UNIVERSITE TOULOUSE III-PAUL SABATIER
U.F.R. PHYSIQUE - CHIMIE - AUTOMATIQUE

THÈSE

en vue de l'obtention du

DOCTORAT DE L'UNIVERSITE DE TOULOUSE
DÉLIVRÉ PAR L'UNIVERSITÉ TOULOUSE III-PAUL SABATIER

Discipline SYSTÈMES INDUSTRIELS

présentée et soutenue

par

HERVÉ RESSENCOURT

17 septembre 2008

Diagnostic hors-ligne à base de modèles : Approche multi-modèle pour la génération automatique de séquences de tests Application au domaine de l'automobile

JURY

Louise TRAVÉ-MASSUYÈS	*Directrice de Recherche CNRS, HdR*	(directrice de thèse)
Jérôme THOMAS	*Docteur-Ingénieur ACTIA*	(co-encadrant)
Etienne CRAYE	*Directeur / Professeur EC Lille*	(rapporteur)
Sylviane GENTIL	*Professeure INPG*	(rapporteure)
Joaquim ARMENGOL LLOBET	*Professeur Université de Girone*	(examinateur)
Michel COMBACAU	*Professeur UPS*	(examinateur)
Christian DESMOULINS	*Directeur Général ACTIA*	(examinateur)
Xavier OLIVE	*Docteur-Ingénieur THALES Alenia Space*	(examinateur)

Remerciements

Ce travail a été réalisé dans le cadre d'une convention CIFRE entre le Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS (LAAS-CNRS) et la société ACTIA. A ce titre, je remercie les directions de ces deux établissements de m'avoir donné les moyens matériels et financiers pour mener à bien mes travaux de recherche.

Par ailleurs, je tiens à remercier mes deux encadrants de thèse :

- Madame Louise Travé-Massuyès, directrice de recherche au CNRS, responsable du groupe DISCO (Diagnostic, Surveillance et Conduite) du LAAS-CNRS et directrice scientifique du laboratoire commun AUTODIAG, qui m'a fait confiance tout au long de ma thèse. Louise m'a notamment appris à faire preuve de rigueur dans mes différentes analyses et contributions sur la modélisation et le diagnostic. Elle a su être présente à chaque fois que j'avais des questions et pour me donner son avis éclairé.
- Monsieur Jérôme Thomas, responsable de l'équipe recherche de la Business Group Diagnostic à ACTIA et directeur du laboratoire commun AUTODIAG qui m'a soutenu quotidiennement dans mes réflexions. Nos discussions ont été nombreuses, enrichissantes et déterminantes dans les déifférentes orientations qui ont été prises durant ma thèse.

Je voudrais exprimer ma profonde reconnaissance à Hervé Poulard, membre de l'équipe recherche à ACTIA, qui a très fortement contribué à la réalisation du prototype logiciel et au développement des différentes idées que nous avons eu. Son aide fut très précieuse. « Que le MBR soit avec toi Hervé ! ! »

Je remercie également mes rapporteurs de thèse :

- Madame Sylviane Gentil, professeure à l'Institut National Polytechnique de Grenoble qui a bien voulu consacrer du temps à la relecture de mon manuscrit et qui a manifesté un grand intérêt à l'égard de mon travail.
- Monsieur Etienne Craye, professeur et directeur à l'Ecole Centrale de Lille qui a bien voulu participer à l'évaluation de mon travail. Sa relecture détaillée de mon manuscrit m'a été profitable.

Un grand merci aux autres membres du jury qui ont également accepté d'examiner notre travail :

- Monsieur Michel Combacau, professeur à l'Université Paul Sabatier à Toulouse. Merci Michel d'avoir accepté de présider mon Jury et surtout de m'avoir permis de t'appeler « Monsieur le Président ». Nos discussions ont souvent été très constructives durant ces trois années de thèse.

– Monsieur Joaquin Armengol Llobet, professeur à l'université de Gérone (Espagne) qui a accepté de venir en France pour assister à ma soutenance.
– Monsieur Xavier Olive, docteur-ingénieur à Thales Alenia Space.
– Monsieur Christian Desmoulins, Directeur Général de la société ACTIA qui a également montré beaucoup d'intérêt aux travaux de l'équipe recherche à ACTIA. Son soutient et son implication étaient sans faille.

Je salue également les chercheurs du groupe DISCO du LAAS-CNRS, les membres de la Business Group Diagnostic de la société ACTIA ainsi que tous le membres du laboratoire commun Autodiag.

Arrive à présent le moment d'exprimer toute ma gratitude à l'enemble de mes amis qui ont toujours été là pour me soutenir : merci à Axel et Siegfried les deux autres p'tits loups de la « dream team recherche » à ACTIA ; merci à Pauline et Elodie mes co-burottes et pardon pour les insectes que j'ai laissé rentrer dans le bureau pendant mes longues soirées de rédaction ; merci à Yannick le breton australien devenu toulousain et qui fait de très bonnes crêpes ; merci à Mehdi, Fabien, Valbert, Xavier et Francesco mes compagnons de la blagounette pendant les pauses au LAAS ; merci Solen, Hanane, Gilles et toutes mes autres potines et potes du swing ; merci aux membres de r@ve2003 et à Laurence tous anciens de l'ENSEM ; merci aux « Amis de la Mer » pour ces belles plongées. J'oublie sans doute de nombreux amis, j'espère qu'ils ne m'en voudront pas.

La dernière partie de mes remerciements s'adresse à ma famille et particulièrement à mes parents qui m'ont toujours soutenu pendant mes études. Ma reconnaissance et mon affection leur sera éternelle car c'est grâce à eux que j'en suis arrivé à ce stade.

Table des matières

Introduction

Le diagnostic est de nos jours un élément clef pour accroître la productivité et la disponibilité des systèmes complexes. C'est un thème de recherche très actif dans le domaine de l'Automatique comme dans celui de l'Intelligence Artificielle, ainsi qu'au carrefour de ces domaines. Le problème qui nous préoccupe est lié à l'automobile et plus particulièrement à la réparation en garage qui, en raison de la part croissante de l'électronique et de la multiplication des fonctionnalités embarquées, ne peut plus se réaliser sans outils d'aide au diagnostic.

Depuis 20 ans, la société ACTIA est spécialiste dans la conception de tels outils. Actuellement les outils de diagnostic automobile conçus par ACTIA sont essentiellement basés sur des arbres de décision élaborés par des experts humains. Ces arbres sont construits à partir de documentations diverses existant chez les constructeurs (conception, fiabilité, AMDEC...). Cette thèse a été réalisée dans le cadre d'une convention CIFRE entre le LAAS-CNRS et la société ACTIA ainsi que dans le cadre du Laboratoire Commun Autodiag (LAAS, IRIT, ACTIA) dont l'objectif est de développer, au travers du projet MODE (Multiple knOwledge Diagnosis Engine), un outil prototype de diagnostic utilisant quatre méthodes afin d'exploiter le maximum de connaissances disponibles sur un véhicule à diagnostiquer et de maximiser le taux de couverture en termes de types de défauts pouvant apparaître sur ce type de système. Ces méthodes sont : le raisonnement à base de modèles débarqué (module MBR - Model Based Reasoning), le raisonnement à base de modèle embarqué (module DDP - Diagnostic Distribué Préventif), la reconnaissance de formes (module MBR) et la recherche d'informations à base d'ontologie (module OBIR - Ontology Based Information Retrieval).

Le travail de cette thèse concerne le module MBR du projet MODE et a consisté à proposer et à mettre en œuvre une méthode opérationnelle à base de modèles qui détermine les meilleures séquences de tests que doit réaliser le garagiste afin d'identifier un défaut sur un véhicule en garage. [Bayoudh *et al.*, 2007; Chandrasekaran, 1994a] Deux précédentes thèses [Faure, 2001; Olive, 2003] ont abouti à la méthode AGENDA (Automatic GENeration of DiAgnosis trees) qui fut développée pour résoudre le « Test Sequencing Problem » à partir d'une modélisation structurelle et comportementale de circuits résistifs intégrant des composants à plusieurs modes de fonctionnement. La solution proposée intégrait notamment des modèles de défauts ensemblistes et un algorithme de recherche heuristique de type AO* pour générer un arbre de diagnostic optimal. Cependant, les tests générés se limitaient à des mesures électriques et le cadre de modélisation supposait que le système pouvait être maintenu dans un mode de comportement stable pour réaliser une mesure alors que les systèmes

actuels font apparaître des comportements dynamiques séquentiels et intermittents.

Dans cette thèse, nous avons étendu l'applicabilité d'AGENDA dans un cadre multi-modèle afin de prendre en compte la complexité fonctionnelle des architectures embarquées actuelles. Notre nouvelle approche permet notamment de modéliser un système mécatronique complet et de relier des symptômes fonctionnels de haut niveau à un ensemble de défauts portant sur des composants matériels et/ou logiciels. La notion de test a également été redéfinie dans un cadre dynamique, avec des modalités de type « courbes de références » et « symptômes fonctionnels », et multi-modèle afin de prendre en compte l'ensemble des observations possibles en garage (codes défauts, oscilloscope, symptômes clients, observations perceptives...).

Pour le séquencement des tests, nous avons développé une stratégie de choix du prochain meilleur test qui est basée sur une heuristique locale que nous comparons à l'approche globale d'AGENDA. Ce nouveau type de séquencement permet notamment de réaliser un diagnostic interactif, autorisant plus de souplesse et d'initiative à l'opérateur humain puisqu'il peut accepter ou refuser le test proposé.

Le prototype logiciel réalisé a été testé avec succès sur la fonction essuyage arrière des véhicules « Peugeot 1007 », « Citroën C4 », « Fiat Grande Punto » et démontré à des constructeurs automobile.

Ce manuscrit est organisé selon trois parties :

- La première partie, composée de trois chapitres, présente un état de l'art de l'ensemble des problèmes qui sont traités dans cette thèse. Le chapitre 1 présente la problématique du diagnostic dans le domaine automobile. Le chapitre 2 positionne le diagnostic par rapport aux autres méthodes à base de modèles. Le chapitre 3 donne un état de l'art de la représentation hiérarchique de la connaissance pour le raisonnement de diagnostic.
- La seconde partie, composée également de trois chapitres présente notre approche de diagnostic. Le chapitre 4 propose une approche de représentation multi-modèle des systèmes mécatroniques embarqués mêlant composants logiciels et matériels dans le cas d'un fonctionnement nominal. Le chapitre 5 définit la notion de test dans un cadre multi-modèle et présente le processus de prédiction (simulation) du résultat de chaque test disponible sous l'occurrence de chaque mode de défaut anticipé. Le chapitre 6 présente l'étape de séquencement dynamiques des tests. Nous revenons également sur l'algorithme AO* auquel nous ajoutons deux heuristiques prenant en compte la notion de coût dynamique des tests.
- La troisième partie présente le prototype logiciel que nous avons développé dans le cadre de nos travaux ainsi que les résultats obtenus (chapitre 7). L'application est notamment basée sur la simulation numérique de systèmes décris avec le langage de modélisation orienté objet MODELICA. Enfin, nous concluons cette thèse (8) et proposons un certain nombre de perspectives.

Première partie

Formulation du problème de diagnostic et état de l'art

1 Contexte du diagnostic en garage

Dans ce chapitre, nous définissons les principales caractéristiques des systèmes mécatroniques embarqués tels qu'ils existent actuellement dans le domaine automobile. Nous montrons notamment que les architectures électriques et électroniques ont nettement évolué avec une part de plus en plus importante du composant logiciel dans le coût global d'un véhicule. Cette évolution a eu impact significatif sur la tâche de diagnostic car elle s'est accompagnée d'une réduction du nombre de points de mesure, en raison de la miniaturisation des composants, et d'un accroissement du nombre de causes de défaillances possibles pour de tels systèmes.

1.1 Architectures embarquées dans le domaine automobile

1.1.1 Définition d'un système mécatronique

Un système mécatronique est un ensemble complexe et structuré de composants mécaniques, électromécaniques, électroniques et informatiques en interaction permanente et assurant une ou plusieurs fonctions d'usage. L'intérêt de ce domaine d'ingénierie interdisciplinaire est de concevoir des systèmes automatiques puissants et de permettre le contrôle de systèmes hybrides complexes. En reprenant la définition du groupe de travail inter GdR AFSEC[1] un système mécatronique automobile se découpe en trois sous-système [Elloy, 2006] (cf. Figure 1.1) : une partie commande, une partie opérative et une Interface Homme-Machine.

La partie commande (PC)

La partie commande est le système de pilotage du système mécatronique ; il s'agit d'une intelligence embarquée à dominante électronique et informatique temps réel. Sa fonction principale est d'élaborer les signaux de commande des actionneurs à partir des signaux en-

[1]— Le groupe de travail AFSEC (Approches Formelles des Systèmes Embarqués Communicants) est un une action initiée par le GdR ASR (Architecture Système Réseau). L'idée est de mener une action de réflexion commune avec la communauté scientifique d'automatique du GdR MACS sur la conception des systèmes embarqués.

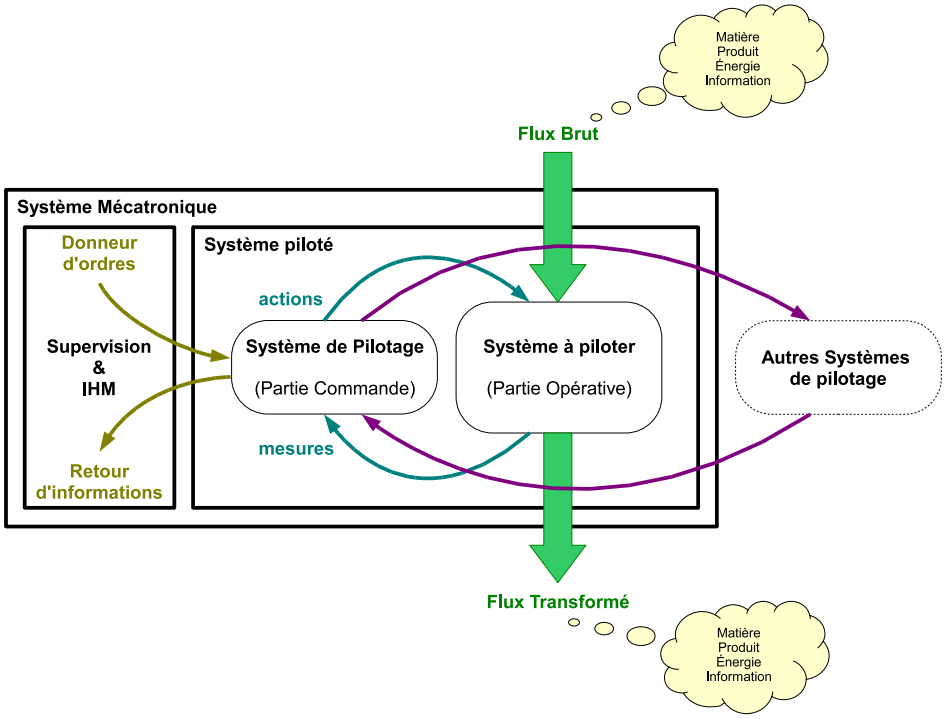


Figure 1.1 — Système mécatronique

voyés par les capteurs et d'informations provenant d'autres systèmes de pilotage. Alors que l'électronique était initialement réservée aux véhicules haut de gamme, elle est devenue un objet de consommation massive pour lequel les prestations clients augmentent considérablement. Pour le constructeur, cette évolution des technologies a été motivée par la mise sur le marché de véhicules offrant des fonctionnalités de plus en plus sophistiquées :

- le contrôle moteur : injection de carburant, allumage électronique, régulation de ralenti, anti-pollution,...
- le comportement routier : anti-bloquage de roues (ABS), suspensions actives, correcteurs de trajectoires (ESP), anti-patinage,...
- la sécurité active : coussins gonflables de sécurité (airbag),...
- le confort : climatisation, chauffage, régulation de vitesse, fermeture centralisée, anti-vol,...

La partie opérative (PO)

La partie opérative (PO) est le système piloté et est en général à dominante mécanique et électromécanique. On y retrouve quatre grandes classes de composants (cf. Figure 1.2) :

- *Les faisceaux électriques* : leur rôle est de relier tous les composants du système. Ils sont constitués d'un ensemble de fils électriques dont les extrémités sont reliées à des connecteurs. Les fils sont assemblés dans une gaine qui les protège des agressions du milieu extérieur. Ces composants peuvent être source d'un grand nombre de défauts, notamment au niveau des connecteurs : cosses mal serties (circuit-ouvert), oxydation des contacts (résistance parasite), humidité dans le connecteur (court-circuit). Il est

également possible de retrouver des défauts au niveau des fils provoqués par des frottements ou des contacts répétés avec le chassis (blessure du faisceau).

- *Les composants d'alimentation* : le rôle de ces composants est d'assurer l'alimentation électrique du système mécatronique ainsi que sa protection contre les risques de courts-circuits. Parmi ces composants, on trouve la batterie, le commutateur d'allumage-démarrage, les pré-actionneurs (relais électriques, commutateurs de puissance) et les fusibles. On retrouve également des défauts du type circuits ouverts (fusibles et relais) et du type faux-contacts.
- *Les capteurs* : leur rôle est de fournir un signal électrique qui est à l'image de la valeur d'un paramètre physique du système. Ces signaux peuvent être logiques, analogiques, périodiques ou apériodiques. La caractéristique entrée-sortie des capteurs est connue mais leur composition interne ne l'est pas toujours.
- *Les actionneurs* : les actionneurs les plus répandus sont les électrovannes et les moteurs électriques. Les signaux de commande varient suivant le type d'actionneur : ils peuvent correspondre à une simple activation du type « tout-ou-rien » ou à des commandes séquentielles (électrovanne papillon, injecteur...).

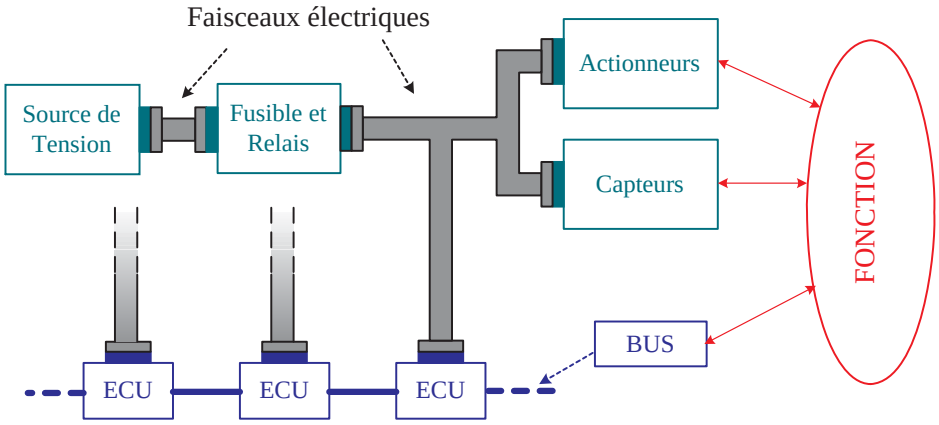


Figure 1.2 — Architecture générale des systèmes embarqués

1.1.2 Architecture multiplexée et complexité fonctionnelle

La complexité des systèmes embarqués est donc marquée par l'interaction entre composants matériels, qui mettent en jeu des phénomènes physiques multiples, et composants logiciels. Ces composants logiciels peuvent être des composants de contrôle (ECU) mais aussi de véritables réseaux informatiques multiplexés destinés à transférer de l'information entre calculateurs (bus CAN). Le multiplexage a apporté des modifications significatives dans la conception des systèmes mécatroniques :

- le partage des informations sur le bus de données a impliqué un partage de ressources entre fonctions du véhicule. Par exemple, certains calculateurs peuvent piloter plusieurs systèmes mécatroniques. Par ailleurs, la majorité des capteurs embarqués transmettent les signaux mesurés sur le bus. Il ne reste guère que les capteurs du type tout-ou-rien qui restent majoritairement en liaison filaire mais même ceux-ci tendent vers

le multiplexage. La Clio 3 et la Modus de Renault ont multiplexé les commandes au volant (comodos) avec une technologie du type « clavier d'ordinateur ».
- une même fonction peut être répartie sur plusieurs calculateurs électroniques. Par exemple, un simple système essuie-vitre avant implique parfois trois calculateurs : un qui gère l'état du comodo, un autre qui gère le comportement séquentiel du système et un troisième qui actionne le moteur électrique.
- la multiplication des fonctionnalités offertes à bord a conduit à des faisceaux électriques d'une très grande complexité qui est toutefois limitée par l'introduction des bus multiplexé qui ont été vus comme un moyen de simplifier l'architecture électrique et d'en augmenter la fiabilité.

Pour donner un ordre de grandeur chiffré de cette évolution, la figure 1.3 donne l'évolution du câblage en terme de nombre de points de connexion et de longueur de câble dans un véhicule. Par ailleurs, l'équipement électronique représentait en 2002 environ 22% du coût d'un véhicule contre 35% en 2007. La part logicielle représentait à elle seule 4,4% en 2002 contre 13% en 2007.

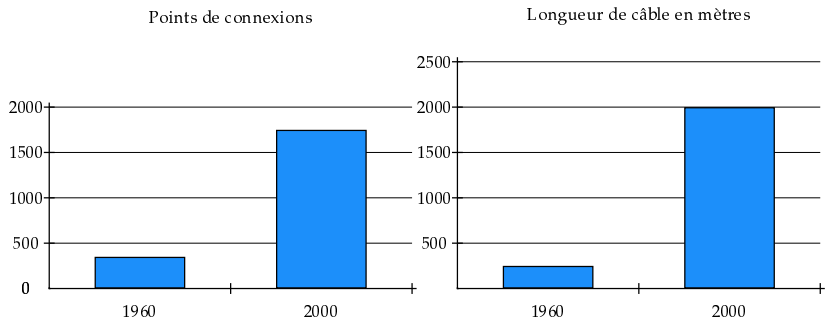


Figure 1.3 — Évolution du câblage

1.1.3 Complexité modale et sûreté de fonctionnement

D'un point de vue constructeur automobile, un système embarqué est un système critique et contraint. Ces contraintes peuvent être de plusieurs types : contraintes fonctionnelles, contraintes d'énergie, contraintes temps réel et contraintes de sûreté de fonctionnement. Concernant ce dernier point, le concepteur du système doit faire en sorte qu'il n'y ait pas de création de dynamiques non anticipées. Ainsi, les calculateurs sont dotés de fonctions d'*auto-diagnostic* et des *modes de fonctionnement dégradés* sont anticipés afin de continuer à fournir un service en cas de défaillance du système.

Auto-diagnostic des calculateurs : une fonction d'auto-diagnostic permet de détecter, de mémoriser, de signaler et de récupérer des défauts de fonctionnement. La détection de ces défauts est réalisée par des moyens matériels et logiciels. Cela consiste à détecter des courts-circuits ou des circuits ouverts à ses bornes pour la détection matérielle ou alors à détecter des incohérences par dépassement de seuil ou par une incompatibilité de la valeur avec le mode de fonctionnement courant pour la détection logicielle. Ces défauts sont ensuite

mémorisés par un «code défaut», et parfois signalés par un voyant sur le tableau de bord pour les défauts les plus graves.

Modes de fonctionnement dégradés : Lors de la phase de conception d'un système embarqué, l'ingénieur doit anticiper et spécifier plusieurs modes de fonctionnements du système. Ceux-ci se classent selon deux grandes catégories :

- les *modes d'utilisation* : ils correspondent aux fonctionnalités à assurer en tant que services spécifiés différents en fonction du contexte (ex : essuie-vitre avant en mode petite vitesse, en mode grande vitesse ou en mode automatique)
- les *modes dégradés* : si on reprend la définition de l'encyclopédie en ligne Wikipedia, fonctionner en mode dégradé, « c'est tenter de fournir le service jugé indispensable, en manquant de ressources complètes ou fiables ou régulières en énergie (dont électrique), en transport, télécommunication... ». Dans le cas pratique, ces modes dégradés peuvent correspondre à des services réduits mais acceptés (la fonctionnalité change) ou alors à des services maintenus mais moins sécurisés. Le cas le plus extrême du mode dégradé est le *mode de repli*, où l'objectif est de protéger le système en désactivant la fonction par exemple.

