LISI
Laboratoire d'Informatique Scientifique et Industrielle
Ecole Nationale Supérieure de Mécanique et d'Aérotechnique
Université de Poitiers
Téléport 2 - BP 109
86960 FUTUROSCOPE Cedex

Validation a priori de tâches utilisateurs dans une IHM

Y. AIT-AMEUR
yamine@ensma.fr

LISI/ENSMA-Université de Poitiers

LISI

Plan

- I. Introduction
- II. Notation : architecture et besoins utilisateurs
- III. Validation de tâches par traces : approche explicite
- IV. Validation de tâches à partir d ’un modèle : approche implicite
- V. Conclusion et perspectives

LISI

UNIVERSITÉ de Poitiers

Plan

- I. Introduction

Introduction

- Présentes dans tous les logiciels.
- Longtemps délaissées au profit des applications.
- Constante progression.
- Boîtes à outils.
- Intervenants divers et plusieurs métiers : ergonomie, psychologie, informatique, sécurité, etc.

Introduction

- Constructeurs d'interfaces
  - Outils d'aide à la conception
  - Absence : architecture, besoins utilisateurs
  - Visual C++, SWING, OSF-Motif)
- Systèmes basés sur modèles.
  - Ensemble de modèles (modèle de tâches…)
  - Langages spécifiques à chaque modèle
  - Description de l'interface, puis génération.
- Notations
  - de description de tâches : centrées utilisateurs.
  - de description modèles d'architecture.
  - Développements empiriques à partir de ces notations

Introduction

- Approches formelles recensées (notre classification)
  - Orientées modèles.
    - Réseaux de Petri, ATN...
  - Vérification automatique sur modèle.
    - SMV, LUSTRE, ESTEREL...
  - Algébriques.
    - Lotos, ...
  - Approche logique.
    - HOL ...
  - Développement incrémental.
    - Z et les interacteurs de York ...

LISI

UNIVERSITÉ de Poitiers

Plan

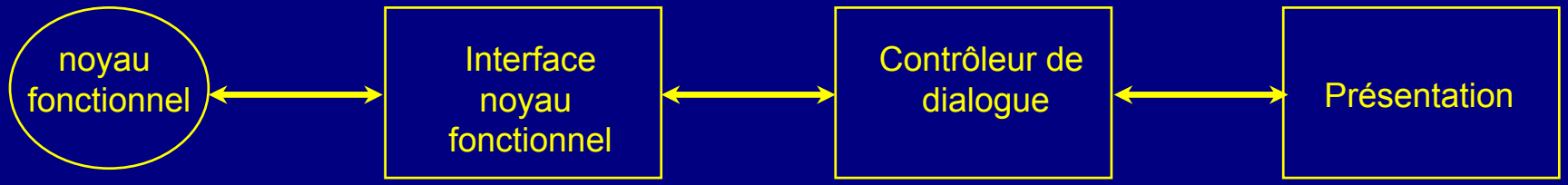
- II. Notation : architecture et besoins utilisateurs

Notations
Modèle du système

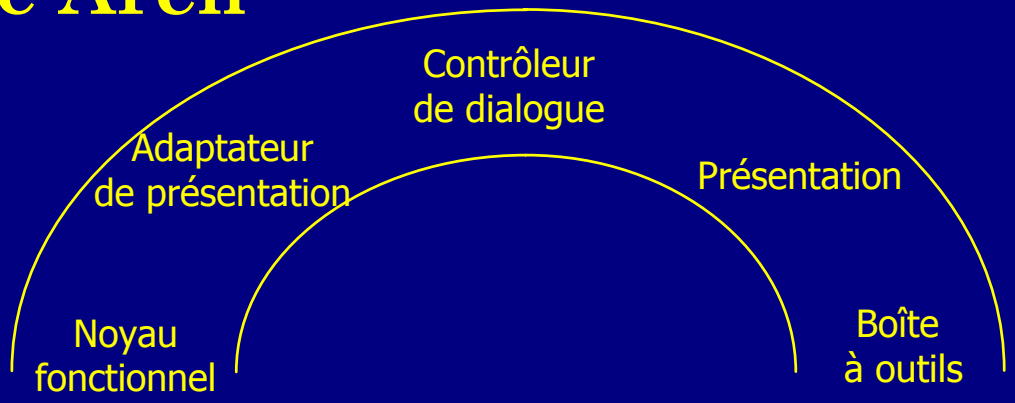
- **Description de l'architecture du système**

- **Modèles d'architecture**
  - Séparation du noyau fonctionnel et de la présentation.
  - Lien effectué par le contrôleur du dialogue
  - Exemples : ARCH, Seeheim, PAC, H4, ...

