

Journées IDM / ADAPT, Brest, 13 Novembre 2008

Des modèles d'architecture
pour l'adaptation
des systèmes embarqués temp-réel

jean-philippe.babau@univ-brest.fr

LISyC, équipe IDM



plus d'information sur www.lisyc.univ-brest.fr/pages_perso/babau/

Plan

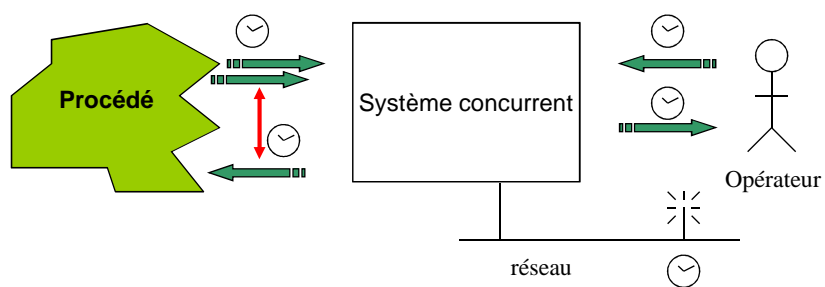
- Introduction
 - Développement des systèmes embarqués
 - Notions de contexte
- **Qinna** : un style architectural pour la gestion de modes d'exécution
- **SAIA** : un style architectural pour l'intégration de plates-formes
 - plates-formes d'acquisition/restitution
- Conclusion et Bilan

Plan

- Introduction
 - Développement des systèmes embarqués
 - Notions de contexte
- Qinna : un style architectural pour la gestion de modes d'exécution
- SAIA : un style architectural pour l'intégration de plates-formes
 - plates-formes d'acquisition/restitution
- Conclusion et Bilan