Notion d'applicabilité d'un véhicule L'*applicabilité* d'un véhicule est l'ensemble des options qui permettent de définir un véhicule particulier. Ainsi, un véhicule ne se définit plus seulement en terme de composants mécaniques qui le composent mais comme la réalisation d'un ensemble de *prestations* c'est à dire de services rendus à l'utilisateur par le biais d'une ou plusieurs fonctions :

- suivant le type de véhicule, des prestations sont fournies ou en option
- une prestation peut se décliner en plusieurs versions
- suivant le type de véhicule, une prestation possible est fournie dans une version imposée, ou choisie par le client parmi les versions proposées

Chez le constructeur, cette complexité grandissante remet en cause les processus de développement et, alors que les problèmes de maintenance et de service après vente n'étaient pas considérés comme des services prioritaires, il apparaît aujourd'hui clairement qu'ils doivent être pris en compte très en amont, et être complètement intégrés dans la chaîne de développement.

1.2 Le diagnostic dans le domaine automobile

Les métiers de la maintenance automobile sont également affectés en profondeur par le mutation « tout électrique » (cf. Figure 1.4). Dans les années soixante, le dépanneur pouvait intervenir sur un véhicule en panne en s'armant d'une simple caisse à outils. Actuellement, un équipement spécifique capable de se connecter au système informatique embarqué, appelé *outil de diagnostic*, pour en extraire les informations disponibles est nécessaire pour envisager un diagnostic de l'état du véhicule.

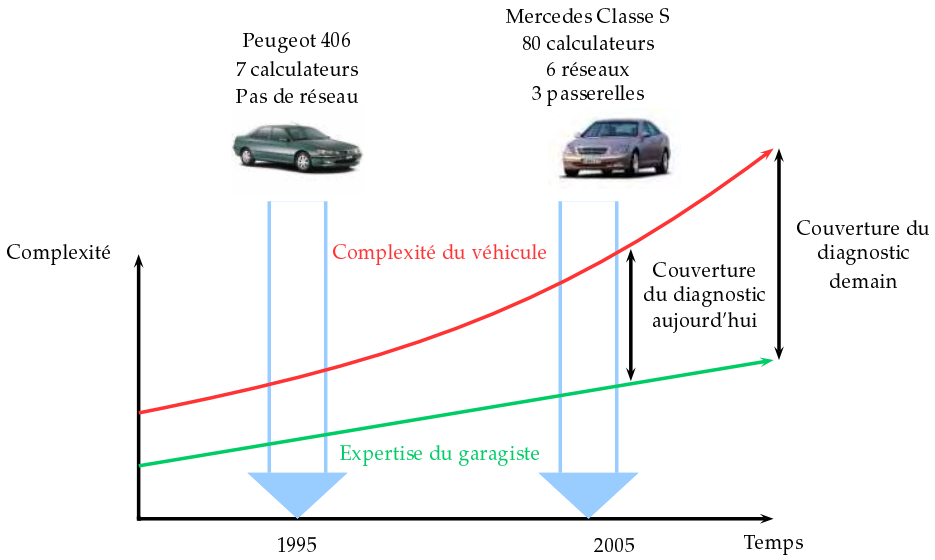


Figure 1.4 — Accroissement de la complexité du véhicule

1.2.1 Terminologie défaut/panne/symptôme

Afin de garder un discours le plus cohérent possible tout au long de ce manuscrit, nous définissons dans ce paragraphe les notions de défaut, panne et symptôme qui sont abondamment employés dans le domaine du diagnostic et de la maintenance mais souvent de manière confuse. Ces définitions ont été proposées par le comité technique SAFEPROCESS de l'IFAC (International Federation of Automatic Control) [Isermann et Ballé, 1997] :

- *défaut* (fault) : déviation non permise d'au moins une propriété ou un paramètre caractéristique du système des conditions acceptables ou(et) standards. On parle également de faute pour désigner un défaut.
- *défaillance* (failure) : interruption permanente de la capacité d'un système à assurer une fonction requise dans des conditions opérationnelles spécifiées. On utilise parfois le terme « panne » pour désigner une défaillance.
- *symptôme* : variation anormale d'une quantité observable. Par la suite, on est amener à parler de « *symptôme-client* » lorsque le comportement anormal est rapporté par le conducteur.

Les notions de défaillance et de défaut sont souvent considérées comme synonymes, notamment dans le discours industriel car elles sont proches. La différence à retenir est qu'un défaut se rattache à la notion de mode de comportement non désiré du système pour lequel on pourrait avoir un modèle alors que la notion de panne est liée à à la notion de fonction, c'est-à-dire qu'elle suppose l'existence d'une connaissance téléologique (connaissance relative à la finalité d'un système physique) du système physique auquel elle s'applique. Par la suite, on sera notamment amené à parler de « fonction défaillante » ou alors de « mode de défaut » d'un composant physique.

1.2.2 Les défauts/pannes dans le domaine automobile

Comme nous l'avons déjà vu dans le paragraphe 1.1, il existe plusieurs types de pannes :

- pannes électriques : elles sont essentiellement dues à des défauts de type courts-circuits ou circuits ouverts,...
- pannes électroniques : elles sont fortement liées aux pannes électriques et sont dues essentiellement à des défauts concernant les dispositifs électroniques (calculateurs, capteurs électroniques,...),
- pannes mécaniques : ruptures d'éléments, frottements excessifs,...
- pannes hydrauliques : elles sont essentiellement dues à des défauts de type fuite,...

Près de 80% des pannes constatées sur les véhicules actuels sont de type électrique ou électronique. Ces types de pannes sont surtout étudiés sur des sous-systèmes spécifiques mais les systèmes électroniques dans le véhicule se multipliant, il devient difficile de trouver des approches permettant une détection et/ou un diagnostic spécifique à chaque sous-système du véhicule. En outre, l'évolution de l'électronique embarquée s'accompagne de l'apparition de nouveaux défauts qui sont inconnus du garagiste ou des constructeurs. Certains défauts entraînent la défaillance de plusieurs fonctions du véhicule, donnant lieu à l'apparition de plusieurs symptômes, alors qu'un seul défaut sur un composant en est la cause. Ainsi, on peut retrouver les symptômes suivants dans des bases d'incidents connus :

- « la ventilation d'habitacle climatisation/chauffage et le dégivrage arrière ne fonctionnent plus »
- « les clignotants, les essuie-glaces et les lève vitres sont en panne »
- « le moteur cale par intermittence à l'enclenchement du compresseur de climatisation »
- « moteur tournant et climatisation enclenchée, lorsque l'on actionne le bouton de commande du pulseur d'air, les essuie-vitres avant et arrière se mettent à balayer lentement »

En outre, une difficulté majeure rencontrée aujourd'hui est l'apparition de défauts dits fugitifs ou intermittents, c'est à dire qui apparaissent dans une certaine situation d'utilisation du véhicule puis ne sont plus présents lors de l'arrivée du véhicule au garage et par conséquent leurs symptômes associés également. Ce type de défaut est très difficile à diagnostiquer pour les garagistes dans la mesure où ils ne sont pas reproductibles sur demande.

1.2.3 Les données accessibles en garage

Pour diagnostiquer les défauts présents sur un véhicule, le garagiste a un certain nombre de données et de moyens à sa disposition qu'on regroupe selon trois classes : la documentation technique, un accès à une base d'incidents connus et des moyens de test et de mesure.

La documentation technique

- *Contexte du véhicule* : le garagiste utilise le code VIN du véhicule et à travers un accès à la base de données « véhicule » du constructeur il récupère l'applicabilité du véhicule, c'est-à-dire toutes les informations contextuelles sur ce véhicule (type, motorisation,

options, date de fabrication, etc.)
- *Prestations du véhicule* : l'ensemble des prestations (« Visibilité », « Climatisation », « Signalisation »,...) est décrit par une décomposition de l'architecture du véhicule. Par exemple, la Figure 1.5 donne la décomposition hiérarchique de la prestation visibilité.
- *Diagrammes électriques* : le garagiste dispose en général des schémas électriques du véhicule. Ceux-ci peuvent provenir d'une documentation constructeur ou de revues techniques généralistes.

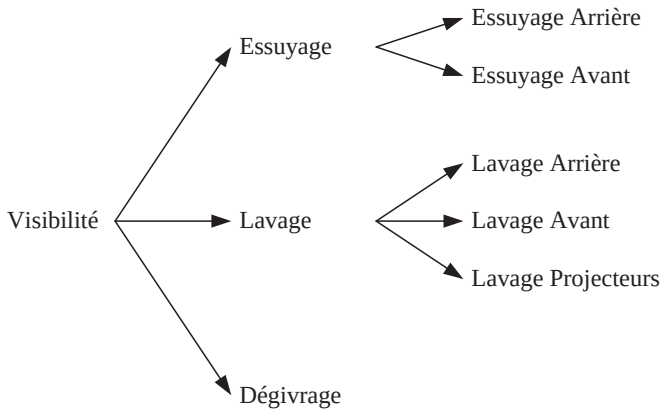


Figure 1.5 — Décomposition hiérarchique de la prestation visibilité

Les moyens de test et de mesure
- *Description des symptômes/effets client* : la première source d'information en vue d'établir un diagnostic est la description par le client des effets perceptibles du défaut présent sur le véhicule. Cette description est notée de manière textuelle dans une fiche remplie par le réceptionniste. Cette fiche est ensuite transmise au garagiste pour qu'il effectue son diagnostic.
- *Outil de diagnostic* : l'outil de diagnostic permet d'accéder à un certain nombre de données du calculateur et constitue donc un instrument de mesure pour réaliser des tests. Ces tests sont :
  - « *lecture des codes défauts* » (cf. section 1.1.3)
  - les « mesures paramètres » permettent de visualiser l'évolution des variables entrées / sorties du calculateur. Les valeurs prises par ces variables peuvent être booléennes (entrée / sortie du type tout-ou-rien) ou continues (capteurs de température, pression, courant, ...).
  - le *test actionneur* permet de piloter directement un sous-système connecté à une sortie du calculateur avec l'outil de diagnostic. Ainsi, il est possible de tester de manière isolée l'allumage d'un voyant du tableau de bord, des moteurs électriques ou des électro-vannes. En général, la procédure consiste à démarrer le test actionneur puis à effectuer une observation visuelle ou auditive (claquement de relais, de l'aiguille d'injection).
- *Mesures électriques* : l'instrument de mesure le plus courant est le multimètre mais de plus en plus de garages sont maintenant équipés de cartes de mesure permettant d'utiliser l'outil de diagnostic comme un oscilloscope. Elles permettent parfois de réaliser

des procédures de test spécifiques sur certain capteurs (sonde à oxygène). Ces mesures sont réalisées au niveau des connecteurs et sont coûteuses car les connecteurs sont difficilement accessibles et il est généralement indispensable d'utiliser des dispositifs de dérivation afin de reporter les points de potentiel électrique pour les rendre accessibles à la mesure.
– *Tests de référence* : la documentation technique du véhicule détaille parfois des valeurs ou des courbes de référence pour des mesures à réaliser localement sur un composant. Généralement, le garagiste devra exécuter la procédure de test à réaliser (en terme de dispositif de mesure à installer, et de configuration du véhicule) et comparer le résultat obtenu avec un résultat de référence.

Base d'incidents connus Le garagiste a accès à une base des fiches des incidents connus. Chaque fiche incident mentionne les conditions d'applicabilité (modèle, type, motorisation, date de fabrication, options, etc.), des symptômes clients observables sur le véhicule, un diagnostic et un mode opératoire permettant de réparer le véhicule.

1.3 Le projet MODE

Toutes les données, précédemment citées, sont utilisées de façon désordonnée sans réelle stratégie de diagnostic. Il convient donc de mettre en œuvre une méthode qui puisse prendre en compte un maximum d'information pour arriver à un diagnostic précis du système. Le diagnostic dans le domaine automobile pose ainsi une vraie problématique scientifique à laquelle se dédie le laboratoire commun AutoDiag qui réunit deux laboratoires toulousains, le LAAS-CNRS et l'IRIT, aux côtés de la société ACTIA [Thomas *et al.*, 2008].

L'objectif fixé est de réaliser une étude et le prototype d'un système de diagnostic utilisant tous les types de données accessibles. Ce système de diagnostic, nommé *MODE* (Multiple Objective Diagnosis Engine) (figure 1.3.1), doit faire collaborer plusieurs méthodes afin d'utiliser le maximum de connaissances disponibles sur le système et de maximiser le taux de couverture en termes de types de défauts pouvant apparaître sur le système.

1.3.1 Modèle du processus de diagnostic en garage

L'approche MODE est basée sur un processus de diagnostic générique [Travé-Massuyès *et al.*, 2004] représenté par un ensemble d'étapes qui caractérisent la session de diagnostic depuis la réception d'un véhicule jusqu'à sa restitution à un client.

1. *Réception véhicule* : dans cette étape, un véhicule est réceptionné suite à la détection d'« anomalie(s) ». Ces anomalies ont pu être constatées par le client (symptôme(s) client), ou il peut s'agir d'un véhicule qui présente des codes défaut. L'origine (symptôme ou défaut) conditionne les actions à mener.
2. *Pré-diagnostic de l'écart de prestation client* : cette étape permet de déterminer s'il s'agit d'une anomalie qui doit être traitée et si elle est d'origine mécanique ou électronique. En effet, dans certains cas, l'anomalie peut se révéler être un mode de fonctionnement

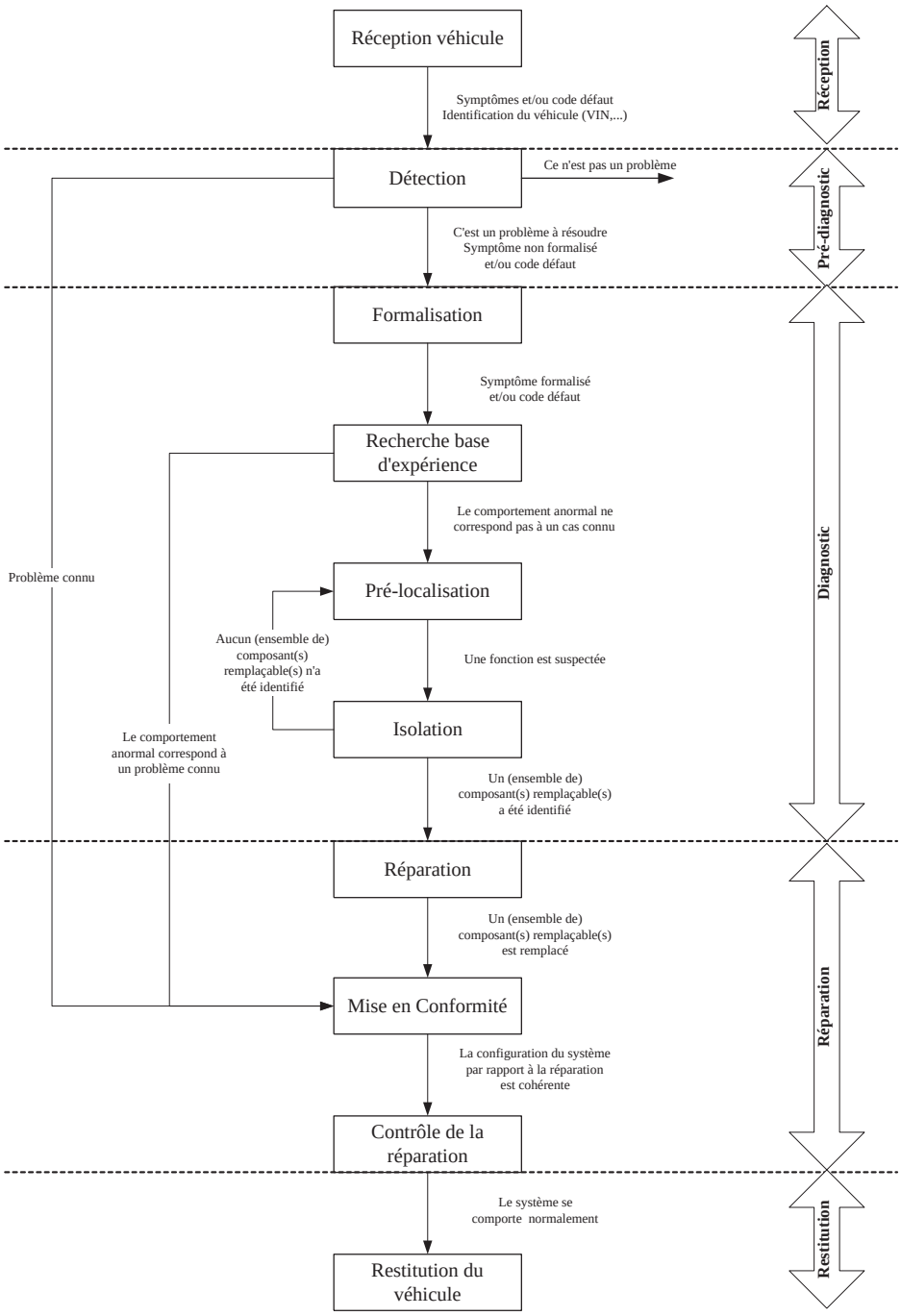


Figure 1.6 — Processus de diagnostic et de réparation d'un véhicule

normal ou normal dégradé qui ne nécessite pas obligatoirement une intervention. De même, lorsque le véhicule présente des défauts, il s'agit de déterminer si ces défauts doivent être traités ou non (cas de codes défaut de conception liés à la « configuration » du véhicule et qui n'ont aucune incidence sur son fonctionnement).

3. *Diagnostic* : cette étape consiste à localiser l'origine de la défaillance d'un système en terme de composants susceptibles d'être remplacés. Par composant, on entend aussi bien des composants électroniques et/ou physiques remplaçables que des logiciels em-

barqués.

4. *Réparation* : cette étape consiste à réparer et/ou remplacer les composants (au sens large) défaillants. On lui associe l'opération de contrôle de la réparation qui consiste à s'assurer que le(s) symptôme(s) a disparu ou que les défauts traités ne sont plus présents.
5. *Restitution véhicule* : il s'agit de l'étape de restitution du véhicule au client. Elle concerne l'écriture d'un compte-rendu des opérations, réparations et échanges de pièces effectués sur le véhicule pour informer le client et fournir les informations nécessaires à la facturation. Ces informations sont parfois consignées dans une fiche qui sera répertoriée dans la base d'incidents connus (cf. section 1.2.3).

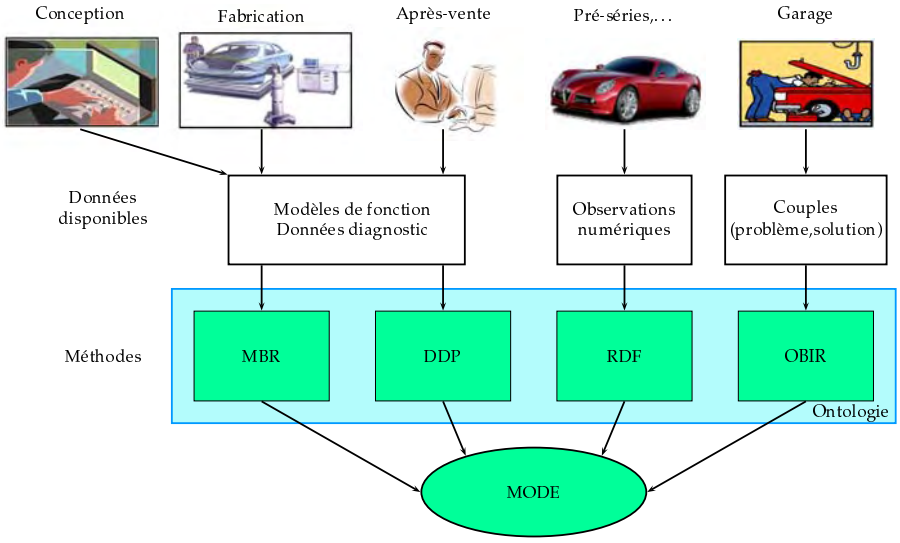


Figure 1.7 — L'approche MODE

1.3.2 Les différents projets de recherche

Le projet OBIR (Ontology Based Information Retrieval) [Reymonet *et al.*, 2006, 2007a,b] Ce module exploite les fiches incidents disponibles pour chaque type de véhicule, et par conséquent, les symptômes/effets clients, le contexte du véhicule, et la réparation. À partir de la description en langue naturelle des symptômes, l'outil doit retrouver un ensemble de fiches traitant d'un problème similaire sur le même type de véhicule. Un simple moteur de recherche basé sur une indexation classique par mots-clés aurait pu suffire mais, en l'absence d'une modélisation des connaissances du diagnostic automobile, il aurait été impossible de raisonner sur les objets du domaine. Un travail a été mené pour formaliser les connaissances du domaine sous le forme d'une ontologie qui permet de prendre en compte la synonymie entre les termes utilisés pour décrire un symptôme. L'ontologie est aussi employée pour une collaboration avec et entre les autres méthodes de raisonnement mises en œuvre au sein du logiciel d'aide au diagnostic.

Le projet RDF (Reconnaissance Des Formes) [Poulard, 1995, 1996; Khayati, 2003] L'objectif de ce projet est de mettre au point un outil de diagnostic utilisant des techniques de

reconnaissance de formes. Il vise donc à améliorer les capacités d'autodiagnostic des calculateurs en prenant en compte les corrélations entre paramètres mesurés par le calculateur. Cela revient à surveiller des paramètres prédéfinis et à diagnostiquer des défauts connus (ou inconnu si on sort simplement du bon fonctionnement). Le projet se focalise sur un sous-système spécifique comme le moteur dont le modèle physique n'est pas connu. Il utilise pour cela essentiellement des données issues de tests de roulage. Une phase d'apprentissage doit donc être effectuée au préalable.

Le projet DDP : Diagnostic Distribué Préventif [Soldani *et al.*, 2006, 2007a,b] Ce module a pour objectif de détecter les défauts intermittents en surveillant les messages qui circulent entre les ECUs au travers des bus multiplexés. Cette zone fait souvent l'objet de pannes difficilement explicables car elles n'ont pas été anticipées en conception. De plus, ces défauts ne sont plus présents lorsque le véhicule est réceptionné en garage et les conditions de leurs occurrences ne sont pas connues.

Le projet MBR (Model Based Reasoning) [Resencourt, 2006; Ressencourt *et al.*, 2006, 2008] Le module MBR fait l'objet des travaux de thèse présentés dans ce manuscrit. Il a pour objectif de localiser le composant remplaçable fautif une fois qu'une fonction suspecte a été localisée. Il utilise pour cela les modèles électriques et fonctionnels des systèmes mécatroniques pour proposer au garagiste une séquence de tests efficace en terme de coût et de probabilité d'occurrence des défauts. Il pourra également prendre en compte les symptômes/effets client ainsi que les codes défauts des calculateurs, les ensembles de fonctions ou encore les résultats mesures et les modèles de bon fonctionnement. Ce module se focalise essentiellement sur des défaillances persistantes.

Dans les outils de diagnostic actuels, cette étape de localisation se fait au moyen d'arbres de tests. Pour construire ces arbres, il faut envisager tous les défauts possibles du système, imaginer des tests qui les mettent en évidence et les organiser sous la forme d'un arbre de décision. Dans ce processus, on doit veiller à ce que l'arbre généré mène sûrement à la localisation des composants défectueux, tout en intégrant des données opérationnelles qui concernent notamment l'accessibilité aux points de mesure. Cette séquence doit être répétée pour chaque fonction du système de pilotage, pour chaque système de pilotage du véhicule et pour chaque véhicule de la gamme. D'autre part, des révisions fréquentes des ces arbres sont réalisées. Cette activité est donc difficile, fastidieuse et mobilise beaucoup de personnes. Ces équipes de développement ont besoin d'outils d'aide à la génération d'arbres de tests afin d'atténuer ces difficultés et de maîtriser les coûts de développement.

Un autre moyen actuellement utilisé pour générer un arbre est de construire une matrice appelée « Matrice Panne-Effet (MPE) ». Celle-ci permet de regrouper dans un tableau tous les défauts anticipés sur un système et leurs effets sur les tests. Un algorithme de génération d'arbre peut ensuite être exécuté en prenant la MPE en entrée. Cette matrice, qui ressemble finalement à une analyse AMDEC (Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité), est générée manuellement ; ce travail est peut-être moins fastidieux que la génération directe d'un arbre mais il reste très coûteux.

