2 • SPÉCIFICATION SELON LA MÉTHODE SA-RT

2.1 Introduction générale à la méthode SA-RT

La méthode SA-RT est une méthode d'analyse fonctionnelle et opérationnelle des applications de contrôle-commande. Cette méthode permet de réaliser une description graphique et textuelle de l'application en termes de besoins, c'est-à-dire de « ce que l'on a à faire » ou le « quoi » (*What* ?). Cette mise en forme du « cahier des charges » de l'application est formelle dans le sens où la méthodologie (ensemble des documents à élaborer) et l'expression (syntaxe graphique) sont définies. En revanche, elle ne permet pas d'effectuer une vérification de propriétés de l'application à partir des seules descriptions SA-RT. Des études ont été menées pour associer à la méthode SA-RT des méthodes formelles afin d'apporter des possibilités de simulation et de vérification. Une de ces méthodes est présentée à la fin du chapitre. Aucune règle officielle ou normalisation n'a été mise en place pour la méthode SA-RT et son utilisation. Par conséquent, il existe de nombreuses mises en œuvre de la méthode SA-RT avec des différences plus ou moins importantes et aussi des extensions spécifiques de la méthode. Ceci fera l'objet du dernier point traité dans ce chapitre. Nous allons nous attacher à décrire la méthode SA-RT la plus générale et la plus usitée, et correspondant à la méthodologie de développement d'une application de contrôle-commande qui est l'objectif de cet ouvrage.

L'accroissement très important de la taille des logiciels développés dans les années 70 a conduit à mettre en place des méthodes d'analyse et de conception permettant une meilleure réalisation et aussi une maintenance plus efficace dans l'exploitation des logiciels. Le mot essentiel de ces méthodes est « structuration » dans le sens d'une décomposition en éléments ou blocs fonctionnels pour un niveau d'analyse donné et d'une décomposition hiérarchique cohérente entre les différents niveaux d'analyse. Ces méthodes d'analyse ou de conception structurées conduisent naturellement à la programmation structurée. La deuxième particularité commune à ces méthodes est la description sous forme de flux ou flots de données, de contrôle ou autres. L'aspect opérationnel de la description est alors visualisé par la propagation de ces flux.

Ainsi, nous trouvons la méthode SA-DT (*Structured Analysis Design Technics*) de spécification d'un système qui permet d'exprimer un bloc représentant soit les activités (fonctions) soit les données. Les flots entrants sont les données, un contrôle ou des mécanismes (méthodes) et les flots sortants correspondent aux sorties de

© Dunod – La photocopie non autorisée est un délit.

données. Cette méthode très générale de description d'un système a été adaptée à la spécification de logiciels avec la méthode très connue SA (*Structured Analysis*) (figure 2.1).

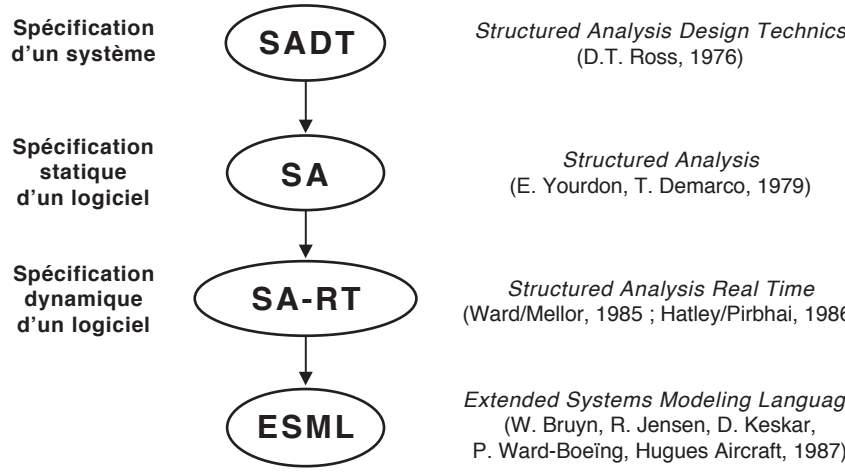

Figure 2.1 – Positionnement chronologique de la méthode SA-RT.

L'analyse structurée SA, définie par E. Yourdon et T. Demarco, est une méthode descendante par affinages successifs des traitements, appelés « process ». Les différents diagrammes sont donc ordonnés hiérarchiquement en faisant apparaître pour les derniers niveaux des fonctions élémentaires, appelées primitives élémentaires ou « process » primitifs. Les différents outils composant cette méthode sont :

– diagrammes de transformations de données ou diagramme de flots de données (DFD) ;
– dictionnaire de données ;
– spécifications des « process » primitifs.

Les diagrammes de flots de données sont construits à partir de quatre éléments graphiques : traitement (cercle), flot de données (flèche), unité de stockage (traits parallèles) et entité externe (rectangle) (tableau 2.1). À partir de ces éléments de base, il est possible de décrire l'aspect fonctionnel d'une application par un diagramme flots de données. Un exemple, présenté sur la figure 2.2, montre l'analyse d'une application très simple de régulation de température avec trois entités externes, deux process et une unité de stockage.

Remarque

Il est intéressant de noter que cette description graphique fonctionnelle d'une application à l'aide de la méthode SA sera presque entièrement reprise dans la méthode SA-RT, montrant bien ainsi sa dépendance chronologique.

Pour exprimer complètement le comportement de l'application, le diagramme flots de données de SA manquait d'un moyen permettant de spécifier l'aspect opérationnel,

Tableau 2.1 – Les différents éléments graphiques de la méthode SA.

Fonction	Signification	Représentation graphique	
Traitement ou process	Unité de travail qui réalise la transformation des données d'entrée en données de sortie	– Cercle ou bulle – Action décrite par : verbe + nom	Acquérir Mesures
Flot de données	Vecteur nommé reliant deux process, sur lequel circule un ensemble de données de même nature	– Flèche en trait plein – Donnée nommée	*Mesures*
Unité de stockage ou réservoir	Entité ou zone de rangement de données	– Deux traits parallèles – Entité nommée	*Mesures*
Entité externe ou terminateur	Provenance, source ou destination des données	– Rectangle – Entité nommée	Capteur

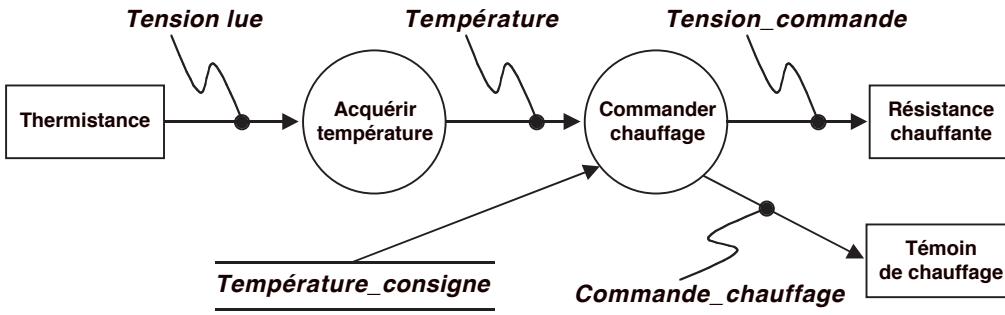

Figure 2.2 – Exemple simple du diagramme flot de données de la méthode SA correspondant à une application de régulation de température.

c'est-à-dire la description de l'enchaînement des différents process. Cette lacune fut comblée par la création de méthode SA-RT (*Structured Analysis-Real Time*). Deux groupes élaborèrent la méthode SA-RT avec des différences notables en termes de représentation : d'une part, la méthode établie par Ward et Mellor en 1985 qui associe le fonctionnel et le contrôle dans un même diagramme et, d'autre part, la méthode proposée par Hatley et Pirbhai en 1986 qui sépare le fonctionnel et le contrôle. Mais ces deux vues de la même méthode restent très similaires en termes de capacité d'expression de la spécification. Nous présentons dans cet ouvrage la méthode SA-RT établie par Ward et Mellor en 1985.

Comme le montre la figure 2.1, la méthode SA-RT a continué à évoluer au sein des entreprises en intégrant des besoins spécifiques à un domaine d'applications. Ainsi, nous trouvons une méthode SA-RT, appelée ESML et utilisée dans l'avionique, qui a été enrichie d'un point de vue flot de contrôle.

© Dunod – La photocopie non autorisée est un délit.

La méthode SA-RT intègre les trois aspects fondamentaux d'une méthode de spécification en mettant l'accent sur les deux premiers qui sont des points essentiels dans les applications de contrôle-commande :

– **aspect fonctionnel** (ou transformation de données) : représentation de la transformation que le système opère sur les données et spécification des processus qui transforment les données ;
– **aspect événementiel** (piloté par les événements) : représentation des événements qui conditionnent l'évolution d'un système et spécification de la logique de contrôle qui produit des actions et des événements en fonction d'événements en entrée et fait changer le système d'état ;
– **aspect informationnel** (données) : spécification des données sur les flots ou dans les stockages. Ce dernier aspect qui est en général assez négligé dans ce type d'application peut faire l'objet d'une description spécifique choisie au sein d'une entreprise.

2.2 Présentation de la syntaxe graphique de la méthode SA-RT

Nous allons présenter la syntaxe graphique complète de SA-RT permettant d'élaborer les différents diagrammes de la méthode. Cette syntaxe graphique très simple peut être scindée en deux parties : la syntaxe graphique afférente à l'aspect fonctionnel et la syntaxe dédiée à l'aspect contrôle ou événementiel.

2.2.1 Syntaxe graphique pour l'aspect fonctionnel

■ Syntaxe graphique du processus fonctionnel

En premier lieu, nous trouvons le **Processus fonctionnel** ou **Processus** qui représente une transformation de données. Un ou plusieurs flux de données en entrées sont traités pour donner un ou plusieurs flux de données en sortie (figure 2.3). Le Processus est représenté par un cercle avec une étiquette ou label explicite formé de :

Étiquette_Processus = verbe (+ un ou plusieurs compléments d'objets) + numéro

■ Syntaxe graphique du flot de données

Puis, nous avons le **Flot de Données** qui supporte ou transporte les valeurs d'une certaine information à différents instants. Ce concept représente le cheminement des données. Le flot de données est représenté par un arc orienté avec une étiquette ou label explicite formé de (figure 2.4) :

Étiquette_Flot_de_Données = nom (+ qualifiant)

Les valeurs de ce flot de données sont supposées disponibles pendant tout le temps où le processus producteur de ce flot est en mesure de les générer.
Le flot de données peut représenter aussi bien une donnée de type continu, codée par un entier ou un réel (`Température`) qu'une donnée discrète codée par un booléen,

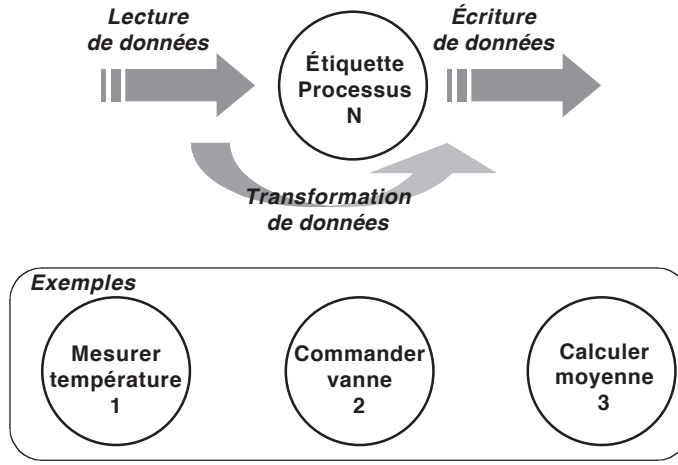

Figure 2.3 – Processus fonctionnel de la méthode SA-RT.

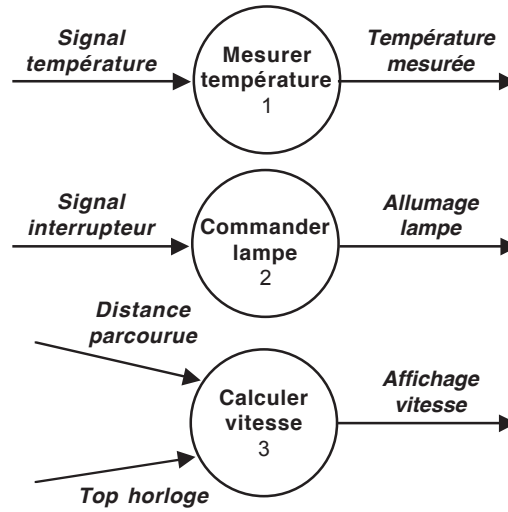

Figure 2.4 – Exemples simples de flots de données de la méthode SA-RT.

exemple « *Position_interrupteur* ». Un flot de données peut décrire aussi bien une donnée élémentaire ou unique, exemple « *Température* » qu'une donnée structurée intégrant plusieurs données élémentaires, exemple la donnée « *Pressions* » qui est composée de « *Pression_huile* » et « *Pression_air* ». Il est alors possible de faire apparaître l'étiquette du flot de données sous la forme suivante :

Étiquette_Donnée_Structurée = Étiquette_Donnée_1, Étiquette_Donnée_2

Une spécification détaillée de cette donnée, véhiculée par le flot de données, est faite dans le **Dictionnaire de données** (voir ci-après).

© Dunod – La photocopie non autorisée est un délit.

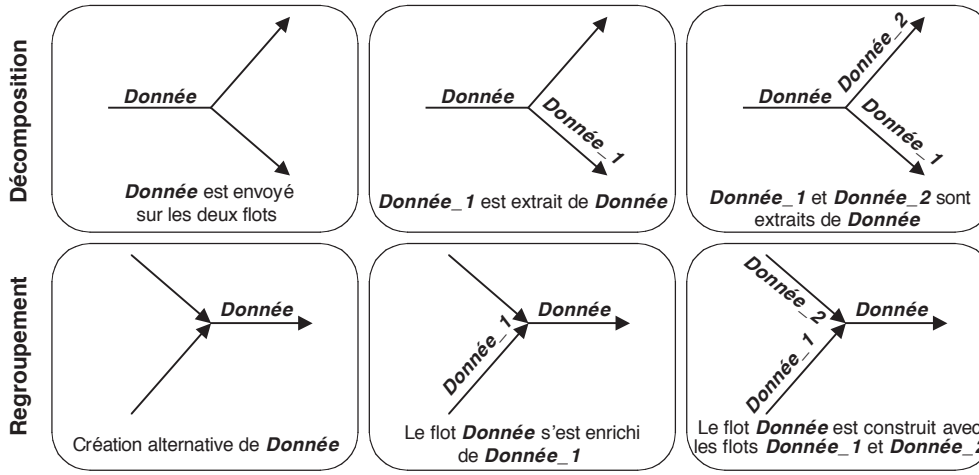

Figure 2.5 – Décomposition et regroupement des flots de données de la méthode SA-RT.

Ces flots de données peuvent se décomposer ou au contraire se regrouper lors des liaisons entre les processus fonctionnels dans le diagramme flot de données (figure 2.5).

■ Syntaxe graphique du stockage de données

Le troisième élément graphique est le **Stockage de Données** qui modélise le besoin de mémorisation d'une donnée de telle façon que sa valeur puisse être relue plusieurs fois. Comme le flot de données auquel il est étroitement associé, il est nommé par une étiquette ou label explicite formé de :

Étiquette_Stockage_de_Données = nom (+ qualifiant)

Le stockage de données est représenté par deux traits horizontaux encadrant l'étiquette définie ci-avant (figure 2.6). Les arcs « flots de données » arrivant ou partant de l'unité de stockage ne sont pas étiquetés s'ils transportent les données mémorisées complètes. Si une partie de la donnée est écrite ou lue, l'arc transportant de façon partielle la donnée doit être étiqueté avec le nom de cette donnée.

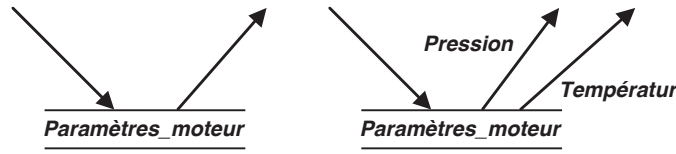

Figure 2.6 – Unité de stockage de la méthode SA-RT.

Un exemple de diagramme flot de données intégrant ces trois éléments graphiques de la méthode SA-RT (processus fonctionnel, flot de données et unité de stockage) est présenté sur la figure 2.7.

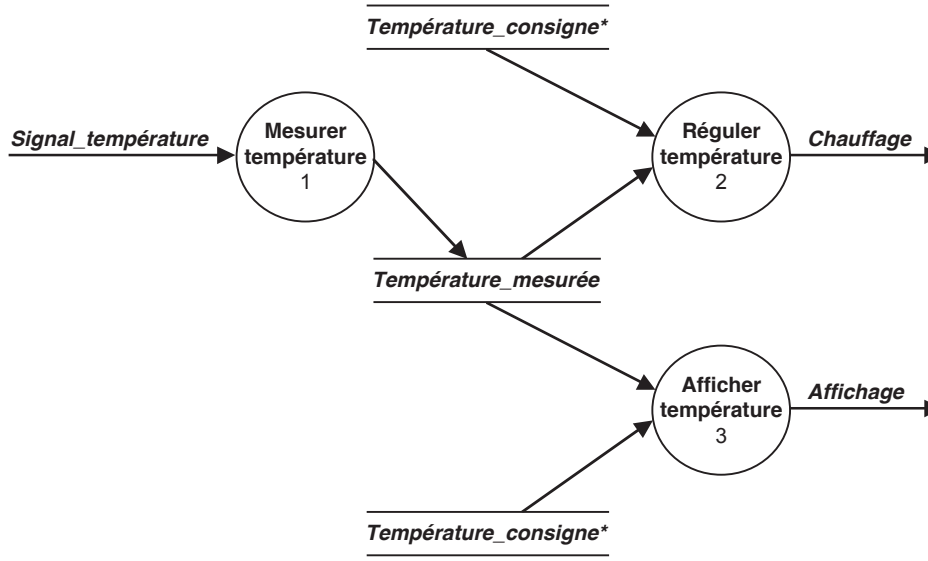

Figure 2.7 – Exemple d'un diagramme flot de données de la méthode SA-RT.

Remarque

Pour des besoins de clarté graphique, un stockage de données peut être visualisé plusieurs fois sur un diagramme flot de données en notant cette duplication par une « * ».

Il est important de noter que l'utilisation de l'élément « stockage de données » peut correspondre à deux cas :

– mémorisation ou représentation des constantes du système ;
– mémorisation de valeurs de données partagées entre deux ou plusieurs processus désynchronisés (consommation et production des données à des instants ou des rythmes différents).

■ Syntaxe graphique de la Terminaison

Enfin, le dernier élément graphique, utilisé dans cet aspect fonctionnel, est la **Terminaison**, ou encore appelée **« bord de modèle »**, qui représente une entité extérieure échangeant des données avec le système modélisé. Une terminaison peut donc être une entité logicielle (programme, base de données…) ou matérielle (capteurs, actionneurs, console opérateur…). Représentée par un rectangle, elle est nommée par une étiquette ou label explicite formé de (figure 2.8) :

Étiquette_Terminaison = nom (+ qualifiant)

© Dunod – La photocopie non autorisée est un délit.

