A propos de l'applicabilité de la théorie de l'ordonnancement temps réel : le projet Cheddar

F. Singhoff, A. Plantec

Laboratoire Informatique des Systèmes Complexes

LISyC/EA 3883

Université de Brest

Sommaire

- Introduction et motivations.
- Une (mini) taxinomie exécutable.
- Ordonnancement temps réel et langages d'architecture : un exemple avec AADL.
- Quand les tests de faisabilité n'existent pas.
- Conclusions et perspectives.

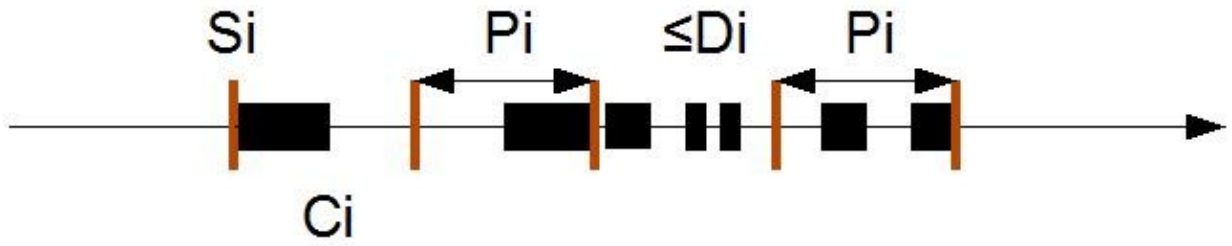
Introduction et motivations (1/5)

- « En informatique temps réel, le comportement correct d'un système dépend, non seulement des **résultats logiques des traitements**, mais aussi du **temps auquel les résultats sont produits** » (J. Stankovic, 1988).

- **Exigences de :**
  - **Déterminisme logique :** le mêmes entrées appliquées au système produisent les mêmes résultats.
  - **Déterminisme temporel :** respect des contraintes temporelles (ex : échéance).

- **Prédictibilité temporelle pour les systèmes concurrents temps réel :** théorie de l'ordonnancement temps réel.

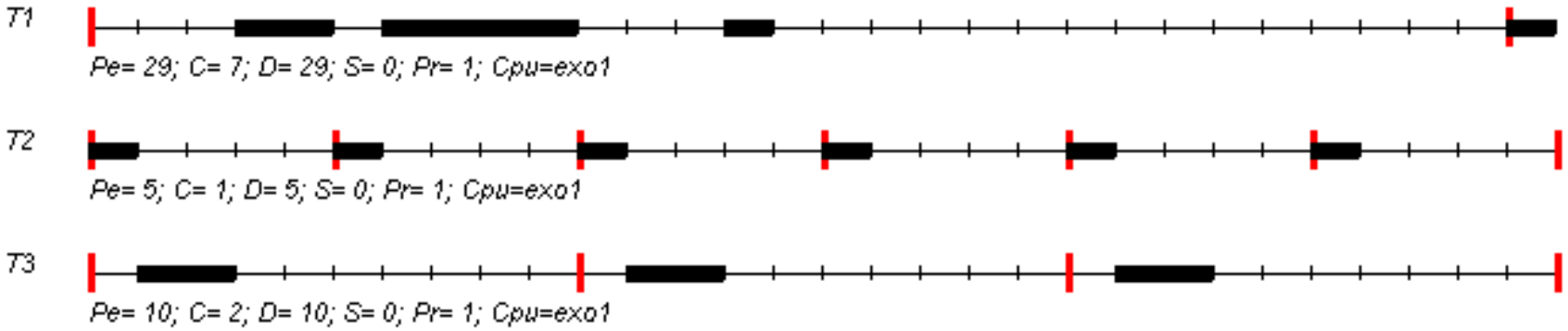
Introduction et motivations (2/5)



- **Tâches périodiques:** (Liu & Layland, 1974)
  - Borne maximale sur le temps d'exécution (capacité) : C_i
  - Délai entre deux activations (période) : P_i
  - Contrainte temporelle à respecter (délai critique) : D_i
- **Ordonnanceurs classiques :** Rate Monotonic, Earliest Deadline First, …
- **Prédictibilité, soit par simulation, soit par méthodes analytiques** (tests de faisabilité).

