Bibliographie SAC

Benjamin Lussier

April 21, 2004

Contents

1 Tolérance aux fautes

- **Guide de la Sûreté de Fonctionnement (seconde édition)**
  [Laprie et al. 1996] : *GUIDE*
  Ce livre présente la terminologie et les concepts de base de la sûreté de fonctionnement des systèmes informatiques, un état de l'art en terme de moyens pour cette sûreté de fonctionnement, et la proposition d'un modèle général de développement de systèmes sûrs de fonctionnement. En particulier il définit et décrit des notions fondamentales au projet Systèmes Autonomes Critiques : les entraves à la sûreté de fonctionnement (fautes, erreurs, défaillances), et la tolérance aux fautes dont il présente également les techniques de mise en oeuvre. Ces techniques

font généralement appel à des principes de redondance et diversification.

- **Tolérance aux fautes dans les systèmes critiques**
  [Essamé et al. 2000] : *SYSCRIT*
  Ce rapport présente une analyse des moyens et méthodes mis en oeuvre pour assurer la sûreté de fonctionnement dans les systèmes de contrôle-commande critiques, à travers des exemples du domaine avionique, nucléaire et ferroviaire. En particulier il s'intéresse à la tolérance aux fautes physiques et aux fautes de conception de ces systèmes, réalisée par redondance et diversification, et met en évidence l'antagonisme entre exigences de sécurité-innocuité et de disponibilité qui contraignent de tels systèmes.

- **The Safety-Bag Expert System in the Electronic Railway Interlocking System Elektra**
  [Klein 1991] : *SBAG*
  Cet article présente rapidement l'architecture de la couche traitement de données du système électronique de signalisation ferroviaire Elektra, diversifiée en une partie commande et une partie surveillance, puis décrit le langage et l'implémentation de la partie surveillance : le système expert Safety-Bag. En particulier il définit les fonctions du Safety-Bag : vérification de la sécurité-innocuité des instructions générées par une commande de l'opérateur, vérification de la sécurité-innocuité des instructions générées par la partie commande, réaction à des événements extérieurs. Actuellement ce terme de Safety-Bag est utilisé pour nommer les mécanismes de tolérance aux fautes qui visent à remplir ces différentes fonctions.

2 Robotique

2.1 Concepts - principes généraux

- **System Principle on Emergence of Mobiligence and Its Engineering Realization**
  [Asama et al. 2003] : *MOBILIGENCE03*
  Cet article présente le concept de recherche Mobiligence, d'après l'idée que l'intelligence peut émerger de l'interaction entre un système artificiel et son environnement, en particulier à travers la mobilité du système. Ce concept est basé sur la boucle fermée "perception, décision, action", l'idée d'un système autonome décentralisé (composé de modules délibératifs mobiles) et de l'appropriation par le système de l'environnement. Trois axes de recherche sont envisagés : l'adaptabilité

corporelle (*embodied plasticity*) qui englobe la diversité et la décentralisation des actionneurs, l'intégration dans l'environnement (*co-embodiment with environment*), et l'inférence d'information (*internal generation of abductive information*) qui vise à déduire par des règles des informations supplémentaires aux perceptions du système.

2.2 Sûreté de fonctionnement

2.2.1 Principes généraux

- **Design and Manufacturing of Actuated Medical Robots : a Safety Point of View**
  [Duchemin et al. 2004] : *MEDSAFE03*
  Cet article présente le problème de la sûreté de fonctionnement de systèmes robotiques dans le milieu médical. Il donne des conseils pour mettre en œuvre la prévision et la tolérance aux fautes de tels systèmes, aux niveaux électromécanique, électrique et logiciel. Pour ce dernier niveau, l'utilisation de processus dédiés et de systèmes de surveillance de type *safety-bag* sont mentionnés. Des notions de redondance, de gestions d'erreur et de modes de défaillance sont discutées. L'article présente ensuite deux systèmes robotiques médicaux et leurs caractéristiques mécaniques et logicielles vis-à-vis de la sûreté de fonctionnement.

2.2.2 Mesures

- **Reliability Analysis of Mobile Robots**
  [Carlson & Murphy 2003] : *RELMOB03*
  Cet article présente des études menées sur les défaillances de robots mobiles non-autonomes en intérieur et extérieur. Treize Robots ont été suivis pendant deux ans, mettant en évidence une faible fiabilité et disponibilité : un MTBF de l'ordre de 8 heures et une disponibilité de moins de 50%. Les robots d'extérieur présentent un plus fort taux de défaillances que les robots d'intérieur (peut-être à cause des plus fortes exigences qu'imposent ce type de terrain), et bien que les plate-formes matérielles soient l'origine la plus commune des défaillances (environ 42%), le système de contrôle (dont le système d'exploitation) en est responsable dans 29% des cas.

- **Designing a Secure and Robust Mobile Interacting Robot for the Long Term**
  [Tomatis et al. 2003] : *ROBOX03*
  Cet article présente l'implémentation du robot-guide autonome RoboX9 et une étude de ses défaillances pendant cinq mois d'utilisation lors

d'une exposition. Le système de contrôle est organisé en modules, répartis en fonction de leur criticité vis-à-vis de la sûreté de fonctionnement entre deux processeurs différents (un processeur de navigation, un autre d'interaction avec les visiteurs de l'exposition). La sûreté de fonctionnement est mise en œuvre par un système de chiens de garde sur les modules les plus critiques, et un processeur supplémentaire qui supervise principalement leur fonctionnement. Le MTBF a atteint 4,61 heures à la quatrième semaine d'utilisation ; près de 80% des défaillances se sont produits sur le processeur d'interaction, pour 16% sur le processeur de navigation, et seulement 4% pour le niveau matériel.