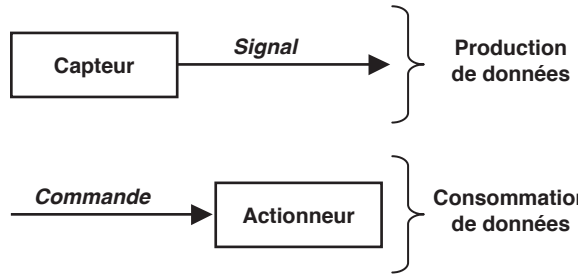

Figure 2.8 – Terminaison ou « bord de modèle » de la méthode SA-RT.

2.2.2 Syntaxe graphique pour l'aspect contrôle

Dans les diagrammes flot de données, le déclenchement de l'exécution des processus de transformation de données peut être lié au rythme d'apparition des données entrantes (diagramme piloté par les données : *data-driven*). Mais, dans le cas de la spécification des applications temps réel, il est préférable d'avoir un contrôle de ces transformations de données piloté par des conditions externes comme l'occurrence d'événements (*event-driven*) (figure 2.9). Cette vue dynamique du modèle impose la mise en place de la partie contrôle : processus de contrôle et flot de contrôle

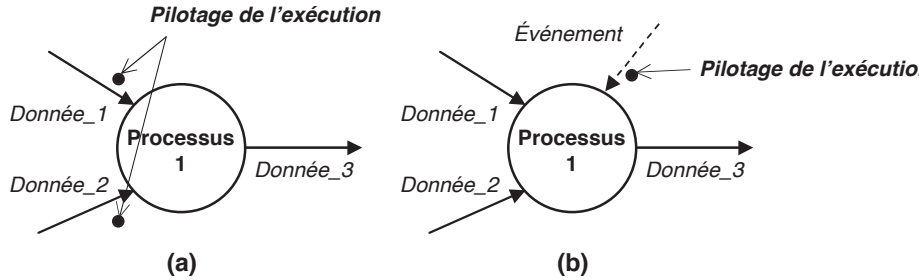

Figure 2.9 – Pilotage de l'exécution d'un processus fonctionnel : (a) piloté par les données et (b) piloté par les événements.

■ Syntaxe graphique du processus de contrôle

Le **Processus de contrôle** représente la logique du pilotage des processus fonctionnels. Il génère l'ensemble des événements qui vont activer ou désactiver les processus fonctionnels. En retour, les processus fonctionnels fournissent au processus de contrôle tous les événements nécessaires aux prises de décision. Le processus de contrôle ne peut en aucun cas gérer des données.

Le Processus de contrôle est représenté par un cercle en pointillé avec une étiquette ou label explicite formé de (figure 2.10) :

Étiquette_Processus_Contrôle = verbe (+ un ou plusieurs compléments) + numéro

Figure 2.10 – Processus de contrôle de la méthode SA-RT.

■ Syntaxe graphique du Flot de contrôle

Le **Flot de Contrôle** transporte les événements ou informations qui conditionnent directement ou indirectement l'exécution des processus de transformations de données. Le flot de contrôle est représenté par un arc orienté pointillé avec une étiquette ou label explicite formé de (figure 2.11) :

Étiquette_Flot_de_Contrôle = nom (+ qualifiant)

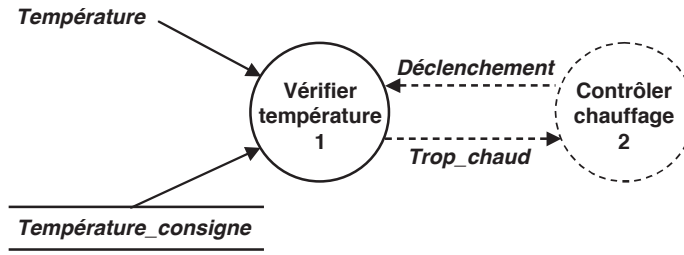

Figure 2.11 – Exemple simple d'une partie contrôle liée à une partie fonctionnelle de la méthode SA-RT.

Les événements, fournis par le **processus de contrôle**, sont généralement liés à l'activation ou à la désactivation des processus fonctionnels. Aussi, ces événements spécifiques ont été formalisés et prédéfinis :

– ***E*** pour *Enable* (activation) ;
– ***D*** pour *Disable* (désactivation) ;
– ***T*** pour *Trigger* (déclenchement).

Les deux premiers événements sont utilisés ensemble « *E/D* » pour piloter un processus fonctionnel de type « boucle sans fin » ou périodique, c'est-à-dire que le processus de contrôle doit lancer l'exécution de ce processus avec l'événement « *E* » et ensuite peut l'arrêter avec l'événement « *D* ». L'événement « *T* » est utilisé pour activer un processus fonctionnel de type « début-fin » ou sporadique, c'est-à-dire que le processus de contrôle doit lancer l'exécution de ce processus avec l'événement « *T* » et ensuite le processus s'arrête à la fin de son exécution sans intervention du contrôle.

© Dunod – La photocopie non autorisée est un délit.

2.3 Les diagrammes flot de données

2.3.1 Présentation d'un exemple simple d'application de contrôle-commande

Afin d'illustrer la méthodologie au fur et à mesure de la présentation, nous allons présenter un exemple simple qui va être décliné en détail au cours de cet ouvrage. Considérons un **système de freinage automobile** qui est constitué d'une part d'un ensemble classique composé d'une pédale de frein (demande de freinage) et d'un frein (actionneur de freinage) et d'autre part d'un système ABS (*Anti-blocking Brake System*). Un capteur de glissement de roues est associé à ce système ABS. Pour simplifier, le fonctionnement de l'ABS est basé sur un arrêt du freinage dès qu'un glissement est détecté sur les roues, et cela même si la demande du conducteur est toujours effective. Le conducteur a la possibilité d'activer ou non ce système ABS à l'aide d'un bouton spécifique (bouton à deux positions stables : interrupteur). Un voyant permet de lui indiquer l'activation du système ABS. En revanche, il n'est pas possible de désactiver le système ABS en cours de freinage, c'est-à-dire pendant l'appui sur la pédale de frein.
La spécification fonctionnelle de cette application à l'aide de la méthode SA-RT va s'effectuer en plusieurs étapes :

– diagramme de contexte ;
– diagramme préliminaire ;
– diagrammes de décomposition.

2.3.2 Diagramme de contexte d'une application

Le diagramme de contexte est une première étape extrêmement importante puisqu'elle va définir le contexte et l'environnement extérieur du système piloté. Nous pouvons la considérer comme le contrat de réalisation entre le concepteur et son client. Les bords de modèle ou terminaisons vont apparaître uniquement dans ce diagramme. Les descriptions précises de ces terminaisons, ainsi que des données ou éventuellement des événements entrants ou sortants de ceux-ci, sont à la charge du donneur d'ordre. Un exemple générique d'un diagramme de contexte est donné sur la figure 2.12.
Nous trouvons un et un seul processus fonctionnel, numéroté « 0 », qui traduit l'application à réaliser effectivement par le concepteur. Autour de ce processus fonctionnel, un ensemble de bords de modèles fournit ou consomme les données ou événements de cette application. Ces bords de modèles peuvent donc être les éléments physiques suivants :

– capteurs (thermocouples, dynamomètres, etc.) ;
– actionneurs (résistances chauffantes, vannes, etc.) ;
– opérateur(s) lié(s) à des capteurs (interrupteurs, potentiomètres, etc.) ;
– systèmes d'affichage (lampes, diodes, écran d'ordinateur, etc.) ;
– système de stockage ou de sauvegarde externe (disque, bande magnétique, etc.) ;
– système d'impression (imprimante, dérouleur papier, etc.) ;
– …

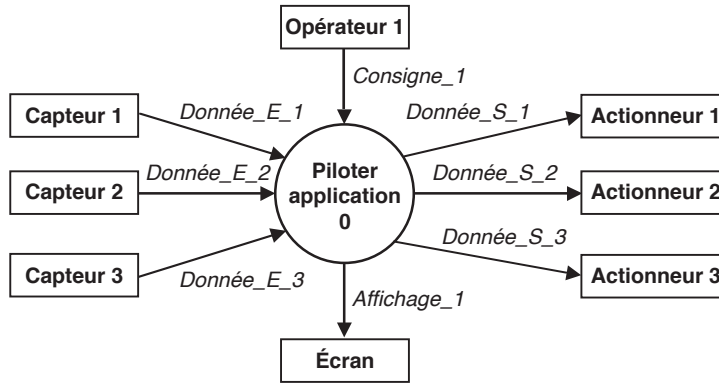

Figure 2.12 – Diagramme de contexte générique de la méthode de spécification SA-RT.

L'ensemble des données ou événements échangés avec l'extérieur du « processus fonctionnel », qui représente l'application, constitue les spécifications d'entrées et de sorties de l'application. La description de ces entrées/sorties sera faite dans le dictionnaire de données (§ 2.5).

Dans l'exemple simple d'un système de freinage automobile (§ 2.3.1), le diagramme de contexte est constitué du processus fonctionnel « Contrôler système freinage 0 » et de cinq bords de modèles (figure 2.13) :

– terminaison « Pédale de frein » fournissant la donnée « *Demande_freinage* » ;
– terminaison « Bouton d'activation de l'ABS » fournissant la donnée « *Activation_ABS* » ;
– terminaison « Capteur de glissement » fournissant la donnée « *Glissement_roue* » ;
– terminaison « Système de freinage » consommant la donnée « *Commande_freinage* » ;
– terminaison « Voyant ABS » consommant la donnée « *Affichage_ABS* ».

Ce diagramme de contexte définit parfaitement l'interface entre le concepteur et le client, c'est-à-dire les données à fournir ou à générer. La suite du travail d'analyse

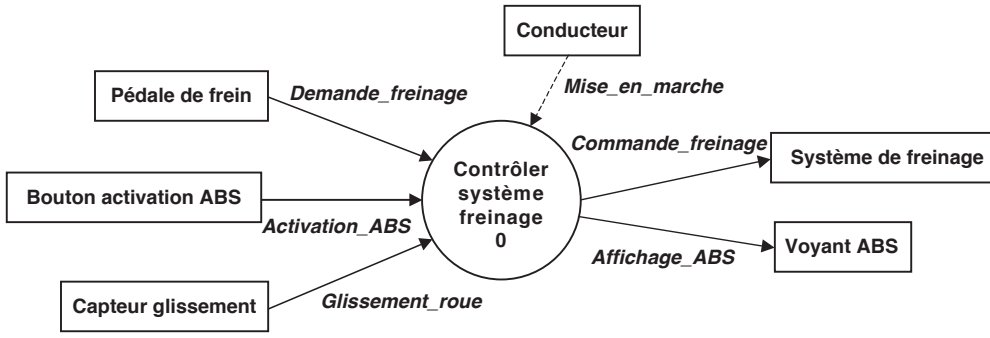

Figure 2.13 – Diagramme de contexte de l'application « système de freinage automobile ».

© Dunod – La photocopie non autorisée est un délit.

va donc se situer dans l'expression du processus fonctionnel à réaliser : « Contrôler système freinage 0 ».

2.3.3 Diagramme préliminaire et diagrammes de décomposition

Le premier niveau d'analyse est représenté par le **diagramme préliminaire**. Ce diagramme préliminaire est la première décomposition du processus à réaliser présenté dans le diagramme de contexte. À ce niveau, le diagramme représente la liste « graphique » des processus fonctionnels nécessaires à l'application sans se soucier de l'enchaînement (séquence d'exécution).

Le nombre de processus fonctionnels, composant ce diagramme préliminaire, doit être limité pour avoir une meilleure lisibilité : **5** à **9** maximum.

Dans ce cadre, nous pouvons aussi trouver un motif générique permettant de décrire une fonction simple ou complexe de contrôle-commande en la divisant selon les trois éléments de base, soit :

– un processus d'acquisition ;

– un processus de traitement (loi de régulation) ;

– un processus de commande.

Selon l'application, tout ou partie de ce diagramme flot de données générique peut être présent pour chaque chaîne de contrôle-commande impliquée dans le niveau d'analyse en cours d'élaboration. Ce diagramme flot de données générique peut être utilisé au niveau du diagramme préliminaire ainsi que dans les niveaux d'analyse suivant des diagrammes de décomposition.

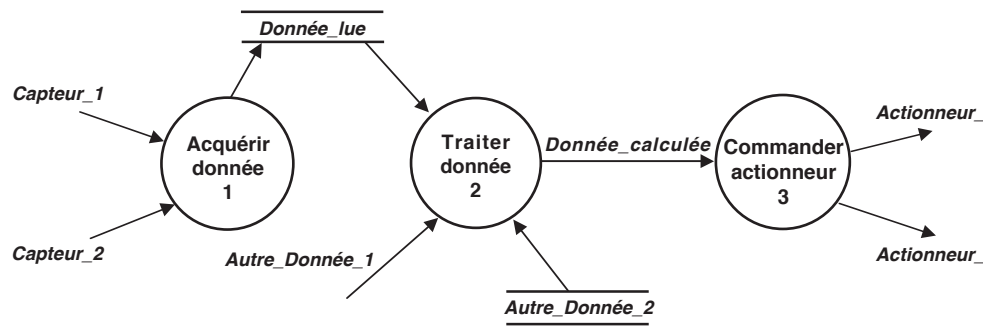

Figure 2.14 – Décomposition fonctionnelle générique.

Le passage des données entre les processus fonctionnels peut être réalisé selon les besoins avec les deux méthodes de base : flots de données direct (exemple entre les processus 2 et 3) ou unité de stockage (cas des processus 1 et 2).

Ainsi, dans l'exemple simple d'un système de freinage automobile (§ 2.3.1), le diagramme préliminaire est constitué de cinq processus fonctionnels (figure 2.15). Au niveau de cet exemple simple, nous n'avons pas une décomposition fonctionnelle aussi complexe que l'exemple générique présenté précédemment : seules les parties

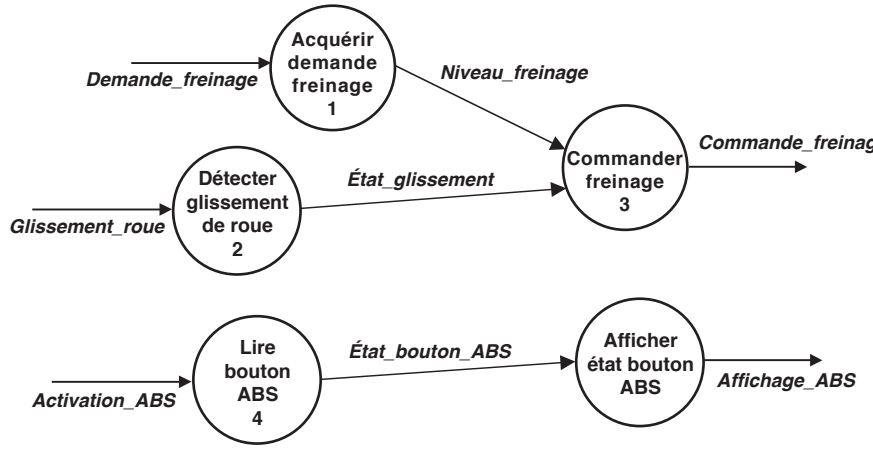

Figure 2.15 – Diagramme flot de données préliminaire de l'application « système de freinage automobile ».

acquisition et commande sont présentes dans le cas du contrôle-commande du freinage et du bouton ABS.

Nous pouvons souligner immédiatement la cohérence obligatoire entre le diagramme de contexte (figure 2.13) et ce diagramme préliminaire au niveau des flots de données entrants et sortants.

Le passage des données entre les processus fonctionnels est effectué de façon directe. Il est important de noter que la donnée « *Niveau_freinage* » est de type entier ou réel alors que les données « *Etat_glissement* » et « *Etat_bouton_ABS* » sont de type booléen. Cela va entraîner dans la suite de cette analyse une modification de ce diagramme préliminaire.

Cette décomposition fonctionnelle en diagrammes flots de données peut se poursuivre en raffinant de plus en plus la description des processus fonctionnels (figure 2.16). Chaque **diagramme de décomposition**, associé à un processus fonctionnel – numéroté « N » – du diagramme hiérarchiquement supérieur, est référencé au niveau des processus fonctionnels par des numéros « N.x ».

L'exemple simple « système de freinage automobile » choisi pour illustrer la méthodologie n'est pas assez complexe pour justifier la décomposition d'un des processus fonctionnels. Donc tous les processus fonctionnels du diagramme préliminaire de la figure 2.15 sont des processus primitifs.

Lorsqu'il n'y a plus d'intérêt à décomposer un processus, celui-ci est appelé **processus primitif** et doit être décrit par une spécification sous forme textuelle, tabulaire ou graphique (voir ci-après).

Nous pouvons énoncer les deux premières règles de cohérence de cette décomposition hiérarchique :

– l'ensemble des flots entrants et sortants du processus décomposé doit se retrouver dans le diagramme de décomposition de ce processus avec les mêmes typages (données ou événements) ;

© Dunod – La photocopie non autorisée est un délit.

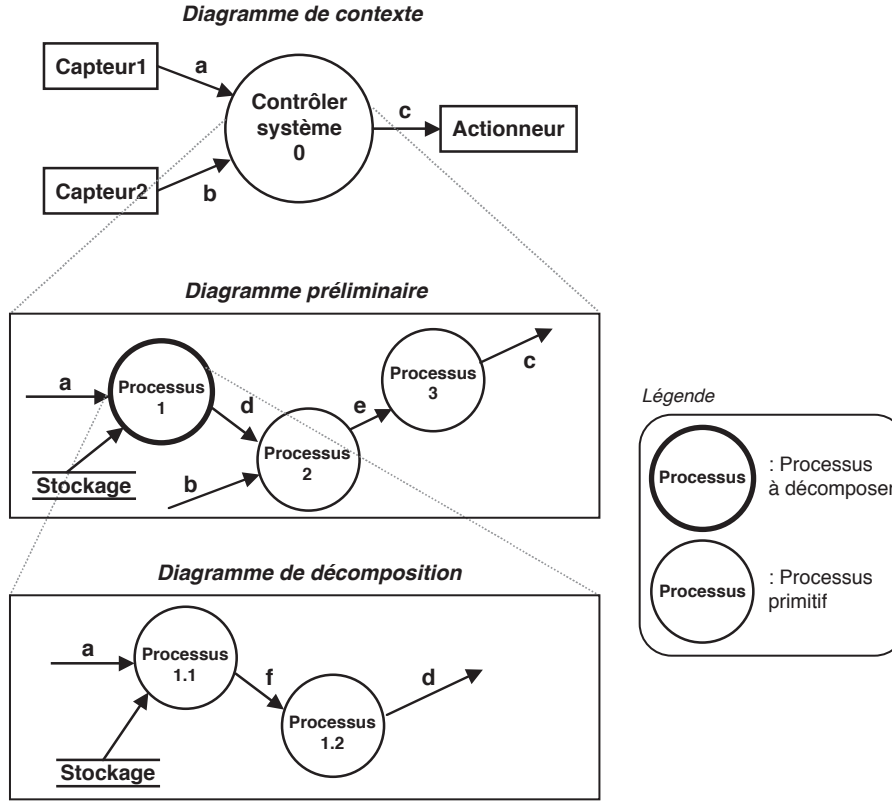

Figure 2.16 – Décomposition hiérarchique en diagramme flots de données de la méthode d'analyse SA_RT.

