Les systèmes temps-réel et l'industrie

Christian Garnier
Responsable du Laboratoire Temps Réel à l'ISEP et Consultant senior pour la Société ARION

Du métro sans conducteur, au pilote automatique dans les avions, les systèmes temps réel font aujourd'hui partie de notre vie quotidienne. Ce monde du temps réel est si vaste que ses technologies s'appliquent à divers secteurs en commençant par le contrôle industriel, le transport, les télécommunications, les systèmes de sécurité, la domotique... Fort de 30 années d'expérience dans ce domaine, l'auteur nous raconte brièvement l'histoire de cette discipline, présente les différentes fonctions ainsi que les composantes des systèmes temps réel. Il aborde enfin les contraintes et dresse un état de l'art des applications.

1. Introduction

Nous sommes environnées par ce concept passé dans le langage courant, parfois galvaudé (on peut penser à cette expression insolite : « Real Time Marketing »), toujours un peu obscur.

Cette opacité de sens est d'autant plus surprenante que nous utilisons de plus en plus de moyens qui sont pilotés par des systèmes temps réel : depuis l'ABS de votre voiture, en passant par la robotique, la gestion des portes d'un métro ou d'un train, jusqu'à la phase d'atterrissage des avions. Même les chaînes de fabrication agroalimentaire sont gérées en temps réel : à titre d'exemple, la béchamel dans les crêpes salées est dosée au gramme près pour respecter les enjeux financiers.

2. Historique

Dans ce chapitre, nous aborderons tout d'abord un rapide historique permettant d'appréhender les étapes clés qui ont permis de rendre matures les différentes technologies utilisées. Puis nous verrons les principales innovations et technologies qui se sont imposées depuis les années 70. Et enfin, nous relaterons quelques expériences vécues.

2.1. Les différentes étapes

Avec le syllogisme, Aristote a jeté les fondements de la logique numérique. Ces principes ont été successivement repris à partir des années 1800 par George Boole, Augustus de Morgan, Charles Saniseks Pierce, John Venn, Leonard Euler et enfin Shannon pour arriver aux bases du codage binaire, c'est-à-dire aux bases mêmes de l'informatique moderne : le fameux « bit ».

Dès 1638, en inventant le « premier ordinateur », Blaise Pascal ouvre la voie. Hollerith suit en 1890, avec sa machine mécanographique à cartes perforées, tandis que John Von Neumann invente la base moderne des calculateurs en 1947.

Au niveau technologique, les événements marquants restent l'avènement de la mécanique quantique, par Max Planck en 1900, puis l'arrivée des tubes cathodiques, avec l'invention de la triode en 1936 par Lee de Forest, qui révolutionne ainsi l'électronique. En 1947, c'est l'avènement du transistor, qui autorise la naissance des premiers calculateurs électroniques dès 1950. En 1973, le premier mini ordinateur conçu à base de technologie « LSI » (Large Scale Integrated) est né.

Côté logiciel, l'année 1950 voit la naissance du langage assembleur, complété dès 1954 par le Fortran (premier langage de haut niveau). Les nouveautés se succèdent ensuite : Unix naît en 1969, le langage « C », en 1973. Dix ans après, le « C++ » innove encore. Dès 1985 « Windows 1.0 » apparaît sur le marché, puis c'est la naissance de Java et de tous les logiciels libres, comme Linux et ses variantes.

Avant que la modernité n'emballe cette chronologie, je voudrais signaler deux faits notables, qui ont littéralement révolutionné notre vie quotidienne : en 1974, Moréno invente la carte à puce, et en 1981, c'est l'arrivée du premier « Personal Computer » que l'on retrouve aujourd'hui dans l'industrie.

2.2. Les technologies temps réel de 70 à nos jours

Dans les années 70-80, l'architecture d'un système temps réel est essentiellement composée d'un calculateur, dans lequel on trouve tout, depuis la carte processeur, jusqu'au logiciel et les entrées-sorties. C'est dans ces années que Motorola et Intel se battent pour obtenir la suprématie sur le monde industriel : si Motorola semble avoir gagné avec le « MC68000 » et le bus « VME » (Versa Module Europe), Intel se focalise sur le monde du PC avec le « 8088 », qui est à l'origine de la saga des processeurs Intel pour aboutir au fameux Pentium.

Les années 80-90 voient l'avènement des réseaux locaux industriels pour assurer la communication entre les différents calculateurs constitutifs du système (déport des entrées-sorties pour limiter les coûts de réalisation des systèmes). Leur mise en œuvre engendre un gain de l'ordre de 30% sur la réalisation des applications : c'est la raison de leur utilisation croissante dans les systèmes temps réel avec entre autres l'invention du réseau « CAN » (Control Area Network) pour l'automobile. C'est aussi le début des composants programmables permettant l'intégration dans un seul composant de plusieurs équations logiques complexes, mises sous la forme d'une somme de produits : ils vont considérablement simplifier la réalisation des cartes électroniques.

La décennie suivante permet la généralisation d'une innovation déjà utilisée dans l'avionique militaire : il s'agit de l'intégration de systèmes multiprocesseur multitâche, soit dans un seul et unique équipement, soit distribué dans un ensemble d'équipements interconnectés avec des réseaux de communication.

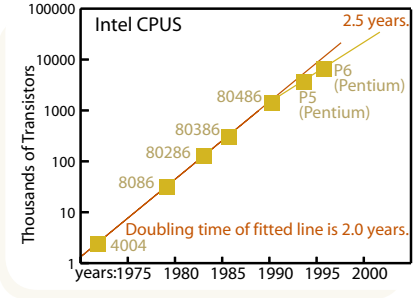
Les années 1990-2000 sont aussi celles de la démocratisation de la conception des systèmes, avec l'arrivée, dans les systèmes temps réel, de langages informatiques de haut niveau connus de tous comme le « C » ou le « C++ ». Il faut se rappeler qu'avant cette mutation, le monde du temps réel était réservé à des spécialistes travaillant avec des langages spécialisés tel que le « LIPS » ou l'« ADA » voire directement en assembleur pour satisfaire aux performances du système. Cette démocratisation des langages a permis au domaine du temps réel d'échapper à l'exclusivité de ces spécialistes. Mais à contrario, cette vulgarisation ne marquerait-elle pas le début d'une certaine confusion ?

Ces années marquent aussi le début de l'intégration de système dans un seul composant, grâce à la grande maturation des composants programmables avec l'arrivée des FPGA (Field Programmable Gate Array) à haute densité d'intégration. On ne parle pas encore de « SOC » (System On Chip) ni de « SOPC » (System On Programmable Chip), mais pourtant c'est bien la fin du siècle qui les a vus naître.

Les années 2000 ont ensuite vu l'explosion des systèmes temps réel, avec une différence notable : ce ne sont plus les industriels qui tirent le marché, mais c'est le marché du « consumer » qui fournit des solutions, avec, entre autres, l'avènement du tout TCP-IP et la démocratisation du PC. C'est aussi à cette époque que les industriels mettent en avant une démarche commencée fin des années 90 : la notion de « COTS » (Component On The Shelf), qui permet de réutiliser des fonctions, voire même des systèmes préexistants, afin de réduire les coûts de réalisation des applications. Parallèlement, les premiers calculateurs temps réel embarqués, réalisés à base de composants programmables voient le jour : c'est le démarrage des solutions « SOPC ».