2.2.3 Algorithmes amenant une meilleure SdF

- **Inevitable Collision States A Step Towards Safer Robots ?**
  [Fraichard & Asama 2003] : *INEVCOL03*
  Cet article présente la notion d'état de collision inévitable (*inevitable collision state*), qui sont des états à partir desquels il est impossible pour un système d'éviter une collision. L'article décrit formellement cette notion, ainsi que celle d'obstacle de collision inévitable (*inevitable collision obstacle*) qui représente l'ensemble des états de collision inévitable lié à un obstacle. Il donne ensuite deux exemples théoriques (un très simple, puis un plus réaliste), et présente des résultats préliminaires dans la planification de déplacements sûrs à partir de ces notions.

- **A Study toward Cognitive Action with Environment Recognition by a Learning Space Robot**
  [Senda et al. 2003] : *COGSPACE03*
  Cet article présente une application d'un algorithme de *reinforcement learning*, le *Q-learning*, permettant à un système autonome de remplir une tâche dont la réalisation a précédemment échouée (à cause par exemple d'une modification de l'environnement). Le principe de l'algorithme consiste à converger vers une solution par une succession de tentatives. L'application étudiée dans l'article est une simulation de robot spatial auquel on demande d'assembler une structure. Cependant pour une raison de temps de calcul, l'algorithme n'est pas applicable en opération dans des cas complexes, et l'article présente ensuite une méthode pour simplifier sa résolution et réduire son temps d'exécution.

- **An Error Restraining Method for Accurate Freeform Surface Cutting**
  [Jaganathan & Lin 2003] : *FREECUT03*
  Cet article présente un algorithme de correction d'erreur (généralement des aléas causés par l'environnement) pour une machine de découpe

controllée par ordinateur. Cet algorithme utilise un feedback : à partir de la position précédente de l'outil l'ordinateur calcule la prochaine position et son écart par rapport à la position désirée, et génère un ordre à la machine suivant ces valeurs théoriques pour corriger l'écart au besoin. L'article présente ensuite rapidement la plate-forme expérimentale, les expériences menées et leurs résultats.

- **Proscriptive Bayesian Programming Application for Collision Avoidance**
  [Koike et al. 2003] : *BAYCOL03*
  Cet article présente une approche probabiliste bayésienne cherchant à traiter les incertitudes d'observation pour l'évitement de collisions. Il décrit rapidement les principes de la programmation bayésienne : le calcul de la répartition probabiliste des valeurs de chaque variable inconnue à partir de celles des variables connues, de connaissances préliminaires et de données expérimentales. L'application de ce type de programmation est très dépendante de l'application, particulièrement du choix des variables à considérer et de leurs relations. L'article donne l'exemple d'une plate-forme expérimentale et de deux approches de comportement : un comportement prescriptif (on cherche à ce que le robot s'éloigne d'un obstacle) et un comportement proscriptif (on cherche à ce que le robot ne s'approche pas d'un obstacle). Ce dernier met en œuvre avec succès l'évitement de collision.

- **Nonstop Update of Mission Critical Robotic Services**
  [Horiuchi 2003] : *NONSTOP03*
  Cet article présente une approche basée sur les brins d'exécutions (*threads*) pour mettre à jour un système autonome en cours d'exécution, en ne souffrant que d'une courte interruption de services. Il donne deux exemples d'application de l'environnement **NSU** (*NonStop Update framework*) basé sur cette approche. Un premier exemple applicable à plusieurs plate-formes est basé sur un courtier CORBA et les mécanismes de gestion d'exceptions du langage C++ ; il consiste en la création d'un brin mis à jour sur une machine distante, puis sa migration pour remplacer le brin original. Le second exemple est spécifique à une plate-forme (RTLinux), et n'est basé que sur les instructions de cette plate-forme ; il installe les modules mis à jour, crée un nouveau brin d'exécution, termine le brin original en récupérant son contexte, et enfin élimine les anciens modules.

- **A Suite of tools for Debugging Distributed Autonomous Systems**
  [Kortenkamp et al. 2002] : *DEBUG02*
  Cet article présente un ensemble d'outils permettant à un développeur

de modifier le code d'un système autonome distribué pour recueillir des données en cours d'exécution, puis d'analyser ces données pour vérifier que le comportement du programme est correct. Il présente tout d'abord le langage de logique temporelle ITCL (*Interval Temporal Checking Logic*) utilisé pour reformuler la spécification à partir de laquelle sera vérifiée l'exactitude des données. Il décrit ensuite la collecte de ces données : les différents processus du système appellent une fonction qui enregistrera les valeurs passées en paramètres dans une base de donnéees. L'article présente finalement un exemple d'application de ces outils sur le test d'un système automatique de recyclage d'eau de la NASA.

- **Fault tolerance in cooperative Manipulators**
  [Tińos et al. 2002] : *FEEDBACK02*
  Cet article présente un système de tolérance aux fautes de manipulateurs coopérant et reliés au même objet. Ce système est basé sur une modélisation mathématique de la position et de la vitesse de ces manipulateurs, pour détecter puis corriger 4 types possibles de fautes sur un manipulateur : une position erronée, une vitesse erronée, un blocage, ou une absence. A partir des commandes données, la position et vitesse théorique des manipulateurs sont détectées, puis comparées aux valeurs données par les capteurs ; la détection de blocage ou d'absence d'un manipulateur se fait à l'aide d'un réseau neuronal. Une fois une faute d'étectée, le système de tolérance aux fautes impose aux manipulateurs une loi de contrôle dépendant du type de fautes. L'article présente ensuite quelques résultats expérimentaux.

