ALASCA — Architecture logicielles avancées pour les systèmes cyber-physiques autonomiques

© Jacques Malenfant

Master informatique, spécialité STL – UFR 919 Ingénierie

Sorbonne Université
Jacques.Malenfant@lip6.fr

Cours 1

Introduction aux systèmes de contrôle cyber-physiques auto-adaptables

Description courte de l'UE

Étude des architectures logicielles pour les systèmes de contrôle cyber-physiques et autonomiques, de leurs concepts, théories, techniques et méthodologies. Architectures à base de composants et fondées sur les services appliquées aux systèmes cyber-physiques, à l'informatique autonomique, à la robotique autonome et aux systèmes embarqués et répartis à grande échelle.

Plan

Généralités

- UE professionnalisante et recherche STL :
  - exige un équilibre entre les aspects conceptuels et pratiques.
- 14 semaines (voir le calendrier du master, onglet M2-STL) :

  Mercredi, 13h45 – 15h45 : 2h cours magistraux.

  Mercredi, 16h00 – 18h00 : 2h de travaux pratiques/dirigés
  - présentations techniques (outils)
  - travail sur les projets
  - audits de projets
  - travail sur les sujets de recherche
- Évaluations :
  - Audit 1 (projets) : 9/10/2024
  - Soutenances mi-semestre : 13/11/2024 (rendu 10/11, minuit)
  - Audit 2 (projets) : 9/01/2025
  - Soutenances finales (projets) : 5/02/2025 (rendu 2/02, minuit)
  - Rapports (sujets de recherche) : 5/02/2025, minuit

Modalités d'évaluation

Deux modalités, selon le parcours :

1. Une modalité parcours professionnel ou recherche empirique :
   - un projet à faire en équipe de 2 à soutenir à la fin de l'UE :
     1. premier audit intermédiaire : 10%
     2. soutenance à mi-semestre (programme) : 30%
     3. second audit intermédiaire : 10%
     4. recette de fin de projet (programme, doc, résultats) : 50%
   - sujets :
     - parcours pro : gestionnaire d'énergie autonome.
     - parcours recherche : modèles de composants et outils pour les systèmes cyber-physiques et autonomiques, extensions au projet.
2. Une modalité parcours recherche conceptuelle et théorique :
   - un travail individuel de recherche bibliographique avec une présentation et un rapport en fin d'UE :
     1. pré-rapport, plan du reste à faire à mi-semestre : 25%
     2. présentation finale : 25%
     3. rapport : 50%

Contenu semaine par semaine I

1. Introduction aux systèmes de contrôle cyber-physiques auto-adaptables
2. Architectures logicielles dynamiquement adaptables
3. Modélisation des systèmes de contrôle cyber-physiques
4. Simulation des systèmes cyber-physiques
5. Simulation modulaire et DEVS
6. Contrôle et autonomicité
7. De l'entité adaptable — principes généraux et conception des capteurs

Contenu semaine par semaine II

8. De l'entité adaptable — conception des actionneurs et méthodologie de développement
9. De l'entité de contrôle — principes et conception
10. De l'entité de contrôle — méthodologie et développement
11. De l'entité auto-adaptable
12. Composition, coopération et coordination des entités auto-adaptables
13. Décision et contrôle
14. Méta-contrôle, informatique systémique et émergence

Objectifs pédagogiques du cours 1

- Comprendre les enjeux de l'informatique contemporaine dont les systèmes :
  - s'exécutent de plus en plus en *permanence*,
  - en lien direct (au sens large) avec le monde réel (*cyber-physiques*)
  - avec lequel ils interagissent de manière autonome (*capteurs, contrôle, actionneurs*)
  - et capables de s'auto-adapter (*autonomiques*).
- Comprendre ce qu'est l'*auto-adaptabilité dynamique* des logiciels, et les besoins auxquels elle répond.
- Comprendre les notions fondamentales des *architectures autonomiques*, leurs caractéristiques ainsi que les connaissances et les savoir-faire mis en jeu dans ce domaine.

Plan

L'informatique aujourd'hui

- Tout devient réparti et embarqué :
  - Omniprésence de l'informatique répartie, ouverte et à ressources partagées (*service-oriented computing*, *cloud computing*, . . .), s'exécutant en permanence (24/24, 7/7). . .
  - . . . sous de fortes variations de charge et de conditions d'exécution.
- Internet des objets et mobilité comme partie intégrante dans une part croissante des applications (réseaux sans fil), et donc une plus grande variation encore des besoins en ressources.
- Systèmes « cyber-physiques » (*cyber-physical systems*) *i.e.*, systèmes informatiques connectés à des phénomènes du monde physique, en temps réel, pour les contrôler.
  - **Définition** : systèmes intégrant de manière forte informatique et monde physique, s'influençant l'un l'autre, parce que les calculs dépendent du monde physique (localisation), influent sur lui (voiture autonome) ou dont la performance dépend (disponibilité de l'énergie).