3

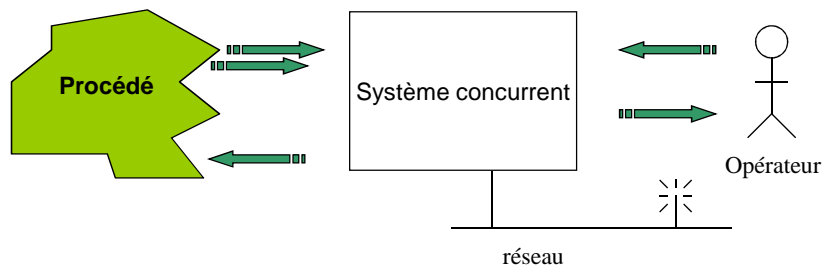
Les systèmes embarqués communicants



Contraintes temporelles
- acquisition, sorties, échéances

4

Les systèmes embarqués communicants



Contraintes temporelles

- acquisition, sorties, échéances

Contraintes d'embarquabilité

- Place limitée, contraintes physiques
- Contraintes d'énergie

5

Contraintes de coût

Les spécificités des systèmes embarqués

❑ Systèmes fortement contraints

- Contraintes temporelles
- Contraintes d'embarquabilité

❑ Sûreté de fonctionnement

- Systèmes critiques
- Systèmes enfouis

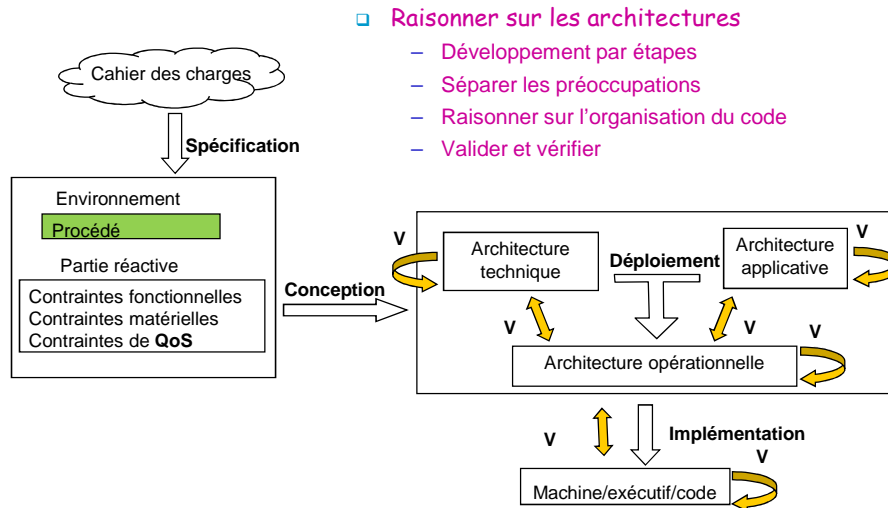
❑ Développement

- Ressources limitées et spécifiques
- Systèmes prédictibles
- Systèmes dédiés

Importance de l'implémentation pour respecter les contraintes

6

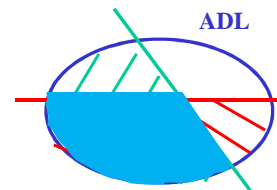
Améliorer le processus de développement



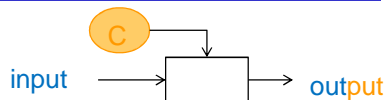
7

Raisonner au niveau architectural

- Maîtrise de la « qualité » des architectures**
 - Langages de modélisation (**ADL**, UML 2, DSL)
 - Paradigmes (Objet, **composant**)
 - **Styles architecturaux**
 - Améliorer la qualité au sens du génie logiciel
 - Permettre l'utilisation de techniques formelles
 - Types de composants, contraintes de structuration
 - Sémantique opérationnelle temporisée
 - **Model Driven Engineering**
 - Définir les concepts pour assurer la séparation des préoccupations
 - modèles et méta-modèles
 - Abstraction
 - Transformations de modèles
 - Liens entre concepts
- Assurer la correction des architectures : analyse de performances



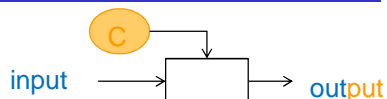
Adaptation au contexte



- **Contextes** : l'utilisateur du système, l'environnement physique, l'environnement pour l'exécution (les machines et les mediums de communication) et le temps
- **Impact du contexte** : *contexte borné*, une partie du comportement (QoS) modifié
- **Variabilité du contexte** : *état des ressources* et des *technologies*
- **Adaptation au contexte**
 - Technologies : paramètres de configuration, *plate-forme abstraite*
 - État des ressources : intergiciel dont le rôle est l'acquisition du contexte puis la modification du mode de fonctionnement : *introduction de points de variabilité*
 - Les changements de mode de fonctionnement sont « sûrs de fonctionnement »
- **Points de variabilité**
 - Lié au domaine
 - Au sein des composants : un état, une *implémentation*, des propriétés, un comportement optionnel, des connexions

9

Adaptation au contexte



- **Objectif**
 - Adaptation « QoS-Aware » au contexte
 - Contexte : ressources d'exécution et de communication
 - Introduction de points de variabilité pour réagir aux variations de l'état des ressources
 - Abstraction des communications pour intégrer la variabilité des technologies de communication
- Proposer des modèles d'architectures à composants pour une gestion sûre (QoS) et dynamique de divers contextes d'exécution

10

Plan

- Introduction
 - Développement des systèmes embarqués
 - Notions de contexte
- **Qinna** : un style architectural pour la gestion de modes d'exécution
- SAIA : un style architectural pour l'intégration de plates-formes
 - plates-formes d'acquisition/restitution
- Conclusion et Bilan

11

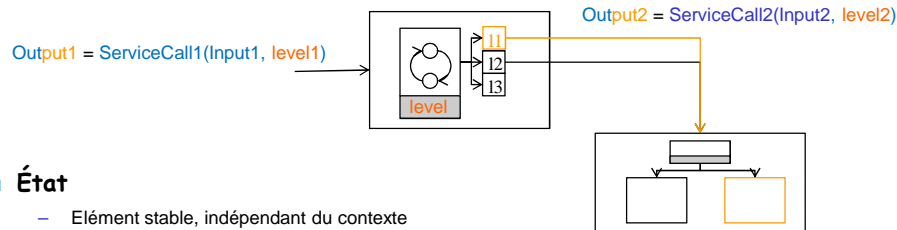
Qinna : un style architectural pour la gestion dynamique de ressources

- **Systèmes embarqués communicants**
 - Gestion sûre des ressources d'exécution
 - Adaptation (applications, ressources)
 - Ajout / retrait de composants
 - Ressources matérielles aux capacités variables
- **Modèle de composants fractal**
 - Tout est composant
 - Services fournis et requis
 - Composition : contrats
 - Signature / pré-post / protocole / **Qualité de Service**
 - Think : implémentation de Fractal pour l'embarqué
 - Gestion dynamique des composants à l'exécution



12

Introduction des modes de fonctionnement



□ État

- Élément stable, indépendant du contexte
 - état du composant vis-à-vis de son utilisation (*protocol-statechart*)

□ Connexions, code optionnel

- Architecture dont la structure est dynamique
 - Plus difficile à analyser et à maintenir
- Programmable via les composants
 - Encapsulation dans un composite