Introduction et motivations (3/5)

- **Simulation** : calcul GANNT + analyse. Preuve si période d'étude (à de rares occasions …)



- **Méthodes analytiques/tests de faisabilité :** limitées à certains cas

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil . C_j$$

Introduction et motivations (4/5)

- **Bilan sur la théorie de l'ordonnancement temps réel :**

  1. Résultats théoriques importants dans le cas mono-processeur de 1974 à 1994.
  2. Maturité technologique (systèmes d'exploitation, profil Ravenscar/Ada 2005, …).
  3. Forte demande du milieu industriel.

  …

  Et pourtant, pas ou peu d'utilisations !

Introduction et motivations (5/5)

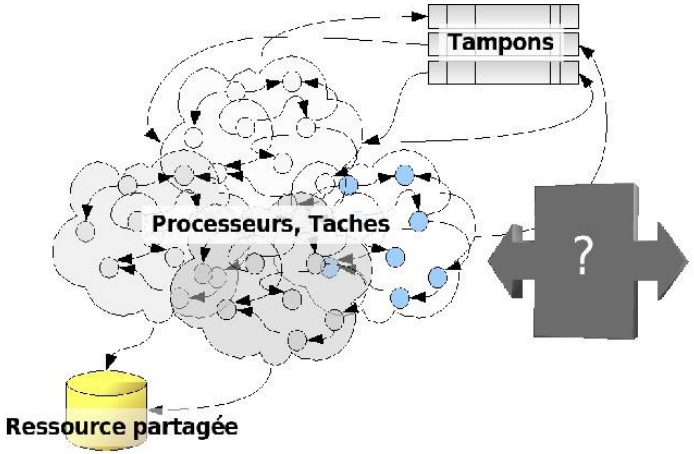
- **Quelques explications (possibles) :**

1. Explications ni scientifiques, ni techniques.
2. Nécessite un niveau d'expertise élevé pour l'employer:
   - Formation des ingénieurs.
   - Beaucoup de résultats théoriques. Synthèse délicate.
   - Langage de conception/architecture: centraliser les données.
   - Nécessité d'automatiser par l'outil/la méthode … or, peu d'outils et peu d'intégrations dans les processus d'ingénierie.
3. La théorie reste incomplète (ex : mémoire, systèmes répartis).
4. Et quand les tests de faisabilité n'existent pas ?
5. …

Sommaire

- Introduction et motivations
- Une (mini) taxinomie exécutable.
- Ordonnancement temps réel et langages d'architecture : un exemple avec AADL.
- Quand les tests de faisabilité n'existent pas.
- Conclusions et perspectives.

Un outil pour automatiser l'analyse



Quels modèles ?
Quels critères ?
Quels tests de faisabilité (hypothèses, exactitude, complexité) ?

1. **Analyse automatisée tout en maintenant la tracabilité :**
   - Sélectionner les "bons" tests (généraux, complexité, exactitude).
   - Vérifier les hypothèses des tests automatiquement.
   - Présenter les critères pertinents.
2. **Outil pédagogique, plusieurs niveaux d'accès.**
3. **Interopérabilité et ouverture des outils. Doit s'insérer dans une démarche d'analyse (processus d'ingénierie).**

Sommaire

- Introduction et motivations
- Une (mini) taxinomie exécutable.
- Ordonnancement temps réel et langages d'architecture : un exemple avec AADL.
- Quand les tests de faisabilité n'existent pas.
- Conclusions et perspectives.