– la numérotation des différents processus fonctionnels doit intégrer le numéro du processus fonctionnel analysé « N » sous la forme « N.x » ;
– les stockages doivent apparaître dans tous les diagrammes où les processus les utilisent.

2.3.4 Conclusion

La décomposition hiérarchique fonctionnelle, explicitée dans ce paragraphe, est la base de la méthode d'analyse SA-RT. Mais l'aspect temporel ou plus exactement le contrôle de l'enchaînement dans l'exécution de ces différents processus fonctionnels n'est pas décrit au travers de ces diagrammes flots de données. Ceci va donc faire l'objet d'un complément au niveau de ces diagrammes par l'aspect « contrôle ».

2.4 L'aspect contrôle de la méthode SA-RT

2.4.1 Mise en place du processus de contrôle

Le processus de contrôle, tel que nous l'avons déjà explicité dans le paragraphe 2.2.2, va permettre de spécifier l'enchaînement des différents processus fonctionnels de l'application. Ainsi, ce processus de contrôle va être positionné dans le diagramme préliminaire et dans les diagrammes de décomposition s'ils existent.

■ Flots de contrôle et diagramme de contexte

Nous pouvons souligner qu'il ne peut pas exister de processus de contrôle au niveau du diagramme de contexte puisqu'un seul diagramme fonctionnel est représenté. En revanche, il est tout à fait possible d'avoir des flots de contrôle au niveau de ce diagramme de contexte entre les terminaisons et le processus fonctionnel. Ces flots de contrôle, correspondant à des signaux tout ou rien, doivent être réservés à des signaux particuliers ne nécessitant aucun processus fonctionnel particulier (acquisition, mise en forme…). Ainsi, nous pouvons citer les événements externes tels que « frappe clavier », « click souris », « interrupteur de mise sous tension », etc. Pour préciser de façon plus complète ces notions de signaux et d'événements, nous pouvons considérer qu'un signal nécessite une acquisition (scrutation périodique par exemple) et une mise en forme minimum. Le résultat de ce « prétraitement » peut alors effectivement devenir un événement. La figure 2.17a illustre cette différence signal/événement en considérant l'exemple d'un interrupteur. Le signal électrique fourni par l'interrupteur tout ou rien peut être considéré comme un événement

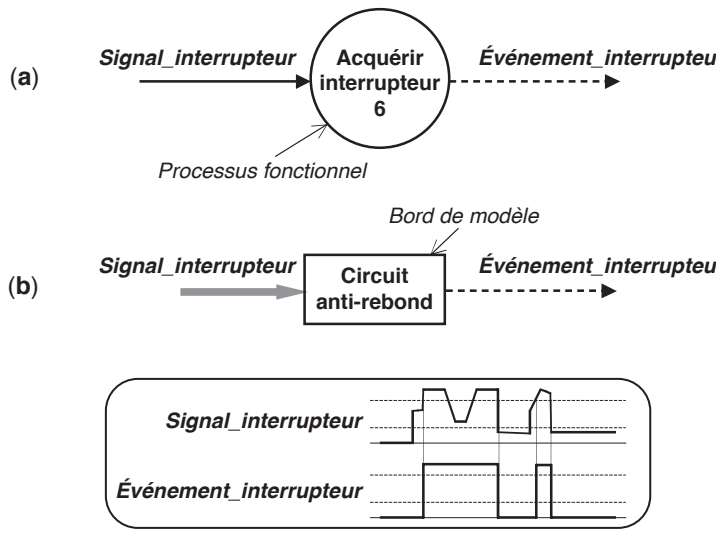

Figure 2.17 – Méthodes de prise en compte d'un signal électrique afin de réaliser une transformation données-événements : (a) processus fonctionnel dans le diagramme préliminaire, (b) circuit électronique visualisé au niveau du diagramme de contexte.

© Dunod – La photocopie non autorisée est un délit.

s'il est directement compréhensible par le système informatique, par exemple connexion à une ligne d'interruption. Dans le cas contraire, il nécessite une lecture régulière et une mise en forme pour être intégré à la logique du programme. Il est important de noter que ces processus de mise en forme des signaux afin de transformer une donnée en un événement peuvent parfois être remplacés de façon beaucoup plus facile et efficace par un circuit électronique (figure 2.17b).
Dans le cas de l'exemple simple du système de freinage automobile (§ 2.3.1), le diagramme de contexte peut être enrichi d'un flot de contrôle émanant d'un nouveau bord de modèle « Conducteur » qui fournit un flot de contrôle « *Mise_en_marche* » au processus fonctionnel (figure 2.18).

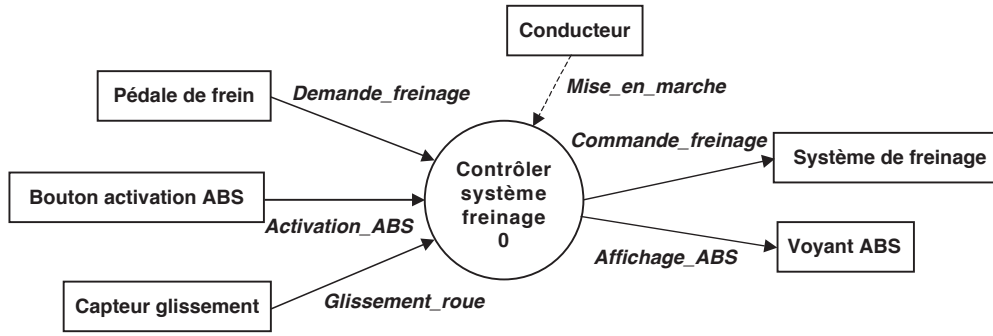

Figure 2.18 – Diagramme de contexte complet de l'application « système de freinage automobile ».

Rappelons que l'ensemble des flots entrants et sortants de l'unique processus fonctionnel du diagramme de contexte doit se retrouver dans le diagramme préliminaire y compris les flots de contrôle s'ils existent.

■ Processus de contrôle et diagramme préliminaire

Nous allons donc implanter un processus de contrôle dans le diagramme préliminaire afin de coordonner l'exécution des différents processus fonctionnels. Les concepts et les règles de mise en place de ce processus de contrôle sont identiques dans le cas des diagrammes de décomposition. Ce processus de contrôle va donc interagir avec un processus fonctionnel pour lancer ou activer son exécution et, en retour, le processus fonctionnel fournira si nécessaire un événement indiquant le résultat de son traitement afin de donner des informations utiles aux changements d'états du contrôle. Nous pouvons ainsi illustrer cette interaction entre le processus de contrôle et les processus fonctionnels selon le schéma générique utilisant la décomposition fonctionnelle complète présentée sur la figure 2.14. Rappelons que l'événement couplé « *E/D* » est utilisé pour piloter un processus fonctionnel de type « boucle sans fin » et l'événement « *T* » est utilisé pour activer un processus fonctionnel de type « début-fin ». Dans cet exemple de la figure 2.19, deux processus fonctionnels, supposés de type « début-fin », sont activés l'un après l'autre par le processus de contrôle par un événement « T » et envoient un événement de fin d'exécution vers

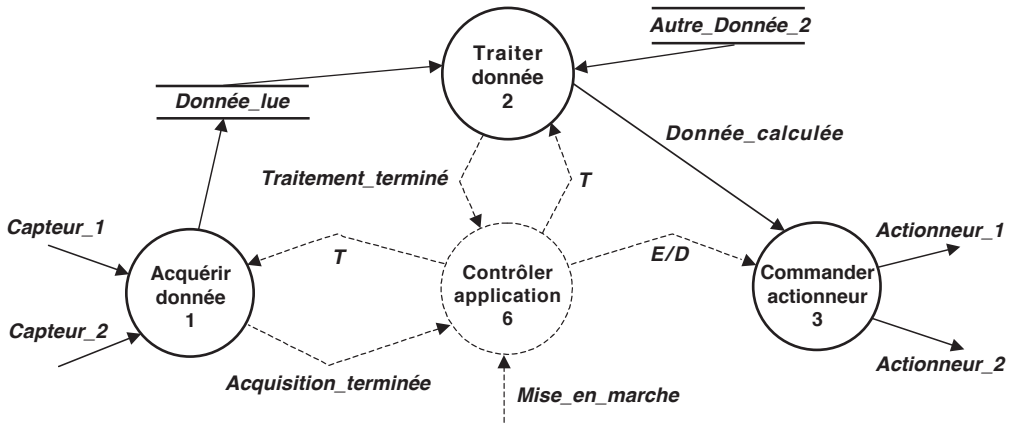

Figure 2.19 – Schéma générique de l'interaction entre le processus de contrôle et les processus fonctionnels.

ce même processus de contrôle. Le troisième processus fonctionnel, activé par un événement « E/D », est lié à la commande des actionneurs, qui doit être faite en continu, soit type « boucle sans fin ».
Concernant cette mise en place d'un processus de contrôle dans un diagramme préliminaire, nous pouvons faire plusieurs remarques :

– Un diagramme préliminaire ne doit contenir qu'un seul processus de contrôle. En effet, il est difficilement concevable d'avoir plusieurs organes de contrôle-commande pour une seule application, pour des raisons de cohérence.
– Un diagramme préliminaire, ou, *a fortiori*, un diagramme de décomposition, peut ne pas avoir de processus de contrôle. Dans ce cas, tous les processus fonctionnels sont supposés s'exécuter en même temps avec pour seule règle celle des flots de données.
– Un processus fonctionnel peut ne pas être connecté au processus de contrôle. Dans ce cas, il est supposé être activé au démarrage de l'application et ne jamais s'arrêter. Aussi, pour augmenter la lisibilité, il est préférable de le connecter au processus de contrôle avec un événement de type « E/D » et de l'activer définitivement au début de l'application en utilisant l'événement « E ».

Dans le cas de l'exemple du système de freinage automobile, le diagramme préliminaire, représenté sur la figure 2.15, va être modifié pour intégrer un processus de contrôle. En particulier, les deux flots de données « *Etat_glissement* » et « *Etat_bouton_ABS* » de type booléen deviennent des événements envoyés respectivement par les processus fonctionnels « Détecter glissement roue » et « Lire boutons ABS ». Nous obtenons alors le diagramme préliminaire complet de la figure 2.20.
Cet exemple de diagramme préliminaire, présenté sur la figure 2.20, doit faire l'objet de plusieurs remarques qui sont généralisables à la réalisation d'un diagramme préliminaire quelconque. Ainsi, nous pouvons noter :

– Tous les différents processus fonctionnels sont liés au processus de contrôle par des flots de contrôle de type « E/D » ou « T ». Le choix de l'un ou l'autre de ces

© Dunod – La photocopie non autorisée est un délit.

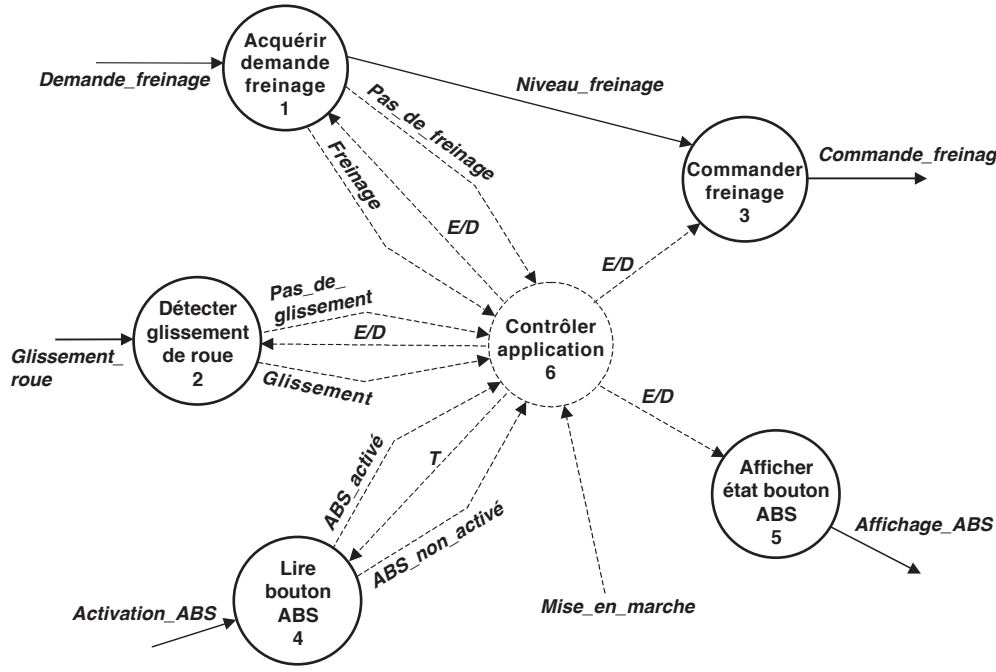

Figure 2.20 – Diagramme préliminaire complet de l'application « système de freinage automobile ».

événements est fait, comme nous l'avons déjà vu, en fonction de la structure de fonctionnement interne du processus fonctionnel : processus de type « boucle sans fin » ou de type « début-fin ».

– Certains processus fonctionnels possèdent des retours d'exécution vers le processus de contrôle. Dans ce cas, il est souhaitable d'expliciter les événements indiquant le résultat du traitement, par exemple « *ABS_activé* ». Ces événements pourraient être caractérisés par des variables de type booléen ; ainsi l'absence d'occurrence de l'événement « *ABS_activé* » pourrait être interprétée comme NON (« *ABS_activé* ») – NON étant l'opération booléenne. Mais, dans beaucoup de cas, il est préférable pour des raisons de clarté opérationnelle, de faire apparaître les deux événements possibles : « *ABS_activé* » et « *ABS_non_activé* ».
– Il ne faut pas oublier de connecter les flots de contrôle du diagramme précédent réalisé dans l'analyse descendante, par exemple l'événement « *Mise_en_marche* » du diagramme de contexte.

La mise en place du processus de contrôle au niveau d'un diagramme préliminaire ou d'un diagramme de décomposition avait pour but d'exprimer l'exécution ou l'enchaînement des processus fonctionnels. L'objectif n'est pas complètement atteint puisque le diagramme flot de données ainsi complété ne reflète pas cette exécution. Il est nécessaire d'ajouter cette information supplémentaire décrivant le fonctionnement du processus de contrôle, cela se traduit généralement par un diagramme état/transition que nous allons décrire dans le paragraphe suivant.

2.4.2 Diagramme état/transition

■ Représentation d'un diagramme état/transition

Dans le cas où un diagramme flots de données possède un processus de contrôle, la compréhension de ce diagramme de flot de données, tracé à un certain niveau d'analyse, nécessite une description ou spécification du processus de contrôle. Cette spécification, représentant l'aspect comportemental ou temps réel de l'application, peut être faite de diverses manières : diagramme état-transition, table état-transition, matrice état-transition ou éventuellement un grafcet.
La représentation la plus courante est le **diagramme état-transition**, ou automate synchronisé, qui est composé de quatre éléments (figure 2.21) :

– **état courant** correspondant à un fonctionnement précis du système, en particulier à un état des processus fonctionnels (exécution ou non) ;
– **événement** : occurrence d'un événement émanant d'un processus fonctionnel vers le processus de contrôle qui va provoquer le franchissement de la transition et donc faire changer l'état du système ;
– **action** : occurrence d'un événement émanant du processus de contrôle vers un ou plusieurs processus fonctionnels pour les activer (« E » ou « T ») ou les désactiver (« D »). Ces actions caractérisent l'effet du franchissement de la transition ;
– **état suivant** correspondant au fonctionnement après les actions faites par le processus de contrôle en direction des processus fonctionnels.

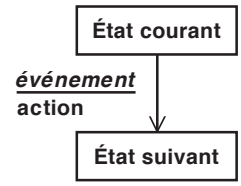

Figure 2.21 – Représentation de base d'une cellule élémentaire d'un diagramme état/transition.

Pour illustrer cette description du processus de contrôle avec un exemple très générique, reprenons le diagramme préliminaire de la figure 2.19 qui représente la coordination d'un système acquisition-traitement-commande par un processus de contrôle. La figure 2.22 montre la séquentialité évidente de l'exécution d'une telle application. À propos de ce diagramme état/transition, nous pouvons faire les remarques générales suivantes :

– Les noms des différents états atteints par le système explicitent l'état de fonctionnement du système. Ces états ne doivent pas en principe être des états fugitifs.
– Les événements intervenant au niveau de la transition entre ces états doivent tous appartenir au diagramme préliminaire et réciproquement tous les événements du diagramme état/transition doivent être représentés dans le diagramme préliminaire.

© Dunod – La photocopie non autorisée est un délit.

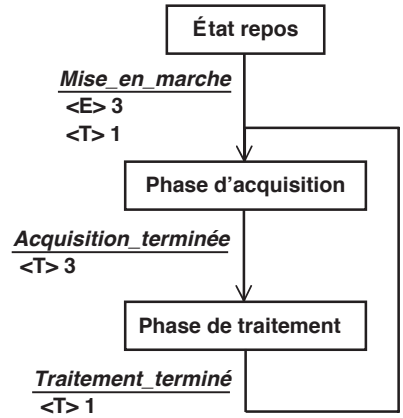

Figure 2.22 – Représentation du diagramme état/transition du processus de contrôle du diagramme préliminaire de la figure 2.19.

– Les actions du processus de contrôle sur les processus fonctionnels sont notées par « **< E** *ou* **D** *ou* **T >** + ***numéro*** *ou* ***nom*** *du processus fonctionnel* ». Il peut y avoir plusieurs actions de ce type associées à une seule transition.

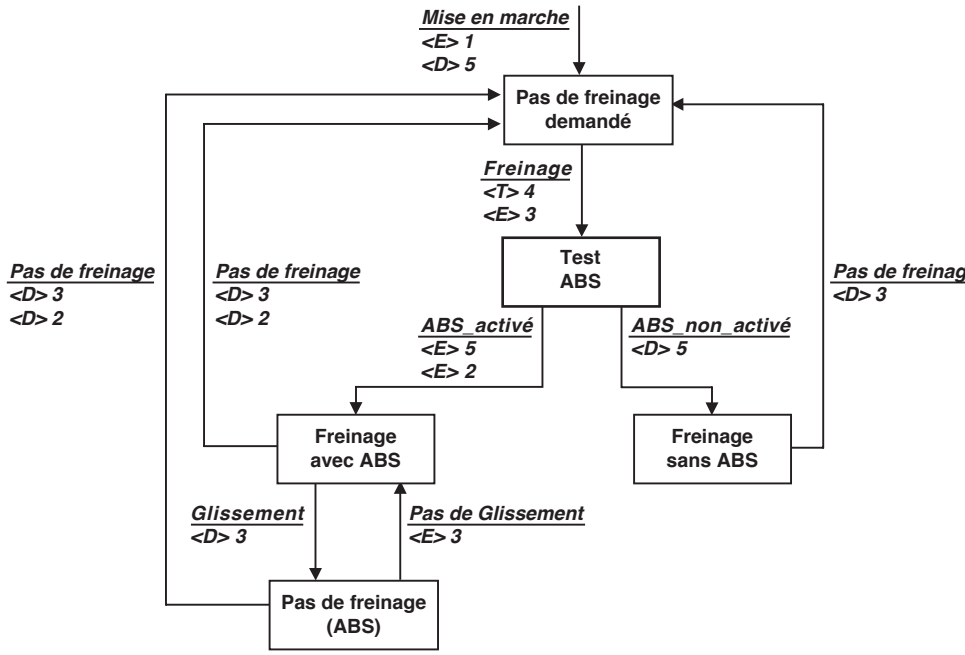

Figure 2.23 – Première représentation du diagramme état/transition du processus de contrôle « Contrôler application » du diagramme préliminaire de l'application « système de freinage automobile » : comportement non conforme.