Vers les notions plus générale de « système »

- La notion de système cyber-physique pousse à s'intéresser plus généralement à la convergence entre l'informatique et l'ensemble des systèmes technologiques et sociaux-économiques.
  - D'un train à grande vitesse au système de transport à l'échelle d'un pays, tout peut maintenant être vu comme des systèmes cyber-physiques à plus ou moins grande échelle, souvent intégrés les uns aux autres.
- Ceci nous ramène à des théories qui ont été élaborées dès l'émergence des premiers ordinateurs : la *cybernétique*, cherchant à appréhender comment de tels systèmes pourraient ête conçus, modélisés, analysés puis construits.
- Cette vision a donné naissance à la *théorie des systèmes*, apparue dès les années 1950, dont les principes sont fondés sur l'étude de la structure des systèmes, de leurs flux, de la dynamique de leurs interactions et de leur évolution.

Qu'est-ce qu'un « système » et pourquoi en étudier ?

- La science et l'ingénierie s'intéressent depuis longtemps à la notion de système, la première pour mieux les *comprendre*, l'autre pour mieux les *construire*.
- La notion de *système* est d'autant plus difficile à cerner qu'elle a été étudiée sous différents angles et dans différents contextes :
  - systèmes *naturels*, comme le système gravitationnel du soleil, de ses planètes et de leurs satellites ;
  - systèmes *sociaux*, comme la ville, avec ses habitants, ses activités, ses transports, etc. ;
  - systèmes *technologiques*, comme un vaisseau spatial, son fonctionnement mécanique et électronique, ses instruments de pilotage, ses instruments scientifiques, etc.
- L'une des principales raisons d'étudier les systèmes est de savoir les *contrôler*, c'est-à-dire agir sur certaines de leurs propriétés, sur lesquelles on a prise, de manière à en influencer d'autres (performance, consommation, etc.) qui ont un intérêt particulier.

Fonctions et finalités des systèmes

- On attribue généralement à un système une *fonction*, intentionnelle ou non, qui explique son comportement et associe ses différentes propriétés.
- À cette fonction, on prête des *qualités* qui vont donner lieu à une *évaluation*, ce qui conduit, lorsque c'est possible, à la recherche des *meilleures solutions*, celles présentant les meilleures qualités, la meilleure évaluation.
  $\Rightarrow$ *En informatique : la performance, la qualité de service, ...*
- Ainsi, pour *comprendre*, *analyser* et *mieux construire* un système, il est très utile :
  - de le *modéliser* de manière à *capturer* ses propriétés essentielles,
  - d'en *inférer* les qualités à partir de ce modèle et de ses propriétés
  - et ainsi *prévoir* le plus fidèlement possible le comportement du système réel.

Nature, comportement et propriétés des systèmes

- Les systèmes informatiques exhibent comportements déclenchés par des *phénomènes discrets* vus comme des *événements* se produisant à des *instants ponctuels* dans le temps :
  - recevoir d'une requête,
  - appeler d'un service,
  - presser un bouton, etc.

  $\Rightarrow$ Des phénomènes modélisables par des *automates*, changeant d'état aux occurrences d'*événements*.
- Mais ils exhibent également des évolutions d'état *graduelles* mieux capturés par des *modèles continus* :
  - la durée d'un calcul,
  - le nombre moyen de requêtes par seconde,
  - la consommation d'énergie, etc.

  $\Rightarrow$ Des phénomènes plutôt modélisables par des *équations* mathématiques *continues*.

Les systèmes hybrides

- Les systèmes qui exhibent *à la fois* des évolutions *continues* et *discrètes* sont dits ***hybrides***.
- C'est le cas des systèmes cyber-physiques dont la partie informatique implique des évolutions discrètes et dont la partie physique implique des évolutions continues.
- Le terme « *systèmes hybrides* » désigne une famille de modèles mathématiques (formels, donc) étudiés depuis 25 ans environ, par quatre communautés de recherche :
  - mathématiques,
  - informatique théorique,
  - automatique (théorie du contrôle) et
  - recherche opérationnelle et théorie de la décision.
- Dans la suite d'ALASCA, nous allons nous appuyer particulièrement sur ces systèmes hybrides formels pour modéliser les CPS, et qui seront le sujet central du cours 3.

Systèmes complexes

- La théorie des systèmes s'est rapidement intéressée aux systèmes dits *complexes*.
  - La notion de système complexe possède plusieurs acceptions ; en théorie des systèmes, il ne s'agit pas simplement de systèmes *compliqués* à comprendre ou à construire, mais de systèmes qui *défient* les méthodes scientifiques et techniques usuelles.
  - La méthode *cartésienne*, se fondant sur la décomposition puis l'étude des parties pour comprendre le tout, n'arrive plus à rendre compte du comportement de tels systèmes.
- Compte tenu des limitations de l'informatique au début de la théorie des systèmes, les seuls systèmes réellement complexes étaient des systèmes naturels (écosystèmes, par exemple) ou des systèmes sociaux (économie de la France, par exemple).
- L'étude des systèmes complexes a alors pris un axe *analytique* (compréhension de systèmes existants) plutôt que *constructif* (comment construire un système complexe correct).