AADL et analyse de performance (1/7)

- **Architecture Analysis and Design Language (AADL) :**
  - Langage permettant la modélisation et l'analyse de systèmes temps réel (architecture, matériel et logiciel).
  - Standard SAE publié en novembre 2004 (document AS 5506).
- **Un modèle AADL, c'est un ensemble de composants :**
  - Thread : flot de contrôle qui exécute un programme (ex: thread POSIX).
  - Data : n'importe quelle structure de données d'un programme (ex: struct C).
  - Processeurs, bus, périphériques, mémoires : architecture matérielle.
- **Un modèle AADL, c'est aussi :**
  - Connexions entre composants : relations entre composants.
  - Propriétés de composants : implémentation, besoin en ressources, comportement ...

AADL et analyse de performance (2/7)

Un modèle AADL, pourquoi faire :

1. Générer le code et documentation conformes au modèle (ex: Ocarina de l'ENST, STOOD d'Ellidiss Technologies).
2. Analyse : vérification sémantique, fiabilité, performance, …

Analyse de performances :

1. Un modèle AADL est il compatible avec les méthodes d'analyse de la théorie de l'ordonnancement temps réel ?
2. Dimensionnement simultané des ressources : processeurs + empreinte mémoire + communications (applications réparties).

AADL et analyse de performance (3/7)

- **Analyse de l'ordonnancement des threads. Propriétés principales présentes, mais :**

  1. Manque les propriétés classiques liées aux ordonnanceurs usuels (ex : préemptif, POSIX 1003.1b).
  2. Manque certaines propriétés classiques des threads nécessaires à l'analyse (ex : priorité, gigue).
  3. Quid de l'accès aux ressources partagées => comportement des threads AADL ?
  4. Certaines ambiguïtés existent (ex : contraintes de précédences).

- Proposition d'un jeu de propriétés supplémentaires … à intégrer à la nouvelle version du standard AADL ?

AADL et analyse de performance (4/7)

- **Exemple :** threads périodiques + POSIX 1003.1b

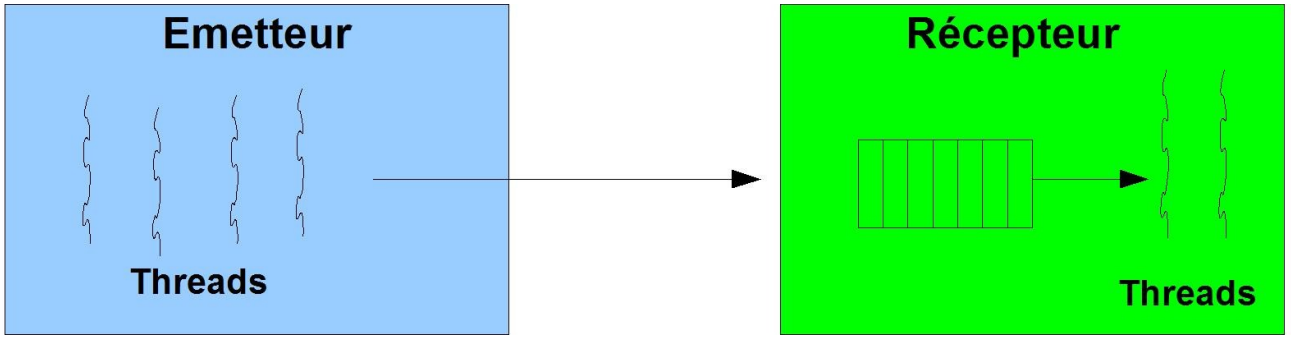
```
thread implementation T3.i
      properties
            Source_Text => “mes_threads.c”;
            Dispatch_Protocol => Periodic;
            Compute_Execution_time =>  1 ms ..  2 ms;
            Deadline  =>  10 ms;
            Period =>  10 ms;
end T3.i;
thread implementation fifo2.i
      properties
           Dispatch_Protocol => Background;
           Compute_Execution_time => 1 ms ..  3 ms;
           Cheddar_Properties::POSIX_Scheduling_Policy =>
                      SCHED_FIFO;
           Cheddar_Properties::Fixed_Priority =>  5;
           Cheddar_Properties::Dispatch_Absolute_Time => 4 ms;
end fifo2.i;
```

```
process implementation proc0.i
      subcomponents
            a_T3 : thread T3.i;
            ....
processor implementation rma_cpu.i
      properties
            Scheduling_Protocol => RATE_MONOTONIC;
            Cheddar_Properties::Preemptive_Scheduler => true;
            Cheddar_Properties::Scheduler_Quantum =>  3 ms;

end rma_cpu.i;
system implementation a_system.Impl
   subcomponents
      a_cpu : processor rma_cpu.i:
      an_application : process  proc0.i;
   properties
```