- **Pusher-washer : An approach to fault-tolerant tightly-coupled robot coordination**
  [Gerkey & Matarić 2002] : *PUSHER02*
  Cet article présente une approche de tolérance aux fautes dans la coopération robots, basée sur le systême d'allocation de tâches MURDOCH ; le cadre de travail est composé de plusieurs robots (dans la pratique 2 actionneurs et un superviseur) dont le but est de pousser un objet jusqu'à un point donné. Suivant la position de l'objet et de l'objectif, un des robots (le superviseur) établit des actions à effectuer, puis attribue ces actions aux autres robots en fonction de la compétence (présence de capteurs ou d'actionneurs) et la situation de chacun. Une défaillance d'un des robots est détecté par le superviseur si la situation n'évolue pas en accord avec les ordres donnés, et prise en compte dans la suite des opérations. L'article présente une implémentation des robots, et plusieurs jeux d'expériences, simulant la désactivation ou la défaillance puis récupération d'un des robots actionneurs.

- **Robotic Fault Detection Using Nonlinear Analytical Redundancy**
  [Leuschen et al. 2002] : *NLAR02*
  Cet article présente une version non-linéaire de la technique consistant à comparer les sorties de capteurs à une modélisation mathématique du comportement d'un système ; cette méthode est appelée NLAR (*NonLinear Analytical Redundancy*). Il décrit tout d'abord le principe de cette méthode, basée sur les matrices d'observabilité du système non-linéaire étudié. En fonction de ces matrices et de la sensibilité des capteurs, NARL donne la liste des équations à considérer pour détecter **toutes** les déviations possibles du système par rapport au modèle mathématique. L'article décrit ensuite en détail des expériences menés sur un robot pour valider cette approche à travers plusieurs simulations de fautes (blocage d'un capteur, imprécision graduelle d'un capteur) et une compariason avec un modèle linéaire du mee système.

- **An Analysis of Cooperative Repair Capabilities in a Team of Robots**
  [Bererton & Khosla 2002] : *REPAIR02*
  Cet article présente une analyse théorique du MTTF (*Mean Time To Failure*) entre une équipe de robots réparables et une de robots non-réparables. La réparation considérée consiste à prélever les composants nécessaires à la réparation sur des robots qui ont déjà défailli. L'analyse est basée sur des modèles de Markov et fait plusieurs assomptions, par exemple que les défaillances des robots sont exclusivement dues aux défaillances des composants ou que le temps de réparation est nul. Les résultats de cette analyse théorique prévoient que le MTTF de l'équipe de robots réparables serait plus élevé que celui de l'autre équipe, mais ne deviendrait vraiment significatif qu'à partir d'un certain nombre de robots dans l'équipe (autour de quatre ou cinq, suivant le nombre considéré de composants pouvant défaillir).

- **Merging Planning and Verification Techniques for "Safe Planning" in Space Robotics**
  [Aiello et al. 2001] : *SAFEPLAN01*
  Cet article présente une méthode de planification visant à rajouter des contraintes de sûreté de fonctionnement. Il décrit tout d'abord le langage du planificateur (utilisé dans le système JERRY) pour obtenir des plans valides, puis il introduit des notions de logique temporelle formelle pour ajouter des contraintes de sûreté de fonctionnement que les plans candidats devront vérifier. Il donne ensuite des expressions pour valider et vérifier le plan en même temps, et une règle de génération prenant en compte les contraintes ajoutées.

2.3 Architecture

2.3.1 Principes généraux

- **Learning New Representations and Goals for Autonomous Robots**
  [Paquier & Chatila 2003] : *REPGOAL03*
  Cet article présente les premiers travaux sur une architecture de réseau neuronal capable d'acquérir de nouveaux objectifs et attitudes à partir d'un ensemble initial restreint. Ce réseau neuronal est un réseau pulsé, dont les relations entre éléments atomiques sont caractérisées par un poids excitateur et inhibiteur et un seuil d'activation qui évoluent en fonction de la situation. Les conséquences d'une action sont évaluées par un composant (qui va induire l'apprentissage), et le développement de nouveaux comportement est causé par l'évolution de l'environnement (qui va causer de nouvelles entrées sensorielles), l'affinement des perceptions sensorielles (qui va permettre de diversifier les réponses aux perceptions) et l'évolution du seuil d'activation des neurones (qui va causer l'activation de configurations peu souvent activées).

- **A Reactive Robot Architecture with Planning on Demand**
  [Ranganathan & Koenig 2003] : *REPLAN03*
  Cet article présente une architecture originale vis-à-vis de la navigation de robot autonome, qui combine planification et navigation réactive. Contrairement à d'autres approches qui soit s'appuient sur plusieurs types de comportements, soit mettent le même comportement à jour dynamiquement en fonction de la situation, cette architecture utilise un niveau réactif plus simple de réalisation et fonctionne selon trois modes. Dans le premier mode le niveau réactif dirige seul le robot, dans le second mode le planificateur intervient en mettant en place un point de passage (*waypoint*), enfin dans le troisième mode le planificateur prend totalement le contrôle de la navigation. Le passage d'un mode à un autre se fait en fonction de deux paramètres qui déterminent si le robot a des difficultés à atteindre sa destination. L'article présente également des expériences qui permettent de comparer les performances de leur approche avec celles d'autres approches existantes.

- **A New Task-Based Control Architecture for Personal Robots**
  [Kim et al. 2003b] : *SHA03*
  Cet article présente une architecture hybride délibérative/réactive pour le niveau décisionnel d'un système autonome, appelée SHA (*Supervised Hybrid Architecture*). Il commence par présenter rapidement plusieurs types possibles d'architectures hybrides avant de décrire SHA, qui est composée d'un superviseur principal et de plusieurs modules. Le super-

viseur est délibératif : un planificateur connecté aux modules à travers un arbitrateur ; chaque module est constitué d'un ou plusieurs éléments délibératifs et réactifs, et d'un arbitrateur qui choisit en fonction de la situation quel élément solliciter. L'article présente ensuite les principes de fonctionnement de l'architecture SHA, ainsi que deux expériences.