La systémique

Systémique : science de l'étude des systèmes trop complexes pour être abordés par raisonnement analytique cartésien.

Raisonnement analytique : méthode dérivée des travaux de Descartes où la compréhension d'un tout s'obtient par *induction* à partir de la compréhension *individuelle* de ses parties.

- Les systèmes complexes évoluent dans le temps et se caractérisent globalement par des mécanismes diffus visant :
  - au maintien des structures existantes (homéostasie, autorégulation), et
  - à l'évolution par « apprentissage » à partir des interactions avec l'environnement et d'une mesure d'efficacité.
- Grandes références accessibles :
  - « *La méthode* » d'Edgar Morin centrée sur les systèmes naturels/sociaux ;
  - « *Le macroscope* » de Joël de Rosnay centré sur les systèmes biologiques.

Plan

Évolution contrôlée

- Les systèmes évoluent dans le temps sous l'impulsion de deux grands types de phénomènes :
  1. phénomènes *subis*, qui forcent une évolution, comme la dégradation due à l'usure ;
  2. phénomènes *contrôlés*, qui permettent d'imposer volontairement une évolution souhaitée, comme le redémarrage d'un système pour résoudre une panne transitoire.
- Dans le cas d'un système cyber-physique, les phénomènes subis proviennent de son contexte d'exécution ; amener une évolution souhaitée suppose de pouvoir modifier :
  - des paramètres (configuration, quantité de ressources disponibles, etc.),
  - mais également son comportement, à savoir son code.
- Par ailleurs, comme souligné en systémique, ces évolutions contrôlées devraient pouvoir être impulsées par le système lui-même, de manière *autonome*.

Des systèmes adaptables aux systèmes autonomiques I

Les besoins d'adaptation des systèmes découlent des conditions dans lesquelles ils s'exécutent :

1. Un besoin d'adapter les systèmes à leur contexte d'exécution pour assurer la meilleure performance et la meilleure utilisation des ressources possibles.
   - **Systèmes adaptables** : systèmes qui peuvent être adaptés à leurs différents contextes d'exécution soit par choix des algorithmes ou par des choix de paramètres de déploiement.
2. Si les conditions d'exécution évoluent dynamiquement de manière importante, un besoin de s'adapter en permanence, pendant toute la durée de l'exécution.
   - **Systèmes *dynamiquement* adaptables** : systèmes qui peuvent être adaptés pendant leur exécution.
3. Lorsque les évolutions sont plus rapides et les interventions manuelles plus complexes, un besoin de systèmes sachant s'adapter eux-mêmes de manière autonome.

Des systèmes adaptables aux systèmes autonomiques II

- **Systèmes auto-adaptables** : systèmes qui peuvent s'adapter eux-mêmes pendant leur exécution.

4. Et, concommittamment, le besoin de réagir aux évolutions de l'environnement physique en prenant des décisions et en exerçant sur lui une forme de *contrôle* :
   - **Systèmes de contrôle cyber-physiques** : systèmes cyber-physiques interagissant avec le monde physique grâce à des *capteurs* lui permettant d'en connaître l'état et à des *actionneurs* lui permettant d'agir sur ce dernier (ex.: robotique autonome, pilote automatique, . . .).

   Et sur la partie informatique, une déclinaison particulière :
   - **Systèmes autonomiques** : systèmes informatiques auto-adaptables exerçant sur eux-mêmes un contrôle de même nature que celui exercé par les systèmes de contrôle cyber-physiques.

- *Focalisation de l'UE :*

  les ***systèmes de contrôle cyber-physiques autonomiques****.*

Auto-adaptabilité dynamique des systèmes informatiques

- Plusieurs moyens :
  - modification de paramètres de configuration (taille de tampons de données, nombre de *threads*, etc.) ;
  - modification de l'allocation des ressources à l'exécution du programme (capacité système, bande passante réseau, etc.) ;
  - modification du code (programme de l'application, bibliothèques chargées dynamiquement, ...) ou de la façon d'exécuter le code (comment il se compile — au sens large, signification donnée aux instructions, compilation à la volée) ;
  - modification de l'architecture de l'application (nombre de composants, déploiement sur serveurs virtualisés, ...).
- Donc, *à la louche*, quatre grands types d'adaptations :
  1. Paramétriques (paramètres d'exécution, tailles, ...).
  2. Ressources (qualité de service versus ressources allouées).
  3. Fonctionnelles (modifications des fonctionnalités, du code).
  4. Architecturales (modifications de l'assemblage de composants).