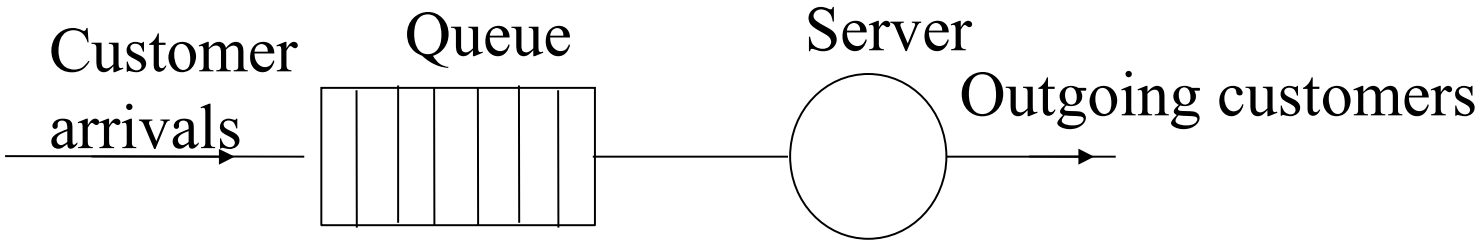
…

AADL et analyse de performance (5/7)



- Les event data ports AADL modélisent l'échange de messages entre des threads. Les événements peuvent être mémorisés dans un tampon avant consommation.
- **Applications réparties :** gestion conjointe processeurs/mémoires/...
- **Dimensionnement par théorie des files d'attente mais:**
  1. Threads AADL périodiques/sporadiques ?
  2. Ordonnancement temps réel des threads ?

AADL et analyse de performance (6/7)



- **Théorie des files d'attente :** λ/μ/n
  - λ : taux d'arrivée des clients (M,G,D).
  - μ : taux de service (M,G,D).
  - n : nombre de serveurs.
  - Exemples : M/M/1, M/D/1, M/G/1, …
- **Objectif :** calculer le temps d'attente, et le nombre de client en attente.
- **Les clients et le service peuvent être périodiques et ordonnancés avec Rate Monotonic (thèse de J. Legrand) :**
  - Définir des nouvelles lois d'arrivée/service : P.
  - Files d'attente P/P/1 et M/P/1.

AADL et analyse de performance (7/7)

- **Exemple d'un test de faisabilité sur les tampons (file d'attente P/P/1) :**

  Nombre de messages dans un tampon partagé par n tâches périodiques dont les échéances sont inférieures aux périodes :

  1. L=2.n (tâches harmoniques)
  - L=2.n+1 (sinon)

Sommaire

- Introduction et motivations
- Une (mini) taxinomie exécutable.
- Ordonnancement temps réel et langages d'architecture : un exemple avec AADL.
- Quand les tests de faisabilité n'existent pas.
- Conclusions et perspectives.

Modéliser un ordonnanceur (1/5)

- **Modéliser, puis simuler. Période d'étude ?**
- **Modéliser un ordonnanceur temps réel, c'est:**

1. Modéliser des instructions arithmétiques et logiques

   (ex: comment calculer une priorité, comment appliquer une règle de tri).

2. Modéliser des synchronisations entre tâches/ordonnanceurs

   (ex; quand une tâche est elle réveillée par un ordonnanceur, comment les ordonnanceurs doivent se coordonner, …).