Depuis la dernière décennie, un attention particulière a été portée à l'utilisation des mo-

dèles pour le diagnostic automobile. Le groupe de travail Automotive Task Group [Monet2, 2003] du réseau d'excellence MONET2 est le principal exemple de collaboration européenne entre laboratoires de recherche et industriels de l'automobile. Ce qui ressort des réflexions menées, (auxquelles le LAAS-CNRS et la société ACTIA ont pris part) c'est le désir des constructeurs automobile d'avoir des outils de diagnostic avancés à base de modèles et d'intégrer au plus tôt dans le cycle de vie d'un véhicule les éléments nécessaires au processus de diagnostic, expliquant ainsi l'utilisation des AMDEC, de l'analyse de diagnosticabilité et de l'analyse des causes insidieuses (SCA - Sneak Circuit Analysis). Une feuilles de route fixant les directions vers lesquelles devront évoluer les futurs outils de diagnostic a notamment été délivrée en 2004 [Automotive, 2004]. Le travail de cette thèse se situe dans cette mouvance.

2 Diagnostic à base de modèle et diagnostic hors-ligne

Ce chapitre est consacré au problème du séquencement de tests et son positionnement par rapport aux travaux de la communauté du diagnostic à base de modèles. Nous reviendrons notamment sur la méthode AGENDA (Automatic GENeration of DiAgnosis trees) développée lors des précédents travaux de thèse réalisés au sein de la société ACTIA. Nous identifierons ses limites en ce qui concerne le diagnostic de systèmes mécatroniques complets tels qu'ils ont été définis dans le chapitre précédent. Enfin, nous énoncerons les objectifs de la brique MBR de MODE qui fait l'objet de cette thèse en nous appuyant notamment sur les résultats d'AGENDA.

2.1 Le Diagnostic à Base de Modèles

Pour un système donné, la tâche de diagnostic consiste à détecter un comportement défaillant du système et à localiser l'origine du défaut à partir des observations disponibles. Ce processus peut se découper en trois étapes :

1. *Détection de défauts* : identification d'un comportement du système ne correspondant pas à celui qui était attendu. Cette étape donne lieu à la description de symptômes rendant compte de cette observation et indique la présence d'un ou plusieurs défaut(s) dans le système.

2. *Génération d'hypothèses de diagnostic* : à partir des symptômes disponibles, il est possible de déduire l'ensemble des défauts possibles (hypothèses de diagnostic) pouvant être à l'origine du comportement défaillant du système.

3. *Sélection d'observations* : analyse des hypothèses de diagnostic afin d'identifier les observations supplémentaires nécessaires pour discriminer l'ensemble des hypothèse de diagnostic. Ces observations, une fois réalisées, donnent lieu à de nouveaux symptômes.

Ces étapes sont à rapprocher des étapes 2 et 3 du processus de diagnostic en garage de l'approche MODE (cf. section 1.3.1). Le diagnostic est dit à base de modèles si les modèles du système sont disponibles et utilisés dans le processus de raisonnement.

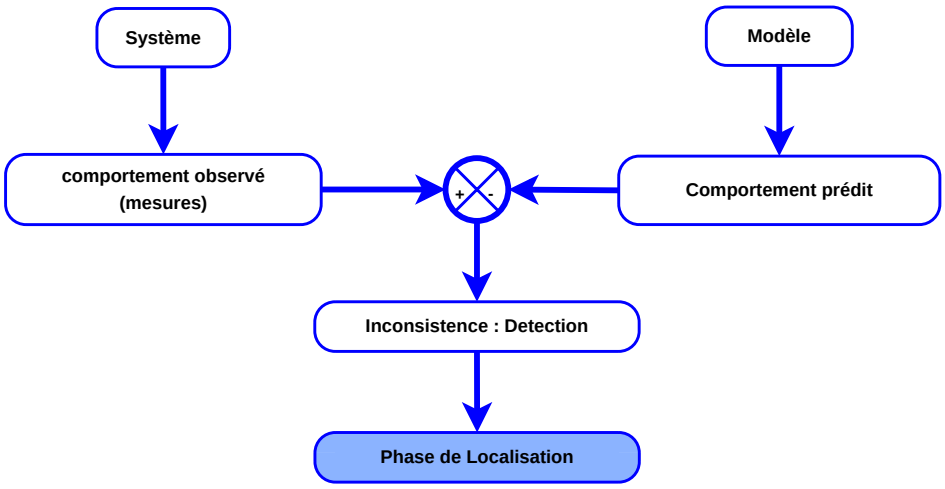


Figure 2.1 — Principe du diagnostic à base de modèles

2.2 Les deux grandes approches du diagnostic à base de modèles

Les approches de diagnostic à base de modèles (DBM) exploitent un modèle de fonctionnement explicite du système à diagnostiquer. Elles fondent la détection de défaut (cf. Figure 2.1) sur la constatation d'incohérences entre le comportement prédit par le modèle et le comportement réellement observé, et leur localisation en recoupant les groupes de composants impliqués dans chacune de ces incohérences.

Deux communautés de recherche distinctes se sont positionnées suivant l'approche DBM avec des différences importantes sur la démarche, les modèles, les algorithmes utilisés et l'interprétation des résultats : la communauté DX (approche de l'intelligence artificielle) et la communauté FDI (approche automaticienne). Dans le passé, il y a eu peu d'échanges entre ces deux communautés, mais plus récemment, un nombre croissant de chercheurs entreprirent de déterminer une terminologie commune, d'identifier les similarités et les complémentarités des approches de part et d'autre et de concevoir des méthodes de diagnostic plus performantes en exploitant les synergies des techniques issues des deux communautés [Cordier *et al.*, 2004].

2.2.1 La théorie logique du diagnostic de DX

La théorie du diagnostic logique à base de modèles, a vu le jour au milieu des années 70 et a été formalisée au début des années 80. Un nombre croissant de travaux ont été menés depuis et cette problématique est devenue un domaine de recherches à part entière de l'Intelligence Artificielle (I.A.). Les articles les plus marquants jusqu'à 1991 sont regroupés dans [Hamscher *et al.*, 1992]. Une excellente synthèse de ce domaine est également faite dans [Dague, 2001] que nous résumons dans cette section.

Modèles utilisés Les connaissances incluses dans les modèles utilisés décrivent la structure (connexions entre composants) du système à diagnostiquer et le comportement des différents composants intervenant dans le système. Seules les connaissances structurelles sont

spécifiques au système en question ; les connaissances comportementales, idéalement issues directement des lois de la physique, sont génériques et ne dépendent que du domaine (électronique, thermodynamique, hydraulique, mécanique, etc.). Ces connaissances sont donc a priori réutilisables pour tout système du domaine décomposable en composants génériques. Elles sont exprimées en toute généralité sous forme de contraintes constituant une bibliothèque de modèles des composants (la notion de composant dépend de la granularité souhaitée du diagnostic et de la réutilisabilité des modèles pour d'autres systèmes).

Ce qui distingue cette approche du diagnostic est qu'il n'est pas nécessaire d'anticiper les défauts ou dysfonctionnements pouvant affecter un système et leurs effets pour pouvoir les détecter et les localiser : modéliser le comportement correct est suffisant. L'idée fondamentale est de comparer le comportement réel du système tel qu'il peut être observé par l'intermédiaire des capteurs installés et son comportement attendu tel qu'il est prédit grâce aux modèles de bon comportement. Toute incohérence entre les observations et les prédictions déduites des modèles est interprétée comme la manifestation d'un dysfonctionnement, i.e. la présence d'un ou plusieurs défauts. Ceci est bien sûr conditionné par le fait que les modèles soient corrects, c'est à dire qu'ils représentent réellement le comportement du système.

Conflits et diagnostics Les contradictions entre observations et prédictions mettent en évidence la présence de défauts, et renseignent sur leur localisation. C'est principalement dans ce but que DX les utilise. On utilise pour cela la trace des prédictions qui ont mené aux incohérences : si une prédiction a été faite en utilisant les modèles de bon comportement des composants $C1, ..., Cn$, et qu'elle est incohérente avec une observation, c'est que les composants $C1, ..., Cn$ ne peuvent être tous corrects. Ainsi l'un au moins d'entre eux est nécessairement défectueux : on dit que ces composants forment un *conflit* [Reiter, 1987]. Plus on accumule ou l'on utilise d'observations et plus on effectue de prédictions, plus on a de chances d'obtenir des incohérences, donc de nouveaux conflits. En recoupant ces conflits, on affine progressivement la localisation du (ou des) défaut(s).

La seconde phase est de générer des hypothèses (sur le fonctionnement correct ou non des composants) qui rendent compte de tous les conflits. En termes logiques, cela signifie de changer l'hypothèse de fonctionnement correct de certains composants de façon à ce que toutes les incohérences disparaissent, i.e. qu'il n'y ait plus de conflit. Un ensemble de composants qui, cessant d'être supposés corrects, tous les autres le demeurant, rétablit ainsi la cohérence avec les observations est précisément appelé un *diagnostic*. C'est pourquoi on parle aussi de diagnostic à base de cohérence. Pour une situation donnée, il y a généralement plusieurs diagnostics. Les diagnostics de cardinal un (singletons) correspondent aux défauts simples, ceux de cardinal supérieur aux défauts multiples. La théorie du diagnostic logique traite ainsi dans un même cadre les défauts uniques et multiples, ce que ne font pas les autres approches. En vertu du principe de parcimonie, on s'intéressera tout particulièrement aux diagnostic minimaux au sens de l'inclusion ensembliste.

Modèle de défaut Malgré le bénéfice de pouvoir arriver à des résultats de diagnostic à partir du seul modèle de comportement nominal d'un système, une connaissance sur les

défauts permet d'enrichir le diagnostic, en permettant notamment d'aller au delà de la localisation et d'identifier les défauts. Aussi, des extensions du formalisme furent proposées pour permettre d'utiliser des modèles de défaut : les moteurs de diagnostic GDE+ [Struss et Dressler, 1989] et Sherlock [de Kleer et Williams, 1989] sont les travaux les plus marquants. Cependant, contrairement aux autres approches utilisant des modèles de défaut, la validité logique du diagnostic établi demeure garantie.

Aux deux modes de comportement correct et incorrect utilisés précédemment, on vient ajouter les modèles des modes de défauts connus, considérés deux à deux exclusifs, ainsi qu'un mode Inconnu sans modèle associé qui représente tous les modes de défaut non anticipés. Si l'on appelle OK le mode correct, F_i les modes défectueux et I le mode Inconnu, un composant C aura les modes suivants :

$$\{OK(C), F_1(C), ..., F_k(C), I(C)\}$$

Un conflit correspond dans ce cadre à une assignation de modes comportementaux à certains composants incohérents avec les observations. Un diagnostic est alors une assignation de modes comportementaux pour tous les composants du système qui rétablit ainsi la cohérence avec les observations. Les diagnostics ne sont plus caractérisés comme sur-ensembles des diagnostics minimaux mais conservent une caractérisation logique en terme de conflits [de Kleer *et al.*, 1992a].

2.2.2 L'approche FDI

La communauté connue sous le nom de FDI (Fault Detection and Isolation) utilise des techniques provenant de l'automatique et de la décision statistique. Elle a maintenant atteint un stade de maturité et un grand nombre de résultats existent dans ce domaine [Patton et Chen, 1991; Frank, 1996; Gertler, 1998; Kinnaert, 2003]. L'ouvrage [Dubuisson, 2001] constitue une excellente synthèse de ce domaine et l'intégration du diagnostic et de la supervision dans une démarche de contrôle est exposée dans [Maquin *et al.*, 2007]. Les travaux des automaticiens se basent sur l'utilisation de modèles analytiques numériques et abordent le problème par la tâche de détection en se proposant de générer des *indicateurs de défauts*, appelés « *résidus* » [Maquin *et al.*, 1997; Ragot et Maquin, 2001]. La théorie est bien assise en ce qui concerne les modèles linéaires, bien que des résultats plus récents aient été obtenus pour des modèles non linéaires [Frank, 1987; Staroswiecki et Declerck, 1989] et les système hybrides [Cocquempot *et al.*, 2005; Bayoudh *et al.*, 2008a].

Notion de résidu Un résidu est l'expression d'une redondance introduite soit par le système de mesure, soit par des modèles. Par exemple si l'on dispose de deux capteurs pour mesurer la même variable, alors un résidu est obtenu en comparant la valeur fournie par les deux capteurs. De manière plus générale, si l'on a plusieurs chemins de calcul pour obtenir l'estimation d'une même variable, alors on a autant de résidus. Ceci soulève bien sûr le problème de l'instrumentation nécessaire du système, autrement dit du nombre et du choix du positionnement des capteurs, conduisant à la diagnosticabilité [Travé-Massuyès *et al.*, 2006; Bayoudh *et al.*, 2006, 2007; Basseville *et al.*, 1987; Nyberg, 2002].

La valeur d'un résidu est nulle dans le cas nominal et non nulle en cas de défaut. L'expression des résidus est obtenue a priori, comme une fonction des grandeurs observables (phase de calcul des résidus). Elle est ensuite évaluée en utilisant les mesures (phase d'évaluation des résidus). Une procédure de décision, prenant en compte les caractéristiques statistiques des bruits de mesure et les erreurs de modélisation est en général nécessaire pour évaluer les résidus, c'est-à-dire décider si leur valeur est nulle ou pas.

Localisation de défauts Parmi les approches existantes pour la localisation, nous pouvons citer les résidus structurés et les résidus directionnels.

- Les *résidus structurés* sont conçus pour que chaque résidu soit sensible à un sous-ensemble de défauts et insensible aux autres. Le *support* d'une relation de redondance sera ainsi défini par l'ensemble des défauts dont l'occurrence est susceptible d'influer sur son résidu associé. Inversement, quand un défaut survient, certains résidus répondent et d'autres restent à zéro : ceci constitue la *signature du défaut*. L'ensemble des signatures prédéfinies pour les différents défauts à considérer constitue la *matrice de signatures*.
- Les *résidus directionnels* sont conçus de telle sorte que, lorsqu'un défaut survient, le vecteur de résidus soit confiné suivant une direction particulière de l'espace des résidus. En pratique, on va tâcher de structurer ou d'orienter au mieux un ensemble de résidus en minimisant/maximisant la sensibilité des résidus par rapport à divers sous-ensembles de défauts.

Hypothèse de défaut unique En partant du principe que l'apparition de défauts simultanés est généralement de probabilité faible, l'approche FDI adopte souvent l'hypothèse de défaut unique. Pour traiter le cas de défauts multiples, il faut élaborer une nouvelle signature qui est construite selon l'idée qu'un défaut multiple peut affecter un résidu si et seulement si au moins l'un des défauts simples constituant le défaut multiple peut affecter l'équation de redondance associée à ce résidu. Ceci est une différence fondamentale avec l'approche DX qui n'a aucune limitation sur le nombre de défauts simultanés.

2.3 Diagnostic post-mortem : problème du test

2.3.1 Diagnostic en-ligne versus Diagnostic hors-ligne

Indépendamment de l'approche utilisée (DX ou FDI), deux types de diagnostic doivent être distingués en fonction du contexte dans lequel le système doit être diagnostiqué : le diagnostic en-ligne et le diagnostic hors-ligne.

Le diagnostic en-ligne Dans le diagnostic en-ligne, on est dans une approche de supervision dans laquelle il faut effectuer le suivi temporel d'un processus évolutif. Ce diagnostic se fait sur un système en cours de fonctionnement sans connaissance a priori de la présence d'un défaut ou non. De plus, l'ensemble des observations disponibles est fixé et il est rarement possible de procéder à des tests. Ainsi, le processus de diagnostic est exécuté pour

toutes observations qui ne sont pas consistantes avec les prédictions obtenues à partir du comportement nominal du système. Une étape de localisation (pouvant intégrer des modèles de défauts anticipés) peut aussi être réalisée en ligne mais celle-ci est limitée par le nombre de capteurs disponibles qui ne garantie pas toujours une discrimination totale de l'ensemble des défauts.

Le diagnostic hors-ligne Dans le diagnostic hors-ligne, on ne se préoccupe que de la localisation (ou identification) du défaut ; c'est le problème du test qui est prépondérant. Il s'agit de déterminer l'information additionnelle qui permettra de discriminer au mieux, et à coût minimal, les hypothèses de diagnostic obtenues avec les symptômes déjà présents. Ce cadre sous-entend de démarrer la session de diagnostic avec un ensemble réduit de mesures et de pouvoir obtenir des mesures supplémentaires pour affiner la localisation du composant défaillant.

Le problème du test pour le diagnostic à base de modèle est traité aussi bien dans la communauté DX [de Kleer et Williams, 1987; McIlraith et Reiter, 1992; McIlraith et Scherl, 2000] que dans la communauté FDI [Pattipati et Alexandridis, 1990; Pattipati et Dontamsetty, 1992; Narasimhan *et al.*, 1998]. Celui-ci peut se formuler selon un *diagnostic séquentiel* hors-ligne lorsque l'objectif est de déterminer de nouvelles observations à réaliser sans agir sur les entrées du système. Parmi les techniques développées, celle du choix optimal de la prochaine mesure de l'approche GDE (General Diagnosis Engine) proposée par De Kleer [de Kleer et Williams, 1987] est la plus couramment utilisée. Elle vise à choisir comme prochaine meilleure mesure celle qui minimise l'entropie des probabilités des différents diagnostics pouvant résulter de cette mesure en fonction du contexte courant. On trouvera également d'autres techniques basée sur des critères de discrimination alternatifs [Singh, 1992; Mosterman *et al.*, 1997; Narasimhan *et al.*, 1998].

Dans le cas des systèmes hybrides, un défaut ne se manifeste pas nécessairement dans tous les modes d'utilisation du système. Par ailleurs, si le système est dans un mode dans lequel ce défaut se manifeste, les observations disponibles dans ce mode peuvent se révéler insuffisantes pour le localiser. Certaines approches dites de diagnostic actif proposent alors, si cela est possible, de changer les entrées du système afin de le placer dans un nouveau mode d'utilisation et faire ensuite des observations dans ce mode. Le «Test Sequencing Problem» (problème de séquencement des tests) présenté ci-après rentre dans ce cadre.

2.3.2 Problème de séquencement des tests

Le problème de séquencement des tests (TSP - Test Sequencing Problem) pose le problème du diagnostic hors-ligne lorsque les défauts et les tests sont anticipés et en faisant l'hypothèse de défaut unique. Dans cette approche de diagnostic, dite à base de *dictionnaire de défauts*, l'objectif est de rechercher les séquences de tests optimales, sous la forme d'un arbre de diagnostic, permettant de localiser chacun des défauts en considérant toutes les éventualités pour la valeur des tests. Le problème de séquencement des tests a été initialement défini dans [Pattipati et Dontamsetty, 1992] pour des test binaires[1] puis étendu dans

[1] Chaque test possède seulement deux résultats possibles, 0 ou 1

le cadre de la méthode AGENDA pour des tests multivalués [Faure *et al.*, 1999; Faure, 2001; Olive, 2003] :

- Un ensemble fini $\mathcal{F}$ de $n_F + 1$ défauts , $\mathcal{F} = \{f_0, ..., f_{n_F}\}$, dans lequel f_0 se rapporte par convention au comportement nominal et $f_i (1 \leq i \leq n_F)$ constituent les défauts possibles du système.
- Un ensemble $\mathcal{P}$ de $n_F + 1$ probabilités d'occurrence a priori p_i avec $i \in \{0, ..., n_F\}$ tel que $\sum_{i=0}^{n_F} p_i = 1$. Nous supposons que les probabilités p_i sont indépendantes les unes des autres (i.e. les défauts ne sont pas liés entre eux).
- Un ensemble $\mathcal{S}$ de n_S tests s_j avec $j \in \{1, ..., n_S\}$. Les tests sont supposés **multi-valués** : chaque test s_j possède au moins deux modalités. Le nombre de modalités du test s_j est noté n_M^j.
- Un ensemble $\mathcal{C}$ de n_S coûts de test $c_j \geq 0$ avec $j \in \{1, ..., n_S\}$ qui représente le coût de réalisation du test s_j en termes de temps et ressources (humaine et matériel).
- Une matrice $\mathbf{A} = [a_{ij}]$ de dimension $n_F + 1 \times n_S$ où a_{ij} représente les modalités du test s_j en présence d'un défaut f_i. Cette matrice est appelée «matrice de signature des défauts».

Le problème consiste à construire une séquence de tests qui est capable d'isoler chacun des défauts en minimisant le coût J de l'arbre de diagnostic donné par l'équation donné par l'équation (2.1) où $D = [d_{ij}]$ est une matrice binaire de dimension $n_F \times n_S$ telle que $d_{ij} = 1$ si le test s_j est utilisé dans la séquence de tests menant à l'identification du défaut f_i et $d_{ij} = 0$ sinon. Chaque sous-ensemble de défauts présent au niveau d'un nœud de l'arbre de diagnostic est appelé un *ensemble d'ambiguïté*. L'ensemble $\mathcal{F}$ constitue l'*ensemble d'ambiguïté totale*. Ce problème est connu comme étant NP-complet.

$$J = \sum_{i=0}^{n_F} p_i \times \left(\sum_{j=1}^{n_S} d_{ij} \times c_j \right) \tag{2.1}$$

Notons que la matrice de signature des défauts **A** est à rapprocher de celle définie par l'approche FDI. Dans le cas du TSP, la signature d'un défaut exprime l'influence de celui-ci sur le résultat de chacun des tests.

2.3.3 Stratégies de choix d'un test

Les algorithmes exacts comme la programmation dynamique sont rarement utilisés pour résoudre le TSP car comme tous les problèmes NP-complets, sa complexité est exponentielle. Les solutions existantes utilisent généralement des méthodes de recherche heuristique afin de fournir rapidement une solution optimale ou qui s'en approche. Deux catégories de solutions à base d'heuristiques existent : les heuristiques locales et les arbres de recherche optimaux ET/OU.

Optimisation selon une heuristique locale

Ces méthodes permettent de construire un arbre de diagnostic en réalisant une optimisation locale. Un exemple d'heuristique est celle basée sur le *gain d'information* [Johnson, 1960]

utilisée dans le cadre de tests binaires. Un test s_k est choisi pour un ensemble d'ambiguïté x s'il maximise le gain d'information par unité de coût comme montré par l'équation (2.2) où IG est le gain d'information (entropie) donné par l'équation (2.3).

$$k = arg \max_j \left(\frac{IG(x, s_j)}{c_j} \right) \tag{2.2}$$

$$IG(x, s_j) = -\left(P(x_{j0}) \times \log_2 P(x_{j0})\right) - \left(P(x_{j1}) \times \log_2 P(x_{j1})\right) \tag{2.3}$$

avec x_{j0} et x_{j1} correspondant respectivement aux modalités 0 et 1 du test binaire s_j.

L'arbre qui résulte de ce type d'approche est certainement moins bon que celui optimisé de manière globale mais sa génération a l'avantage d'être peu coûteuse en calcul. De plus, à chaque étape de la génération de l'arbre, le problème revient à un problème de génération du prochain meilleur test.

Génération d'un arbre de recherche ET/OU

Le problème de recherche d'arbre de diagnostic optimal peut être formulé comme un problème de recherche du meilleur d'abord sur un arbre ET/OU [Pattipati et Alexandridis, 1990]. Celui-ci est composé de deux sortes de nœuds :

- Un nœud OU représente un ensemble de défauts (i.e. ensemble d'ambiguïté)
- Un nœud ET représente un test et ses différentes modalités.

Le nœud racine est évidemment un nœud OU comprenant tous les défauts possibles du système étudié (i.e. ambiguïté totale).

La figure 2.2 montre un arbre de diagnostic qui représente le sous-arbre de l'arbre de recherche ET/OU sélectionné. Il est solution du problème de séquencement des tests composé des 4 défauts $\{f_1, f_2, f_3, f_4\}$ et des 5 tests $\{s_0, s_1, s_2, s_3, s_4\}$. Le symbole "-" représente une feuille vide (le résultat du test s_5 correspondant n'est pas présent dans l'ensemble de défauts $\{f_3, f_4\}$).

Comme la génération d'un arbre de diagnostic optimal est un problème NP-complet, il est nécessaire de considérer des approches heuristiques pour guider la recherche dans le graphe ET/OU. La clé de cette approche [Pattipati et Alexandridis, 1990] est d'utiliser une fonction d'évaluation heuristique (FEH) afin d'éviter de considérer l'ensemble complet des arbres de diagnostic potentiels. La FEH est une estimation facilement calculable $h(x)$ du coût objectif optimal $h^*(x)$ de chaque nœud du sous-ensemble d'ambiguïté x jusqu'aux feuilles.