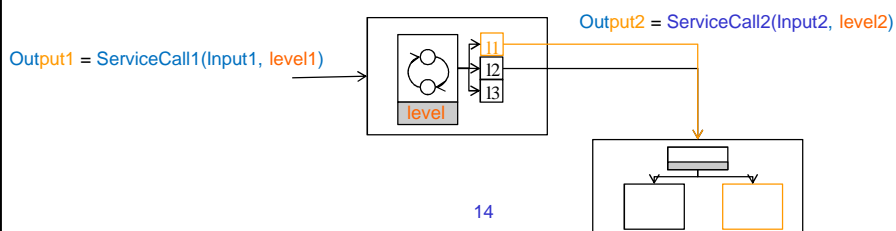
□ Comportement

- Niveau de service : Algorithme, valeur d'une propriété

13

Modes de fonctionnement

- **Modèle Fractal** : un service fourni = un comportement + un ensemble de service requis
- **Modèle Qinna** : Un comportement = une liste ordonnée, a priori, de niveaux de service
 - Ensemble de comportements
 - Appel de service : appel explicite d'un niveau de service
- **Un niveau de service offert requiert un niveau de service pour chaque service requis**
 - La relation entre niveaux (fourni/requis) est réifié dans une table de *mapping*
- **Un appel à un niveau correspond à un arbre d'appels à des niveaux donnés**
 - Pour un appel de service, l'arbre des invocations de service est toujours le même
 - seul les niveaux de service changent
 - Relation d'ordre entre les niveaux définie par la relation d'ordre des niveaux à la racine de l'arbre



14

Gestion des modes d'exécution

□ Contrat

- Demande : intervalle ordonné de niveaux de service acceptables
- Réponse : niveau maximal possible accepté ou rejet

□ Contexte : les ressources

- Un composant = une ressource
 - Contraintes de ressources pour le type de composants et pour chaque instance
 - Contraintes vérifiées à la création du type de composant, à la création de chaque instance, à chaque appel de service
- Un niveau de service
 - Une certaine utilisation (consommation) du composant
 - Une certaine consommation de la ressource (composant) utilisée

15

Quantity Of Resource (QoR) Vs Quality of Service (QoS)

□ Contraintes de QoR : niveau composant

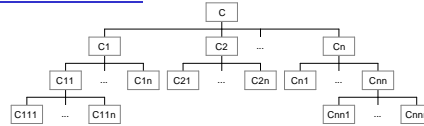
- Logique d'expression
 - Génération de contrôleurs embarquables
- Exemples
 - Type : quantité mémoire disponible, nombre maximal d'instances
 - Instance : quantité mémoire, mono-utilisateur
 - Service : quantité mémoire consommée

□ Contrat de QoS : niveau liaison

- Niveau de service et niveau d'importance
- Établissement du contrat : recherche du niveau maximal acceptable correct
 - Maximal : niveaux de service pour un appel et importance entre contrats
 - Acceptable : p parmi n exprimé lors de la demande du contrat
 - Correct : respect des contraintes d'utilisation de ressources pour chaque service invoqué
- Dégradation
 - Selon le niveau d'importance

16

Gestion dynamique de la QoS



□ Suivi des contrats

- À chaque changement
 - Demande / Arrêt d'un F-component
 - Modification : demande d'une application ou état d'une ressource
- Recherche d'un contrat acceptable
 - Algorithme décentralisée

□ Bilan

- Compromis gaspillage de QoS / adaptations
- Confiance au niveau ressource

17

Première évaluation de Qinna

□ Etudes de cas

- Ordonnancement temps réel
 - Structures hiérarchiques, régisseur
- Pic de demande, de charge
 - « saut(s) » selon la forme du pic, suivi périodique
- Variation d'une ressource
 - « Marches » prédéfinies, suivi événementiel

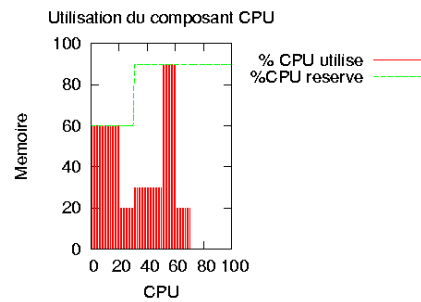
□ Utilisation

- Systèmes multimédias
 - Adaptation, coûts raisonnables
 - Application vidéo
 - 7 QoSComponent, processeur 200Mhz, 60 Mo de RAM
 - Établissement et adaptation d'un contrat : 4,38 ms et 1,08 ms
 - Place : 749 ko, 1,5 %
- Systèmes temps réel
 - Structuration et intégration des politiques de gestion de la QoS