Modéliser un ordonnanceur (2/5)

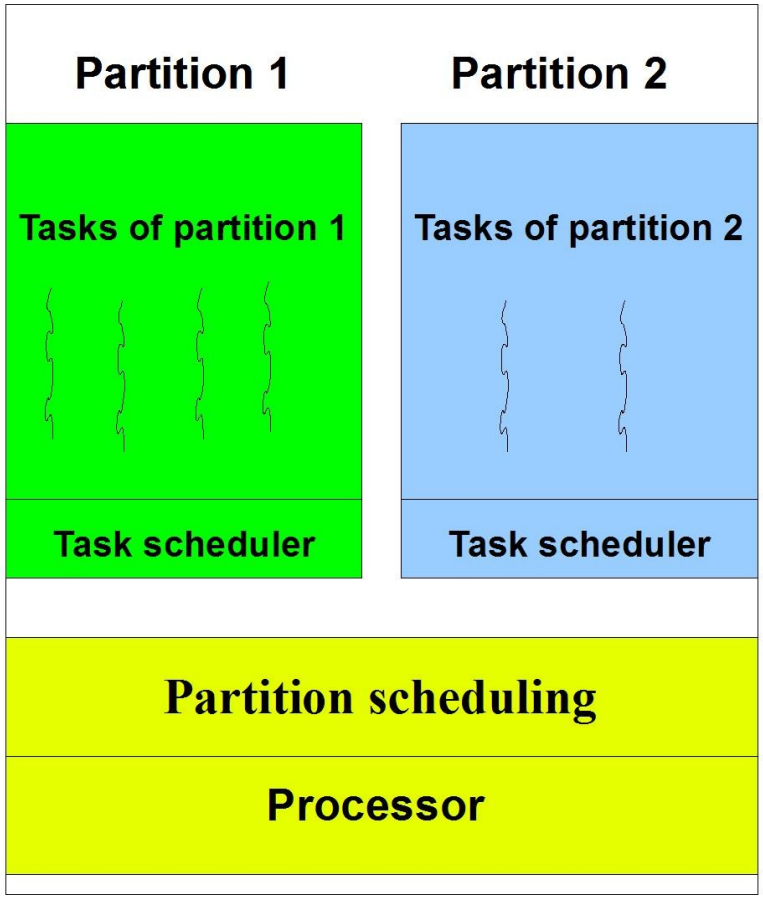
- **Le langage Cheddar c'est:**

1. **Un sous-ensemble d'Ada pour les instructions arithmétiques et logiques:**
   - Programme Cheddar = ensemble de sous-programmes (sections).
   - Les sous-programmes sont typés :
     - Start sections : déclarations/initialisation de variables.
     - Priority sections : calcul sur données de simulation (ex : priorités).
     - Election sections : sélection d'un tâche à exécuter (règles de tri).

- **Automates temporisés pour la modélisation des synchronisations:**
  - Réseaux d'automates UPPAAL.
  - Etats et transitions. Une transition peut comporter une opération sur horloges, une garde et/ou finalement une synchronisation (qui peut être une section Cheddar).