2.3. Un retour d'expérience

Quand j'ai commencé à travailler dans le domaine du temps réel, un professionnel d'expérience, « vénéré » pour sa grande sagesse m'a expliqué plusieurs choses :
La première concerne les évolutions de l'informatique : pour lui, elles sont liées à la technologie, qui ne cesse de progresser. Sa grande marotte était de considérer les évolutions technologiques d'une année sur l'autre, et de se dire : « depuis que je travaille (et cela faisait un moment !), l'évolution des performances et des capacités croit globalement d'une puissance de 2 chaque année, et ce n'est pas près de finir ».



La loi de Moore

Sa réflexion prit également la tournure inattendue d'une charade : « quand je fais du temps réel, je prends mon temps quand je suis pressé pour me dépêcher quand j'ai le temps ». Le mot « pressé » sous-entend « phase temps réel » tandis que « j'ai le temps » correspond à la « phase d'initialisation ». Dès lors, on peut réécrire cette charade de la façon suivante : « pendant la phase temps réel, je me contente d'exécuter ce qui a été préparé pendant la

phase d'initialisation », car le temps réel est avant tout un système dit « prédéfini fermé ».

Son troisième « enseignement », qui fera frémir pas mal d'entre vous, pouvait se résumer à « Si tu ne veux pas être embêté, réalise les parties pointues de l'application à partir de solutions matérielles ».

Enfin, les systèmes temps réel doivent être perçus comme des aides, et non des contraintes, avec des modes de fonctionnement faciles à maîtriser par les utilisateurs (les faits divers de ces derniers mois concernant le régulateur de vitesse pour automobile sont de parfaits contre-exemples !).

3. Définitions et principes

Il existe une multitude de définitions du temps réel. À titre d'exemples, Pouget propose comme définition : « Le temps réel est le pilotage à tout instant d'un système évolutif dans le temps ». De son coté, Elloy se veut plus précis : « Peut être qualifiée de temps réel toute application dont le déroulement est assujetti à l'évolution dynamique d'un procédé extérieur qui lui est connecté et dont il doit contrôler le comportement ».

Avec un peu plus de précision, nous pouvons dire qu'un système temps réel est un système qui contrôle (ou pilote) un procédé physique à une vitesse adaptée à l'évolution du procédé contrôlé. Il génère des données en sortie à partir des données d'entrée et contrôle le comportement du procédé auquel il est associé, et cela quelles que soient les conditions d'utilisation.

Pour parfaire cette définition, il convient d'ajouter les concepts suivants, qui se déclinent à tous les niveaux de l'application :

Le concept du « JUSTE ET À TEMPS »

L'ensemble des traitements à réaliser pour générer les sorties est fait correctement dans le temps imparti, à partir de données d'entrée valides. Cela veut dire qu'il faut d'abord respecter les contraintes temporelles de l'application, puis travailler avec des données cohérentes spatialement (toute image de la donnée doit être la même dans tout le système) et temporellement (toute évolution d'une donnée doit se propager dans le système dans un temps borné connu à l'avance, de préférence le plus faible possible), et enfin respecter la durée de vie de la donnée.

Le Concept du « MÊME CAUSE MÊME EFFET »

L'axiome précédent se répète toujours, quelle que soit l'occurrence des événements dans le système.

La difficulté de réalisation des systèmes temps réel a amené certains à essayer de les différentier des autres systèmes informatiques par la prise en compte de contraintes temps réel dont le respect est aussi important que l'exactitude du résultat. Dès lors les puristes essayent de dissocier le vrai du faux en utilisant un qualificatif :

Hard Real Time

Dans ce cas, les contraintes temporelles doivent être respectées car tout dépassement peut engendrer des situations critiques, voire catastrophiques (par exemple, l'ouverture inopinée des portes d'un métro le soir à 18 heures...).

Soft Real Time

Dans ce cas, le système « s'accommode » de certains dépassements des contraintes temporelles. Néanmoins, il faut quand même respecter certaines limites au-delà desquelles le système devient inutilisable : visioconférence, jeux en réseau, ...

Parmi la multitude d'exemples qui pourraient être trouvés, il convient de distinguer ceux qui utilisent réellement des systèmes temps réel de ceux qui y ressemblent : la gestion d'une salle de marché, avec tout le traitement des données boursières que cela comporte, ne fait ainsi pas appel à de véritables systèmes temps réel, car on attend du système qu'il fournisse des informations justes, mais dans un délai qualifié « d'au mieux ». Par contre, le système de pilotage automatique du métro parisien de la ligne 14 (Météor) assure le contrôle complet de l'ensemble des métros présents sur la ligne.

De plus, il faut faire attention, car le temps réel n'est pas forcément là où on l'attend. Il dépend d'un point de vue. Prenons l'exemple de la retransmission en « temps réel » d'un match de football : si l'on prend le point de vue du déroulement du match, l'éclairage du stade doit être contrôlé en temps réel, car toute défaillance va entraîner l'arrêt du match, alors que la panne de la retransmission n'a pas d'impact sur le déroulement du match (par contre vous risquer de rater le but en direct ! ...).

Comme on vient de le voir, c'est le temps, élément différentiateur, qui change tout d'une application à une autre. Il est donc nécessaire de qualifier cette notion.
Là encore, il faut savoir que ce terme recouvre une pluralité de formes :

Le temps de réponse à un événement (signal discontinu dans le temps)
C'est peut-être le temps le plus simple à définir, car il correspond au temps maximum admissible par le

procédé, pour obtenir du système une réponse pertinente à tout événement issu du procédé. Néanmoins, tout système doit être capable de traiter tous les événements issus du procédé en respectant leur temps de réponse spécifique. Dès lors va se poser une kyrielle de questions concernant entre autres la criticité de l'événement et la notion de priorité relative entre tous les événements, car même un système temps réel (le plus puissant soit-il) est incapable de traiter de façon simultanée toutes les occurrences des événements. Le système sera temps réel si, et seulement si, il est capable de traiter tous les événements dans le temps imparti.

Le temps de cycle d'un système

Cette notion est bien appropriée pour traiter les signaux continus dans le temps. Elle s'applique correctement au système échantillonné et se calcule grâce au théorème de Shannon. De plus, le temps de traitement cyclique de l'information doit respecter en tout point le temps de réponse du système (Entrée → Sortie) pour assurer une gestion pertinente du procédé. On peut aussi dire que les systèmes échantillonnés sont relativement simples à mettre en œuvre, car ils sont faciles à dimensionner du fait de leur prévisibilité.

