Conception et Validation des Systèmes Temps-Réel : Architecture, Parallélisme et Sûreté de Fonctionnement

V. David
Expert Senior IRSN
Expert Systèmes Temps-Réel, Sûreté de Fonctionnement
Précédemment Krono-Safe, Directeur Technique
Fondateur du LaSTRE du CEA LIST
Pr. INSTN

Introduction

Un Système de traitement de l'information est appelé à remplir
une MISSION **:**

- mise en œuvre d'un certain nombre de *fonctions*,
- *suites* d'instructions codées (programme),
- des informations *codées* et *organisées*.

La réalisation :
- nécessite la mise en œuvre de ressources matérielles (H/W relativement figées),
- et de logiciels (M/W et le S/W spécifiques).

Ceci implique :

- **une** conception conforme **à la mission du système**
- **une** réalisation conforme **à la conception**

Faire une analyse des besoins, et spécifier :

- **pour la conception, la mise en œuvre, le codage**
  - **des méthodes et des règles**
  - **des techniques élémentaires**

Qualité des logiciels

Ils doivent être :

- **Efficaces (réaliser les fonctions *requises* avec des performances *adaptées*)**
- **Fiables : corrects, complets, sûrs**
- **Testables : compréhensibles, lisibles, structurés, auto-descriptifs**
- **Transportables (≠ machines)**
- **Maintenables (corrections)**
- **Réutilisables (évolutions)**
- **Certifiables/Qualifiables (Démontrer le respect des exigences)**

Quelques chiffres, fin des années 80 :

- **le logiciel représente 70% à 80% du coût**
- **50% du budget pour les tests et la maintenance**
- **60% des erreurs dues à la conception (80% en coût)**
- **54% des erreurs sont détectées chez les clients**

Aujourd'hui :

- **coût estimé des « bugs » logiciels : 60 Milliard de $**

Un *concept* général, l'approche modulaire :

- **permet la division de la complexité d'un problème**
- **permet le partage du travail à réaliser**

Les méthodes pour la cohérence des modules :

- **cohérence logique (même classe de pb)**
- **cohérence temporelle (synchronisations)**
- **cohérence procédurale (org. des algorithmes)**
- **cohérence par données communes**
- **cohérence fonctionnelle**

Degré de Maturité des équipes

- **Classification du Software Engineering Institute**
  - **Initial où chaotique**
  - **Reproductible**
  - **Défini**
  - **Géré**
  - **Optimisé**

