5.1. Objet régime

L'objet régime nécessite l'affichage dans des cadres imbriqués des différents noms de régime. Il existe déjà une classe de RMG qui permet l'affichage d'un nom dans un cadre. Il s'agit de la classe **RMGString**. Nous définirons donc l'objet régime à partir de cette classe. Les méthodes à utiliser seront l'affichage ou l'effacement du cadre et de la chaîne de caractère qu'il contient. Ces méthodes sont disponibles dans la classe **RMGString**.

5.2. Objets agent initiateur et agent répondeur

Tout comme l'objet régime, les objets agent initiateur et agent répondeur sont symbolisés par un cadre dans lequel nous afficherons la primitive FTAM. Nous utiliserons donc comme ci-dessus la classe **RMGString** et ses méthodes. Nous avons notamment le choix de la position du texte relative au cadre, de la couleur de fond, de la couleur du texte, de la police de caractère.

5.3. Objet fournisseur

L'objet fournisseur doit permettre de matérialiser l'envoi d'un PDU entre les deux entités FTAM paires. Pour ce faire, nous symboliserons le déplacement en traçant une ligne qui relie les deux entités à travers l'objet fournisseur. Nous ferons appel à l'objet **RMGLine** qui implémente toutes les méthodes pour faire cela. Nous avons défini une méthode réalisant l'animation qui simule le passage du PDU avec les méthodes de la classe **RMGLine**.

5.4. Objet message

L'objet message doit permettre l'affichage des messages d'aide, d'accompagnement ou d'erreur. Il s'agit à chaque fois de l'affichage d'une chaîne de caractères dans un cadre. Nous pouvons donc utiliser la classe **RMGString** et ses méthodes. La couleur utilisée et la police de caractère dépendent du type de message comme nous l'avons signalé dans la description des objets du scénario. De plus, lorsqu'il s'agit d'un message d'erreur, nous avons employé un signal sonore (le "bip" système) en plus des méthodes RMG afin d'attirer l'attention de l'utilisateur.

L'emplacement de l'objet message a évolué lui aussi. Nous avons décidé de le mettre non plus au milieu de l'écran comme précédemment, mais au bas de l'écran. Le milieu sert ainsi uniquement à ce qui relève de l'animation. C'est aussi par similitude vis-à-vis d'autres logiciels où l'aide en ligne est donnée en bas d'écran que nous avons décidé de ce changement. L'ergonomie du logiciel est augmentée par ce déplacement de l'objet message.

5.5. Objets Etat-Initiateur et Etat-Répondeur

Les objets Etat-Initiateur et Etat-Répondeur sont des instances de la classe Ftametat que nous avons décrite plus haut.

Dans l'état actuel, seules les primitives **F-INITIALIZE request** et **F-TERMINATE request** correspondant au régime d'association FTAM ont été

implémentées. Nous avons placé en annexe (annexe 5) une copie des programmes implémentant les classes Ftam et Ftametat. Le paragraphe suivant décrit brièvement comment nous envisageons le développement complet du scénario.

6. Implémentation future

Dans le futur, il faudra développer une classe à part entière pour l'objet menu déroulant. Ceci pour répondre aux exigences définies ci-avant. Les objets fichier initiateur et fichier répondeur devront être construits sur le même principe que les objets précédemment définis. Dans le cadre restreint du scénario, l'utilisation de la classe **RMGString** permettra de résoudre les problèmes. Si nous souhaitons dans l'avenir, modéliser tous les types de fichiers et toutes les interactions disponibles sur les fichiers, il peut s'avérer utile de développer l'objet fichier comme une classe à part entière. Il sera alors facile de développer des versions de plus en plus complexes que l'on utilisera au fur et à mesure de leur réalisation. Le développement d'une application dans l'environnement est complexe, c'est pourquoi, nous ne pouvons affirmer catégoriquement que l'implémentation est réalisable sans l'avoir expérimentée. Mais, dans la mesure où les principes d'extension sont semblables aux principes utilisés dans la partie que nous avons implémentée, nous pensons que le développement complet ne devrait pas poser de problème particulier.

Chapitre 6 : Extensions du scénario

1. Introduction

Dans ce chapitre nous envisageons des extensions au scénario proposé dans le chapitre 4. Nous donnons ainsi une liste des applications intéressantes à réaliser dans le domaine de FTAM. Nous parlerons d'applications différentes pour la raison suivante : la réalisation de didacticiels demande la définition d'unités d'enseignement qui correspondent à des objectifs partiels. Il nous semble intéressant de construire plusieurs applications différentes dont le but correspond chaque fois à un objectif partiel. FTAM est un sujet complexe et vaste où il faut prendre garde de ne pas embrouiller l'apprenant avec des détails n'ayant rien à voir avec l'unité d'apprentissage en cours. A titre d'exemple, nous pouvons dire que lorsque l'enseignement visé est celui du comportement des agents FTAM pairs, il n'est pas absolument nécessaire de voir comment les primitives sont communiquées sous forme de FPDU.

2. Application 1 : le transfert de fichiers

Comme nous l'avons signalé lors du chapitre 4, l'étude du transfert de fichier mérite à lui seul une étude approfondie. Dans ce cadre, il serait bon de pouvoir modéliser tout type de fichier permis par la norme. Nous envisageons plusieurs moyens d'apprentissage.

2.1. Les ensembles de contraintes (constraint sets)

Rappelons que la norme permet de spécialiser la définition de fichier virtuel pour qu'elle corresponde plus précisément à une catégorie de fichiers (cfr. chapitre 2). Nous allons décrire un exemple de simulation concrète où l'apprenant s'investit totalement. Dans ce didacticiel, l'apprenant peut construire le fichier par manipulation directe des différents éléments. Il s'agit de créer des noeuds, d'y attacher des unités de données et de relier les noeuds entre eux. Lors de l'assemblage des éléments, l'ordinateur vérifie que l'étudiant entreprend une action valide dans l'ensemble actuel de contraintes. Pour chaque erreur, l'ordinateur donne la cause de l'erreur et éventuellement un moyen d'y remédier. L'apprentissage se réalise donc suite aux erreurs de l'apprenant.

2.2. Les contextes d'accès

Après avoir construit un fichier, l'apprenant peut envisager d'examiner le transfert de celui-ci. Pour ce faire, il définit un contexte d'accès. Rappelons que le contexte d'accès permet de particulariser la liaison entre les deux

associations FTAM au niveau de la structure du fichier et des données (cfr. chapitre 2). Nous pouvons envisager deux types de stratégies d'apprentissage qui se révèlent compatibles.

Une première stratégie pourrait être de simuler le transfert du fichier dans le modèle FTAM. Nous reprenons l'option que nous avons choisie lors du chapitre 4. L'apprenant définit interactivement le fichier, sélectionne le FADU et le contexte d'accès. Il assiste ensuite passivement au transfert simulé du fichier. Nous suggérons de présenter d'une part le fichier de départ ,et d'autre part le fichier d'arrivée tel qu'il peut être reconstruit grâce aux éléments d'information transférés.

Une autre stratégie consisterait à demander à l'apprenant d'indiquer quel élément du fichier fera l'objet du transfert suivant. L'ordinateur vérifie si la réponse est correcte. Si une erreur a été commise, il signale la cause de l'erreur, sinon, il effectue le transfert. L'étudiant peut ainsi vérifier qu'il maîtrise les contextes d'accès. On pourrait également envisager de demander à l'apprenant de reconstruire le fichier en fonction des informations disponibles dans le contexte d'accès spécifié.

Pour l'apprentissage complet, on envisage dans un premier temps le mode de simulation qui permet à l'étudiant de voir comment les choses se passent puis dans un deuxième temps de vérifier ses connaissances en indiquant lui-même les éléments à transférer.

Un autre problème se dégage. Le but de l'apprentissage que nous décrivons ici concerne le transfert de fichier en relation avec les contextes d'accès. La question que nous sommes en droit de nous poser est la suivante : faut-il bâtir cette application de simulation sur la base du didacticiel FTAM que nous avons développé? Si la réponse est oui, cela signifie que pour se familiariser avec les contextes d'accès, l'apprenant doit initialiser une association FTAM, sélectionner un fichier, l'ouvrir. Ne pourrait-on pas envisager que l'application lui donne directement un fichier sur lequel travailler (ou une proposition de fichiers)?

3. Application 2 : les paramètres FTAM

Les paramètres véhiculés dans les primitives FTAM jouent un rôle fondamental pour comprendre le fonctionnement détaillé du modèle. Il serait intéressant de montrer comment les renseignements sont fournis et obtenus à travers les primitives. Nous pourrions proposer de modifier notre application de manière à démontrer l'utilisation des paramètres plutôt que le

fonctionnement des machines de protocole de fichier. Pour chaque primitive sélectionnée par l'apprenant un sous-menu apparaît où il peut modifier les paramètres de la primitive. La simulation lui permet alors de vérifier les conséquences de ses modifications. Une zone est réservée sur l'écran de manière à permettre l'affichage des valeurs de la primitive en cours. Le déroulement des primitives symbolisé de manière identique à notre application permet à l'apprenant de situer le contexte de la simulation et d'établir le lien entre les valeurs des paramètres et la situation actuelle (figure 6.1).

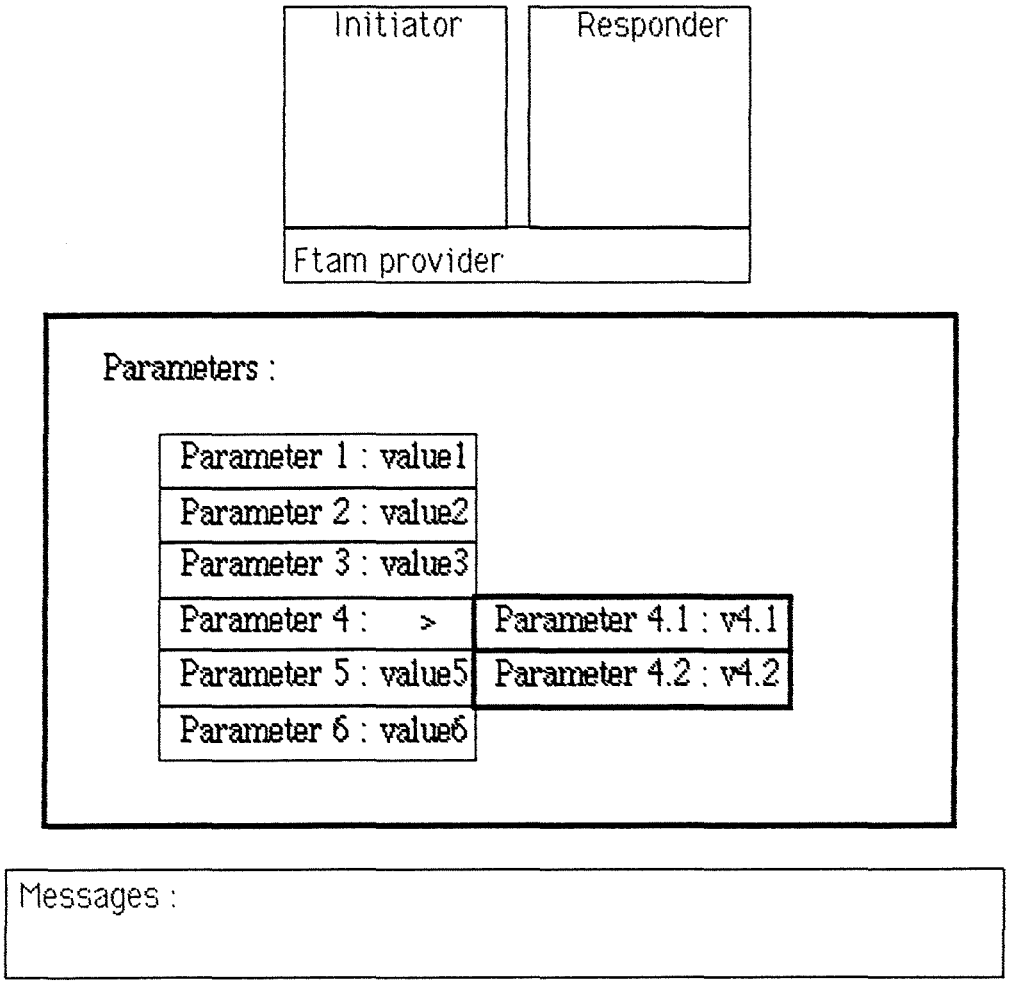


Figure 6.1 : la représentation des paramètres dans le contexte de l'application FTAM.

4. Application 3 : le recouvrement d'erreurs

Il est intéressant de disposer d'une application permettant d'envisager l'utilisation des primitives F-CHECK, F-RESTART, F-RECOVER, F-CANCEL. L'apprentissage du fonctionnement de ces primitives est considérablement plus complexe étant donné la charge informationnelle

élevée. En effet, pour simuler le comportement de ces primitives, il faut tenir compte de l'historique des transferts. La quantité d'information à accumuler est importante. La récupération peut se produire à partir de n'importe quel endroit étant donné qu'une erreur peut survenir à tout moment.

On pourrait envisager l'utilisation de la couleur pour marquer la dimension temps dans l'application, mais il faut se restreindre à un nombre peu élevé d'éléments pour ne pas multiplier les couleurs. Ce qui est sûr, c'est que ce problème est suffisamment compliqué pour mériter une application à lui seul. Il se révèle fondamental dans le modèle ISO puisqu'il se pose au niveau de la couche application (RTSE, niveau 7) et de la couche session (niveau 5), mais également dans la procédure HDLC au niveau 2. Sur la figure 6.2 nous avons représenté les données à transférer, les données déjà transférées et les données en cours de transfert.

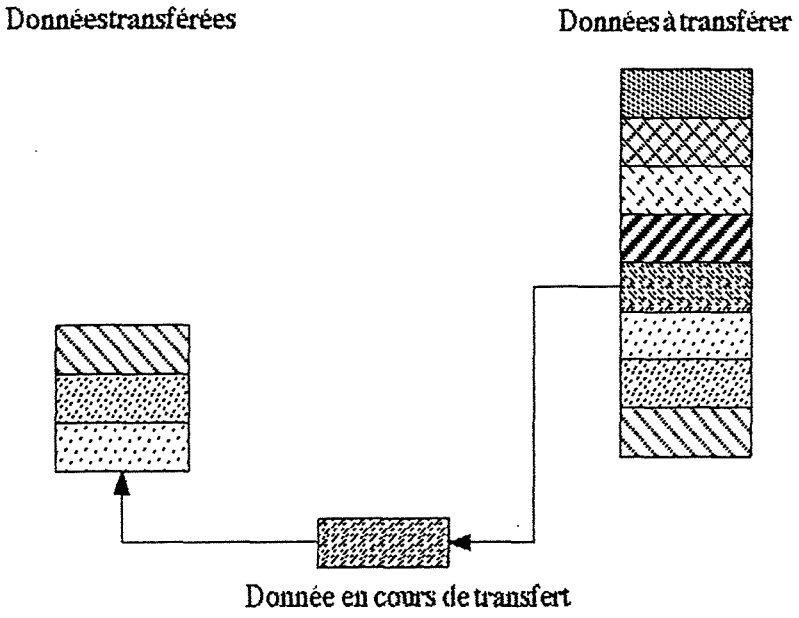


Figure 6.2 : schéma illustrant un transfert de données en cours avec mémorisation des données transférées.

La figure suivante (figure 6.3) représente la sélection de l'unité de données à partir de laquelle les transferts vont reprendre (suite à la simulation d'une erreur par exemple).

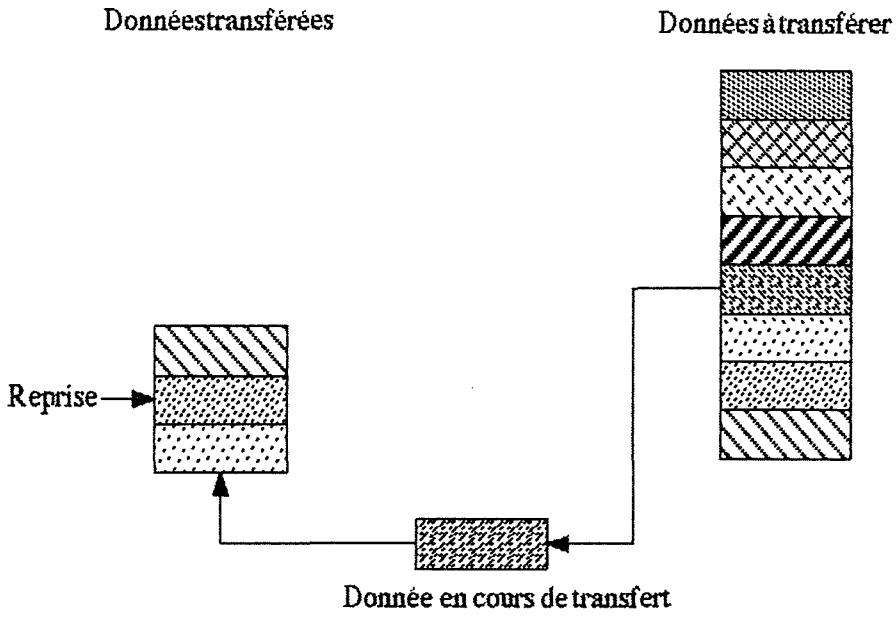


Figure 6.3 : choix de la donnée à partir de laquelle le transfert doit reprendre.

La figure suivante représente l'état du transfert lors de la reprise (figure 6.4).

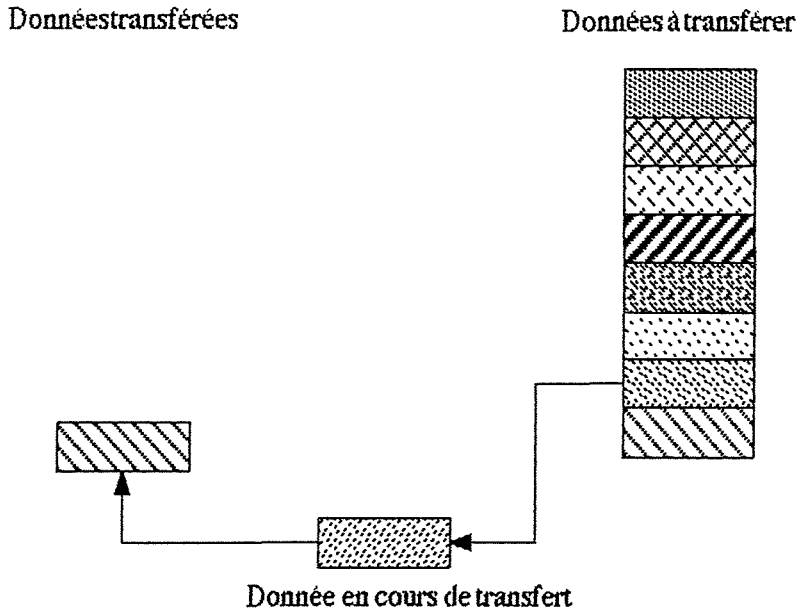


Figure 6.4 : état du transfert de données après la reprise.