La relativité du concept

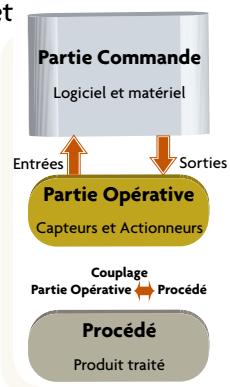
Réaliser un système temps réel, c'est à la fois traiter des informations cycliques et événementielles, voire isochrones, pour prendre en compte les informations multimédias (cas du téléphone où l'on doit à la fois être capable d'assurer un débit constant entre les postes distants et de la vidéo avec des débits de plusieurs dizaines de Mbits/sec hors compression). Cela veut dire que nous devons être capable d'accepter dans un même système tout type d'information, et de respecter pour chaque information les critères temporels et les traitements qui lui sont attachés, en sachant que les résultats de ces traitements doivent être justes et à temps, pour que l'application fonctionne. Le temps réel est une notion bien relative. Quand on en parle, il faut essayer de le qualifier, de le quantifier : c'est en fait le temps de réponse du système combiné avec la charge de travail à effectuer pendant ce temps.

4. Les fonctions dans les systèmes temps réel

Un système temps réel peut être considéré comme un système en couches, dans lequel il existe deux axes de communications privilégiés. Sa complexité impose bien souvent une décomposition en fonctions pour rendre le système compréhensible et réalisable.

Les différents niveaux d'une application :

Le premier niveau correspond au procédé qu'il faut contrôler.
Le second correspond à la partie opérative qui est constituée de l'ensemble des capteurs et actionneurs formant l'interface entre le procédé et la partie commande. Ces capteurs et/ou actionneurs peuvent être élémentaires (un relais, ...) ou intelligents (une caméra, ...). Dans ce dernier cas, ils peuvent assurer le traitement des boucles d'asservissement et/ou les « arcs réflexes ».

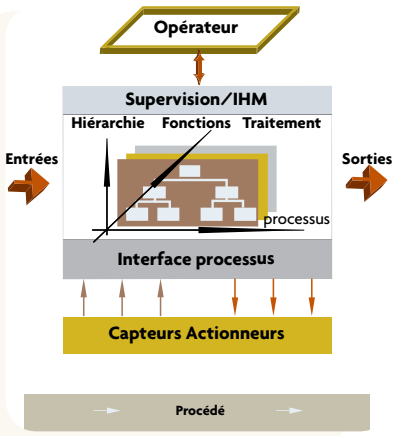


Le dernier niveau assure le contrôle commande de l'ensemble et à ce titre effectue l'ensemble des traitements :

- Les traitements de supervision qui assurent l'interface avec l'utilisateur.
- Les traitements de contrôle commande gérant l'ensemble des entrées-sorties.
- Les traitements des boucles d'asservissements et/ou des arcs réflexes pour ceux qui ne sont pas traitées par les capteurs actionneurs intelligents.

Les axes de communication dans un système :

L'axe vertical permet la transmission des ordres ainsi que la remontée des informations de contrôle commande et des données à enregistrer.
L'axe horizontal supporte la communication inhérente aux traitements de synchronisation entre les différents éléments constitutifs de l'application, ainsi que la communication entre les systèmes adjacents.



La décomposition en fonctions :

Cette décomposition fonctionnelle est le résultat d'une analyse méthodologique mettant en exergue l'aspect fonctionnel, mais aussi toute la problématique des relations industrielles existant entre les différentes sociétés participant à la réalisation du système. À titre d'exemple, la réalisation d'un système de navigation et d'attaque d'un avion engendrerait un découpage de type radar, système avion, contre-mesure, reconnaissance, ... Celui d'un métro serait décomposé en traction/freinage, gestion des portes, pilotage automatique, ...
Il ne faut pas oublier, à l'issue de cette analyse, de

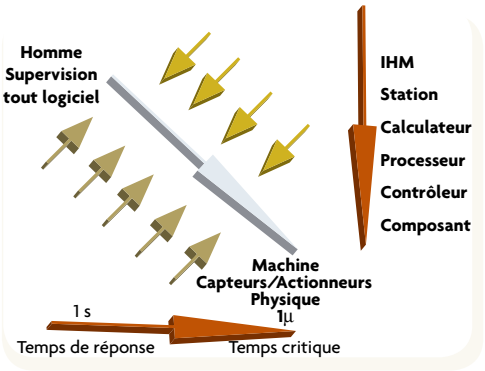
consolider les résultats de façon à obtenir à l'arrivée un système et non pas autant de systèmes que de fonctions.

5. Le temps réel et ses composantes

Toute gestion de procédés nécessite la mise en œuvre d'une pluralité de technologies. Les différents niveaux de l'application qui sont couverts par ces techniques vont de l'homme, pour la supervision des procédés, à la gestion des entrées-sorties pour le contrôle du procédé.

5.1. L'axe temporel

L'axe temporel permet de définir les différentes contraintes temporelles en fonction d'un point de vue : l'homme a un temps de réponse qui est de l'ordre de la seconde.



À l'opposé, il peut y avoir des besoins de temps de réponse beaucoup plus courts, parfois inférieurs à la micro seconde, pour faire fonctionner un système (par exemple la synchronisation des axes d'un robot de « peak and place », les traitements de vos liaisons ADSL, des applications dans le monde du radar, ...). On parle alors de temps critiques.

5.2. Les différents types de matériel

De façon bien évidente, les technologies à mettre en œuvre dépendent directement des performances temporelles à respecter. Le tableau ci-dessous en donne une illustration :

Fonction	Matériel	Système d'exploitation	Temps de réponse
IHM	PC	XP, Linux	>100ms
Contrôle commande global	Station de travail	Lynx OS, HPUX	10ms à 100ms
Fonction TR	Calculateur TR	VxWorks, VRTX	1ms à 10ms (voire 100µs)
Prise en compte Evénement	Calculateur TR	OS + gestion des interruptions	10µs à 100µs
Arcs Réflexes	Composants spécialisés	Machines à états finis	< 10µs

5.3. Les choix offerts au réalisateur

Pour réaliser un système temps réel, le concepteur dispose au moins de 5 axes de liberté. Le véritable problème provient du fait que ces axes de liberté sont étroitement liés : en modifier un revient à corriger des choix effectués sur les autres, pour essayer de toujours trouver le meilleur compromis.

L'architecture

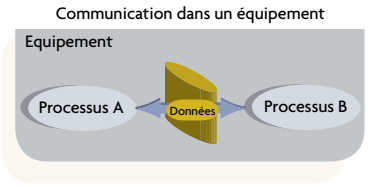
Le premier axe concerne l'architecture, qui peut se décliner à plusieurs niveaux : on parle d'architecture du système et d'architecture(s) équipement(s).

Au niveau système, cette architecture met en œuvre des moyens de communication entre les différents équipements constitutifs du système. Ce premier niveau doit prendre en compte les différentes contraintes de l'application : la topologie, la CEM (Compatibilité Electromagnétique Magnétique), la distance entre les équipements, la distance totale, les volumes de données à transférer, ou encore les contraintes associées telles que le temps de réponse et la qualité de service.