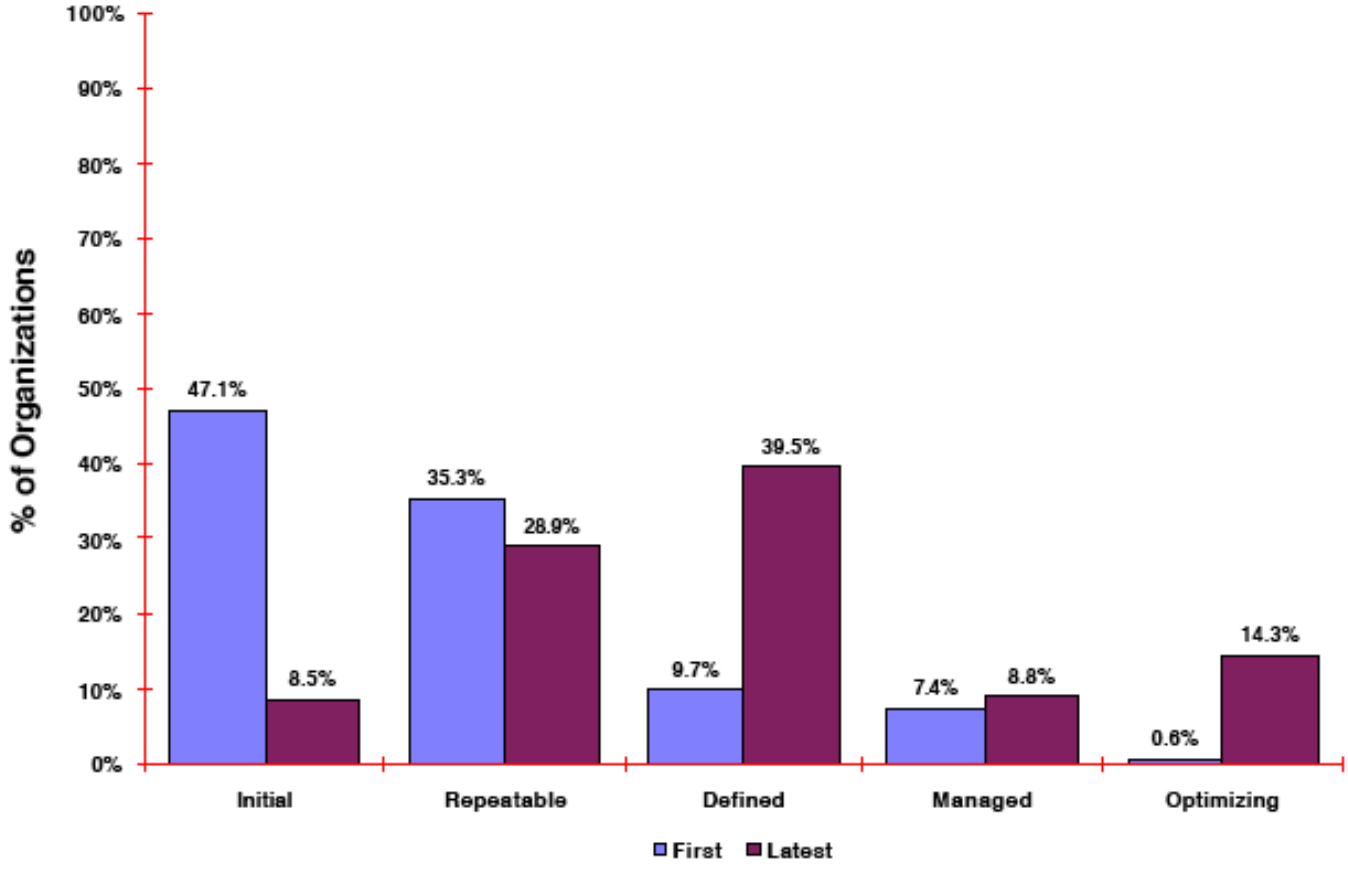
1991, 47% des équipes testées, niveau initial.
Production du logiciel : Problème industriel

Terme même de Génie Logiciel fait rentrer l'informatique dans le cadre industriel

Nécessité d'identifier des étapes liées à des activités et de les relier.

