R-PORTS : une plate-forme d'aide à la conception d'applications réparties

Lionel Blanc — Marie-Agnès Péraldi-Frati — Charles André

Laboratoire I3S
41 Boulevard Napoléon III
06041 Nice Cedex
{lblanc,map,andre}@i3s.unice.fr

RÉSUMÉ. *Dans cet article nous présentons la plate-forme R-PORTS qui automatise la répartition du contrôle pour des applications temps-réel. L'approche est basée sur des modèles synchrones. Elle garantit un comportement prévisible de l'application du point de vue logique et temporel. R-PORTS supporte tout protocole associé aux réseaux de terrain par encapsulation des services de communication dans un middleware. Le middleware disponible dans R-PORTS est dédié à CAN/VAN.*

ABSTRACT. *This paper describes the R-PORTS framework that provides automated distribution of control in real-time systems. Our approach is based on a synchronous model. It ensures the logical and temporal deterministic behaviour of the application. R-PORTS may accept any fieldbus protocol since it encapsulates its communication services inside a middleware. The present version of R-PORTS supports CAN/VAN fieldbus.*

MOTS-CLÉS : *réseaux de terrain, temps-réel, middleware, synchrone, validation logique et temporelle.*

KEY WORDS: *fieldbus, real-time, middleware, synchronous, temporal and logical validation*

1. Introduction

Les applications auxquelles nous nous intéressons lors de cette étude sont les systèmes automatisés de production (SAP) à contrôle prédominant. De telles applications, essentiellement réactives, intègrent des contraintes temporelles qui doivent être impérativement respectées sous peine, de dégradations de matériels, de perte de production ou de mise en danger des personnes. Plus généralement, ces systèmes incluent de fortes exigences de sûreté de fonctionnement : il s'agit de systèmes sur lesquels on peut «compter » à tout moment.

L'introduction des réseaux de terrain dans la mise en œuvre des SAP a conduit à des améliorations mais leur utilisation n'exploite pas encore pleinement leurs potentialités. Leur emploi est encore trop souvent restreint à la transmission d'informations d'entrées/sorties déportées à destination d'un contrôleur centralisé. L'architecture résultante, appelée mono-maître, est vulnérable : la disponibilité de l'installation repose sur un unique contrôleur.

Une approche plus rationnelle permettrait d'obtenir des systèmes plus sûrs. Nous en proposons une basée sur la répartition du contrôle. Il existe une partition matérielle de fait des constituants (les capteurs/actionneurs) sur laquelle on peut s'appuyer. Répartir le contrôle signifie descendre l'intelligence vers ces constituants. L'information est ainsi traitée au plus près des besoins, ce qui accroît sensiblement l'immunité aux bruits de l'installation notamment électromagnétiques (e.g., une automobile équipée d'un bus CAN transitant sous une ligne à haute tension). La sûreté de fonctionnement est améliorée puisqu'il subsiste moins de points sensibles, c'est-à-dire de sites influant fortement sur les paramètres que sont la *disponibilité*, la *maintenabilité*, la *fiabilité* et la *sécurité* du SAP. Nous obtenons alors un ensemble symétrique d'éléments communicants qui caractérise une architecture multi-maîtres. Comme le soulignent Thomesse et Mammeri [THO 93], la répartition du contrôle n'est pas usuelle : *« Qu'est-ce qui est réparti... ? Les algorithmes, avec un contrôle et des données centralisés, c'est ce qui est le plus facilement réalisable ; le contrôle, c'est déjà plus rare ».*

