3ème Partie :
Conception et réalisation

Développement de Systèmes Temps Réel

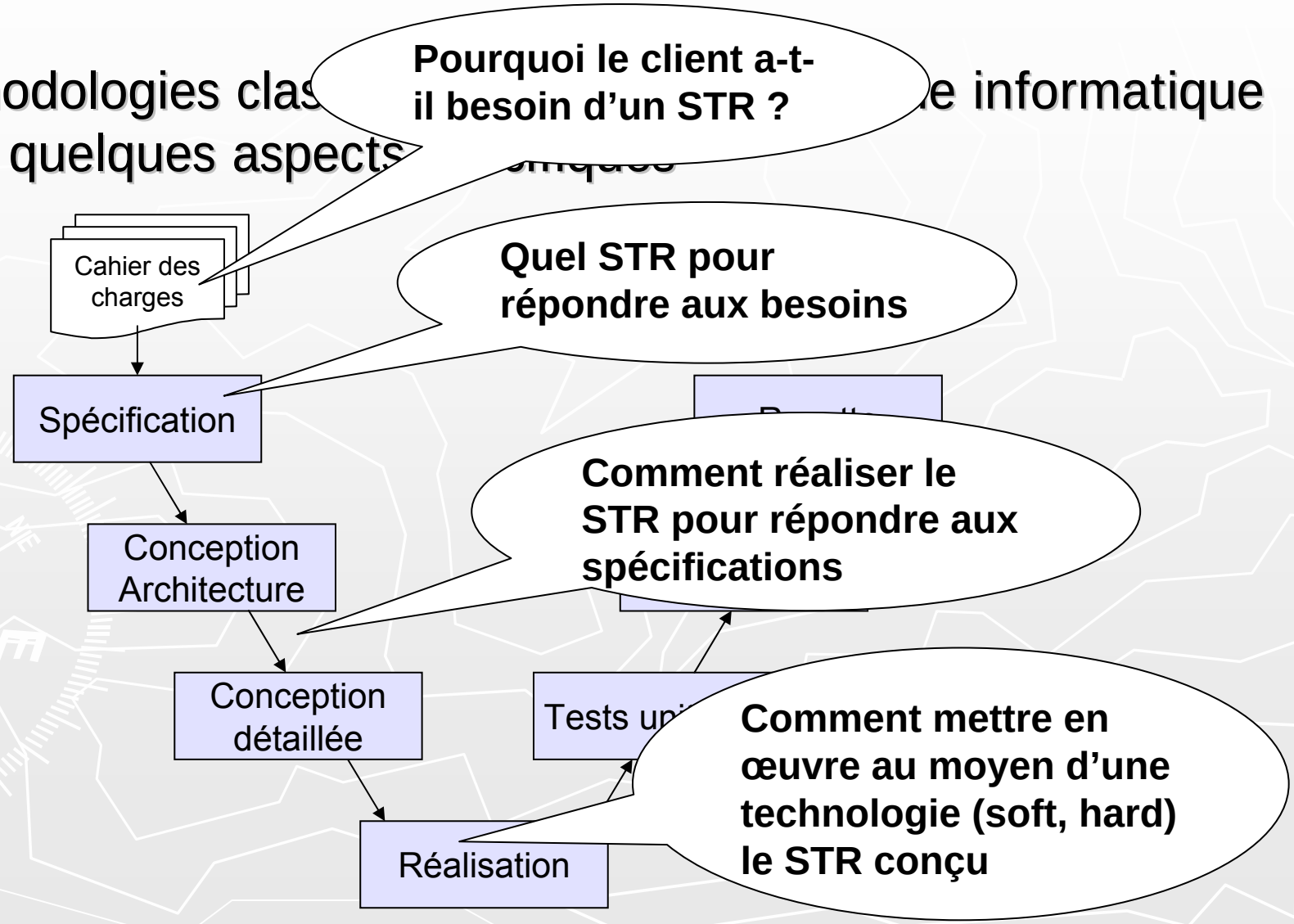
(méthode proposée pour le tp)

Plan

1. Spécification fonctionnelle avec SADT
2. Conception préliminaire
3. Approche asynchrone (préemptif)
   - Conception générale et détaillée
   - Codage
   - Tests et Validation
4. Approche synchrone (non préemptif)
   - Conception générale et détaillée
   - Codage
   - Tests et Validation