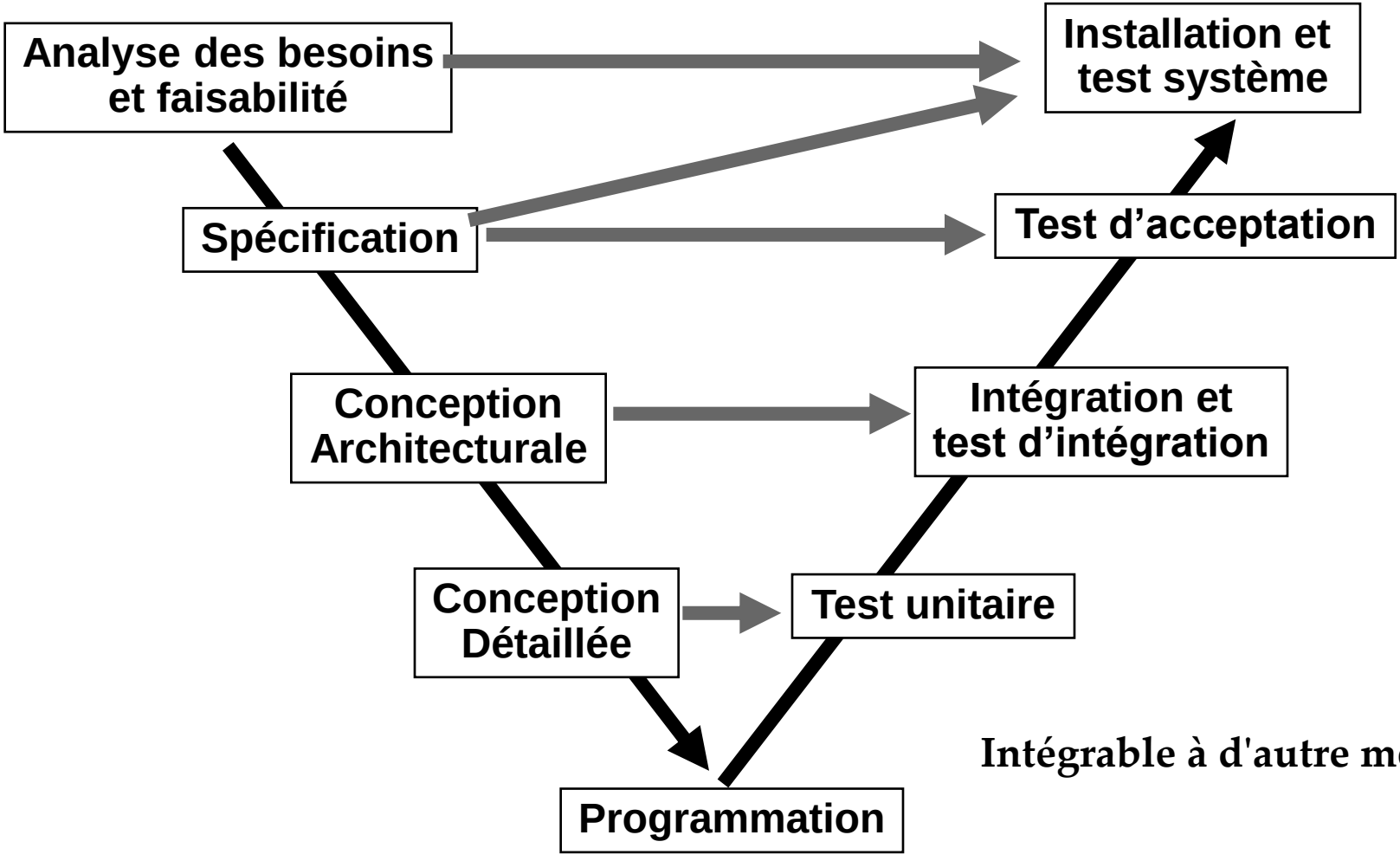
Évaluation de 2003

Maturity Level of First and Latest Appraisal



Modèle en V

Les premières étapes doivent prendre en compte les dernières



Intégrable à d'autre modèles

Activités

Consensus général pour :

- **Analyse des besoins**
  - **Experts du domaine d'application**
  - **Environnement, rôle, ressources, contraintes**
- **Spécification globale**
  - **Description de l'action du logiciel, pas de décision sur la réalisation**
- **Conception architecturale détaillée**
  - **Décomposition du logiciel, spécification des interfaces, description de la réalisation des composants**
- **Programmation**
  - **15% de l'activité**
- **Gestion de configuration et intégration**
  - **Permettre l'évolution des composants, assemblage**
- **Validation et vérification**
  - **Adéquation aux besoins, satisfaction de la spécification**
  - **Analyses, tests**

CAS DES SYSTEMES TEMPS REEL

Présentation

Reproductibilité des tests

Ordonnancement

Systèmes critiques

Tests et déterminisme

Définition d'un système temps-réel :

C'est un système qui a la capacité de répondre à des événements asynchrones issus du monde physique dans des délais prédéterminés

Hard Real-time system (time-critical) :

- **risque de dysfonctionnement si les contraintes temporelles ne sont pas respectées**

Life Critical real-Time System (safety-critical) :

- **idem, mais incident → accident**

Système simple

Notion de flot de données

ei : entrées vues par le système

si : sorties réalisées par le système

$$si = F(e_i)$$

$$si = F(e_0, \ldots, e_{i-1}, e_i) = F(E_{i-1}, e_i)$$

Caractéristiques :

- **non terminaison du système**
- **cadencement événementiel**
- **les traitements et les E/S se succèdent**
- **le système doit pouvoir attendre une entrée**

Programmation en boucle :

- **systèmes purement séquentiels**
- **systèmes cycliques**
- **machines à états**
- **automates**
- **approches synchrones**

Système temps-réel

Les flots de données suivent des *lois temporelles*

ei, ti : événements reçus par le système S

si, wi : événements émis par le système S

si = F (Ei-1, ei, ti)

wi= G (Ei-1, ei, ti)

(si fonctionnelle …)

Système temps-réel *numérique*

On passe d'un milieu *continu* à un milieu *discontinu*

Les lois événementielles sont discrétisées :

effets souvent imperceptibles,

mais *réels*…

Heisenbugs dans les systèmes non déterministes

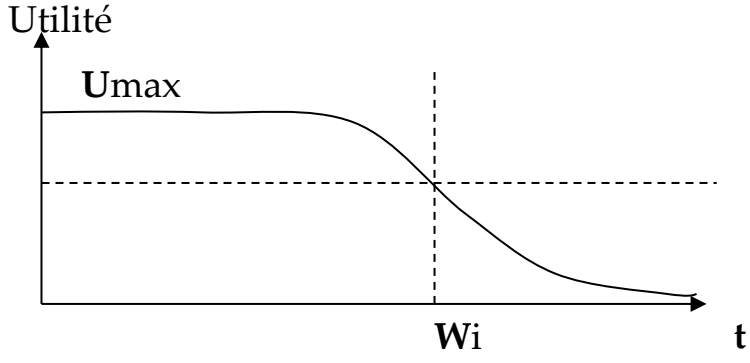
(i.e. non reproductibles par rapport aux conditions initiales et événementielles)