Modèle de Seeheim



Modèle Arch



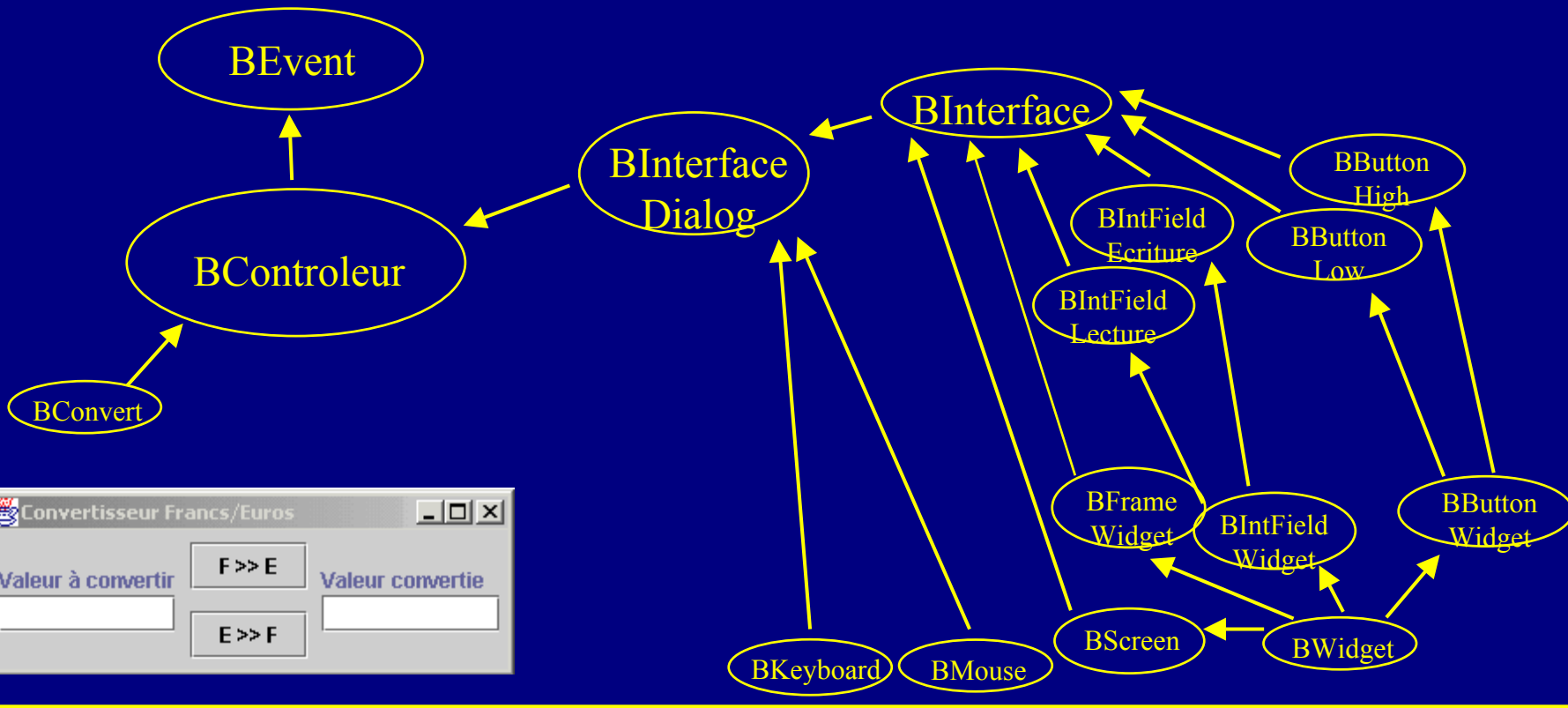
❖ Utilisation du modèle d'architecture : ARCH

- Description par modules et liens entre modules.
- Disposition des descriptions B dans le le modèle ARCH.

❖ B et les machines abstraites.

- Représentation des éléments du noyau fonctionnel
- Rétro-conception des éléments réutilisés : ex. boîtes à outils.
- Exemples de machines.
  - Gestion de la souris, du muti-fenêtrage, de la manipulation directe, etc...

Décomposition de l'application suivant ARCH



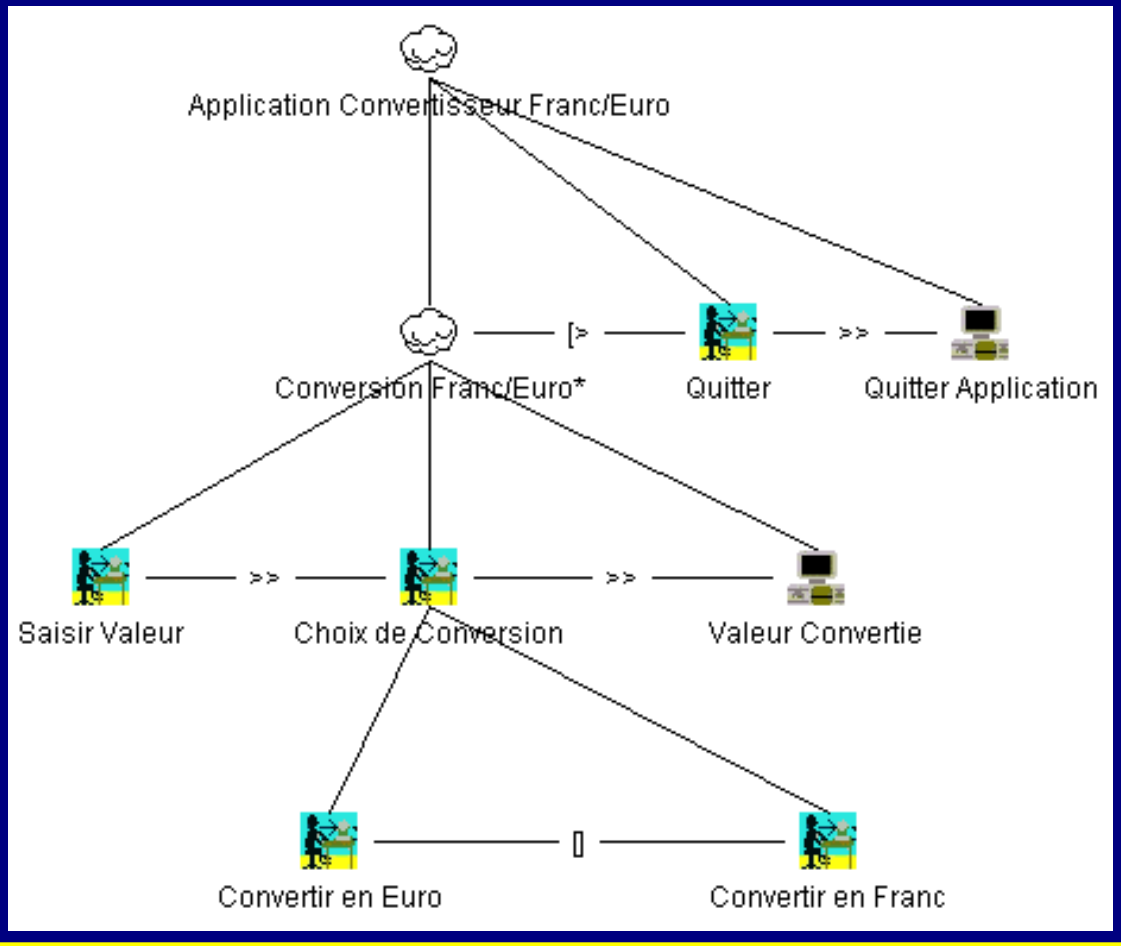
Notations
Les besoins utilisateurs

- **Description de l'application du point de vue de l'utilisateur**
  - Toutes les notations se fondent sur la décomposition d'une tâche de haut niveau en plusieurs sous-tâches elles mêmes décomposées, ...
- **Modèles de tâches**
  - Description des séquences d'événements observés
  - Langages d'expression de tâches utilisateurs ou de scénarii d'utilisation.
    - Sémantique vaguement définie.
    - MAD, UAN, X-UAN, CTT, ...