5. Application 4 : les couches inférieures du modèle ISO lors d'une application FTAM

Jusqu'à présent, nous nous sommes toujours placés du point de vue de l'agent FTAM. Nous allons maintenant considérer l'entité FTAM. Pour exécuter les demandes venant de son agent, l'entité construit des unités de protocole de données, elle utilise les services des couches inférieures (ACSE et Présentation). Cette application est intéressante car elle ne vise plus à montrer le principe d'utilisation de FTAM mais bien le principe de son fonctionnement.

Nous suggérons d'illustrer en détail le traitement de chaque primitive FTAM. Il faut explicitement montrer qu'en fonction de la primitive, l'entité FTAM entreprend un certain nombre d'actions pouvant aller d'un simple appel de service au niveau inférieur (par exemple pour le F-DATA request), à la construction et l'envoi d'un FPDU (via le service de la couche inférieure). De même, il faudra montrer comment l'entité traite les FPDU qu'elle reçoit. La liaison entre les FPDU et les primitives doit apparaître clairement ainsi que l'utilisation des services de la couche inférieure par les entités FTAM. Une dimension souvent oubliée et pourtant fondamentale consisterait à montrer la collaboration entre les entités FTAM paires de manière à exécuter le service demandé.

Ces principes sont fondamentaux dans le modèle ISO et méritent d'être considérés dans une optique beaucoup plus large qui reprendrait l'ensemble des couches. Pour l'enseignement, il est important de tenir compte des éléments suivants.

Tout d'abord l'enseignement doit se faire de manière claire, précise et concise. Il n'est pas possible dans les premiers temps de l'apprentissage d'envisager le fonctionnement global du modèle. Nous suggérons donc de construire une application par couche qui reçoive les primitives de la couche supérieure et utilise les primitives de la couche inférieure. L'apprentissage est réalisé en profondeur avec les détails de fonctionnement et tous les messages d'aide en accompagnement.

Dans un deuxième temps, on peut envisager l'intégration des applications en une seule. Celle-ci permettrait de visualiser les enchaînements des primitives à travers les couches du modèle. Il est évident que le but poursuivi par cette unité d'apprentissage ne concerne plus l'enseignement approfondi des différentes couches mais bien l'enchaînement de ces primitives dans le

modèle. Les messages d'aide et d'accompagnement ne devraient plus être détaillés au niveau des couches (surcharge trop lourde pour des choses déjà connues) mais bien concerner l'enchaînement des primitives. Une architecture orientée objets devrait pouvoir permettre ce genre de chose très simplement pour autant qu'une analyse détaillée de ce que l'on veut faire soit réalisée avant la conception d'une couche. Cette étape d'analyse pendant laquelle toute programmation est exclue devrait être assez longue et poser par ce fait des problèmes matériels importants par rapport à l'avancement des autres équipes COLOS, il faut donc voir si le jeu en vaut la chandelle.

Chapitre 7 : Critique

Pour réaliser notre travail, nous avons dû acquérir de nombreuses connaissances dans de multiples domaines. Aussi, nous n'avons pu, dans la mesure de nos possibilités, qu'effleurer les différents sujets. En aucun cas, nous ne pouvons prétendre que ce travail constitue une référence en la matière. Nous dirons plus modestement qu'il contribue à l'état de l'art.

Dans le cadre de notre travail, le scénario FTAM que nous avons élaboré ne s'appuie que sur une partie des possibilités offertes par le monde FTAM. Mais comme nous avons déjà eu le loisir de vous le dire, l'enseignement assisté par ordinateur est une discipline où il faut savoir cerner le problème et retirer l'essentiel d'une matière avant de passer au stade de la conception d'un didacticiel.

L'étape la plus difficile à juger est sans nul doute l'étape de spécification du scénario. Nous ne pouvons prétendre avoir réussi à spécifier clairement et sans ambiguïté le déroulement de la simulation. Nous pensons cependant que notre approche est intéressante. Elle manipule dès le départ des objets et se prête donc assez naturellement à une implémentation dans un environnement orienté objets. Le langage utilisé nous paraît suffisamment rigoureux pour permettre la réalisation concrète du scénario. Néanmoins, il ne faut pas perdre de vue que nous nous sommes occupés de l'étape de spécification et de l'étape d'implémentation ; aussi, certains éléments qui dans notre esprit étaient évidents n'ont pas été exprimés. Nous avons souligné le fait que pour se faire comprendre, il vaut mieux réaliser l'application par prototypage. Ainsi chaque petite étape réalisée permet la discussion. Nous avons également souligné le fait que notre langage de spécification pourrait faire l'objet d'un outil de prototypage. Cette voie reste ouverte.

Nous pouvons regretter de n'avoir pas eu suffisamment de temps matériel pour parvenir à l'implémentation complète de notre scénario. Malgré cela, nous pensons que la partie de la simulation que nous avons développée est intéressante. Elle utilise la plupart des objets qui ont été décrits pour le scénario. Cela laisse à nos successeurs une tâche plus facile à porter s'ils envisagent de poursuivre notre projet. Les objets et les méthodes ont été largement décrits et le passage de la maquette papier à la médiatisation devrait se faire sans problèmes majeurs.

L'implémentation dans l'environnement RMG est rendue plus délicate par le fait que la bibliothèque de classes est très volumineuse et donc nécessite un

long apprentissage. Néanmoins, en nous limitant à quelques classes de base, nous avons pu réaliser ce que nous souhaitions.

Nous nous sommes demandés si un enseignement assisté par ordinateur tournant sur PC n'aurait pas été préférable au développement de didacticiels sur RMG. D'une part, la médiatisation du produit serait plus efficace. Un PC est autonome, il peut être déplacé facilement d'une classe à une autre, d'un bâtiment à un autre. De plus, les PC eux-mêmes sont très répandus, il suffit dès lors de se procurer l'application. La diffusion d'un didacticiel réalisé grâce à RMG est bien moins évidente. Les endroits où peut se dérouler l'apprentissage des applications développées sont limités. L'avantage de RMG est sa rapidité d'exécution et son graphisme de haute qualité, mais en a-t-on toujours besoin pour le sujet qui nous préoccupe?

Nous regrettons l'absence (suite à un heureux événement) de l'assistante du projet COLOS à partir du mois de mars. En effet, la réalisation d'un projet d'EAO est avant tout une question de collaboration. Le travail entamé auparavant était plus constructif suite aux remarques provenant de plusieurs personnes.

Nous regrettons enfin de ne pas avoir eu le temps d'évaluer notre produit grâce à un test sur la population cible. Nous pensons qu'une telle épreuve permettrait d'enrichir considérablement notre expérience. C'est lorsqu'il sera véritablement confronté à la pratique que le projet pourra débuter efficacement.

Conclusion

L'enseignement assisté par ordinateur constitue un domaine complexe puisqu'il nécessite la connaissance de notions pédagogiques, ergonomiques, informatiques, sans oublier les connaissances relatives à la matière à enseigner. Nous espérons avoir accumulé suffisamment d'éléments dans ce travail pour permettre à l'équipe de Namur de progresser vers les objectifs qu'elle s'est fixés.

La participation au projet COLOS nous a permis de mettre en oeuvre une démarche de conception de logiciels. Nous avons tâché d'être méthodique, de ne rien laisser au hasard. Le développement de didacticiel n'est pas une chose simple, nous espérons que notre méthode est acceptable et que les membres de l'équipe en tireront quelque chose d'utile.

En ce qui concerne l'avenir, l'environnement orienté objets ne permettra l'intégration de l'ensemble du programme d'enseignement que si une étude très approfondie est menée avant toute réalisation concrète. Une telle étude nous paraît trop complexe pour aboutir avant long terme. Nous pensons qu'il vaut mieux s'orienter vers la conception d'unités d'enseignement plus petites. En travaillant comme cela, l'équipe pourra produire plus rapidement des produits finis de bonne qualité et réaliser leur évaluation sur une population cible.

Nous avons fait l'expérience de spécifier un scénario de manière très proche de l'implémentation dans un environnement orienté objets. C'est pourquoi nous pensons qu'il serait profitable de s'intéresser aux outils de prototypage qui pourraient accélérer la réalisation du projet (l'outil de prototypage par manipulation graphique que l'on trouve dans RMG étant loin de donner satisfaction).

Nous n'avons malheureusement pas eu le temps matériel de réaliser toutes les étapes de développement de notre projet. Dans un avenir proche l'équipe de Namur devra tester le résultat de ces recherches auprès des étudiants. Cela permettra de se faire une idée plus précise de la qualité pédagogique des didacticiels développés.

Bibliographie

BESN88 : R. BESNAINOU, C. MULLER & Ch. THOUIN, CONCEVOIR ET UTILISER UN DIDACTICIEL, Les Editions d'Organisation, Paris 1988.

BONN90 : R. BONNAIRE et H. PERRIN, DEVELOPPEMENT DE SIMULATIONS PEDAGOGIQUES DANS UN ENVIRONNEMENT ORIENTE OBJETS, Communication interne, Université Pierre et Marie Curie, Paris 1990.

DONN84 : J. DONNAY & M. ROMAINVILLE, GRILLE D'ANALYSE DE DIDACTICIELS, "Réseau OSE des FUNDP", Namur 1984.

DUBO90 : E. DUBOIS, METHODOLOGIE DE DEVELOPPEMENT DE LOGICIELS, cours présenté durant l'année scolaire 1990/1991, FUNDP.

FAZA89 : ZVONKO FAZARINC, RMG USER'S FIRST AID KIT, Hewlett-Packard, Palo Alto, California, 1989.

HAIN86 : J-L HAINAUT,CONCEPTION ASSISTEE DES APPLICATIONS INFORMATIQUES. 2.CONCEPTION DE LA BASE DE DONNEES, Masson 1986.

HENS88 : J. HENSHALL & S. SHAW, OSI EXPLAINED : END-TO-END COMPUTER COMMUNICATION STANDARDS, Ellis Horwood Chichester, 1988.

ISO8571 ISO/DIS 8571 INFORMATION PROCESSING SYSTEMS. OPEN SYSTEMS INTERCONNECTION. FILE TRANSFER, ACCESS AND MANAGEMENT, 1986.

ISO8824 : ISO 8824 INFORMATION PROCESSING SYSTEMS. OPEN SYSTEMS INTERCONNECTION. ABSTRACT SYNTAX NOTATION 1, 1986

ISO8825 : ISO 8825 INFORMATION PROCESSING SYSTEMS. OPEN SYSTEMS INTERCONNECTION. BASIC ENCODING RULES FOR ABSTRACT SYNTAX NOTATION ONE (ASN.1) 1986

MARI91 : LUC MARIAUX, Communication interne, Ecole Centrale de Lyon, 1991

TANN81 : ANDREW S. TANNENBAUM, COMPUTER NETWORKS, Prentice-Hall Englewood Cliffs(N.J.), 1981.

TOLH88 : MARK TOLHURST, OPEN SYSTEMS INTERCONNECTION, Macmillan Basingstoke, 1988.

STAL87 : W. PH. STALLINGS, DATA AND COMPUTER COMMUNICATION, Mac Millan, 1987.

Annexe 1 : primitives FTAM

Dans cette annexe, nous reprenons la liste des primitives de service FTAM. Pour chacune d'elle, la liste comprend le rôle et le type de service qu'elle joue.

Une seconde liste reprend les paramètres associés à chaque primitive.

COUCHE APPLICATION I.S.O. 8571/3	-1-	F.T.A.M. File Transfer, Access and Management

Noms des primitives de service	D	I	R	C	Rôle des primitives
F_INITIALIZE	X	X	X	X	L'initiateur crée un régime FTAM en reliant les utilisateurs aux fournisseurs du service de fichiers dans une association de niveau application
F_TERMINATE	X	X	X	X	L'initiateur détruit le régime FTAM en déliant l'association existant entre les utilisateurs et fournisseurs du service fichiers
F_U_ABORT	X	X			L'utilisateur du service FTAM détruit inconditionnellement l'association FTAM
F_P_ABORT		X			Le fournisseur du service FTAM détruit inconditionnellement l'association FTAM
F_SELECT	X	X	X	X	L'initiateur sélectionne un fichier existant. Il y a alors disponibilté de ce fichier selon certaines priorités.
F_DESELECT	X	X	X	X	L'initiateur désélectionne un fichier existant. Le fichier est dès lors indisponible.
F_CREATE	X	X	X	X	L'initiateur crée un fichier et le sélectionne.
F_DELETE	X	X	X	X	L'initiateur abandonne une sélection existante, et détruit le fichier existant
F_READ_ATTRIB	X	X	X	X	L'initiateur consulte les attributs du fichier sélectionné
F_CHANGE_ATTRIB	X	X	X	X	L'initiateur modifie les attributs du fichier sélectionné
F_OPEN	X	X	X	X	L'initiateur établit le contexte de présentation, ainsi que les contrôles de concurrence et de confirmation du transfert des données
F_CLOSE	X	X	X	X	L'initiateur libère le contexte établi par le service d'ouverture du fichier

D = Demande (request)
I = Indication
R = Response (Réponse)
C= Confirmation

Noms des primitives de service	D	I	R	C	Rôle des primitives
F_BEGIN_GROUP	X	X	X	X	L'initiateur indique le début d'un groupement de primitives
F_END_GROUP	X	X	X	X	L'initiateur indique la fin d'un groupement de primitives
F_RECOVER	X	X	X	X	L'initiateur recrée le régime d'ouverture après une rupture survenue dans le régime d'ouverture
F_LOCATE	X	X	X	X	L'initiateur spécifie l'identité d'une FADU qui doit être localisée par le répondeur
F_ERASE	X	X	X	X	L'initiateur efface une FADU du fichier
F_READ	X	X			L'initiateur demande au répondeur de lire certaines données du fichier et de les lui transmettre
F_WRITE	X	X			L'initiateur demande au répondeur d'écrire les données qui seront envoyées, dans le fichier
F_DATA	X	X			Envoi de données à lire ou à écrire, par l'expéditeur
F_END_DATA	X	X			Fin d'envoi d'un bloc de données par l'expéditeur
F_TRANSFER_END	X	X	X	X	L'initiateur confirme que le transfert des données est terminé
F_CANCEL	X	X	X	X	Annulation d'une activité de transfert de données
F_CHECK	X	X	X	X	Marquage dans le flux de données pour préparer une éventuelle reprise sur erreurs
F_RESTART	X	X	X	X	Interruption du transfert en cours, et négociation du point de reprise (marque "checkpoint")

D = Demande (request)
I = Indication
R = Response (Réponse)
C= Confirmation

COUCHE APPLICATION I.S.O. 8571/3	F.T.A.M. File Transfer, Access and Management

Noms des primitives de service	Paramètres des primitives	D	I	R	C
F_INITIALIZE	résultat d'état			X	X
	résultat d'action			X	X
	titre de l'application appelée	X	X		
	titre de l'application appelante	X	X		
	titre de l'application répondante			X	X
	adresse de présentation appelée	X	X		
	adresse de présentation appelante	X	X		
	adresse de présentation répondante			X	X
	gestion du contexte de présentation	X	X	X	X
	niveau de service	X	X		
	classe de service	X	X		
	unités fonctionnelles	X	X	X	X
	groupes d'attributs	X	X	X	X
	disponibilité de retour en arrière	X	X	X	X
	qualité de service	X	X	X	X
	liste du contenu des types	X	X	X	X
	identité de l'initiateur	X	X		
	somme (compte)	X	X		
	mot de passe du "filestore" (ensemble de fichiers)	X	X		
	diagnostique			X	X
	fenêtre de point de contrôle	X	X	X	X
F_TERMINATE	frais (charging)			X	X
F_U_ABORT	résultat d'action	X	X		
	diagnostique	X	X		
F_P_ABORT	résultat d'action		X		
	diagnostique		X		
F_SELECT	résultat d'état			X	X
	résultat d'action			X	X
	attributs	X	X	X	X
	accès requis	X	X		
	mots de passe d'accès	X	X		
	contrôle de concurrence	X	X		
	contrôle des engagements	X	X		
	somme (compte)	X	X		
	diagnostique			X	X
F_DESELECT	résultat d'action			X	X
	frais (charging)			X	X
	contrôle des engagements	X	X	X	X
	diagnostique			X	X

D = Demande (request)
I = Indication
R = Response (Réponse)

C= Confirmation

COUCHE APPLICATION I.S.O. 8571/3	F.T.A.M. File Transfer, Access and Management	-2-

Noms des primitives de service	Paramètres des primitives	D	I	R	C
F_CREATE	résultat d'état			X	X
	résultat d'action			X	X
	excédent (override)	X	X		
	attributs initiaux	X	X	X	X
	accès requis	X	X		
	mots de passe d'accès	X	X		
	mot de passe de création	X	X		
	mot de passe de suppression	X	X		
	contrôle de concurrence	X	X		
	contrôle des engagements	X	X		
	somme (compte)	X	X		
	diagnostique			X	X
F_DELETE	résultat d'action			X	X
	mot de passe de suppression	X	X		
	frais (charging)			X	X
	diagnostique			X	X
F_READ_ATTRIB	résultat d'action			X	X
	noms des attributs	X	X		
	attributs			X	X
	diagnostique			X	X
F_CHANGE_ATTRIB	résultat d'action			X	X
	attributs	X	X	X	X
	diagnostique			X	X
F_OPEN	résultat d'état			X	X
	résultat d'action			X	X
	mode de traitement	X	X		
	type de contenu	X	X	X	X
	contrôle de concurrence	X	X	X	X
	contrôle d'engagement	X	X	X	X
	diagnostique			X	X
	identificateur d'activité	X	X		
	mode de recouvrement	X	X	X	X
F_CLOSE	résultat d'action			X	X
	contrôle d'engagement	X	X	X	X
	diagnostique			X	X
F_BEGIN_GROUP	seuil	X	X		

D = Demande (request)
I = Indication
R = Response (Réponse)
C= Confirmation

Noms des primitives de service	Paramètres des primitives	D	I	R	C
F_END_GROUP	* sans paramètre				
F_RECOVER	résultat d'état			X	X
	résultat d'action			X	X
	identificateur d'activité	X	X		
	nombre de transfert de bloc	X	X		
	accès requis	X	X		
	mots de passe d'accès	X	X		
	type des contenus			X	X
	point de recouvrement	X	X	X	X
	diagnostique			X	X
F_LOCATE	résultat d'action			X	X
	identité de l'unité de donnée d'accès au fichier	X	X	X	X
	diagnostique			X	X
F_ERASE	résultat d'action			X	X
	identité de l'unité de donnée d'accès au fichier	X	X		
	diagnostique			X	X
F_READ	spécification de transfert de données en bloc	X	X		
F_WRITE	spécification de transfert de données en bloc	X	X		
F_DATA	valeur de donnée	X	X		
F_END_DATA	résultat d'action	X	X		
	diagnostique				
F_TRANSFER_END	résultat d'action			X	X
	contrôle d'engagement	X	X	X	X
	diagnostique			X	X
F_CANCEL	résultat d'action	X	X	X	X
	diagnostique	X	X	X	X
F_CHECK	identificateur de point de contrôle	X	X	X	X
F_RESTART	identificateur de point de contrôle	X	X	X	X

D = Demande (request)
I = Indication
R = Response (Réponse)
C= Confirmation

Annexe 2 : grille d'analyse du didacticiel

Dans cette annexe, nous avons réalisé l'analyse de notre didacticiel sur base de la grille proposée par Marc Romainville et Jean Donnay [DONN84]. Tous les points n'ont pas été repris. Nous avons suivi les conseils des auteurs et nous nous sommes contenté de reprendre les points pertinents et/ou intéressants.