- **Flat-distributed network architecture (FDNet) for rescue robots**
  [Koji et al. 2003] : *FDNET03*
  Cet article présente l'architecture du niveau délibératif FDNet, développé pour être intégré à un robot autonome de sauvetage. Ce niveau délibératif est un réseau dynamique de "data" (qui représentent des données de capteurs ou des actions à exécuter, mais aussi des intentions du système) et de "relations" reliant ces data (des morceaux de code qui par exemple créent de nouveaux datas à partir de datas existants, comme une nouvelle position à partir d'une position courante et d'une intention de déplacement). L'article présente ensuite l'implémentation de ce FDNet à l'aide de JAVA, CORBA et LinuxRT sur une plate-forme de robot de sauvetage. Il décrit finalement la conception de modules nécessaires au niveau exécutif du système autonome et leur interface avec le niveau délibératif FDNet.

- **Tripodal Schematic Design of the Control Architecture for the Service Robot PSR**
  [Kim et al. 2003a] : *TRIPSR03*
  Cet article présente l'architecture de système autonome PSR (pour *Public Robot Service*) et sa stratégie d'implémentation. C'est une architecture décrite à travers trois diagrammes : un diagramme des niveaux fonctionnels (*deliberative*, *sequencing*, *reactive*), un diagramme de classe et un diagramme de configuration (qui contient une décomposition des tâches du robot, et leur représentation en réseau de Pétri). Ces trois diagrammes spécifient complètement l'implémentation du robot (respectivement l'arrangement et les liens entre les composants, les fonctionnalités qu'ils fournissent, et le déroulement de ces fonctionnalités). Le niveau *deliberative* est réalisé par un planificateur, et le superviseur exécute séquentiellement chacune des tâches du plan. Une expérience sur une implémentation de cette architecture est ensuite rapidement présentée.

2.3.2 Architecture LAAS

- **Architecture and Tools for Autonomy in Space**
  [Ghallab et al. 2001] : *LAAS-SPACE01*
  Cet article présente l'architecture LAAS développée pour des robots autonomes, en considérant son application dans le domaine spatial ; c'est

une architecture à trois niveaux (fonctionnel, exécutif et décisionnel) et cet article s'intéresse en particulier aux niveaux exécutif et décisionnel. Le niveau exécutif s'exécute en temps réel dur ; il sert d'interface entre le niveau fonctionnel synchrone, et le niveau décisionnel asynchrone ; il gère les ressources et vérifie la consistance de l'état du système. Les spécifications de ce niveau exécutif sont écrites dans le langage de règles Kheops, d'après la description formelle du niveau fonctionnel du système et un modèle des états acceptables du système. Le niveau décisionnel est réalisé par un système superviseur/planificateur, et l'article décrit plus particulièrement le planificateur IxTeT, basé sur une approche de satisfaction de contraintes ; IxTeT propose entre autres une représentationdu temps par différentes contraintes métriques entre points temporels, la gestion de différents types de ressources (consommable, partagée, produite), un algorithme général assurant la complétude de la recherche de plan.

2.3.3 Architecture CLARAty

- **Decision-Making in a Robotic Architecture for Autonomy**
  [Estlin et al. 2001] : *CLARATY01*
  Cet article présente l'architecture de systèmes autonomes CLARAty ; cette architecture est composée de 2 niveaux : un niveau fonctionnel, et un niveau décisionnel (qui regroupe les niveaux exécutif et décisionnel d'une architecture trois niveaux classique). Ce niveau décisionnel unique a pour but d'éviter la gestion d'un modèle du système différent pour chaque niveau, et de permettre un meilleur contrôle des domaines décisionnels concurrents réactif/planification. L'implémentation CLARAty présentée dans cet article utilise le planificateur CASPER et l'exécutif TDL. Le niveau fonctionnel est conçu suivant une approche objet, pour traduire la modularité de la couche matérielle et permettre une granularité d'abstraction au niveau décisionnel. Enfin, un objet du niveau fonctionel peut utiliser un autre objet de ce niveau et l'état du système est gardé séparément dans chaque objet, qui sont interrogés par le niveau décisionnel quand il a besoin d'une information.

2.3.4 Environnements de programmation associés à des architectures spécifiques

- **The ORCCAD Architecture**
  [Borrelly et al. 1998] : *ORCCAD98*
  Cet article présente l'environnement de programmation ORCCAD, développé pour servir d'interface entre le niveau délibératif et le niveau matériel d'un système robotisé. ORCCAD fait le lien entre ces deux

niveaux, respectivement discret et continu, à travers deux structures de programmation : la *robot-task* qui représente l'instanciation d'une action continue du matériel (par exemple le déplacement d'une position à une autre) et la *robot-procedure* qui est un assemblage de *robot-tasks* et de *robot-procedures* remplissant un objectif. Des mécanismes d'exceptions sont à la disposition de l'utilisateur pour traiter les erreurs. L'article décrit les outils de l'environnement ORCCAD pour la conception graphique, la vérification formelle ou en simulation, et la génération de code du niveau de commande réalisé avec ces deux types de structures. Il présente finalement un exemple d'application sur un système autonome sous-marin.