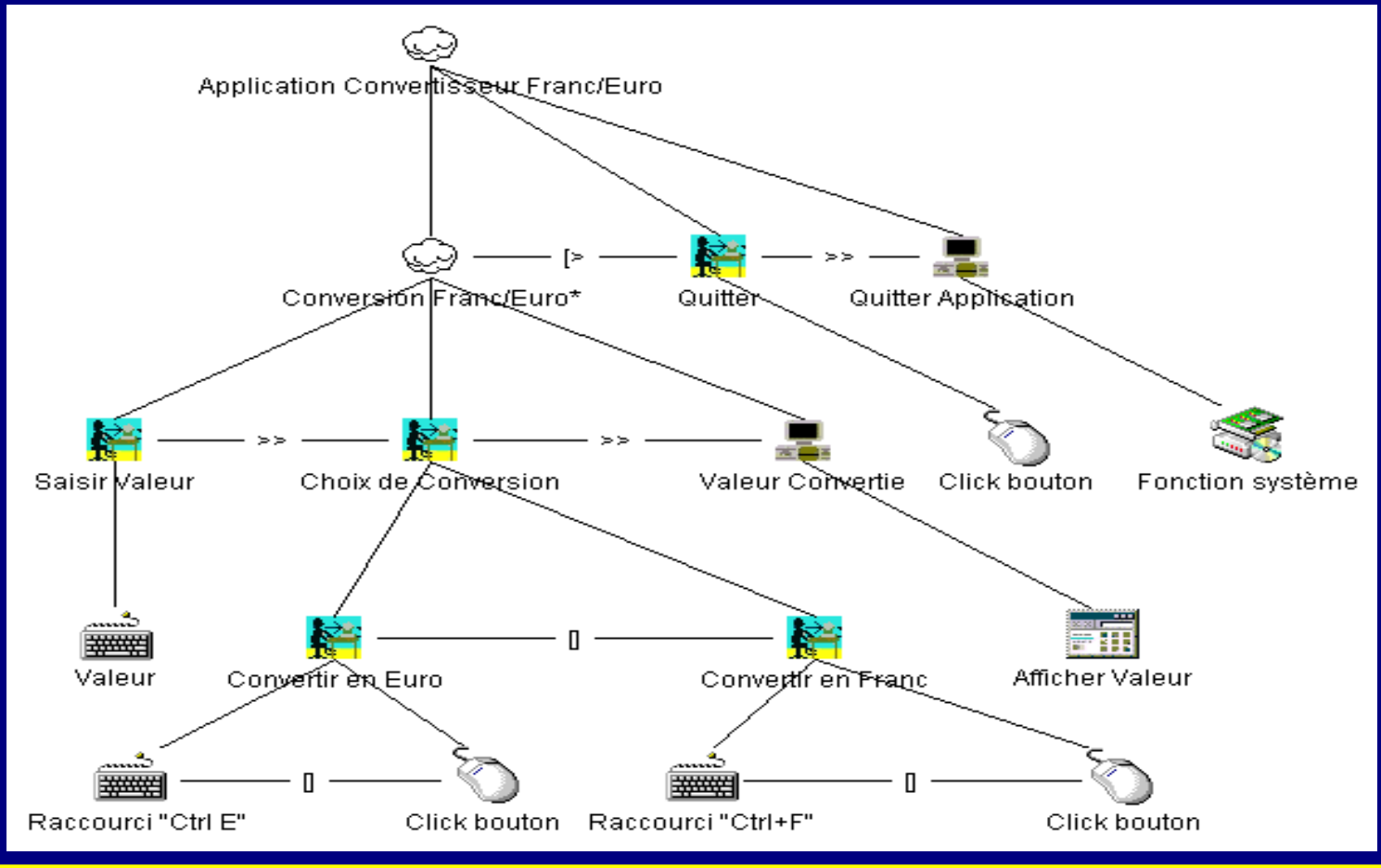
Notations
Les besoins utilisateurs
CTT Concurrent Task Tree

- **Représentation graphique**
- **Diversité des tâches**
  - utilisateur 
  - application 
  - interaction 
  - abstraite 
  - Sémantique et outils proposés fondés sur LOTOS.