Caractéristiques générales

Public cible : les étudiants de deuxième licence en informatique qui suivent le cours de télécommunications matières approfondies sont concernés par ce didacticiel.

Sujet : il s'agit de l'étude du fonctionnement du modèle FTAM du point de vue de l'échange de primitives entre agents FTAM pairs.

Objectif : l'étudiant sera capable d'enchaîner correctement les primitives pour réaliser un but fixé. Ce but peut concerner un transfert de fichier ou la simple lecture d'un attribut par exemple.

Prérequis : les connaissances générales concernant les principes du modèle ISO, ainsi que le cours magistral sur FTAM sont indispensables pour l'utilisation efficace de ce didacticiel.

Durée moyenne d'exécution : la durée moyenne d'exécution est indéfinissable étant donné le caractère de simulation du logiciel.

Mode d'utilisation : une utilisation en "autoapprentissage" est envisageable. Le logiciel permet également au professeur d'illustrer son cours.

Caractéristiques techniques

Ordinateur : HP 9000/360.
Système d'exploitation : UNIX.
Périphériques d'entrée : souris, clavier.
Périphérique de sortie : écran.
Langage de programmation : Objective-C.

Analyse de l'unité de dialogue

L'unité de dialogue est définie comme la plus petite unité pédagogique significative. Elle se décompose en une unité d'interaction (envoi d'information, envoi d'une sollicitation à l'apprenant, réception d'un message , analyse de la réponse) et en une unité de décision (envoi d'un feedback, branchement vers une autre unité d'interaction).

Unité d'interaction

Envoi d'information à l'apprenant

L'information transmise est-elle pertinente par rapport aux objectifs ?

Dans tous les cas, les informations décrivent directement ce qui se passe au niveau de la simulation. La pertinence de certaines informations s'estompe au fur et à mesure de la connaissance du didacticiel. A partir d'un certain niveau, il faudrait offrir à l'utilisateur la possibilité de court-circuiter certains messages.

La structure des phrases favorise-t-elle la lisibilité ?

Généralement, une seule phrase sert de commentaire à une situation donnée. Cela provient de la découpe très fine de la simulation. Nous avons veillé à construire une phrase complète (sujet, verbe) à chaque fois.

Le programme présente-t-il l'information d'une manière telle qu'il serait difficile de réaliser le même type de présentation avec un autre support pédagogique ?

Dans la mesure ou il s'agit de simuler un modèle complexe avec des animations, nous pensons que la réponse est oui.

Le programme présente-t-il l'information avant la question ?

Nous avons veillé à poser nos questions au bas de l'écran, de sorte que l'apprenant regarde d'abord le centre de l'écran et y détaille les informations relatives à la simulation. Les objets interactifs servant à la réponse de l'apprenant (menu déroulant, boutons de contrôle) sont présentés uniquement lorsque la réponse est nécessaire à la poursuite de la simulation.

Le programme revient-il sur les notions importantes pour faciliter la mémorisation ?

Dans le cadre d'une simulation, il est difficile de réaliser ce genre de chose si ce n'est à l'initiative de l'apprenant lui-même. Nous reconnaissons que

sur ce point notre didacticiel est faible. L'apprenant n'a peut être pas le discernement nécessaire pour revenir lui-même sur les notions importantes.

Le graphisme focalise-t-il l'attention sur les éléments essentiels ?

Nous avons représenté tous les objets de manière graphique. Nous n'avons pas surchargé le graphisme des différents objets étant donné le nombre élevé de ceux-ci. Le nombre des objets est une faiblesse qui peut distraire l'attention de l'apprenant. Néanmoins, le clignotement du bord de l'objet avant qu'un changement se passe, semble être une bonne solution pour focaliser l'attention.

Le temps de présentation du graphisme est-il suffisant ?

Dans la mesure où l'apprenant déclenche lui-même la poursuite de la simulation, le temps de présentation est adéquat pour tout utilisateur.

Le graphisme est-il bien relié au texte ?

Le texte se rapporte toujours au déroulement de la simulation et donc à l'évolution du graphisme.

Le graphisme est-il correctement situé sur l'écran ?

Nous pourrions regretter le fait que le texte et le graphisme sont séparés sur l'écran. Néanmoins, la scène présentée, bien qu'évolutive, est constamment affichée, les messages sont toujours affichés au même endroit. Nous pensons que cette cohérence spatiale permet de pallier à la séparation du texte et du graphisme.

Le recours à la représentation graphique facilite-t-il réellement la compréhension des données ?

Chaque objet comporte un libellé qui permet son identification immédiate. Dans notre logiciel, c'est surtout la localisation spatiale des objets qui facilite la compréhension. La symbolique associée au comportement des objets nécessite un commentaire textuel.

L'image est-elle ambiguë ?

Le clignotement des attributs lors d'une lecture ou d'une écriture est bien évidemment une ambiguïté. Nous la levons grâce à un commentaire approprié. Il ne faut pas perdre de vue que c'est l'apprenant qui a demandé une lecture ou une modification des attributs. De par ce fait, il nous semble que celui-ci est à même de différencier les deux situations.

La couleur est-elle utilisée pour porter l'attention sur ce qui est important ?

RMG fournit la possibilité d'utiliser jusqu'à 256 couleurs simultanément. Nous n'avons pas trop porter attention à l'usage des couleurs qui nous semble relever des goûts personnels de chaque individu. Nous avons limité le nombre de couleurs utilisées et nous sommes restés dans des teintes proches les unes des autres pour des objets proches les uns des autres. Pour marquer une différence, nous avons utilisé une couleur différente. Comme nous n'étions pas expert, nous avons prévu que la modification des couleurs dans le programme puisse se faire facilement.

Les mouvements sur l'écran suivent-ils le mouvement habituel des yeux pendant la lecture ?

La réponse est "non" et provient du fait que la simulation concerne des objets dont la localisation à l'écran est fixe. Le trajet des yeux relève d'un balayage en "U". Ce qui nous semble être une innovation un peu malheureuse. Néanmoins la disposition adoptée est celle communément admise dans les ouvrages traitant du modèle ISO.

Pour des notions nouvelles, les illustrations complexes sont-elles subdivisées ?

Par exemple, la représentation d'un fichier est décomposée en trois parties (nom, attributs, structure des données). Nous avons de plus veillé à ne montrer que les informations accessibles à un moment donné.

Les informations complémentaires sont-elles disposées au bas de l'écran ?

Toutes les informations qui commentent l'état de la simulation sont disposées au bas de l'écran.

Envoi d'une sollicitation

Les questions sont posées par l'affichage des objets interactifs au moment opportun. Il s'agit de l'affichage du menu déroulant lors de la sélection d'une primitive ou de l'affichage des boutons de contrôle lorsque l'apprenant doit choisir s'il accepte ou refuse la demande de l'agent initiateur FTAM. De plus, un commentaire en bas de page dit à l'utilisateur ce qu'il doit faire.

Réception du message de l'apprenant

Le didacticiel utilise-t-il les touches les plus simples ?

La sélection dans les objets interactifs se réalise au moyen de la souris.

Avant de répondre, l'apprenant peut-il toujours formuler une requête? Par exemple peut-il faire appel à des ressources complémentaires ou à des informations présentées antérieurement ?

Nous n'avons pas prévu ce genre de chose mais RMG fournit un moyen facile d'atteindre ce but. Il suffit d'associer un menu principal à l'application. Ce menu peut être appelé lorsque l'utilisateur a le contrôle de la souris. Le fait que l'utilisateur possède le contrôle de la souris permet d'envisager qu'il puisse intervenir à différents endroits sur l'écran. Nous pensons en particulier à la possibilité d'obtenir une aide complémentaire en visualisant les primitives qui permettent de passer de l'état courant à un état suivant de la machine de protocole de fichier. Pour ce faire l'utilisateur pourrait se positionner sur l'arc joignant l'état courant à l'état suivant qu'il désire atteindre.

Unité de décision

Feedback

Existe-t-il un feedback positif ?

Nous pouvons considérer le fait que le déroulement de la simulation se poursuit comme un feedback positif pour l'apprenant.

L'évaluation de la réponse est-elle neutre en cas d'erreur ?

Nous pouvons difficilement nous prononcer sur ce point fondamental étant donné le manque d'expérience. Nous attirons l'attention de l'apprenant par l'affichage d'un message d'erreur dans une autre couleur accompagné d'un signal sonore. Le message commente la raison de l'erreur et demande de répondre à nouveau à la question.

Branchement

Les possibilités de branchements sont laissées à l'initiative de l'apprenant dans la limite des contraintes imposées par le modèle. Par exemple, seules les primitives valides pour l'état courant de la machine de protocole de fichier permettrons un branchement. Etant donné les contraintes du modèle, un retour en arrière n'est envisageable que dans les limites définies.

Analyse de l'environnement du didacticiel

Type de didacticiel

Nous avons conçu un logiciel de simulation. L'ordinateur sert de simulateur. Il place l'apprenant devant une situation, une tâche ou un environnement "réels". Celui-ci est alors invité à faire varier des paramètres.

Objectif

L'objectif est-il explicité ?

Notre objectif est l'apprentissage du fonctionnement du modèle FTAM dans son ensemble du point de vue des agents FTAM pairs. L'élève sera capable d'utiliser le modèle pour accomplir une tâche relative à la gestion, à l'accès ou au transfert de fichiers.

L'objectif s'inscrit-il dans un programme d'étude ?

L'objectif pédagogique visé par la simulation s'inscrit dans le cadre du cours de télécommunications matières approfondies de deuxième licence informatique du professeur Ph. Van Bastelaer.

Le type de didacticiel convient-il à l'objectif ?

L'étude du modèle ISO et plus particulièrement de FTAM en ce qui nous concerne se prête particulièrement bien à une simulation. Nous devons également signaler que dans la mesure où les applications doivent être développées dans l'environnement RMG, il est impensable de concevoir d'autres types de didacticiels.

Insertion du didacticiel

Quelle place prend-il par rapport au professeur ?

En fonction de ce que nous avons pu observer tout au long du cours de seconde licence, nous pouvons affirmer que les concepts relatifs au modèle ISO ne sont pas évidents à saisir. Souvent les mêmes questions sont posées par les étudiants. Nous pensons qu'en manipulant lui-même, les concepts dans le cadre d'une simulation, l'apprenant sera plus à même de saisir la signification de ceux-ci.

Documentation

Ce mémoire constitue une source de documentation relative au didacticiel conçu. Mis à part l'apprentissage des principes d'utilisation de l'environnement RMG, il nous semble que le programme réalisé se suffit à lui-même. Concevoir une documentation reviendrait à éditer une brochure de présentation du didacticiel.

Contrôle du programme

L'étudiant peut-il arrêter le déroulement du didacticiel et en sortir quand il veut ?

L'environnement RMG permet d'attacher un menu à une application. Ce menu peut être appelé lorsque l'utilisateur possède le contrôle de la souris ce

qui est pratiquement toujours le cas. Il suffit alors de prévoir dans ce menu la possibilité de quitter l'application.

L'étudiant peut-il faire appel au besoin à une fonction d'aide?

Le même mécanisme permet de définir un appel à une fonction d'aide. Notre didacticiel est conçu de telle manière que les messages d'assistance sont présents en permanence au bas de l'écran.

Peut-il s'apesantir sur un chapitre plus difficile ? Peut-il contrôler la vitesse de présentation ?

L'apprenant maîtrise complètement la vitesse de son apprentissage. Il peut utiliser le modèle comme il le souhaite et donc répéter les actions qu'il souhaite.

Peut-il revenir directement là où il a quitté le cours lors d'une séance précédente ?

Cette question constitue un véritable problème dans notre cas. En effet, si l'utilisateur est intéressé à examiner le transfert de fichier, il devra entrer dans les différents régimes FTAM et donc utiliser un certain nombre de primitives avant de pouvoir commencer l'étude de la partie qui l'intéresse. Il serait intéressant de pouvoir se placer dans l'état qui intéresse l'apprenant. Malheureusement, étant donné le caractère de simulation de l'application, il semble difficile de définir tous les paramètres qui placeraient directement le système dans l'état désiré par l'utilisateur.

Peut-il contrôler la quantité d'information ?

Nous proposons qu'une version future du logiciel autorise l'exécution en mode expert. Dans ce mode, l'apparition des messages d'accompagnement ne serait plus systématique mais reprendrait seulement les événements caractéristiques de la situation. Une autre facilité qui peut être offerte concerne l'affichage de certains objets sous forme iconique (cas des objets agent initiateur et agent répondeur). Cet affichage permet de ne pas représenter les actions relatives à ces objets.

Le professeur peut-il modifier le programme ?

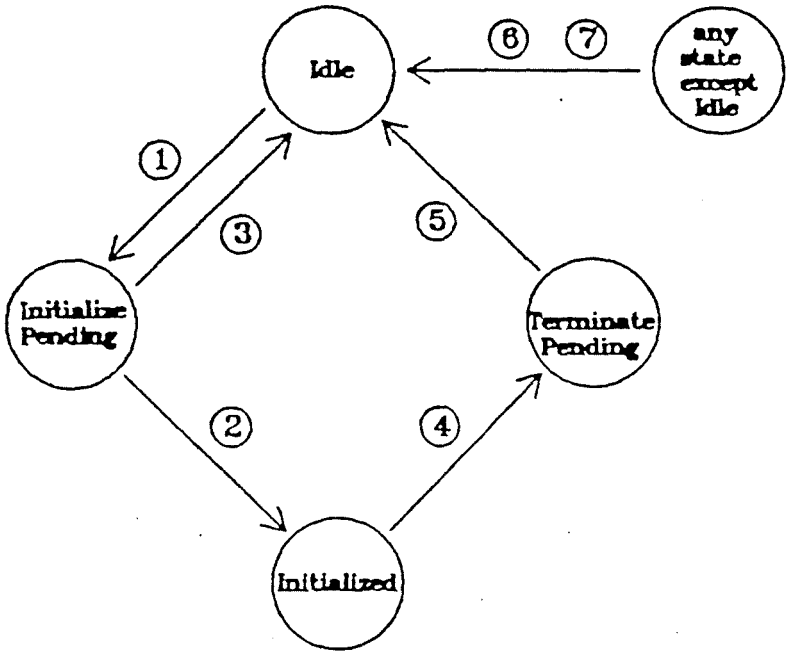
L'intervention doit se réaliser dans le code du programme. L'apprentissage de RMG est long et fastidieux. Aussi, seules certaines permutations dans l'enchaînement des actions peuvent-elles être réalisées sans trop de difficultés. L'approche orientée objet facilite grandement la programmation de la simulation si l'on se contente de faire appel à des méthodes relatives aux objets définis. Seul le concepteur du programme sera à même de modifier ou de réaliser les méthodes qui se rapportent aux objets. La programmation des unités d'interaction que nous avons adoptée permet une grande souplesse au niveau de la suppression de certains passages, de la modification du contenu d'un message, du changement des couleurs, etc.

Evaluation de l'efficacité du didacticiel et des apprentissages

Dans l'état actuel du projet, il n'est pas possible de répondre aux questions relatives à cette partie.

Annexe 3 : diagrammes de transitions d'états

Dans cette annexe nous avons regroupé les diagrammes de transitions d'états présentés dans la norme FTAM [ISO8571]. Ces diagrammes permettent de savoir quels sont les états suivants accessibles au départ de l'état courant de la machine de protocole de fichier, et cela pour l'initiateur comme pour le répondeur.

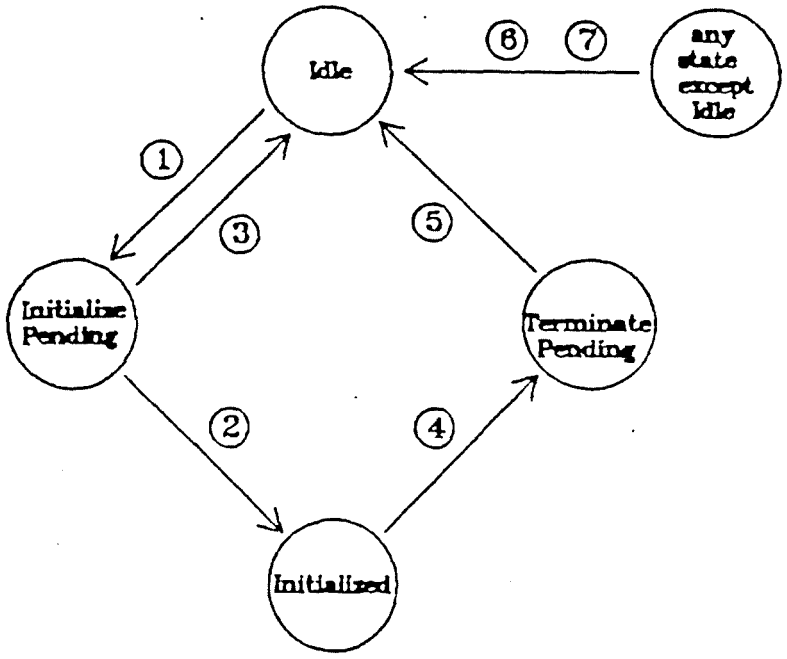


Key

Transitions

1 F-INITIALIZE request
2 F-INITIALIZE confirm (positive)
3 F-INITIALIZE confirm (negative)
4 F-TERMINATE request
5 F-TERMINATE confirm
6 F-U-ABORT request or indication
7 F-P-ABORT indication

Figure 10 - State Transition Diagram for Association Establishment (Initiator)

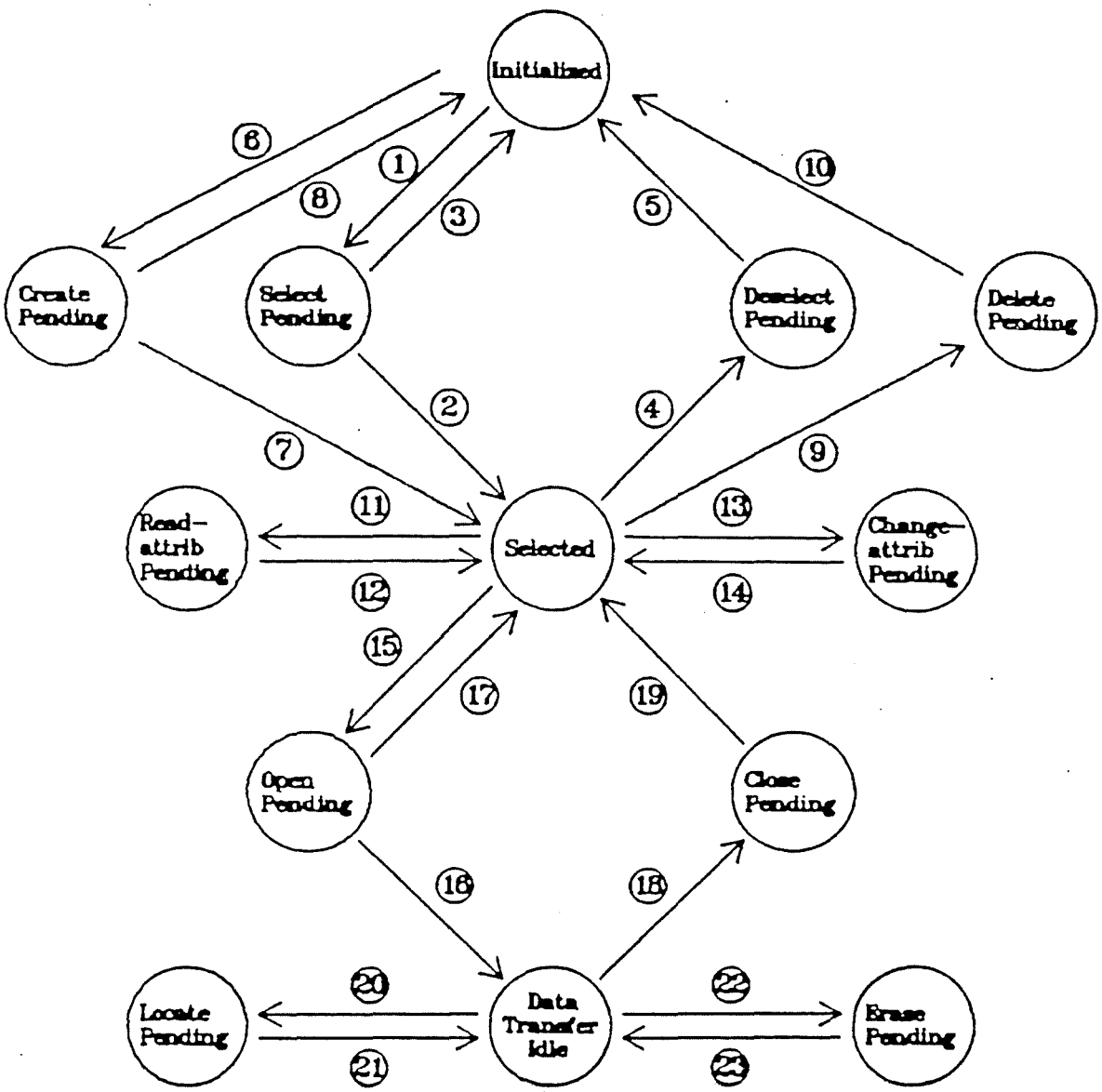


Key

Transitions

1 F-INITIALIZE indication
2 F-INITIALIZE response (positive)
3 F-INITIALIZE response (negative)
4 F-TERMINATE indication
5 F-TERMINATE response
6 F-U-ABORT request or indication
7 F-P-ABORT indication

Figure 11 - State Transition Diagram for Association Establishment (Responder)



ey See next page.