Si nous considérons l'exemple du système de freinage automobile, une solution possible du diagramme état/transition du processus de contrôle « Contrôler application » du diagramme préliminaire est présentée sur la figure 2.23. Ce diagramme état/transition explique le comportement de l'application.
Une analyse rapide de cet exemple simple montre une incohérence dans ce comportement ; en effet, l'état initial de l'application « Pas de freinage demandé » n'est sensible qu'à l'occurrence de l'événement « *Freinage* » provenant du processus fonctionnel 1 « Acquérir demande freinage ». Le bouton d'activation ou de désactivation du système ABS avec le voyant associé n'est pris en compte que lors d'un freinage (événement « *Freinage* »). Ce comportement ne correspond pas à un fonctionnement correct du système où le conducteur doit accéder à cette fonction et voir son résultat en dehors de la phase de freinage. En effet, avec un tel diagramme état/transition, le conducteur ne peut voir la modification du voyant ABS qu'en actionnant la pédale de frein.
Un deuxième diagramme état/transition propose un autre comportement de l'application plus conforme aux spécifications d'un tel système. Ce diagramme état/transition, présenté sur la figure 2.24 (page suivante), montre clairement les deux fonctionnements du système en cas de freinage avec le système ABS activé ou sans le système ABS. Dans ce cas du diagramme état/transition, le conducteur voit immédiatement la modification du voyant ABS lors de l'appui sur le bouton ABS.
Une modification au niveau des événements d'activation par le processus de contrôle a été effectuée pour le processus 4 « Lire bouton ABS ». Dans le diagramme préliminaire de la figure 2.20, l'événement est de type « T ». Or, dans le nouveau diagramme état/transition de la figure 2.24 par rapport au premier diagramme de la figure 2.23, cet événement est de type « E/D ». Il est très important de noter que, pour conserver la cohérence entre les diagrammes d'analyse, c'est-à-dire le diagramme préliminaire et le diagramme état/transition du processus de contrôle, il est indispensable de modifier le diagramme préliminaire de la figure 2.20 comme le montre la figure 2.25, page suivante.

■ Règles générales d'élaboration d'un diagramme état/transition

L'exemple simple traité précédemment montre de façon évidente qu'il est indispensable de conduire en parallèle d'une part la réalisation du diagramme préliminaire au niveau de la mise en place du processus de contrôle et d'autre part la construction du diagramme état/transition de ce processus de contrôle. En effet, que ce soit au niveau des événements d'activation ou de désactivation issus du processus de contrôle, ou pour les événements provenant des processus fonctionnels, il est nécessaire de les mettre en place dans le diagramme préliminaire au fur et à mesure de la construction du diagramme état/transition. Aussi, à la fin de ce travail, nous devons vérifier impérativement que tous les événements du diagramme préliminaire ont été utilisés et uniquement ceux-ci.
D'autre part, le passage d'un état à un autre dans un diagramme état/transition peut être réalisé par l'occurrence d'un événement ou de la combinaison logique de plusieurs événements. Il est donc possible d'utiliser les opérateurs de la logique combinatoire pour grouper les événements : ET, OU, NON. Dans ce cas, il faut être vigilant pour ne pas diminuer fortement la lisibilité du diagramme qui représente

© Dunod – La photocopie non autorisée est un délit.

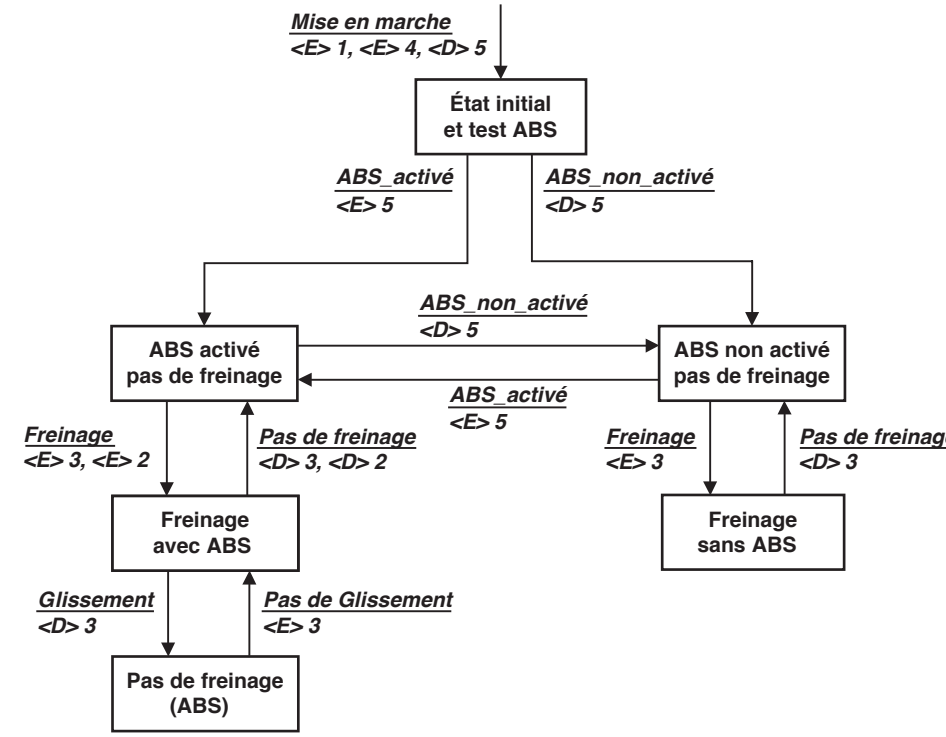

Figure 2.24 – Deuxième représentation plus correcte du diagramme état/transition du processus de contrôle « Contrôler application » du diagramme préliminaire de l'application « système de freinage automobile ».

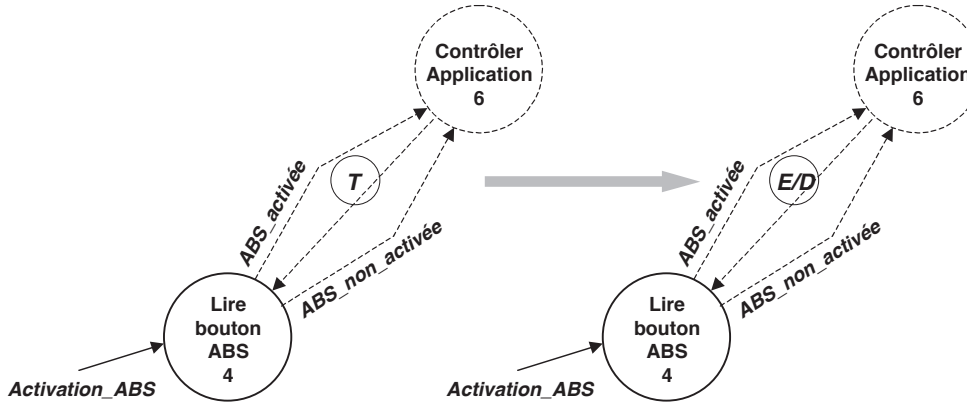

Figure 2.25 – Modification du diagramme préliminaire de l'application « système de freinage automobile » de la figure 2.20 pour garder la cohérence avec le diagramme état/transition de la figure 2.24.

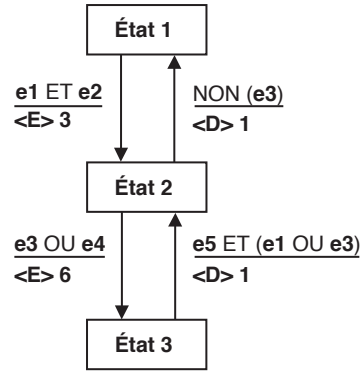

Figure 2.26 – Exemple d'utilisation de combinaisons logiques d'événements.

le fonctionnement de l'application (figure 2.26). Aussi, il est fortement déconseillé d'utiliser des combinaisons « complexes » d'événements comme celui du passage de l'état 3 à l'état 2 de la figure 2.26. Ainsi, se limiter à la combinaison de deux événements semble être une règle de bon compromis : lisibilité et puissance d'expression.

2.5 Spécification des processus primitifs

Le processus du diagramme de contexte étant numéroté « 0 », les processus du diagramme préliminaire seront notés 1, 2, 3… Les processus de ce diagramme flot de données sont ensuite décomposés si nécessaire et décrits par des diagrammes flot de données, appelés diagrammes de décomposition. Ainsi, la décomposition du processus « 1 » donnera naissance à des processus numérotés « 1.1, 1.2, 1.3… ». Lorsque le processus fonctionnel est suffisamment simple, c'est un processus primitif ; il doit être décrit par une spécification sous forme textuelle (spécification procédurale, par précondition et postcondition), tabulaire ou graphique. La méthode SA-RT n'étant pas normalisée et sans indication particulière concernant la spécification des processus primitifs, les utilisateurs décrivent ces processus fonctionnels selon les méthodes usitées dans l'entreprise.
Une des méthodes de spécifications de processus fonctionnels, la plus usitée et adaptée à ce domaine, est celle qui s'appuie sur une spécification procédurale. Celle-ci se décline en 6 mots-clés :

– « E/ données : *Nom_flots_de_données…* » : liste des flots de données en entrée du processus fonctionnel ;
– « E/ événements : *Nom_flots_d'événements…* » : liste des flots d'événements en entrée du processus fonctionnel, c'est-à-dire en général « E/D » ou « T », ou éventuellement les événements produits directement par les bords de modèle ;
– « S/ données : *Nom_flots_de_données…* » : liste des flots de données en sortie du processus fonctionnel ;
– « S/ événements : *Nom_flots_d'événements…* » : liste des flots d'événements en sortie du processus fonctionnel ;

© Dunod – La photocopie non autorisée est un délit.

– « Nécessite : » : liste des contraintes sur les données en entrée du processus fonctionnel ;
– « Entraîne : » description algorithmique du traitement à réaliser.

Dans le cas de l'exemple simple « système de freinage automobile », nous allons considérer le processus fonctionnel 1 « Acquérir demande de freinage ». La description procédurale de ce processus est la suivante :

Processus_fonctionnel : Acquérir demande freinage
E/ données : *Demande_freinage*
E/ événements : *E/D*
S/ données : *Niveau_freinage*
S/ événements : *Freinage*, *Pas_de_freinage*
Nécessite : *Demande_freinage* = 0
Entraîne :
    Faire toujours
        Acquérir *Demande_freinage*
        Émettre *Niveau_freinage* vers Processus 3 (Commander freinage)
        Si *Demande_freinage* > 0
            Alors Émettre *Freinage* vers Processus de contrôle 6 (Contrôler freinage)
            Sinon Émettre *Pas_de_freinage* vers Processus de contrôle 6 (Contrôler freinage)
        Finsi
    Fin faire

Il est aussi possible de faire une spécification de type pré et postcondition sur le modèle suivant :

– « Précondition : » : liste des flots de données ou d'événements en entrée du processus fonctionnel ave les conditions associées.
– « Postcondition » : liste des flots de données ou d'événements en sortie du processus fonctionnel ave les conditions associées.

Donc, dans le cas de l'exemple simple « système de freinage automobile », nous avons le processus fonctionnel 1 « Acquérir demande de freinage » spécifié de la façon suivante :

Processus_fonctionnel : Acquérir demande freinage
Précondition :
    Demande_freinage est fourni et *Demande_freinage* = 0
    E est fourni
Postcondition :
    Niveau_freinage est fourni
    Freinage est fourni si *Demande_freinage* > 0
    Pas_de_freinage est fourni si *Demande_freinage* = 0

2.6 Spécification des données

Au fur et à mesure de la réalisation des différents diagrammes flots de données (diagramme de contexte, diagramme préliminaire et diagrammes de décomposition), un ensemble de données et d'événements est défini. Les données et les événements, qui interviennent à tous les stades du modèle, sont alors réunis dans un dictionnaire de données.

De la même manière que, pour la spécification des processus primitifs, il n'y a pas de méthode imposée, les utilisateurs spécifient les données selon les méthodes personnalisées dans l'entreprise. Pour cet aspect informationnel de SA-RT, nous allons décrire deux méthodes qui peuvent répondre à des besoins simples de spécifications des données : représentation textuelle et représentation graphique. Des méthodes plus complexes et plus formelles existent et sont nécessaires si les données au niveau de l'application ont une place importante en nombre ou en complexité.

2.6.1 Représentation textuelle des données

La spécification des données de manière textuelle peut utiliser une notation précise comme celle basée sur la notation BNF (*Backus-Naur Form*) décrite dans le tableau 2.2.

Tableau 2.2 – Notation B.N.F. permettant de spécifier les données.

Symbole	Signification
= ...	Composé de...
.....	Commentaire
+	Regroupement sans ordre
{.....}	Itération non bornée
n{.....}p	Itération de *n* à *p*
(.....)	Optionnel
" ....."	Expression littérale
(...|...) ou (.../...)	Ou exclusif

Ainsi, nous pouvons proposer une description de chaque donnée ou événement de la façon suivante :

– Donnée ou événement à spécifier = désignation.

– **Rôle** : description fonctionnelle de la donnée ou de l'événement.

– **Type** : description du codage, domaine de valeurs, etc.

© Dunod – La photocopie non autorisée est un délit.

La désignation de la donnée ou de l'événement peut être faite avec quelques mots prédéfinis : signal en entrée, signal en sortie, événement en entrée, événement en sortie, donnée interne, événement interne ou composition formée d'un ensemble des éléments précédents.

La formalisation de la représentation est limitée, mais elle permet toutefois d'avoir une description homogène et précise de l'ensemble des données ou des événements. En particulier, cette représentation permet d'exprimer la composition entre plusieurs données ou événements. Ainsi, nous pouvons donner les quelques exemples suivants :

– Opérateur de composition « = » et opérateur de séquence « **+** »

Pression = *Pression_air* + *Pression_huile*

– Opérateur de sélection « **[ ]** » et ou exclusif « **|** » ou « **/** »

Pression = [*Pression_calculée*| *Pression_estimée*]

– Opérateur d'itération « **{ }** » et avec bornes

*Pression*s = 1{*Capteur_pression*}5

– Opérateur indiquant le caractère optionnel « **( )** »

Température = *Capteur_température* + (*Valeur_par_défaut*)

Pour l'exemple simple « système de freinage automobile » qui nous intéresse, le dictionnaire de données est assez limité puisque nous avons seulement 6 données et 7 événements. Ainsi, nous obtenons :

Demande_freinage = **Signal en entrée**
Rôle : donne un signal proportionnel à l'appui sur la pédale de frein
Type : entrée analogique codée de type entier sur 8 bits, conversion : [0-5V] -> [0-255]
Activation_ABS = **Signal en entrée**
Rôle : donne la position de l'interrupteur ABS
Type : entrée numérique de type booléen
Glissement_roue = **Signal en entrée**
Rôle : donne l'état du glissement de la roue
Type : entrée numérique de type booléen
Affichage_ABS = **Signal en sortie**
Rôle : permet d'allumer le témoin d'ABS (lampe ou LED)
Type : sortie numérique de type booléen
Commande_freinage = **Signal en sortie**
Rôle : permet de commander le frein
Type : sortie analogique de type entier codée sur 8 bits, conversion : [0-255] -> [0-10V]
Mise en marche = **Événement en entrée**
Rôle : indique la mise en marche du véhicule
Type : entrée numérique de type booléen

Niveau_Freinage = **Donnée interne**
Rôle : donne la valeur du freinage à transmettre au frein
Type : entier codée sur 8 bits
Freinage, Pas_de_freinage = **Événements internes**
Rôle : indique l'appui ou non du conducteur sur la pédale de frein
Type : type booléen (*vrai* : freinage demandé et *faux* : pas de freinage)
ABS_activé, ABS_non_activé = **Événements internes**
Rôle : indique l'activation du système ABS par le conducteur
Type : type booléen (*vrai* : ABS activé et *faux* : ABS non activé)
Glissement, Pas_de_glissement = **Événements internes**
Rôle : indique l'état du glissement de la roue
Type : type booléen (*vrai* : glissement et *faux* : pas de glissement)

2.6.2 Représentation graphique des données

Une autre représentation des données et des événements, qui interviennent à tous les stades du modèle, est une représentation graphique de type Jackson. Cette notation est équivalente à la notation BNF. Nous pouvons donner quelques exemples simples correspondant à ceux du paragraphe précédent, soit la représentation d'une séquence ou d'un « ou exclusif » (figure 2.27), soit la représentation d'une itération avec ou sans bornes (figure 2.28).

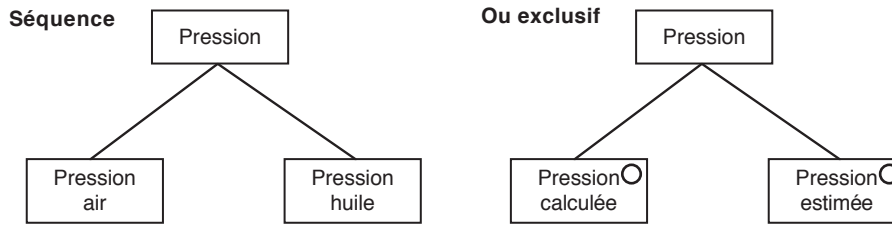

Figure 2.27 – Représentation graphique d'une donnée : séquence et « ou exclusif ».

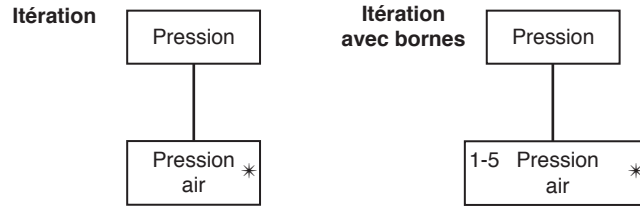

Figure 2.28 – Représentation graphique d'une donnée : itération avec ou sans bornes.

© Dunod – La photocopie non autorisée est un délit.

2.7 Organisation générale de la méthode SA-RT

La description précédente nous a permis de présenter la méthode d'analyse SA-RT dans sa globalité. Les traits essentiels de cette méthodologie hiérarchique descendante résident dans la mise en avant des aspects fonctionnels et comportementaux (exécution) de l'application analysée. L'aspect informationnel, description des données au sens large du terme, est traité de façon minimale.
Le schéma de la figure 2.29 présente l'organisation générale de la méthode SA-RT avec l'enchaînement des différentes étapes et l'ensemble des documents produits. Nous pouvons ainsi décliner :

– Diagramme de contexte : premier diagramme flot de données permettant de décrire l'environnement de l'application à développer.
– Diagramme préliminaire : diagramme flot de données présentant le premier niveau d'analyse fonctionnelle de l'application.
– Diagrammes de décomposition : diagramme flot de données présentant les analyses des processus fonctionnels non primitifs.
– Spécifications des processus fonctionnels primitifs : spécification textuelle des fonctions réalisées par les processus fonctionnels.
– Spécifications des processus de contrôle : diagrammes état/transition décrivant le fonctionnement des processus de contrôle.
– Dictionnaire de données : liste exhaustive des données et des événements utilisés dans la spécification.