L'approche proposée

À partir d'une description fonctionnelle d'une application, de ses contraintes et de l'architecture cible (AC), nous obtenons une *architecture opérationnelle* (AO) *validée,* résultat de la projection des entités fonctionnelles sur l'AC. Cette approche repose sur le paradigme synchrone. Une application est décrite par un ensemble de sites. Le comportement de chaque site est modélisé par le langage synchrone ESTEREL. Ces comportements sont qualifiés de « synchrones forts » puisque les causes et les effets d'une réaction se font dans un même instant. Les communications entre sites sont générées automatiquement par l'outil. L'interaction entre sites est, par contre, fondée sur une sémantique synchrone faible. En effet, ces communications distantes « durent » au sens ESTEREL, la sémantique synchrone forte ne s'y projette donc pas intuitivement. Le modèle synchrone faible [BON 95] est un concept plus naturel qui stipule que l'effet d'une réaction sur un site sera sensible sur un site distant à l'instant suivant. Un ensemble de services logiciels interfacés dans une couche appelée *middleware* offre ce caractère synchrone au réseau.

Pour mener à bien cette répartition nous avons développé la plate-forme R-PORTS (Rapid Prototyping Of Real Time Systems). Cette plate-forme intègre un langage graphique de description d'architectures fonctionnelles (AF) et une couche *middleware* dédiée aux réseaux de terrain CAN/VAN. R-PORTS rend transparent au concepteur le processus de répartition et livre une AO validée ayant les propriétés de déterminisme et de réactivité.

Dans cet article nous présentons les concepts d'architectures [BAY 95] étendus à notre approche. Dans un premier temps nous introduisons le langage de description d'architectures permettant de représenter une AF ainsi que les contraintes d'une application. Dans un deuxième temps nous présentons le middleware dans le cadre de l'AC. Enfin nous montrons comment obtenir une AO et valider les contraintes exprimées.

2. Les modèles d'architectures de R-PORTS

Trois modèles d'architecture interviennent lors de la description d'un SAP avec la plate-forme R-PORTS : l'AF, l'AC et l'AO. Ces modèles et leurs interactions sont représentés sur la Figure 1.

2.1 *L'architecture fonctionnelle*

Une AF décrit la structure et le comportement externe des activités du SAP. Elle est le résultat de l'étape de spécification. Cette architecture constitue le modèle d'entrée de l'approche et est fournie par le concepteur au travers d'un langage de description d'architectures hiérarchique et graphique. L'atome de base du langage est appelé *composant terminal*. Il est par définition insécable. Un composant est vu au travers de ses interfaces d'entrées/sorties et communique par signaux via ses points d'accès. La sémantique de communication par signaux est sensiblement différente d'une communication par services. Dans une sémantique orientée services, la non disponibilité d'un service (i.e, conséquence d'une faute) est traitée par une exception alors qu'elle se traduit par l'absence d'un signal dans notre cas. La réutilisabilité d'un composant en est facilitée. La modélisation du comportement des composants terminaux est réalisée avec les langages synchrones textuels (ESTEREL [BER 98]) ou graphiques (SYNCCHARTS [AND 96]). Ces langages ont été choisis pour plusieurs raisons :

– ils intègrent des structures adaptées à la modélisation des applications de contrôle/commande où le contrôle est prédominant ;

– ils s'appuient sur une sémantique mathématique bien définie ;
– les compilateurs de langages synchrones génèrent un code déterministe sous forme de systèmes d'équations booléennes assurant des temps de transitions courts et prévisibles ;
– l'environnement des langages synchrones offre des outils de validation logique de ces systèmes.