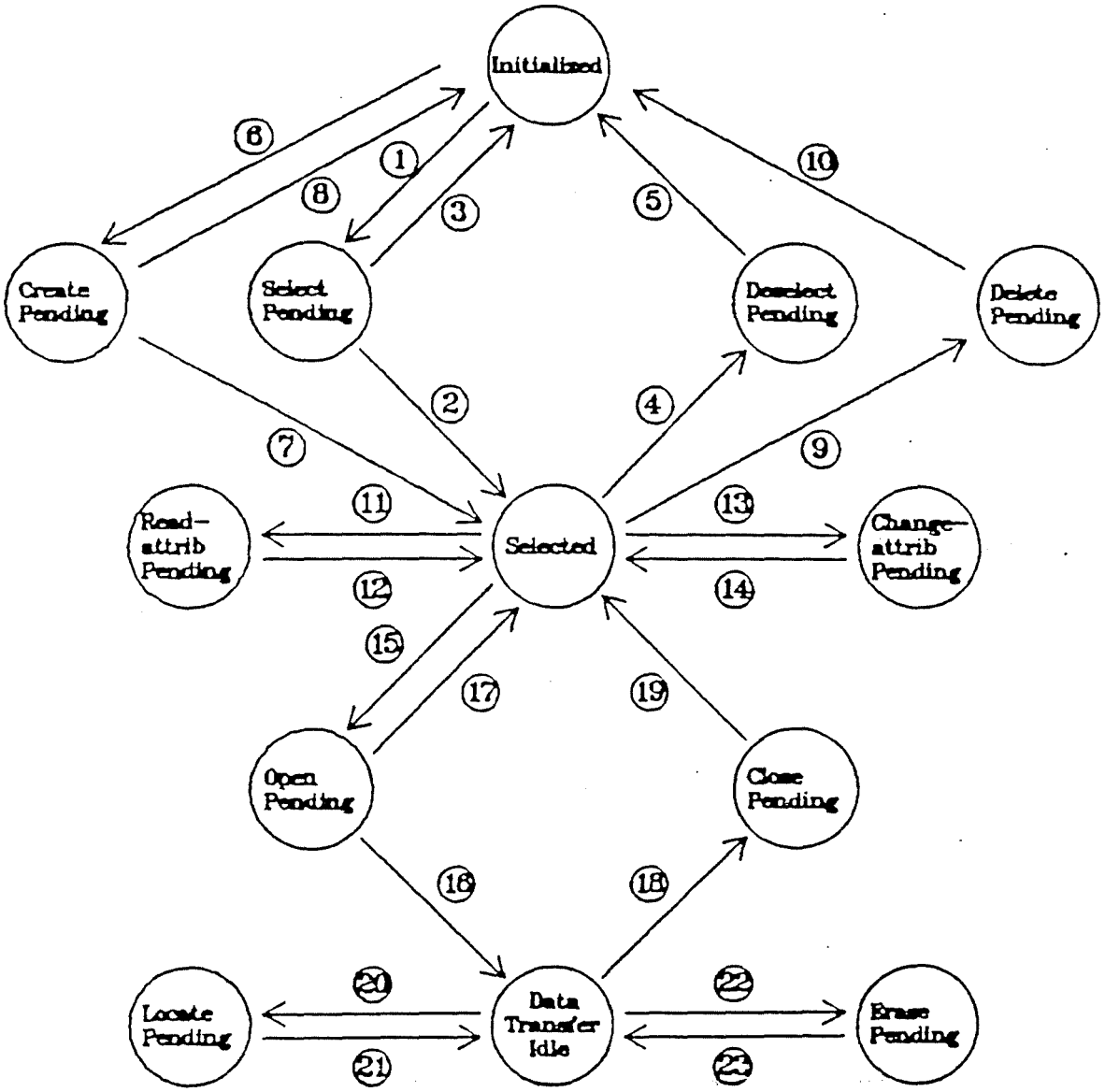
Figure 12 - State Transition Diagram of the File Regime Establishment Service (Initiator)

Key to Figure 12

Transitions

1 F-SELECT request
2 F-SELECT confirm (positive)
3 F-SELECT confirm (negative)
4 F-DESELECT request
5 F-DESELECT confirm
6 F-CREATE request
7 F-CREATE confirm (positive)
8 F-CREATE confirm (negative)
9 F-DELETE request
10 F-DELETE confirm
11 F-READ-ATTRIB request
12 F-READ-ATTRIB confirm
13 F-CHANGE-ATTRIB request
14 F-CHANGE-ATTRIB confirm
15 F-OPEN request
16 F-OPEN confirm (positive)
17 F-OPEN confirm (negative)
18 F-CLOSE request
19 F-CLOSE confirm
20 F-LOCATE request
21 F-LOCATE confirm
22 F-ERASE request
23 F-ERASE confirm

Figure 12 - Concluded



Key See next page.

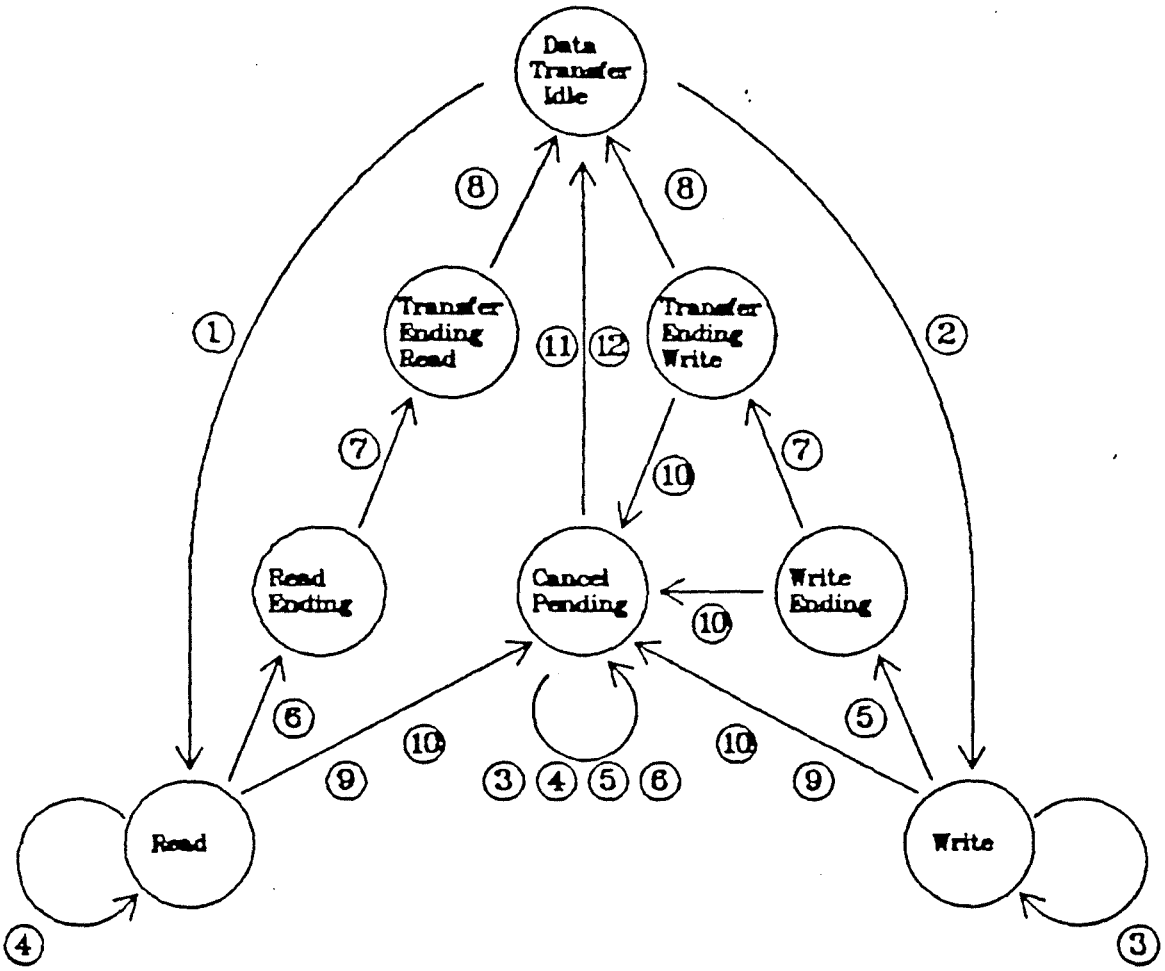
Figure 14 - State Transition Diagram of the File Regime Establishment Service (Responder)

Key to Figure 14

Transitions

. F-SELECT indication
2 F-SELECT response (positive)
3 F-SELECT response (negative)
4 F-DESELECT indication
5 F-DESELECT response
6 F-CREATE indication
7 F-CREATE response (positive)
8 F-CREATE response (negative)
9 F-DELETE indication
10 F-DELETE response
11 F-READ-ATTRIB indication
12 F-READ-ATTRIB response
13 F-CHANGE-ATTRIB indication
F-CHANGE-ATTRIB response
15 F-OPEN indication
16 F-OPEN response (positive)
17 F-OPEN response (negative)
18 F-CLOSE indication
19 F-CLOSE response
20 F-LOCATE indication
21 F-LOCATE response
22 F-ERASE indication
23 F-ERASE response

Figure 14 - Concluded

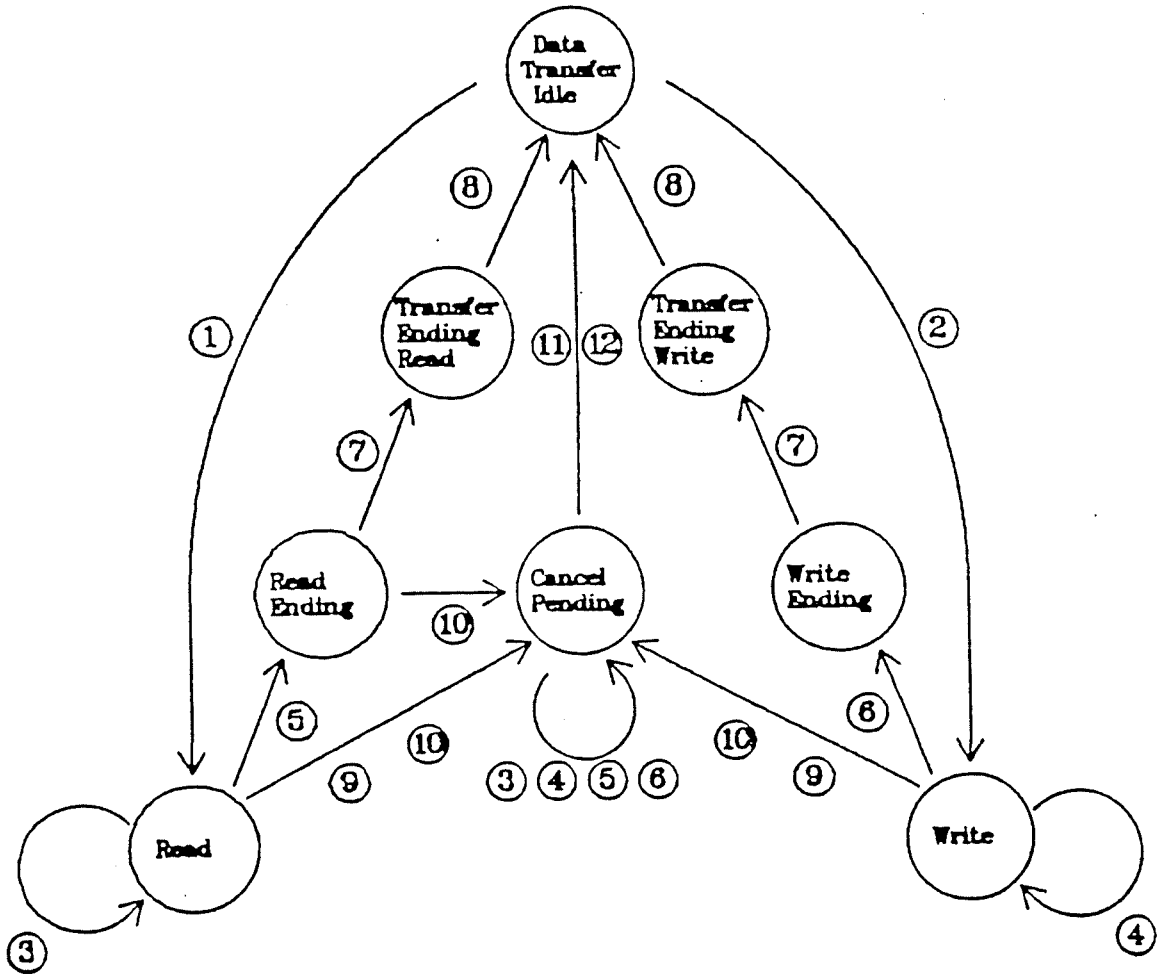


Key

Transitions

1 F-READ request
2 F-WRITE request
3 F-DATA request
4 F-DATA indication
5 F-DATA-END request
6 F-DATA-END indication
7 F-TRANSFER-END request
8 F-TRANSFER-END confirm
9 F-CANCEL request
10 F-CANCEL indication
11 F-CANCEL response
12 F-CANCEL confirm

Figure 16 - State Transition Diagram for the Bulk Data Transfer Service (Initiator)



Key

Transitions

1 F-READ indication
2 F-WRITE indication
3 F-DATA request
4 F-DATA indication
5 F-DATA-END request
6 F-DATA-END indication
7 F-TRANSFER-END indication
8 F-TRANSFER-END response
9 F-CANCEL request
10 F-CANCEL indication
11 F-CANCEL response
12 F-CANCEL confirm

Figure 17 - State Transition Diagram for the Bulk Data Transfer Service (Initiator)

Annexe 4 : spécification graphique de la primitive F-SELECT

Cette annexe illustre la spécification graphique complète de la primitive F-SELECT. La représentation permet peut-être de mieux se représenter comment nous envisageons les choses. Elle montre aussi le nombre important de graphes à réaliser pour spécifier un scénario complet.

L'apprenant sélectionne la primitive F-SELECT dans le menu déroulant. Le système se trouve dans le régime d'association FTAM. Les machines de proctocole de fichier sont dans l'état "Initialized".

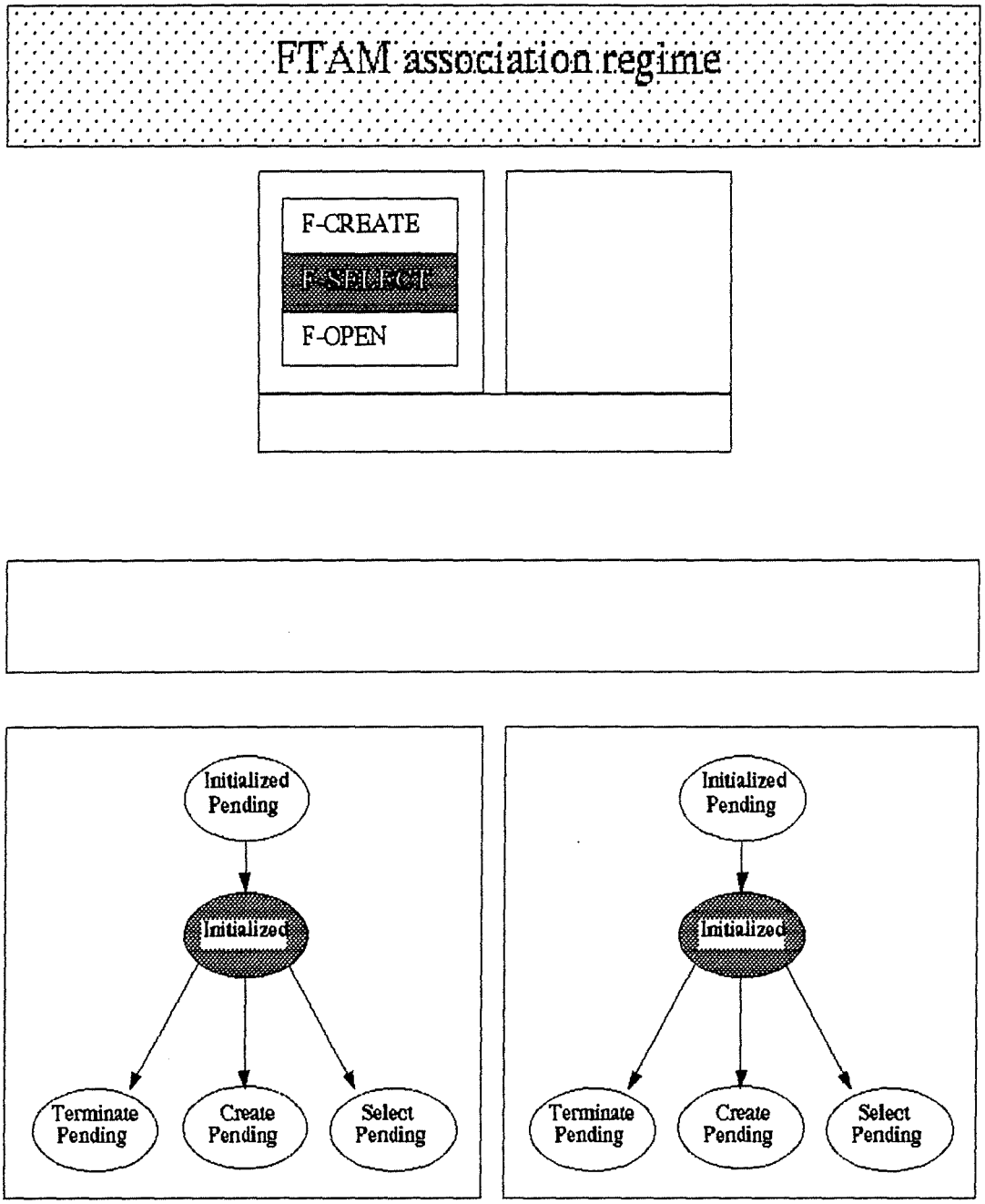


Figure 1 : sélection d'une primitive FTAM.