Phases d'un projet TR

- Méthodologies clas… …e informatique avec quelques aspects …



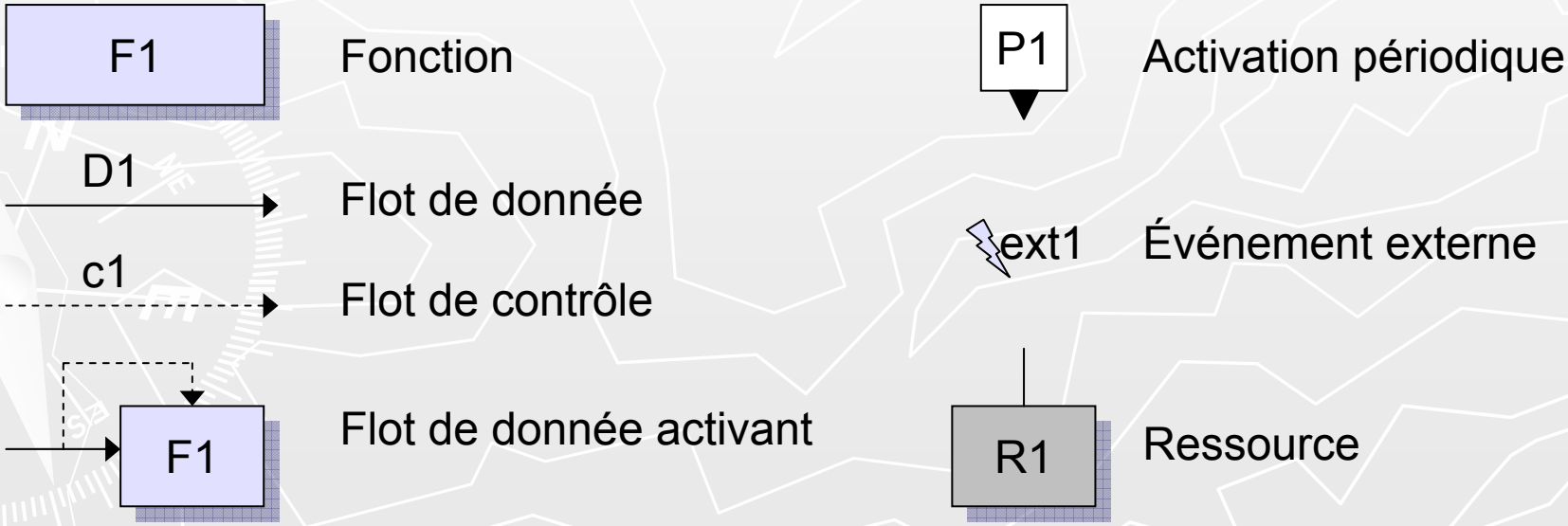
1. Spécification fonctionnelle

Spécification fonctionnelle

- A partir du cahier des charges
  - Identifier et décrire les fonctions
  - Identifier et décrire les relations entre les fonctions
  - Décrire le comportement du système
  - Décrire les contraintes temporelles d'exécution des fonctions

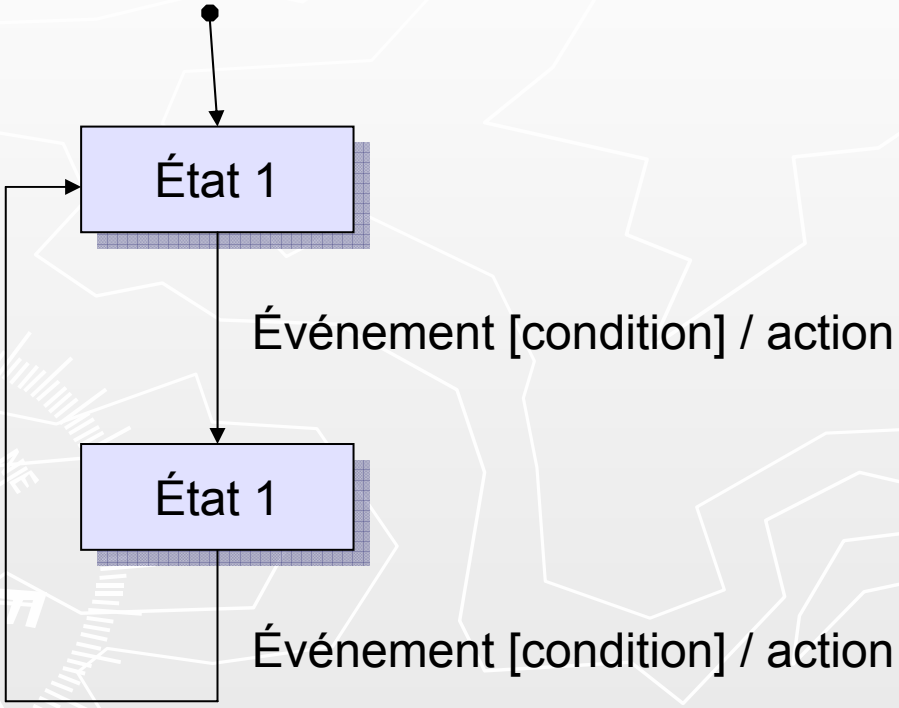
Outil de modélisation fonctionnel

- Formalisme SADT
  - Actigrammes
  - Dictionnaire des données
  - Spécification des fonctions



Spécification des fonctions (comportement)