- **ControlShell : A Software Architecture for Complex Electromechanical Systems**
  [Schneider et al. 1998] : *CSHELL98*
  Cet article présente l'environnement de programmation ControlShell, développé pour faciliter la programmation de composants logiciels de systèmes complexes, particulièrement à travers la vitesse de développement, la visibilité de ses outils, et la réutilisabilité de ses structures. Un composant est implémenté de manière différente suivant son fonctionnement : un composant périodique est décrit par un assemblage de blocs interconnectés, et un composant séquentiel par une suite de transition entre états. Le code du composant est généré automatiquement, et peut être testé en simulation. L'article présente trois exemples d'application de l'environnement.

- **The Real-Time Motion Control Core of the Orocos Project**
  [Bruyninckx et al. 2003] : *OROCOS03*
  Cet article présente le projet OROCOS (*Open RObot COntrol Software*) qui cherche à offrir un canevas de structure logicielle portable et temps-réel dur pour les composants fonctionnels de systèmes robotiques. Cette structure est constituée de neuf composants : trois composants infrastructurels, complètement implantés, qui mettent en place la fonctionnalité temps-réel et supportent l'exécution des autres composants, et six composants fonctionnels à implanter. Parmi ces six composants fonctionnels, trois forment la boucle d'exécution que l'on retrouve souvent dans les systèmes autonomes : Générateur (décision), Commande (action), Observateur(observation). L'article présente ensuite un exemple d'application du projet OROCOS au problème de déplacement, qui utilise deux instances de cette structure logicielle (un niveau temps-réel dur et un niveau temps-réel mou), mais ne donne pas de résultats (le prototype n'étant pas terminé).

- **Specifying and Verifying Active Vision-Based Robotic Sys-**

tems with the SIGNAL Environment
[Marchand et al. 1998] : *SIGNAL98*
Cet article présente un environnement de programmation qui utilise le langage SIGNAL; il permet d'implanter les niveaux fonctionnels et exécutifs de robots basés sur la vision. Il décrit tout d'abord les caractéristiques requises pour ce type de systèmes, ainsi qu'un inventaire d'autres environnement de programmation existant. Il explique ensuite de quelle manière SIGNAL remplit les caractéristiques énoncées : une méthode de programmation par flux de données, une possibilité de programmation préemptive, et de vérification formelle. Les systèmes considérés sont limités à ceux basés sur la vision pour deux raisons : le langage SIGNAL ne gère pas les interruptions (ou autres signaux asynchrones), et ne traite pas le temps de manière dynamique. L'article présente ensuite une application : un robot dont la tâche est d'explorer et reconstituer des entassements d'objets cylindriques et polygonaux.

- **RoboDaemon - A device independent, network-oriented, modular mobile robot controller**
  [Dudek & Sim 2003] : *ROBDAEMON03*
  Cet article présente Robodaemon, un environnement de programmation pour des robots mobiles multi plate-formes qui a pour but de faciliter le développement de prototypes de composants logiciels. Il propose pour cela un niveau de programmation indépendant de l'architecture matérielle, ainsi que des fonctions de simulation, de contrôle et de commande. Il contient des pilotes pour des robots mobiles existant, et peut aussi être étendu par des plug-ins ou modules. L'article présente ensuite une courte description de l'utilisation de cet environnement.

2.3.5 Architectures spécifiques

- **Development of an Underwater Robot, ODIN-III**
  [Choi et al. 2003] : *ODIN03*
  Cet article présente l'URV (*underwater robotic vehicle*) ODIN-III. La partie logicielle de l'URV s'exécute sous le COTS Windows2000 avec RTX, et utilise le système de DLL pour implanter et tester plus facilement différents types d'algorithmes. L'URV est connecté à une station au sol, qui peut le commander directement ou simplement servir de station d'observation. L'architecture du système est composée de cinq niveaux (mais on peut considérer que les deux premiers font partie du même niveau fonctionnel, ce qui porterait le nombre a quatre) : un niveau matériel et un niveau de périphériques, un niveau exécutif, un niveau décisionnel, et un niveau de supervision qui contrôle la sûreté de fonctionnement du système d'après un certain nombre de règles et de

feedback. Pour le moment le niveau décisionnel et le niveau de supervision ne sont pas très développés. L'article présente ensuite quelques résultats expérimentaux vis-à-vis du planificateur de trajectoire.

- **Design of a Prototype Miniature Autonomous Underwater Vehicle**
  [Gadre et al. 2003] : *MOD003*
  Cet article présente un prototype d'AUV (*Autonomous Underwater Vehicle*) appelé MOD0, dont les objectifs sont : un faible coût de production et une petite taille, la maximisation de la portée et de la robustesse, de permettre une modularité, d'atteindre une profondeur de 200m, et de réaliser en équipe des tâches d'observation. Le prototype réalisé a une longueur de l'ordre de 80cm pour un coût de moins de 3000$, et doit régulièrement remonter à la surface pour déterminer sa localisation. L'article décrit en détail la structure matérielle du prototype (système de propulsion, composants électroniques, capteurs) avant de décrire les modifications apportées au prototype suivant.

- **Remote Operation with Supervised Autonomy (ROSA)**
  [Gillett et al. 2001] : *ROSA01*
  Cet article présente l'architecture ROSA qui propose un système dont l'autonomie est variable, pour ravitailler des satellites. Le niveau fonctionnel de cette architecture est composée d'une partie commande et d'une partie capteur ; le niveau exécutif met en place une hiérarchie de "comportements" réactifs : des fonctionalités nominales du système. Le niveau décisionel contient trois éléments : un moteur d'inférence qui traduit un script d'ordres de l'opérateur en succession de comportements à exécuter par le niveau exécutif, un superviseur de mission intelligent qui planifie les actions du système en fonction des missions fixées par l'opérateur, et un multiplexeur de commande, qui permet à l'opérateur de définir le mécanisme décisionnel à utiliser et de transmettre des scripts d'ordre ou de nouvelles taches à celui-ci. L'article présente ensuite l'environnement du projet ROSA : la capture puis libération d'un satellite pour son ravitaillement.