Notations
Les besoins utilisateurs
Exemple CTT



Notations
Les besoins utilisateurs
Exemple détaillé CTT



LISI

UNIVERSITÉ de Poitiers

Plan

- III. Validation de tâches par traces : approche explicite

Approche fondée sur les traces explicites

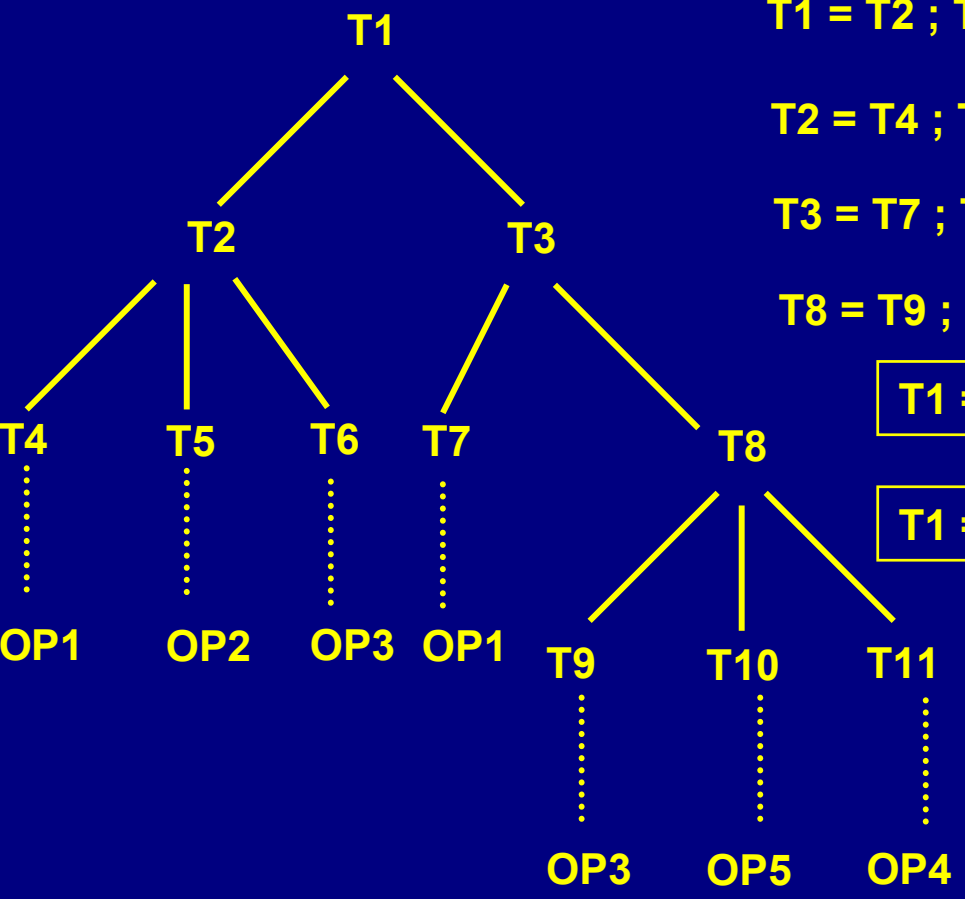
Utilisation de B

- Ensemble de variables d 'état
- Initialisation de l 'état
- Opérations modifient les états
- Invariant préservé par l'initialisation et par les opérations.
- Raffinement : introduction de détails
  - Nouvelles variables d 'état.
  - Nouvelles opérations.
  - Invariant de collage.
- Principe de preuve : substitutions généralisées et plus faible précondition [S]Q
  - Décharger les OP générées
  - Utilisation du prouveur
  - Pas d 'explosion combinatoire
  - Les preuves sont diffusées dans les raffinement
    => Simplification du processus de preuve.

Approche fondée sur les traces explicites

- **Première approche**
- **Diagramme de séquence de UML**
- **Représentation avec B**
  - La tâche de plus haut niveau est représentée dans une machine abstraite.
    - État de départ.
    - État final.
    - Propriétés.
  - Le raffinement consiste à introduire la séquence.

Approche fondée sur les traces explicites



T1 = T2 ; T3

T2 = T4 ; T5 ; T6

T3 = T7 ; T8

T8 = T9 ; T10 ; T11

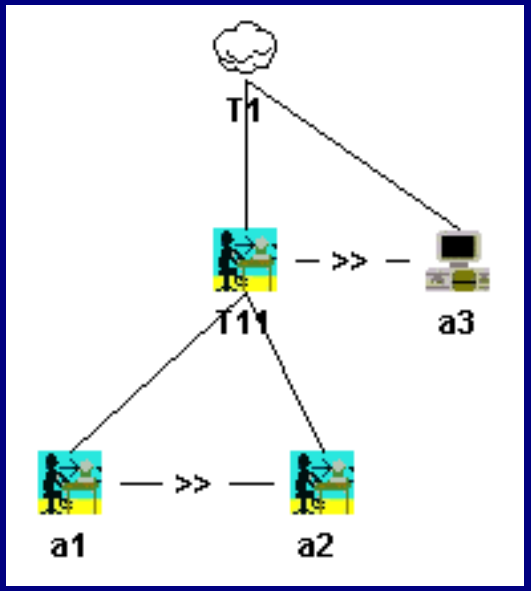
T1 = T4 ; T5 ; T6 ; T7 ; T9 ; T10; T11

T1 = OP1 ; OP2 ; OP3 ; OP1 ; OP3 ; OP5; OP4

Approche fondée sur les traces explicites


```
MACHINE Task_T1
INVARIANT
…
ASSERTIONS
…
 op_T1 =
  SELECT …
  THEN …
 END
END
```

```
 MACHINE Task_T11
 …
  op_T11 =
   SELECT …
   THEN …
  END
 END
```




```
REFINEMENT Ref1_Task_T1
EXTENDS Architecture, Task_T11
…
 op_T1 =
  SELECT …
  THEN
    op_T11; op_a3
 END
END
```

```
REFINEMENT Ref2_Task_T1
EXTENDS Architecture, Task_T11
…
 op_T1 =
  SELECT …
  THEN
    op_a1; op_a2; op_a3
 END
END
```

Approche fondée sur les traces explicites

- **Avantages**
  - Représentation de tous les diagrammes séquences finies.
  - Validation a priori.
    - Validation de tâches et faisabilité.
    - Validation de l ’architecture (complétude).
  - Preuve de propriétés sur les tâches (par des invariants ou des assertions).
  - Utilisation de B classique.

Approche fondée sur les traces explicites

- **Inconvénients**
  - On ne représente que la séquence de tâches.
  - Revient à faire du model checking à la main.
  - La décomposition est réalisée par l ’utilisateur.
  - Difficulté de représentation de l ’itération.
  - Preuve des propriétés à chaque étape.
  - Représentation par les traces de
    - L ’entrelacement,
    - L ’interruption,
    - La désactivation.

Plan

- IV. Validation de tâches à partir d ’un modèle : approche implicite

Approche fondée sur les traces implicites

- **La description du système reste inchangée.**
- **Utilisation d ’un langage de haut niveau pour l ’expression des tâches**
  - Approche implicite.
  - Le langage CTT.

Approche fondée sur les traces implicites

Syntaxe du langage

T ::=	T >> T	|	-- Activation, séquence
	T [] T	|	-- Choix
	T || T	|	-- Concurrence
	T |=| T	|	-- Ordre indépendant
	[T]	|	-- Optionnel
	T |> T	|	-- Interruption
	T [> T	|	-- Désactivation
	T^{*} [> T	|	-- Désactivation de tâche itérative
	T^{N}	|	-- Tâche itérative
	T_A		-- Tâche atomique

Représentation par des modèles

- Proche du traitement des langages.
- Travaux sur les programmes séquentiels.
- Enrichissement par des propriétés.