2.3.4 La méthode AGENDA

AGENDA (Automatic GENération of DiAgnosis trees)[Faure, 2001; Olive, 2003] est une méthode opérationnelle qui permet de générer automatiquement des arbres de diagnostic à partir des données de conception fournies par les constructeurs automobiles ; un schéma synoptique de la méthode est donné Figure 2.3. La solution proposée du problème de séquencement des tests, énoncé précédemment, est basée sur des modèles de défauts ensemblistes

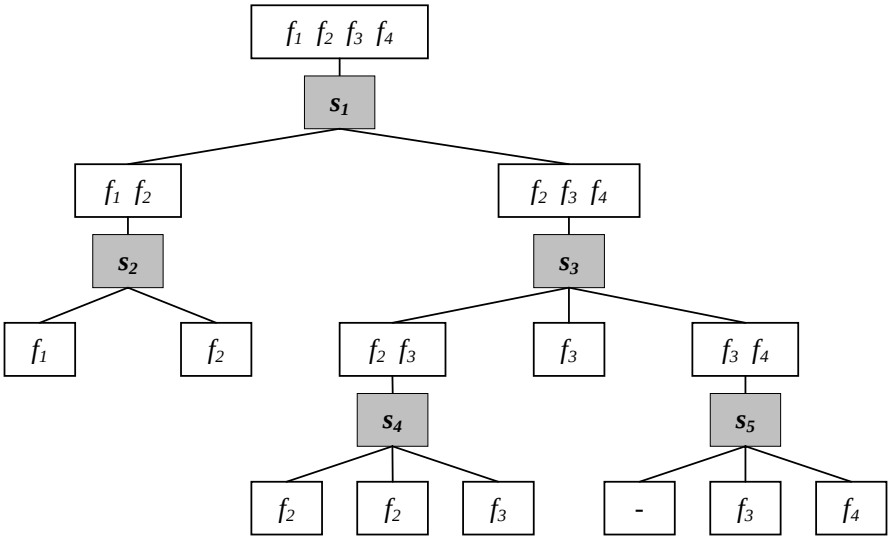


Figure 2.2 — Arbre optimal de diagnostic

et un algorithme de recherche heuristique de type AO* pour générer un arbre de diagnostic optimal. Cette optimalité tient compte de coûts statiques pour les tests.

Anticipation des défauts AGENDA, par une modélisation structurelle et comportementale, anticipe l'ensemble $\mathcal{F}$ des défauts envisageables pour un circuit électrique purement résistif sous l'hypothèse de défaut unique et persistant. Quatre catégories de défauts sont définies :

- un *défaut de paramètre* correspond à une valeur non nominale et extrême (valeur nulle ou valeur infinie) assignée à un de ses paramètres. Cela correspond à des comportement de type circuit ouvert ou court-circuit sur un composant.
- un *défaut de commutation* représente une commutation qui est dans un mode défaillant (interrupteur restant collé fermé ou ouvert).
- un *défaut de connecteur* correspond à un court-circuit entre deux broches voisines appartenant au même connecteur.
- un *défaut de connexion* correspond à un circuit ouvert entre deux broches en vis à vis d'un connecteur.

Anticipation des tests Chaque test est défini par un triplet < Configuration, Variable, Modalités >. Les configurations correspondent aux actions à mener par le garagiste sur le système pour que celui-ci soit dans le mode opératoire permettant de réaliser la mesure. Elles sont de deux types

- Les *configurations structurelles* correspondent à l'état structurel de chaque connecteur électrique (connecté/déconnecté).
- Les *configurations utilisateur* correspondent à l'état des interrupteurs de commande (i.e. commandes aux volant) que le garagiste doit activer pour placer le circuit électrique dans le mode recherché.

Les tests anticipés sont exclusivement de nature électrique et statique, réalisés à l'aide d'un multimètre. On retrouve ainsi les tests de tension et les tests de résistance équivalente réalisés entre deux points de potentiels accessibles.

Etape de prédiction L'étape de prédiction correspond au calcul de la matrice de signature des défauts comme définie dans le problème de séquencement des tests. Dans AGENDA, les résultats de chacun des tests sont exprimés sous forme d'intervalles afin de prendre en compte la notion de précision sur les paramètres du modèle (les résistances). Ce calcul se fait en deux étapes :

1. Dans un premier temps, le résultat de chaque test de $\mathcal{S}$ pour chacun des défauts de $\mathcal{F}$ est exprimé de manière symbolique. A partir de l'hypothèse selon laquelle un système à diagnostiquer peut se mettre sous la forme d'un réseau résistif comprenant une source de tension, l'expression matricielle symbolique du modèle du système est mis sous la forme $\mathbf{A}(Y) \times X = B$ où $\mathbf{A}(Y)$ est une matrice carrée où intervient l'ensemble Y des paramètre résistifs. X est un vecteur de variables et B est un vecteur de constantes et de paramètres actifs (sources tensions). La résolution de ce système se fait selon la méthode Cramer classique.
2. Dans un deuxième temps, une méthode d'optimisation globale du type «*Branch and Band*» est utilisée afin de propager l'imprécision des paramètres résistifs sur le résultat des tests.

Génération de l'arbre Dans AGENDA, un algorithme du type AO* (cf. Chapitre 6) permet de générer l'arbre de diagnostic qui est du type ET/OU. L'algorithme se base sur une hypothèse de départ qui est que les tests sont *non exclusifs* et *multivalués*. La non exclusivité vient du fait que l'expression du résultat des tests sous forme d'intervalles implique pour le même test des chevauchements d'intervalles pour différents défauts. La conséquence de cela est qu'un même défaut peut se retrouver sur plusieurs feuilles de l'arbre.

Limites de la méthode L'approche développée dans AGENDA a donné des résultats intéressants pour isoler des défauts de manière optimale sur des circuits électriques. De manière opérationnelle, elle permet de générer un arbre de diagnostic pour chaque circuit électrique connecté à un calculateur et pouvant être associé à un code défaut (cf. section 1.1.3). Cependant, la méthode est insuffisante pour être appliquée sur les architectures des véhicules actuels :

- La modélisation exclusive sous forme de réseau résistif limite le périmètre d'application de la méthode de prédiction.
- Le fait qu'un arbre de diagnostic optimal soit généré pour chaque code défaut limite le périmètre d'application de la méthode lorsqu'il y a apparitions de défauts sans codes défauts générés ou lorsque plusieurs codes défauts sont générés en chaîne alors qu'un seul défaut est présent sur le système. Ces cas, dûs à la complexité fonctionnelle des véhicules actuels, sont de plus en plus fréquents.
- Les tests proposés au garagiste sont exclusivement de nature électrique et sont donc coûteux à réaliser car ils nécessitent un certain nombre de manipulations mécanique

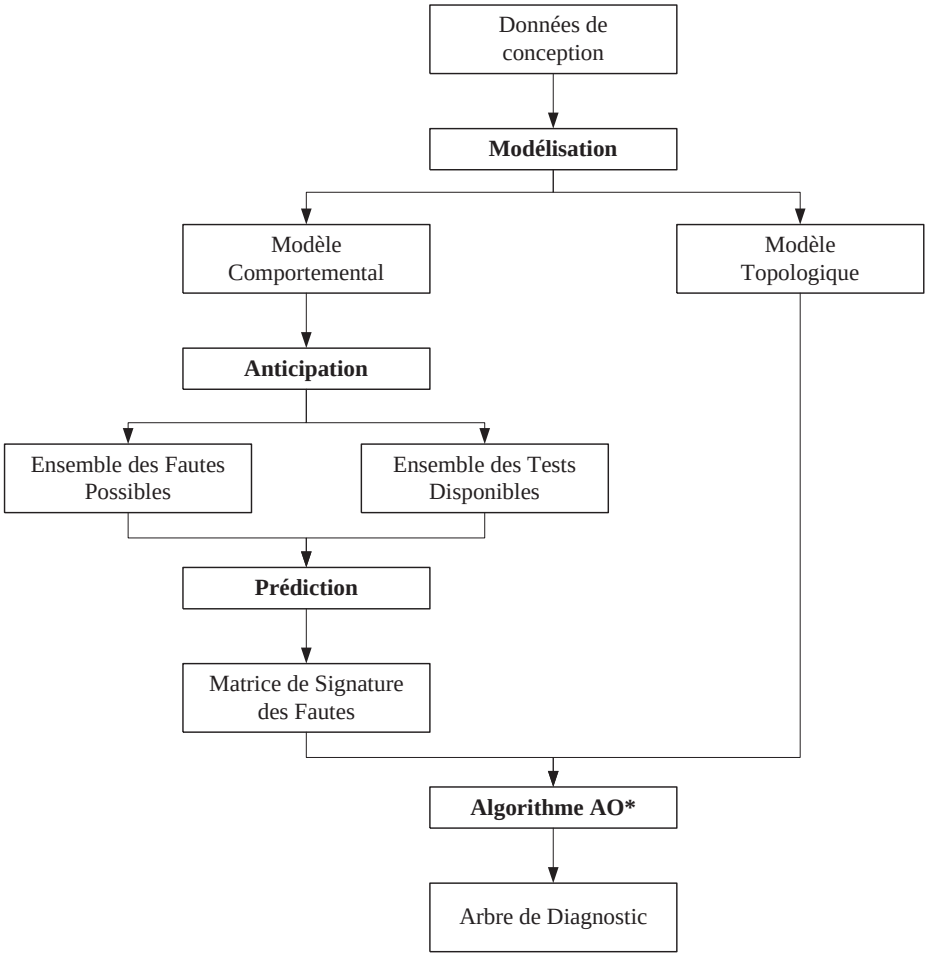


Figure 2.3 — Principe général de AGENDA

de dépose de composants pour accéder aux points de mesure.

- La prédiction des tests dans AGENDA est statique, elle donne lieu à la prédictions de mesure qui n'évoluent pas au cours du temps. Or les systèmes embarqués actuels sont des systèmes dynamiques hybrides avec des commandes à dominante séquentielle. Ainsi, l'hypothèse selon laquelle le système peut être mis dans un mode opératoire stable pour réaliser une mesure n'est pas toujours vraie. Par ailleurs, un défaut peut affecter le comportement séquentiel d'une fonction sans pour autant modifier l'amplitude d'une tension électrique. Enfin, certains défauts donnent lieu à des comportements séquentiels dégradés anticipés dans les fonctions d'autodiagnostic des calculateurs.
- La génération d'arbre est certes intéressante pour garantir une optimalité globale mais a l'inconvénient majeur de ne pas être « interactive » : si un test proposé au garagiste se trouve non réalisable, cela entraîne l'échec du parcours de l'arbre de diagnostic et le garagiste ne peut refuser ce test.

2.4 Diagnostic à Base de Modèles et automobile

2.4.1 Les principaux outils

Dans ce paragraphe, nous présentons de manière non exhaustive les principales réalisations européennes en terme d'outils de raisonnement à base de modèles. La plupart de ces outils sont souvent destinés à réaliser des analyses de conception automatiques (AMDEC, SCA...) mais peuvent proposer pour certains des fonctionnalités de diagnostic. Ces outils ont notamment été développés (ou améliorés lorsqu'ils existaient déjà) suite aux réflexions du groupe de travail Automotive Task Group [Monet2, 2003] du réseau d'excellence MONET2.

AutoSteve [Price *et al.*, 1995; Price, 1997; Snooke et Price, 1998; N.A.Snooke et J.Bell, 2002] Le but d'AutoSteve est d'automatiser les méthodes d'analyse de conception de systèmes embarqués et a été développé à l'université d'Aberystwyth. Une version commerciale au nom d'« AutoSteve 2 » a été adoptée par Ford en 1999. Le projet a été maintenant repris par la société Mentor Graphics spécialisée dans les outils de CAO (Conception assistée par Ordinateurs) de systèmes électroniques embarqués. Deux techniques d'intelligence artificielle sont utilisées pour l'analyse de conception :

- La simulation qualitative : elle permet de simuler le comportement des systèmes électriques du véhicule à partir de modèles qualitatifs.
- L'abstraction fonctionnelle : l'abstraction fonctionnelle permet d'abstraire les résultats issus de la simulation qualitative à un niveau plus abstrait, facilement interprétables par l'expert.

L'approche était initialement basée sur le même type de système qu'AGENDA mais à été étendue récemment, à travers le projet SoftFMEA, afin de prendre des systèmes embarqués complets intégrant composants logiciels et composants matériels. Cet outil a particulièrement retenu notre attention, car c'est l'un des seuls qui utilise des modèles fonctionnels.

Rodon (Sörman Information) [Seibold, 1994; Lunde *et al.*, 2006] Rodon est un outil basé modèle pour la simulation, la détection de défauts, le diagnostic et l'analyse de conception (AMDE, SCA), développé et commercialisé initialement par ROSE Informatik puis repris par Sörman Information. Cet outil s'appuierait sur des modèles symboliques, basés sur des intervalles ainsi que sur des modèles dynamiques algébro-différentiels et leur représentation discrétisée (Siebold 1994a, 1994b). Malgré le fait que Rodon soit un outil dont la documentation actuelle ne permet pas réellement une évaluation détaillée, l'approche de diagnostic semble couvrir un périmètre sensiblement équivalent à celui d'AGENDA.

Raz'r (OCC'M Software GmbH) [Sachenbacher *et al.*, 1998, 2000] Raz'r est un outil qui met en œuvre un raisonnement à base de maintien de cohérence pour différentes tâches de raisonnement appliqués à des systèmes automobile : diagnostic (embarqué ou débarqué), analyses de conceptions automatiques (AMDEC, SCA), génération automatique de tests,... La modélisation est orientée composant, qualitative et le moteur de diagnostic est

basé sur les principes de l'IA.

Autas [Picardi *et al.*, 2004] est un outil qui permet de générer automatiquement des analyses AMDEC. La modélisation est orientée composant (avec représentation uniquement structurelle), qualitative et permet des descriptions hiérarchiques des systèmes.

2.4.2 Intérêt du raisonnement à base de modèles

Les avantages principaux et souvent recherchés de l'approche à base de modèles sont les suivants :

- La représentation des connaissances sur le système physique (modèles) d'une part et des connaissances sur la tâche de diagnostic (moteur de diagnostic avec ses stratégies) d'autre part se font de manière séparée, ce qui peut avoir des conséquences significatives sur les coûts de développement, de validation et de maintenance. Cet aspect est essentiel lorsqu'il s'agit de systèmes hybrides, dont la structure même du modèle varie avec le temps.
- La représentation du système physique, sous forme de modèles, s'appuie sur une connaissance disponible dès la conception du système et ne nécessite pas de connaissances opérationnelles.
- La modélisation est modulaire, c'est-à-dire que l'on peut dériver un modèle global à partir d'une bibliothèque de modèles génériques élémentaires pour les composants de base. Ceci facilite la réutilisation des modèles et décroît également les coûts de développement et de maintenance.
- Les modèles de défaut ne sont pas nécessaires (pour le diagnostic purement fondé sur la cohérence, qui permet la localisation des défauts) mais ceux qui existent peuvent être efficacement pris en compte (extension au diagnostic abductif, qui permet l'identification des défauts et l'explication des symptômes).

2.4.3 Les limites des approches à base de modèles appliqués à l'automobile

Le point dur du diagnostic à base de modèles réside justement dans les modèles même et leur construction. Leur élaboration peut être un frein à ces approches sur plusieurs points :

- Bien qu'un modèle doive provenir de données de conception, ces dernières ne sont pas toujours disponibles, souvent pour des raisons de protection industrielle émanant des constructeurs automobile. Dans ce cas, les modèles comportementaux peuvent être très difficiles à obtenir pour des composants complexes mettant en jeux des phénomènes provenant de différents domaines de la physique.
- Dans le cas où la connaissance de conception est disponible, celle-ci peut être trop détaillée pour une tâche de diagnostic. Par exemple, il n'est pas nécessaire d'avoir un modèle physique complet d'un capteur, une fonction de transfert est parfois largement suffisante. De même, un calculateur électronique n'a pas besoin d'être modélisé physiquement mais une simple description de son comportement événementiel peut se révéler suffisante. Cela pose naturellement le problème de déterminer la connaissance nécessaire et suffisante à la tâche de diagnostic visée.

- Les modèles structurels et comportementaux utilisés dans les approches classiques et notamment dans AGENDA ne permettent d'avoir que des mesures physiques dans la séquence de tests. Or, les symptômes de haut niveau tels que les symptômes client ou des observations fonctionnelles peuvent être assimilés à des tests et sont très intéressants à exploiter dans la mesure où ils sont peu coûteux. Pour prendre en compte ce type de test il semble nécessaire d'adapter la tâche de raisonnement pour qu'elle porte sur d'autres types de modèles plus qualitatifs tels que les modèles fonctionnels.

2.4.4 Positionnement des travaux du module MBR de MODE

Afin de tenir compte des différentes contraintes liées au diagnostic en garage ainsi que de la complexités des systèmes embarqués, nous avons choisi d'orienter nos travaux suivant plusieurs points :

- *Réalisation d'un diagnostic multi-modèle* : La complexité des systèmes à diagnostiquer et l'hétérogénéité des connaissances disponibles sur les véhicules incitent à utiliser plusieurs types de modèles et notamment des modèles fonctionnels. Ces modèles, organisés de manière hiérarchique, ont plusieurs objectifs :
  - réaliser des abstractions afin de diversifier le raisonnement.
  - prendre en compte tous les types d'observation dans la phase de diagnostic. La hiérarchie permettra notamment d'associer un symptôme de haut niveau au comportement d'un composant matériel. Nous allons pour cela nous appuyer sur les travaux qui concernent la modélisation fonctionnelle et hiérarchique présentés dans le chapitre 3.
  - avoir un raisonnement pouvant porter également sur les fonctions du véhicule.
- *Anticipation des défauts* : La méthode AGENDA a permis de montrer que l'anticipation des défauts augmente l'efficacité du diagnostic, notamment par le caractère multivalués des tests. Nous souhaitons donc conserver ce cadre en l'étendant à la problématique multi-modèle et au cas des systèmes hybrides.
- *Choix du prochain meilleur test* : Le séquencement de tests par arbre de diagnostic, tel qu'il existe dans AGENDA, n'autorise pas suffisamment de souplesse pour le garagiste pendant la session de diagnostic. Nous avons donc adopté une stratégie basée sur le choix du prochain meilleur test.
- *Coût dynamique des tests* : La réalité du terrain fait que le coût de réalisation d'un test n'est pas statique car il dépend de manipulations successives (en termes de déposes de composants) qui sont effectuées par le garagiste durant la session de diagnostic. Le problème d'évaluation des coûts dynamiques de ces tests avait été posé dans AGENDA mais il n'est pas pris en compte dans l'algorithme proposé car celui-ci remettait en question l'admissibilité de l'heuristique.
- *Implémentation des modèles* : le langage qui a été choisi pour implémenter les modèles dans AGENDA et le moteur de prédiction des tests était spécifique aux réseaux résistifs. Afin d'éviter le coût supplémentaire de développement d'un simulateur hybride, nous avons choisi de nous tourner vers un langage standard qui est MODELICA.

3 Représentation de la connaissance pour le diagnostic à base de modèles

La complexité croissante des systèmes réels actuels a orienté de nombreux travaux vers une structuration du raisonnement de diagnostic pouvant exploiter des modèles à différents niveaux d'abstraction organisés suivant une hiérarchie. Ces abstractions permettent de réduire la complexité de calcul des raisonnements à effectuer et de traiter des systèmes pour lesquels les connaissances et les observations disponibles sont hétérogènes. L'objectif de ce chapitre est de dresser un bref état de l'art des différentes approches utilisant les abstractions pour le raisonnement à base de modèles.

3.1 La modélisation : principale difficulté

Un modèle peut être vu comme la représentation mathématique d'un système physique réel appropriée pour un objectif de raisonnement précis. Il s'agit d'une représentation seulement partielle de la réalité soumise aux choix subjectifs de son concepteur en fonction de la nature du système considéré, de l'ontologie des entités modélisées et du niveau d'abstraction choisi.

3.1.1 Ontologie des modèles

Construire un modèle implique la description d'entités ayant un certain nombre de propriétés et des relations entre ces entités. Les choix faits dans cette description définissent l'ontologie du modèle. En général, trois classes d'ontologies sont représentées dans la littérature [Chittaro *et al.*, 1993] :

- L'*ontologie orientée objet* (ou « *orientée composant* ») suppose que la réalité est composée d'objets individuels dont les propriétés peuvent être définies de manière générale, objective et indépendamment du contexte.
- L'*ontologie orientée système* suppose que la réalité est composée de systèmes organisés en unités ne pouvant être définis séparément.
- L'*ontologie orientée processus* est une ontologie hybride, à l'interface des deux précédentes. Les composants ne sont pas modélisés explicitement mais seulement les pro-

cessus qui sont mis en jeu.

La théorie du diagnostic logique se place résolument suivant une approche de modélisation orientée composant. Les composants sont les éléments remplaçables du système physique et constituent donc l'objet du diagnostic. La représentation de la connaissance s'organise ainsi de manière générique autour d'une bibliothèque de modèles de comportement des composants du domaine. Ces composants sont instanciés pour chaque système particulier puis le modèle est complété par la description de la structure du système, c'est à dire des interconnexions entre composants.

Cependant, l'hétérogénéité et la complexité des systèmes actuels nécessite l'utilisation d'autres points de vue comme la modélisation fonctionnelle qui a été introduite comme une alternative aux modèles comportementaux [Sticklen et Chandrasekaran, 1989; Abu-Hanna et Jansweijer, 1994; Chittaro *et al.*, 1993] ; les théories s'organisant par exemple autour de la notion de processus physiques interagissant entre eux [Forbus, 1984]. Dans ce cadre, les processus modélisés doivent être nécessairement liés aux composants du système afin qu'un diagnostic établi sur des processus puisse être relié à des éléments structurels du système pouvant faire l'objet d'actions réparatrices.

3.1.2 Nature des systèmes

Les systèmes continus et les systèmes à événements discrets ont été classiquement considérés comme deux axes de modélisation différents. Dans le cas des systèmes continus le domaine du diagnostic logique s'est développé en étroite relation avec le domaine du raisonnement qualitatif ([Travé-Massuyès et Dague, 2003; Weld et de Kleer, 1990; Armengol *et al.*, 1998]) qui propose une alternative aux modèles numériques. En effet, la tâche de diagnostic doit avant tout disposer de modèles corrects dans la perspective de localisation et d'identification de fautes afin de ne pas conduire à des diagnostics (issus de conflits) erronés. Ainsi, il n'est pas nécessaire que les modèles soient précis, certaines distinctions qualitatives pertinentes peuvent être suffisantes.

Par l'abstraction qualitative des quantités physiques, le continu se trouve discrétisé, que ce soit en terme de signes (partition la plus grossière en -, 0 , +) ou d'une partition plus fine de l'ensemble des réels $\mathbb{R}$.

Pour ce qui concerne les systèmes à événements discrets et leur représentation, on reste dans des cadres de modélisation classiques par machines à états finis (i.e. automates, réseaux de Petri, etc.) qui véhiculent une abstraction événementielle et qui peuvent aussi s'utiliser dans un cadre hiérarchique.

Toutefois, les systèmes complexes comme les système embarqués ont un comportement hybride et font à la fois intervenir des modèles discrets et continus. De nombreux travaux sont actuellement consacrés à la modélisation et au diagnostic des systèmes hybrides [Zaytoon, 2001; Cocquempot *et al.*, 2005; Travé-Massuyès *et al.*, 2006; Bayoudh *et al.*, 2007].

3.2 Notion de hiérarchie

La complexité grandissante des systèmes réels incite à abstraire les modèles comportementaux afin de réduire la complexité de la tâche de raisonnement à effectuer. Dans certains cas, réaliser ces abstractions consiste à représenter des modèles selon plusieurs niveaux hiérarchiques. Le plus haut niveau permet souvent de considérer le système dans son ensemble, la hiérarchie permet ensuite de focaliser progressivement le raisonnement sur un composant spécifique du système. Dans cette section, nous définissons de manière générale la notion de hiérarchie d'abstractions et plus spécifiquement la hiérarchie fonctionnelle.

3.2.1 Les axes hiérarchiques

Les abstractions s'organisent classiquement autour de deux axes hiérarchiques [Lind, 1999; Modarres et Chehon, 1999] :

- La *hiérarchie par agrégations* consiste à construire plusieurs niveaux d'abstractions où l'objet décrit à un niveau d'agrégation donné est une partie d'un objet décrit à un niveau d'agrégation supérieur. Les abstractions structurelles ont notamment été utilisées pour le diagnostic [Autio et Reiter, 1998; Chittaro et Ranon, 2004].
- La *hiérarchie fonctionnelle* (appelée *hiérarchie multi-modèle* dans [Chittaro *et al.*, 1993]) réalise une interprétation téléologique du comportement du système en exhibant les rôles que jouent chacun des composants structurels dans la réalisation de la fonction globale du système (but pour lequel le système a été conçu). Ces abstractions ont fait l'objet de nombreuses approches et ont été appliquées sur des domaines industriels très différents souvent à forte dominante technologique (ex : centrales nucléaires, systèmes mécatroniques).