- **Risks evaluation and Failures Diagnosis for Autonomous tasks execution in Space**
  [Finzi et al. 2001] : *SITCALC01*
  Cet article présente un système de comande basé sur le Situation Calculus, permettant d'évaluer la probabilité de succés des actions du système, et de diagnostiquer leur défaillance. Le système est composé de trois niveaux : un superviseur qui choisit un des objectifs du système et le décompose en plusieurs tâches, un planificateur qui décompose la tâche choisie en une séquence d'action basique et les fait

exécuter, et un module de vision qui met à jour la base de données du système et la carte locale. Le fonctionnement du système est de : (1) choisir un objectif puis une tâche à exécuter, (2) vérifier son exécutabilité, déterminer les probabilités de succés et l'exécuter, (3) observer l'état de l'environnement résultant de cette exécution, le diagnostiquer et replanifier la tâche au besoin. L'étape (2) est réalisée en construisant l'arbre de toutes les actions possibles en considérant leurs défaillances, évaluer la probabilité de chacun des résultats à partir de probabilité définies à l'avance pour chaque action, et de choisir celle remplissant l'objectif avec la plus grande chance de succés. L'étape (3) est un diagnostique par comparaison entre les observations réalisées et chaque résultat possible de l'arbre de la tâche pour déterminer la succession d'exécution la plus probable qui a menée à la situation observée. L'article décrit ensuite en détail le langage utilisé par le système pour évaluer le succés d'une tâche et diagnostiquer une défaillance. Il présente enfin l'implémentation du module de vision, et le langage utilisé par le superviseur.

- **A New Breed of Explorer : Development of a Network of Mobile Robots for Space Exploration**
  [Barfoot et al. 2001] : *RISE01*
  Cet article présente le mini-rover autonome RISE, conçu selon une architecture *subsumptive* dans un cadre multi-agent pour l'exploration spatiale. Il décrit les différentes caractéristiques du robot, en commençant par ses capacités sensorielles : des capteurs infra-rouge et d'autres haptiques de type moustache). La navigation est mise en place en étudiant la position de trois repères, naturels ou artificiels, et par odométrie entre chacun de ces repérages. Les robots communiquent entre eux par fréquence radio pour connaitre la position de chacun. L'architecture logicielle est de type *subsumptive* : 14 comportements ordonnés par priorité. L'article présente quelques résultats, en simulation et en opération, et une discussion sur les avantages et défauts de ces robots.

- **Onboard Autonomy on the Three Corner Sat Mission**
  [Chien et al. 2001] : *3CS01*
  Cet article présente les composants relatifs à l'autonomie de la mission *Three Corner Sat*, prévue pour lancer trois satellites fin 2002, mais qui semble avoir été abandonnée. Trois composants sont particulièrement étudiés : SCL (*Spacecraft Command Language*), un exécuteur de script COTS, CASPER, un planificateur également utilisé dans l'architecture CLARAty, et SELMON, un système de contrôle et de détection des défaillances. SCL met en oeuvre le niveau exécutif de cette architecture,

générant des tâches et réagissant à des événements à travers des scripts de règles et de fonctions, et maintient une base de données sur l'état du système. CASPER est un logiciel de planification continue, qui à chaque appel modifie le plan précédent et l'étend jusqu'à un nouvel horizon, afin de minimiser le temps pris pour chaque planification. Un cylce de planification se déroule comme suit : modification des objectifs et de l'état courant du plan, propagation de ces modifications dans le plan, appel d'algorithmes de réparation pour résoudre les conflits et satisfaire les nouveaux objectifs. SELMON est supposé s'exécuter en complément de SCL pour enregistrer des informations sur l'état du système et détecter des anomalies à travers différentes techniques : chien de garde, comparaison entre les valeurs des capteurs et un modèle du système...

2.4 Algorithmes

- **Information-based Intelligent Unmanned Ground Vehicle Navigation**
  [Madhavan & Messina 2003] : *ENTROPY03*
  Cet article présente une métrique basée sur l'entropie pour évaluer la quantité et l'utilité de l'information que contient une image de capteur (caméra par exemple). Il décrit tout d'abord la notion d'entropie probabiliste et la métrique utilisée pour l'étude d'image ; il présente ensuite très brièvement l'architecture RCS de leur UGV (*Unmaned Ground Vehicle*) puis donne trois exemples possibles d'application de cette métrique : traitement de l'image (pour extraire l'information significative), localisation à l'aide de filtres de Kalman, et prédiction de l'utilité de l'information d'une image (par exemple pour suivre un objet en déplacement).

- **Rover Autonomy for Long Range Navigation and Science Data Acquisition on Planetary Surfaces**
  [Huntsberger et al. 2002] : *FIDO02*
  Cet article présente trois méthodes de navigation pour le rover autonome FIDO ; il décrit d'abord rapidement l'architecture matérielle et logicielle du rover, puis présente les 3 méthodes. Dans le *long range navigation*, le rover utilise deux types de cartes : une carte globale de 12 mètres de portée pour générer un chemin, et une carte locale d'une portée de un mètre pour éviter des obstacles ; ces deux cartes ne sont pas partagées, et la génération de la carte globale demande une vingtaine de minutes. Dans le *long range rendezvous*, le rover suit successivement quatre comportements différents en fonction de sa distance et de sa position par rapport à la plate-forme sur laquelle il doit

monter. Dans le *autonomous science target rendezvous*, le rover se déplace mètre par mètre pour corriger sa position et arriver exactement à une cible choisie par un opérateur au sol. L'article présente ensuite des résultats expérimentaux effectués en extérieur sur Terre.