Il est évident que l'ensemble de ces diagrammes doit être cohérent par rapport aux deux points de vue :

– Cohérence de l'analyse :
  • Diagramme de contexte, diagramme préliminaire, diagramme de décomposition, spécification d'un processus primitif.
  • Processus de contrôle dans un diagramme « flots de données », diagramme état/transition du processus de contrôle.
– Cohérence de l'enchaînement des différentes étapes
  • Données et événements correspondant entre deux diagrammes flots de données.
  • Événements identiques entre un diagramme flot de données intégrant un processus de contrôle et le diagramme état/transition du processus de contrôle.

Nous pouvons ainsi rappeler les différentes règles d'élaboration des diagrammes flots de données que nous avons vues au cours des paragraphes précédents :

– Un seul processus fonctionnel et pas de processus de contrôle dans un diagramme de contexte.
– Le nombre de processus fonctionnels, composant un diagramme préliminaire, doit être limité de **5** à **9** maximum.
– Les stockages doivent apparaître dans tous les diagrammes où les processus les utilisent.

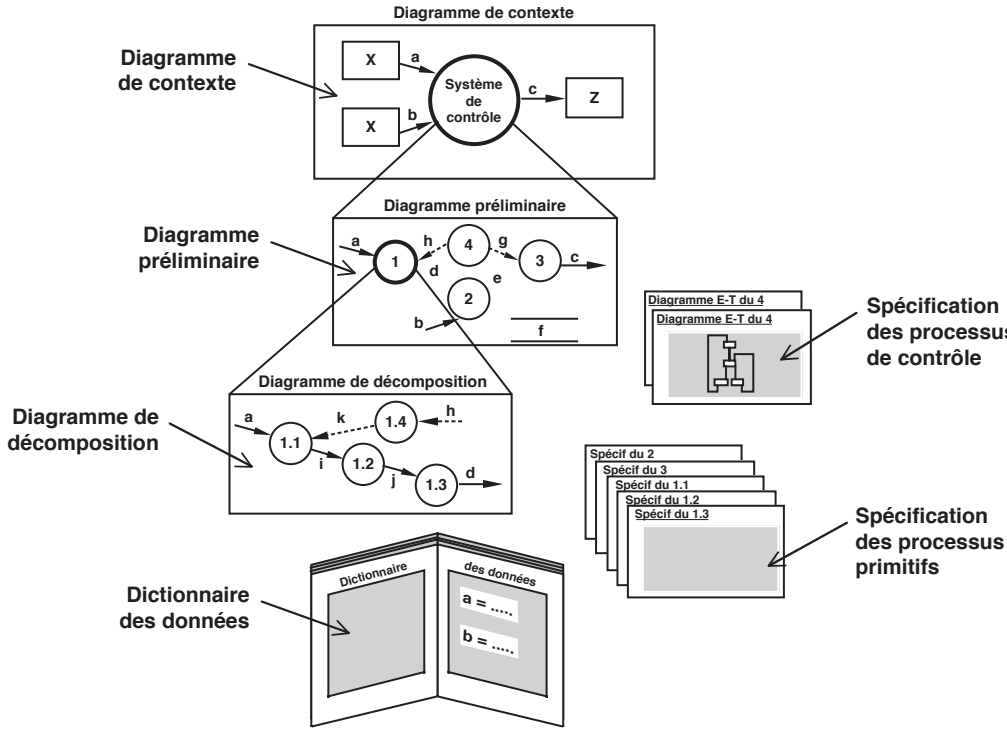

Figure 2.29 – Organisation générale de la méthode SA-RT.

– Un seul ou aucun processus de contrôle par niveau de diagramme : diagramme préliminaire ou diagrammes de décomposition.
– L'ensemble des flots entrants et sortants du processus décomposé doit se retrouver dans le diagramme de décomposition de ce processus avec les mêmes typages.
– La numérotation des différents processus fonctionnels doit intégrer le numéro du processus fonctionnel analysé « N » sous la forme « N.x ».
– Un processus fonctionnel peut ne pas être connecté au processus de contrôle. Dans ce cas, il est supposé être activé au démarrage de l'application et ne jamais s'arrêter.
– Les événements intervenant au niveau de la transition entre les états du diagramme état/transition doivent tous appartenir au diagramme flot de données et réciproquement tous les événements du diagramme flot de données doivent être utilisés dans le diagramme état/transition.
– Pas de flots de données entre les processus fonctionnels et le processus de contrôle.
– Pas de flots d'événements entre les processus fonctionnels.

Les deux dernières règles sont essentielles et, de façon plus générale, nous pouvons décrire les possibilités de relations entre les différentes entités du modèle SA-RT par le tableau 2.3 pour les flots de données et tableau 2.4 pour les flots de contrôle.

© Dunod – La photocopie non autorisée est un délit.

Tableau 2.3 – Liaisons flots de données entre les entités du modèle SA-RT.

De \ Vers	Processus fonctionnel	Processus de contrôle	Unité de stockage	Bord de modèle
Processus fonctionnel	OUI			
Processus de contrôle	NON	NON		
Unité de stockage	OUI	NON	NON	
Bord de modèle	OUI	NON	OUI	–

Tableau 2.4 – Liaisons flots de contrôle entre les entités du modèle SA-RT.

De \ Vers	Processus fonctionnel	Processus de contrôle	Unité de stockage	Bord de modèle
Processus fonctionnel	NON			
Processus de contrôle	OUI	OUI		
Unité de stockage	NON	OUI*	NON	
Bord de modèle	OUI	OUI	OUI*	–

* si le modèle intègre le stockage d'événements.

2.8 Exemples

Nous allons mettre en œuvre cette méthodologie d'analyse SA-RT pour quelques exemples plus complexes que l'exemple décrit jusqu'à présent « système de freinage automobile ». Toutefois, nous nous limitons à la description principale, c'est-à-dire : diagramme de contexte, diagramme préliminaire avec un processus de contrôle et diagramme état/transition du processus de contrôle.

2.8.1 Exemple : gestion de la sécurité d'une mine

■ Présentation du cahier des charges

En plus de la gestion automatisée d'une mine, l'aspect sécurité est le point essentiel de la gestion d'une zone d'extraction de minerai, située en sous-sol, avec une présence humaine. Nous allons limiter notre étude au système de gestion de la sécurité qui concerne principalement le contrôle des deux fluides :

– Eau : les eaux de ruissellement sont évacuées par un ensemble de conduites vers un lieu unique, appelé puisard. Le niveau de ce puisard, qui récupère toutes les

eaux, est contrôlé en permanence avec une évacuation à l'aide de pompes afin de la maintenir entre deux valeurs de niveaux.

– Air : la ventilation des galeries de la mine est effectuée en permanence. Lors de l'extraction, il peut se produire un accès accidentel à une poche de gaz comme du méthane. Fortement toxique, la meilleure protection consiste à procéder à l'évacuation de la zone ou de la mine entière sachant que le volume de méthane n'est, *a priori*, pas connu. D'autre part si le niveau de méthane est élevé, il faut éviter toutes les productions d'étincelles (moteur…).

Nous allons considérer un système simple de gestion de la sécurité de la mine vis-à-vis de ces deux facteurs. Le pilotage de cet ensemble comprend donc les éléments suivants (figure 2.30) :

– un capteur analogique de niveau d'eau, appelé **LS** (*Level Sensor*), pour détecter les deux niveaux limites de régulation, soit le niveau bas (**LLS**, *Low Level Sensor*) et le niveau haut (**HLS**, *High Level Sensor*) ;
– une pompe à eau à débit réglable permettant l'évacuation du puisard ;
– un capteur analogique du taux de méthane contenu dans l'air, appelé **MS** (*Methane Sensor*) ;
– une interface vers l'opérateur pour affichage de l'alarme.

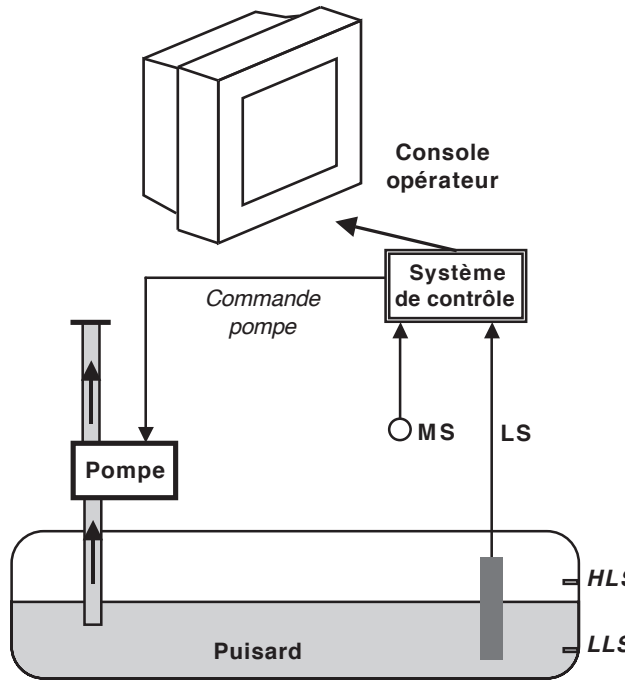

Figure 2.30 – Représentation schématique de l'application de la gestion de l'aspect sécurité d'une mine.

© Dunod – La photocopie non autorisée est un délit.

La mine doit donc fonctionner tant que la sécurité maximale peut être maintenue. Ainsi, les indications générales ou règles de fonctionnement sont les suivantes :

– Règle 1 : La pompe doit être mise en route si le niveau d'eau dépasse le niveau maximum (**LS** > **HLS**). La pompe s'arrête dès que le niveau descend en dessous de la valeur inférieure (**LS** < **LLS**).

– Règle 2 : Une alarme doit être lancée vers la console de l'opérateur dès que le niveau limite du capteur **MS** est franchi afin de pouvoir opérer une évacuation de la mine. Cette valeur limite est appelée **MS_L1** (*Methane Sensor Level 1*).

– Règle 3 : La pompe ne doit pas fonctionner quand le niveau du capteur de méthane (**MS**) est supérieur à une limite fixée **MS_L2** (*Methane Sensor Level 2*) avec la condition **MS_L2** > **MS_L1** afin d'éviter les risques d'explosion.

Dans cet exemple d'application simple, nous sommes donc en présence de deux lignes de régulation : le contrôle du niveau de l'eau dans le puisard (règle 1) et le contrôle du taux de méthane dans l'air (règle 2). L'interaction entre ces deux régulations se situe au niveau de la commande de la pompe pour l'évacuation de l'eau du puisard dont le fonctionnement est lié non seulement au niveau d'eau, mais aussi au taux de méthane (règle 3).

■ Analyse SA-RT

□ Diagramme de contexte

Le premier niveau d'analyse consiste à élaborer le diagramme de contexte de l'application. Ce diagramme, représenté sur la figure 2.31, intègre les quatre bords de modèle correspondant aux deux entrées ou capteurs (capteur de méthane, capteur de niveau d'eau) et aux deux sorties ou actionneurs (pompe, console opérateur). Il est souvent nécessaire d'ajouter un événement de démarrage « *Mise_sous_tension* » délivré par un bord de modèle « interrupteur ». Le processus fonctionnel initial 0 « Gérer sécurité mine » constitue l'application à réaliser. En résumé, en plus de

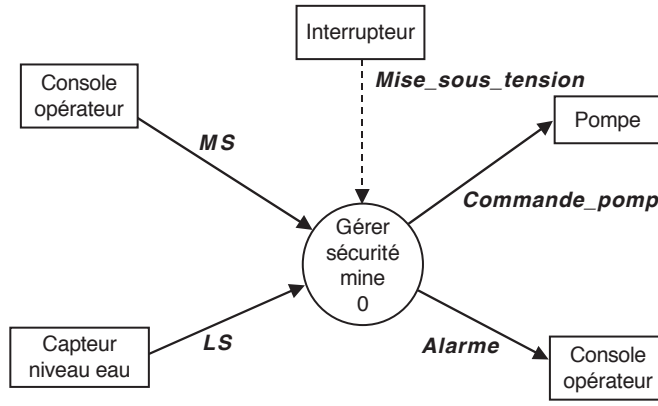

Figure 2.31 – Analyse SA-RT de l'application de la gestion de l'aspect sécurité d'une mine : Diagramme de contexte.

l'événement « *Mise_sous_tension* », nous avons quatre flots de données : deux entrants (*MS*, *LS*) et deux sortants (*Commande_pompe*, *Alarme*). L'ensemble de ces flots doit se retrouver dans le diagramme préliminaire : premier niveau d'analyse du processus fonctionnel 0.

□ Diagramme préliminaire et diagramme état/transition

Le diagramme préliminaire, présenté sur la figure 2.32, donne une analyse ou décomposition fonctionnelle du processus fonctionnel initial 0 « Gérer sécurité mine ». Cette analyse fait apparaître quatre processus fonctionnels de base et un processus de contrôle permettant de séquencer l'ensemble. Nous pouvons vérifier la cohérence des flots de données ou d'événements entrants ou sortants par rapport au diagramme de contexte.

Les processus 1 (Acquérir capteur méthane) et 4 (Afficher alarme) concernent le contrôle du taux de méthane dans l'air avec un processus d'acquisition et de comparaison aux niveaux de consignes (MS_L1, MS_L2) stockés dans une mémoire de stockage « *Niveaux_consignes_méthane* » et un processus de commande pour déclencher l'alarme.

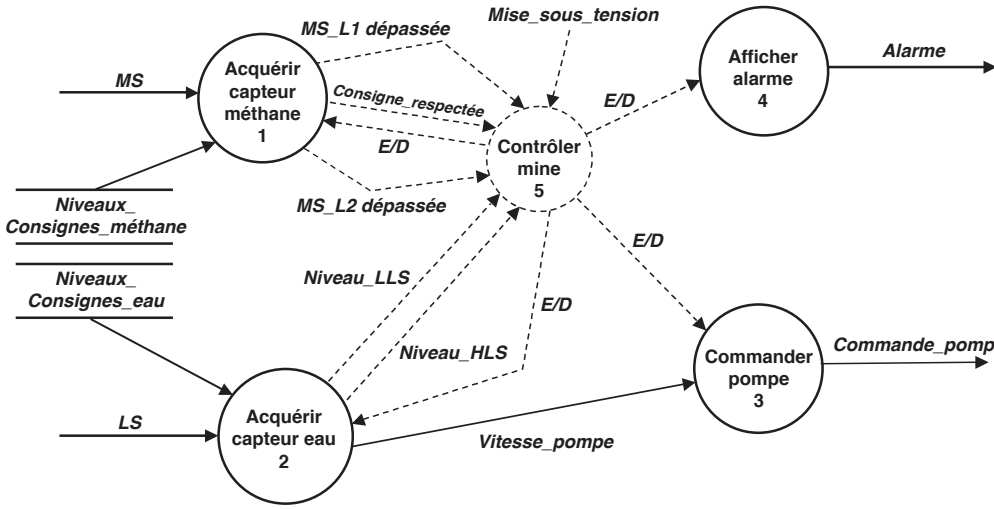

Figure 2.32 – Analyse SA-RT de l'application de la gestion de l'aspect sécurité d'une mine : Diagramme préliminaire.

Les processus 2 (Acquérir capteur eau) et 3 (Commander pompe) concernent la régulation du niveau d'eau dans le puisard avec un processus d'acquisition et de comparaison aux niveaux de consignes (LLS, HLS) stockés dans une mémoire de stockage « *Niveaux_consignes_eau* » et un processus de commande de la pompe. Contrairement à la chaîne de régulation du taux de méthane où les deux processus fonctionnels sont indépendants en termes de données, les deux processus fonctionnels de la chaîne de régulation du niveau d'eau sont liés par le transfert d'une donnée « *Vitesse_pompe* » qui est, par exemple, proportionnelle à la hauteur du niveau d'eau.

© Dunod – La photocopie non autorisée est un délit.

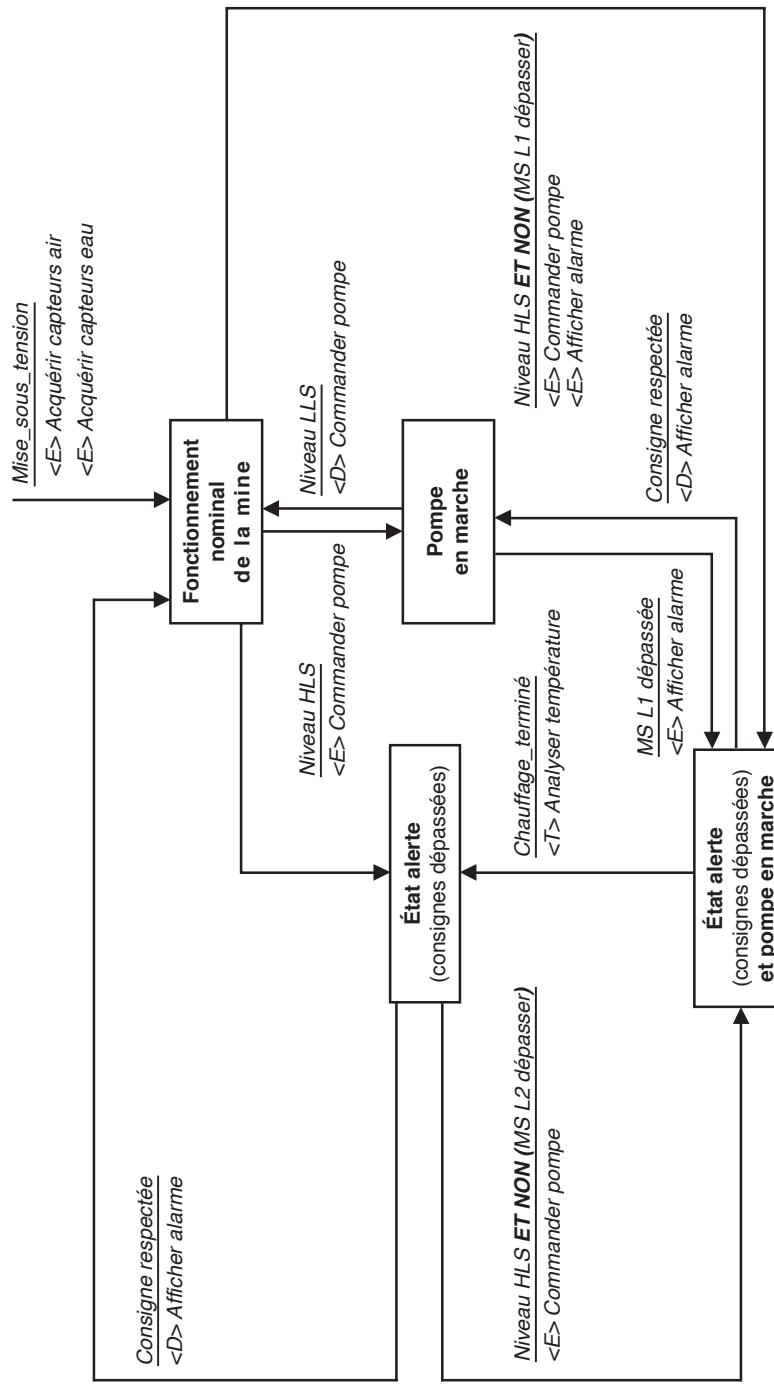

Figure 2.33 – Analyse SA-RT de l'application de la gestion de l'aspect sécurité d'une mine : Diagramme état/transition.