3.2.2 La hiérarchie fonctionnelle

3.2.2.1 Le concept de fonction

En reprenant la définition donnée par l'IFMAA (International Functional Modeling and Application Association), la modélisation fonctionnelle s'appuie d'une part sur l'identification du but global défini par le concepteur que le système (ou sous-système) doit atteindre, et d'autre part sur les fonctions définies par l'utilisateur que le système doit réaliser. Une hiérarchie fonctionnelle doit donc répondre aux trois questions suivantes [Modarres et Chehon, 1999] :

- « Pourquoi le système a-t-il été conçu à l'origine ? » La réponse à cette question renseigne sur les intentions du concepteur et le besoin de l'utilisateur.
- « Que doit faire le système afin d'atteindre le but (ou l'objectif) initial ? »
- « Comment doivent interagir les différentes parties du système afin de réaliser les fonctions ? » La réponse à cette question clarifie le comportement de la structure physique, c'est à dire le comportement de chaque composant et les interconnexions entre composants.

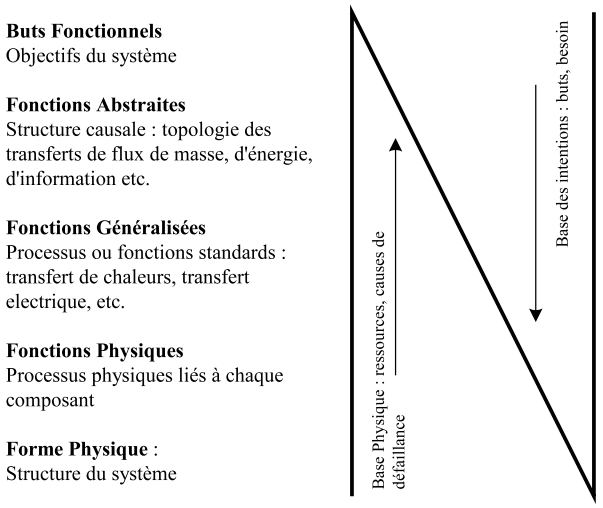


Figure 3.1 — Hiérarchie d'abstraction de Rasmussen [Rasmussen, 1986]

Les différentes approches font souvent référence à la hiérarchie d'abstraction proposée par Rasmussen [Rasmussen, 1986] illustrée Figure 3.1. Celle-ci aborde le concept de fonction selon différents points de vue : au plus haut niveau, une fonction représente une intention du concepteur, au niveau composant il représente un rôle individuel et entre ces deux extrémités il peut se rapporter à la notion de processus. L'emploi du terme de fonction prête parfois à confusion si ce statut n'est pas clairement précisé.

3.2.2.2 Hiérarchie basée sur quatre types épistémologiques

Quatre grandes classes de connaissances sont identifiées comme pouvant servir au raisonnement à base de modèles [Chittaro *et al.*, 1993] :

- La *connaissance structurelle* porte sur la topologie du système. Elle décrit les composants comme constitués de ports destinés à établir des interactions avec les autres éléments du système et les connexions entre les différents composants.
- La *connaissance comportementale* décrit les propriétés internes de chaque composant structurel en terme d'équations physiques.
- La *connaissance fonctionnelle* décrit comment le comportement individuel des composants contribue en terme de rôles (appelés fonctions de base dans [Kitamura *et al.*, 2002]) et de processus à la réalisation des fonctions prévues par le concepteur du système.
- La *connaissance téléologique* spécifie les fonctions du système en terme de buts (appelés méta-fonctions dans [Kitamura *et al.*, 2002]) qui lui ont été assignés par son concepteur.

Cette classification rejoint celle de Rasmussen et se justifie par le type d'ontologie (cf. section 3.1.1) utilisé dans chaque type épistémologique. Ainsi, les modèles structurel et comportemental sont décrits selon une ontologie orientée objet et le modèle téléologique selon une ontologie orientée système. Le concept de fonction établit clairement un pont entre comportement et téléologie. La représentation fonctionnelle vise à décrire comment le comportement de chaque composant élémentaire contribue à la réalisation du ou des buts communs

assignés au système par son concepteur.

3.2.2.3 Représentation d'une entité fonctionnelle

Une entité fonctionnelle se représente par un ensemble de variables d'entrées et un ensemble de variables de sorties pour lesquelles certaines propriétés doivent être vérifiées afin de déterminer l'état fonctionnel de cette entité. Ces propriétés se formulent par un triplet <pré-conditions, effets, post-effets> dont les éléments se définissent de la manière suivante :

- *pré-conditions* : ensemble de prédicats logiques qui caractérisent la situation en entrée nécessaire à la réalisation de la fonction.
- *effets* : ensemble de prédicats logiques qui caractérisent la situation en sortie qui est vraie lorsque la fonction est réalisée.
- *post-effets* [Chittaro *et al.*, 1993] : ensemble de prédicats logiques qui caractérisent la situation en sortie qui doit être vraie lorsque la fonction n'est plus activée, c'est-à-dire lorsque les pré-conditions deviennent fausses. Les post-effets sont parfois absents de la description.

Notons que ces propriétés peuvent être des instances de variables physiques. La création de ces instances correspond à une discrétisation des variables continues (par abstraction qualitative par exemple) d'un système afin d'exhiber certains modes de fonctionnement spécifiques.

3.3 Approches de modélisation fonctionnelle

Il existe deux grandes écoles de la modélisation fonctionnelle. La première, dite basée état, est née des travaux de la communauté IA et propose une discrétisation des variables continues dès le niveau composant. La seconde, basée sur le concept de variables généralisées, est née des travaux des communautés de l'automatique et de l'électrotechnique cherchant un cadre unifié pour simuler et commander des systèmes multi-physique complexes.

3.3.1 Approches basées état

Les différentes approches dites « basées état » ont leur origine dans les travaux de B. Chandrasekaran et de son équipe de l'université de l'Ohio aux Etats-Unis [Sumbugamoorthy et Chandrasekaran, 1986; Goel et Chandrasekaran, 1989] sur la méthode FR (Functional Representation). Elles se basent sur le concept d'*état partiel* qui correspond à des prédicats sur les variables d'état du système. Par exemple, le fait que la commande d'un moteur « essuie-vitre » soit activée (cf. Figure 3.2) sera noté par une affirmation sur la valeur de la variable de commande du type « commande = vraie ». Ce cadre est donc à rapprocher de la notion d'abstraction qualitative des variables d'état d'un système.

3.3.1.1 Principes de l'approche FR

Dans la forme originale de la méthode, le modèle fonctionnel est élaboré à partir d'une analyse structurée qui cherche à mettre en évidence le rôle fonctionnel des sous-systèmes et

les variables caractéristiques qui permettent d'en décrire le bon fonctionnement. La méthodologie proposée s'appuie sur les trois premiers niveaux de connaissance (Structure, processus causal, Fonction) :

- Le *modèle structurel* : la première étape consiste à décrire ce qu'est le système en le décomposant récursivement en sous-systèmes. Cette décomposition s'arrête dès qu'on a atteint un niveau satisfaisant (composants remplaçables) pour la tâche de raisonnement à effectuer (analyse de conception, diagnostic...).
- Le *modèle fonctionnel* : la seconde étape consiste à attribuer une fonction globale pour le système et une fonction à chaque sous-système. Une fonction se décrit classiquement à partir des éléments < pré-conditions, effets > auxquels on relie un comportement.
- Le *processus causal* (CPD : Causal Process Description [Iwasaki et Chandrasekaran, 1992; Chandrasekaran, 1994b]) : le processus causal est un graphe orienté dont les nœuds donnent une description de l'état partiel du système et dont les arcs annotés indiquent le changement logique qui fait passer le système d'un état partiel (nœuds en amont) vers un autre (nœuds en aval). L'annotation d'une transition indique quel est le mécanisme (fonction d'un sous système, loi physique) qui permet la passage entre états partiels. Dans l'approche FR, ce modèle est qualifié de modèle comportemental.

Le concept de fonction défini dans cette approche ne fait pas de différence entre la notion de rôle et la notion de but, il s'agit de la même notion. Une fonction se rapporte à un objet physique et est définie par les effets qu'a cet objet sur son environnement [Chandrasekaran et Josephson, 1997].

Pour illustrer ces concepts, nous donnons une partie du modèle fonctionnel FR d'un système « essuie-vitre » (cf. Figure 3.2 pour le système et Figure 3.3 pour son modèle fonctionnel). La fonction `essuyer_vitre` du système principal est donné par :

```
Pré-conditions : {Masse_Batterie AND commande AND Masse_Moteur}
Effets         : {balais_bougent}
Processus causal : {CPD_1}
```

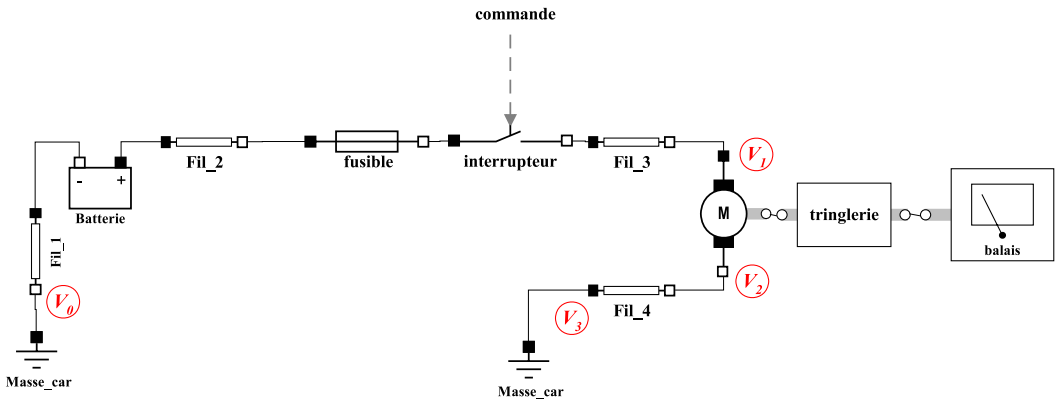


Figure 3.2 — Exemple d'un système « essuie-vitre » simple

Le graphe `CPD_1` (cf. Figure 3.3) de cette fonction fait apparaître des sous-fonctions qui appartiennent aux sous-systèmes ; ceci est indiqué pars les arcs discontinus entre structure et

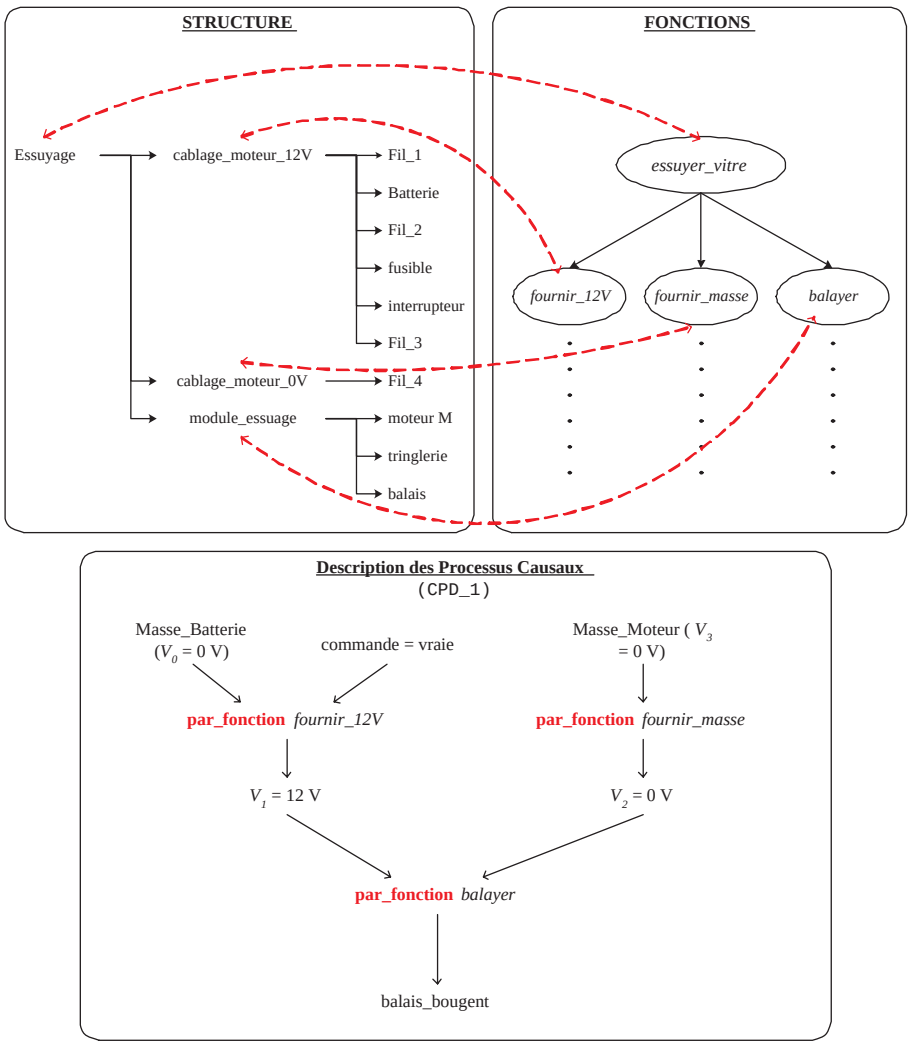


Figure 3.3 — Application de l'approche FR sur le système « essuie-vitre »

fonctions. Par exemple, le rôle du sous-système `cablage_moteur_12V` est d'amener le potentiel d'alimentation au moteur. On notera que les pré-conditions et les effets de la fonction figurent sur le graphe causal. Les fonctions référencées sur les arcs du graphe sont exprimées en fonction des fonctions des sous-systèmes de plus bas niveau par l'intermédiaire du mot-clé **`par_fonction`**.

3.3.1.2 Discussion sur l'approche

Les approches basées état ont été essentiellement utilisées pour la conception [Goel et Chandrasekaran, 1989; Iwasaki et Chandrasekaran, 1992; Chandrasekaran *et al.*, 1993] et l'analyse de conception (génération automatique d'AMDEC) [Hunt *et al.*, 1995; Hughes *et al.*, 1999; Price *et al.*, 1997; Price, 1997]. Certains travaux proposent toutefois des applications en diagnostic [Keuneke, 1991; Hawkins *et al.*, 1994] et notamment dans le domaine automobile [Duffaut, 1994].

Le fait que le modèle fonctionnel soit en correspondance directe avec la structure du système permet d'avoir une représentation qui peut varier suivant le besoin courant en choisissant notamment une granularité de modèle adéquat. Cependant, ce mode de représentation présente certains inconvénients :

- la décomposition hiérarchique n'est pas systématique car elle dépend fortement de la subjectivité des choix faits par le concepteur du modèle. Par ailleurs, la hiérarchie fonctionnelle établie dépend directement de l'implantation physique de la fonction, ce qui limite les possibilités de réutiliser les modèles pour un autre système qui réalise la même fonction globale mais dont l'implantation physique est différente.
- l'exhaustivité des relations causales décrites lors de la description du graphe de processus n'est pas garantie : elles sont écrites de manière « ad-hoc ». Il existe le risque de passer à côté de certains aspects physiques importants.
- l'abstraction qualitative de l'état d'un système sous la forme d'états partiels dès le niveau composant nécessite d'anticiper tous les cas importants. De manière générale, le cadre se limite donc à la modélisation du comportement nominal d'un système car l'introduction de modèles de défauts induirait une combinatoire trop importante.

3.3.2 Approches basées sur les variables généralisées

Les approches basées sur la notion de *variable généralisée* constituent la deuxième grande école concernant la représentation du concept de fonction. Elles sont couramment utilisées par les automaticiens pour le contrôle et l'analyse des systèmes complexes en se basant sur les concepts d'effort et de flux généralisés développés dans la théorie de Paynter [Paynter, 1961].

Cette théorie présente un cadre unifié pour modéliser un système physique en le définissant en terme de distribution d'énergie, de masse ou d'information traversant les composants qui le constituent. L'état du système s'exprime ainsi en fonction de quatre types de variables généralisées possibles qui sont les variables d'effort e (tension électrique, force, pression), de flux f (courant électrique, vitesse, volume), de moment p (flux magnétique) et de déplacement q (angle, déplacement).

Les méthodologies classiques basées sur ces concepts sont l'approche multi-modèle [Chittaro *et al.*, 1993], la méthode MFM (Multilevel Flow Modelling) [Lind, 1994], les graphes à liens (appelés aussi Bond Graphs) et le graphe informationnel causal [Hautier *et al.*, 1999]. Par la suite, nous présentons uniquement l'approche multi-modèle car elle présente un cadre complet de modélisation hiérarchique et tous les concepts qui y sont développés sont communs aux autres approches basées sur les variables généralisées.

Approche multi-modèle

Chittaro et son équipe furent parmi ceux qui effectuèrent le travail le plus approfondi concernant la modélisation des systèmes physiques en exhibant explicitement les quatre niveaux de connaissance et les liens entre ces niveaux dans une approche appelée multi-modèle [Chittaro *et al.*, 1993]. La méthodologie d'abstraction proposée est ascendante : elle part d'une description structurelle la plus fine possible d'un système et construit progressi-

vement la hiérarchie afin d'en identifier les buts. Une ontologie complète basée sur la notion de variables généralisées a notamment été établie dans l'objectif d'une génération systématique des modèles à chaque niveau de la hiérarchie.

Définition du concept de fonction La méthodologie proposée dans l'approche multi-modèle se base sur trois niveaux de modélisation pour abstraire de manière progressive le comportement en buts. Ces trois niveaux sont le modèle des rôles fonctionnels (ontologie orientée composant), le modèle des processus (ontologie hybride) et le modèle des phénomènes (ontologie orientée système).

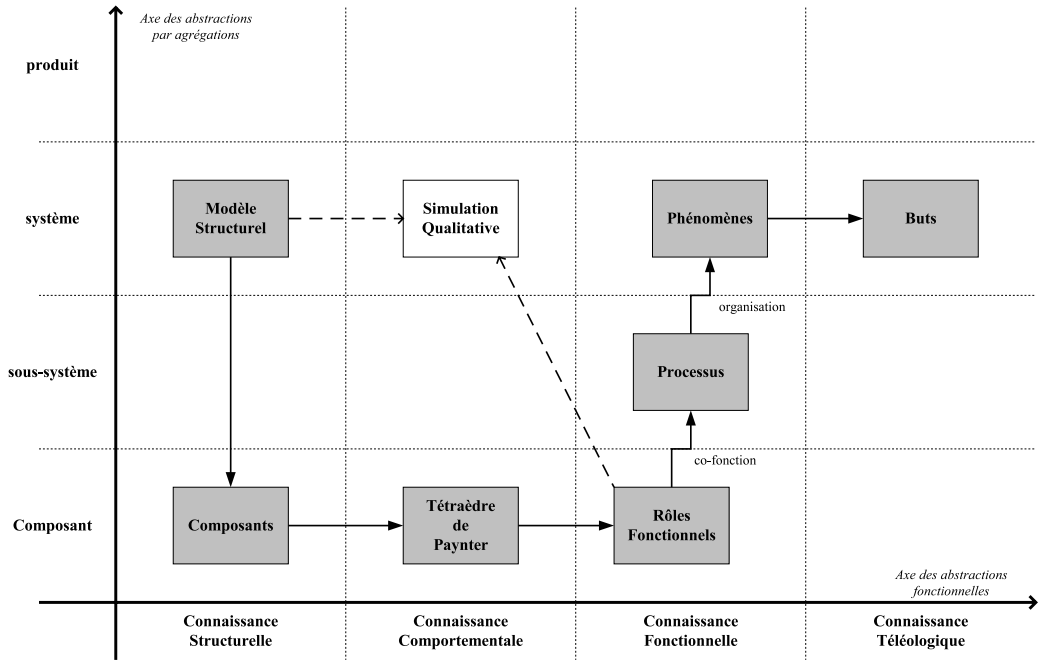


Figure 3.4 — Exemple de représentation multi-modèle

- Les *rôles fonctionnels* réalisent une interprétation du comportement des composants comme des opérateurs sur les variables de flux et d'effort généralisées. Ces rôles fonctionnels forment un réseau en étant reliés grâce à deux types de relations : la *relation de dépendance mutuelle*, lorsque deux rôles fonctionnels partagent une même variable généralisée, et la *relation d'influence* lorsqu'on change de domaine de la physique. Un même composant peut avoir plusieurs rôles fonctionnels à la fois. Par exemple, une résistance électrique chauffante présente les rôles de conduit de courant et de générateur de chaleur (il existe notamment une relation d'influence entre ces deux rôles).
- Les *processus* sont construits en associant des rôles fonctionnels appartenant au même domaine de la physique et possédant une relation de dépendance mutuelle. Cette notion de processus rejoint celle développée dans la théorie des processus qualitatifs de Forbus [Forbus, 1984]. Un processus s'exprime dans le cadre logique classique de la modélisation fonctionnelle grâce au quadruplet < co-fonction ; pré-conditions ; effets, post-effets >. La co-fonction est le réseau de rôles fonctionnels qui permet l'occurrence du processus. Le modèle des processus se présente comme un réseau d'influences

entre processus pour lequel chaque influence est typée en fonction de sa nature : causalité directe, support, régulation etc.
- Les *phénomènes* sont construits en associant plusieurs processus appartenant au même chemin causal dans le réseau de processus. Un phénomène s'exprime grâce à un quadruplet < organisation ; pré-conditions ; effets ; post-effets >.

Nous avons choisi de reprendre la représentation de ces modèles sous forme de grille comme proposé par Bell [Bell, 2006] car celle-ci permet de bien visualiser l'ensemble de la hiérarchie et l'interaction entre les modèles qui la composent (cf. Figure 3.4). L'axe vertical de cette grille correspond à une décomposition structurelle du système et l'axe horizontal représente les quatre types épistémologiques.

Nature des liens hiérarchiques L'approche multi-modèle définit trois types de liens hiérarchiques permettant d'inter-connecter les différents modèles :
- les *liens épistémologiques* permettent de passer d'un type épistémologique à un autre. Ils existent notamment entre le modèle structurel et le modèle comportemental, entre le modèle comportemental et le modèle des rôles fonctionnels puis entre le modèle des phénomènes et le modèle téléologique.
- les *liens d'agrégation* permettent de passer d'un niveau d'agrégation à un autre (cas des abstractions structurelles).
- les *liens ontologiques* permettent de passer d'une ontologie à une autre. Dans le cadre du modèle fonctionnel, ils correspondent à l'abstraction par interprétation causale du comportement.

Applications et intérêt de la méthode La hiérarchie proposée dans l'approche multi-modèle repose sur des principes sensiblement différents de ceux développés dans l'approche orientée état. Dès le modèle des processus, elle apporte une représentation qui devient indépendante de la réalisation matérielle du système physique modélisé. Le fait de reporter la discrétisation des variables physiques continues au niveau des processus permet aussi de réduire la combinatoire des processus causaux de l'approche FR.

Le cadre proposé, permet une modélisation hiérarchique quasi-systématique des systèmes multi-physique à dynamique continue. Ceci est obtenu grâce au cadre proposé par la théorie de Paynter et de l'ontologie qui en découle. En effet, les rôles fonctionnels et les processus sont typés de manière générique par un ensemble fini de primitives, ce qui fait que le processus d'abstraction est indépendant de la subjectivité du concepteur au moins jusqu'au modèle des processus. Il est néanmoins certain que plus on se rapproche du modèle téléologique, et donc des aspects cognitifs, plus les modèles deviennent subjectifs avec un typage des phénomènes et des buts dépendants du domaine d'application. Certains travaux ont exploité la correspondance entre les éléments des graphes à liens avec les rôles fonctionnels pour proposer une méthode de construction automatique du modèles des processus [Thétiot *et al.*, 1998; Thétiot, 1999; Zouaoui *et al.*, 1999]. Ils ont montré l'intérêt de l'analyse causale du graphe pour générer les co-fonctions des processus.

L'approche multi-modèle a essentiellement été utilisée pour des tâches de diagnostic de systèmes physiques continus. La hiérarchie est utilisée suivant une stratégie de focalisation

du diagnostic en partant des buts vers les rôles fonctionnels :

- Le modèle téléologique permet d'appréhender les symptômes perçus par l'opérateur humain et notamment d'analyser si le système est utilisé correctement par l'opérateur (si toutes les pré-conditions sont remplies) et conclure en terme de buts non réalisés.
- En exploitant tous les liens entre les modèles il est ainsi possible de déduire un ensemble de conflits en terme de phénomènes, de processus puis de rôles fonctionnels.
- A partir des rôles fonctionnels, il est alors possible d'exécuter la méthode de diagnostic de son choix sur un espace de recherche réduit.