- **Implementation and Evaluation of a Satisfaction/Altruism Based Architecture for Multi-Robot Systems**
  [Lucidarme et al. 2002] : *ALTRUISM02*
  Cet article présente une méthode pour mettre en place la coopération de systèmes multi-agents ; les agents considérés sont de type réactifs plutôt que délibératifs. L'article présente d'abord les concepts de la méthode proposée : la satisfaction personnelle d'un agent qui représente son progrès dans la tâche demandée, et la satisfaction interactive de l'agent qui est positive si l'agent demande de l'aide, et négative s'il est gêné ou en conflit avec un autre agent. Les agents vont chercher à s'approcher ou s'éloigner de l'agent dont la valeur absolue de satisfaction interactive est la plus élevée. L'article présente ensuitel'architecture matérielle et logicielle d'une implémentation de tels agents, et quelques applications théoriques et expérimentales.

- **Human Error Recovery for A Human/Robot Parts Conveyance System**
  [Yamada et al. 2002] : *HMM02*
  Cet article présente une méthode permettant à un robot-manipulateur industriel de détecter une erreur de commande provenant de son utilisateur. Cette méthode utilise un HMM (*Hidden Markov Model*) pour prévoir la trajectoire du robot ; en cas de changement brutal de direction, le robot juge statistiquement si l'intention de l'opérateur a changé, et calcule alors la nouvelle trajectoire. Le robot peut ainsi corriger ou avertir son opérateur lorsqu'il trouve une incohérence entre cette intention et la tâche demandée. L'article présente par exemple une expérience où des opérateurs doivent déplacer des barils numérotés sur différents sites en fonction de ces numéros ; en détectant un mauvais choix de site par l'opérateur, le robot permet de réduire le temps pris par ces opérations.

- **Planetary Rover Control as a Markov Decision Process**
  [Bernstein et al. 2001] : *ROVERMDP01*
  Cet article présente une méthode combinant un MDP (*Markov Decision Process*) et un algorithme d'apprentissage par renforcement pour guider le planificateur d'un rover autonome. Il décrit tout d'abord la modèlisation du rover employée, puis cherche à maximiser l'efficacité d'un plan d'une part par une heuristique, d'autre part par un algorithme d'apprentissage par renforcement. Il compare ensuite ces deux

approches : l'algorithme d'apprentissage obtenant de meilleurs résultats. L'article présente ensuite des points à préciser dans cette méthode, en particulier la simplification du modèle qui a été faite pour obtenir les résultats, et l'utilisation de meilleures politiques d'heuristique et d'apprentissage pour obtenir une meilleure validation de cette comparaison.

References

[Aiello et al. 2001] L. C. Aiello, A. Cesta, et al., Merging Planning and Verification Techniques for "Safe Planning" in Space Robotics, In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space*, St-Hubert, Quebec, 2001.

[Asama et al. 2003] H. Asama, M. Yano, et al., System Principle on Emergence of Mobiligence and Its Engineering Realization, In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1715–1720, Las Vegas, Nevada, 2003.

[Barfoot et al. 2001] T. D. Barfoot, E. J. P. Earon, & G. M. T. D'Eleuterio, A New Breed of Explorer : Development of a Network of Mobile Robots for Space Exploration, In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space*, St-Hubert, Quebec, 2001.

[Bererton & Khosla 2002] C. Bererton & P. Khosla, An Analysis of Cooperative Repair Capabilities in a Team of Robots, In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 476–482, Washington, DC, 2002.

[Bernstein et al. 2001] D. S. Bernstein, S. Zilberstein, et al., Planetary Rover Control as a Markov Decision Process, In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space*, St-Hubert, Quebec, 2001.

[Borrelly et al. 1998] J. J. Borrelly, E. Coste-Manière, et al., The ORCCAD Architecture, *The International Journal of Robotics Research*, 17(4):338–359, April 1998.

[Bruyninckx et al. 2003] H. Bruyninckx, P. Soetens, & B. Koninckx, The Real-Time Motion Control Core of the Orocos Project, In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, pages 797–802, Taipei, Taiwan, 2003.

[Carlson & Murphy 2003] J. Carlson & R. R. Murphy, Reliability Analysis of Mobile Robots, In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, pages 274–281, Taipei, Taiwan, 2003.

[Chien et al. 2001] S. Chien, B. Engelhardt, et al., Onboard Autonomy on the Three Corner Sat Mission, In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space*, St-Hubert, Quebec, 2001.

[Choi et al. 2003] H. T. Choi, A. Hanai, et al., Development of an Underwater Robot, ODIN-III, In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 836–841, Las Vegas, Nevada, 2003.

[Duchemin et al. 2004] G. Duchemin, P. Poignet, et al., Design and Manufacturing of Actuated Medical Robots : a Safety Point of View, *IEEE Magazine on Robotics & Automation*, 2004.

[Dudek & Sim 2003] G. Dudek & R. Sim, RoboDaemon - A device independent, network-oriented, modular mobile robot controller, In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, pages 3434–3440, Taipei, Taiwan, 2003, www.cim.mcgill.ca/ simra/publications/icra03.pdf.

[Essamé et al. 2000] D. Essamé, J. Arlat, & D. Powell, Tolerance aux fautes dans les systèmes critiques, Technical Report 00151, LAAS-CNRS, 2000.

[Estlin et al. 2001] T. Estlin, R. Volpe, et al., Decision-Making in a Robotic Architecture for Autonomy, In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space*, St-Hubert, Quebec, 2001.