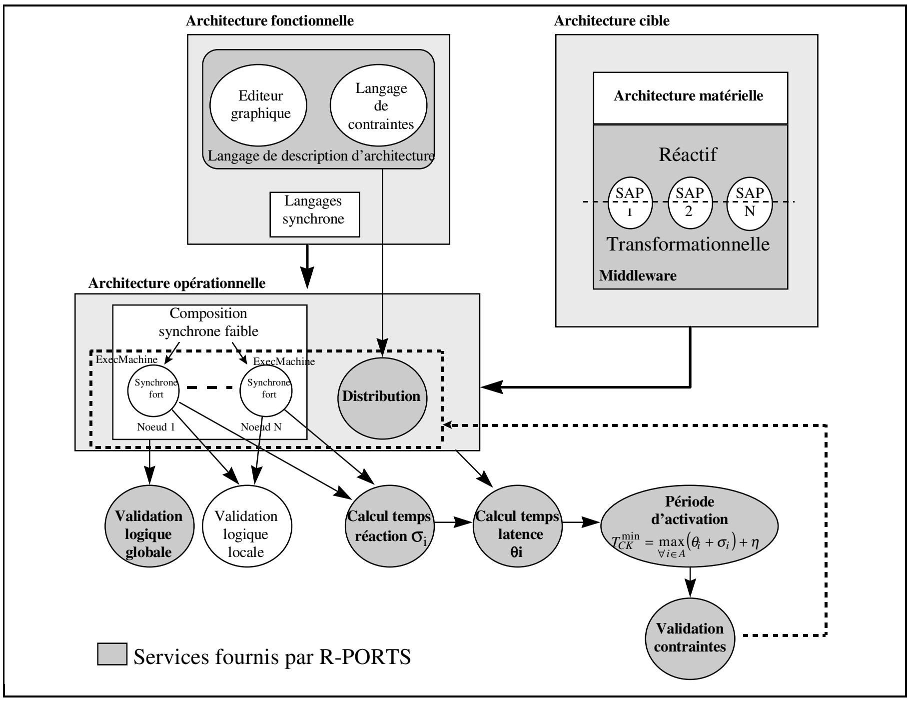

Figure 1. *Modèles intervenant dans le processus de répartition*

La hiérarchie du langage de description s'exprime au travers de *macro-composants* résultat de l'encapsulation de composants (composants terminaux et/ou macro-composants). Il est possible de spécifier un ensemble de contraintes fonctionnelles sur les composants. Les composants terminaux internes au macro-composant héritent des contraintes fonctionnelles de celui-ci. Ces contraintes peuvent être de type géographique ou temporel. Les contraintes géographiques sont associées aux composants terminaux et procurent une indication sur la répartition imposée par des critères de localisation (système à commander), de dimensionnement (système de commande)... Les contraintes temporelles se divisent en deux catégories : les périodes d'échantillonnages et les fenêtres temporelles. Les premières sont classiques pour les applications de contrôle fréquemment cycliques. Nous spécifions les secondes à l'aide d'un langage utilisant les opérateurs de la logique temporelle d'Allen associés à des fenêtres d'activités (FA). Ce langage permet d'exprimer des contraintes de type *date au plus tôt*, *date au plus tard* et *intervalles temporels*. Une fenêtre temporelle *W* peut se définir de deux manières différentes :

1. ses instants de début et de fin respectivement notés $\uparrow W$ et $\downarrow W$
2. son instant de début et une durée Δ avec $\downarrow W = \uparrow W + \Delta$

Une FA correspond à une fenêtre temporelle (définie selon la notation 1) à laquelle on associe un événement w tel que $\uparrow W \leq w \leq \downarrow W$. Remarquons que nous notons par le même identificateur un événement et sa date d'occurrence. Nous avons identifié deux FAs nécessaires pour la spécification d'intervalles temporels qui nous permettent de définir des *contraintes temporelles relatives* :
– *fenêtre de production* d'un signal, laquelle correspond à l'intervalle durant lequel est émis le signal V: *emit(V)* ;
– *fenêtre de consommation* d'un signal qui caractérise l'intervalle durant lequel un signal est visible pour sa consommation : *receive(V)*.