Exigences pour réaliser l'auto-adaptabilité

Un système informatique auto-adaptable doit avoir :

- une **connaissance de lui-même**, de sa forme, de son contenu, de sa configuration, de son état d'exécution courant, etc. ;
- une **connaissance de son contexte d'exécution**, à tous les niveaux, y compris les propriétés de l'exécution en cours ;
- des **mécanismes lui permettant de se modifier lui-même**, c'est-à-dire de changer les ressources mises à sa disposition, de même que modifier sa propre architecture et son propre code ;
- une **capacité à décider quand et comment** il doit s'adapter par rapport à son état courant ;
- et enfin, *en poussant à la limite*, une **capacité à modifier sa manière de s'auto-adapter** *i.e.*, l'application « récursive » de la fonction d'auto-adaptabilité à elle-même.

Problématiques induites en développement logiciel

- Représentation du système lui-même de telle manière à que ce dernier puisse se manipuler et se modifier à l'exécution.
- Mécanismes d'auto-adaptation :
  - Configuration au déploiement et reconfiguration dynamique.
  - Adaptation pendant l'exécution, par exemple lors de la connexion avec d'autres systèmes.
  - Allocation et réallocation régulières des ressources disponibles à l'exécution.
  - Déploiement continu, à chaud, des évolutions logicielles.
- Tout en maintenant une conception modulaire et ouverte en présence d'auto-adaptabilité : composabilité parallèle des fonctionnalités et des mécanismes d'auto-adaptation.
- Dans un contexte où les coûts humains (développement, maintenance, exploitation) croissent continuement.
  $\Rightarrow$ Besoin d'***automatisation***.

Plan

Typologie des causes d'adaptation à prendre en compte

- Les adaptations sont réalisées pour répondre à des besoins de nouvelles fonctionnalités, de meilleure qualité de service, de pannes à circonvenir ou de réactions imposées par l'évolution de l'environnement.
- *Comment* ces besoins apparaissent-ils et *à quel rythme* ?

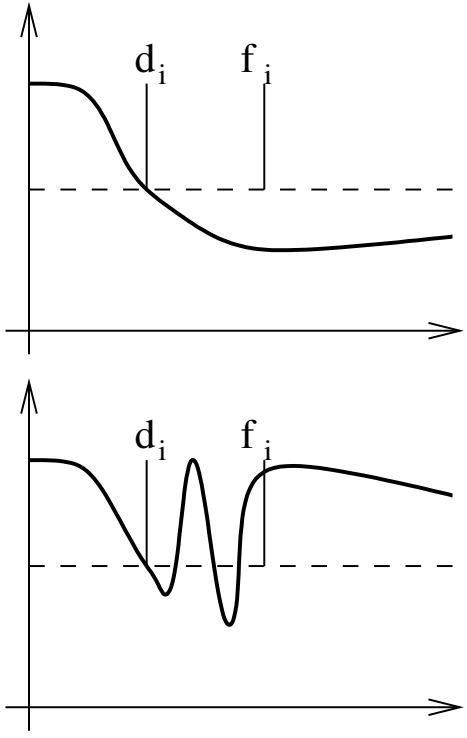
Continuum multi-dimensionnel entre

- des changements rares, prédictibles et déterministes
  Exemples : changements planifiés par l'utilisateur dans l'évolution logicielle ou le mode d'opération, niveau de charge de la batterie (+ ou -) d'un portable, ...
- des changements fréquents et aléatoires
  Exemples : disponibilité de nouveaux services (ambiants *i.e.*, localisation), évolution de la bande passante du réseau, etc.

Problème : *le temps* et l'*inertie !*

- Exemple: adaptation à la bande passante d'un réseau sans fil.

- S'adapter prend du **temps** et consomme des **ressources**.
- L'état (interne et contexte) évolue dans le temps (ici, par exemple, la bande passante).
- *Que se passe-t-il si l'état courant n'est plus compatible avec une adaptation qui vient de se terminer ? Que se passe-t-il si les ressources disponibles sont insuffisantes pour adapter complètement le système sur une durée donnée ?*
- Le nouvel état va requérir une nouvelle adaptation.

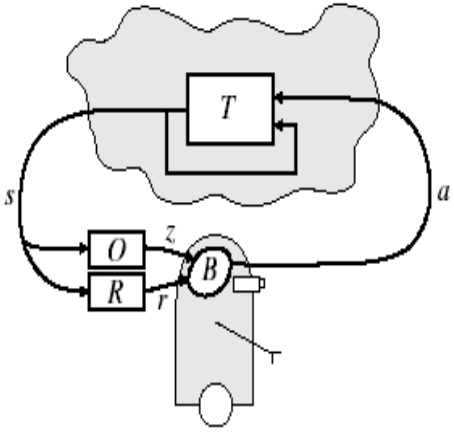


d_i : instant de décision
f_i : fin de l'adaptation lancée à d_i

- Ce processus peut potentiellement mener à une forme d'emballement (« *thrashing* ») : le système s'adapte continuellement, consommant pour cela toutes les ressources disponibles, et ne faisant plus rien d'autre.