Au niveau équipement, elle doit tenir compte des entrées-sorties et des moyens d'interconnexion avec le reste du système. Elle est basée sur un bus de fond de panier (VME, PCI, ...) permettant d'assurer la communication entre les cartes électroniques qui constituent l'équipement.

L'architecture doit aussi être en parfaite adéquation avec le choix du système d'exploitation, de façon à garantir un niveau de fonctionnalité à l'ensemble.

Les choix résultant de l'analyse de l'architecture vont permettre d'obtenir une solution offrant toutes les garanties vis-à-vis de la modularité, de l'évolutivité mais aussi de la fiabilité et de la maintenabilité afin d'aboutir à un niveau de sûreté de fonctionnement en adéquation avec les besoins de l'application.



La communication entre les processus

La communication entre les processus offre un second axe de liberté. Cette communication a toujours nécessité des soins particuliers.

Dans les années 80, ce problème a été résolu avec la mise en œuvre des « IPC » (Inter Process Communication) sous le contrôle des systèmes d'exploitation.

Nous utilisons maintenant des « systèmes décentralisés ou distribués », qui nécessitent une communication entre processus distants fonctionnant sur des processeurs différents et utilisant des systèmes d'exploitation différents.

Il faut aussi noter que la configuration architecturale et topologique d'un système est généralement différente

d'une application à une autre. De plus, pour une même application, cette configuration peut évoluer ou être modifiée dans le temps en fonction des besoins à satisfaire.

Le besoin de communication entre équipements distants est donc né de cette difficulté. La fonctionnalité du système et sa propension à évoluer dépendent directement du bon dimensionnement de la communication. On parle alors de l'élasticité du réseau, qui se définit par l'aptitude du réseau à tolérer un changement de configuration avec le minimum de programmation possible, afin d'assurer le transfert des données, la synchronisation des processus d'application et d'offrir des services de niveau système.
À ce jour, et pour faciliter la lecture de ce document, nous pouvons classifier les moyens de communications en deux catégories :

- les réseaux dits temps réel : ces réseaux sont eux mêmes subdivisés en groupes en fonction de leur domaine d'activité. En effet, le militaire utilise des réseaux durcis garantissant le déterminisme des échanges d'informations dans le système (mil-std 1553, digibus, ...), tandis que l'avionique civile préfère des réseaux faisant référence aux normes ARINC (Arinc429, Arinc653, ...). À ce titre, AIRBUS vient de développer dans le cadre du programme A380 un nouveau réseau de communication, l'AFDX (Avionics Full Duplex eXchange), qui est basé sur de l'Ethernet redondé. Quant au monde de l'automatisme, il utilise des réseaux comme Fip, Profibus-DP, Interbus-S, ASI-bus et CAN, alors que celui de la robotique emploie des réseaux comme Sercos. Le réseau CAN semble nettement dominer pour l'automobile après avoir supplanté les autres prétendants comme le J1850, ou encore le VAN. De son coté, le ferroviaire essaye de se doter de nouveaux réseaux pour assurer l'interopérabilité des trains et des wagons entre les différentes sociétés nationales de chemin de fer : le TCN (Train Communication Network) comprenant le MVB (Multiple Vehicule Bus) et le WTB (Wire Train Bus).

- les réseaux informatiques : Ethernet, sous ces différentes formes, associé au protocole TCP-IP, est aujourd'hui de plus en plus utilisé comme moyen de communication dans les systèmes temps réel. D'autres réseaux comme l'USB, le WIFI ou encore l'IEEE1394 sont aussi mis en œuvre. Il est vrai que ces réseaux apportent un ensemble de services importants aux systèmes, et notamment des fonctions de type « web services » permettant au travers d'une connexion « intranet voire extranet » d'accéder aux équipements terminaux : il est ainsi possible, via des protocoles de type SNMP, de les contrôler à distance. Néanmoins, il faut quand même se poser un certain nombre de questions car le débit de la liaison et les services offerts ne constituent pas forcément des critères de choix pour faire du temps réel. En effet, si ces réseaux offrent une solution de communication pour les couches hautes ou en tant que liaison spécialisée, qu'en est-il du niveau de qualité de service obtenu (la garantie de déterminisme offerte, la disponibilité de la liaison, ...) pour les couches basses des systèmes ? Car c'est à ce niveau que se joue réellement le « Hard Real Time ».

La technologie des calculateurs

Les solutions offertes concernant la technologie des calculateurs et/ou processeurs sont nombreuses, ce qui peut paradoxalement poser quelques problèmes : parmi cette multitude de solutions, il faut trouver celle(s) qui sera (seront) la (ou les) mieux adaptée(s) pour répondre aux contraintes de l'application.

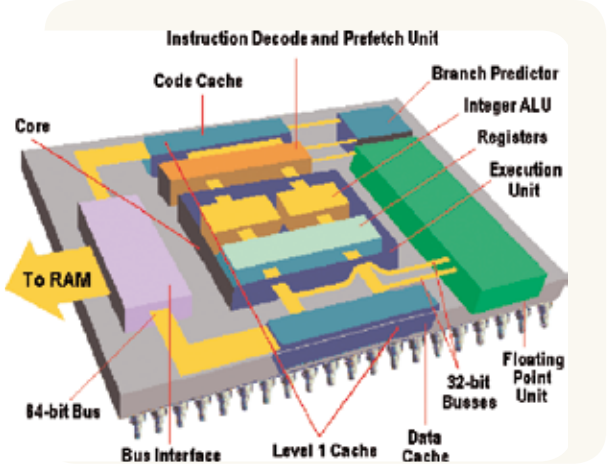
Le monde du PC : Etant donné le nombre d'offres SCADA disponibles sur le marché, le PC sous ses différentes formes semble à ce jour un élément incontournable pour réaliser des fonctions de supervision ou tout simplement d'IHM. De plus, il offre une solution communicante qui, si elle n'est pas toujours techniquement optimale, apporte une réponse pour l'interconnexion de l'application vers les niveaux supérieurs. Il propose un coût qui ne cesse de diminuer pour des performances toujours plus importantes, et cela en gardant une compatibilité ascendante. C'est pour ces mêmes raisons que le PC est aussi souvent utilisé comme frontal pour effectuer les traitements de contrôle commande globaux au système, remplaçant ainsi de plus en plus les stations de travail. On le retrouve aussi dans les équipements terminaux au format PC104.
L'utilisation du PC est encore renforcée du fait de la pluralité de l'offre en systèmes d'exploitation : sont actuellement disponibles bien évidemment les solutions XP et Linux (et ses dérivés : LinuxRT, TRAI, ...), mais aussi des solutions comme VxWorks ou RTX, offrant ainsi du vrai temps réel.

Les stations de travail : Si dans les années 80 à 90, cette solution semblait être la solution pour gérer les systèmes temps réel, voire pour réaliser les interfaces homme/machine avec le logiciel X11 motif, il semblerait qu'elle soit progressivement abandonnée au profit du PC.
Néanmoins, on voit apparaître de nouvelles solutions basées sur des architectures multiprocesseur, toujours plus performantes et utilisant Linux comme système d'exploitation.