Approche fondée sur les traces implicites

Utilisation de B événementiel

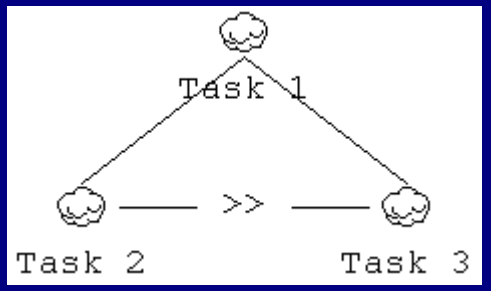
- Ensemble de variables d 'état
- Initialisation de l 'état
- Évènements instantanés modifient les états
  - Les événements sont gardés
  - Faisabilité des événements est prouvé par l'absence de blocage
  - Vivacité du système établie par la définition de variants.
- Les événements sont déclenchés lorsque leur gardes sont vraies.
- Invariant préservé par l'initialisation et par les événements.
- Raffinement : introduction de détails
  - Nouvelles variables d 'état.
  - Nouveaux événements.
  - Invariant de collage.

Approche fondée sur les traces implicites

Représentation en B événementiel

- Principe : chaque règle de production est un raffinement.
- T ::= T1 op T2 est décomposée en
  - un événement T.
  - un raffinement avec les événements T, T1 et T2.
  - Les événements T1 et T2 « travaillent » pour T
  - Introduction des variants.

◆ Activation ou séquence



T :: = T 1 >> T2 | -- Activation, séquence

INITIALISATION

B:= 2 ;

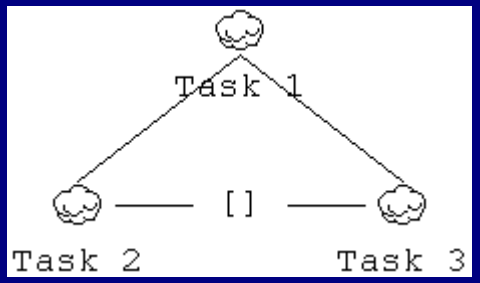
Evt_T1 = SELECT
G1 ∧ B = 2
THEN
S1 ||
B := 1
END ;

Evt_T2 = SELECT
G2 ∧ B = 1
THEN
S2 ||
B := 0
END ;

Approche fondée sur les traces implicites

◆ Choix

T :: = T1 [] T 2 | -- Choix



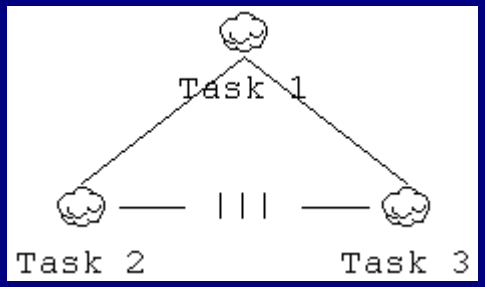

```
INITIALISATION
   ANY
          N
   WHERE
          N ∈ {1, 2}
   THEN
          Choix:= N ;
    END;
```

```
Evt_T1 = SELECT
                G 1 ∧ Choix = 1
          THEN
                S1 ||
                Choix := 0
          END ;
```

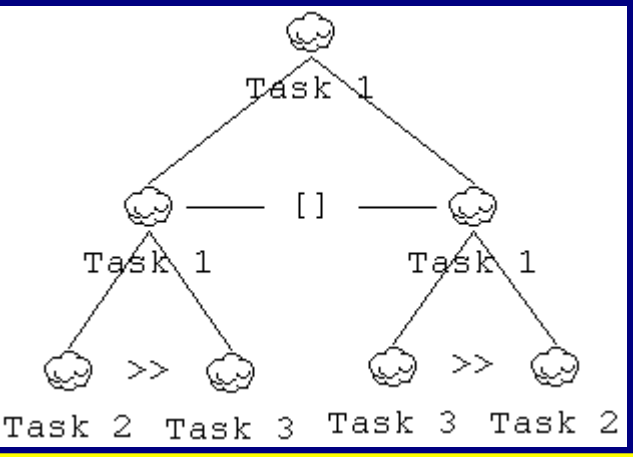
```
Evt_T2 = SELECT
                G2 ∧ Choix = 2
          THEN
                S2 ||
                Choix := 0
          END ;
```

Approche fondée sur les traces implicites

Concurrence



T :: = T1 || T2 | -- Concurrence

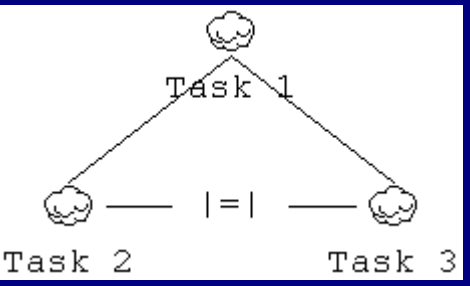



```
Evt_T1 = SELECT G1
         THEN S1
         END ;
```

```
Evt_T2 = SELECT G2
         THEN S2
         END ;
```

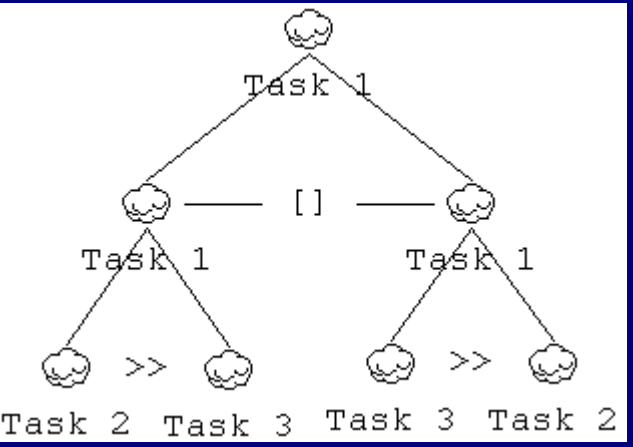
Approche fondée sur les traces implicites