Problème bien connu

- Problème d'automatique (contrôle) connu depuis les premières applications, comme le pilotage automatique ou l'automatisation des canons anti-aériens pendant la deuxième guerre mondiale.
- Modèle d'automate de contrôle
  - T : fonction de transition (évolution) de l'environnement selon la décision *a* et l'état *s*
  - O : observation *z* de l'état *s*
  - R : récompense *r* (reward) combinant la *désirabilité* de l'état *s* et le *coût* du contrôle *a*
  - B : comportement (behaviour) c'est-à-dire la décision menant au contrôle *a* exercé



- L'auto-adaptation doit se fonder sur des décisions équilibrant le gain potentiel de faire l'adaptation par rapport à l'état atteint, incluant l'évolution future du système à partir de cet état.
- Elle doit aussi assurer certaines propriétés sur l'évolution de l'état du système, comme éviter l'emballement cité précédemment.

Auto-adaptation et contrôle = représentation + décision

- Pour s'adapter à la variation des conditions d'exécution et des ressources disponibles en agissant sur un système ou son contexte, il faut savoir décider *quand* et *comment* s'adapter.
- Cela devient un problème :
  - de décision, au sens de la *théorie de la décision* c'est-à-dire quelle adaptation choisir en fonction d'une analyse coût/bénéfice de l'action choisie versus l'état qu'elle permet d'atteindre mais également les évolutions subséquentes rendues possibles ;
  - de contrôle au sens de la *théorie du contrôle*, c'est-à-dire comment gérer son impact sur la dynamique du système étant donné l'*écoulement du temps*, son *inertie intrinsèque* et les *contraintes* posées sur les adaptations possibles.

  Ce qui peut être abordé par les approches du *contrôle (optimal)*, de la *recherche opérationnelle*, ou des techniques de *IA*.
- Les notions de contrôle seront abordées au cours 4 alors que celles de la décision le seront au cours 13.

Du besoin au logiciel

- Réaliser un système auto-adaptable *correct* et *sûr* est une tâche complexe.
  - Ils sont reconnus comme très difficiles à spécifier, mettre au point, tester, vérifier et valider.
- L'objectif principal de l'UE ALASCA est de chercher à répondre aux questions suivantes :
  - *Que doivent proposer un **modèle à composants** et une **architecture logicielle** pour faciliter ces activités et soutenir la mise en œuvre d'une méthodologie de développement propre aux systèmes de contrôle cyber-physiques autonomiques ?*
  - *Comment les construire **efficacement** tout en s'assurant qu'ils soient **corrects** et **sûrs** ?*
- Pour répondre à ces questions, nous allons d'abord regarder ce qui se fait sous le nom d'*autonomic computing*, que nous traduisons par *informatique autonomique* pour plus de simplicité.

Informatique autonomique : objectifs et concepts

Définition : ensemble des concepts et techniques utilisés pour permettre à un système informatique de gérer lui-même les éléments qui le composent et d'entreprendre automatiquement les actions nécessaires à son fonctionnement optimal, sans intervention humaine.

Principaux instigateurs : IBM et ses chercheurs au début des années 2000 qui visaient à

- Mettre l'informatique au service de la gestion des applications.
- Élever considérablement le niveau de fiabilité des services informatiques.
- Réduire le coût humain de la fonction de gestion des services et applications.

Caractéristiques d'un système autonomique selon IBM

- *Auto-configurabilité*, en fonction des conditions de déploiement et de leur évolution en cours d'exécution ;
- *Auto-optimisabilité*, en fonction des usages et des ressources disponibles ;
- *Auto-réparabilité*, lors de pannes ou d'événements imprévus ;
- *Auto-protection*, face aux attaques externes ;
- *Ouverture*, ne s'enferme pas dans des contextes hermétiques mais on s'ouvre plutôt à l'extérieur en adhérant à des standards reconnus.

Lien entre autonomicité et contrôle cyber-physique

- L'informatique autonomique s'intéresse pour l'essentiel à l'auto-adaptabilité informatique *fermée sur elle-même*.
  - Pour cela, le contrôle autonomique reprend des notions de la théorie du contrôle (capteurs, actionneurs, politique, etc.).
  - Le patron architectural de base (IBM, inspiré de l'automatique) est formé d'un *élément de contrôle* qui interagit avec un *élément adaptable* via des capteurs et des actionneurs.
  - Ce sont donc exactement les mêmes approches que celles utilisées par les systèmes de contrôle cyber-physiques pour suivre les évolutions du monde physique et agir sur lui.
- Un système de contrôle cyber-physique autonomique utilise ainsi les mêmes modèles et techniques à deux niveaux :
  1. Au niveau de l'élément logiciel de base (élément adaptable en informatique autonomique) quand il est lui-même la partie « *cyber* » d'un système cyber-physique.
  2. Au niveau de l'élément de contrôle parce qu'il agit sur l'élément logiciel de base de manière similaire à ce que ce dernier fait avec le monde physique.