Limites de l'approche L'approche multi-modèle, dans son état actuel, présente toutefois certaines limites :

- Même si l'approche apporte une ontologie complète pour représenter les systèmes physiques, l'intervention d'experts du domaine reste nécessaire car la hiérarchie ne peut être établie de manière automatique si les connaissances fondamentales ne sont pas disponibles (structure et comportement).
- L'approche ne s'intéresse qu'à l'étude de la partie opérative d'un système pourvu que celui-ci reste dans un mode de fonctionnement stable. Ceci exclue donc les composants de contrôle/commande (logiciels) pour lesquels nous n'avons en général qu'une connaissance de très haut niveau (machines à états finis, diagrammes fonctionnels, schémas-blocs) et qui reste néanmoins suffisante pour une tâche de diagnostic. L'approche demande donc un certain nombre d'évolutions afin de prendre en compte les dynamiques hybrides et séquentielles omniprésentes dans les systèmes embarqués.

3.4 Méthodes de spécification fonctionnelle des logiciels embarqués

Les approches de la modélisation fonctionnelle présentées dans les sections précédentes sont celles qui sont principalement utilisées pour des tâches de raisonnement. Un certain nombre d'autres méthodes de modélisation sont utilisées de manière opérationnelle dans l'industrie pour la spécification des systèmes à développer. Elles sont souvent qualifiées de *méthodes d'analyse descendantes* car elles permettent, à partir d'une analyse du besoin d'arriver à spécifier une solution technologique.

Dans le cadre du développement des logiciels temps réel, SART et Statemate® sont des exemples de méthodes fréquemment utilisées pour la spécification des fonctionnalités que doivent assurer les calculateurs électroniques. Nous les décrivons succinctement dans la suite de cette section car elles correspondent au type de données de conception que nous pouvons exploiter pour modéliser les organes de commande des systèmes mécatroniques.

3.4.1 La méthode SA-RT

3.4.1.1 Principes de la méthode

La méthode SA-RT (Structured and Analysis - Real Time) est une méthode ancienne issue du génie logiciel et utilisée pour l'analyse fonctionnelle de programmes informatiques temps réel [Ward et Mellor, 1991; Hatley et Pirbhai, 1987]. Le système étudié, le logiciel embarqué, est vu comme une collection de processus traitant des flux d'informations, les *flux de données*, et activés/synchronisés par des flux d'événements, les *flots de contrôle*. Cette méthode s'appuie alors sur des diagrammes de flots de données et des diagrammes de flots de contrôle qui se définissent de la manière suivante :

- un *Diagramme de Flot de Données (DFD)* décrit un traitement sous la forme d'un réseau de processus s'échangeant des données. Il s'agit d'une modélisation fonctionnelle descendante (analyse SA) avec au plus haut niveau le modèle de contexte de données dans lequel le système de pilotage est perçu depuis son environnement comme une entité à part entière. Le système est ensuite décomposé par raffinements successifs où chaque processus se décompose en sous-processus et chaque flot de données en sous-flots de données.
- un *Diagramme de Flot de Contrôle (DFC)* complète et se calque sur le modèle précédent. Son rôle est de définir les modes opératoires (états) du système (en terme d'activation et d'inactivation des processus de données) et les événements.

Les modèles DFD et DFC de la méthode SA-RT sont souvent présentés sur des planches différentes mais ils peuvent très bien être inclus dans un même modèle. La description de plus haut niveau ne ferait apparaître qu'une fonction très générale, le diagramme de contexte de données et de contrôle, liée aux composants / éléments qui lui sont externes : les sources et les puits. Dès le deuxième niveau de description, des processus de contrôle et des processus fonctionnels peuvent cohabiter dans le même diagramme, les processus de contrôle étant utilisés pour expliciter la partie événementielle de la fonction.

Dans la méthode SA-RT, la notion de modèle comportemental correspond à la description des processus de contrôle par des machines à états finis (machines de Mealy[1]) ou des expressions booléennes. Pour des comportements séquentiels plus complexes, l'utilisation de modèles à événements discrets au pouvoir descriptif plus élevé comme les State-Charts et les réseaux de Petri peut être envisagée.

3.4.1.2 Le modèle d'architecture de SA-RT

Le modèle d'architecture permet de tenir compte de contraintes d'architectures logicielles et matérielles lors de la conception du système. Ce modèle, qui peut être assimilé à de la connaissance structurelle, est peu utilisé dans la pratique. Toutefois, il contient quelques concepts intéressants pour la modélisation de la partie commande d'un calculateur électronique. La description du modèle se fait de manière hiérarchique et se présente sous la forme de *diagrammes de flots d'architecture* avec au plus haut niveau le *diagramme de contexte d'archi-*

[1] une machine de Mealy est un automate fini pour lequel les valeurs des variables de sortie dépendent à la fois de l'état courant et des variables d'entrée du système.

tecture. L'entité principale d'un modèle est le module d'architecture. Chaque diagramme de la hiérarchie est décrit selon un canevas type (cf. Figure 3.5) : les modules décrits sont entourés d'une couche d'interfaces comportant l'IHM, le traitement des entrées, le traitement des sorties et l'interface de maintenance et d'auto-diagnostic. Cette représentation rejoint la définition générale des systèmes mécatroniques donnée en section 1.1.1.

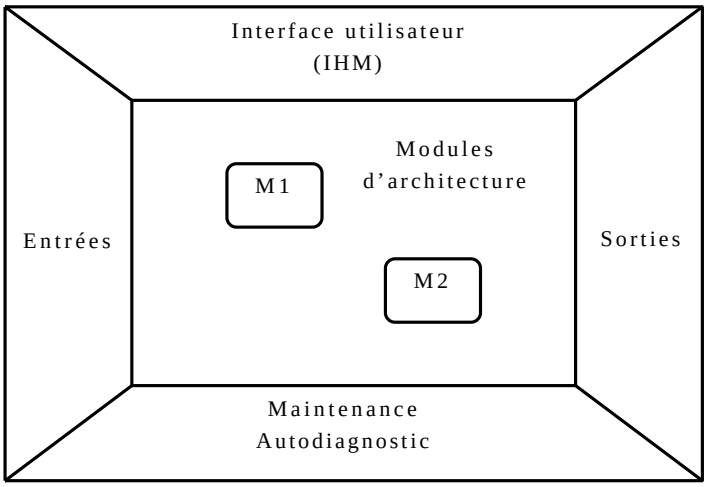


Figure 3.5 — Canevas d'un diagramme du modèle d'architecture

3.4.1.3 Niveau de description de la méthode

La forme originale de la méthode SA-RT se limite à la spécification des systèmes de contrôle temps réel et ne prévoit pas la description fonctionnelle de systèmes dynamiques hybrides. Cependant, l'interaction entre processus de données et processus de contrôle montre qu'une évolution pourrait se faire dans ce sens à l'aide d'automates hybrides, en particulier dans le cadre qui nous intéresse du diagnostic de systèmes hybrides complexes. Le concept de processus physique de l'approche multi-modèle pourrait avoir un statut équivalent au processus de données de la méthode SA-RT, et leur activation serait conditionnée par des processus de contrôle.

3.4.2 La méthode STATEMATE

STATEMATE® est un environnement de développement servant à la spécification, l'analyse, la conception et la documentation de systèmes réactifs complexes tels que les systèmes embarqués, les systèmes de contrôle et de communication [Harel et Naamad, 1996]. Il permet notamment à l'ingénieur de préparer, d'analyser et de déboguer, à l'aide de diagrammes, les descriptions des systèmes en cours de réalisation. Cette analyse se fait selon trois points de vue :

– Le *diagramme de modules* (module-chart) apporte un point de vue structurel en décomposant hiérarchiquement le système en terme de composants physiques appelés *modules* et en identifiant l'*information* qui circule entre eux. Le concept « décomposition physique » de l'approche se veut très général car il peut aussi bien porter sur

un élément matériel pour certains systèmes que sur des sous-routines ou des tâches logicielles pour d'autres systèmes.

- Le *diagrammes d'activités* (activity chart) apporte une décomposition fonctionnelle du logiciel embarqué. Il spécifie les activités, en terme de processus mis en jeu dans la réalisation d'une fonction globale, et les flots d'information (données) circulant entre eux. Nous retrouvons dans ce modèle le même type d'information que dans les diagrammes DFD et DFC de la méthode SA-RT.
- Le *diagramme d'états* (state-chart) apporte un point de vue comportemental du logiciel embarqué en spécifiant le contrôle séquentiel du système mécatronique piloté. Le state-chart [Harel et Naamad, 1996] n'est en fait qu'une extension des machines à états finis qui permet de représenter le comportement événementiel sur plusieurs plans hiérarchiques.

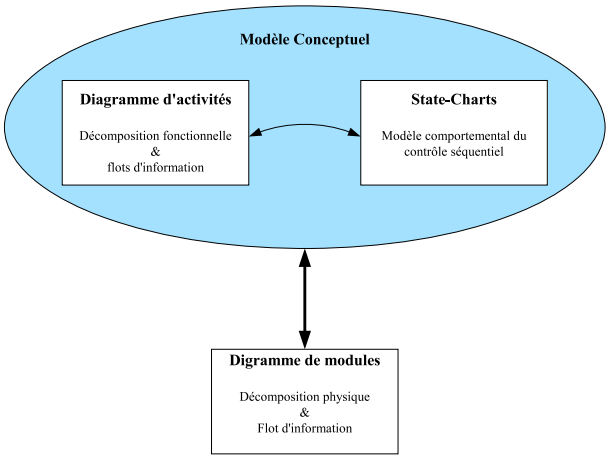


Figure 3.6 — Structure de l'approche STATEMATE

Comme pour la méthode SA-RT, l'approche STATEMATE originale se limite à la description de système à événements discrets mais une extension aux systèmes hybrides est envisageable. Elle semble a priori équivalente à l'approche SA-RT dans la mesure où le diagramme des activités est équivalent au diagramme de flots de données et de flots de contrôle.

3.5 Problèmes spécifiques à la modélisation des systèmes embarqués

Les différentes approches de la modélisation fonctionnelle que nous avons présentées dans ce chapitre montrent que ce domaine de recherche était très actif dans la communauté du diagnostic pendant les années 90. L'idée générale qui ressort est que la connaissance fonctionnelle fait le pont entre la connaissance comportementale et la connaissance téléologique en précisant par exemple les rôles fonctionnels respectifs des sous-systèmes dans l'accomplissement d'une fonction globale. L'automatisation de son acquisition et la définition de primitives suffisamment génériques pour exprimer les concepts utiles pour les descriptions fonctionnelle et téléologique ont été au centre de nombreux travaux. Le développement de cette approche multi-modèle paraît nécessaire pour mener à bien le diagnostic des systèmes

complexes.

Bien que de nombreuses théories formelles permettent actuellement de modéliser les systèmes physiques, peu de travaux traitent des systèmes mêlant des composants logiciels et matériels. Par ailleurs, il n'existe actuellement pas de cadre de modélisation générique concernant les composants logiciels. Ces composants, qui ont un sérieux impact dans le domaine de l'automobile, doivent pourtant faire l'objet d'un diagnostic et nous amènent alors vers un certain nombre de questions posées ci-dessous :

Quelle sont les limites entre fonction et comportement ?

Les méthodes SA-RT et Statemate décrivent le comportement d'une fonction par des machines à états finis. Cette forme de description est une spécification de ce que doit faire la partie commande d'un système mécatronique et est très abstraite vis-à-vis de son implantation réelle sous forme de calculateur électronique. D'autre part, l'approche multi-modèle définit le modèle comportemental comme une connaissance fondamentale à base de lois physiques. Il est donc nécessaire d'expliciter clairement le statut du composant logiciel dans la hiérarchie.

Comment modéliser un système dynamique hybride ?

Le cadre logique actuellement utilisé pour représenter une entité fonctionnelle ne permet pas de décrire les dynamiques séquentielles introduites par la partie commande des systèmes embarqués. En effet, vérifier qu'une fonction est réalisée peut dépendre de l'observation de la valeur que prend une variable de sortie mais aussi d'une séquence de valeurs au cours du temps. Ce problème n'a été abordé que récemment dans le cadre du projet SoftFMEA de l'université d'Aberystwyth [Bell et Snooke, 2004; Bell *et al.*, 2005]. Dans ce projet, l'objectif était d'abstraire les résultats de simulation d'un système embarqué automobile sous la forme de buts réalisés afin de générer automatiquement des rapports d'analyse de conception (AMDEC, SCA,...). Pour décrire les aspects séquentiels et intermittents, J. Bell propose d'utiliser des opérateurs de logique temporelle en complément des opérateurs booléens classiques.

Quels modèles pour le diagnostic hors-ligne ?

Le problème actuel de la modélisation des systèmes automobiles pour le diagnostic après-vente réside dans la disponibilité et l'hétérogénéité des modèles. L'information contenue dans les données de conception sur un composant peut, d'une part, être trop détaillée et trop complexe pour le raisonnement de diagnostic et, d'autre part, être insuffisante pour parvenir à décrire un modèle comportemental de ce composant. Nous verrons par la suite de ce mémoire que les observables disponibles sur le véhicule et la notion de composant remplaçable sont des éléments déterminants dans le choix des modèles.

Deuxième partie

Elaboration d'une stratégie multi-modèle pour le diagnostic

4 Approche de modélisation

L'objectif de ce chapitre est de proposer une approche de modélisation hiérarchique des systèmes mécatroniques embarqués en prenant comme référence les quatre types épistémologiques de l'approche multi-modèle développée par Chittaro. Nous étendons notamment les définitions de ces quatre niveaux de connaissance afin d'inclure la notion de composant logiciel et de structurer les connaissances hétérogènes dont nous disposons sur ces systèmes. Les modèles exposés dans ce chapitre se limitent à la description du comportement nominal, la prise en compte des modes de défaut est discuté dans le chapitre 5. Nous illustrons cette approche sur un système réel, le système « essuie-vitre arrière », que nous avons simplfié dans un souci de clarté.

4.1 Problème de construction des modèles pour la simulation

Dans cette section, nous définissons de manière générale la représentation multi-modèle que nous utilisons pour la modélisation des systèmes mécatroniques embarqués. Nous identifions ensuite les modèles nécessaires à la simulation de ces systèmes et le problème lié à leur construction. Enfin, nous donnons une présentation générale du système « essuie-vitre ».

4.1.1 Plan de représentation multi-modèle

Dans le chapitre précédent (cf. section 3.2), nous avons présenté deux axes d'abstraction possibles pour décrire le modèle hiérarchique d'un système : les abstractions par agrégations et les abstractions fonctionnelles. Comme dans les travaux de Bell [Bell, 2006], nous choisissons de combiner ces deux axes et de les représenter sous la forme d'une grille, appelée *plan de représentation multi-modèle*, afin de structurer la connaissance dont nous disposons sur les systèmes mécatroniques embarqués. L'axe horizontal de ce plan représente l'axe des abstractions fonctionnelles selon les quatre types épistémologiques définis plus en détail dans les sections suivantes et l'axe vertical représente l'axe des abstractions par agrégations que nous divisons en quatre autres niveaux :

- **Niveau groupe de prestations** : ce niveau est le plus haut de la hiérarchie. Un groupe de prestations agrège des systèmes qui se rapportent au même domaine de prestations du véhicule. Par exemple, le système essuie-vitre arrière fait parti du groupe de

prestations « visibilité » (cf. Figure 1.5).

- **Niveau système** : un système est un ensemble de composants inter-connectés qui réalisent une fonction ou plusieurs fonctions si celles-ci portent sur des aspects connexes. Par exemple, le système « essuie-vitre arrière » réalise les fonctions « essuyer la lunette arrière lors d'une demande utilisateur » et « essuyer la lunette arrière lorsque l'essuie-vitre avant est activé et la marche arrière enclenchée ».
- **Niveau module / sous-système** : les modules sont des parties identifiables d'un système en terme de séparation physique. Pour notre problème de diagnostic, un module correspond à un composant physique remplaçable (ex : calculateur habitacle, module essuie-vitre, calculateur des commandes au volant...).
- **Niveau composants élémentaire** : le niveau composant élémentaire constitue le niveau le plus bas de la hiérarchie d'agrégation. La décomposition d'un module en composants élémentaires permet d'une part, de décomposer un module en composants plus génériques et donc réutilisables, et d'autre part, de séparer la partie logicielle de la partie matérielle lorsque le module est un calculateur électronique.

Ces niveaux sont propres à l'architecture des systèmes embarqués que nous étudions mais le découpage proposé peut très bien être réalisé autrement pour d'autres systèmes. Comme précisé dans le chapitre précédent, l'orthogonalité des ces axes n'est pas à prendre dans le sens strict car certains modèles fonctionnels, comme les processus de Forbus [Forbus, 1984], peuvent se détacher de la notion de structure.

Les éléments contenus dans les cellules du *plan de représentation multi-modèle* et les flèches qui les relient se lisent de la manière suivante :

- Les rectangles à bordure continue correspondent aux modèles qui doivent être décrits par l'expert.
- Les flèches continues donnent le sens de construction des différents modèles et sont étiquetés par la nature de la transformation de la connaissance qui est effectuée.
- Le rectangle à bordure discontinue correspond au modèle de simulation numérique et est implicite car il s'agit d'une représentation compilée du comportement du système propre à l'outil de simulation.

4.1.2 Connaissance nécessaire à la simulation

La simulation d'un système complexe permet de prévoir son comportement au cours du temps sous différentes situations à des fins d'analyse. Cette simulation peut être ensuite interprétée en terme de fonctions réalisées grâce au modèle fonctionnel qui relie comportement et fonction. Dans le cas du diagnostic hors-ligne, tel que nous l'abordons, nous verrons dans le chapitre 5 que la simulation est nécessaire afin de prévoir les résultats des tests sous l'occurrence de différents modes de défaut.

Pour simuler un système, il est nécessaire de disposer d'un modèle comportemental de chaque composant qui le composent. Cependant, cette connaissance n'est pas disponible de la même manière selon le type de composant modélisé :

- pour les composants physiques formant la partie opérative, le modèle comportemental est généralement obtenu après analyse de son modèle structurel. Chaque compo-

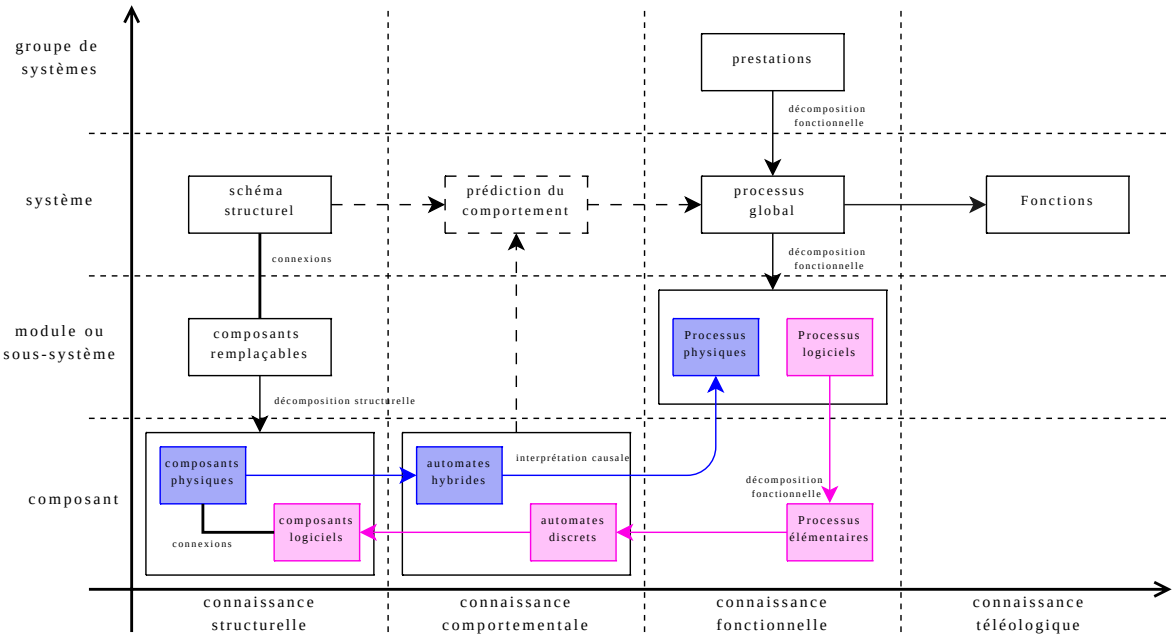


Figure 4.1 — Plan de représentation multi-modèle

sant élémentaire du modèle structurel est une instance d'un composant physique générique (résistance, interrupteur, fil, moteur électrique...). Le comportement est abstrait ensuite sous la forme de processus. La démarche de modélisation est alors ascendante.

– pour les composants de la partie commande, le problème est différent. En effet, la connaissance disponible sur ces composants provient des spécifications fonctionnelles des calculateurs (Modèles SA-RT ou Statemate). Ces spécifications font apparaitre les modèles de comportement souhaités par le concepteur et sont donc à un haut niveau d'abstraction vis-à-vis de l'implémentation physique réelle. Un comportement est souvent décrit par des machines à états finis pour la modélisation de lois de commandes discrètes mais se présente parfois sous forme de phrases dans un document textuel, ne décrivant que des propriétés sur les entrées et les sorties du composant de pilotage modélisé. Cette connaissance est souvent suffisante car ces propriétés portent en général sur les seuls observables dont nous disposons pour tester les composants de commande.

La représentation multi-modèle des systèmes mécatroniques nécessite donc de faire cohabiter des composants et des modèles de nature très différente.

4.1.3 Présentation du système essuie-vitre arrière

Nous présentons dans cette section l'architecture générale du système essuie-vitre arrière qui nous a servi de fil conducteur durant notre travail. Même si nous avons procédé à certaines simplifications du système réel par souci de clarté, nous montrons que l'exemple est très représentatif de ce qu'est un système mécatronique embarqué actuel.

Synoptique du système Le synoptique du système donné Figure 4.2 est composé de deux calculateurs électroniques (`ECU_hdc` et `ECU_hab`) communiquant via le réseau `CAN_car1`, d'un commutateur essuyage arrière (`CEAR`) et d'un module essuie-vitre arrière (`EVAR`). Les rôles assurés par les calculateurs se définissent de la manière suivante :

- `ECU_hdc` est « le calculateur des commandes au volant ». Pour ce système, il acquiert et filtre la position du commutateur essuyage et transmet l'information à `ECU_hab` via le réseau `VAN_car1`.
- `ECU_hab` est le « calculateur habitacle ». Il gère le fonctionnement général du système et commande directement le moteur essuie-vitre arrière `EVAR`. Il acquiert et filtre également le signal `arret_fixe_ar` provenant du capteur de position repos du module `EVAR`.
- `ECU_hab` gère d'autres systèmes qui interagissent avec l'essuyage arrière comme l'essuyage avant et l'état du capteur de la position « marche-arrière » du levier de vitesse.

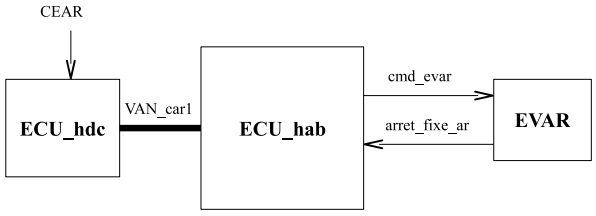


Figure 4.2 — Schéma synoptique du système essuie-vitre

Contextes d'activation du système Nous considérons deux contextes d'activation différents du système :

- activation de l'essuie-vitre arrière lors d'une mise sur position 2 du commutateur par l'utilisateur. L'essuyage est intermittent : le mouvement du balai observé est périodique où une période est composée d'un cycle de balayage [1] et d'un temps de pause en position repos.
- activation de l'essuie-vitre arrière lorsque l'essuie-vitre avant est actif et lorsque la marche arrière est enclenchée par l'utilisateur. L'essuyage est intermittent.

4.2 La connaissance structurelle

4.2.1 Définitions

La connaissance structurelle décrit la topologie du système, c'est à dire l'ensemble des composants physiques et logiciels qui constituent le système mécatronique ainsi que la manière dont ces composants sont connectés entre eux. Comme un système mécatronique se décompose en une partie commande et une partie opérative, la description de son modèle structurel consistera à décrire chacune de ces deux parties et comment elles interagissent.