L'agent initiateur FTAM demande un service à son fournisseur par le biais de la primitive F-SELECT request.

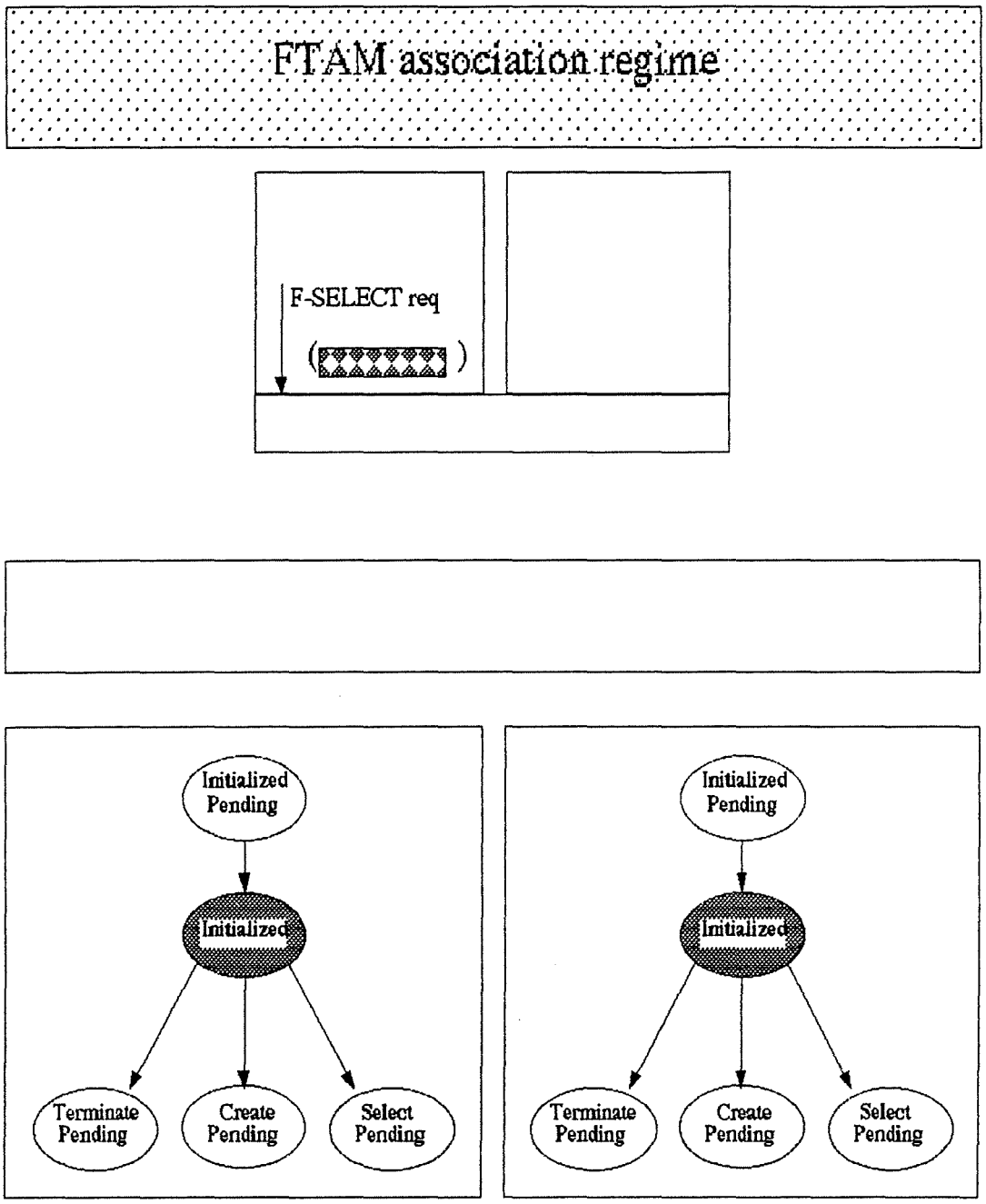


Figure 2 : envoi d'une requête FTAM

La machine de protocole de fichier côté initiateur change d'état. Elle passe dans l'état "Select pending". Le PDU relatif à la primitive de F-SELECT request est envoyé vers l'entité paire.

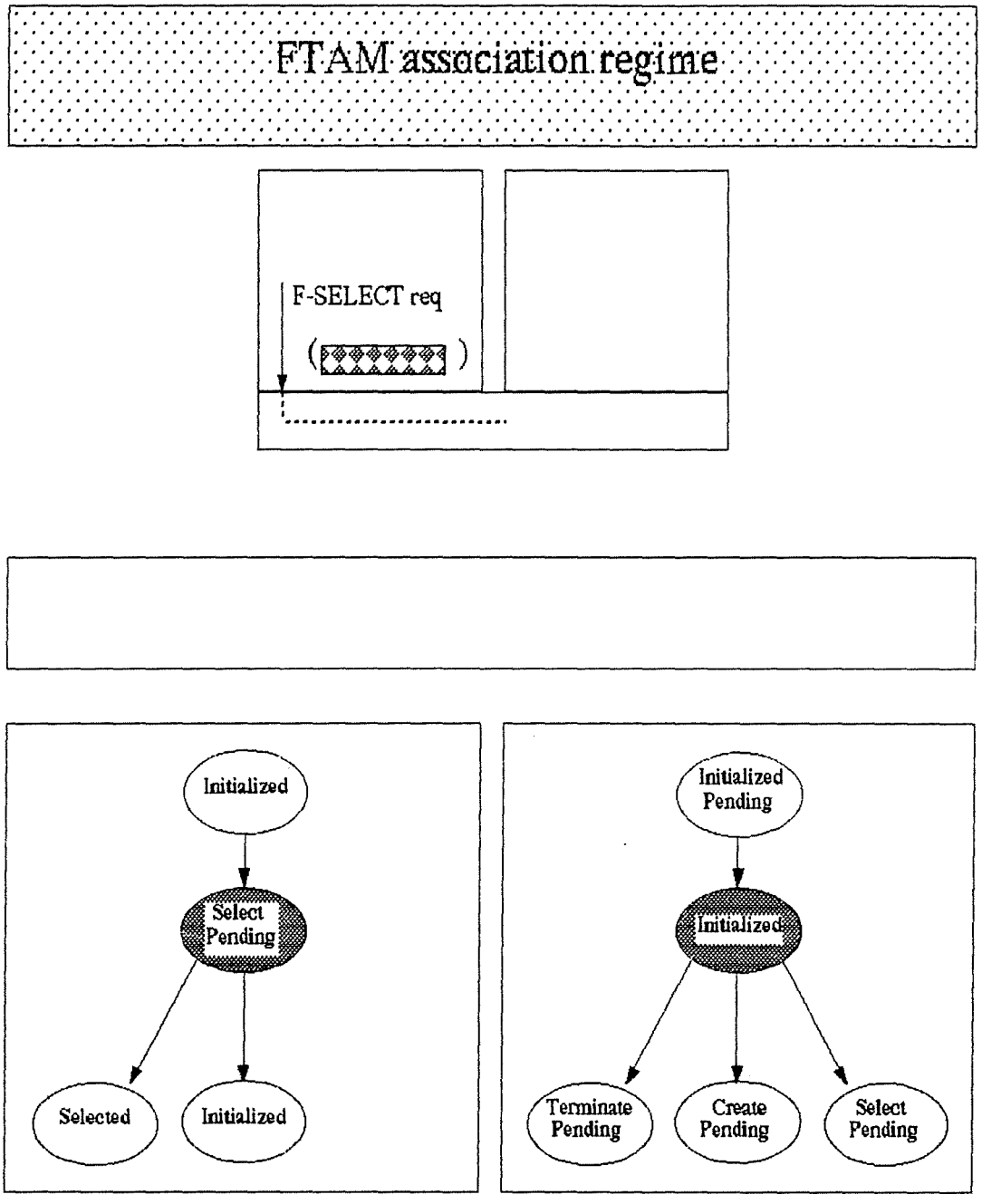


Figure 3 : changement d'état de la machine de protocole de fichier côté initiateur.

La machine de protocole de fichier côté répondeur change son état courant. Le PDU arrive de l'autre côté.

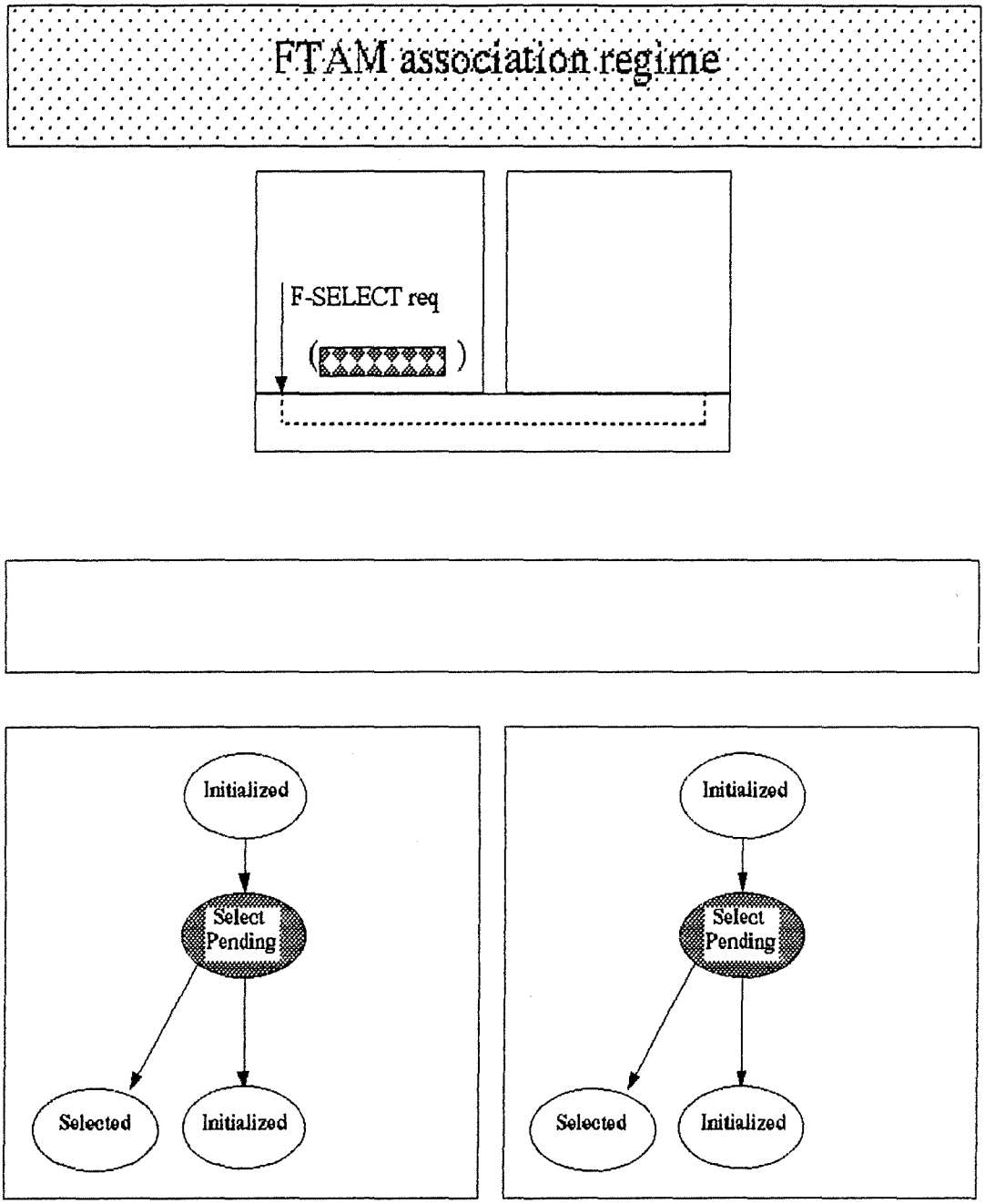


Figure 4 : changement d'état de la machine de protocole de fichier côté répondeur.

L'agent répondeur FTAM reçoit la demande de son agent pair. La primitive de F-SELECT indication lui est communiquée par son fournisseur.

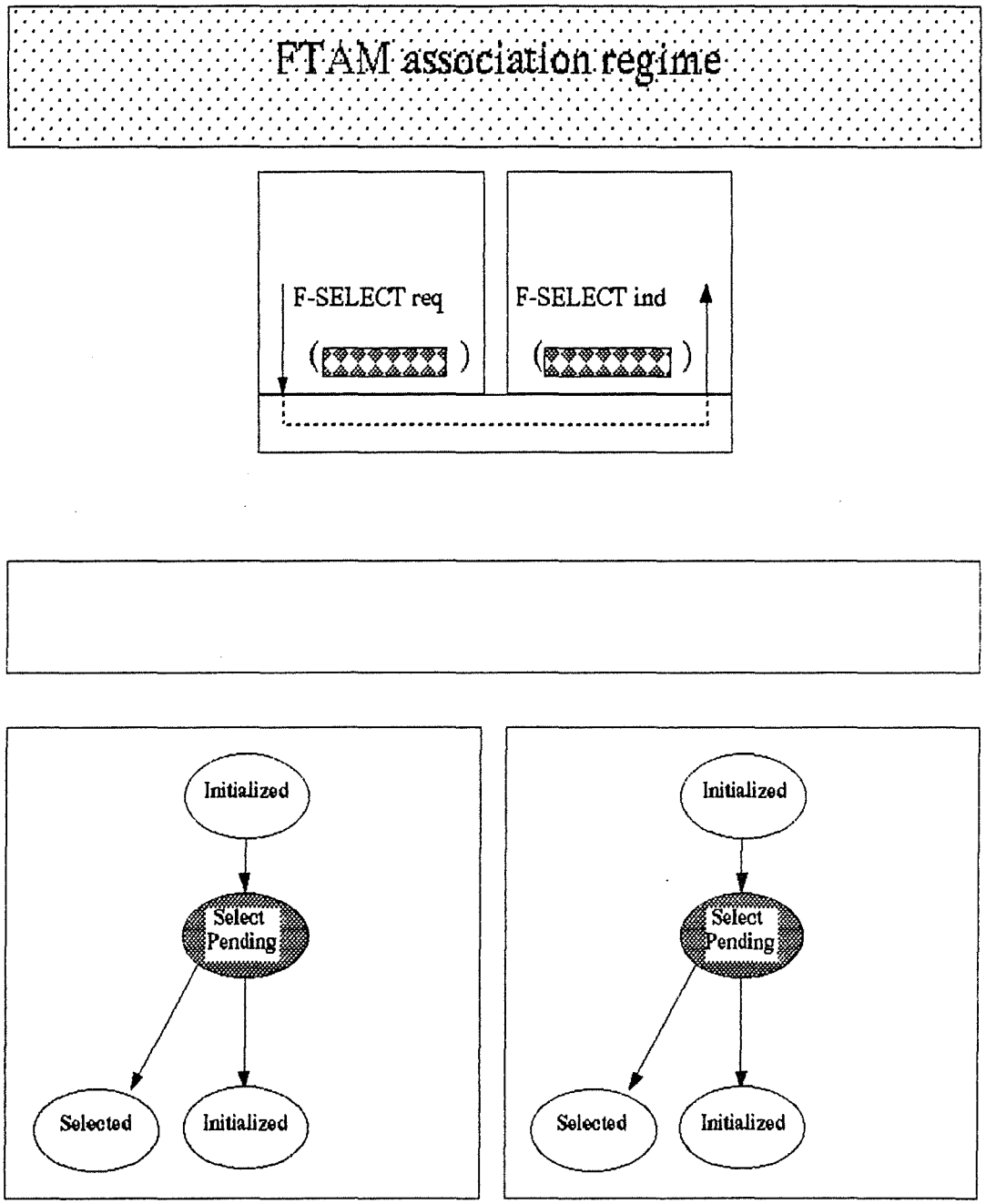


Figure 5 : communication d'une primitive d'indication FTAM.

Le didacticiel demande à l'apprenant d'accepter ou de refuser la sélection du fichier via les boutons de contrôle (ACKnowledged ou Not ACKnowledged) apparus dans l'agent répondeur.