Les calculateurs temps réel : Ce type de calculateurs permet de réaliser des architectures dites centralisées. Si, de 1980 à 1995, il n'y a pas eu de doute sur les choix pour la réalisation des calculateurs temps réel : « utilisation d'un rack VME avec une ou plusieurs cartes CPU sur une base de processeurs choisis dans la famille MC68000 puis powerPC et une collection de cartes d'entrées-sorties », les choses sont bien différentes aujourd'hui : le bus VME, malgré un débit théorique maximum de 80 Moctets/sec, ne semble plus suffisant pour les applications de nouvelles générations. Et surtout, son coût devient prohibitif vis-à-vis de solution à base de PC.
Les conceptions à base de rack VME se font donc de plus en plus rares. Seuls quelques « irréductibles Gaulois » persistent dans cette voie, car c'est encore une des rares solutions qui puisse garantir le « hard real time ».
Des groupes de travail recherchent actuellement des solutions de remplacement. Mais la majeure partie de ces nouvelles propositions se heurtent généralement à l'un de ces trois écueils :

- le temps nécessaire à l'aboutissement des travaux puis à l'émergence de la solution ;
- la montée en puissance du PC avec des solutions de plus en plus performantes ;
- les solutions alternatives basées sur une architecture distribuée permettent une délocalisation de l'intelligence au plus près des capteurs actionneurs.

Les processeurs : Sous ce vocable de « processeur », se cache toute une gamme de produits allant du micro-contrôleur le plus élémentaire (un processeur avec des entrées-sorties et de la mémoire sont intégrés dans le même composant), à des processeurs 32 bits, en passant par tout un panel de DSP à virgule fixe ou flottante. Cette gamme de produits est proposée par une pluralité de fournisseurs : Fujitsu, Intel, Motorola, Siemens, Texas Instrument, ...



Beaucoup de ces processeurs sont rendus communicants : on fait appel à des réseaux comme CAN, I2C, des liaisons série à haut débits, ... certains intègrent même un port Ethernet.
Les nouvelles générations de ces processeurs sont, bien souvent, pipelinées afin d'offrir toujours plus de puissance de calcul.
Il est donc possible à chaque utilisateur de choisir judicieusement un processeur en fonction de ces besoins. Mais attention au mauvais choix qui peut également être effectué !

Les solutions spécialisées : enfin, il est toujours possible de faire appel à ces solutions pour satisfaire aux contraintes spécifiques d'un système : une intégration poussée avec l'utilisation d'un cœur RISC ; la possibilité de paralléliser des DSPs ; les contraintes temporelles particulièrement difficiles à tenir nécessitant la mise en place de machines à états finis ; le cryptage et la compression de données ; ou encore la prise en compte du coût afin de satisfaire les contraintes d'une fabrication en série. Il est patent que le recours à ces solutions demandera d'élaborer des fonctions originales, tant du point de vue des développements matériels que logiciels.

Carte PCI ARION 100
utilisant la technologie SOPC -
Carte de communication Temps Réel

Quelle que soit la raison nécessitant le recours à ce type de solution, il nous faut constater un changement important, amorcé à la fin du précédent millénaire, en terme d'industrialisation. Les exemples sont très nombreux et je me contenterai de citer la numérisation des chaînes analogiques pour effectuer les différents traitements au niveau numérique et non plus analogique ou encore l'intégration d'un système complet dans un seul composant avec le recours aux composants programmables à haute densité d'intégration (SOPC) et aux prédiffusés au détriment des solutions à base d'ASIC (SOC) sauf quand le volume de production est suffisant pour amortir les coûts de NRE.
Il est à noter que le recours aux solutions SOPC est principalement dû aux efforts consentis par les principaux fournisseurs de composants programmables que sont Altera, Xilinx, et les autres, qui fournissent des chaînes de développement faciles à mettre en œuvre, augmentent les matrices des composants, font diminuer le coût

unitaire des composants, et autorisent le déploiement des blocs de propriété intellectuelle (évitant ainsi de redévelopper des fonctions standards).

Les systèmes d'exploitation

Le choix du système d'exploitation est un élément important qu'il est nécessaire d'effectuer afin de satisfaire les contraintes inhérentes à l'application. Cette technologie a connu beaucoup d'évolutions pour suivre le cycle de vie des différentes solutions proposées : Unix avec les différentes versions que sont LynxOS, HPUX, ..., Windows avec XP, CE, XP Embedded, Linux, OS9, VxWorks, VRTX, OSE, µCOS, OSEC, ... sans oublier les différents systèmes d'exploitation propriétaires développés un peu partout dans le monde ...

Loin d'être stabilisé, le marché des systèmes d'exploitation temps réel est actuellement secoué par un séisme qui peut se résumer à la question suivante « Faut-il utiliser des logiciels libres ou des solutions proposées et maintenues par des industriels ? ».

Il est intéressant de noter que la réponse varie dans les sociétés, et suivant les individus. Les acheteurs préfèrent souvent les logiciels libres, mais c'est aussi souvent le cas de beaucoup de décideurs : ces logiciels ont en effet l'immense avantage d'être... gratuits ! Les techniciens qui utilisent réellement ces logiciels (eux !) ont en général un avis beaucoup plus mitigé, car entièrement dépendant des contraintes à satisfaire pour l'application.

Sans rentrer dans cette polémique, il est quand même bon de rappeler les points suivants : premièrement, le tout gratuit n'existant pas, il serait utile de se demander où sont les coûts cachés inhérents à la mise en œuvre des logiciels libres. Par ailleurs, la durée de vie des systèmes industriels étant de plusieurs décennies, il serait souhaitable de considérer les procédures de maintenance à mettre en œuvre pour assurer la pérennité du système d'exploitation (les applications ont une fâcheuse tendance à nécessiter des évolutions plus ou moins importantes au cours de leur cycle de vie !). Ensuite, on peut souligner la quasi inexistence de solutions simples aux problèmes qui peuvent être rencontrés dans la mise en œuvre de ces systèmes d'exploitation : poser sa question sur un forum (jeter une bouteille à la mer !) est généralement inefficace, faire appel à une société spécialisée revient souvent très cher, et développer en interne une expertise sur le système choisi, en maintenant opérationnelle l'équipe constituée annihile tout gain financier. Les systèmes temps réel contrôlant de plus en plus notre vie, il devient indispensable de s'interroger sur leur fiabilité : qui peut garantir que tel ou tel logiciel téléchargé est exempt de bugs ? Qu'il est performant ?
La réponse : tout le monde, c'est-à-dire personne... Dès lors, comment garantir la sûreté de fonctionnement du système ? Enfin, il faut rappeler que les logiciels libres sont principalement utilisés sur des machines réalisées à base de PC : le cycle de vie des logiciels libres est donc entièrement dépendant du bon vouloir de la communauté. Dans ces conditions, comment est-il possible de stabiliser une plate-forme pouvant fonctionner sur des machines qui évoluent tous les 6 mois ?