Reproductibilité et tests

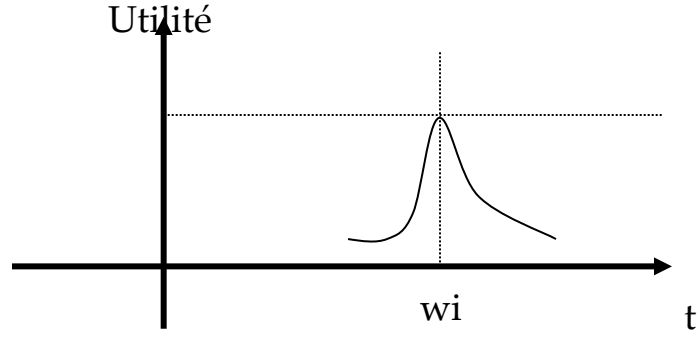
- **Les défauts logiciels**
  - **systématiques**
- **Les défaillances dues aux logiciels**
  - **systématiques (e.g. division par zéro)**
  - **non systématiques (e.g. synchronisation et non cohérence des données)**
    - » **difficulté majeure pour la sûreté de fonctionnement**
- **Les approches statistiques sont inadaptées pour le logiciel**
  - **dès que la probabilité de défaillance admissible du système est très basse**
  - **le logiciel n'a pas de fiabilité quantifiable**
- **Objectifs pour la sûreté de fonctionnement des systèmes à logiciel prépondérant**
  - **comportements du systèmes prédictibles**
  - **tests du systèmes reproductibles**
    - » ***Construction du déterminisme***

Utilité Temporelle

Courbe d'utilité normale



Cas critique



Le caractère *temps-réel* d'un système découle de ses *spécifications (exigences)*

Les lois d'arrivées des entrées doivent être strictement définies et connues

NB : très souvent spécifiés de manières incomplètes :

- interactions sur les données communes

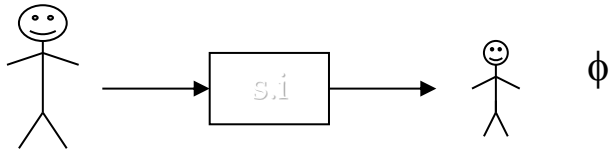
- utilités des résultats

NB (bis) : Le terme « temps-réel » est employé à tort et à travers : tout est temps-réel !

- **Systèmes simples pour lequel la contrainte temps-réel peut être négligée**
  - **la messagerie électronique**
- **Systèmes complexes fondés sur des techniques courantes en temps-réel**
  - **un système d'exploitation multi-utilisateurs**

Relations entre les E/S

Systèmes de transmission

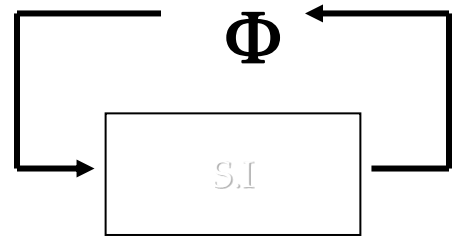


Il peut être temps-réel :

- **débits ou temps de réponse,**
- **fiabilités,**
- **disponibilité.**

NB : un être humain peut toujours attendre un peu

Systèmes de contrôle-commande



Le système physique Φ *impose* ses lois de commande

Les entrées et les sorties se font en temps-réel

Le système est en boucle fermée

Systèmes C-C multifonctions

- **Asservissements**
- **Alarme**
- **Protections**
- **Dialogues opérateur**
- **Communications inter-systèmes**

⇒ plusieurs flots de contrôles

- **Approches Multitâches**
- **1 seul système global à *prédire***

⇒ composition des flots multiples en 1 seul

- **produit synchronisé d'automates ou composition de tâches**
- **ordonnancement statique ou dynamique**

NB : « modulaire » ne signifie pas nécessairement « compositionnel »

La notion de flot de contrôle multiple est liée :