Auto-adaptabilité des systèmes cyber-physiques I

- Pendant longtemps, l'informatique autonomique s'est intéressée à l'auto-adaptabilité mais à l'intérieur des frontières « cyber », c'est-à-dire une adaptation à des phénomènes *endogènes* à l'exécution des programmes :
  - consommation mémoire,
  - appels de procédures (nombre, paramètres fréquents, ...),
  - « poids » dynamiques respectifs des processus sur le processeur,
  - services du système d'exploitation disponibles,
  - etc.
- Avec les systèmes embarqués ou communiquants, elle s'est intéressée de plus en plus à de l'adaptation à des phénomènes *exogènes*, c'est-à-dire de l'environnement :
  - variation de bande passante des réseaux sans fil, la mobilité, la *consommation d'énergie*,
  - performances des services distants utilisés,
  - etc.

Auto-adaptabilité des systèmes cyber-physiques II

- Pour *modéliser correctement* les phénomènes exogènes, mais aussi les phénomènes endogènes quantitatifs (performances, etc.), il faut comprendre leur *nature*, leur *évolution* et leur *impact* sur l'exécution.
- Et tout aussi important, il faut comprendre l'effet des adaptations faites par les applications sur ces phénomènes et en particulier sur leurs évolutions en continu.
- Ces phénomènes ont été étudiés dans les domaines de la *théorie des systèmes* et de la *théorie du contrôle*, et pour lesquels des modèles ont été proposés.
- En ALASCA, nous allons explorer les contributions de ce domaines, non pour en devenir des spécialistes, mais pour en connaître les principaux concepts et être capable d'en comprendre les implications.

Disciplines d'appui : un domaine au spectre très large

- Systèmes répartis en réseau
- Systèmes embarqués temps-réel
- **Génie des logiciels à base de composants**
- Modélisation comportementale
- Modélisation stochastique
- Simulation
- Décision
- Théorie du contrôle
- Coordination
- Systèmes complexes

Et bien d'autres encore...

Plan

Caractère auto-adaptable des architectures logicielles

- Recherche de modularité, deux problèmes fondamentaux :
  1. Comment programmer des *entités modulaires* pour les rendre auto-adaptables ?
     - composant, modèles, connaissance de son propre état et de celui du monde physique, capacité d'auto-adaptation, ...
  2. Comment les composer pour former des systèmes (architectures) auto-adaptables ?
     - compatibilité, composition « parallèle » du code fonctionnel, des modèles, de l'auto-adaptabilité, ...
- Tryptique central en conception modulaire :

  comportements (code, modèles) / interfaces / composition
- L'entité de contrôle cyber-physique et auto-adaptable doit être conçue autour :
  1. de modèles fonctionnels, de comportement et d'adaptation ;
  2. d'interfaces explicites exposant et contractualisant fonctions, adaptations, synchronisation et qualité de service ;
  3. de mécanismes de composition couvrant ces mêmes aspects.

Vers des architectures CCP autonomiques

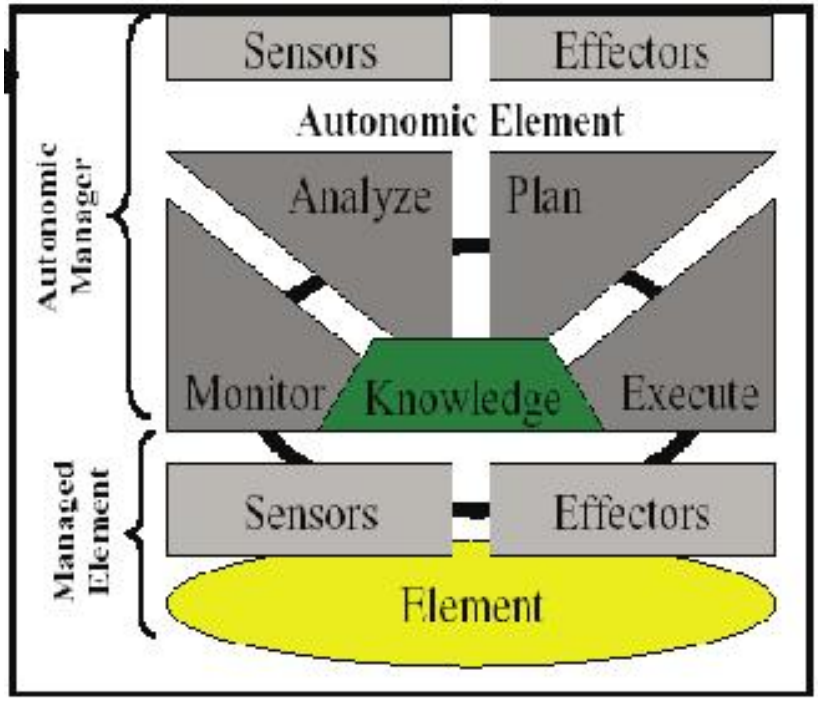
- Composants de contrôle cyber-physiques autonomiques inspirés :
  - des modèles de composants répartis, temps réel et embarqués ;
  - de l'automatique par l'utilisation de capteurs, d'actionneurs et de contrôleurs en boucle fermée ;
  - de la mesure, de l'instrumentation et de la réflexion pour réaliser la « *connaissance de lui-même* » ;
  - des automates et systèmes hybrides pour modéliser le comportement ;
  - des interfaces riches et de la coordination pour la composition,
  - de la simulation modulaire pour le test, la validation et la vérification.
- Intégration de notions de décision : ces composants s'inspirent de l'automatique, de la RO, de l'IA et de la robotique autonome.
- Le contrôle vise deux objectifs :
  1. homéostasie : maintien des équilibres à court terme avec une décision souvent purement réactive ;
  2. évolution : introduction de nouvelles capacités et de nouvelles politiques sur le plus long terme.