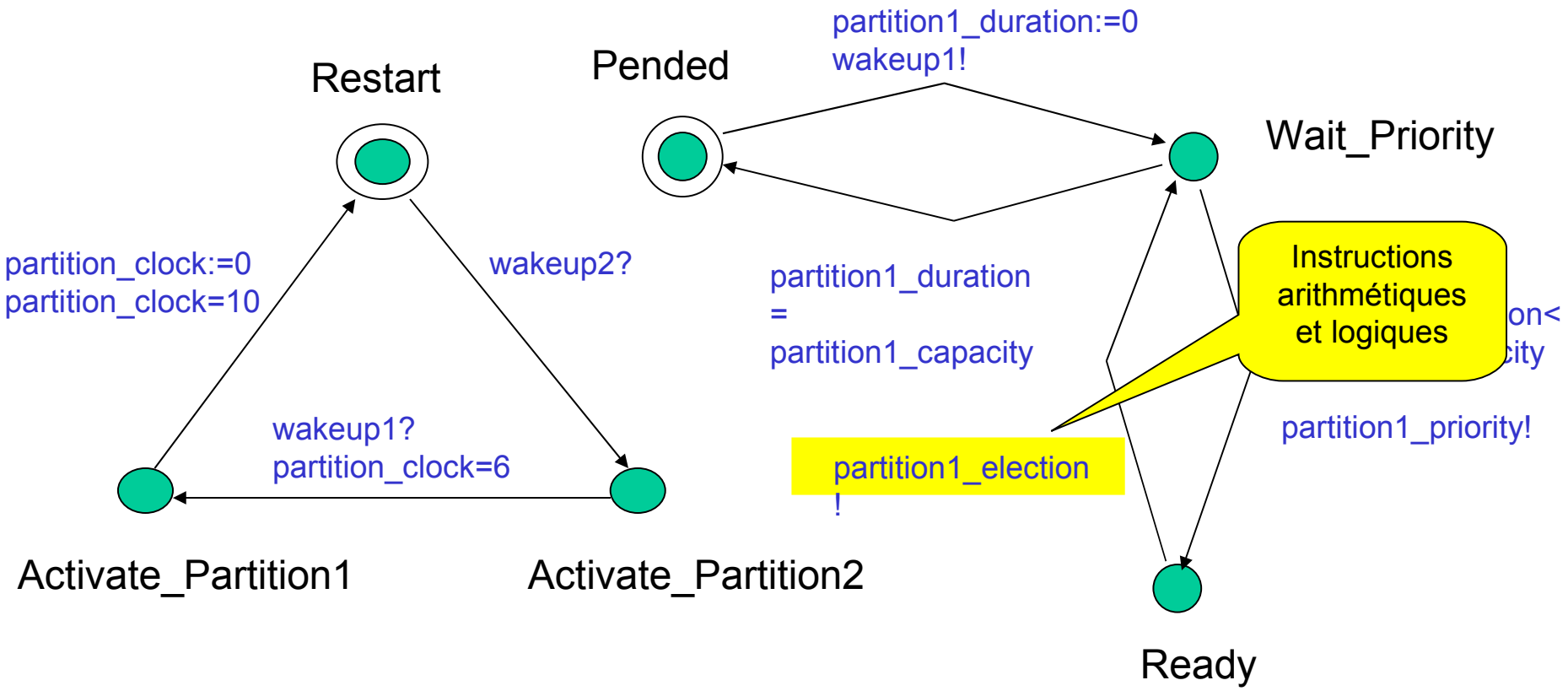
Modéliser un ordonnanceur (3/5)

- **Partition =** application + isolation mémoire et temporelle.
- **Ordonnancement hiérarchique d'ARINC 653:**
  1. Calcul de l'ordonnancement des partitions (hors-ligne).
  2. Ordonnancement des tâches de chaque partition (en-ligne, ordonnancement à priorité fixe).