La deuxième forme d'intervalle est très commode pour exprimer des « attentes ». Ce sont des fenêtres temporelles dans lesquelles l'instant de début est l'événement associé à une *FA* et *Duration* sa durée Δ . Cette fenêtre s'apparente à un monostable réarmable par l'événement associé à *FA* de temporisation *Duration*. On la note $timeout(FA, Duration)$. Ainsi, si l'on désire spécifier une contrainte temporelle de type date au plus tôt, telle qu'un événement *Top* soit disponible pour être consommé au minimum 50 ms après son émission, on écrira :

$$\text{timeout}(\text{emit}(\text{Top}),\text{50 ms})\ \textbf{\textit{before}}\ \ \text{receive}(\text{Top})$$

2.2 *L'architecture cible*

Nous appelons AC (cf. Figure 1) l'architecture matérielle de l'application (le réseau physique, les processeurs, les mémoires, les capteurs actionneurs...) et une couche logicielle appelée middleware. Le middleware interface l'application avec le matériel et rend transparent la distribution au concepteur de l'application. Il offre des services de haut niveau tels que la cohérence temporelle, la tolérance aux pannes, la synchronisation d'horloge. Il participe à l'intelligence des nœuds du réseau. Ainsi, un service d'historique autorise un capteur à archiver ses mesures à des fins d'interpolation et obtenir son statut de capteur intelligent. Le middleware garantit également la pérennité du SAP en cas de migration vers un autre profil de communication (portabilité). Il suffit alors que cette couche d'interfaçage soit disponible pour le nouveau protocole.

Nous avons défini un middleware pour les réseaux CAN/VAN sous la forme d'un macro-composant, chaque service étant modélisé par un composant terminal. Un des intérêts majeurs de cette décomposition est d'obtenir un comportement déterministe pour l'ensemble des services. Dans la plate-forme R-PORTS les services du middleware sont appelés lors du processus de génération de l'AO, le concepteur n'ayant pas directement accès à ces services. Le service « Signal R/W » (primitive *SgWriteRequest(Sg,Time)*) permet par exemple de lire ou écrire un signal avec une qualité de service temporelle. Cette qualité de service est adaptative (elle tient compte de l'application) au travers du paramètre *Duration*. Ainsi la caractéristique temporelle du service *SgWriteRequest(Sg,Duration)* est :

$$\mathrm{emit}(\mathrm{SgWriteResponse})\ \textbf{\textit{during}}\ \ \mathrm{timeout}(\mathrm{receive}(\mathrm{SgWriteRequest}),\mathrm{Duration})$$

2.3 L'architecture opérationnelle

À partir de l'AF et l'AC, nous générons l'AO, résultat de la projection de l'AF sur l'AC. L'objectif de l'outil R-PORTS est d'automatiser cette étape. La première phase concerne la distribution des composants terminaux sur l'ensemble des sites de l'application répartie. Cette phase intègre les contraintes spécifiées dans l'AF telles que les contraintes géographiques ainsi que les dépendances logiques inter nœuds déduites de l'AF. Cette analyse s'appuie sur la différence booléenne et a pour objectif de limiter l'impact de fautes sur le comportement du SAP.

Le *comportement local* à un nœud est issu de la *composition synchrone forte* des comportements des composants projetés sur le dit site. De par les modèles sémantiques sous-jacents aux langages synchrones, la composition des composants livre un composant réactif et déterministe dont la traduction est conforme au spécification. Le *comportement global* d'un SAP est le résultat de la *composition synchrone faible* des comportements locaux (sites) [BON 95]. Cette composition, mise en œuvre par le middleware, produit un système constructif (stable) au sens de la sémantique circuit d'ESTEREL [BER 98]. Ceci signifie que tout réseau de nœuds engendre un comportement prévisible si le comportement local à chaque nœud l'est. Sous certaines conditions, on peut montrer que le comportement du système réparti se conforme au comportement centralisé à un instant de retard près. L'objectif étant de valider le comportement de l'application répartie, nous fournissons, dans notre plate-forme, un modèle d'exécution qui garantit cette sémantique. On donne sur la Figure 2 le modèle d'exécution local à un nœud. Ce modèle introduit un asynchronisme borné par un retard unité de communication, ainsi un signal produit à l'instant d'activation k est disponible pour l'ensemble de ses consommateurs à l'instant d'activation k+1.