[1] un cycle de balayage correspond à un mouvement aller-retour du ballai essuie-vitre

Modèle structurel de la partie opérative

Le modèle structurel de la partie opérative se décrit à partir de deux primitives : les composants matériels et les connexions physiques.

Les composants matériels Les composants matériels sont les objets physiques qui composent le système. Ils disposent de points d'échange appelés *ports* leur permettant d'interagir avec leur environnement. Ces ports sont de différents types :

- Les *ports physiques* permettent à un composant d'échanger des quantités physiques avec son environnement. L'échange réalisé à travers un port physique ne peut être que d'une seule nature (électrique, thermique, mécanique de rotation etc.) et est généralement associé à une variable de flux et une variable d'effort. Par exemple, un port électrique transmet un potentiel (effort) et est traversé par un courant (flux).
- Les *signaux d'entrée et de sortie* sont des ports qui permettent à un composant physique de recevoir des ordres de la partie commande (signal d'entrée) ou de lui retourner des données mesurées (signaux de sortie). Ces données peuvent être de type réel (mesure de température), booléen (commande d'interrupteur) ou énuméré (commutateur à plusieurs positions).

Les connexions physiques Les connexions physiques permettent de décrire comment les composants physiques interagissent en établissant des relations d'égalité entre plusieurs ports du même type. Ces connexions donnent notamment lieu à l'écriture de lois de conservation portant sur les variables de flux et d'effort (ex : lois Kirchhoff dans le domaine électrique) lors de la mise en équation de la partie opérative.

Modèle structurel de la partie commande

Le modèle structurel de la partie commande se décrit à partir de deux primitives : les composants logiciels et les connexions de signaux.

Les composants logiciels Les composants logiciels sont des objets qui représentent les différents traitements qui sont effectués par les calculateurs électroniques. Ils disposent de signaux d'entrée et de sortie et de bus de données qui leur permettent d'interagir avec d'autres composants logiciels et avec la partie opérative. Un bus de données est un port contenant plusieurs signaux pouvant être de différent type (booléen, continu ou énuméré).

Les connexions de signaux Les connexions de signaux permettent de relier plusieurs composants logiciels entre eux et de connecter la partie commande avec la partie opérative. Elle peut être simple lorsqu'elle se fait entre un signal de sortie d'un composant et un signal d'entrée d'un autre composant ou multiple entre un signal de sortie et plusieurs signaux d'entrée.

4.2.2 Construction d'une hiérarchie structurelle

Le niveau le plus haut de la hiérarchie est le niveau système ; il identifie les modules physiques remplaçables qui le composent. Ce modèle peut être extrait directement des schémas de câblage (ou des « Netlists ») des véhicules. Cependant, cette donnée de conception n'apporte pas d'information sur le flux transformé d'un système mécatronique (cf. Figure 1.1). Par exemple, le schéma de câblage d'un système de climatisation représente les calculateurs, capteurs, actionneurs et câbles de connexion qui le composent mais pas le sous-système « circuit de refroidissement ».

L'objectif de la hiérarchie est d'affiner ensuite la description des différents modules pour les décomposer en composants élémentaires. En ce qui concerne les calculateurs électroniques, la décomposition se fait en trois étapes :

1. La première étape consiste à séparer la partie commande du calculateur de la partie physique contenant les composants électroniques de puissance tels que les interrupteurs commandés et les fusibles. Pour un calculateur, nous utilisons les préfixes `Soft_ECU` pour nommer la partie commande et `Hard_ECU` pour la partie physique.
2. La seconde étape consiste à décrire la partie physique à partir d'une bibliothèque de composants électriques génériques.
3. La troisième étape consiste à décrire la partie commande à partir de composants logiciels. Les composants logiciels ne sont en général pas génériques car ils constituent le système de pilotage et dépendent donc du système mécatronique modélisé. Cependant, un cadre générique peut être adopté comme celui proposé par le modèle d'architecture de la méthode SA-RT (figure 3.5). La granularité du modèle structurel de la partie commande est fixée en fonction des signaux d'entrées et de sorties disponibles via l'outil de diagnostic (cf. section 1.2.3).

Connaissance structurelle du système essuie-vitre arrière

Le niveau système Le modèle structurel du système essuie-vitre arrière est donné par la Figure 4.3. Celui-ci reprend les trois principaux modules décris dans le synoptique précédent ainsi que les éléments qui permettent de les relier ; c'est-à-dire les fils {`Fil_1`, `Fil_2`, `Fil_3`} et les connecteurs {`Con_1`, `Con_2`, `Con_MC`}.

Ce modèle est ensuite décomposé suivant la méthodologie présentée dans la section précédente. L'arbre de décomposition hiérarchique de la connaissance structurelle est donné par la figure 4.4. Le parcours de cet arbre permet de déterminer les liens d'agrégation entre chaque niveau de représentation : les éléments encadrés représentent les liens hiérarchiques entre les ports d'un composant père et les ports d'un composant fils. Par exemple, le lien `ECU_hab.p_bat` $\longleftrightarrow$ `Hard_ECU_hab.p_bat` donne le lien de correspondance entre le port électrique `p_bat` du composant père `ECU_hab` et le port électrique `p_bat` du composant fils `Hard_ECU_hab`.

Une légende décrivant les différents symboles graphiques utilisés concernant les ports est donnée en annexe (cf. annexe A).

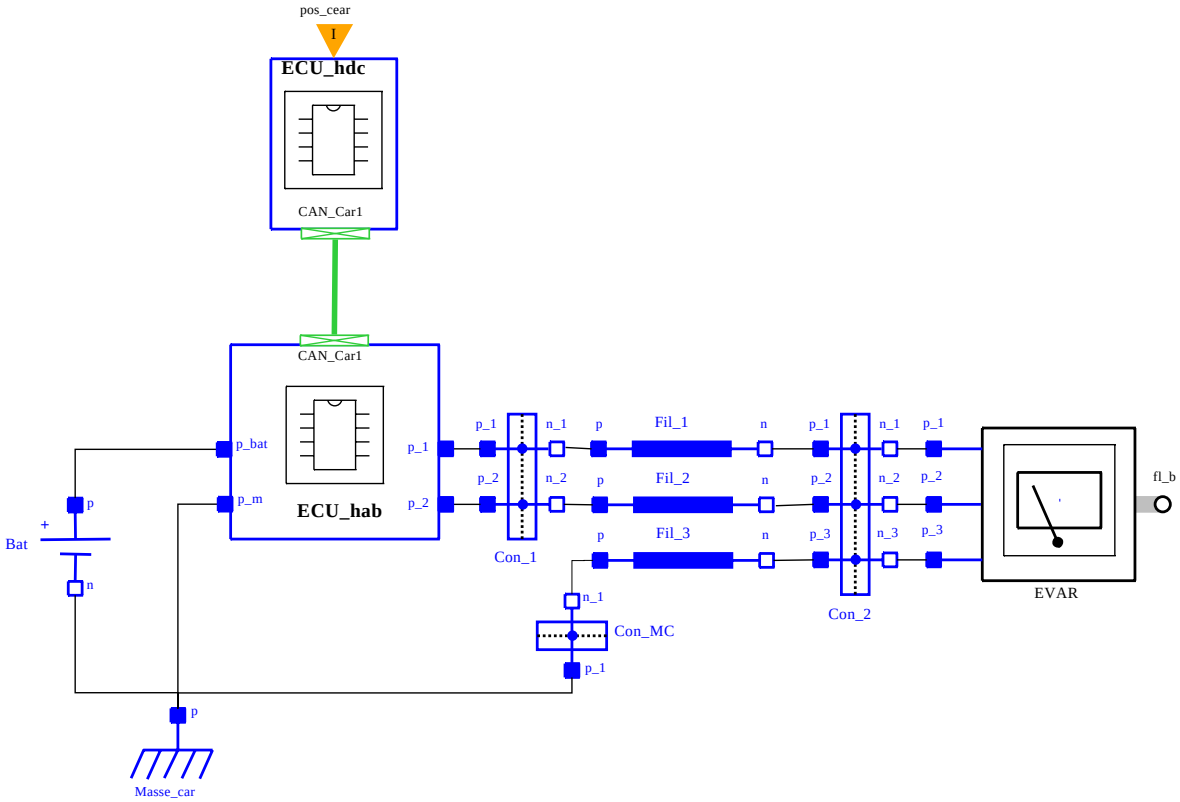


Figure 4.3 — Modèle structurel du système essuie-vitre arrière

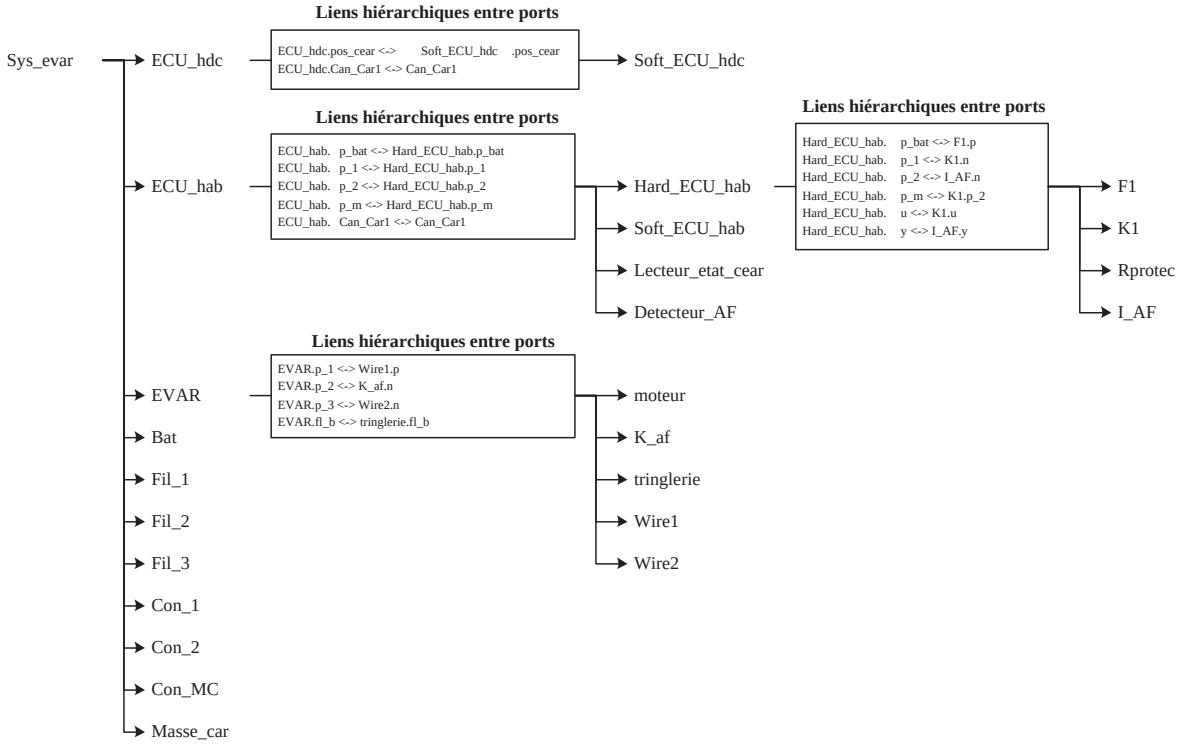


Figure 4.4 — Arbre de la hiérarchie structurelle

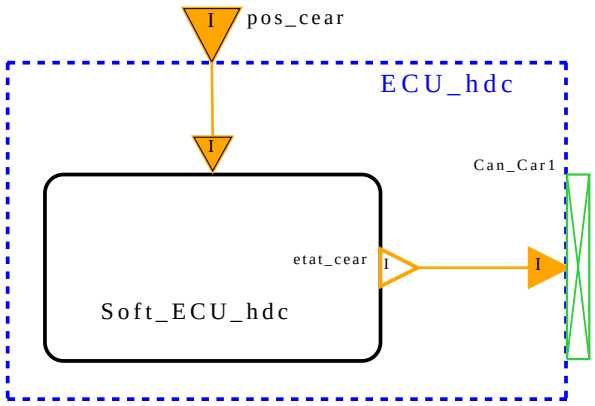


Figure 4.5 — Décomposition structurelle du calculateur `ECU_hdc`

Décomposition structurelle du calculateur `ECU_hdc` La décomposition structurelle du calculateur `ECU_hdc` est donnée par la figure 4.5. Le commutateur d'essuyage n'est pas modélisé explicitement en tant que composant matériel car il est complètement intégré au calculateur `ECU_hdc` et aucun observable électrique n'est disponible. Il est donc réduit au signal d'entrée de type énuméré *pos_cear* pouvant prendre deux valeurs {1,2}. Le seul observable disponible concernant ce composant est le signal de sortie de type énuméré *etat_cear*, pouvant prendre les valeurs {*dmd_repos_ar*, *dmd_ess_ar*, *etat_invalide*}. Il est le résultat de la lecture de la position sélectionnée du commutateur à envoyer sur le bus de données `Can_Car1`. La décomposition structurelle de ce calculateur se limite alors à la description du composant logiciel `Soft_ECU_hdc`.

Sur la figure 4.5, le cadre en ligne discontinue représente le composant père `ECU_hdc` avec ses ports. Ce même principe de représentation graphique a été utilisé pour la décomposition structurelle des autres modules afin d'en faciliter la lecture.

Décomposition structurelle du calculateur `ECU_hab` La décomposition structurelle du calculateur habitacle est réalisée sur deux niveaux. Le premier niveau, donné par la figure 4.6 décompose la partie logicielle de la partie matérielle du calculateur. Le second niveau décompose la partie matérielle en composants élémentaires figure 4.7. La partie logicielle contient trois composants :

- le composant `Lecteur_etat_cear` lit le message *etat_cear* provenant du bus `CAN_Car1`.
- le composant `Detecteur_AF` lit l'entrée du signal de courant provenant du circuit du capteur de position repos du module `EVAR`.
- le composant `Soft_ECU_hab` gère le fonctionnement du système, c'est lui qui implémente la loi de commande discrète qui détermine l'activation de l'interrupteur `K1` de la partie matérielle `Hard_ECU_hab` du calculateur.

Les signaux entrée/sortie booléens du logiciel de commande `Soft_ECU_hab` sont définis dans le tableau 4.1.

Décomposition structurelle du module `EVAR` La décomposition structurelle du module essuie-vitre `EVAR` est donnée par la figure 4.8. Il est composé d'un moteur électrique, d'un

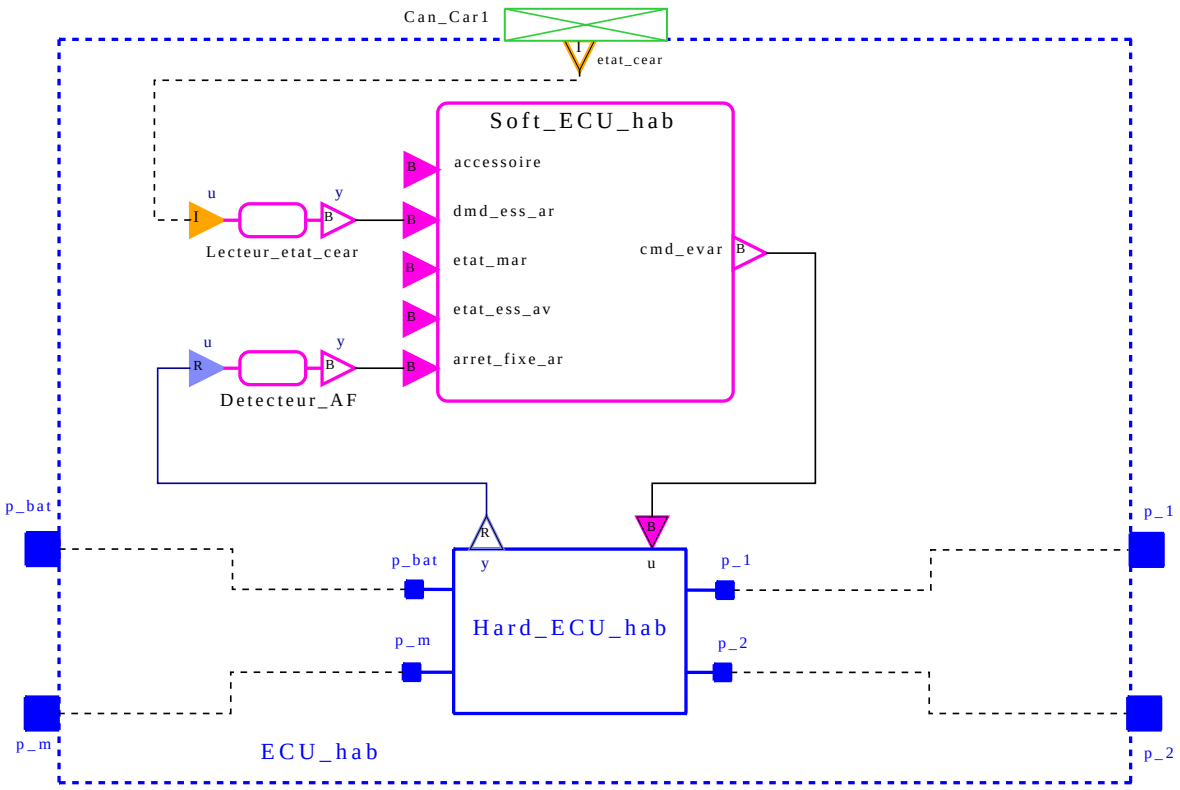


Figure 4.6 — Décomposition structurelle du calculateur `ECU_hab`

Signal Booléenl	Définition
accessoire	clé de contact sur position accessoire
dmd_ess_ar	demande essuyage arrière
etat_mar	marche arrière activée
etat_ess_av	essuyage avant activé
arret_fixe_ar	balais arrière en position repos
cmd_evar	commande de l'actionneur

Tableau 4.1 — Définition des signaux du composant logiciel `Soft_EVAR`

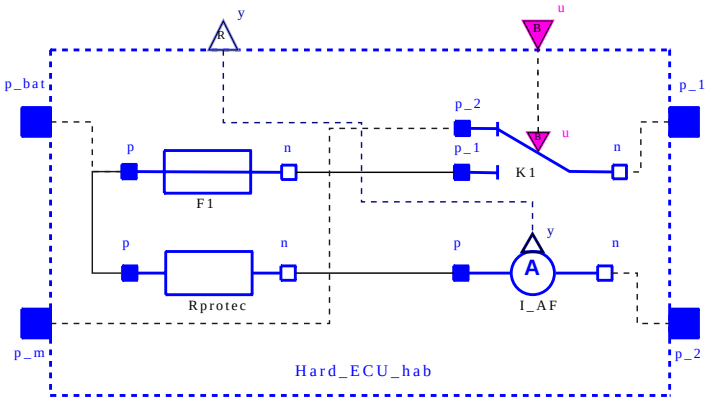


Figure 4.7 — Décomposition structurelle de la partie puissance `Hard_ECU_hab` du calculateur `ECU_hab`

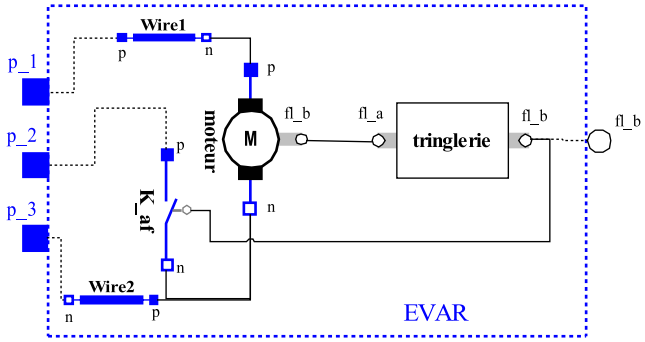


Figure 4.8 — Décomposition structurelle du module `EVAR`

système de tringlerie permettant de transformer le mouvement de rotation continue du moteur en mouvement alternatif des balais et du capteur de position `K_af` permettant de repérer la position repos du balai.

4.3 La connaissance comportementale

Dans une approche orientée composant, la connaissance comportementale d'un système décrit comment les composants se comportent individuellement et interagissent en terme de quantités qui caractérisent leur état et en terme de lois qui régissent l'évolution de cet état au cours du temps. La notion de quantité peut se référer à une quantité physique (variable physique, paramètre) ou à un signal.

Comme un système est un ensemble de composants et de connexions entre composants, son modèle comportemental se divise en deux parties :

- les *équations structurelles* : elles peuvent se traduire par des égalités entre signaux (composants logiciels) ou des lois de conservation (connexions entre ports physique).
- le *modèle des composants*, qui se représente par un automate hybride pour un composant matériel ou par un automate discret pour un composant logiciel.

4.3.1 Modèle comportemental d'un composant matériel

Pour les composants physiques, nous reprenons le cadre de modélisation proposé dans [Olive, 2003]. Ainsi, le modèle comportemental d'un composant physique inclut plusieurs *modes* décrivant les différents contextes opératoires. Des conditions régissent les transitions qui peuvent s'effectuer d'un mode à l'autre. Plus formellement, le modèle comportemental d'un composant physique est représenté par un automate hybride [Cocquempot *et al.*, 2005; Bayoudh *et al.*, 2007] $H = (X, Q, \Gamma, T, C, (X_0, q_0))$ où :

- X est l'ensemble des variables physiques continues du composant
- Q est l'ensemble des modes du composant. Chaque mode $q_i \in Q$ peut représenter un mode nominal ou un mode de défaut du composant.
- Γ est l'ensemble des conditions de garde. Cet ensemble contient des conditions commandables (action de fermeture/ouverture d'un interrupteur) et des conditions non

commandables (tension seuil d'une diode, seuil d'intensité d'un fusible).
- $T : Q \times \Gamma \rightarrow Q$ est la fonction des transitions entre modes du composant
- C est l'ensemble des équations du composant. On associe un sous ensemble $C^{q_i} \subseteq C$ à chaque mode q_i
- $(x_0, q_0) \in X \times Q$ est la condition initiale de l'automate hybride

Exemple du composant `tringlerie` du module `EVAR`

Par exemple, le modèle comportemental du composant mécanique `tringlerie` du module `EVAR` peut être approché par l'automate hybride donné en figure 4.9. Ce modèle est composé de deux modes nominaux, le mode *BF_1* représente le « balayage aller » et le mode *BF_2* représende le « balayage retour ». L'automate hybride $H = (X, Q, \Gamma, T, C, (X_0, q_0))$ de ce composant se définit alors de la manière suivante :

- $X = \{\theta_a, \omega_a, \gamma_a, \theta_b, \omega_b, \gamma_b\}$ avec θ_a et θ_b les positions angulaires au niveau des ports mécaniques de rotation `fl_a` et `fl_b` , ω_a et ω_b les vitesses de rotations, γ_a et γ_b les moments mécaniques.
- $Q = \{BF\_1, BF\_2\}$
- $\Gamma = \{\gamma_1, \gamma_2\} = \{(\theta_b \geq \theta_{max}), (\theta_b \leq \theta_0)\}$, θ_{max} et θ_0 étant respectivement les positions angulaires maximum et au repos du balais « essuie-vitre ».
- Il existe deux transitions T_1 et T_1 où :

  $T_1 : (BF\_1, g1) \stackrel{T}{\rightarrow} BF_2$

  $T_2 : (BF\_2, g2) \stackrel{T}{\rightarrow} BF_1$
- $C = \{C_1^{all}, C_2^{all}, C_3^{all}, C_4^{all}, C_5^{BF\_1}, C_6^{BF\_2}\}$. L'exposant all signifie que l'équation correspondante est active dans tous les modes. Les différentes équations se définissent de la manière suivante

  $C_1^{all} : \gamma_a = -Rf.\omega_a$ avec Rf le coefficient de frottement mécanique

  $C_2^{all} : \gamma_b = 0$

  $C_3^{all} : \omega_a = \frac{d\theta_a}{dt}$

  $C_4^{all} : \omega_b = \frac{d\theta_b}{dt}$

  $C_5^{BF\_1} : \omega_b = k.\omega_a$ avec k le rapport de réduction de vitesse

  $C_6^{BF\_2} : \omega_b = -k.\omega_a$

4.3.2 Modèle comportemental d'un composant logiciel

En faisant l'hypothèse que les calculateurs électroniques implémentent des commandes séquentielles nous assimilons les composants logiciels à des systèmes à événements discrets. Ainsi, leur modèle comportemental peut être représenté sous la forme de machines à états finis tels que les automates. Dans ce manuscrit, nous décrivons des machines de Moore pour lesquelles les sorties sont exprimées en fonction de l'état courant. Un automate de Moore se défini par un 6-uplet, $A = (S, U, Y, T, G, S_0)$, constitué de :