Ce plaidoyer ne veut pas dire que les logiciels libres ne répondent pas à la problématique des systèmes temps réel, mais tente d'éclairer un débat brûlant. Dans tous les cas, il est utile de relativiser, les solutions payantes ne résolvant pas forcément l'ensemble de ces problèmes : la disparition de psos du fait de son rachat par la société Wind River System en est bien la preuve.

Alors que faire ? À mon avis, la seule réponse satisfaisante est de choisir une solution répondant au mieux à l'ensemble des questions qui se posent de façon spécifique pour chaque système, et d'assumer ce choix par la suite. Mais il faut garder à l'esprit que pour réaliser un système temps réel, il n'est pas rare de devoir choisir plusieurs systèmes d'exploitation, suivant les différents niveaux de performance de l'application (pour réaliser une interface utilisateur : Linux, XP, ...sont sûrement de bons candidats ; pour réaliser la gestion des entrées-sorties et le traitement des arcs réflexes VxWorks, OSE, ... apportent des réponses pertinentes).

Les langages

Les langages de haut niveau font aussi partie des critères de choix. De façon incontestable, ils facilitent le développement des applications du fait du niveau d'abstraction offert, et participent ainsi à la démocratisation du monde du temps réel. Leur utilisation procure une portabilité du code pouvant être réutilisé dans d'autres applications. Ce point est d'ailleurs en parfaite harmonie avec les souhaits des donneurs d'ordre.

À cela vient s'ajouter les progrès des compilateurs permettant d'obtenir un taux d'expansion très faible. Ceci a permis d'augmenter le taux d'utilisation des langages de haut niveau, au dépend de l'assembleur qui est seulement utilisé dans les cas extrêmes. Mais qui s'en plaindra ?

Malheureusement cette facilité de développement peut aussi engendrer des comportements néfastes du type « il est facile de coder l'application sans savoir comment le système fonctionne » qui sont à l'origine de dysfonctionnements du système. Il faut ici rappeler

que pour développer une application temps réel, il est nécessaire de savoir de quoi on parle, de connaître le process à contrôler, et non pas de se satisfaire de coder la fonction demandée en dehors de tout contexte d'utilisation. Combien de fois ai-je pu mesurer les résultats d'une telle démarche !!!

Il faut être conscient des enjeux inhérents à la réalisation des systèmes temps réel : ils doivent être opérationnels dans tous les modes de fonctionnement, et cela quelles que soient les occurrences d'évènements ou les pannes détectées.

On arrive donc à un paradoxe : d'un côté, les langages de haut niveau permettent d'appréhender facilement le monde du temps réel, mais de l'autre, on a besoin de connaître le fonctionnement du système lui-même, souvent occulté par des fonctions « magiques » écrites par d'autres ... Quel dilemme cornélien !
Quels sont les langages de haut niveau utilisés dans les applications temps réel ? Eh bien, comme pour les systèmes d'exploitation, on peut trouver de tout, du « C », à l'« ADA » ou encore du « Java » en passant par le « C++ » et cela en fonction du niveau et des contraintes temps réel à respecter. Mais attention, ce choix peut impacter de façon plus ou moins importante les performances du système. Alors, là encore il faut faire le Bon Choix...

5.4. Les contraintes applicables

Dans ce chapitre, nous allons aborder l'ensemble des contraintes applicables à la réalisation d'un système temps réel.

La prise en compte des réels besoins fonctionnels de l'application est la première de ces contraintes. Il ne s'agit pas d'une lapalissade que d'y faire référence ici. En effet, combien de fois nous avons pu constater en fournissant un système aux utilisateurs finaux que celui-ci ne correspond pas totalement à leur besoin : le célèbre dessin réalisé par Piem de l'arbre et de la balançoire en est d'ailleurs la parfaite illustration. La spécification d'un système temps réel est une chose complexe qui est malheureusement réalisée par des personnes, pas toujours au fait des besoins opérationnels, et qui bien souvent confondent solution et besoin. Ce phénomène est d'ailleurs renforcé par les méthodes de développement mises en œuvre, qui sont bien souvent du type « essai → échec → modification » et qui peuvent entraîner une dérive entre le système réalisé et le besoin.
S'ensuit alors un cycle de modifications pour se rapprocher du besoin des utilisateurs ...

La seconde prend en compte les contraintes applicables à l'application telles que :

- Le cycle de vie de l'application : il va de quelques années pour les applications comme la voiture, à plusieurs décennies pour les applications industrielles comme le ferroviaire ou l'avionique.

- L'aspect géographique (on parle alors de topologie de l'application) : elle prend en compte la répartition des équipements dans l'aire couverte par l'application. Elle influe très directement sur l'architecture du système et sur les moyens de communication permettant l'interconnexion des équipements.

- Les choix faits au niveau d'une société concernant : le processeur, le système d'exploitation, le langage, ...

Ces contraintes ne sont données qu'à titre d'exemple car la liste peut être très longue : température, vibration, humidité, compatibilité électromagnétique, le poids, le volume, ...

La troisième concerne la qualité et la sûreté de fonctionnement du système. Cette démarche s'appuie sur au moins 4 qualités que doit présenter le système :

- La fiabilité : c'est l'aptitude d'un système à accomplir le traitement de données dans des conditions d'utilisation et pour une période de temps donnée. Elle est caractérisée par une probabilité de succès.

- La maintenabilité : c'est l'aptitude d'une entité à être maintenue ou rétablie dans un état dans lequel elle peut accomplir une fonction requise, lorsque la maintenance est accomplie dans des conditions données, avec des procédures et des moyens prescrits.

- La disponibilité : elle intègre la disponibilité instantanée (c'est l'aptitude d'un système, sous les aspects combinés de sa fiabilité, de sa maintenabilité et de sa logistique de maintenance, à remplir ou à être en état de remplir une fonction à un instant donné), la disponibilité opérationnelle (c'est la mesure de l'efficacité d'un système en phase opérationnelle. On parle de qualité de service. Elle est influencée principalement par le matériel mobile, les installations fixes, la logistique), la disponibilité intrinsèque du matériel (c'est la part prise par celui-ci dans la disponibilité opérationnelle. Elle est évaluée par la qualité de service et correspond à la perception par l'usager de la qualité et de la régularité du service : retard, immobilisation, répercussion sur l'exploitation) et de la disponibilité du système (elle se calcule comme

étant le temps moyen de bon fonctionnement divisé par le temps moyen entre avaries).

◇ La sécurité : c'est l'aptitude d'une entité à éviter de faire apparaître, dans des conditions données, des événements critiques ou catastrophiques. On peut parler de sécurité intrinsèque - les équipements conçus sur ce principe sont tels que leur défaillance, quelle qu'elle soit, ne porte pas atteinte à la sécurité du système - ou de sécurité probabiliste. Cette dernière s'applique à un équipement dont il est possible de calculer la probabilité d'occurrence des défaillances pouvant engager la sécurité.