◆ Ordre indépendant



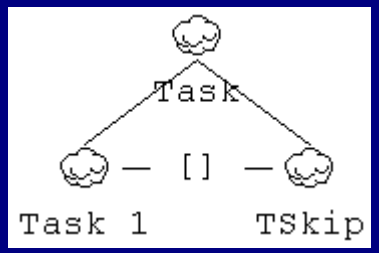
T :: = T1 |=| T 2 | -- Ordre indépendant

T :: = (T1 >> T2) [] (T2 >> T1)



◆ Tâche optionnelle

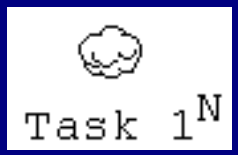
T :: = [T1] | -- Optionnel



$T = T1 [] T_{SKIP}$

◆ Tâche itérative

T :: = T1 ^{N} | -- Tâche itérative




```
INITIALISATION
 ANY
        P
  WHERE
        P ∈ NAT
  THEN
        I:= P
  END;
```

```
Evt_Loop = SELECT
                I > 0
        THEN
                S1  ||
                I:= I - 1
        END ;
```

```
Evt_End = SELECT
                G3 ∧ I = 0
        THEN
                Skip
        END ;
```

Approche fondée sur les traces implicites

Désactivation de tâche T1 [> T2

- La tâche T1 est désactivée par la tâche T2.
- T1 ne reprend pas son exécution.
- La désactivation intervient dans les états observables.

Approche fondée sur les traces implicites

Deux approches

- T1 est une tâche atomique ou bien non raffinée

  T1 [> T2 devient T1 [ ] T2

- T1 n 'est pas atomique

  $T_{1,1}$ Op_1 $T_{1,2}$ Op_2 … Op_n $T_{1,n+1}$ [> T2 devient

  T2 []

  $T_{1,1}$ >> T2 []

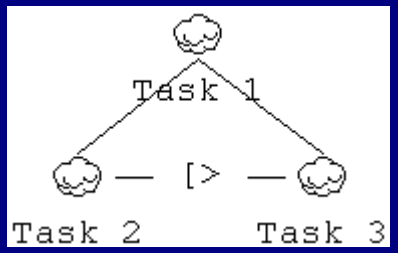
  $T_{1,1}$ Op_1 $T_{1,2}$ >> T2 []

  …

  $T_{1,1}$ Op_1 $T_{1,2}$ Op_2 … Op_n $T_{1,n+1}$

- Connaissance de l 'arbre.
- Définition de tous les parcours.

◆ Désactivation de tâche.



T :: = T1 [> T2 | -- Désactivation

INITIALISATION

Desact := 3 ;

```
vt_T1 = SELECT
          G1 ∧ desact = 2
        THEN
          S1 ||
           desact := 1 ou 0
        END ;
```

```
Evt_T2 = SELECT
           G2 ∧ desact = 1
         THEN
           S2 ||
           desact := 0
         END ;
```

```
Evt_desactivation = ANY
                    P
          WHERE
                    P ∈ {1,2} ∧
                    Not(desact = 0)
          THEN
                    desact :=  P ||
                    Réparation !!!!
           END;
```

Approche fondée sur les traces implicites

- **Interruption de tâche T1 |> T2**
  - La tâche T1 est interrompue par la tâche T2.
  - T1 peut reprendre son exécution.
  - T2 peut être exécutée un nombre arbitraire de fois.
  - L ’interruption intervient dans les états observables.

Approche fondée sur les traces implicites

Deux approches

- T1 est une tâche atomique ou bien non raffinée

  T1 |> T2 devient $T2^N$ >> T1 (N peut être Nul)

- T1 n 'est pas atomique

  $T_{1,1}$ Op_1 $T_{1,2}$ Op_2 … Op_n $T_{1,n+1}$ |> T2 devient

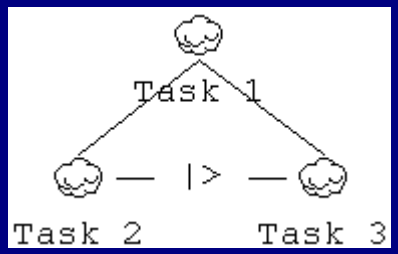
$T2^N$ >> $T_{1,1}$ >> Op_1 >> $T2^M$ >> $T_{1,2}$ >> $T2^K$ >> Op_2 … >> $T2^P$ >> Op_n $T_{1,n+1}$

- Connaissance de l 'arbre.
- Définition de tous les parcours.

Approche fondée sur les traces implicites

Interruption de tâche

T :: = T1 |> T2 -- Interruption



INITIALISATION
Interrupt := 2 ;

```
vt_T1 = SELECT
        G1 ∧
        Interrupt = 2
      THEN
        S1 ||
        Interrupt := 0
      END
```

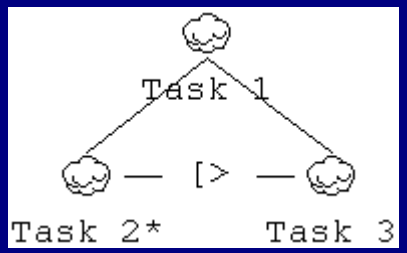
```
Evt_Loop_T2 = SELECT
        G2 ∧ I >0 N
      THEN
        S2 ||
        I := I - 1
      END
```

```
Evt_End_T2 = SELECT
        G3 ∧ I = 0
      THEN
        interrupt :=2
      END
```

```
Evt_init_loop_T2 =
   ANY
        P
   WHERE
        P ∈ NAT ∧
        not (interrupt =
   THEN
        I:=P ||
        interrupt := 1
    END
```

◆ Désactivation de tâche itérative

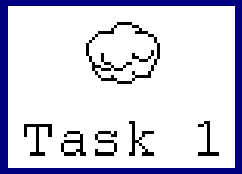
T :: = T1 * [> T2 | -- Désactivation de tâche itérative