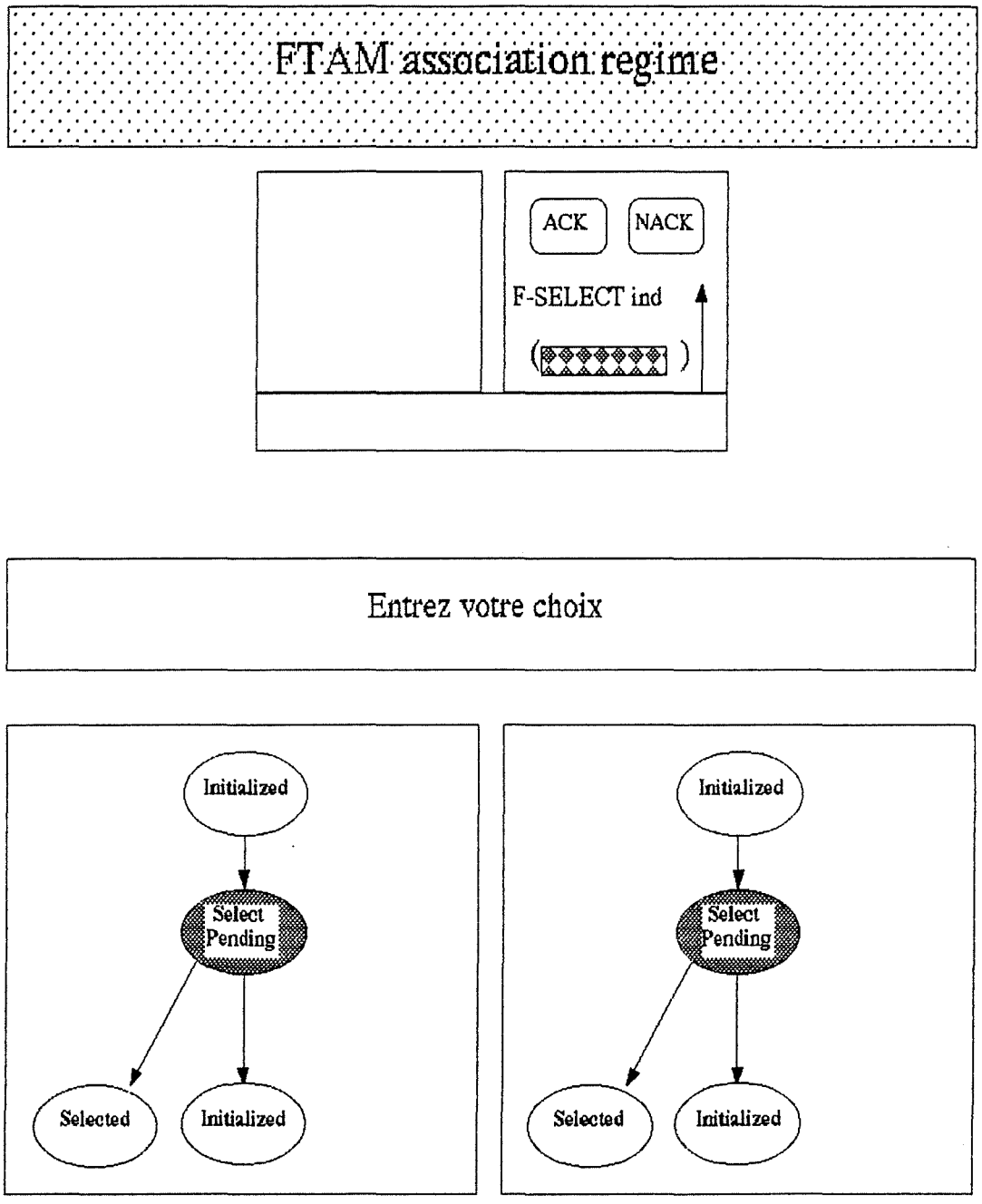


Figure 6 : affichage des boutons de contrôle.

L'utilisateur ayant choisi d'accepter la sélection du fichier, l'agent répondeur envoie une primitive de confirmation positive.

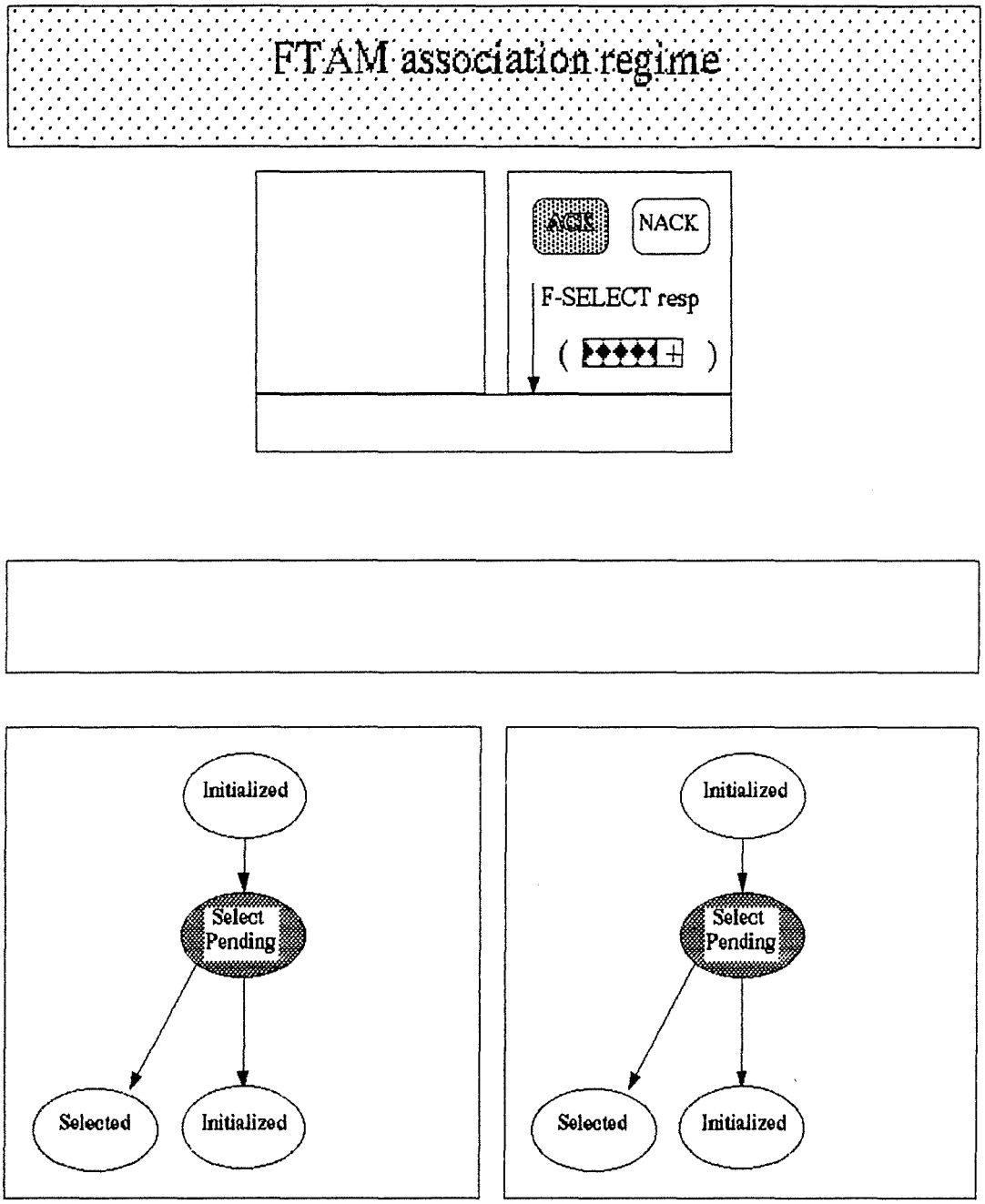


Figure 7 : envoi d'une primitive de réponse positive.

Le fichier sélectionné apparaît du côté répondeur. Les attributs sont affichés car ils sont accessibles à partir du régime de sélection de fichier. La

machine de protocole de fichier du répondeur passe dans l'état "Selected". Le PDU contenant la réponse est transmis à l'entité paire.

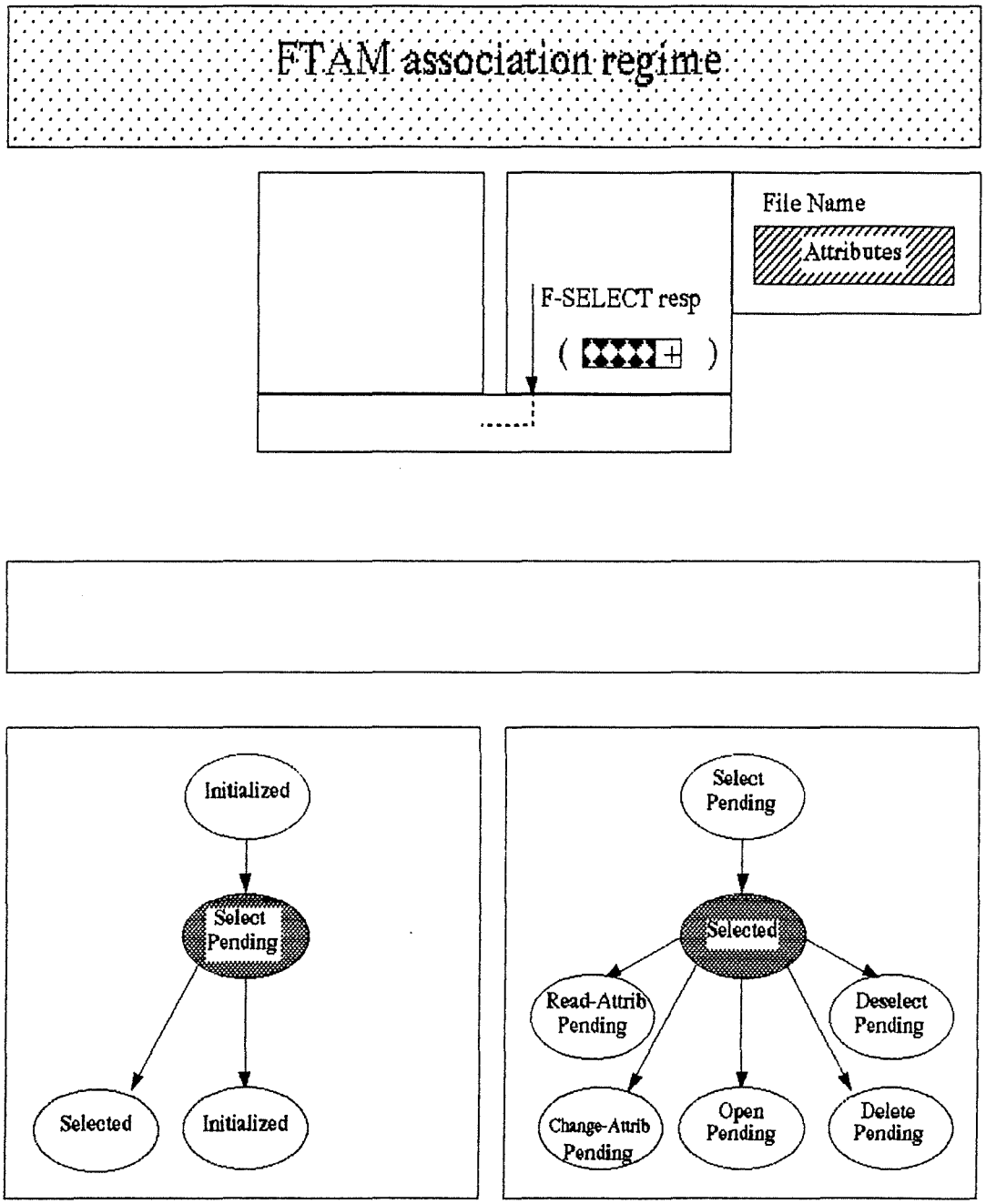


Figure 8 : changement d'état de la machine de protocole de fichier côté répondeur.

La machine de protocole de fichier côté initiateur passe dans l'état "Selected". Le PDU arrive du côté initiateur.

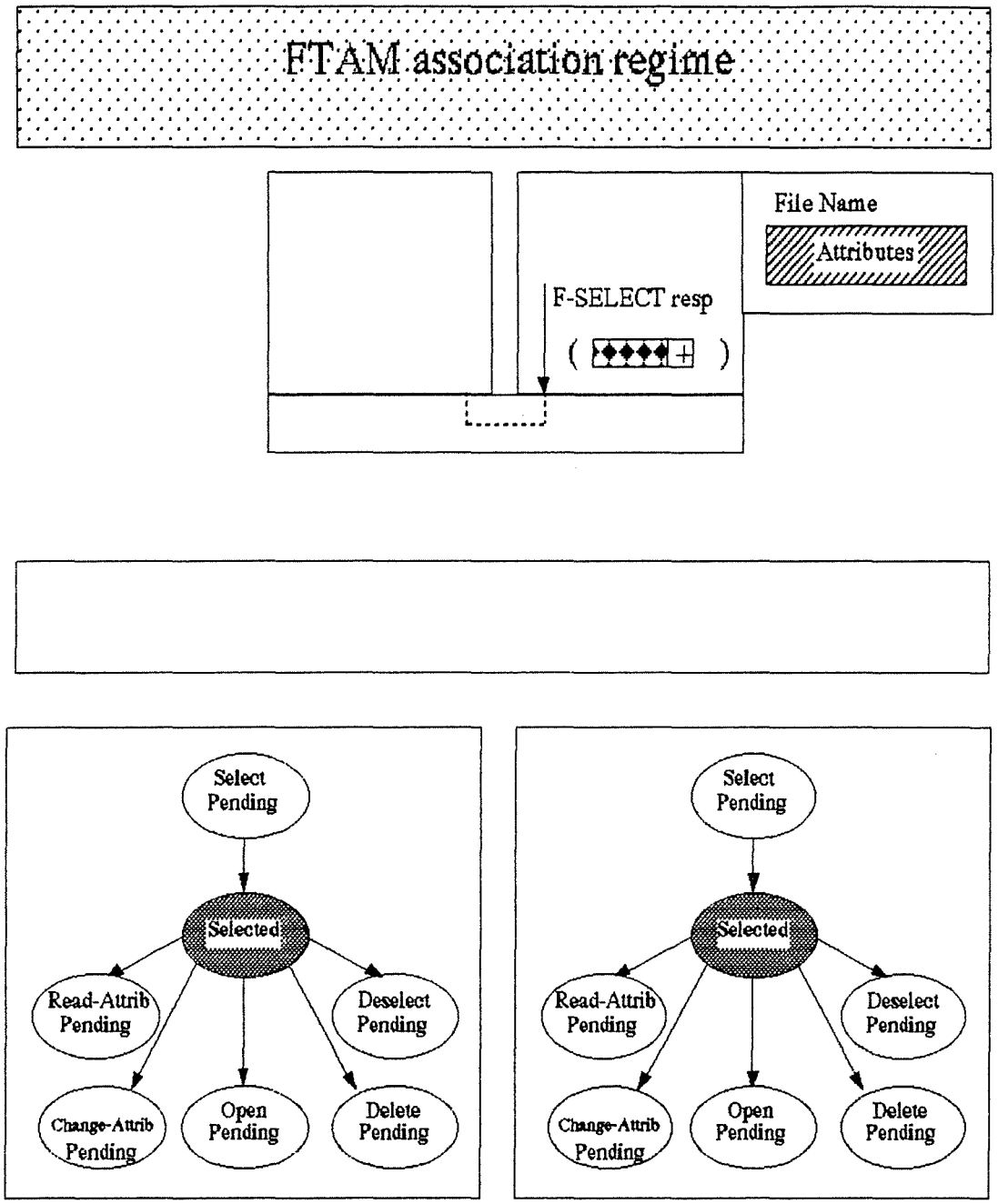


Figure 9 : changement d'état de la machine de protocole de fichier côté initiateur.

La confirmation positive relative à la demande de sélection de fichier est transmise à l'agent initiateur sous la forme d'une primitive de F-SELECT confirmation. Le régime de sélection de fichier est établi.

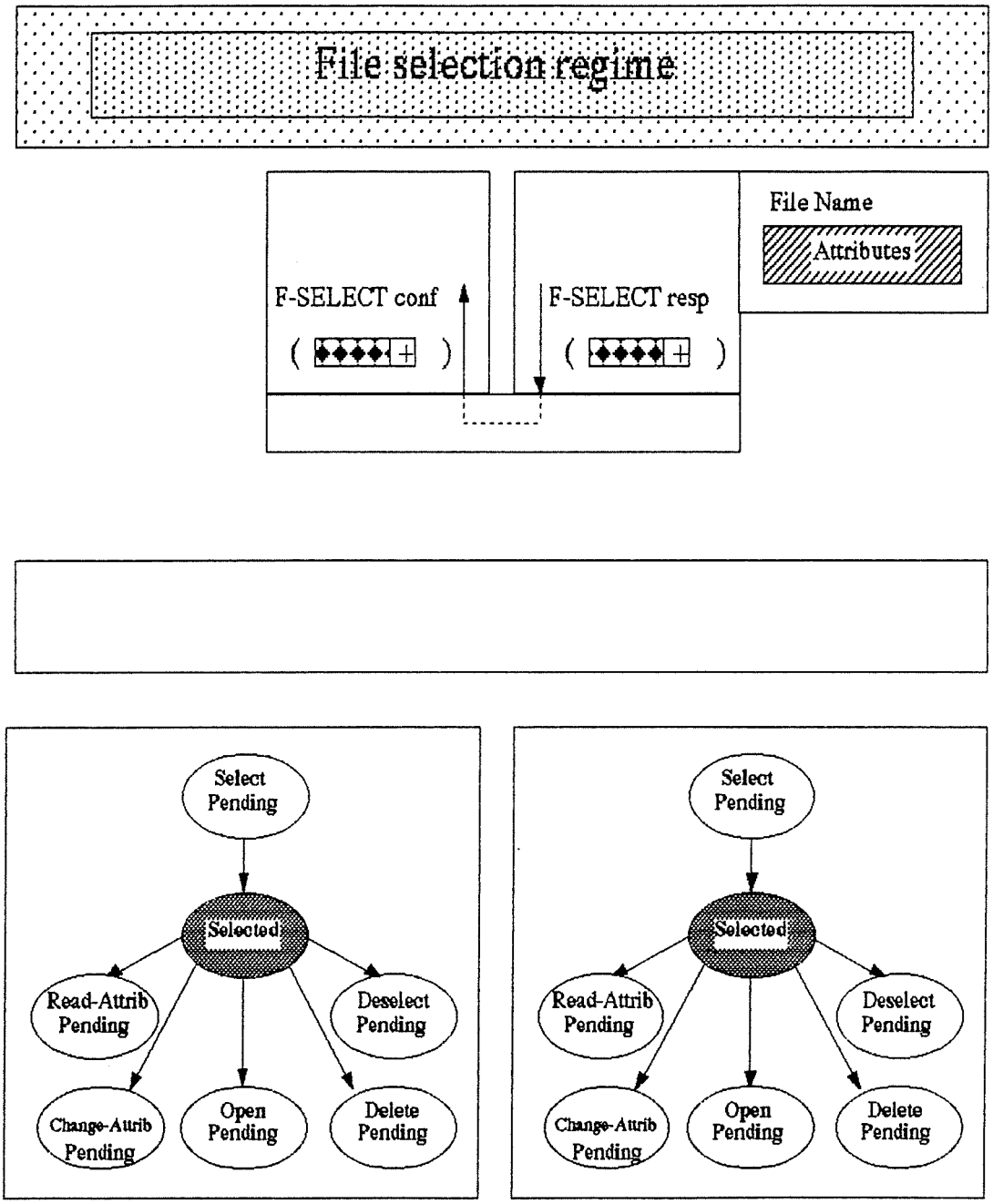


Figure 10 : communication d'une primitive FTAM de confirmation.

Les informations disponibles sont affichées du côté initiateur. Il s'agit du nom du fichier sélectionné.