- Diagrammes états / transitions

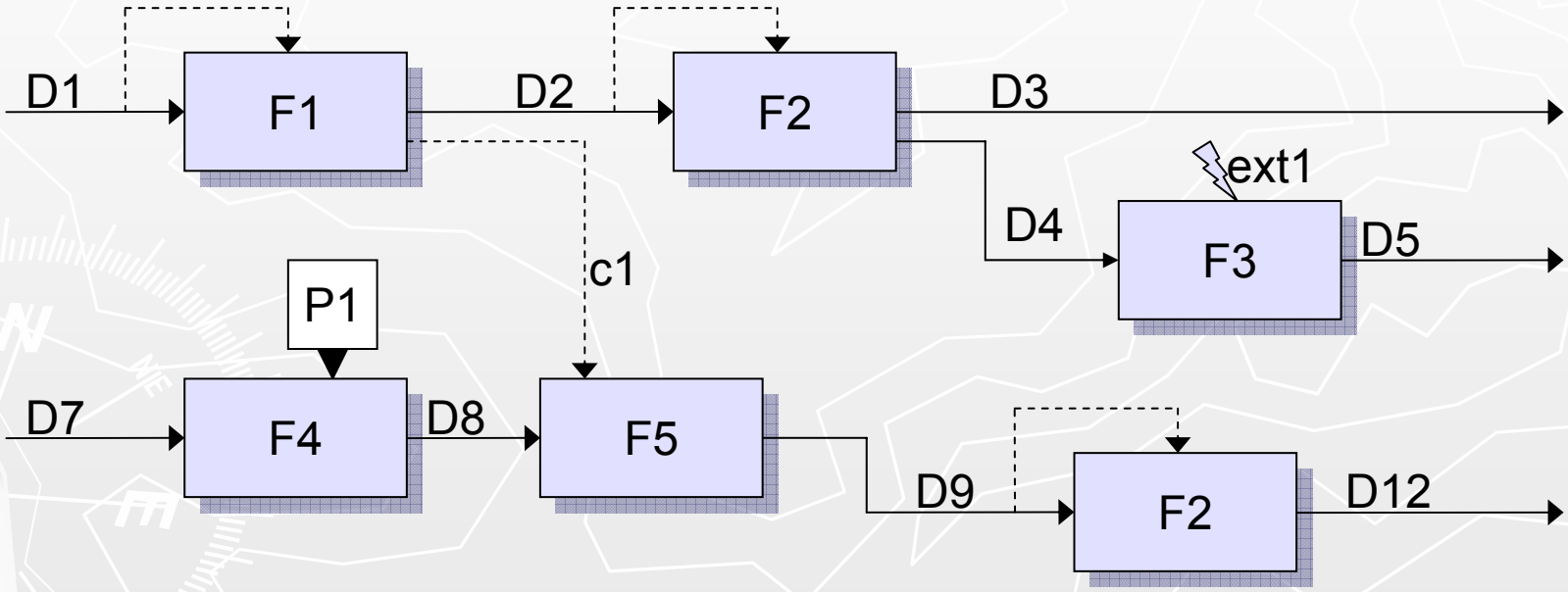


- Contraintes temporelles