Architecture des éléments autonomiques (inspirés IBM)

- Composants autonomiques : un continuum du matériel aux composants logiciels métiers en passant par le système d'exploitation, le réseau, les intergiciels et les programmes.
- Se découpent en deux parties : élément géré et élément contrôleur.
- Éléments gérés (*managed elements*) : composants adaptables implantant des interfaces capteurs et actionneurs.
- Éléments contrôleurs (*autonomic manager*) : boucle de contrôle apportant l'auto-adaptabilité par :
  1. la collecte des informations sur l'exécution et le contexte,
  2. leur traitement et le diagnostic pour arriver à une décision,
  3. l'élaboration des actions d'adaptation nécessaires, et enfin
  4. l'intervention sur l'élément géré pour l'adapter.

Éléments autonomiques

- La composition d'un élément géré et de son contrôleur autonomique forme un *élément autonomique*.



- Surveillance : réception des données via les senseurs.
- Analyse : obtention d'un diagnostic.
- Planification : détermination des actions à prendre.
- Exécution : mise en œuvre du plan via les actionneurs.

- L'élément autonomique représente un automate de contrôle générique déjà présenté au transparent 28.

De l'élément géré : capteurs et actionneurs

- Les architectures autonomiques à la IBM s'intéressent aux éléments gérés que dans la mesure où ils permettent :
  - d'observer leur structure interne, leur état et leur QdS courants ;
  - de leur appliquer des adaptations.
- Empruntant au vocabulaire de l'automatique, on utilise :
  - le terme *capteur* pour désigner les dispositifs mis en place par l'élément géré pour observer son état courant, et
  - le terme *actionneur* pour désigner l'ensemble des opérations mises en place par l'élément géré permettant de l'adapter dynamiquement.
- En termes d'architecture logicielle, les informations fournies par les capteurs sont rendues visibles et accessibles via des *interfaces capteurs* et les opérations d'adaptation le sont par les *interfaces actionneurs*.

Du contrôleur autonomique : contrôle en boucle fermée I

Quatre grandes étapes :

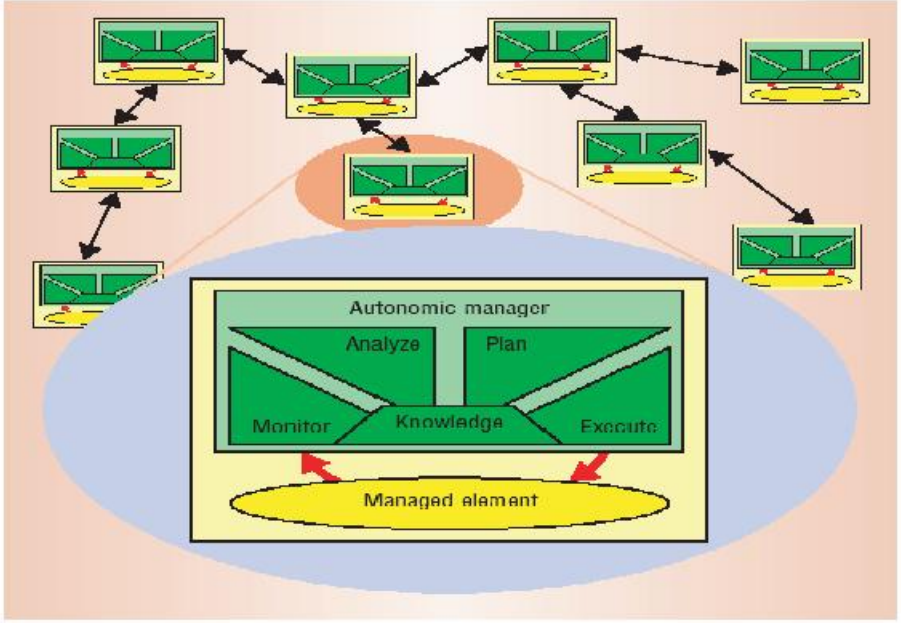
1. Supervision :
   - opérations permettant de reconstruire une image fidèle de l'état de l'élément géré, de son contexte et de son environnement ;
   - interfaces capteurs de l'élément géré, de même que des sondes sur le contexte et l'environnement ;
   - techniques fondées sur l'historique pour filtrer, débruiter, lisser, simplifier et abstraire les données obtenues pour construire un état utilisable par le modèle de décision et contrôle.
2. Analyse : applique des techniques de corrélation, décision et diagnostic pour
   - déterminer l'état courant à partir des données de supervision,
   - identifier les écarts entre cet état courant et l'état souhaitable,
   - décider s'il est nécessaire d'adapter ou non, et
   - identifier l'*action* d'adaptation et les éventuels paramètres de cette adaptation.