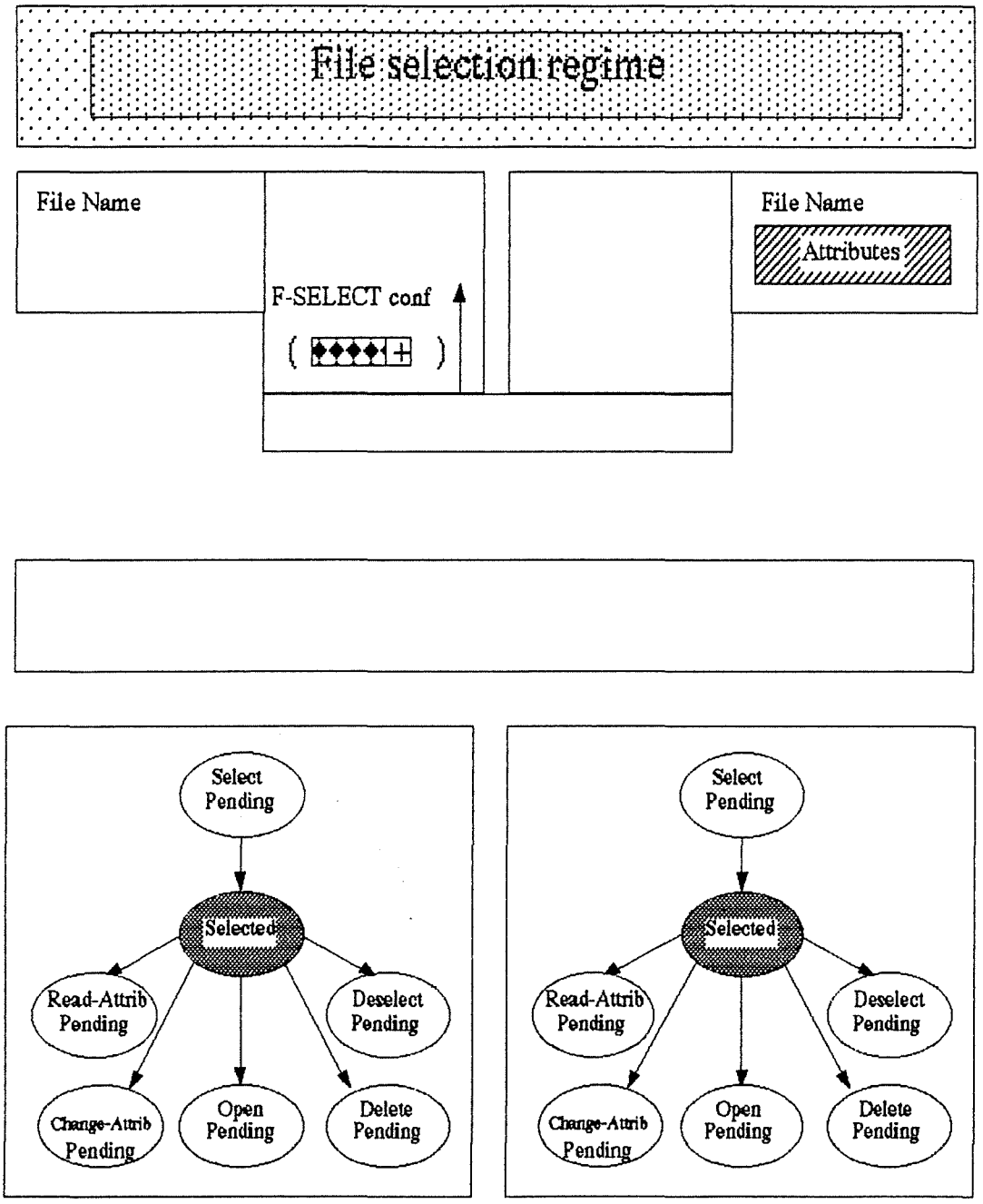


Figure 11 : traitement associé à la primitive reçue.

L'exécution des actions relatives à la primitive F-SELECT est terminée. L'application attend le choix d'une nouvelle primitive par l'utilisateur.

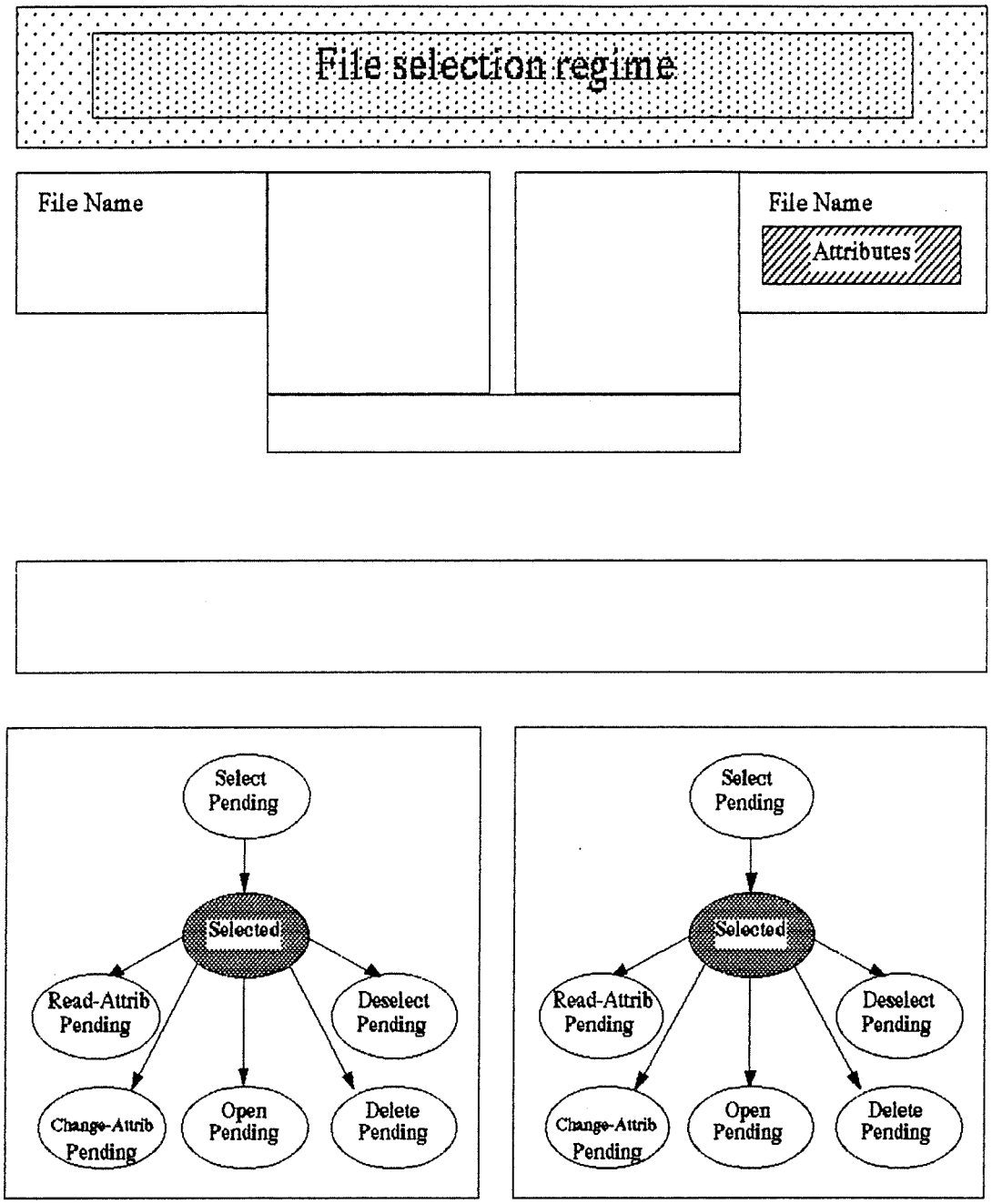


Figure 12 : attente d'une nouvelle primitive FTAM.