Dès le modèle d'exécution, il est nécessaire de quantifier la notion d'instant, en particulier de définir une *période d'activation minimale* qui permette de garantir les sémantiques synchrones. Le calcul théorique de cette période est réalisé dans le « pire cas », cas en outre très peu probable.

Nous avons formulé un certain nombre de contraintes pour le calcul de cette période (qui permettent de borner le calcul) :

1. L'automate le plus proche de son échéance de transition doit être prioritaire sur l'accès au bus. Seule l'occupation du bus au moment de la tentative d'accès peut induire un retard ;
2. Pour tenir compte de l'asynchronisme de production des signaux, chaque signal est émis individuellement sur le bus (chaque signal doit donc y gagner l'accès).

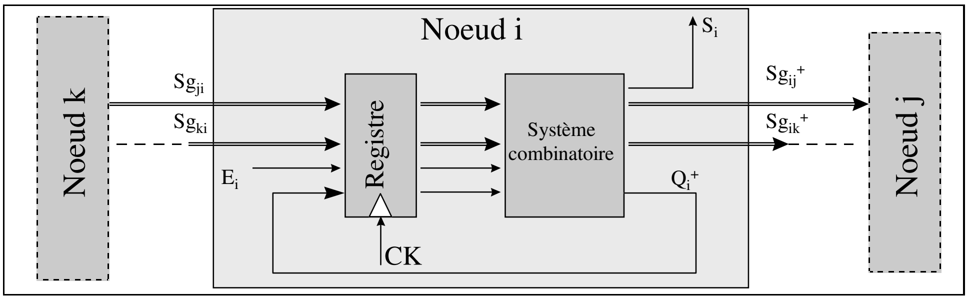

Figure 2. *Modèle d'exécution d'un nœud du réseau*

Cette période d'activation minimale que l'on note $T_{CK}^{\min}$, définit la granularité de l'instant (au sens ESTEREL). Elle dépend de deux paramètres associés à chaque site :
– σ_i le temps maximum de transition de l'automate i ;
– Θ_i le temps maximum de latence de l'automate i (incluant le temps de transmission)[1].
Soit A un ensemble d'automates $A = \{A_1, A_2, \ldots\}$, $T(a)$ l'ensemble des transitions de a, $T_{a,t}$ la durée de l'exécution d'une transition $t \in T(a)$ pour l'automate a et $S(a)$ l'ensemble des signaux de l'automate a, on définit :

$$\sigma_i = \max_{\forall t \in T(i)} \left(T_{i,t}\right) \text{ et } \Theta_i = \max_{\forall s \in S(i)} \left(\theta_i(s)\right)$$

A ce stade, nous présentons succinctement l'algorithme qui permet d'obtenir $T_{CK}^{\min}$.
Il utilise des résultats de Tindell et Burns [TIN 94] relatifs à CAN, des résultats classiques d'ordonnancements à priorités et des spécificités de R-PORTS.

— $\forall\ i \in A$, calculer σ_i
/ le temps de transition d'un automate est une contrainte de l'application qui constitue un minorant du grain de l'instant, en effet un automate doit avoir transité avant d'être réactivé (sinon le code synchrone ne serait pas exécuté de façon atomique en violation avec les hypothèses synchrones fortes) */*

1 Dans cet article, pour simplifier, nous considérons un médium fiable sans erreur de transmission. Dans l'outil cette restriction est levée.