Du contrôleur autonomique : contrôle en boucle fermée II

3. Planification :
   - détermine l'ensemble des *opérations* d'adaptation nécessaires pour réaliser les *actions* d'adaptations choisies, et
   - les organise dans un *plan*, incluant des opérations composites, exécutables sur différentes entités et avec des dépendances temporelles explicites entre elles.
4. Exécution :
   - lance et supervise l'exécution du plan obtenu de la phase de planification ;
   - moteur d'exécution adapté aux types d'opérations élémentaires et composites apparaissant dans le plan, aux actions préparatoires nécessaires (comme l'arrêt momentané d'exécution des éléments gérés) et la synchronisation des actions élémentaires.
   - phase d'exécution est la mise en œuvre du plan par l'ordonnancement, le lancement, le contrôle de réalisation et l'enchaînement des opérations d'adaptation.

Composition des éléments autonomiques

- La plupart des systèmes étant trop complexes et physiquement répartis, ils ne peuvent être contrôlés de manière *monolithique*.
- Il faut donc les voir comme un *assemblage* de composants autonomiques, chacun exerçant son propre contrôle local.



Nouveaux problèmes :

- Connexion entre les éléments autonomiques.
- Coordination des objectifs de contrôle.
- Planification et exécution conjointes des adaptations.

Architectures logicielles des domaines connexes I

- Plusieurs architectures autonomiques ont été développées, soit en suivant le plan architectural d'IBM, soit en suivant des variantes de ce dernier.
- Mais d'autres domaines de l'informatique se posent des problèmes similaires et fournissent des sources d'inspiration et de coopération intéressantes.
- Dans les systèmes temps réels embarqués répartis où des actions doivent être prises en réaction à des événements ou à l'évolution de l'état d'un système :
  - architectures synchrones,
  - architectures réparties de type GALS (*globalement asynchrones, localement synchrones*),
  - architectures asservies par le temps (*time-triggered*).
- Architecture de contrôle pour la robotique autonome, où le robot se contrôle lui-même en boucle fermée et peut se coordonner avec d'autres robots :

Architectures logicielles des domaines connexes II

  - architectures réactives,
  - architectures délibératives,
  - architectures hybrides, délibératives/réactives.
- Dans le domaine des systèmes d'information en réseau, les systèmes de traitement d'événements complexes (*complex event processing, CEP*) cherchent à capturer, identifier et corréler des événements spatialement et temporellement répartis pour déclencher des actions.
- Dans le domaine des systèmes multi-agents où s'exercent beaucoup de coordination, de décision collective et d'émergence de comportements globaux, mais généralement sans exigence de temps réel.

Récapitulons...

1. L'informatique aujourd'hui repose de plus en plus sur des logiciels s'exécutant en permanence dans des contextes d'exécution très variables, ce qui impose de les **adapter dynamiquement**.
2. Historiquement, les disciplines scientifiques de la **cybernétique**, de la **théorie des systèmes** et de la **systémique** ont étudié les systèmes ayant une capacité d'auto-adaptation, ce qui permet de faire le lien avec l'informatique par les systèmes cyber-physiques.
3. L'***auto-adaptabilité dynamique*** vise à utiliser la puissance de l'informatique au service de l'adaptabilité dynamique en construisant des logiciels capables de s'adapter eux-mêmes.
4. L'objectif de l'UE est de ***comprendre les problèmes*** liés à la conception et l'implantation de tels logiciels, et de former des ingénieur informatique ***capables de coopérer avec des spécialistes*** des autres disciplines nécessaires pour mener à bien des projets de développement de logiciels auto-adaptables.

Pour aller plus loin : sélection de lectures recommandées

- *La méthode*, Edgar Morin (différentes éditions, en 5 volumes initialement regroupés en deux dans des éditions récentes ; attention : long et érudit).
- *Le macroscope*, J. de Rosnay, Seuil, 1975 (vulgarisation scientifique très accessible).
- *The Vision of Autonomic Computing*, J.O. Kephart et D.M. Chess, Computer, 36, pp. 41–50.
- *Autonomic Computing*, P. Lalanda, J. McCann et A. Diaconescu, Springer-Verlag, 2013, chapitres 1 à 4.
- *An Architectural Blueprint for Autonomic Computing*, IBM, 2006.
- *Autonomic Computing — Concepts, Infrastructure and Applications*, M. Parashar et S. Hariri, CRC Press, 2007, chapitres 1, 3 et 5.