18

Mise en œuvre et validation

- Application à un viewer d'images distantes (Projet ANR REVE)
 - Taux d'utilisation des ressources
 - Nombres d'opérations d'adaptation
- Implémentation en Fractal et Accord (projet ANR REVE)
- Perspectives
 - Génération de code optimisé
 - Modèles de transformation



19

Plan

- Introduction
 - Développement des systèmes embarqués
 - Notions de contexte
- Qinna : un style architectural pour la gestion de modes d'exécution
- SAIA : un style architectural pour l'intégration de plates-formes
 - plates-formes d'acquisition/restitution
- Conclusion et Bilan

20

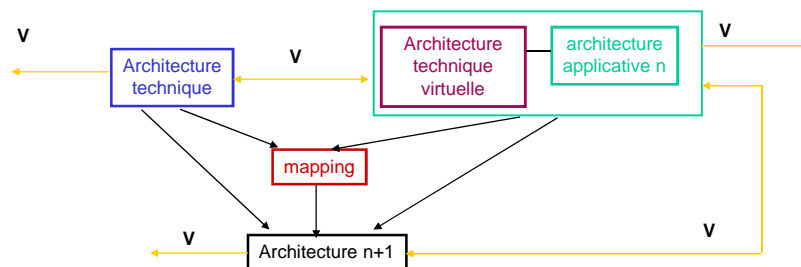
Adaptation aux technologies : les besoins

- **Domaine**
 - Systèmes embarqués temps réel de contrôle
 - Lien fort avec les capteurs/actionneurs
 - QoS (fréquence et retard)
- **Besoins**
 - Portabilité des applications
 - Raisonner au niveau des architectures applicatives
 - Evolution des applications et des plateformes
 - Dissocier les modèles d'architectures
 - Intégration d'applications et de services
 - Vérification avant déploiement

21

Phase de déploiement

- **Approche « MDA » à base de composants**
 - PIM + Plate-forme PM + mapping = PSM
 - Plateforme abstraite
 - Structuration à base de composants
 - sémantique IF, méta-modèles
 - Modélisation par étape



22

Les plateformes

- Un ensemble de ressources
 - Composant
 - Services d'accès

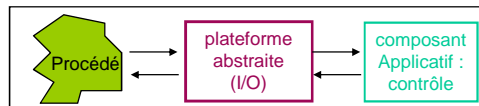
- Plateformes pour les systèmes embarqués
 - I/O (capteurs / actionneurs), IHM
 - Communication (réseau)
 - OS et HW (concurrency, mémoire, énergie)

- Importance de la QoS

23

Les Entrées / Sorties (I/O)

- Communication avec l'environnement
 - Procédé et environnement physique
 - Les I/O sont vues à un haut niveau d'abstraction
 - Plateforme abstraite



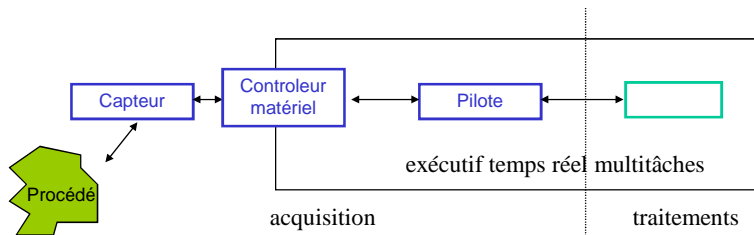
- Dérivation de contrainte
 - Quelle QoS sur les I/O permet d'assurer une certaine QoS applicative

- Simulation Matlab / Simulink
 - QoS applicative = fonction(QoS I/O, comportement de l'applicatif)

24

Système d'acquisition de données

- ❑ Evaluation des plateformes techniques
- ❑ Réutilisation de services d'acquisition
 - Pilote de communication
- ❑ Qualité de service du pilote
 - QoS fournie en terme de fréquence, pertes, retards
- ❑ Mise au point d'architectures de pilote