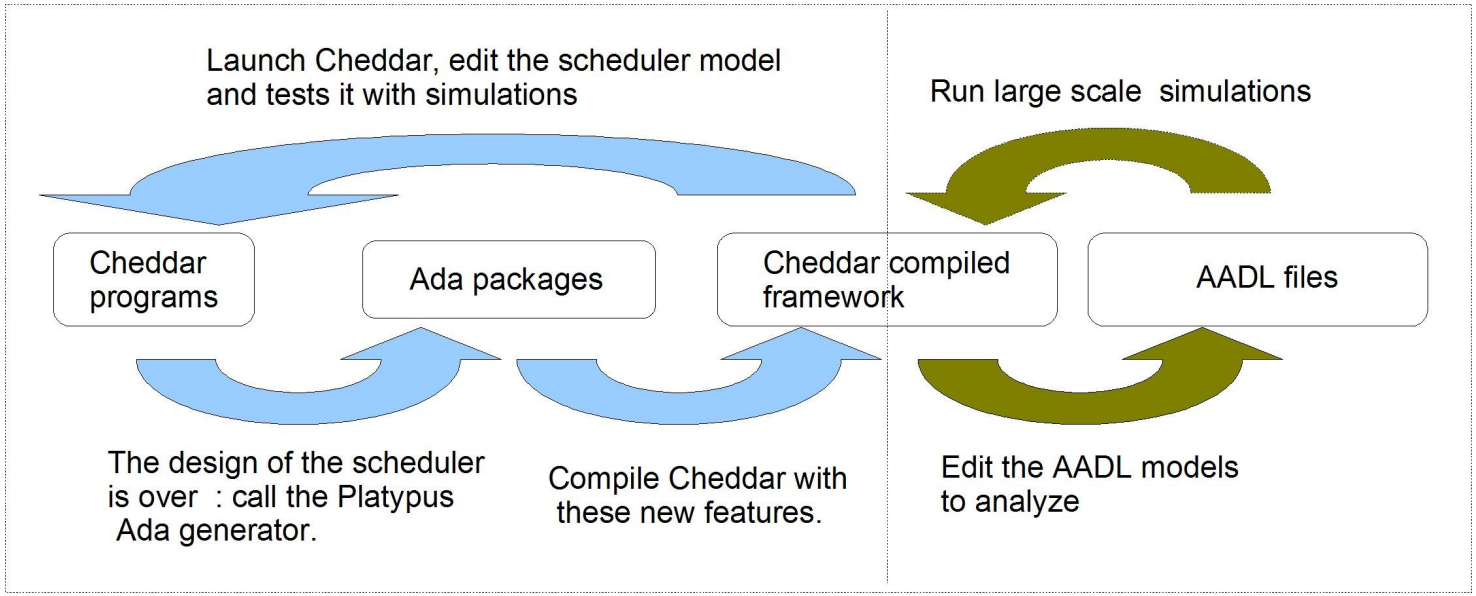


Modéliser un ordonnanceur (4/5)

- **Programme Cheddar modélisant un ordonnanceur hiérarchique :**



Modéliser un ordonnanceur (5/5)



- **Processus d'ingénierie d'un ordonnanceur:**
  1. Conception et test du modèle d'ordonnanceur par un ensemble de programmes Cheddar.
  2. Génération du simulateur grâce au Meta-CASE tool Platypus, puis simulations à large échelle.

Sommaire

- Introduction et motivations
- Une (mini) taxinomie exécutable.
- Ordonnancement temps réel et langages d'architecture : un exemple avec AADL.
- Quand les tests de faisabilité n'existent pas.
- Conclusions et perspectives.

Conclusion et perspectives (1/2)

- **Quelques contributions de Cheddar, “le projet” :**
  - Outil pédagogique (ENST, Univ. Rhode Island, Monash Univ., Centro Universitario de Mérida, Univ Politécnica de Catalunva, …).
  - Outil d’expérimentation (ENST, Ellidiss, Thalès, Airbus, IRIT, …).
  - Intégration CASE tools (STOOD, PPOOA, …).
  - Quelques contributions scientifiques :
    - Synthèse de la théorie de l’ordonnancement temps réel.
    - Théorie des files d’attente et temps réel.
    - Contributions à la validation du standard AADL.

- **Cheddar, “l’outil” :**
  - Distribué sous GPL. Dernière version : février 2007.
  - http://beru.univ-brest.fr/~singhoff/cheddar

Conclusion et perspectives (2/2)

- **Travaux en cours et perspectives :**
  1. Langage d'architecture AADL :
     - Quels sont les bons patrons de conception qui autorisent l'analyse avec la théorie de l'ordonnancement temps réel (Ellidiss technologies).
     - Analyse performances de modèles AADL 2.0 ? (annexe comportementale).
  2. Langage de modélisation Cheddar :
     - Analyse performance "large échelle".
     - De la simulation à la preuve.
     - Comment comparer les modèles d'ordonnanceur ?

- **Partenariats :**
  - ENST Paris (L. Pautet, J. Hugues, F. Kordon) : Ocarina/PolyOrb.
  - Ellidiss technologies (P. Dissaux) : intégration STOOD, patron de conception AADL.