On aboutit ainsi à la sûreté de fonctionnement : c'est l'ensemble des méthodologies mises en œuvre pour qualifier un système. La démarche est étroitement liée à celle de la démarche qualité. La démonstration de la sûreté de fonctionnement est établie à partir de l'étude FMDS (Fiabilité, Maintenabilité, Disponibilité, Sécurité), et d'une analyse des risques. Elle débouche sur un dossier de sécurité. Le but de toute cette démarche est d'éviter ce genre d'accident ...

La quatrième contrainte concerne le coût du système qui doit toujours être le plus faible possible, raison pour laquelle nous faisons de plus en plus fréquemment appel à la démarche COST et aux produits « consumer ». Il est à noter que cette démarche est bien souvent en opposition avec la durée de vie du système. Un autre point concerne le fait que souvent nous ne parlons que du coût d'achat (ou de réalisation) du système. Or, vue la durée de vie de ces systèmes, nous devrions prendre en compte son coût de possession qui intègre, en plus du coût d'achat, celui de l'exploitation du système et de la maintenance associée. Ce coût, bien que plus difficile à calculer, est bien le seul intéressant pour un maître d'ouvrage car c'est ce que le système va effectivement lui coûter au cours de son cycle de vie. À titre d'exemple, nous pouvons citer le cas de la RATP qui, pour diminuer le cycle de maintenance de ces matériels roulants, y a installé des systèmes de contrôle spécifique pour ne plus réaliser de la maintenance préventive mais de la maintenance prédictive. Le gain d'une telle démarche est de limiter les actions de maintenance à celles qui sont strictement nécessaires. Le coût de la maintenance est ainsi optimisé, mais on limite également le nombre des matériels à acheter, puisqu'on a augmenté leur disponibilité.

6. Les applications temps réel et l'état de l'art

Le consommateur tire le marché

C'est maintenant le « consumer » qui tire le marché et plus particulièrement, au niveau du PC, le marché des jeux vidéo. Il a engendré ces dernières années plusieurs modifications d'architecture pour assurer la connexion de cartes vidéo de plus en plus performantes : l'arrivée du bus AGP, puis retour au bus PCI pour maintenant passer au PCI express. Si cette dynamique a du bon pour obtenir toujours plus de performances, elle est contre-indiquée en ce qui concerne la pérennité des systèmes qui doivent vivre dans le monde industriel plusieurs décennies.
La référence permanente au monde du consumer a un autre impact : on a tendance à comparer le coût des systèmes temps réel à celui d'un simple PC. La démarche « cost » se généralise : pour éviter de développer des solutions spécifiques, on souhaite acheter des « produits » sur étagère de préférence multisource. Avant toute utilisation de produit « cost », il serait bon de répondre aux questions suivantes : ces produits répondent-ils à l'ensemble des contraintes inhérentes à l'application ? Est-on capable de les qualifier ? L'interfonctionnalité de ces produits est-elle avérée ? La notion de multisource est-elle réelle ? Quel sera le niveau de performances obtenu ? ... Malheureusement, dans bien des cas, nous ne pouvons répondre par l'affirmative à toutes ces questions. Alors dans ces conditions, comment faire pour mettre en pratique cette démarche « cost » ?

Le TCP/IP

Le politiquement correct tend à influencer la moindre de nos décisions : à titre d'exemple, toute solution qui n'utiliserait pas « TCP-IP » comme protocole de communication serait considérée comme « has been ». Et pourtant, ce protocole est tout sauf temps réel. S'il apporte toute sa puissance pour les échanges d'informations dits non temps réel, (il permet effectivement au travers de services d'assurer la maintenance, ...), il reste toujours à résoudre le problème de l'intrusion, qui engendre des risques importants de sécurité, et surtout la non prise en compte du déterminisme de la communication. Mais là encore, cette indétermination peut être acceptable pour certaines applications et inconcevable dans d'autres.

Les standards

Le respect des normes et standards : ce point incontournable se retrouve au centre d'enjeux industriels importants.

J'en veux pour preuve la démarche de normalisation des réseaux de terrain fin des années 80 : les utilisateurs pensaient que cette démarche de normalisation allait assainir le marché, mais il n'en fut rien. Les lobbies politiques ont quand même réussi à normaliser 3 réseaux entièrement incompatibles, tandis que les industriels, pour promouvoir leurs produits, se battent à coup de normes, ce qui ne facilite pas l'émergence de produits innovants.
Sur ce sujet, il faut avancer avec prudence, car si les normes sont des éléments nécessaires à l'obtention d'une certaine harmonie des produits, elles ne sont en aucun cas les garantes de leur interopérabilité, au moins pour 3 raisons : la première est purement technique, car la mise en adéquation d'une norme est une chose compliquée à réaliser, et surtout elle nécessite avant toute implémentation une interprétation de cette norme ainsi qu'un choix du profil de référence (les réseaux comme Smart-Distributed-System et DeviceNet, respectivement proposés par Honeywell et Rockwell-Automation, sont des réseaux CAN répondant à la même norme ISO 11898 et pourtant ils sont incompatibles !). Par ailleurs, une norme n'est applicable qu'à un élément du système et non pas au système global. Enfin, une dernière difficulté est liée aux différents aspects marketing et commerciaux développés par les industriels, qui cherchent à la fois à se différencier de la concurrence et à augmenter leur part de marché.

Il faut noter que du point de vue des industriels, ces démarches ne sont pas choquantes car ils cherchent avant tout à vendre leurs produits, tout en faisant des efforts importants pour les développer dans le respect de ces normes. Pour les donneurs d'ordre, cette démarche apporte une « garantie » de qualité et un certain niveau de cohérence entre les produits permettant peu ou prou une interopérabilité. Elle apporte une meilleure compréhension et dans le cas des normes inhérentes à un métier, elle permet une meilleure approche du domaine considéré (exemple : la norme CEI 61850 dans le cadre de la gestion des stations électriques de puissance). Mais in fine, c'est le donneur d'ordre qui doit être à l'origine de la norme, et les industriels qui doivent être à l'écoute de leurs besoins si on veut aller vers l'interopérabilité. Dans certains domaines, cette démarche semble se mettre en œuvre, mais est-ce bien vrai dans tous les domaines ?

Un autre aspect normatif existe, il concerne les démarches à mettre en œuvre pour obtenir des systèmes temps réel de qualité. C'est le cas avec les démarches ISO9000 et 14000 maintenant, mais aussi avec les normes spécifiques à la sécurité comme par exemple les normes EN50126, EN50128 et EN50129 dans le ferroviaire, la norme EN61508 pour l'automatisme, ou encore la DO178 et la D0254 pour l'aéronautique, les normes CEM, ... L'utilisation de ces normes est imposée par le maître d'ouvrage, voire par le législateur quand il s'agit de la sécurité des personnes.
Une autre constante est actuellement bien ancrée dans notre industrie : les équipes de développement du matériel régressent, au profit des équipes de développement du logiciel. Les impacts d'une telle évolution sont multiples : on se contente alors de réaliser une sorte de « mécano » à partir de produits du commerce correspondant le plus possible au besoin, et on privilégie le traitement par logiciel pour réaliser une fonction : une fois l'équipe projet dissoute, on peut imaginer la difficulté de maintenir ou de faire évoluer l'application.