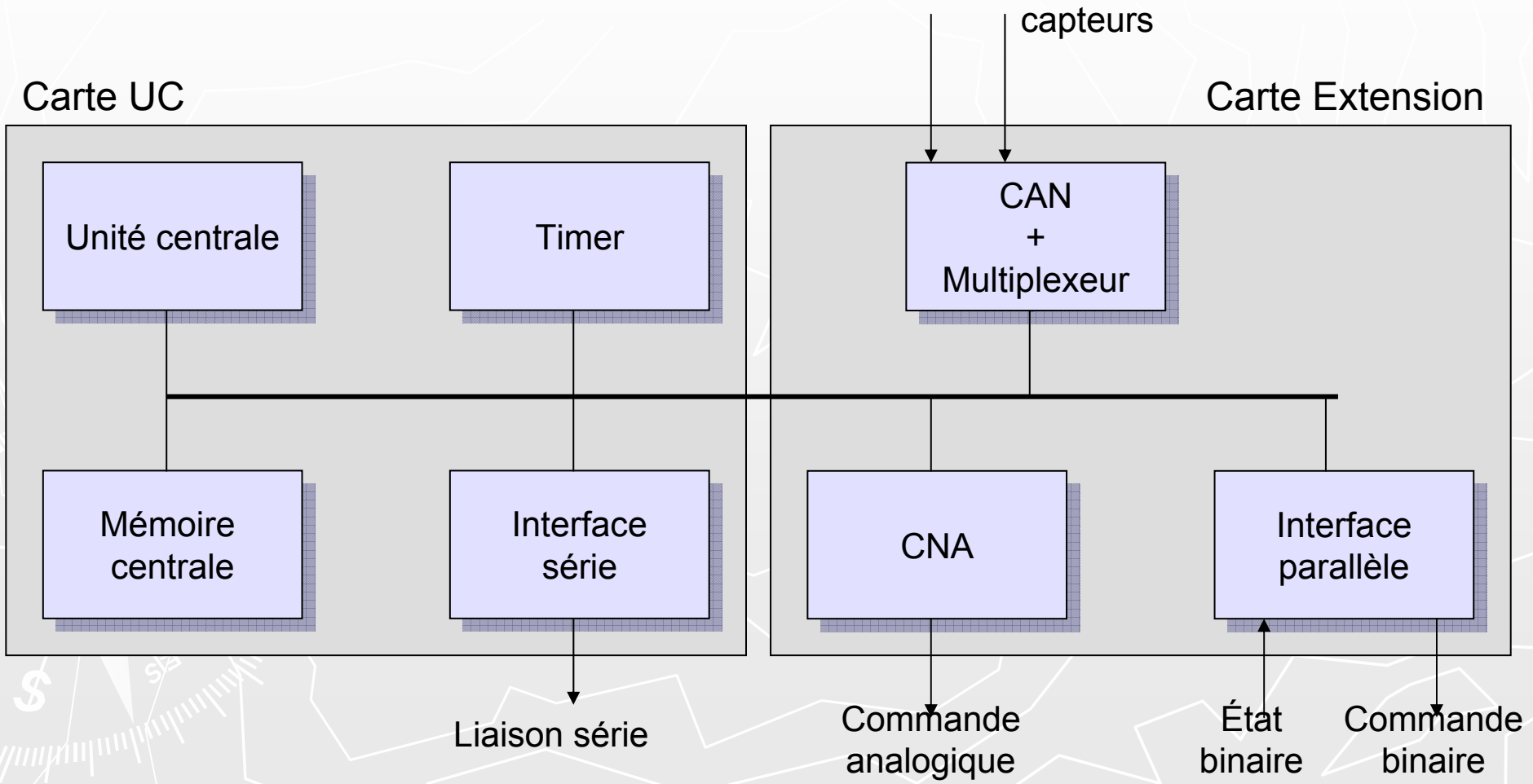
2. Préparation de la phase de conception