[Finzi et al. 2001] A. Finzi, F. Pirri, et al., Risks evaluation and Failures Diagnosis for Autonomous tasks execution in Space, In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space*, St-Hubert, Quebec, 2001.

[Fraichard & Asama 2003] T. Fraichard & H. Asama, Inevitable Collision States A Step Towards Safer Robots ?, In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 388–393, Las Vegas, Nevada, 2003.

[Gadre et al. 2003] A. S. Gadre, J. J. Mach, et al., Design of a Prototype Miniature Autonomous Underwater Vehicle, In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 842–846, Las Vegas, Nevada, 2003.

[Gerkey & Matarić 2002] B. P. Gerkey & M. J. Matarić, Pusher-washer : An approach to fault-tolerant tightly-coupled robot coordination, In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 464–469, Washington, DC, 2002.

[Ghallab et al. 2001] M. Ghallab, F. Ingrand, et al., Architecture and Tools for Autonomy in Space, In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space*, St-Hubert, Quebec, 2001.

[Gillett et al. 2001] R. Gillett, M. Greenspan, et al., Remote Operation with Supervised autonomy (ROSA), In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space*, St-Hubert, Quebec, 2001.

[Horiuchi 2003] E. Horiuchi, Nonstop Update of Mission Critical Robotic Services, In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1469–1474, Las Vegas, Nevada, 2003.

[Huntsberger et al. 2002] T. Huntsberger, H. Aghazarian, et al., Rover Autonomy for Long Range Navigation and Science Data Acquisition on Planetary Surfaces, In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 3161–3168, Washington, DC, 2002.

[Jaganathan & Lin 2003] A. Jaganathan & Y. J. Lin, An Error Restraining Method for Accurate Freeform Surface Cutting, In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, pages 797–802, Taipei, Taiwan, 2003.

[Kim et al. 2003a] G. Kim, W. Chung, et al., Tripodal Schematic Design of the Control Architecture for the Service Robot PSR, In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, pages 2792–2797, Taipei, Taiwan, 2003a.

[Kim et al. 2003b] J. O. Kim, C. J. Im, et al., A New Task-Based Control Architecture for Personal Robots, In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1481–1486, Las Vegas, Nevada, 2003b.

[Klein 1991] P. Klein, The Safety-Bag Expert System in the Electronic Railway Interlocking System Elektra, *Expert Systems with Apllications*, 3:499–506, 1991.

[Koike et al. 2003] C. Koike, C. Pradalier, et al., Prospective Bayesian Programming Application for Collision Avoidance, In *Proceedings of the 2003*

IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 394–399, Las Vegas, Nevada, 2003.

[Koji et al. 2003] Y. Koji, A. Pujol, et al., Flat-distributed network architecture (FDNet) for rescue robot, In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2392–2397, Las Vegas, Nevada, 2003.

[Kortenkamp et al. 2002] D. Kortenkamp, R. Simmons, et al., A Suite of Tools for Debugging Distributed Autonomous Systems, In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 169–174, Washington, DC, 2002.

[Laprie et al. 1996] J. C. Laprie et al., *Guide de la Sûreté de Fonctionnement (2de édition)*. Cépaduès - Éditions, 1996, (ISBN 2-85428-341-4).

[Leuschen et al. 2002] M. L. Leuschen, J. R. Cavallaro, & I. D. Walker, Robotic Fault Detection Using Nonlinear Analytical Redundancy, In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 456–463, Washington, DC, 2002.

[Lucidarme et al. 2002] P. Lucidarme, O. Simonin, & A. Liégeois, Implementation and Evaluation of a Satisfaction/Altruism Based Architecture for Multi-Robot Systems, In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 1007–1012, Washington, DC, 2002.

[Madhavan & Messina 2003] R. Madhavan & E. Messina, Information-based Intelligent Unmanned Ground Vehicle Navigation, In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3485–3490, Las Vegas, Nevada, 2003.

[Marchand et al. 1998] E. Marchand, E. Rutten, et al., Specifying and Verifying Active Vision-Based Robotic Systems with the SIGNAL Environment, *The International Journal of Robotics Research*, 17(4):418–432, April 1998.

[Paquier & Chatila 2003] W. Paquier & R. Chatila, Learning New Representations and Goals for Autonomous Robots, In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, pages 803–808, Taipei, Taiwan, 2003.

[Ranganathan & Koenig 2003] A. Ranganathan & S. Koenig, A Reactive Robot Architecture with Planning on Demand, In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*,

pages 1462–1468, Las Vegas, Nevada, 2003, www.cc.gatech.edu/ai/robot-lab/online-publications/Ran-Koenig03.pdf.

[Schneider et al. 1998] S. A. Schneider, V. W. Chen, et al., ControlShell : A Software Architecture for Complex Electromecanical Systems, *The International Journal of Robotics Research*, 17(4):360–380, April 1998.

[Senda et al. 2003] K. Senda, T. Matsumoto, & Y. Okano, A Study toward Cognitive Action with Environment Recognition by a Learning Space Robot, In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, pages 797–802, Taipei, Taiwan, 2003.

[Tińos et al. 2002] R. Tińos, M. H. Terra, & M. Bergerman, Fault Tolerance in Cooperative Manipulators, In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 470–475, Washington, DC, 2002.

[Tomatis et al. 2003] N. Tomatis, G. Terrien, et al., Designing a Secure and Robust Mobile Interacting Robot for the Long Term, In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, pages 4246–4251, Taipei, Taiwan, 2003.

[Yamada et al. 2002] Y. Yamada, T. Morizono, et al., Human Error Recovery for A Human/Robot Parts Conveyance System, In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 2004–2009, Washington, DC, 2002.