Le processus de contrôle est lié aux différents processus fonctionnels par des événements qui sont mis en place en même temps que la réalisation du diagramme état/transition de la figure 2.33. Ce diagramme état/transition, description du fonctionnement du processus de contrôle 5 (Contrôler mine), comprend quatre états :

– fonctionnement nominal de la mine (niveau d'eau inférieur à HLS et taux de méthane inférieur à MS_L1) ;
– pompe en marche (niveau d'eau supérieur à LLS) ;
– état alerte (consigne MS_L1 dépassée et niveau d'eau inférieur à HLS, ou consigne MS_L2 dépassée et niveau d'eau quelconque) ;
– état alerte (consigne MS_L1 dépassée) et pompe en marche.

Nous pouvons remarquer que, dans ce diagramme état/transition complexe, nous avons utilisé dans certains cas des combinaisons logiques de deux événements pour passer d'un état à l'autre, par exemple « *MS_L2_dépassée* **OU** *Niveau_LLS* » pour passer de l'état « État alerte et pompe en marche » à l'état « État alerte ».
D'autre part, ce diagramme état/transition fait l'hypothèse pour la surveillance du taux de méthane que les événements « *MS_L1_dépassée* » et « *MS_L2_dépassée* » se produisent toujours dans l'ordre cité et, lors du retour à la situation normale, l'événement « *Consigne_respectée* » est émis.

2.8.2 Exemple : pilotage d'un four à verre

■ Présentation du cahier des charges

Un four pour la fabrication du verre fonctionne de façon continue aussi bien du point de vue de l'approvisionnement en matières premières (sables) que du point de vue de l'utilisation du produit (verre). En effet, le four doit rester en fonctionnement permanent avec un niveau toujours suffisant de matières fondues à température constante, une évacuation du trop plein étant prévue en cas d'attente prolongée d'utilisation du verre. Cette application a été simplifiée afin de limiter l'analyse.
Le contrôle-commande de cette application est fait par l'intermédiaire de **3 capteurs** (capteur de température, capteur de niveau du four et capteur de détection de l'arrivée de matières premières) et de **2 actionneurs** (commande d'approvisionnement en matières premières, chauffage du four). Nous avons donc comme précédemment deux chaînes de régulation : température et approvisionnement en sable.
Une représentation schématique de cette application est présentée sur la figure 2.34.
L'acquisition de la température, à partir de capteurs de type thermocouple, doit se faire à des moments réguliers en utilisant l'horloge temps réel interne du système. Le traitement du « signal température » permet de faire un calcul précis de la température (approximation polynomiale correspondant au thermocouple) et lance une tâche de commande de chauffage si la température du four est inférieure à la température de consigne. Le principe de chauffage du four se fait à partir d'ondes hautes fréquences pulsées. Ce chauffage est effectué pendant un temps fixé court mais avec une intensité qui peut dépendre du chauffage nécessaire.
L'acquisition du niveau de matière est liée à l'interruption générée de façon apériodique par les tombées successives mais non régulières du sable détectées par le capteur. Cette détection est réalisée par un capteur tout ou rien comme une cellule photo-

© Dunod – La photocopie non autorisée est un délit.

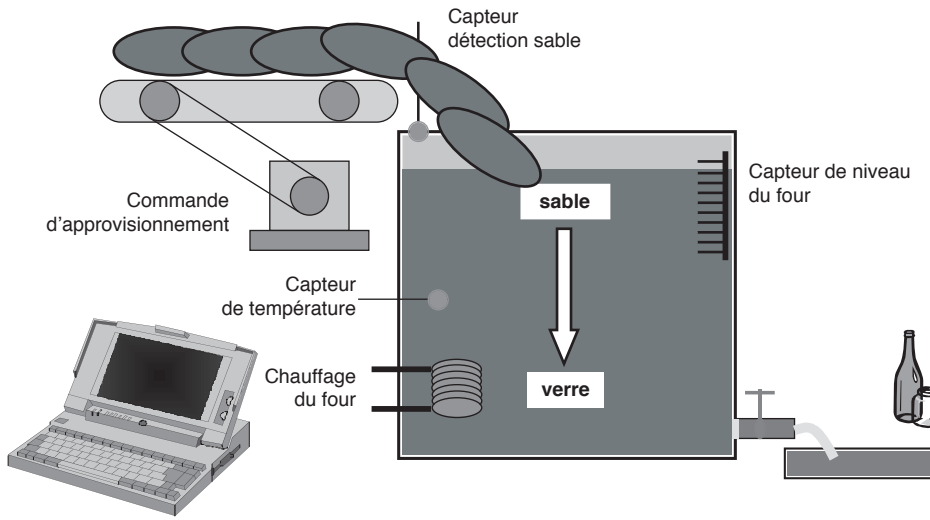

Figure 2.34 – Représentation schématique de l'application de la gestion d'un four à verre.

électrique. Le paramètre « niveau de matière » va impliquer l'approvisionnement ou non en matières premières en commandant la vitesse d'approvisionnement en fonction du paramètre « niveau du four ». Mais cette régulation dépend aussi de la valeur de la température. En effet, afin d'éviter une solidification du sable fondu, il est nécessaire de limiter l'apport en matières premières si la température n'est pas suffisante.

■ Analyse SA-RT

□ Diagramme de contexte

La première étape d'analyse consiste à élaborer le diagramme de contexte de l'application. Ce diagramme, représenté sur la figure 2.35, intègre les six bords de modèles correspondant aux trois entrées ou capteurs (capteur de température – thermocouple, capteur de niveau de sable, capteur tout ou rien de détection d'arrivée du sable) et aux deux sorties ou actionneurs (approvisionnement en sable, commande du four de chauffage). Le dernier bord de modèle correspond à la console opérateur qui fournit les deux événements : « *Arrêt* » et « *Marche* ». Ces événements ne sont utilisés que pour le démarrage de l'application et éventuellement son arrêt. Le processus fonctionnel initial 0 « Piloter four à verre » constitue l'application à étudier. En résumé, en plus des deux événements précédemment cités, nous avons cinq flots de données : trois flots entrants (*Température*, *Niveau_sable*, *Arrivée_sable*) et deux flots sortants (*Sable*, *Chauffage*). L'ensemble de ces flots doit se retrouver dans le diagramme préliminaire : premier niveau d'analyse du processus fonctionnel 0.

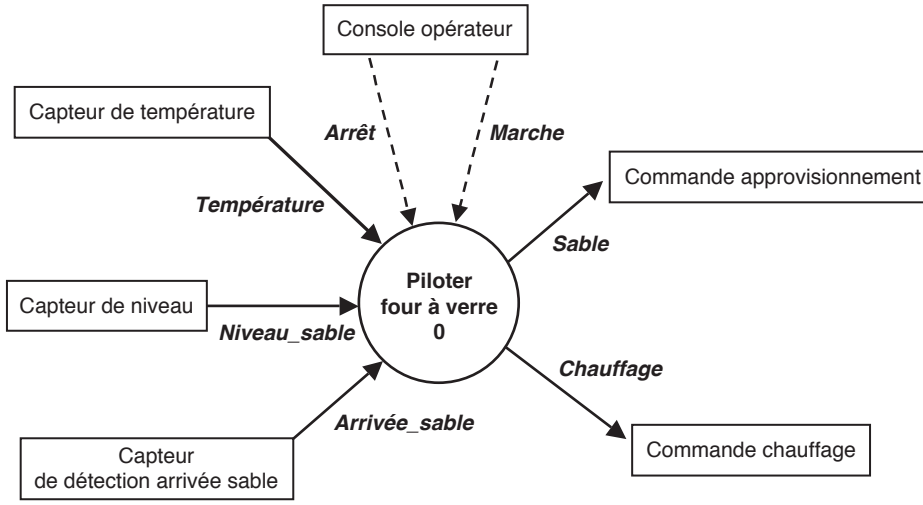

Figure 2.35 – Analyse SA-RT de l'application de la gestion d'un four à verre : Diagramme de contexte.

□ Diagramme préliminaire et diagramme état/transition

Le diagramme préliminaire, présenté sur la figure 2.36, donne une analyse ou décomposition fonctionnelle du processus fonctionnel initial 0 « Piloter four à verre ». Cette analyse fait apparaître six processus fonctionnels de base et un processus de contrôle permettant de séquencer l'ensemble. Nous pouvons vérifier la cohérence des flots de données ou d'événements entrants ou sortants par rapport au diagramme de contexte.

Les différents processus fonctionnels correspondent aux deux chaînes de régulation : température (processus fonctionnels 1 à 3) et approvisionnement en sable (processus fonctionnels 4 à 6). La régulation de la température suit exactement la décomposition fonctionnelle générique que nous avons vue (figure 2.14). Ainsi, les trois processus fonctionnels de base existent : acquisition (1 – Acquérir température), traitement (2 – Analyser température) et commande (3 – Chauffer four). Les deux unités de stockage sont utilisées dans les deux cas classiques : soit pour la mémorisation d'une constante (*Température_consigne*) soit pour une donnée partagée (*Température_mesurée*). Dans ce dernier cas, nous pouvons noter que l'unité de stockage est utilisée deux fois dans le diagramme et donc comporte une « * ».

Dans le cas de la régulation du niveau du sable, les trois processus fonctionnels mis en œuvre ne correspondent pas exactement au modèle de décomposition générique précédent ; mais nous avons uniquement deux processus fonctionnels : acquisition (5 – Acquérir niveau), traitement et commande (6 – Analyser besoin sable). Nous pouvons remarquer que ce dernier processus utilise pour élaborer la décision de commande trois données : *Niveau_consigne* (constante placée dans une unité de stockage), *Niveau_mesurée* (donnée fournie directement par le processus 5) et *Température_mesurée* (unité de stockage partagée avec la chaîne de régulation de la

© Dunod – La photocopie non autorisée est un délit.

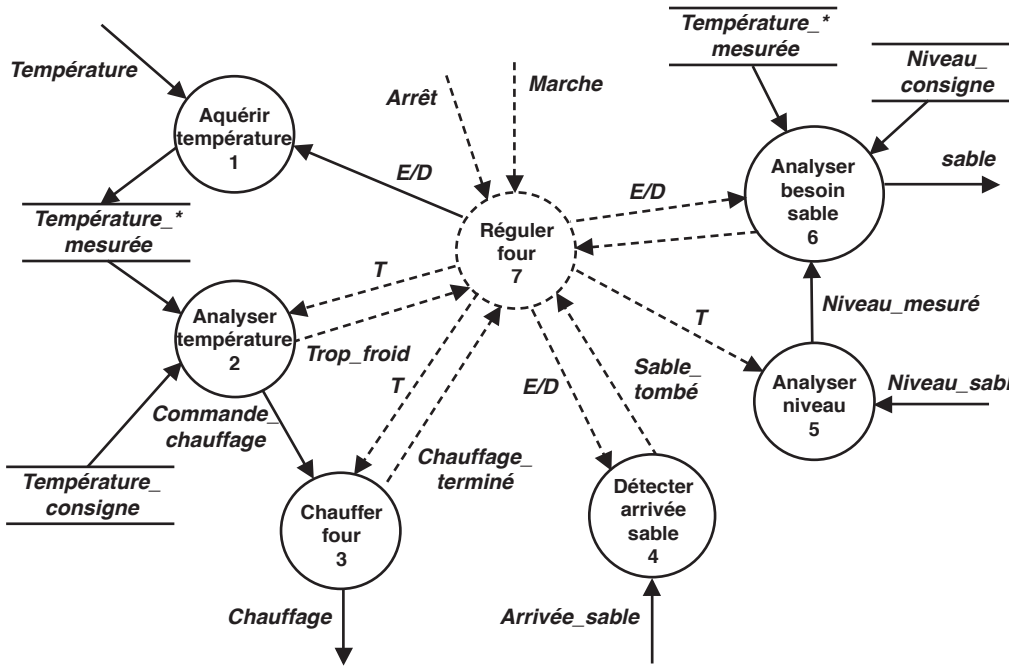

Figure 2.36 – Analyse SA-RT de l'application de la gestion d'un four à verre : Diagramme préliminaire.

température). Le troisième processus utilisé dans cette régulation de niveau (4 – Détecter arrivée sable) correspond en fait à un processus de déclenchement qui est activé sur interruption liée à la donnée « *Arrivée_sable* ».

Le processus de contrôle (7 – Réguler four) est lié aux différents processus fonctionnels par des événements qui sont mis en place en même temps que la réalisation du diagramme état/transition de la figure 2.37. Ce diagramme état/transition, description du fonctionnement du processus de contrôle 7, comprend quatre états :

– état repos (attente de fonctionnement ou arrêt du four) ;
– fonctionnement nominal (four en fonctionnement et attente des événements pour effectuer les régulations soit de température, soit de niveau de sable) ;
– chauffage du four (la température de consigne n'étant pas atteinte, un cycle de chauffage est lancé) ;
– régulation du niveau de sable (une interruption due à la chute d'un paquet de sable lance la régulation du niveau du four).

Nous pouvons remarquer que, dans ce diagramme état/transition simple, nous avons utilisé un état stationnaire (Fonctionnement nominal) dans lequel l'application reste et deux autres états plus transitoires dans lesquels l'application se situe en cas de régulation de l'un des deux paramètres.

D'autre part nous pouvons noter la difficulté concernant la régulation du niveau de sable ; en effet l'ajustement de l'approvisionnement en sable ne peut se produire

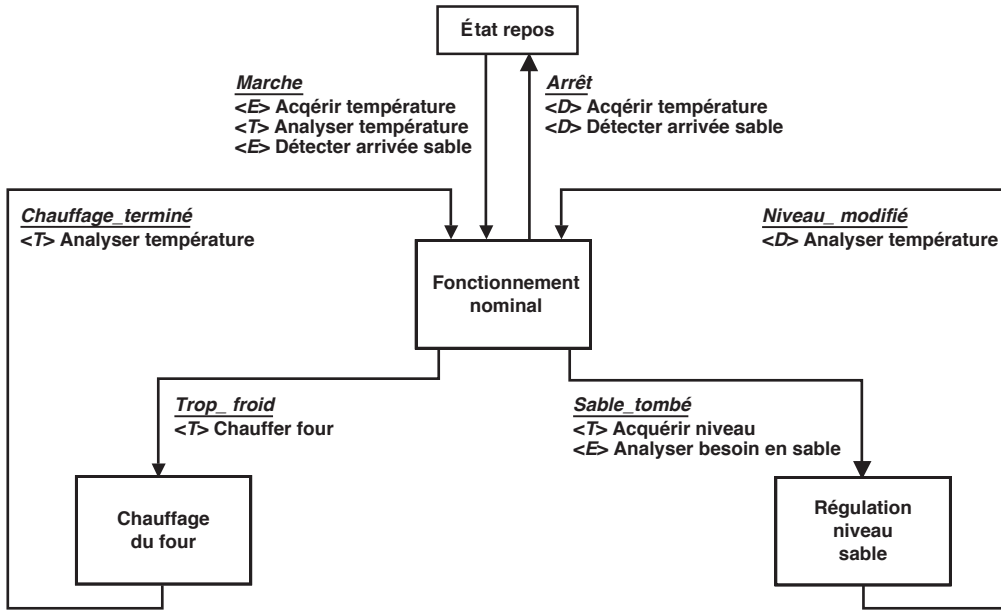

Figure 2.37 – Analyse SA-RT de l'application de la gestion d'un four à verre : Diagramme état/transition.

que si la mesure du niveau est effectuée, c'est-à-dire que le processus fonctionnel « 5 – Acquérir niveau » est activé. Pour cela, il est nécessaire que la donnée « *Arrivée_ sable* » se produise pour qu'elle soit transformée en événement « *Sable_tombé* » par le processus fonctionnel « 4 – Détecter arrivée sable ». Dans le cas contraire, la régulation du niveau de sable ne peut s'exécuter, conduisant à un dysfonctionnement de l'application. Ce problème est étudié au niveau de la conception décrite dans le chapitre 3.

2.8.3 Exemple : Commande d'un moteur à combustion

■ Présentation du cahier des charges

Le contrôle-commande d'un moteur à combustion est devenu de plus en plus complexe au fur et à mesure des besoins en rendement, consommation et pollution. En effet, en plus de fournir une puissance mécanique en fonction de l'appui sur la pédale de l'accélérateur, il est nécessaire d'avoir une optimisation de la consommation du véhicule en contrôlant les différents paramètres de la combustion (pression air, température, mélange, allumage, etc.). D'autre part, la pollution du véhicule doit être minimisée. Les différents états du moteur peuvent être présentés au conducteur afin de le prévenir de sa consommation excessive ou de la pollution de son véhicule pour qu'il effectue un changement dans son mode de conduite. L'ensemble de ces paramètres moteur est géré par un calculateur spécifique. Actuellement, un véhicule possède de nombreux calculateurs dédiés à des fonctions très diverses (freinage ABS, gestion moteur, éclairage, climatisation, etc.). Ces différents calculateurs commu-

© Dunod – La photocopie non autorisée est un délit.

niquent entre eux par un bus de terrain comme CAN (voir chapitre 4) afin de partager des informations et gérer ainsi le véhicule de façon cohérente.
Cette application de commande d'un moteur à combustion est représentée schématiquement sur la figure 2.38. Le contrôle-commande de cette application est fait par l'intermédiaire de **sept capteurs** (pédale accélérateur, température air, pression air, température eau, rotation vilebrequin et deux capteurs de pollution) et de **quatre actionneurs** (injection essence, allumage, admission air, réinjection gaz échappement ou brulés). Le calculateur est donc aussi relié au bus de communication CAN.

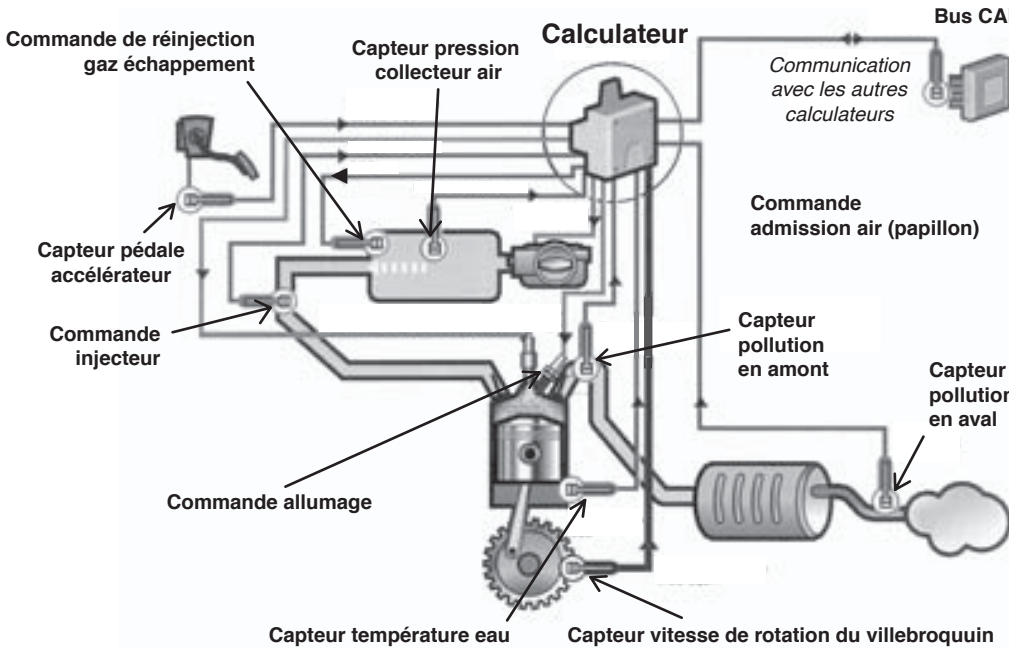

Figure 2.38 – Représentation schématique de l'application de la commande d'un moteur à combustion.