- **aux acquisitions non synchrones (loi ≠),**
- **aux différentes courbes d'utilités.**

Critère de priorités :

- **utilité temporelle des résultats,**
- **importance de la fonction**

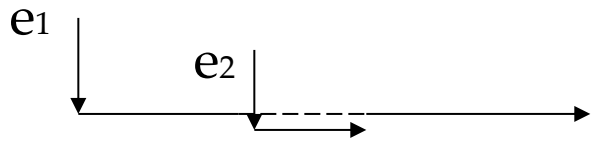
Conséquences :

- **préemption (interruption) de traitement long au profit de traitement cours,**
- **entrelacement possible de plusieurs suites d'instructions (problèmes de cohérence possibles),**
- **ou bien, éviter les arrivées asynchrones (difficile en pratique).**

Ordonnancement des traitements

La programmation multitâche

Gérer les différentes échelles de temps :



- préemption possible = utilité (importance, échéance),

- ordonnancer de façon cohérente et planifiée.

Avantages :

- grande souplesse, facile, optimisation possible.

Inconvénients :

- cohérence des données (entrelacement),

- pertinence (violation d'une contrainte essentielle),

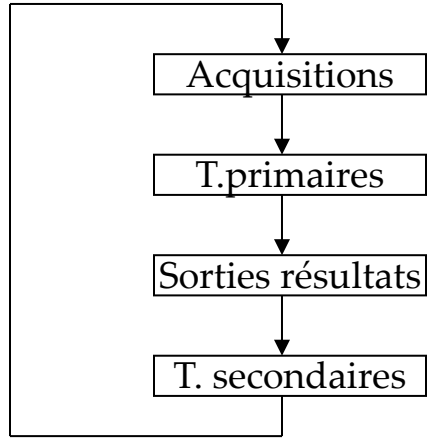
- partage de ressources (dead-lock, famine).

Il n'existe pas d'algorithme « magique ».

La programmation en boucle

On évite le multitâche en adoptant une programmation en boucle :

- **flot de contrôle statique,**
- **Pas de préemption.**



Avantages : très simple, garantie par le temps de cycle.

Inconvénients : rigide, pas optimale, peu performante.

Composition de flot de contrôle

- **Compositions « parallèles » de programmes**

  P = P1 * P2

  Connaissant P1 et P2, que peut-on dire de P ?

  Caractériser les interactions, explicites ou implicites :

  - **cas asynchrone (produit possible ou pas),**
  - **cas synchrone : produit synchronisé d'automates.**

  La programmation en boucle permet une composition rigide, simple, mais ni facile ni efficace.
  Une préemption peut introduire une désynchronisation.

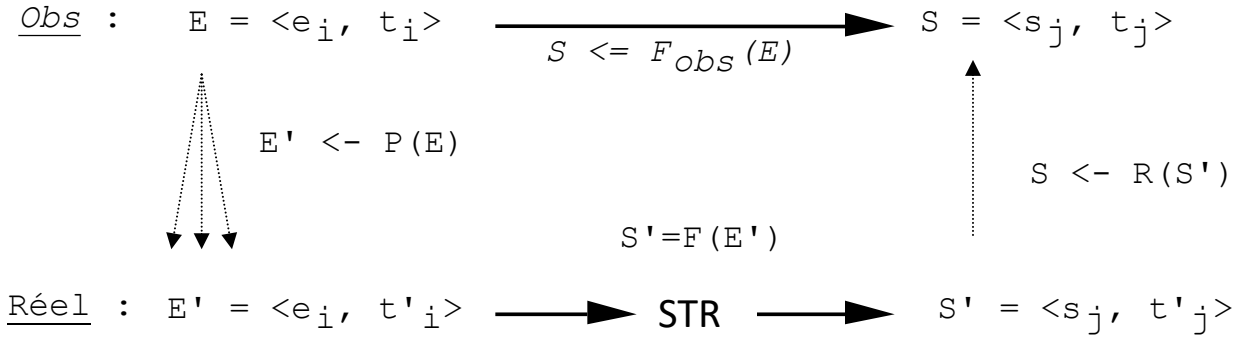
  N.B.1 : approche modulaire ≠> approche compositionnelle
  N.B.2 : CSP (Occam, Ada) obéit à cette logique.