Annexe 5 : code objective-C des classes réalisées

Cette annexe reprend les listings commentés des classes que nous avons développées. Il s'agit des classes Ftam et Ftamétat.

```
#include "objc.h"
#include "rmg.h"
#include "envir.h"

/* definition des couleurs */

#define coulcadre CYAN
#define coulfondregime WHITE
#define coulfondagent PERIWINKLE
#define coulfondbouton WHITE
#define coulbouton_ok BLUE
#define coulbouton_non_ok RED
#define coulfondfournisseur PERIWINKLE
#define coulfondmessage BLUE
#define coulagent PINK
#define coulerreur YELLOWGREEN5
#define coulfondmenu WHITE
#define coulcadremenu RED

/* definition des temps de pause */

#define temps 1000
#define anitemps 4

/* definition des primitives Ftam */

#define FINIRQ 0
#define FINIRSPPOS 1
#define FINIRSPNEG 2
#define FTERRQ 3
#define FTERRSP 4

/* definition des messages courants */

static char *sregime="FTAM regime :";
static char *sregime1="FTAM association regime.";
static char *sinitiateur="Initiator agent";
static char *sini="F-INITIALIZE req";
static char *srepondeur="Responder agent";
static char *srep="F-INITIALIZE ind";
static char *sfournisseur="FTAM provider";
static char *smessage="Messages :";
static char *smess1="Please choose a primitive in the menu.";
static char *sligne1="F-INIT";
static char *sligne2="F-TERM";
static char *sligne3="!!QUIT!!";
static char *serreur="INVALID PRIMITIVE FOR THIS STATE.  Please choose an other one.";
static char *smessini="The initiator FPM changes its current state.";
static char *smessrep="The responder FPM changes its current state.";
static char *sparam="( PARAMETERS )";
static char *sparamplus="( RESULT: + )";
static char *sparammoins="( RESULT: - )";

/* definition des fonctions C a objectiver */

static id choix_finirq = NULL;
static id choix_fterrq = NULL;
static id choix_quit = NULL;
static id choix_ok = NULL;
static id choix_non_ok = NULL;

@requires FontMngr, RMGString, RMGAction, Ftametat, Mouse, RMGLine;
```

```
= Ftam: Envir(corbugy,working,RMGVW,Primitive,Collection)
{

/* definition des objets de la scene */

id obj_regime, regime, titre_reg;
id obj_initiateur, initiateur, initiateur1, titre_ini;
id obj_repondeur, repondeur, repondeur1, titre_rep;
id obj_fournisseur, fournisseur, titre_fourn;
id obj_message, message, titre_mess, souris;
id bouton_ok, bouton_non_ok;
id obj_menu, ligne_1, ligne_2, ligne_3;
id obj_inietat, obj_repetat;

/* definition des lignes d'animation */

id ligne1, ligne2, ligne3;

/* definition des indicateurs */

int choix;
int ok, non_ok;

}

/* definition des fonctions C a objectiver */

static id c_choix_finirq(caller) id caller;
{
/* cette fonction appelle l'unite d'interaction UIO avec le choix=1 (F-INIT-REQ) */

 id receiver;

 receiver = [caller viewIcon];
 receiver->choix=1;
 [receiver UIO];
}

static id c_choix_fterrq(caller) id caller;
{
/* cette fonction appelle l'unite d'interaction UIO avec le choix=2 (F-TER-REQ) */

 id receiver;

 receiver = [caller viewIcon];
 receiver->choix=2;
 [receiver UIO];
}

static id c_choix_quit(caller) id caller;
{
/* cette fonction termine l'application */

 id receiver;

 receiver = [caller viewIcon];
 [receiver quit_app];
}

static id c_choix_ok(caller) id caller;
{
/* cette fonction appelle l'unite d'interaction UIO avec le choix ok realise */

 id receiver;

 receiver=[caller viewIcon];
 receiver->ok = 1;
 [receiver UIO];
}
```

```
static id c_choix_non_ok(caller) id caller;
{
/* cette fonction appelle l'unite d'interaction UIO avec le choix non_ok realise */

 id receiver;

 receiver=[caller viewIcon];
 receiver->non_ok = 1;
 [receiver UIO];
}

+newIn: rootView
{
/* cette methode definit la scene et les objets de la scene */

/* definition des polices de caracteres */

id nameFont, regimeFont, messFont, etatFont;

nameFont = [FontMngr sysfont];
regimeFont = [FontMngr pica18x30];
etatFont = [FontMngr lp10x20_b];
messFont = [FontMngr cour12x20];

/* initialisation des fonctions C a objectiver */

if (! choix_finirq) {choix_finirq = [RMGAction newFuncAction: c_choix_finirq]; };
if (! choix_fterrq) {choix_fterrq = [RMGAction newFuncAction: c_choix_fterrq]; };
if (! choix_quit) {choix_quit = [RMGAction newFuncAction: c_choix_quit]; };
if (! choix_ok) {choix_ok = [RMGAction newFuncAction: c_choix_ok]; };
if (! choix_non_ok) {choix_non_ok = [RMGAction newFuncAction: c_choix_non_ok]; };

/* creation des objets de la scene */
/* creation de la scene */
self=[[[[self origin: crossx: crossy extent: 1200: 1000 superview: rootView bkgd: LIGHTGREY]
                frameWidth: 3] frameColor: coulcadre] idAction: moveWithBandAction];

/* creation de l'objet regime */
obj_regime = [[[RMGString relative: 20: 910 extent: 1160: 70 superview: self bkgd: coulfondregime]
                           frameWidth: 3] frameColor: coulcadre];
titre_reg = [RMGString relative: 0: 45 extent: 120: 25 superview: obj_regime bkgd: coulfondregime];
[[[[[[titre_reg font: nameFont] color: BLACK] string: sregime] horFill: TRUE] center] centerV];
regime = [[[RMGString relative: 100: 0 extent: 1000: 40 superview: obj_regime bkgd: coulfondregime]
                       font: regimeFont] color: BLACK];

/* creation de l'objet agent initiateur */
obj_initiateur = [[[RMGString relative: 340: 690 extent: 200: 200 superview: self bkgd: coulfondagent]
                            frameWidth: 3] frameColor: coulcadre];
titre_ini = [RMGString relative: 0: 175 extent: 200: 25 superview: obj_initiateur bkgd: coulfondagent];
[[[[[[titre_ini font: nameFont] color: WHITE] string: sinitiateur] horFill: TRUE] center] centerV];
initiateur = [RMGString relative: 0: 30 extent: 200: 25 superview: obj_initiateur bkgd: coulfondagent];
[[[[[initiateur font: etatFont] color: coulagent] horFill: TRUE] center] centerV];
initiateur1 = [[[[RMGString relative: 0: 0 extent: 200: 30 superview: obj_initiateur bkgd: coulfondagent]
                           font: etatFont] color: coulagent] horFill: TRUE];

/* creation de l'objet agent repondeur */
obj_repondeur = [[[RMGString relative: 660: 690 extent: 200: 200 superview: self bkgd: coulfondagent]
                            frameWidth: 3] frameColor: coulcadre];
titre_rep = [RMGString relative: 0: 175 extent: 200: 25 superview: obj_repondeur bkgd: coulfondagent];
[[[[[[titre_rep font: nameFont] color: WHITE] string: srepondeur] horFill: TRUE] center] centerV];
repondeur = [[[RMGString relative: 0: 30 extent: 200: 25 superview: obj_repondeur bkgd: coulfondagent]
                       font: etatFont] color: coulagent];
repondeur1 = [[[[RMGString relative: 0: 0 extent: 200: 30 superview: obj_repondeur bkgd: coulfondagent]
                      font: etatFont] color: coulagent] horFill: TRUE];
```

```
/* creation de l'objet fournisseur */
obj_fournisseur = [[[RMGString relative: 340: 615 extent: 520: 78 superview: self bkgd: coulfondfournisseur]
                                frameWidth: 3] frameColor: coulcadre];
titre_fourn = [RMGString relative: 0: 0 extent: 520: 28 superview: obj_fournisseur bkgd: coulfondfournisseur];
[[[[[[titre_fourn font: nameFont] color: WHITE] string: sfournisseur] horFill: TRUE] center] centerV];

/* creation de l'objet message */
obj_message = [[[RMGString relative: 0: 0 extent: 1200: 125 superview: self bkgd: coulfondmessage]
                            frameWidth: 3] frameColor: coulcadre];
titre_mess = [RMGString relative: 0: 100 extent: 90: 25 superview: obj_message bkgd: coulfondmessage];
[[[[[[titre_mess font: nameFont] color: WHITE] string: smessage] horFill: TRUE] center] centerV];
message = [RMGString relative: 30:65 extent: 1140: 30 superview: obj_message bkgd: coulfondmessage];
[[[[message font: messFont] color: WHITE] string: smess1] horFill: TRUE];
souris = [RMGString relative: 0: 10 extent: 1200: 30 superview: obj_message bkgd: coulfondmessage];
[[[souris font: messFont] color: DARKPINK] horFill: TRUE];

/* creation des boutons de selection */
bouton_ok = [[[RMGString relative: 40: 100 extent: 40: 30 superview: obj_repondeur bkgd: coulfondbouton]
                          frameWidth: 3] frameColor: coulbouton_ok];
[[[[[[[[bouton_ok font: nameFont] color: coulbouton_ok] string: "YES"] horFill: TRUE] center] centerV]
                    idAction: choix_ok] viewIcon: self];
bouton_non_ok = [[[RMGString relative: 120: 100 extent: 40: 30 superview: obj_repondeur bkgd: coulfondbouton]
                              frameWidth: 3] frameColor: coulbouton_non_ok];
[[[[[[[[bouton_non_ok font: nameFont] color: coulbouton_non_ok] string: "NO"] horFill: TRUE] center] centerV]
                        idAction: choix_non_ok] viewIcon: self];

/* creation de l'objet menu */
obj_menu = [[[RMGString relative: 290: 770 extent: 100: 90 superview: self bkgd: coulfondmenu]
                         frameWidth: 0] frameColor: coulcadremenu];
ligne_1 = [[[RMGString relative: 0: 60 extent: 100: 30 superview: obj_menu bkgd: coulfondmenu]
                        frameWidth: 3] frameColor: coulcadremenu];
[[[[[[[[ligne_1 font: nameFont] color: BLACK] string: sligne1] horFill: TRUE] center] centerV]
                  idAction: choix_finirq];
[ligne_1 viewIcon: self];
ligne_2 = [[[RMGString relative: 0: 30 extent: 100: 30 superview: obj_menu bkgd: coulfondmenu]
                        frameWidth: 3] frameColor: coulcadremenu];
[[[[[[[[ligne_2 font: nameFont] color: BLACK] string: sligne2] horFill: TRUE] center] centerV]
                  idAction: choix_fterrq];
[ligne_2 viewIcon: self];
ligne_3 = [[[RMGString relative:0 : 0 extent: 100: 30 superview: obj_menu bkgd: coulfondmenu]
                        frameWidth: 3] frameColor: coulcadremenu];
[[[[[[ligne_3 font: nameFont] color: BLACK] string: sligne3] horFill: TRUE] center] centerV];
[[ligne_3 viewIcon: self] idAction: choix_quit];

/* creation des objtes etat initiateur et repondeur */
obj_inietat=[Ftametat creer: self: 50: 200: "Initiator states"];
obj_repetat=[Ftametat creer: self: 650: 200: "Responder states"];

return self;
}

- extraNewIn: rootView
{
/* cette methode initialise les objets de la scene */

[self showAll];
[obj_fournisseur hide];
[bouton_ok hide];
[bouton_non_ok hide];
[obj_inietat initEtat];
[obj_repetat initEtat];
```

```
/* initialisation des variables */

ok = 0;
non_ok = 0;
choix = 0;

return self;
}

- UIO
{
/* cette methode appelle la methode correspondant a l'unite d'interaction choisie */

 [obj_menu hide];
 switch(choix)
 { case 1 : [self UI1]; break;
   case 2 : [self UI2]; break;
   default : printf("erreur!! %d",choix);
 };
}

- UI1
{

  if (ok == 0 && non_ok == 0 )
  {
   /* execution de la methode dans le cas du F-INITIALIZE request */

   /* affichage de la primitive de l'agent initiateur */
   [[[[initiateur string: "F-INITIALIZE req"] center] centerV] update];
   [[[[initiateur1 string: sparam] center] centerV] update];
   [[[message string: "The FTAM initiator agent wants to establish an association."] horFill: TRUE] update];
   [self temporise];
   if ([obj_inietat afficheEtat: FINIRQ])
     {
       /* execution du F-INITIALIZE request car la machine de protocole est dans un etat valide pour cette primitive */
       /* envoi du PDU */
       [[message string: smessini] update];
       [self temporise];
       [obj_fournisseur showAll];
       [[message string: "The Ftam provider conveys the F-INITIALIZE REQUEST PDU."] update];
       [self animationIniRep];
       [self temporise];
       /* affichage de l'indication */
       [obj_repetat afficheEtat: FINIRQ];
       [[message string: smessrep] update];
       [self temporise];
       [[[[repondeur string: "F-INITIALIZE ind"] center] centerV] update];
       [[[[repondeur1 string: sparam] center] centerV] update];
       [[[message string: "The FTAM provider sends an indication relative to an association establishment."]
                 horFill: TRUE] update];
       delay(temps);
       /* effacement des messages obsoletes */
       [initiateur erase];
       [initiateur1 erase];
       [self animationEfface];
       [self temporise];
       /* demande de reponse positive ou negative */
       [[[message string: "Accept or refuse the association."] horFill: TRUE] update];
       [bouton_ok showAll];
       [bouton_non_ok showAll];
     }
```

```
    else
    {
      /* execution de l'erreur car la machine de protocole n'est pas dans un etat valide pour la primitive F-INITIALIZE request */
      [[[message string: serreur] color: coulerreur] update];
      [message color: WHITE];
      printf("\007");
      [initiateur erase];
      [initiateur1 erase];
      [obj_menu showAll];
    };
}
else
{
 [bouton_ok hide];
 [bouton_non_ok hide];
 [repondeur erase];
 if (ok == 1)
 {
  /* execution de la methode dans le cas du F-INITIALIZE response positif */
  ok = 0;
  [[[[repondeur string: "F-INITIALIZE resp"] center] centerV] update];
  [[[[repondeur1 string: sparamplus] center] centerV] update];
  [[message string: "The FTAM responder agent accepts the association."] update];
  [self temporise];
  [obj_repetat afficheEtat: FINIRSPPOS];
  [[message string: smessrep] update];
  [self temporise];
  [[message string: "The FTAM provider conveys the F-INITIALIZE RESPONSE(+) PDU."] update];
  [self animationRepIni];
  [self temporise];
  [obj_inietat afficheEtat: FINIRSPPOS];
  [[message string: smessini] update];
  [self temporise];
  [[[[initiateur string: "F-INITIALIZE conf"] center] centerV] update];
  [[[[initiateur1 string: sparamplus] center] centerV] update];
  [[message string: "The FTAM provider sends a confirmation relative to an association establishment."] update];
  delay(temps);
  [repondeur erase];
  [repondeur1 erase];
  [self animationEfface];
  [self temporise];
  [[[[[regime string: sregime1] horFill: TRUE] center] centerV] update];
  [[message string: "The FTAM association regime is established."] update];
  delay(temps);
  [initiateur erase];
  [initiateur1 erase];
  [self temporise];
  [[[message string: smess1] horFill: TRUE] update];
  [obj_menu showAll];
 }
 else
 {
  /* execution de la methode dans le cas du F-INITIALIZE response negatif */
  non_ok = 0;
  [[[[repondeur string: "F-INITIALIZE resp"] center] centerV] update];
  [[[[repondeur1 string: sparammoins] center] centerV] update];
  [[message string: "The FTAM responder agent refuses the association."] update];
  [self temporise];
  [obj_repetat afficheEtat: FINIRSPNEG];
  [[message string: smessrep] update];
  [self temporise];
  [[message string: "The FTAM provider conveys the F-INITIALIZE RESPONSE(-) PDU."] update];
  [self animationRepIni];
  [self temporise];
```

```
        [obj_inietat afficheEtat: FINIRSPNEG];
        [[message string: smessini] update];
        [self temporise];
        [[[[initiateur string: "F-INITIALIZE conf"] center] centerV] update];
        [[[[initiateur1 string: sparammoins] center] centerV] update];
        [[message string: "The FTAM provider sends the negative confirmation relative to an association establishment."] update];
        delay(temps);
        [repondeur erase];
        [repondeur1 erase];
        [self animationEfface];
        [self temporise];
        [obj_fournisseur hide];
        [[message string: "The lower layers connections are released."] update];
        delay(temps);
        [initiateur erase];
        [initiateur1 erase];
        [self temporise];
        [[[message string: smess1]  horFill: TRUE] update];
        [obj_menu showAll];
      }
    }
  }

  - UI2
  {
    /* execution de la methode pour le F-TERMINATE request */

    [[[[initiateur1 string: "F-TERMINATE req"] center] centerV] update];
    [[[message string: "The FTAM initiator agent wants to release the association."] horFill: TRUE] update];
    [self temporise];
    if ([obj_inietat afficheEtat: FTERRQ])
  {
    [[message string : smessini] update];
    [self temporise];
    [self animationIniRep];
    [[message string: "The FTAM provider conveys the F-TERMINATE REQUEST PDU."] update];
    [self temporise];
    [obj_repetat afficheEtat: FTERRQ];
    [[message string: smessrep] update];
    [self temporise];
    [[[[repondeur1 string: "F-TERMINATE ind"] center] centerV] update];
    [[[message string: "The FTAM provider sends an indication relative to an association release."] horFill: TRUE] update];
    delay(temps);
    [initiateur1 erase];
    [self animationEfface];
    [self temporise];
    [repondeur1 erase];
    [[[[repondeur string: "F-TERMINATE resp"] center] centerV] update];
    [[[[repondeur1 string: sparam] center] centerV] update];
    [[[message string: "The FTAM responder agent sends a response for association release."] horFill: TRUE] update];
    [self temporise];
    [obj_repetat afficheEtat: FTERRSP];
    [[message string: smessrep] update];
    [self temporise];
    [[message string: "The FTAM provider conveys the F-TERMINATE RESPONSE PDU."] update];
    [self animationRepIni];
    [self temporise];
    [obj_inietat afficheEtat: FTERRSP];
    [[message string: smessini] update];
    [self temporise];
    [[[[initiateur string: "F-TERMINATE conf"] center] centerV] update];
    [[[[initiateur1 string: sparam] center] centerV] update];
    [[[message string: "The FTAM agent receive the confirmation."] horFill: TRUE] update];
    delay(temps);
    [repondeur erase];
    [repondeur1 erase];
    [self animationEfface];
    [self temporise];
```

```
    [obj_fournisseur hide];
    [[message string: "The FTAM association is released."] update];
    delay(temps);
    [initiateur erase];
    [initiateur1 erase];
    [self temporise];
    [regime erase];
    [[message string: "This is the end of the FTAM association regime."] update];
    [self temporise];
    [[[message string: smess1] horFill: TRUE] update];
  }
  else
  {
  /* execution de l'erreur car la machine de protocole n'est pas dans un etat valide pour cette primitive */
    [[[message string: serreur] color: coulerreur] update];
    [message color: WHITE];
    printf("\007");
    [initiateur1  erase];
  };
  [obj_menu showAll];
  }

  - temporise
  {
  /* cette methode attend que l'utilisateur clique sur la souris pour se terminer */

    [[[[souris string: "Click any button."] center] centerV] update];
    while ([Mouse getNewButtons]!=0);
    while ([Mouse getNewButtons]==0);
    [souris erase];
  }

  - animationIniRep
  {
  /* cette methode anime le deplacement du PDU de l'initiateur vers le repondeur */

    int i;

    ligne1=[[[RMGLine superview: obj_fournisseur color: coulagent] relP1: 50: 78 length: 0: -28] show];

    ligne2=[[RMGLine superview: obj_fournisseur color: coulagent] relP1: 50: 50 length: 0: 0];
    for (i=0;i<420;i++) { [[ligne2 moveP2By: 1: 0] show]; delay(anitemps); };

    ligne3=[[[RMGLine superview: obj_fournisseur color: coulagent] relP1: 470: 50 length: 0: 28] show];
  }

  - animationRepIni
  {
  /*cette methode anime le deplacement du PDU du repondeur vers l'initiateur */

    int i;

    ligne1=[[[RMGLine superview: obj_fournisseur color: coulagent] relP1: 470: 50 length: 0: 28] show];

    ligne2=[[RMGLine superview: obj_fournisseur color: coulagent] relP1: 470: 50 length: 0: 0];
    for (i=0;i<420;i++) { [[ligne2 moveP2By: -1: 0] show]; delay(anitemps); };

    ligne3=[[[RMGLine superview: obj_fournisseur color: coulagent] relP1: 50: 50 length: 0: 28] show];
  }
```

```
- animationEfface
{
/* cette methode efface toute trace de l'animation relative au deplacement du PDU */

    [ligne1 erase];
    [ligne2 erase];
    [ligne3 erase];
    [ligne1 free];
    [ligne2 free];
    [ligne3 free];
```

```
#include "objc.h"
#include "rmg.h"
#include "envir.h"

/* definition des couleurs des objets */

#define coulfondetats GRAY7
#define coulcadre CYAN
#define coulligne CYAN

/* definition des etats */

#define IDLE 0
#define INITIALIZE_PENDING 1
#define TERMINATE_PENDING 2
#define INITIALIZED 3
#define NONE 4

/* definition des messages */

#define FINIRQ 0
#define FINIRSPPOS 1
#define FINIRSPNEG 2
#define FTERRQ 3
#define FTERRSP 4

/* definition du nombre d'etats suivants */

#define NBSUIVANT 2

/* definition du nombre de changements d'etats */

#define NBETATS 5

/* definition des denominations d'etats */

static char *etat[5]={ "Idle", "Initialize pending" , "Terminate pending" , "Initialized" , "none" };

@requires FontMngr, RMGString, RMGAction, RMGLine;

= Ftametat: Envir(corbugy,working,RMGVW,Primitive,Collection)
{

/* definition des objets de la scene */

id obj_previous, ligne_previous, obj_current, obj_next[NBSUIVANT], ligne_next[NBSUIVANT];

/* definition des indicateurs d etats */

int previous, current, next[NBSUIVANT];

/* definition des changements d'etats */

int chgtetat[NBETATS][3];
}
```

```
+ newIn: rootView
{
/* cette methode cree un objet Ftametat (methode obsolete) */

/* definition des polices de caracteres */

id nameFont;

nameFont=[FontMngr sysfont];

/* affichage des objets de la scene */

self=[[[self origin: crossx: crossy extent: 500: 500 superview: rootView bkgd: PASTELBLUE]
            frameWidth: 3] frameColor: CYAN];

[[[[[[[RMGString relative: 0: 475 extent: 500: 25 superview: self bkgd: PASTELBLUE]
                font: nameFont]
                color: BLACK]
                string: "Initiator states"]
                horFill: TRUE]
                center]
                centerV];

/* etat precedent */
obj_previous = [[[RMGString relative: 165: 400 extent: 160: 30 superview: self bkgd: WHITE]
                            frameWidth: 3] frameColor: RED];
[[obj_previous font: nameFont] color: RED];

/* etat courant */
obj_current =  [[[RMGString relative: 165: 235 extent: 160: 30 superview: self bkgd: PERIWINKLE]
                            frameWidth: 3] frameColor: RED];
[[obj_current font: nameFont] color: RED];

/* etats suivants */
obj_next[0] = [[[RMGString relative: 60: 30 extent: 160: 30 superview: self bkgd: WHITE]
                           frameWidth: 3] frameColor: RED];
[[obj_next[0] font: nameFont] color: BLACK];
obj_next[1] = [[[RMGString relative: 280: 30 extent: 160: 30 superview: self bkgd: WHITE]
                           frameWidth: 3] frameColor: RED];
[[obj_next[1] font: nameFont] color: BLACK];

return self;
}

- extraNewIn: rootView
{
/* cette methode initialise les objets de la scene */

[self showAll];

/* initialisation des variables */

previous = NONE;
current = IDLE;
next[0] = INITIALIZE_PENDING;
next[1] = NONE;

/* definition des changements d'etats */

chgtetat[0][0]=IDLE;
chgtetat[0][1]=FINIRQ;
chgtetat[0][2]=INITIALIZE_PENDING;
chgtetat[1][0]=INITIALIZE_PENDING;
chgtetat[1][1]=FINIRSPPOS;
chgtetat[1][2]=INITIALIZED;
chgtetat[2][0]=INITIALIZE_PENDING;
```

```
chgtetat[2][1]=FINIRSPNEG;
chgtetat[2][2]=IDLE;
chgtetat[3][0]=INITIALIZED;
chgtetat[3][1]=FTERRQ;
chgtetat[3][2]=TERMINATE_PENDING;
chgtetat[4][0]=TERMINATE_PENDING;
chgtetat[4][1]=FTERRSP;
chgtetat[4][2]=IDLE;

[self afficheEtat: FTERRQ];
/* delay(10000);
[self afficheEtat: FINIRQ];
delay(10000);
[self afficheEtat: FINIRSPPOS];
delay(10000);
[self afficheEtat: FTERRQ];
delay(10000);
[self afficheEtat: FTERRSP];*/

return self;
}

+ creer: (id)rootView: (int)posx: (int)posy: (char*)titre
{
/* cette methode permet de creer un objet Ftametat a la position posx,posy relative a l'objet qui demande la creation.     */
/* titre contient la chaine de caractere affichee dans l'objet Ftametat pour l'identifier                                 */

/* definition des polices de caracteres */

id nameFont;

nameFont=[FontMngr sysfont];

/* affichage des objets de la scene */

self=[[[self relative: posx: posy extent: 500: 300 superview: rootView bkgd: coulfondetats]
             frameWidth: 3] frameColor: CYAN];

[[[[[[[RMGString relative: 0: 275 extent: 500: 25 superview: self bkgd: coulfondetats]
                  font: nameFont]
                  color: BLACK]
                  string: titre]
                  horFill: TRUE]
                  center]
                  centerV];

/* etat precedent */
obj_previous = [[[RMGString relative: 165: 210 extent: 160: 30 superview: self bkgd: WHITE]
                            frameWidth: 3] frameColor: coulcadre];
[[obj_previous font: nameFont] color: BLACK];
ligne_previous = [[[RMGLine superview: self color: coulligne] relP1: 245: 150 length: 0: 60] show];

/* etat courant */
obj_current =  [[[RMGString relative: 165: 120 extent: 160: 30 superview: self bkgd: GRAY12]
                            frameWidth: 3] frameColor: coulcadre];
[[obj_current font: nameFont] color: YELLOW];

/* etats suivants */
obj_next[0] = [[[RMGString relative: 60: 30 extent: 160: 30 superview: self bkgd: WHITE]
                           frameWidth: 3] frameColor: coulcadre];
[[obj_next[0] font: nameFont] color: BLACK];
ligne_next[0] = [[[RMGLine superview: self color: coulligne] relP1: 245: 120 length: -105: -60] show];
obj_next[1] = [[[RMGString relative: 280: 30 extent: 160: 30 superview: self bkgd: WHITE] frameWidth: 3]
                           frameColor: coulcadre];
[[obj_next[1] font: nameFont] color: BLACK];
ligne_next[1] = [[[RMGLine superview: self color: coulligne] relP1: 245: 120 length: 105: -60] show];
```

```
    /* initialisation des variables */

    previous = NONE;
    current = IDLE;
    next[0] = INITIALIZE_PENDING;
    next[1] = NONE;

    /* definition des changements d'etats */

    chgtetat[0][0]=IDLE;
    chgtetat[0][1]=FINIRQ;
    chgtetat[0][2]=INITIALIZE_PENDING;
    chgtetat[1][0]=INITIALIZE_PENDING;
    chgtetat[1][1]=FINIRSPPOS;
    chgtetat[1][2]=INITIALIZED;
    chgtetat[2][0]=INITIALIZE_PENDING;
    chgtetat[2][1]=FINIRSPNEG;
    chgtetat[2][2]=IDLE;
    chgtetat[3][0]=INITIALIZED;
    chgtetat[3][1]=FTERRQ;
    chgtetat[3][2]=TERMINATE_PENDING;
    chgtetat[4][0]=TERMINATE_PENDING;
    chgtetat[4][1]=FTERRSP;
    chgtetat[4][2]=IDLE;

    [self initEtat];

    return self;
    }

    - initEtat
    {
    /* cette methode permet l'initialisation de l'objet ftametat */
      [self afficheEtat: FTERRQ];
    }

    - (BOOL) afficheEtat: (int)message
    {
    /* cette methode verifie que pour la primitive contenue dans message et pour l'etat courant de la machine de protocole */
    /* il existe un etat suivant valide. Si c'est le cas, les etats sont mis a jour et la valeur retournee est TRUE.       */
    /* Sinon, les etats sont inchanges et la valeur retournee est FALSE.                                                    */

    int i,j,k;
    BOOL temp=FALSE;

    i = 0;
    j = 0;
    k = 0;

    /* recherche d'un etat suivant valide */

    while ((i<NBETATS) && (!temp))
    {
     if ((chgtetat[i][0] == current) && (chgtetat[i][1] == message))
     {
      previous = current;
      current = chgtetat[i][2];
      while (j<NBETATS)
      {
       if (chgtetat[j][0] == current) next[k++] = chgtetat[j][2];
       j++;
      };
      while (k<NBSUIVANT) next[k++] = NONE;
      temp=TRUE;
     };
     i++;
    };

    /* si temp=TRUE alors, il existe un etat valide, on met a jour les etats */
```

```
    if (temp) for (i=1;i<=10;i++)
     {
      /* clignotement de l'objet */
      [self frameColor : BLACK];
      delay(100);
      [self frameColor : CYAN];
      delay(100);
     };

    if (previous != NONE)
     {
      /* mise a jour de l'etat precedent */
      [[[[[obj_previous string: etat[previous]] horFill: TRUE] center] centerV] update] showAll];
      [[ligne_previous update] showAll];
     }
    else
     {
      [obj_previous hide];
      [ligne_previous hide];
     };

    /* mise a jour de l'etat courant */
    [[[[obj_current string: etat[current]] horFill: TRUE] center] centerV] update];

    /* mise a jour des etats suivants */
    i = 0;
    while (i<NBSUIVANT)
    {
     if (next[i] != NONE)
        {
         [[[[[obj_next[i] string: etat[next[i]]] horFill: TRUE] center] centerV] update] showAll];
         [[ligne_next[i] update] showAll];
        }
     else
        {
         [obj_next[i] hide];
         [ligne_next[i] hide];
        };
     i++;
    };

    return temp;
    }
    =:
```