Excepté les deux capteurs de pollution amont et aval, la loi de régulation du moteur à combustion prend en compte l'ensemble des données d'entrée et élabore les différentes sorties de commande. Nous n'analyserons pas cette loi de commande qui présente une relative complexité et correspond à une spécificité constructeur pour un type de moteur donné. Ainsi, l'élaboration de la loi de commande du moteur à combustion est considérée comme une « boîte noire » fonctionnelle qui utilise un ensemble de données en entrée (*Paramètres_moteur_entrée*) et qui fournit des données en sortie (*Paramètres_moteur_sortie*). Les données sur la pollution ne sont pas utilisées dans ce calculateur ; mais elles sont fournies par l'intermédiaire du réseau de communications à un autre calculateur, par exemple, pour affichage.

■ Analyse SA-RT

□ Diagramme de contexte

Le diagramme de contexte de l'application est représenté sur la figure 2.39. Il donne les 11 bords de modèles correspondant aux sept entrées ou capteurs et aux quatre sorties ou actionneurs, énumérés précédemment. Nous avons ajouté un bord de modèle correspondant à la connexion au bus CAN et un bord de modèle représentant l'action du conducteur. Le dernier bord de modèle fournit les deux événements : « *Arrêt* » et « *Marche* ». Pour le bord de modèle du bus CAN, nous supposons que les communications bidirectionnelles sont identifiées : en sortie (*Com_S*) et en entrée (*Com_E*).

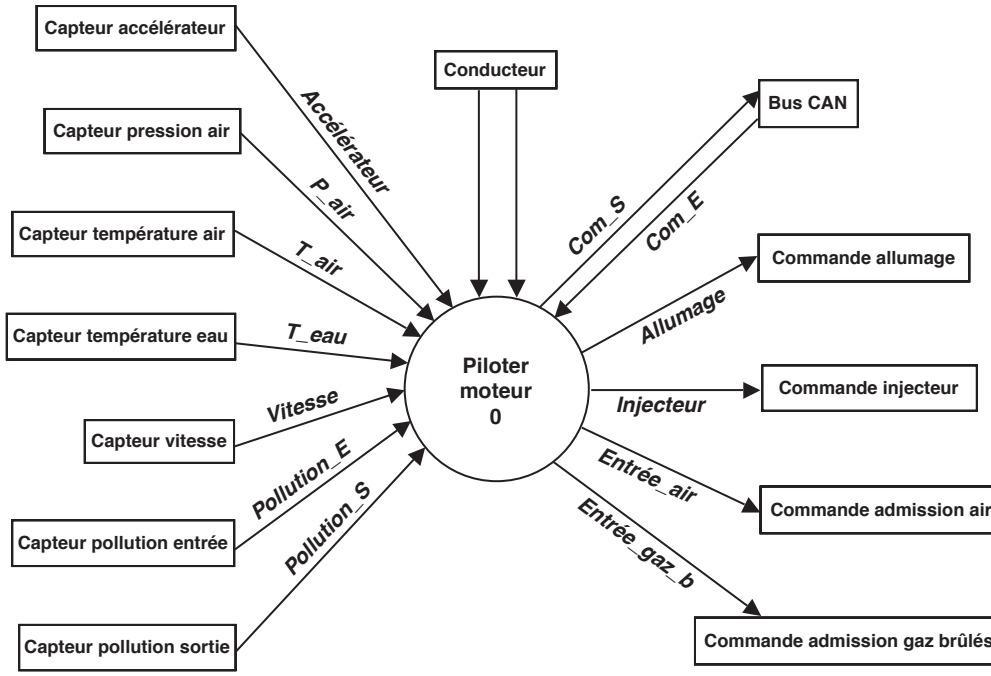

Figure 2.39 – Analyse SA-RT de l'application de la commande d'un moteur à combustion : Diagramme de contexte.

Le processus fonctionnel initial 0 « Piloter moteur » constitue l'application à étudier. En résumé, en plus des deux événements précédemment cités (*Arrêt*, *Marche*), nous avons 13 flots de données : huit entrants (*Accélérateur*, *P_air*, *T_air*, *T_eau*, *Vitesse*, *Pollution_E*, *Pollution_S*, *Com_E*) et cinq sortants (*Allumage*, *Injecteur*, *Entrée_air*, *Entrée_gaz_b*, *Com_S*). L'ensemble de ces flots doit se retrouver dans le diagramme préliminaire : premier niveau d'analyse du processus fonctionnel 0.

© Dunod – La photocopie non autorisée est un délit.

□ Diagramme préliminaire et diagramme état/transition

Le diagramme préliminaire, présenté sur la figure 2.40, donne une analyse ou décomposition fonctionnelle du processus fonctionnel initial 0 « Piloter moteur ». Cette analyse fait apparaître neuf processus fonctionnels de base et un processus de contrôle permettant de séquencer l'ensemble. Malgré le regroupement de certaines fonctions d'acquisition ou de commande, le nombre de processus fonctionnel de cette première décomposition correspond à la limite fixée pour avoir une bonne compréhension globale du diagramme flots de données. Nous pouvons remarquer que les processus fonctionnels 6, 7 et 8 auraient pu être regroupés car ils sont déclenchés en même temps. Ce travail de regroupement est effectué en partie au niveau de la conception étudiée dans le chapitre 3.

Nous pouvons vérifier la cohérence des flots de données ou d'événements entrants ou sortants par rapport au diagramme de contexte de la figure 2.39. Afin d'augmenter la lisibilité du diagramme, nous avons utilisé des unités de stockage de données complexes, ou encore appelées énumérées, dans le sens où elles intègrent un ensemble ou groupe de données. Comme nous l'avons déjà présenté, deux unités de stockages renferment les paramètres en entrée du moteur, excepté les données pollutions (*Paramètres_moteur_entrée*) et les paramètres en sortie du moteur (*Paramètres_moteur_sortie*).

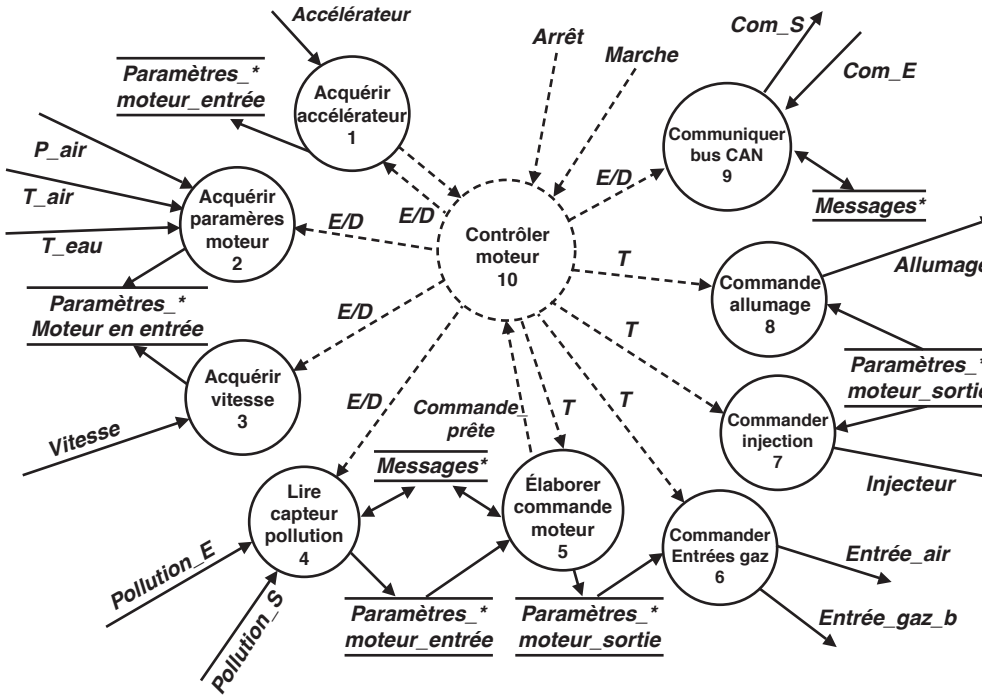

Figure 2.40 – Analyse SA-RT de l'application de la commande d'un moteur à combustion : Diagramme préliminaire.

Une autre unité de stockage est dédiée aux messages (*Messages*). Cette unité de stockage est un peu particulière par rapport aux autres unités de stockage. En effet, en entrée, nous avons une file gérée de façon FIFO ou à priorité et, en sortie, la file est gérée de manière FIFO.
Remarquons que toutes ces unités de stockages sont dupliquées au niveau de ce diagramme et donc notées avec une « * ».
Le processus de contrôle (10 – Contrôler moteur) est lié aux différents processus fonctionnels par des événements qui sont mis en place en même temps que la réalisation du diagramme état/transition de la figure 2.41. Nous avons simplifié le fonctionnement de ce processus de contrôle en supposant que les processus fonctionnels d'acquisition étaient lancés au début de l'exécution (1, 2, 3 et 4) et fournissaient les données de façon périodique ; en particulier le processus fonctionnel « 1 – Acquérir accélérateur » fournit régulièrement l'événement *État_accélérateur* qui déclenche l'élaboration de la loi de commande à partir de toutes les données d'entrée acquises. Lorsque la commande est prête, le processus « 5 – Élaborer commande moteur » déclenche l'ensemble des processus fonctionnels de commande (6, 7 et 8). Enfin, le processus fonctionnel 9 de communication gère périodiquement les messages au niveau réception et émission.

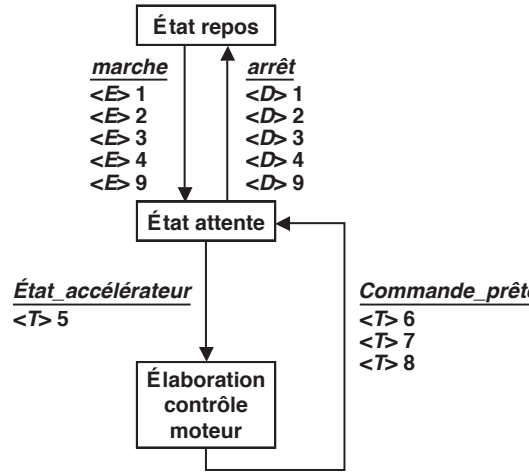

Figure 2.41 – Analyse SA-RT de l'application de la commande d'un moteur à combustion : Diagramme état/transition.

Nous pouvons remarquer que, dans ce diagramme état/transition très simple, nous avons utilisé uniquement deux états actifs en plus de l'état repos : Un état attente qui est périodiquement interrompu par l'événement « *État_accélérateur* » et un état correspondant à l'élaboration de la loi de commande du moteur à combustion.

© Dunod – La photocopie non autorisée est un délit.

2.9 Extensions de la méthode SA-RT

2.9.1 Extensions de la syntaxe graphique de la méthode SA-RT

Comme nous l'avons déjà dit précédemment en introduction de cette méthode d'analyse SA-RT, la description non formelle des applications est très abstraite et ne permet pas dans beaucoup de cas de spécifier précisément certaines fonctions ou certaines données. D'autre part, la méthode n'étant pas normalisée, il est tout à fait possible d'étendre la méthode afin d'offrir plus d'expressivité. Aussi de nombreuses entreprises ont adapté la méthode à leurs besoins et ajouté des éléments graphiques pour exprimer des spécifications plus précises. Nous présentons dans ce chapitre deux de ces extensions très utilisées : une extension concernant les flots de données et une extension pour les événements. Ces extensions sont intégrées dans la méthode « ESML++ » développée dans le cadre de la société Boeing. Cette méthode, utilisée pour les développements des applications de contrôle-commande du domaine de l'avionique, est une méthode SA-RT enrichie.

■ Extensions liées aux flots de données

Une syntaxe graphique plus complète permet de préciser les flots de données discrets ou continus. Ainsi, l'analyse des flots de données permet et, donc oblige, de distinguer un flot de données discret (arc orienté simple comme précédemment) et un flot de données continu (arc orienté avec une double flèche). Mais, dans ce cas, où la richesse d'expression des données véhiculées par ces flots est augmentée, il est nécessaire de préciser la sémantique attachée à cette nouvelle description. Or deux types de sémantique peuvent être attachés aux flots de données (figure 2.42) :

– Sémantique 1 :
  • Flot de donnée discret : valeur discrète de donnée (type booléen) ;
  • Flot de donnée continu : valeur continue de la donnée (type entier ou réel).

– Sémantique 2 :
  • Flot de donnée discret : donnée consommable ou lisible une fois (existence de la donnée à des temps discrets) ;

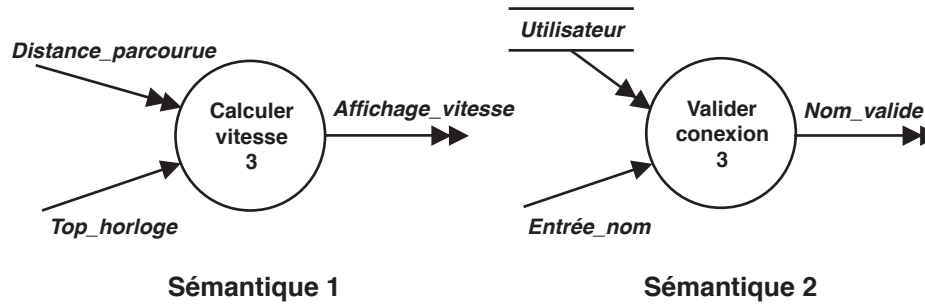

Figure 2.42 – Extension de la syntaxe graphique des flots de données selon deux sémantiques.

- Flot de donnée continu : donnée continuellement disponible (existence permanente de la donnée).

■ Extensions liées aux événements

Comme pour les flots de données, une syntaxe graphique plus complète permet de préciser les flots d'événements discrets ou continus. Ainsi, l'analyse des flots d'événements permet et, donc oblige, de distinguer un flot d'événements discret (arc orienté simple comme précédemment) qui concerne des événements consommables ou lisibles une seule fois (existence à des temps discrets). De même un flot d'événements continu (arc orienté avec une double flèche) décrit un événement continuellement disponible (existence permanente).
Il est important de noter que les événements prédéfinis « E, D, T » sont des événements discrets, envoyés à chaque fois pour activer ou arrêter un processus fonctionnel. En revanche, un flot d'événements continu permet au processus de contrôle de tester en permanence le résultat d'un processus fonctionnel qui est indépendant du processus de contrôle (figure 2.43).

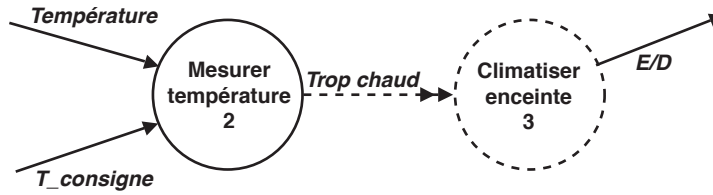

Figure 2.43 – Extension de la syntaxe graphique des flots d'événements.

De même, pour les stockages de données, le **stockage d'événements** modélise le besoin de mémorisation d'un événement de façon à ce que son occurrence soit utilisée plusieurs fois. Comme le flot d'événements auquel il est étroitement associé, il est nommé par une étiquette ou label explicite formé de :

Étiquette_Stockage_événements = nom (+ qualifiant)

et il est représenté par deux traits parallèles en pointillés (figure 2.44).

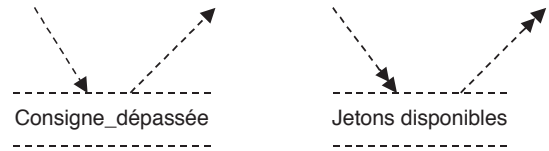

Figure 2.44 – Extension de la syntaxe graphique de SA-RT : Stockages d'événements.

© Dunod – La photocopie non autorisée est un délit.

2.9.2 Méthode SA-RT et méthode formelle : projet IPTES

■ **Introduction**

La méthode SA-RT, qui vient d'être présentée dans les sections précédentes, est une méthode de spécification simple avec un langage graphique très compréhensible par un utilisateur. Cette facilité d'expression a un inconvénient majeur, celui de pouvoir conduire à des ambiguïtés. En effet, lors de l'analyse d'une application, basée sur la méthode SA-RT, la réalisation des différents diagrammes flots de données reflète l'idée du concepteur au travers d'un langage simple, limité et très abstrait. Ainsi, deux utilisateurs peuvent avoir exprimé deux analyses différentes sous la forme d'un même diagramme flots de données SA-RT. Pour lever cette ambiguïté, il est nécessaire de disposer d'un outil formel qui offre un moyen d'expression précis et qui peut être validé.

Ainsi, il est possible d'utiliser en complément ou en parallèle de cette méthodologie SA-RT, un modèle formel comme des automates à états finis ou des réseaux de Petri. Ces modèles présentent l'avantage d'être rigoureux au niveau de l'expression et d'être analysable mathématiquement. En revanche, ces modèles sont en général complexes et d'une lisibilité faible. Aussi le projet européen IPTES (*Incremental Prototyping Technology for Embedded Real-Time Systems*), dont les résultats ont été présentés en 1998, a voulu donner une sémantique à la méthode SA-RT basée sur les réseaux de Petri. Nous pouvons schématiser cette association par le graphique de la figure 2.45. Ainsi, la méthode SA-RT peut être vue comme l'interface de la méthode d'analyse globale avec un langage graphique de haut niveau pour exprimer la spécification, c'est l'interface utilisateur. Directement liée à ce modèle graphique, nous trouvons la correspondance dans le modèle formel apporté par les réseaux de Petri qui constituent le noyau fonctionnel, c'est-à-dire la partie permettant d'exécuter ce modèle graphique représentant la spécification.

L'utilisateur décrit sa spécification avec la syntaxe graphique SA-RT et, de façon automatique, le modèle formel, basé sur les réseaux de Petri, se construit. Cela permet ensuite une analyse mathématique de la spécification : simulation et vérification. Pour atteindre ce but, il est nécessaire d'avoir une traduction unique d'un modèle vers l'autre.