Systèmes critiques

- **Confiance fondée sur quelques grands principes :**
  1. **robustesse, tolérance aux fautes**
     (détection et confinement d'anomalies, modes dégradés, etc.)
  2. **maîtrise des mécanismes mis en œuvre**
     (ex : problèmes des logiciels à interruptions)
  3. **déterminisme comportemental**
     (reproductibilité des tests et essais : les mêmes causes impliquent les mêmes effets)
  4. **application du principe de diversification**
     (prévenir les pannes de cause commune, ex : redondance ou div. fonct., etc.)
  5. **défense en profondeur**

Tests et déterminisme

- **Signification de l'observation d'un système temps-réel testé :**



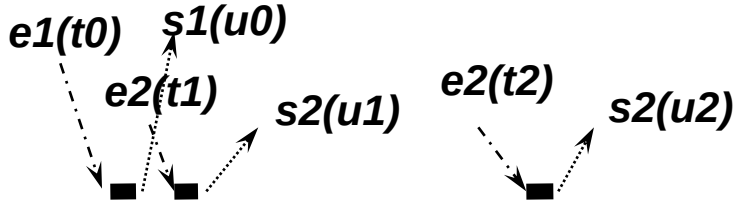
- **Le scénario de test E défini est perçu de façon différente par le système temps-réel qui observe son environnement à sa propre cadence (qq. ns) :**
  - **E est défini avec une précision qui fait partie des spécifications, mais P n'assure pas $P(t_i^1)=P(t_i^2)$ quand $t^1=t^2$ (modulo la précision)**
- **Pour un système non déterministe (ex. : multitâche conventionnel), S' (et donc S) n'est pas déterminé par E :**
  - **le test a peu de valeur pour la validation du système**
- **Il faut agir en amont pour assurer structurellement le déterminisme**

« C4 »

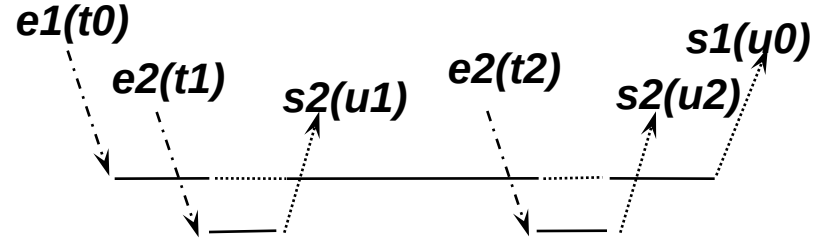
- **Activités à exécuter sur des calculateur :**
  - **Acquérir les données décrivant le procédé et l'environnement (entrées)**
  - **Traiter ces données (algorithmes)**
  - **Commander les actionneurs (sorties)**
- **Contraintes temporelles spécifiées (explicites ou non) :**
  - **Rythmes: Tstart = T0, T1, T2...**
  - **Intervalles temporelles de validité pour les activités élémentaires : Tstart < AE < Tend**
  - **Dépendences entre les activitées (relations d'ordre, sections critiques)**
- **Coordination de toutes les activités :**
  - **Cohérence temporelle : synchronisations sur le temps physique (temps réel);**
  - **Branchements conditionnels : synchronisations « logiques »**
- **Problèmes :**
  - **De quelques centaines à quelques milliers activités élémentaires à coordonner**
  - **La répartition sur une architecture multiprocesseur à mémoire distribuée induit plus de communications, donc d'autres sources de synchronisation**
- ***La coordination déterministe de toutes les activités est difficile***

Choix d'implantation

Modèle d'exécution séquentiel (découpage statique)



Modèle d'exécution parallèle asynchrone (découpage dynamique)



Contraintes très fortes sur la réalisation (réalistes ?)

Vérifications analytiques possibles ?

=> « micro-design » pour fragmenter les tâches

Le « déterminisme idéal »

- **Déterminisme logique et temporel sont indissociables**
  - **unicité et invariance du comportement dynamique**
  - **asynchronismes du système maîtrisés**
- **Impact du déterminisme**
  - **comportement dynamique du système prédictible et invariant :**
    reproductibilité du comportement dynamique
  - **tests (vérification/validation pour la sûreté) :**
    exhaustivité des comportements atteignable (recherche du pire des cas)
  - **indépendance de l'implantation**
  - **simulation exacte sans la machine cible**
- **Empêcher les propagations d'erreurs et maîtriser l'impact des anomalies**
  - **mécanismes déterministes de détection et de confinement d'anomalies**

- ***L'impact d'une anomalie de fonctionnement doit (devrait) être déterministe***

- ***Non interférence (multicoeur, bus mémoire, mémoire cache)***