Attribuer à chaque automate une priorité telle que $\forall j,k \in A : \sigma_k < \sigma_j \Rightarrow P(j) < P(k)$
$\forall\ s \in S(i)$, attribuer une priorité $P(s)$ héritée de la priorité de l'automate i
Des $P(s)$ construire un ordre total compatible sur l'ensemble des identificateurs

/ Conséquence de la 1ère contrainte (p. 7) où les automates les plus proches de leur échéance sont prioritaires sur l'accès au bus. La dernière phase de cette étape est une conséquence de l'émission dans des trames distinctes des signaux d'un automate (seconde contrainte, p. 7). La première contrainte permet de déduire un ordre total sur les priorités d'automates mais partiel pour l'ensemble des identificateurs. */*

— Pour $i \in A$ pris dans l'ordre de priorité décroissante

$\forall\ s \in S(i)$, calculer $\theta_i(s) = I_s + C_s$

avec I_s le temps d'interférence et C_s le temps de transmission de s

/ Le temps de latence correspond au temps maximal pour qu'un signal soit réellement reçu par ses consommateurs. Il inclut évidemment le temps de transmission matérialisé par C_s. Le temps d'interférence est l'écart entre la requête d'émission et l'émission effective, il caractérise le temps maximal pendant lequel l'émission d'un signal peut être différée. C'est une caractéristique d'un médium à accès par compétition. L'avantage du protocole CSMA/CA du bus CAN (ou VAN) est qu'il permet de borner ce temps (contrairement à CSMA/CD) */*

Pour CAN [TIN 94] :

$$C_s = \left(\frac{\lfloor 34 + 8d_s \rfloor}{4} + 47 + 8d_s \right) \tau_{bit}$$ avec d_s la taille du signal s en octets et τ_{bit} le débit du réseau

/ Cette équation est la conséquence du format de codage d'une trame */*

$$I_s = B + \sum_{\forall j \in (hp(s) \cap Ord)} \left(C_j + \tau_{bit} \right)$$ avec $hp(s)$ l'ensemble des signaux plus prioritaires que s et Ord l'ensemble des signaux restant à ordonnancer à l'instant σ_i.

/ L'émission d'une trame peut-être différée si le bus est occupé à l'instant de la tentative d'émission (représenté par B) ou si un signal plus prioritaire tente d'y accéder simultanément. Dans le pire des cas, l'ensemble des signaux plus prioritaires que s (hp(s)) différeront l'émission du signal (2nd terme de l'équation). L'espace inter-trames est caractérisé par le terme τ_{bit} */*

$$B = \max_{\forall k \in (lp(s) \cap Ord)} \left(C_k \right) - \tau_{bit}$$ avec $lp(s)$ l'ensemble des signaux moins prioritaires que s.

/ Le terme B caractérise une occupation du bus antérieure à la tentative d'accès. Dans le pire des cas, le bus est occupé par le signal dont la durée de transmission est la plus importante et qui vient de débuter à la tranche de temps précédente */*

— $T_{CK}^{\min} = \max\limits_{\forall i \in A}\left(\sigma_i + \Theta_i\right) + \eta$ avec η la déviation maximale des horloges synchronisées.

La Figure 3 donne un exemple du modèle d'exécution présenté.

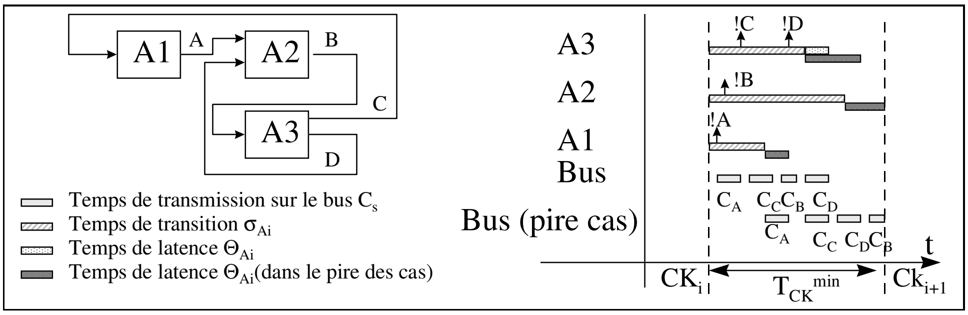

Figure 3. *Un exemple de communication inter automates sur le bus*

Le calcul de $T_{CK}^{\min}$ permet de valider les contraintes temporelles exprimées dans l'AF. Sur l'exemple de la Figure 3 les priorités d'automates sont dans l'ordre décroissant A1, A3, A2. La priorité induite sur les signaux est A, C, D, B (ordre décroissant). On exprime de plus les contraintes suivantes :