Préparation de la phase de conception

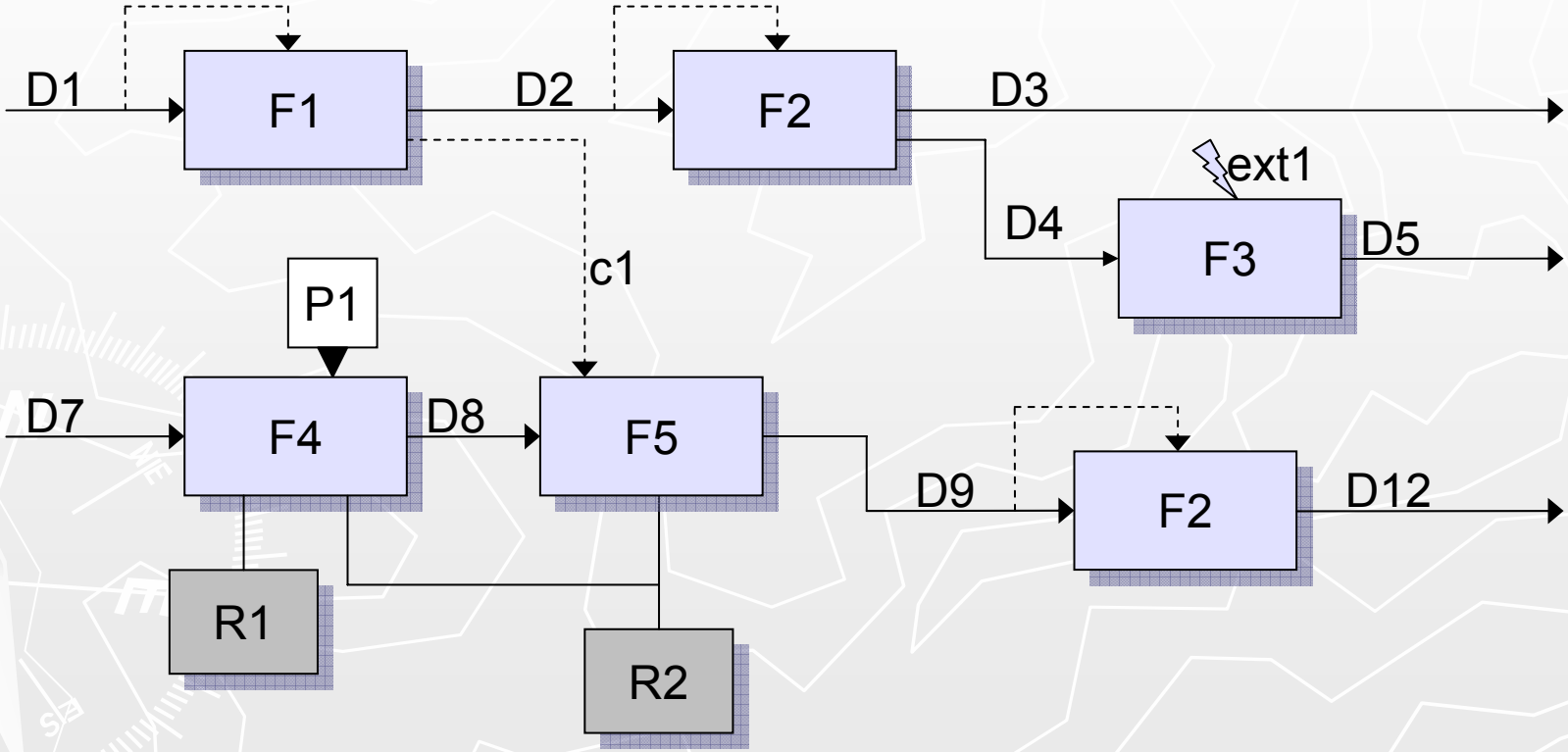
- Synthèse des diagrammes SADT



Architecture matérielle



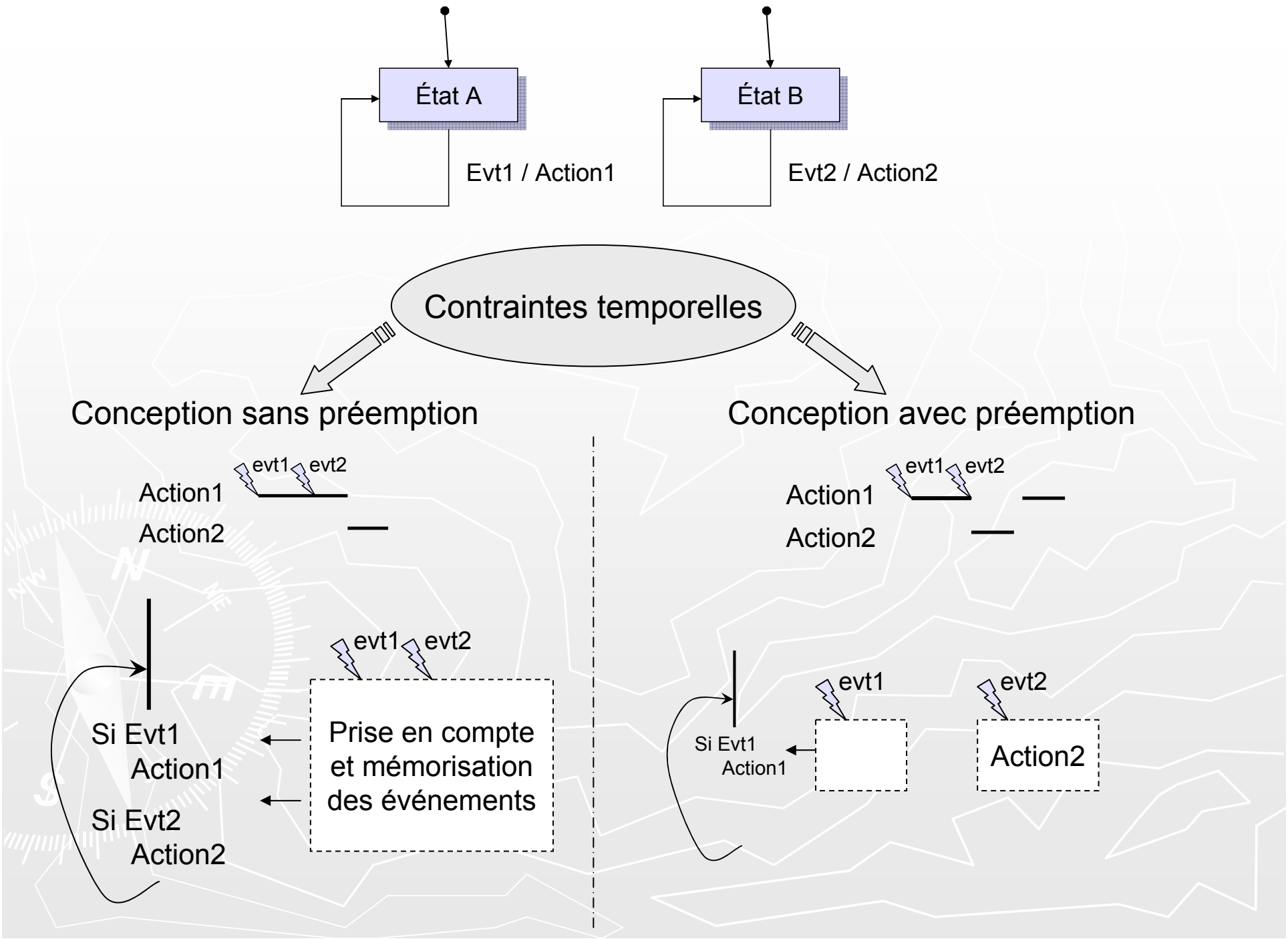
Ajout des ressources matérielles



Estimation des durées d'exécution

Fonction	Activation	Durée estimée	Deadline

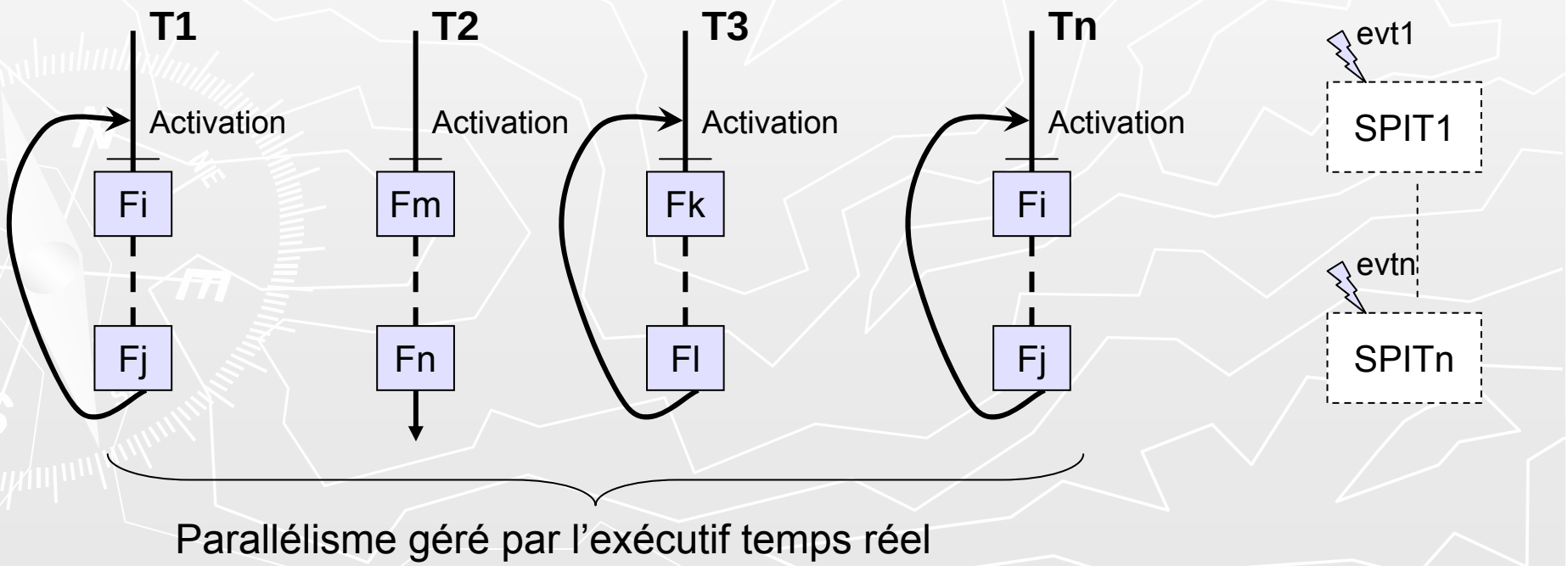
- L'analyse de ces données permet de choisir une solution avec ou sans préemption.
- Si la durée d'exécution d'une fonction est supérieure à un délai de forclusion, une solution autorisant la préemption doit être utilisée
- Si le cumul des durées d'exécution des fonctions concurrentes est inférieur au délai de forclusion le plus court, on peut aisément mettre en œuvre une solution sans préemption


État A
Evt1 / Action1
État B
Evt2 / Action2
Contraintes temporelles
Conception sans préemption
Conception avec préemption
evt1
evt2
Action1
Action2
Si Evt1
Action1
Si Evt2
Action2
Prise en compte
et mémorisation
des événements
Si Evt1
Action1
Action2