❖ $T1^{*}$ [> T2 devient

$T1^{N}$ [> T2

◆ Tâche atomique

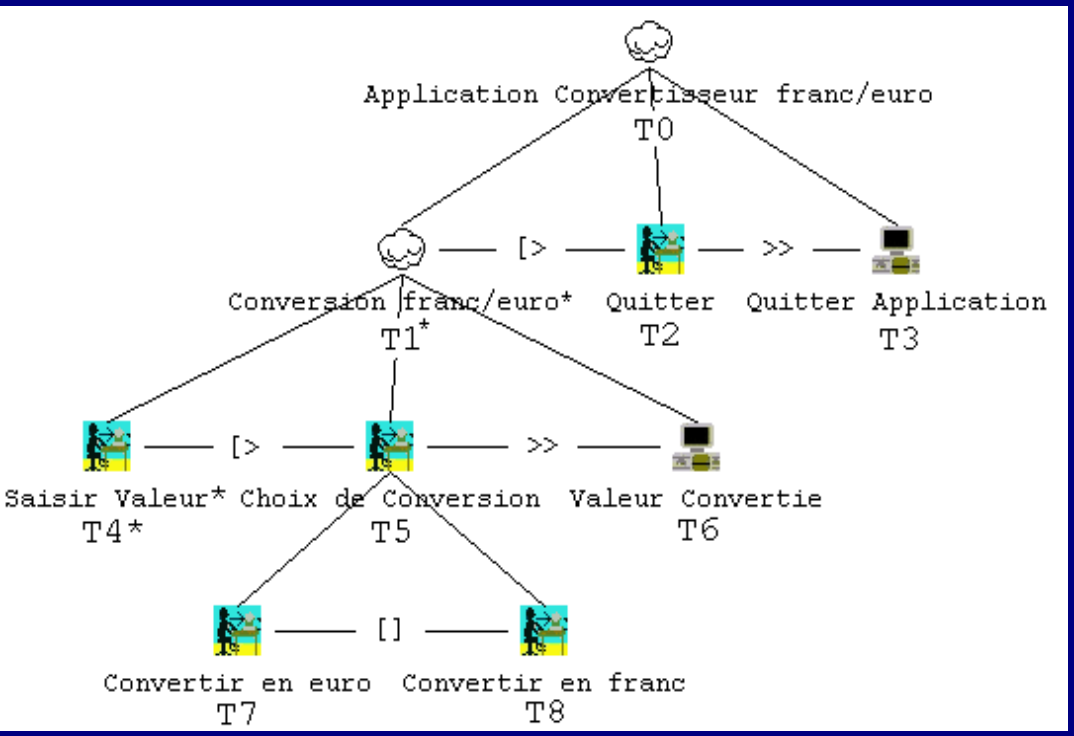


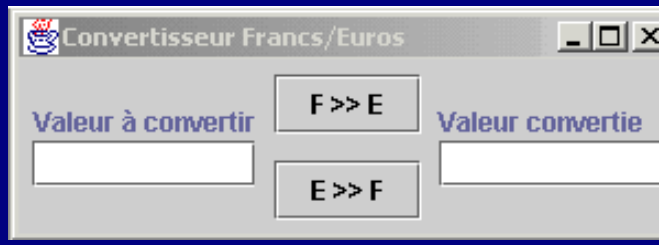

```
T  :: =     T_A                    -- Tâche atomique

Evt_T1 = SELECT
                    GA
          THEN
                    SA
          END ;
```

❖ SA correspond à un événement ou une action du contrôleur de dialogue dans le modèle d ’architecture ou modèle du système.

Approche fondée sur les traces implicites Un exemple





T0 = T1* [> T2 >> T3

T1 = T4 * [> T5 >> T6

T4 = T7 [ ] T8

Approche fondée sur les traces implicites Un exemple


```
MODEL M1


EVENTS

Evt_T0 = BEGIN
                    S0
         END
END
```

LISI
Université de Poitiers

Approche fondée sur les traces implicites Un exemple


```
MODEL M2
 REFINES M1
..
NITIALISATIOn
   BT0 := 3
..

EVENTS
Evt_T0 = SELECT
                    BT0=0
          THEN
          Mise à jour des variables de
          l 'abstraction ou skip

          END;

Evt_init_T1 = ANY
                    p
                WHERE
                    p ∈ NAT ∧ BT0 = 3
                THEN
                    I1 := p
                END;
END;
```

$T0 = T1^{*} [> T2 >> T3$

```
Evt_loop_T1 = SELECT
                  G1 ∧ I1> 0
                      ∧ BT0 = 3
              THEN
                S1 ||
                I1:= I1 - 1
              END;

Evt_T1 = SELECT
                  G1 ∧ I1= 0
                      ∧ BT0 = 3
              THEN
                  BT0 := 2
              END;

END;
```

```
Evt_T2 = SELECT
              G2 ∧ BT0=2
             THEN
               evt_click_Quit_button ||
                        BT0:=1
             END;
END;

Evt_T3 = SELECT
                G3 ∧ BT0=1
             THEN
                evt_click_Quit_application ||
               BT0:=0
             END;
END;
```

Approche fondée sur les traces implicites Un exemple


```
MODEL M2
  REFINES M2
...
INITIALISATION
BT1 := 3
...
EVENTS

Evt_T1 = SELECT
                BT1=0
            THEN
            Mise à jour des variables
            de l 'abstraction ou skip
            END ;

Evt_init_T4 = ANY
                 p
               WHERE
                 p ∈ NAT ∧ BT1 = 3
               THEN
                 I4 := p
               END
END ;
```

T1 = T4 * [> T5 >> T6

```
Evt_loop_T4 = SELECT
                G4 ∧ I4 > 0 ∧
                        BT1 = 3
            THEN
               event_Input_Value ||
               I4 := I4 - 1
            END ;

Evt_T4 = SELECT
                G4 ∧ I4 = 0 ∧
                        BT1 = 3
              THEN
                  BT1 := 2
              END

END ;
```

```
Evt_T5 = SELECT
              G5 ∧ BT1=2
            THEN
               evt_click_Quit_button ||
               BT1 := 1
            END
END ;

Evt_T6 = SELECT
              G6 ∧ BT1 =1
            THEN
               evt_click_Quit_application |
               BT1 := 0
            END
END ;

Evt_Desac_Raf_T1 = SELECT
                        BT1 ∈ {1, 2,3}
                    THEN
                        BT1 : =  0 ||
                        BT0 :=   2 ||
                        Réparation ????
                    END ;
```

