■ Modèles d'exécution et ordonnancement

Cette architecture logicielle peut être vue comme un ensemble de tâches synchronisées, communicantes et partageant des ressources critiques. Le rôle essentiel du système informatique est donc de gérer l'enchaînement et la concurrence des tâches en optimisant l'occupation du processeur, cette fonction est appelée l'**ordonnancement**. Ce principe d'ordonnancement est un point crucial des systèmes de contrôle-commande ; en effet l'ordonnancement va déterminer les caractéristiques temporelles et être le garant du respect des contraintes de temps imposées à l'exécution de l'application.

Nous pouvons distinguer deux modèles d'exécution de ces systèmes de contrôle-commande : l'exécution dite **synchrone** et l'exécution **asynchrone**. Nous allons présenter ces deux modèles d'exécution à l'aide d'un modèle d'application très simple. Cette application, constituée d'un ensemble de tâches pour gérer le procédé, intègre en particulier les deux tâches suivantes :

– Tâche de lecture des données entrées par l'opérateur à l'aide d'un clavier, appelée « Lecture_consigne ». L'intervention humaine fait que cette tâche peut être longue.

– Tâche d'alarme qui se déclenche sur un événement d'alerte correspondant au dépassement d'un paramètre critique, appelée « Alarme ». Celle-ci doit s'exécuter au plus vite pour éviter l'endommagement du procédé.

Pour mettre en avant les différences entre les deux modèles d'exécution, nous allons étudier la situation dans laquelle la tâche « Lecture_consigne » s'exécute et la tâche « Alarme » demande son exécution alors que la tâche « Lecture_consigne » n'est pas terminée.

Dans le modèle d'**exécution synchrone**, la perception de l'occurrence de tout événement par le système est différée du temps d'exécution de la tâche en cours. Dans l'exemple proposé, nous pouvons constater que la prise en compte d'un signal d'alerte n'est effective que lors de la fin de la tâche « Lecture_consigne » (figure 1.8). D'un point de vue du procédé, la réaction est perçue comme différée, alors que du point de vue du système informatique, elle est perçue comme immédiate. L'occurrence

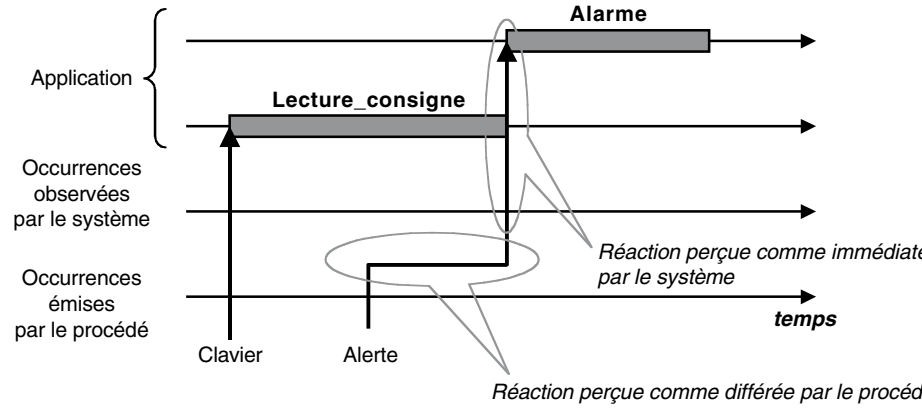

Figure 1.8 – Modèle d'exécution synchrone d'une application de contrôle-commande.

des événements externes a donc été artificiellement synchronisée avec le système informatique, d'où le nom d'exécution synchrone.

Ce retard peut affecter la prise en compte de n'importe quel événement, quelle qu'en soit la gravité pour l'application. Il faut donc vérifier que l'architecture opérationnelle choisie permettra de prendre en compte les contraintes temporelles : hypothèse de la fenêtre de visibilité des événements ou d'instantanéité des actions. La capacité du système à appréhender un événement externe est caractérisée par la durée de la tâche la plus longue puisque les tâches sont non interruptibles ou **non préemptibles**.

Dans le cas du modèle synchrone d'exécution, nous avons un système d'ordonnancement complètement prévisible et, en conséquence, il est possible en faisant une analyse exhaustive de l'exécution de produire une séquence d'exécution qui est jouée de façon répétitive. Cette étude de la séquence est appelée analyse de l'ordonnancement **hors ligne**. L'ordonnancement peut se réduire à un séquencement. Nous avons alors un environnement informatique très simple de l'application développée puisqu'il se réduit à une liste de tâches à exécuter. L'environnement informatique pour piloter cette liste de tâches se réduit à un système très simple : un **séquenceur**.

Dans le modèle d'**exécution asynchrone**, l'occurrence de tout événement est immédiatement prise en compte par le système pour tenir compte de l'urgence ou de l'importance. Dans l'exemple proposé, nous pouvons constater que la prise en compte d'un signal d'alerte est immédiate sans attendre la fin de la tâche « Lecture_consigne » (figure 1.9). La prise en compte de l'événement « alerte » est identique pour le procédé et le système informatique. L'occurrence des événements externes n'est pas synchronisée avec le système informatique, d'où le nom d'exécution asynchrone.

Dans ce contexte, nous avons des tâches qui sont interruptibles ou **préemptibles**. En conséquence, l'ordonnancement n'est pas totalement prévisible et l'analyse de l'exécution des tâches doit se faire **en ligne** par simulation ou par test. Cela nécessite l'utilisation d'un gestionnaire centralisé des événements et de la décision d'exécution : **exécutif ou noyau temps réel**.

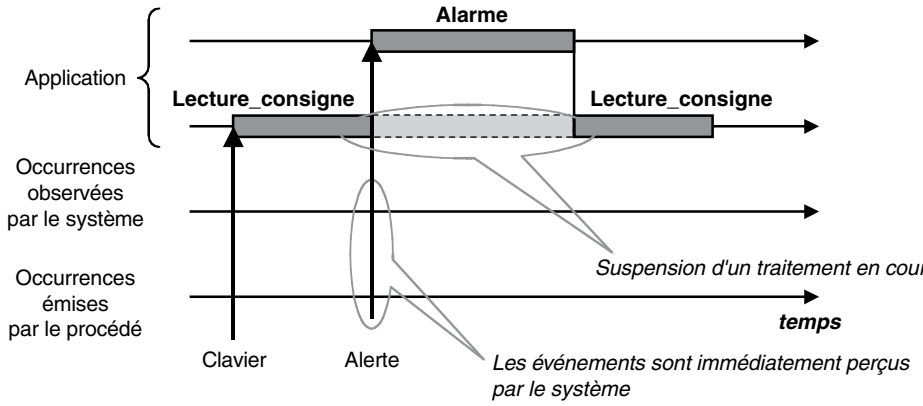

Figure 1.9 – Modèle d'exécution asynchrone d'une application de contrôle-commande.

© Dunod – La photocopie non autorisée est un délit.