3. Approche asynchrone (préemptif)

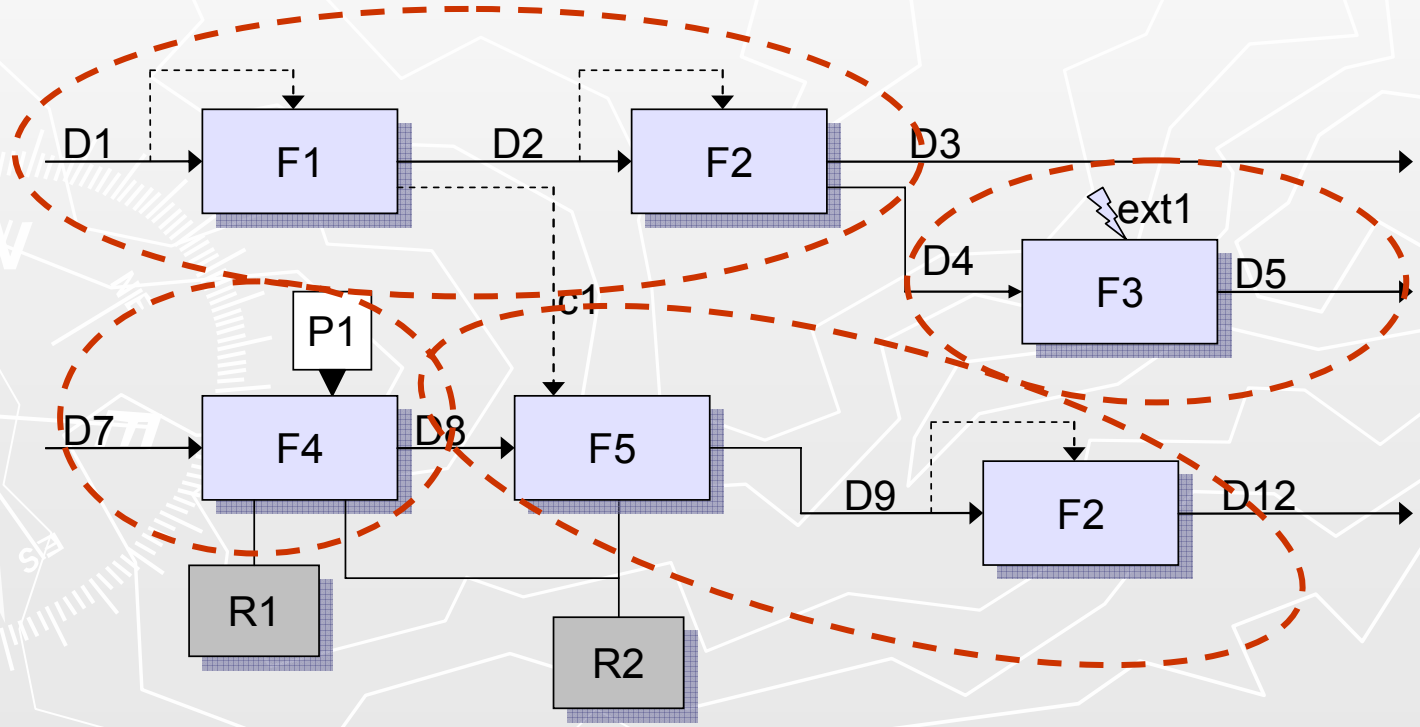
3.1 Conception générale du logiciel

- Définir l'architecture du logiciel permettant aux différentes fonctions de s'exécuter en respectant les contraintes temps réel
- Modèle d'exécution



3.1 Conception générale du logiciel

- Définition des tâches
  - Répartir les différentes fonctions dans des tâches
  - Critère à prendre en compte : indépendance des conditions d'activation



3.1 Conception générale du logiciel

- Définition des mécanismes de partage des ressources critiques
  - Identifier les ressources critiques partagées entre plusieurs tâches
  - Choisir le mécanisme permettant un accès en exclusion mutuelle
- Deux outils principaux
  - Sémaphore d'exclusion mutuelle
    - Bloquant, problème d'inversion de priorités des tâches
  - Tâche de gestion de la ressource
    - Non bloquant

3.1 Conception générale du logiciel

- Définition des mécanismes de communication entre tâches
  - Décrire dans un tableau les besoins liés à la communication entre tâches
    - Nom du lien
    - Besoin de synchro
    - Besoin de tampon
    - L/E asynchrone (donnée > 1 octet)
  - Choisir l'outil le mieux adapté
    - variable partagée, file de messages, sémaphore, ...

3.1 Conception générale du logiciel

- Définition des mécanismes de communication entre tâches

Mécanisme	Symbole
variable partagée seule	
variable partagée protégée par un mutex	
sémaphore binaire	
file de messages non activante de taille n (protégée contre les accès simultanés)	
file de messages activante de taille n (protégée contre les accès simultanés)	