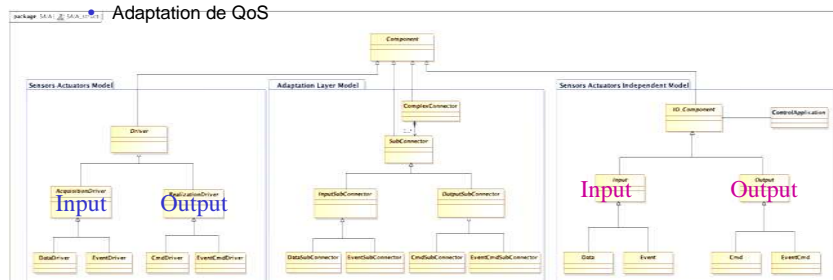


25

Architecture applicative SAIA

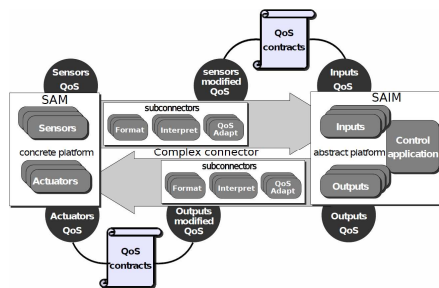
(Sensor Actuator Independent Architecture)

- ❑ Lien I/O <-> pilotes de capteurs/actionneurs
 - Indépendance vis-à-vis des mécanismes d'entrée/sortie
 - Simulation/prototypage, portage sur autre cible, modifications (E/S, fonction)
- ❑ Architecture en couches
 - SAIM : (I/O + QoS) requis pour assurer une bonne qualité de fonctionnement
 - SAM : (I/O + QoS) fournis par la plate-forme
 - ALM : Connecteur complexe
 - Ensemble de connecteurs (un par I/O abstraite)
 - formatage + **interprétation**
 - Adaptation de QoS



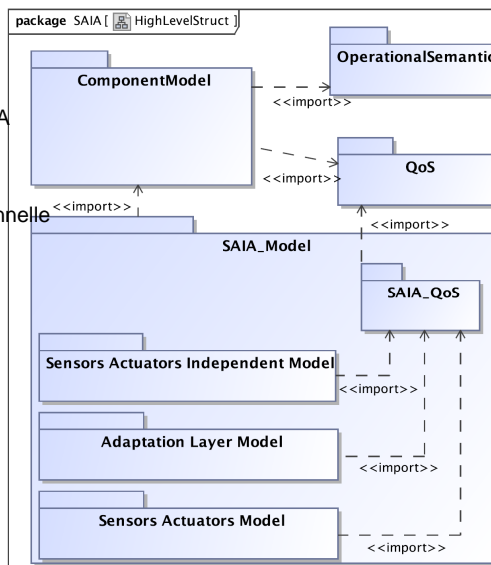
Composition correcte au niveau de la QoS

- Evaluation de la QoS : vérification de l'assemblage (SAIM + ALM + SAM)
 - Observateurs d'occurrence de flot en IF
 - Connexion : relation de satisfaction
 - QoS Fournie (ALM) « satisfait » QoS Requite (SAIM)
 - Intervalles : inclusion
 - LTS : la QoS requise *simule* la QoS fournie



Modèles pour le style architectural

- Méta-modèle
 - Un package générique
 - les concepts pour appliquer SAIA
 - Modèle de composant minimal
 - Concepts structurels
 - Modèle de sémantique opérationnelle
 - IF (temporisé)
 - Modèle de QoS
 - Flots, contrats, ...
 - Style architectural
 - Un package par couche
 - Un package pour la QoS
- Implémentations
 - GME, EMF



Modèles pour la méthode

- Mise au point de la QoS du SAIM
 - Modèles de l'automaticien
 - Non intégré et non automatisé dans l'outil
- Style architectural
 - Implémenté via l'outil d'édition
- Evaluation de la QoS
 - Transformations vers les modèles du formel (IF, CADP)

29

Architectures et IDM

- Styles architecturaux pour les Systèmes embarqués communicants
 - Plateformes abstraites avec contraintes de QoS
 - Modèle de variabilité dans le comportement
 - Lien QoR / QoS
- Séparation des préoccupations
 - Abstraction des concepts
 - Mise en œuvre
 - Mise en œuvre dans un ADL donné
 - Mise en œuvre à l'exécution
 - Liens avec la vérification formelle
- QoS
 - Sémantique de composition

30

L'adaptation et l'IDM pour les systèmes embarqués

- **Intégration d'un modèle de style architectural**
 - Intégration de contraintes dans un ADL
 - ADL minimal
 - Concepts minimaux pour appliquer le style
 - ADL existant
 - Types de composants et règles d'assemblage
 - Liens vers les techniques formelles
 - Sémantique opérationnelle temporisée
 - Modèles formels pour assurer la correction d'assemblages donnés

- **Perspectives**
 - Modèle d'un style architectural indépendant d'un langage ou d'un paradigme
 - Avec intégration de la QoS
 - Intégration de plusieurs styles architecturaux
 - Génération de code optimisée : formaliser et évaluer des modèles de génération

31

Discussion

32