— date au plus tard pour la réception de A :

$$\text{receive}(\text{A}) \ \textbf{\textit{during}} \ \text{timeout}(\text{emit}(\text{A}),70 \text{ ms})$$

— période d'échantillonnage de B : $T_B = 30\, ms$

La validation de ces contraintes passe par la vérification du système d'équations suivant :

— $\text{timeout}\left(\uparrow \text{CK}, \text{T}_{\text{CK}}^{\min}\right)$ ***during*** $\text{timeout}(\text{emit(A)},70 \text{ ms})$ (cf. Figure 4a.)

— $\text{timeout}\left(\uparrow \text{CK}, \text{T}_{\text{CK}}^{\min}\right)$ ***during*** $\text{timeout}\left(\uparrow \text{CK}, \text{T}_\text{B}\right)$ (cf. Figure 4b.)

Figure 4. *Représentation graphique du système d'équations*

Dans le cas où les contraintes ne sont pas respectées, il n'y a pas de solution à ce système. Il est alors nécessaire soit de modifier la distribution des composants, soit d'optimiser le code synchrone. Actuellement, l'obtention et la vérification de ces équations sont manuelles. Des résultats numériques sont présentés dans Blanc et al. [BLA 98] au travers d'une application multimédia.

3. Conclusion

Dans cet article nous avons proposé une méthode visant à améliorer l'utilisation des réseaux de terrain dans les SAP. Cette méthode est basée sur la répartition du contrôle au niveau des nœuds de l'application et s'appuie sur le modèle synchrone. La sémantique synchrone forte exprime le comportement intra-nœuds et la sémantique synchrone faible le comportement inter nœuds. La plate-forme de prototypage rapide R-PORTS implémente cette méthode et permet à partir de la description de l'AF et de ses contraintes d'obtenir une AO validée. La tolérance aux fautes n'a pas été abordée dans cet article mais la plate-forme intègre des politiques visant à l'améliorer.

Les travaux à venir porteront sur la vérification automatique des contraintes temporelles afin de valider ou d'invalider une AO. En cas de rejet nous souhaitons pouvoir établir un diagnostic qui permette de modifier l'AF et/ou la répartition.

4. Bibliographie

[AND 96] ANDRE C. « Representation and analysis of reactive behaviour : A synchronous approach », *In Proc. CESA'96*, 1996, Lille, France.

[BAY 95] BAYART M., SIMONOT-LION F., « Impact de l'émergence des réseaux de terrain et de l'instrumentation intelligente dans la conception des architectures des systèmes d'automatisation de processus », Convention MESR, 1995

[BER 98] BERRY G., « The Foundations of Esterel », *Proof, Language and Interaction : Essays in Honour of Robin Milner*, MIT Press, 1998

[BLA 98] BLANC L., ANDRE C., PERALDI-FRATI M.A., « La plate-forme R-PORTS : Application à un magnétophone maître/esclave », Rapport interne, I3S, 1998

[BON 95] BONIOL F., « Processus réactifs communicants synchrones », *Journées Formalisation des Activités Concurrentes*, LAAS-CNRS, 1995

[THO 93] THOMESSE J.P., MAMMERI Z., *réseaux locaux industriels - FIP et MAP dans les systèmes automatisés*, Edition Eyrolles, 1993

[TIN 94] TINDELL K., BURNS A., « Guaranteed message latencies for distributed safety critical hard real time control networks », Technical Report YCS229, Department of Computer Science, Univ. Of York (UK), 1994