3.1 Conception générale du logiciel

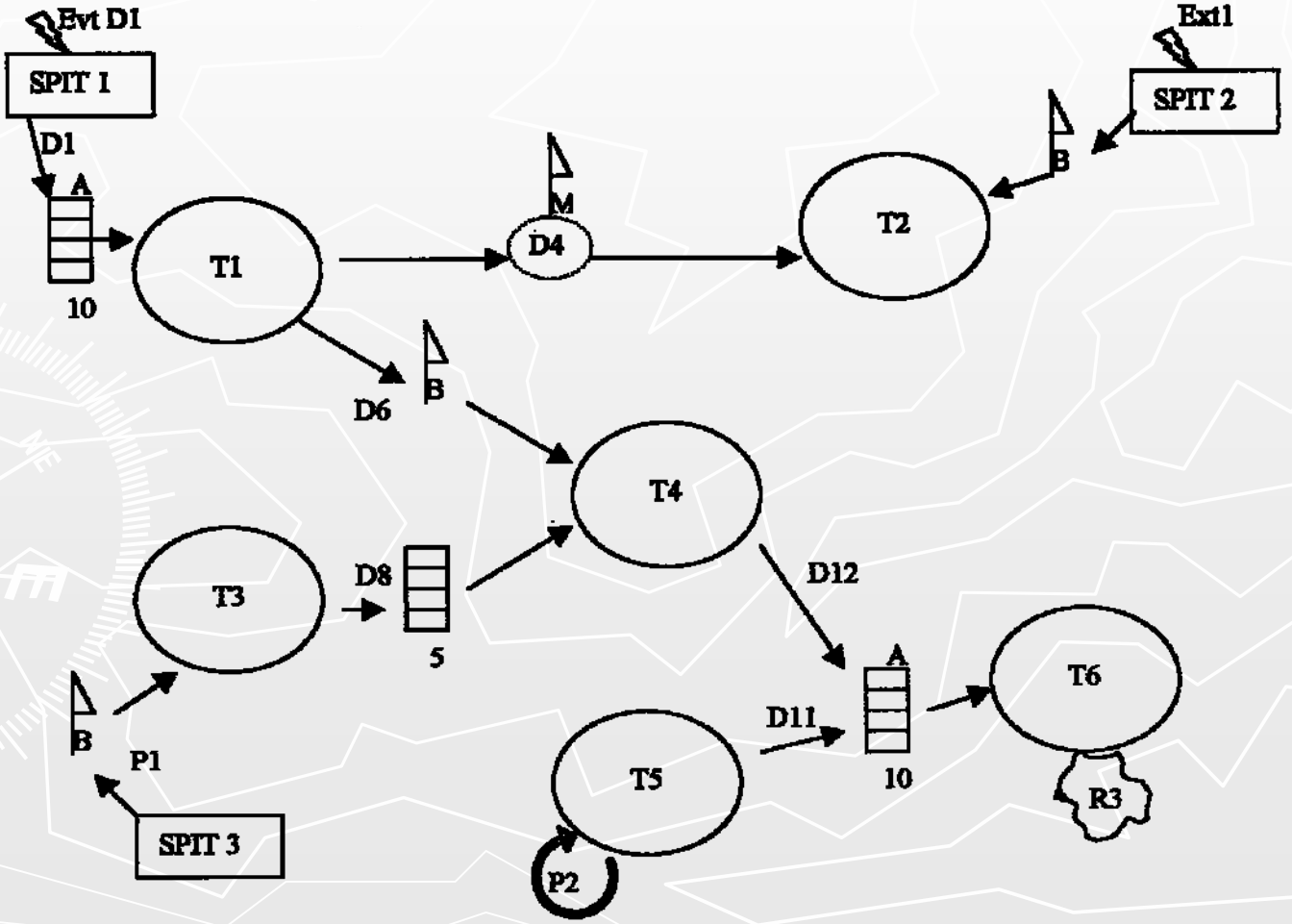
- Définition des mécanismes de synchronisation entre tâches
  - Pour implémenter une synchronisation, non liée à une communication, le concepteur a le choix entre :
    - Le sémaphore binaire, dans le cas où il est nécessaire d'attendre l'occurrence de l'évènement,
    - Un simple flag (variable booleenne partagée), dans le cas où l'occurrence de l'évènement est détectée sans mise en attente.

3.1 Conception générale du logiciel

- Définition des mécanismes d'activation périodique
  - Selon la précision souhaitée au niveau de la période
    - Attente de délai (primitive taskDelay)
    - Watchdog
    - Mise en œuvre d'une horloge externe générant un signal d'interruption périodique. Le sous programme d'interruption associé peut alors libérer un sémaphore binaire pour la synchronisation.
- Définition de la prise en compte d'évènements externes
  - Mécanisme d'interruption
- Échanges d'informations avec l'environnement
  - Utilisation des drivers VxWorks

3.1 Conception générale du logiciel

- Diagramme de synthèse d'architecture



3.1 Conception générale du logiciel

- Détermination des priorités relatives des tâches
  - Les priorités d'exécution des tâches seront calculées par application de la méthode du Rate Monotonic ou Deadline Monotonic
  - Vérifier la condition d'application de cet algorithme.

3.2 Conception détaillée

- Algorithmes de haut niveau des tâches
  - Doivent faire apparaître :
    - Activation des fonctions
    - Mise en œuvre des éléments de communication et de synchronisation

```
Tache T1

  Début
    Répéter indéfiniment
      Attendre message dans file
      Fonction F1
      V(sémaphore)
      Fonction F2
      P(mutex)
      D4 ← nouvelle valeur
      V(mutex)
    Fin répéter
  Fin
```

- 3.3 Codage de l’application (recommandations)
  - Elaborer une stratégie de développement et d’intégration, identifier les différentes versions
  - Utiliser des tâches de lancement et de terminaison de l’application
  - Tester tous les codes d’erreur
- 3.4 Tests et validation
  - Vérification fonctionnelle
  - Vérification temporelle (WindView)