C'est l'ensemble de ces contraintes qui orientent à ce jour le marché. Si on ne peut être que d'accord sur les grands principes édictés, il faut prendre garde à ce que ces vœux pieux ne soient pas en complète contradiction avec les véritables besoins de l'application.

Et demain de quoi sera-t-il fait ?

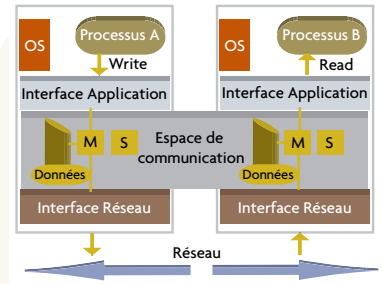
Une chose semble certaine : c'est la marche en avant vers la standardisation. De ce fait, les grandes tendances d'aujourd'hui ne peuvent que se prolonger dans les prochaines années avec l'utilisation de plus en plus intensive de solutions à base de PC, de protocoles comme TCP-IP basés sur des réseaux informatiques comme Ethernet pour assurer la communication, des OS comme Linux, WxWorks, XP, ... pour gérer les applications écrites avec des langages de haut niveau.

Le seul doute restant concerne la capacité des industriels à répondre aux demandes concernant les aspects de fiabilité et de disponibilité de la solution. Une réponse pourrait provenir du fait des processeurs « dualcore ». Ajoutez à cela un système d'exploitation capable de gérer les processus temps réel sur un processeur, les non temps réel sur l'autre, et vous obtenez une solution fort intéressante, mais qui existe déjà ! Pour l'obtenir, il suffit d'ajouter le système d'exploitation RTX à XP, mais il faut un certain niveau d'expertise pour gérer cet assemblage. Et si Microsoft, la communauté Linux ou Wind River System, ... intégrait le tout dans le même système d'exploitation de façon à faciliter son utilisation ?

Une seconde chose semble sûre : elle concerne la mise en œuvre de systèmes décentralisés faisant appel à des capteurs actionneurs intelligents. Cette architecture offre beaucoup d'avantages : elle permet une répartition facile des compétences, une intégration plus aisée, ainsi qu'une plus grande évolutivité du système. À contrario, cette architecture se heurte aujourd'hui au manque de

performances temps réel de la communication entre les différents équipements constitutifs du système. Seule la réponse à ce problème permettrait d'asseoir cette architecture. Les industriels y travaillent déjà et proposent des solutions ouvertes comme l'IEEE1588, ou d'autres solutions métiers comme PowerLink, Profinet, EtherCat, ...

Un autre élément concerne le cycle de développement. Sans entrer dans les détails, nous pouvons facilement dire qu'il se décompose en au moins 4 phases : la spécification, la conception, la réalisation et la validation.
À ce jour, on passe d'une phase à l'autre en transmettant un simple dossier d'une équipe à l'autre, c'est-à-dire qu'il existe un risque potentiel d'interprétation par les uns de ce qui a été fait par les autres. Ce risque non négligeable génère des surcoûts dans la réalisation du système : un souhait des industriels serait alors de pouvoir passer d'une phase à l'autre en assurant une parfaite continuité, et de pouvoir distribuer cette application de façon transparente sur un ensemble de calculateurs communicants.
Dans le passé, les universitaires avaient déjà essayé d'innover dans ce sens, avec par exemple Corba Temps Réel : les industriels s'y sont risqués avec des projets comme l'ASAAC en avionique. Mais, à chaque fois, le problème rencontré provenait de la communication trop gourmande en couches protocolaires, il fallait alors choisir entre faire fonctionner cet ensemble de communication ou les processus d'application.
Aujourd'hui, une autre approche semble voir le jour grâce aux travaux d'universitaires (Distributed Simulation Communication Through an Active Real Time Database – Marcus Brohede and Sten F. Andler – Department of Computer Science – University of Skövde, Sweden), publiés en partie dans la revue IEEE. Même si la clé du problème reste liée à l'obtention d'une véritable communication temps réel et à la fourniture de services systèmes, la solution proposée permet d'ouvrir la voie à la distribution de l'application dans le système.



Un groupe d'industriels travaille aussi sur ce sujet et propose des solutions opérationnelles respectant tous les standards en vigueur. Il est constitué de Wind River System pour les ateliers logiciels et les systèmes d'exploitation, d'Altéra pour les composants, d'Arion pour la solution temps réel et la distribution des applications. Cette solution a été présentée pour la première fois au salon Automation 2005, et des industriels comme Dassault Aviation l'ont déjà retenue. Cette solution permet de voir tous les calculateurs de l'application comme un seul et unique. Le rêve devient enfin réalité.

Du coté équipement, une solution de type SOPC, qui permet la réalisation du système sur un seul composant, semble prendre de plus en plus d'importance : grâce à un AGL de haut niveau supportant des langages comme le VHDL, cette solution autorise des designs intégrant à la fois des fonctions spécifiques réalisées à base de machines à états finis, le (les) processeur(s) (coeurs RISC, DSP, ..) et aussi le logiciel (application et système d'exploitation) du fait de la présence in situ de mémoire de grande taille.

Demain comme aujourd'hui, le monde du temps réel sera exigeant : pour le satisfaire, il faudra l'appréhender dans le cadre d'une démarche philosophique, mixant pragmatisme et technologie de pointe, sans pour autant omettre les contraintes de l'application, l'accompagnement des utilisateurs, ainsi qu'une parfaite connaissance du procédé à contrôler, et cela sans oublier le fameux « time to market ». Dès lors, la pluralité des solutions offertes apporte pour chaque application une possibilité de réponse pertinente susceptible de respecter en tout point les contraintes de l'application. Reste à chacun à trouver la sienne ...

Christian Garnier
Responsable du Laboratoire Temps Réel à l'ISEP et Consultant senior pour la Société ARION

Il a eu, au cours de sa carrière, la chance de pouvoir concilier divers points de vue concernant le monde du temps réel et ainsi de se forger une solide expérience :
- en tant qu'ingénieur, il a travaillé dans les départements de R&D de grands industriels : TITN, CII, Cern, SFIM, Dassault Aviation, ...
- en tant que consultant senior, il est intervenu sur une pluralité de systèmes dans quasiment tous les domaines d'activité : les chaînes industrielles (Renault Automation), l'automotive (Valéo), l'aéronautique (Hispano Suiza), le ferroviaire (RATP), ...
- en tant que fournisseur, il a étudié, conçu et fabriqué des calculateurs embarqués à base de SOPC pour Inéo, ...
- en tant qu'expert en sécurité ferroviaire, il a évalué le niveau de sécurité de système du tramway de Clermont-Ferrand, le niveau de sécurité de calculateur développé par Techniatome, ...
- en tant qu'enseignant chercheur, il a œuvré sur les solutions de demain ; il a en particulier travaillé sur la problématique de la communication objet dans les systèmes distribués.