Approche fondée sur les traces implicites Un exemple


```
MODEL M2
  REFINES M2
...
INITIALISATION
 BT4 := 3
...
EVENTS

Evt_T4 = SELECT
                BT4=0
            THEN
            Mise à jour des variables de
            l 'abstraction ou skip
            END;

Evt_init_T4 = ANY
                      p
                WHERE
                      p ∈ {1,2}
                THEN
                      BT4 := p
                END
END;
```

T4 = T7 [ ] T8

```
Evt_T7 = SELECT
                G7 ∧  BT4=2
            THEN
              Evt_Click_Euro_Button ||
                      BT4 := 0
            END;
END;

Evt_T8= SELECT
                G8 ∧ BT4 =1
            THEN
             Evt_Click_Franc_Button ||
              BT4 := 0
            END;
END;

Evt_Desac_Raf_T4 = SELECT
                      BT4 ∈ {1, 2,3}
                  THEN
                      BT4 : =  0 ||
                      BT1 :=   2 ||
                      Réparation ???
                  END;
```

Approche fondée sur les traces implicites

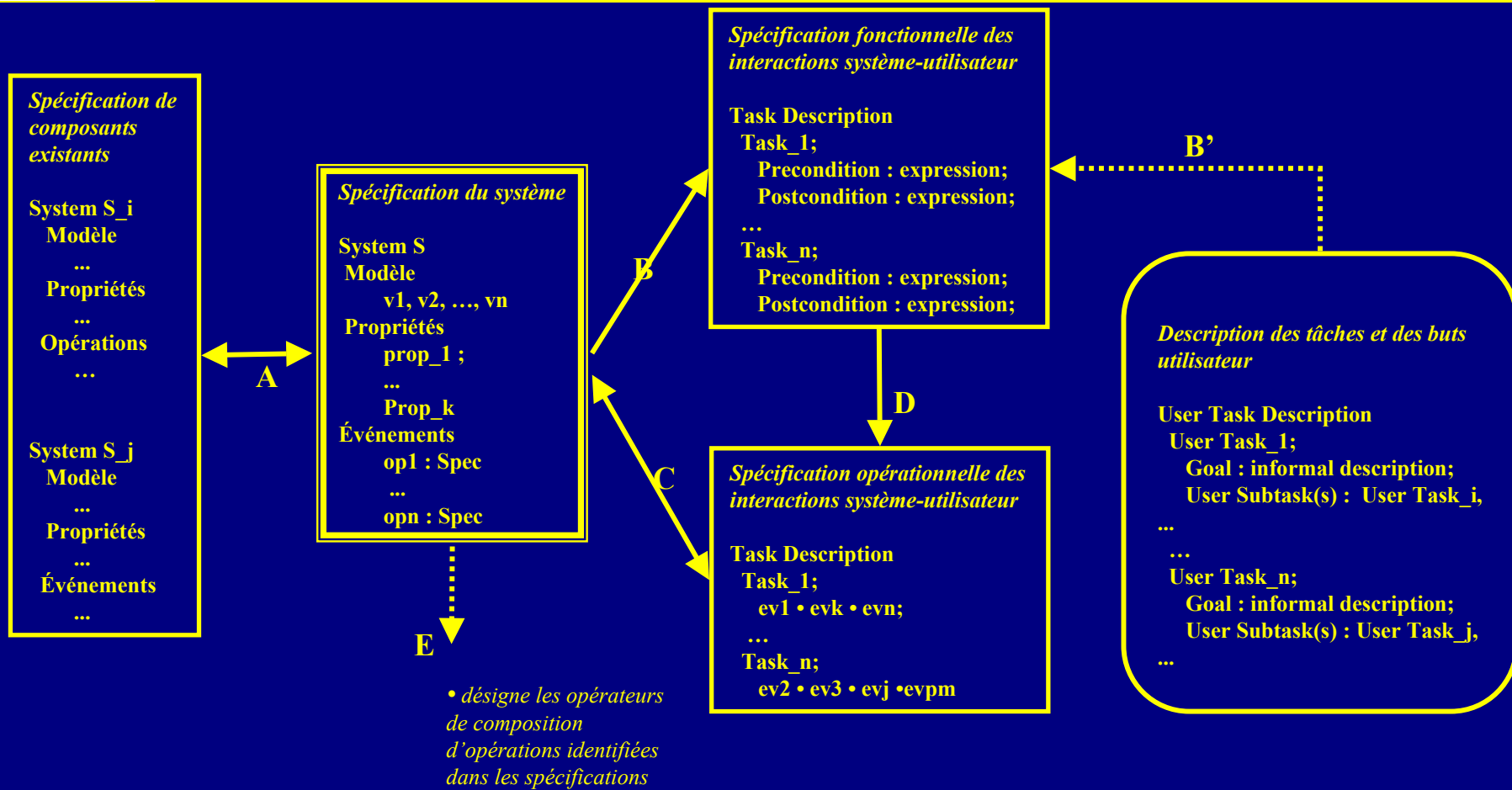
- **Avantages**
  - Expression de plusieurs traces sans énumération.
  - Support avec un outil de preuve.
  - Évite les problèmes de model checking notamment grâce à la clause ANY.
  - Représentation de boucles, des interruptions et désactivations.
  - Preuve de propriétés sir les tâches grâce aux clauses invariant et assertions.

Conclusion et Perspectives

- Représentation et prise en compte amont des notations centrées utilisateurs dans les IHM de type WIMP.
- Validation a priori de modèles de tâches.
- Validation a priori de l'architecture (faisabilité).
- Approche fondée sur la preuve.
- Évitement de l'énumération et du test qui représentent les pratiques courantes.
- Plus généralement, cette approche pourrait être étendue aux traitements de certaines BNF

E ::= F devient F raffine E

Conclusion et Perspectives



Conclusion et Perspectives

Perspectives

- Étude du raffinement de la désactivation et de l'interruption.
- Propriétés ergonomiques issues des recommandations.
  - Nombre de clicks, nombre de sous-menus
    - dans le système et dans les tâches
- Expressions de chemins multiples.
- Génération de tests à partir des descriptions.
- Scénarii non nominaux.
- Étude de la multi-modalité

❖ Projet RNRT Verbatim.

❖ Etude ALIDAF en cours de labellisation