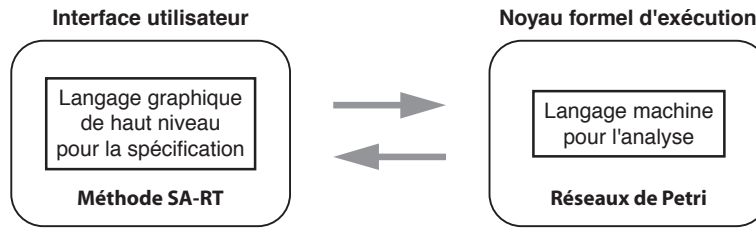

Figure 2.45 – Représentation schématique de l'association d'un modèle formel avec la méthode SA-RT.

■ Principes généraux

La méthode SA-RT, utilisée pour ce projet, correspond à celle décrite précédemment avec toutefois l'extension à des flots de données discrets et continus suivant la sémantique 2 (§ 2.9.1).
Les réseaux de Petri ont de nombreuses variantes : abréviations et extensions. Les réseaux de Petri choisis pour ce projet sont des réseaux de Petri autonomes ayant les principales caractéristiques suivantes :

– Association des données et des événements aux jetons.
– Jeton correspondant à une variable typée.
– Places typées.
– Association à chaque transition :
  • Un prédicat lié aux places en amont de la transition ;
  • Une action.
– Transition temporisée suivant le modèle Merlin et Farber $[\partial_{min}, \partial_{max}]$ où t_0 est la date de sensibilisation de la transition :
  • Franchissement de la transition après le temps $t_0 + \partial_{min}$;
  • Franchissement de la transition avant le temps $t_0 + \partial_{max}$.

En partant de ces hypothèses sur les modèles utilisés, nous avons donc une traduction ou correspondance entre tous les éléments syntaxiques du modèle SA-RT et un réseau de Petri. Toutefois, afin de simplifier cette présentation, les éléments traduits font abstraction de la modélisation temporisée des transitions qui a été présentée précédemment, c'est-à-dire que toutes les transitions présentées dans la suite sont considérées comme des transitions immédiates, soit [0,0] avec la notation du modèle Merlin et Farber.
En premier, pour les flots de données, nous trouvons le modèle d'un flot de données discret, qui sont des données consommables ou lisibles une fois, sur la figure 2.46 et le modèle d'un flot de données continu, qui sont des données continuellement disponibles, sur la figure 2.47. Ces deux représentations réseaux de Petri nécessitent deux places (Vide et Valeur) et trois transitions (*Écrire_donnée_vide*, *Écrire_donnée* et *Lire_donnée*). La place « Vide » désigne une donnée qui n'a jamais été produite

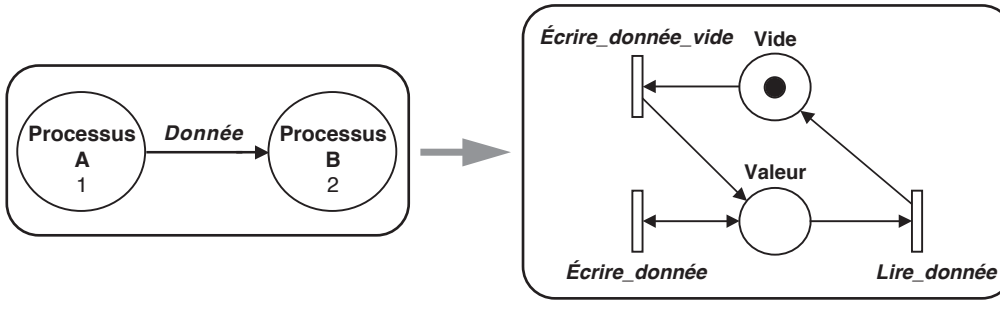

Figure 2.46 – Traduction d'un flot de données discret de la méthode SA-RT par un réseau de Petri.

© Dunod – La photocopie non autorisée est un délit.

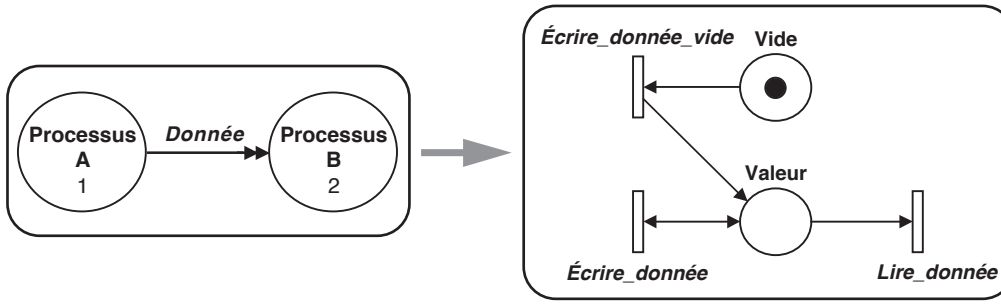

Figure 2.47 – Traduction d'un flot de données continu de la méthode SA-RT par un réseau de Petri.

(cas flot continue) ou qui a été consommée (cas du flot discret). Cette place possède un jeton à l'initialisation. Dans le cas du flot de données continu, à la première écriture de la donnée la place « Vide » perd son jeton et la place « Valeur » sera ensuite toujours marquée. En revanche, dans le cas du flot de données discret, le franchissement de la transition « *Lire_donnée* » fait passer le jeton de la place « Valeur » à la place « Vide ».

La modélisation de l'unité de stockage, représentée sur la figure 2.48, est plus simple d'un point de vue réseau de Petri. Ainsi, nous avons une place « Stockage », intégrant un jeton représentant la donnée, et les deux transitions correspondant à l'écriture et à la lecture (*Écrire_donnée* et *Lire_donnée*). En revanche, dans le but d'obtenir un modèle formel précis, une sémantique particulière est associée au jeton décrivant l'état de la donnée selon la lecture ou non de cette donnée. L'enregistrement dans l'unité de stockage comprend deux champs : la donnée elle-même et son état (tableau 2.5). Après chaque écriture, la donnée a un état déclaré nouveau qui peut donc être utilisé par le processus fonctionnel qui lit cette donnée.

Figure 2.48 – Traduction d'une unité de stockage de la méthode SA-RT par un réseau de Petri.

Tableau 2.5 – Caractérisation du contenu de l'enregistrement dans l'unité de stockage du modèle SA-RT.

	Donnée	**État**
Initialement	Non définie	Ancien
Après « ***Écrire_donnée*** »	Définie	Nouveau
Après « ***Lire_donnée*** »	Définie	Ancien

La modélisation d'un processus fonctionnel est relativement complexe dans le sens où il peut posséder de nombreuses entrées et sorties en termes de flots de données. La figure 2.49 décrit un cas général avec des entrées de type flot discret, flot continu et unité de stockage. Ainsi, nous avons en entrée un flot discret (*Donnée_d_1*), deux flots continus (*Donnnée_c_1* et *Donnée_c_2*) et deux unités de stockages (*Donnnée_s_1* et *Donnée_s_2*). En sortie, nous trouvons deux flots discrets (*Donnée_d_1* et *Donnée_d_2*), un flot continu (*Donnée_c_1*) et une unité de stockage (*Donnée_s_3*).

Le processus fonctionnel est décrit par deux places qui représentent les deux états possibles : en exécution ou traitement (place « Exécution ») et à l'arrêt (place « Oisif »).

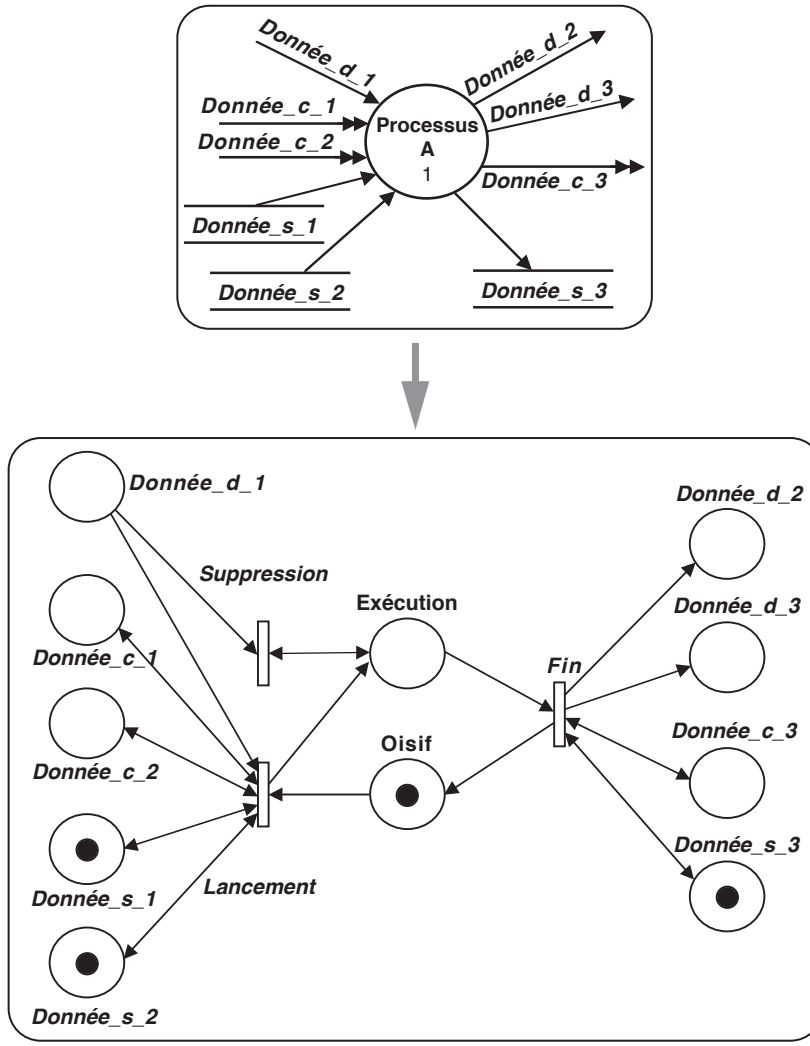

Figure 2.49 – Traduction d'un processus fonctionnel de la méthode SA-RT par un réseau de Petri.

© Dunod – La photocopie non autorisée est un délit.

Trois transitions permettent de modéliser le fonctionnement du processus : une transition « *Lancement* » au démarrage de l'exécution qui fait passer le jeton de la place « Oisif » à la place « Exécution », et une transition « *Fin* » à la fin de l'exécution qui fait passer le jeton de la place « Exécution » à la place « Oisif ».
Une transition supplémentaire est ajoutée : « *Suppression* ». En effet, une hypothèse fondamentale, concernant les flots de données discrets, est mise en exergue sur cet exemple. Le flot de données discret ne peut être consommé qu'une seule fois pendant l'exécution d'un processus fonctionnel ; par conséquent une transition supplémentaire (« *Suppression* ») est nécessaire pour éliminer les flots de données discrets arrivées pendant l'exécution.
La modélisation du processus de contrôle est très complexe et déborde largement le propos de cet ouvrage. Le lecteur intéressé pourra se reporter aux documents de référence cités dans la bibliographie en fin d'ouvrage.

■ Exemple simple

La difficulté de ce genre de modélisation par élément est le recollement des différents modèles lors de l'analyse d'une application complète. Il n'est pas possible d'ajouter ou de juxtaposer les modèles de chaque élément d'une application sans analyser l'ajustement de deux modèles l'un à la suite de l'autre. En effet, dans notre cas, les transitions ou les places situées aux limites des modèles doivent se lier avec le modèle précédent ou suivant.
Nous allons étudier cet aspect sur un exemple simple composé de deux processus fonctionnels (Processus A et Processus B) et d'une unité de stockage, appelée « *Donnée_s* » (figure 2.50). Ces éléments sont liés par des flots de données discrets. D'autre part des flots de données discrets arrivent sur les deux processus fonctionnels (*Donnée_a* et *Donnée_b*) et un flot de données discret est émis (*Donnée_c*).

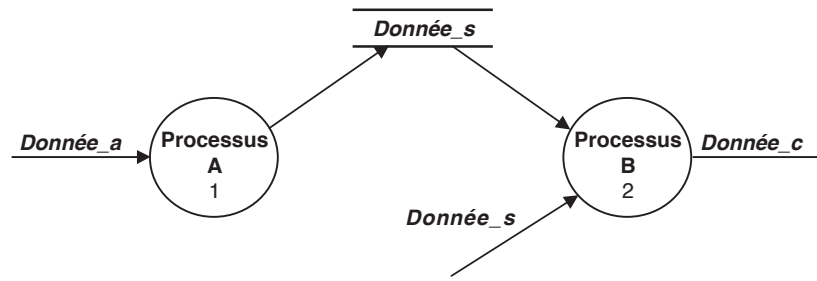

Figure 2.50 – Exemple d'un diagramme flots de données de la méthode SA-RT pour appliquer la transformation en réseaux de Petri.

La traduction de ce diagramme flot de données est représentée sur la figure 2.51. Ce réseau de Petri est composé de 11 places et de 7 transitions. Nous pouvons immédiatement remarquer que nous retrouvons les différentes places correspondant aux modèles initiaux de chaque élément. Ainsi, les deux processus fonctionnels « Processus A » et « Processus B » possèdent chacun les deux places : « A_Exécution » et « A_Oisif » pour le processus A et « B_Exécution » et « B_Oisif » pour le proces-

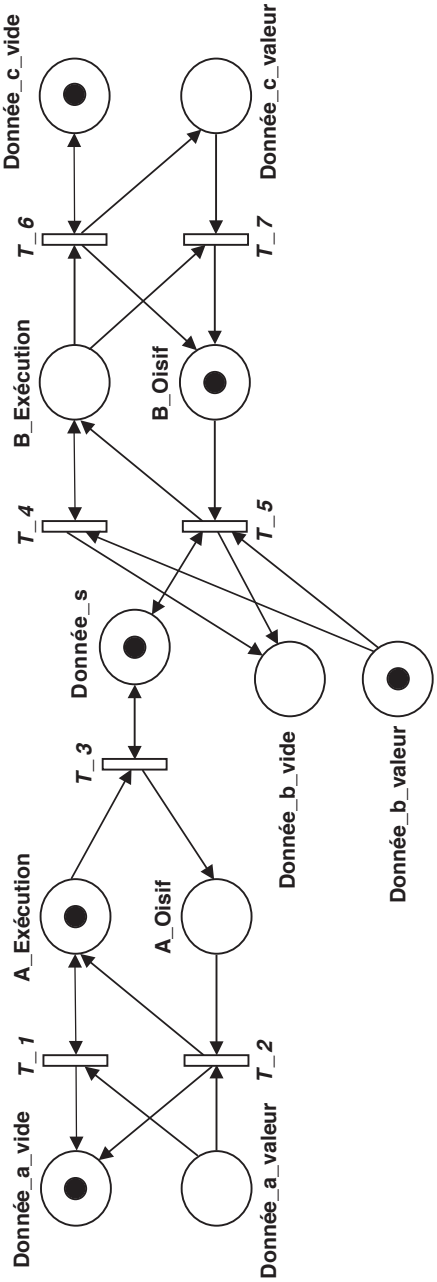

Figure 2.51 – Traduction du diagramme flots de données de la figure 2.50 en réseaux de Petri.

© Dunod – La photocopie non autorisée est un délit.

sus B. L'unité de stockage est modélisée par une seule place « Donnée_s » comme le montre le modèle initial de la figure 2.48. En se basant sur le modèle générique de la figure 2.46, les trois flots de données discrets sont représentés chacun par deux places, soit, par exemple pour le flot de données « *Donnée_a* » du modèle SA-RT : « Donnée_a_vide » et « Donnée_a_valeur ». En résumé nous pouvons noter que, lors de la traduction d'un diagramme flots de données de SA-RT en un réseau de Petri, il y aura toujours un nombre de places égal à la somme des places des modèles génériques des différents éléments du diagramme SA-RT.

En revanche, nous pouvons constater que les transitions sont partagées entre les différents éléments modélisés. Considérons le cas du raccordement du flot de données « *Donnée_a* » au processus fonctionnel A. La transition *T_2* représente à la fois la transition « *Lire_donnée* » du modèle du flot de données discret et la transition « *Exécution* » du modèle d'un processus fonctionnel. De même la transition *T_1* représente à la fois la transition « *Écrire_donnée* » du modèle du flot de données discret et la transition « *Suppression* » du modèle d'un processus fonctionnel. La dernière transition *T_3*, attachée à la modélisation du processus fonctionnel « Processus A », correspond en même temps à la transition « *Fin* » du modèle générique d'un processus fonctionnel et à la transition « *Écrire_donnée* » du modèle de l'unité de stockage « *Donnée_s* ». Nous pouvons répéter cette constatation pour l'ensemble du réseau de Petri.

Le réseau de Petri, qui est l'exact modèle du diagramme flots de données SA-RT, peut être utilisé pour vérifier certaines propriétés de la spécification comme la non occurrence de l'exécution simultanée de deux processus fonctionnels, le cheminement des données, etc. Une fonction de simulation peut être réalisée à l'aide du modèle formel réseau de Petri sous-jacent aux diagrammes SA-RT. Dans ce cadre, le réseau de Petri est analysé en traçant le graphe de marquages dans le cas de transition immédiate (modèle [0,0] de Merlin et Farber), c'est-à-dire l'évolution de la position des jetons dans le réseau. Si le modèle des transitions temporisées est pris compte l'analyse est réalisée à l'aide d'un graphe des classes. En parallèle avec cette évolution des jetons correspondant à l'activation ou non de l'action associée à la place, il est alors possible de visualiser l'activité au niveau du diagramme flots de données SA-RT. Prenons l'exemple précédent, le marquage présenté sur la figure 2.51 correspond à un moment de l'exécution où nous avons les éléments suivants :

– pas de donnée sur le flot de données discret « ***Donnée_a*** » entrant ;
– le processus fonctionnel « **Processus A** » en exécution ;
– une donnée présente dans l'unité de stockage « ***Donnée_s*** » ;
– une donnée présente sur le flot de données discret « ***Donnée_b*** » entrant ;
– le processus fonctionnel « **Processus B** » en arrêt ;
– pas de donnée sur le flot de données discret « ***Donnée_c*** » sortant.

Par conséquent, il est possible de visualiser ces différents états sur le diagramme flots de données initial, représenté sur la figure 2.50. Cette visualisation peut être animée et donc ainsi permettre au concepteur de voir le déroulement de l'exécution de diagramme flots de données SA-RT. Cette visualisation est schématisée sur la

figure 2.52 qui reprend la figure 2.50 avec une marque symbolisant l'activité d'un processus fonctionnel ou la présence d'une donnée.

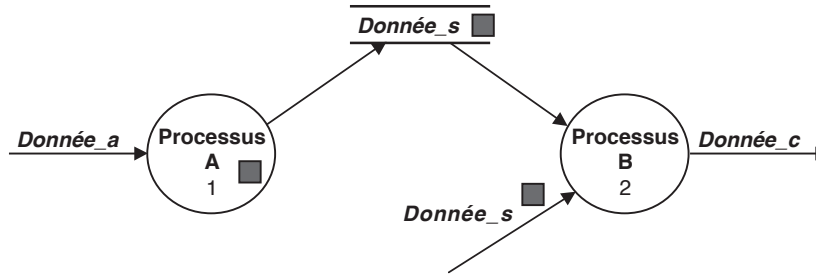

Figure 2.52 – Visualisation de l'exécution d'un diagramme flots de données en utilisant le réseau de Petri.