- S est l'ensemble fini des états discrets

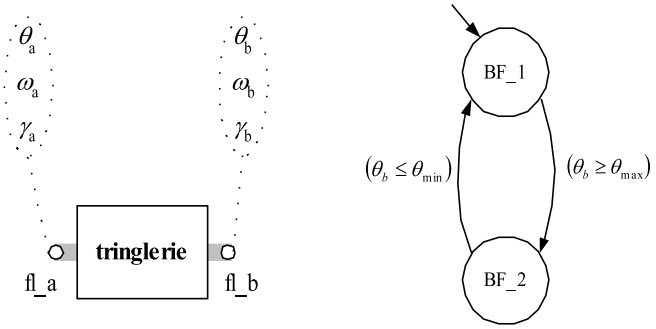


Figure 4.9 — Modèle comportemental de la tringlerie

- U est l'ensemble fini des entrées (alphabet d'entrée)
- Y est l'ensemble fini des sorties (alphabet de sortie)
- $T : S \times U \rightarrow S$ est la fonction des transitions entre états de l'automate. Les transitions sont gardées par des formules logiques portant sur des conditions d'entrée.
- $G : S \rightarrow Y$, la fonction de sortie dépendant de l'état actif
- S_0 est l'état initial

Par la suite nous adoptons la syntaxe suivante pour l'écriture de conditions portant sur une variable booléenne b :

- l'écriture b correspond à $b = 1$
- l'écriture $\neg b$ correspond à $b = 0$
- l'écriture $\uparrow b$ correspond à un font montant sur b
- l'écriture $\downarrow b$ correspond à un front descendant sur b

Comportement des composants logiciels du système essuie-vitre

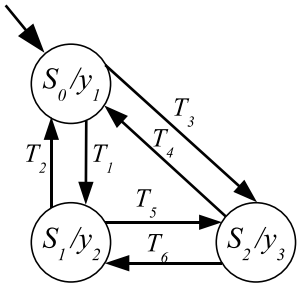
Les automates décrivant les comportements des composants logiciels sont donnés par la figure 4.10 pour le composant `Soft_ECU_hdc` du calculateur `ECU_hdc`, par la figure 4.12 pour les composants `Lecteur_etat_cear` et `Detecteur_AF` du calculateur `ECU_hab`. L'automate de commande implémentée par le composant `Soft_EVAR` est donné par la figure 4.11 et contient 5 états définis de la manière suivante :

- S_0 : « essuie-vitre arrière non activable »
- S_1 : « essuie-vitre arrière à l'arrêt et activable »
- S_2 : « essuie-vitre arrière activé »
- S_3 : « essuie-vitre arrière en repos intermittent »
- S_4 : « protection essuie-vitre arrière »

4.4 La connaissance fonctionnelle

4.4.1 Définitions

Comme nous l'avons vu dans le chapitre précédent, la connaissance fonctionnelle d'un système se définit comme une passerelle entre son comportement et les fonctions (buts, be-



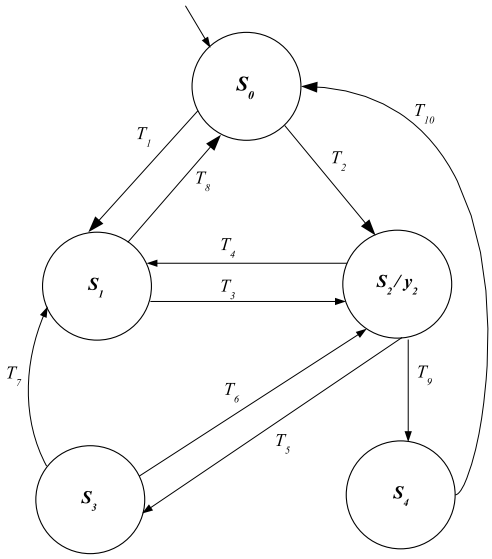
Conditions de garde des transitions

$T_1 : \neg b_1 \wedge b_2$
$T_2 : b_1 \wedge \neg b_2$
$T_3 : (b_1 \wedge b_2) \vee (\neg b_1 \wedge \neg b_2)$
$T_4 : b_1 \wedge \neg b_2$
$T_5 : (b_1 \wedge b_2) \vee (\neg b_1 \wedge \neg b_2)$
$T_6 : \neg b_1 \wedge b_2$

Définition des actions

y_1 : *(etat_cear = dmd_repos_ar)*
y_2 : *(etat_cear = dmd_ess_ar)*
y_3 : *(etat_cear = etat_invalide)*

Figure 4.10 — Modèle comportemental du composant logiciel `Soft_ECU_hdc`. b_1 et b_2 sont deux variables booléennes correspondant respectivement aux conditions $pos\_cear = 1$ et $pos\_cear = 2$



Conditions de garde des transitions

T_1 : *accessoire* $\wedge$ *arret_fixe_ar*
T_2 : *accessoire* $\wedge \neg$ *arret_fixe_ar*
T_3 : *dmd_ess_ar* $\vee$ (*etat_ess_av* $\wedge$ *etat_mar*)
T_4 : $\uparrow$ *arret_fixe_ar* $\wedge \neg$ *dmd_ess_ar* $\wedge \neg$ (*etat_ess_av* $\wedge$ *etat_mar*)
T_5 : $\uparrow$ *arret_fixe_ar* $\wedge$ (*dmd_ess_ar* $\vee$ (*etat_ess_av* $\wedge$ *etat_mar*))
T_6 : ($C \geq 12$ sec) $\wedge$ (*dmd_ess_ar* $\vee$ (*etat_ess_av* $\wedge$ *etat_mar*))
T_7 : $\neg$ *dmd_ess_ar* $\wedge \neg$ (*etat_ess_av* $\wedge$ *etat_mar*)
T_8 : $\neg$ *accessoire*
T_9 : ($C \geq 7$ sec)
T_{10} : $\neg$ *accessoire*

Définition des actions

y_2 = *cmd_evar*

Figure 4.11 — Modèle de la loi de commande du système essuie-vitre

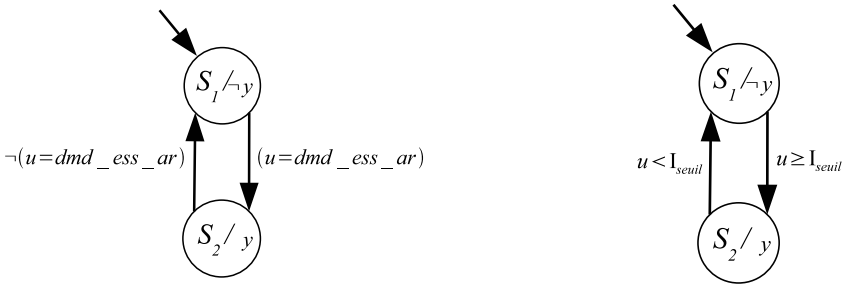


Figure 4.12 — Comportement des composants `Lecteur_etat_cear` (à gauche) et `Detecteur_AF` (à droite)

soins) prévues par son concepteur. Le modèle que nous représentons pour caractériser cette connaissance est le modèle des processus fonctionnels[2]. La notion de processus est en effet commune aux différentes approches de modélisation fonctionnelle pour le diagnostic et elle se retrouve dans la méthode de spécification SA-RT.

Nous représentons alors la connaissance fonctionnelle par une hiérarchie de processus. Le plus haut niveau de la hiérarchie identifie le processus global du système (prestation) dans son environnement. Par exemple, la prestation « essuyage-arrière » (cf. figure 4.13) interagit avec l'utilisateur, la prestation « essuyage-avant », la prestation « alimentation » et le capteur de marche arrière.

La prestation est ensuite décomposée en processus élémentaires. Nous distinguons deux classes de processus pouvant cohabiter sur un même niveau fonctionnel : les *processus logiciels* et les *processus physiques*.

Les processus physiques Les processus physiques représentent les différentes transformations qui ont lieu sur les variables physiques de la partie opérative d'un système mécatronique. Un processus physique correspond en général à un domaine de la physique particulier dans l'approche de Chittaro (électrique, mécanique, hydraulique...). Toutefois, nous décrivons aussi des processus électromécaniques, mélangeant des aspects électriques et mécaniques, afin d'abstraire le comportement d'un actionneur par exemple.

Le modèle des processus physiques représente explicitement comment s'influencent les processus entre eux. Les influences sont issues d'une interprétation causale des modèles comportementaux sous-jacents [Dague et Travé-Massuyès, 2003].

Par exemple, nous décrivons deux processus physiques pour le système essuie-vitre :
- Le processus électromécanique `Pelm_1` est issu de l'ensemble des relations physiques (électriques et mécaniques) qui caractérisent la partie actionneur du système. Ce sont celles qui relient la présence d'une tension (*tension_batterie*) aux bornes de la batterie `Bat` et la présence de la consigne d'activation *cmd_evar* à l'observation d'un mouvement des balais caractérisé par la vitesse *omega_balais* en sortie du moteur (au niveau du port mécanique `EVAR.fl_b`)
- Le processus `Pelec_2` est issu l'ensemble des relations relatives au circuit électrique alimentant le capteur de position repos de module `EVAR`.

Les processus logiciels Les processus logiciels représentent les différents traitements effectués par la partie commande d'un système mécatronique. La description d'un modèle des processus logiciels reprend les trois primitives de description de la méthode SA-RT :
- Les *flots de données* correspondent aux signaux de type énuméré ou réel définis dans le modèle structurel et comportemental.
- Les *flots de contrôle* correspondent aux signaux booléens.
- Les *processus logiciels* peuvent représenter une transformation de donnée en une nouvelle donnée (*processus de donnée*), un stockage de donnée pouvant être partagé par

[2]Attention, nous réservons le terme de fonction pour la connaissance téléologique pour exprimer des buts/besoins. Ceci peut prêter parfois à confusion avec le notion de connaissance fonctionnelle que nous définissons dans cette section.

d'autres processus logiciels (*processus de stockage*) ou le contrôle séquentiel d'un système (*processus de contrôle*).

Par exemple, la décomposition de la prestation essuyage arrière, donnée par la figure 4.14, fait apparaître pour la partie logicielle trois processus de données {`Pdat_1`, `Pdat_11`, `Pdat_12`}, un processus de contrôle `Pctr_1` et un processus de stockage du message *etat_cear*(représenté par deux traits parallèles sur la figure). Les flèches avec une ligne continue représentent les flots de données et les flèches avec une ligne discontinue représentent les flots de contrôle. Le sens des flèches donne le sens de l'influence d'un processus sur un autre.

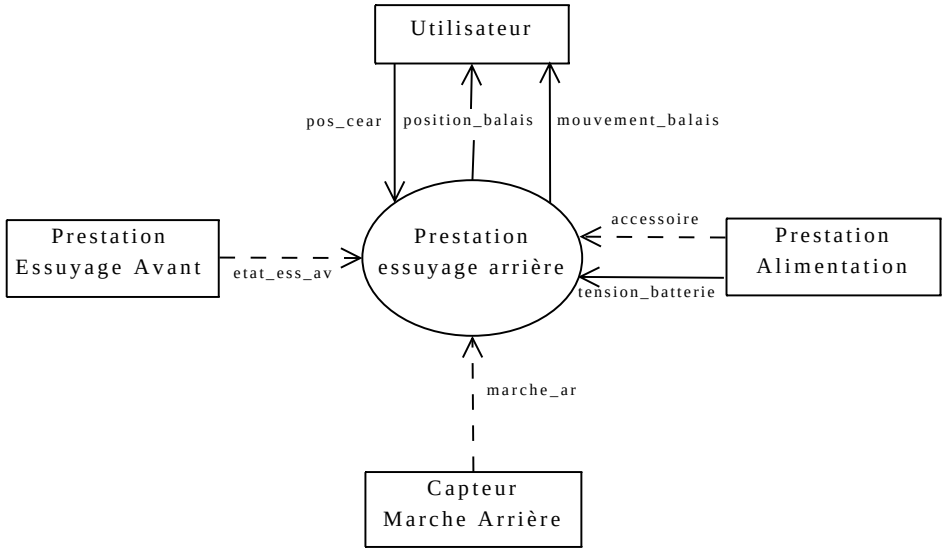


Figure 4.13 — Modèle de la prestation essuyage arrière

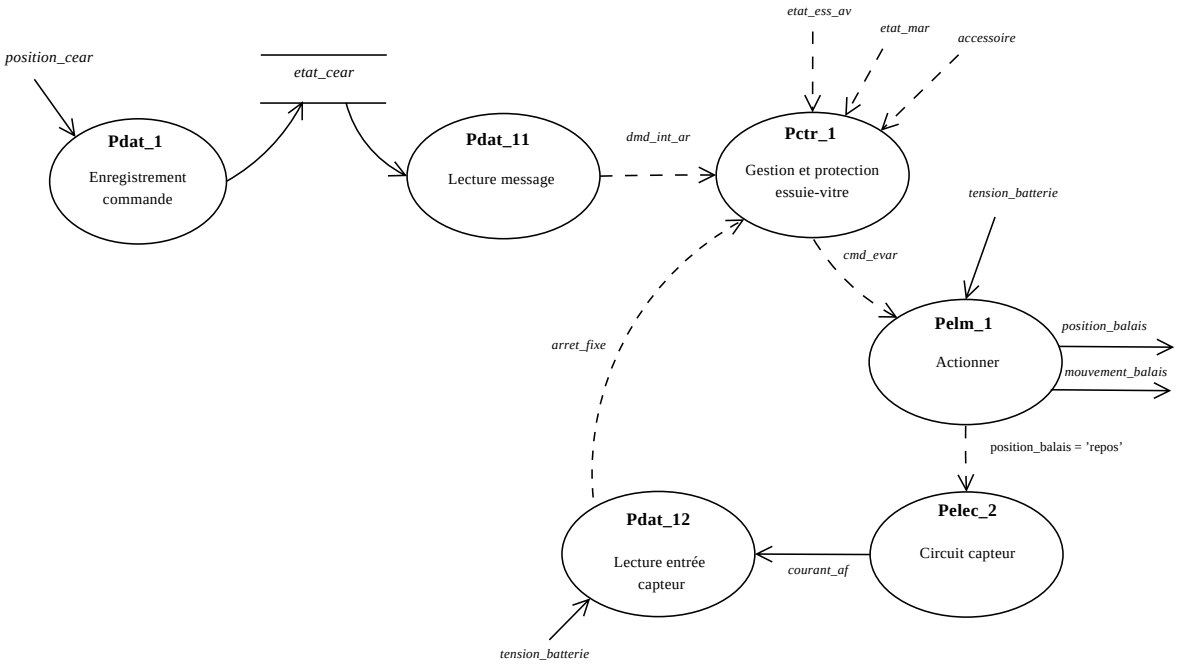


Figure 4.14 — Décomposition fonctionnelle de la prestation

4.4.2 Etat fonctionnel d'un processus

Pour réaliser des tâches de raisonnement (vérification de conception, diagnostic), un processus peut être associé à un état fonctionnel qui peut avoir une des deux valeurs : « actif », « inactif ». Cet état est caractérisé par les pré-conditions et les effets (cf. section 3.2.2.3) portant respectivement sur les entrées et les sorties d'un processus. Ainsi, si les pré-conditions et les effets sont vrais, le processus est actif, sinon il est inactif. Cet état fonctionnel est représenté pour les processus physiques et les processus de données et le processus global qui représente le système étudié. Les propriétés liées aux processus de contrôle ne sont prises en compte que dans le processus global du système.

Pour l'écriture des pré-conditions et des effets, nous définissons un ensemble de propositions logiques à partir des variables du modèle comportemental. La syntaxe utilisée pour décrire ces propositions rejoint celle proposée par J. Bell [Bell, 2006]. Elle consiste à utiliser des opérateurs de la logique classique et de la logique temporelle pour relier des propositions atomiques (conditions sur les variables comportementales).

Utilisation des opérateurs de la logique classique les opérateurs de la logique classique AND et OR permettent de décrire des conjonctions et des disjonctions entre propositions définies sur les entrées ou sur les sorties du processus.

Description d'une séquence sur les effets Dans le cadre de l'approche multi-modèle de Chittaro, la description des effets d'une entité fonctionnelle se représente de manière statique, c'est-à-dire que la valeur observée, qu'elle soit numérique ou qualitative, est stable durant tout le temps de l'observation. Cependant, ce cadre est insuffisant pour représenter les effets d'un système séquentiel. En effet, l'abstraction d'un comportement séquentiel nécessite de représenter des effets dynamiques en décrivant des séquences de différentes propriétés, appelées *séquences d'épisodes*[3], portant sur les variables de sorties d'une entité fonctionnelle. Par exemple, pour exprimer la dynamique intermittente du système essuie-vitre, il est nécessaire de représenter que l'effet, portant sur la vitesse de rotation *omega_balais*, devant être observé correspond à un phénomène cyclique où chaque cycle est composé de la séquence des épisodes qualitatifs suivants :

1. Episode 1 : $(omega\_balais > 0)$
2. Episode 2 : $(omega\_balais < 0)$
3. Episode 3 : $(omega\_balais = 0)$ et $(teta\_balais = 0)$ pendant 7 secondes puis reprise du cycle

Ce problème a été traité dans le cadre des travaux de Bell [Bell et Snooke, 2004; Bell *et al.*, 2005]. Un langage complet permettant de décrire des séquences et des cycles y est proposé. Nous présentons dans ce manuscrit les principaux mots clés que nous utilisons :

- `SEQ` : est un mot-clé qui permet de représenter l'opérateur **N** [Emerson, 1990] de la logique temporelle signifiant « à l'épisode suivant ». Ainsi, la relation (a `SEQ` b) reliant les deux prédicats logiques a et b, signifie que a est vrai dès que b devient faux.

[3]Nous préférons le terme d'épisode au terme d'état afin d'éviter la confusion avec la notion d'état fonctionnel.

– CYCLE et NEW_CYCLE sont deux mots-clés permettant de marquer le début d'un cycle et le retour au début de ce cycle. Ainsi, la relation (CYCLE a SEQ b SEQ NEW_CYCLE) exprime une séquence stricte entre les prédicats a et b qui se reproduit de manière cyclique.

Les mots clés AFTER et BEFORE peuvent être également utilisés pour exprimer des contraintes temporelles en terme de délais :

– AFTER est utilisé pour spécifier un délais minimum après lequel l'effet doit être observé.
– BEFORE est utilisé pour spécifier un délais maximum avant lequel un effet doit être observé.

Ainsi, la description de la dynamique intermittente du balais essuie-vitre peut être exprimée par la relation :

CYCLE $(omega\_balais > 0)$ SEQ $(omega\_balais < 0)$ SEQ $(omega\_balais = 0)$ SEQ NEW_CYCLE [AFTER 6.9s BEFORE 7.1s]

Une visualisation graphique de cette séquence est donnée par la figure 4.15.

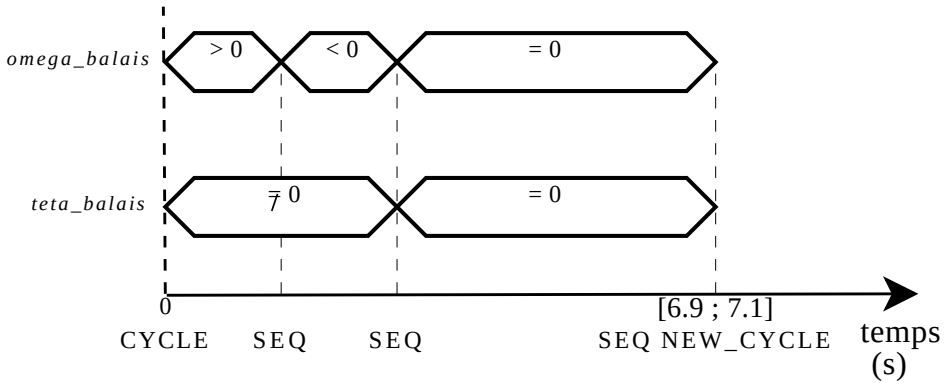


Figure 4.15 — Représentation d'une séquence sur les effets

Description des processus du système essuie-vitre

Pour le système essuie-vitre présenté dans ce chapitre, nous pouvons ainsi décrire les propriétés des processus de données et des processus physiques :

GLOBAL_PROCESSUS P0 : "Essuyer Lunette Arrière"

PRECONDITIONS : $(pos\_cear = 2)$ AND $accessoire$ AND $(tension\_batterie > O)$ OR $(etat\_ess\_av$ AND $etat\_mar)$ AND $accessoire$ AND $(tension\_batterie > O)$

EFFETS : CYCLE $(omega\_balais > 0)$ SEQ $(omega\_balais < 0)$ SEQ $(omega\_balais = 0)$ SEQ NEW_CYCLE [AFTER 6.9s BEFORE 7.1s]

DATA_PROCESS Pdat_1 : "Enregistrement commande"

PRECONDITIONS : $(pos\_cear = 2)$

EFFETS : $(etat\_cear = dmd\_ess\_ar)$

DATA_PROCESS Pdat_11 : "Lecture message"

PRECONDITIONS : $(etat\_cear = dmd\_ess\_ar)$

EFFETS : dmd_ess_ar

DATA_PROCESS Pdat_12 : "Lecture entré capteur"

PRECONDITIONS : $courant\_af > 0$

EFFETS : $arret\_fixe\_ar$

PHYSICAL_PROCESS Pelm_1 : "Actionner"

PRECONDITIONS : $(tension\_batterie > 0)$ AND cmd_evar

EFFETS : $(omega\_balais > 0)$ OR $(omega\_balais < 0)$

PHYSICAL_PROCESS Pelm_2 : "Circuit capteur"

PRECONDITIONS : $(tension\_batterie > 0)$ AND $(teta\_balais = 0)$

EFFETS : $(courant\_af > 0)$

4.5 La connaissance téléologique

4.5.1 Définition

Le modèle téléologique spécifie les fonctions du système en terme de buts assignés par son concepteur. Nous utilisons le terme de fonction uniquement à ce niveau de connaissance et nous le définissons de la manière suivante :

Définition 1 (FONCTION). *Une fonction est une application qui, sous une conjonction de conditions d'entrées atomiques, place le système dans un état donné, souhaité (concepteur) ou utile (usager).*

Cette définition implique la notion de testabilité d'une fonction afin de vérifier si celle-ci est réalisée ou si elle est défaillante. Les conditions opérationnelles nécessaires à son activation (pré-conditions) et le comportement devant être observé (effets) doivent donc être précisés.

Les fonctions d'un système sont directement liées à son processus global, décrit dans le modèle des processus fonctionnels. S'il existe plusieurs chemins pour activer le processus global d'un système, c'est à dire s'il existe des disjonctions dans la description des pré-conditions du processus global, alors il y aura autant de fonctions réalisables que de chemins possibles.

4.5.2 Abstraction sémantique du comportement

La description des pré-conditions et des effets d'une fonction nécessite d'abstraire les différentes propositions définies sur les entrées et les sorties du processus global, par des propositions sémantiques de plus haut niveau. Ces étiquettes permettent d'établir un lien cognitif avec l'utilisateur du système. Pour le système essuie-vitre, ces propositions sont données par le tableau 4.2 pour les pré-conditions et le tableau 4.3 pour les effets.

Connaissance fonctionnelle	Connaissance téléologique
$accessoire$	cle_de_contact_sur_accessoire
$etat\_mar$	marche_arrière_activee
$etat\_ess\_av$	essuyage_avant_active
$pos\_cear = 2$	commutateur_essuyage_arrière_en_position_2

Tableau 4.2 — Construction des propositions sémantiques sur les entrées du processus global

Connaissance fonctionnelle	Connaissance téléologique
$(teta\_balais = 0)$	balais_arrière en_position_repos
$(teta\_balais \neq 0)$	balais_arrière_en_position_intermediaire
$(omega\_balais > 0)$	balayage_aller
$(omega\_balais < 0)$	balayage_retour
$(omega\_balais < 0)$	balais_arretes

Tableau 4.3 — Construction des propositions sémantiques sur les sorties du processus global

Les fonctions du système essuie-vitre arrière

Comme le système essuie-vitre arrière a deux contextes d'activation, il réalise deux fonctions qui se décrivent de la manière suivante :

FONCTION 1 : "Essuyer la lunette arrière lors d'une mise sur position 2 du commutateur"

PRECONDITIONS:

cle_de_contact_sur_accessoire

AND commutateur_essuyage_arriere_en_position_2

EFFETS:

CYCLE balayage_aller

SEQ balayage_retour

SEQ (balais_arretes AND balais_arriere_en_position_repos)

NEW_CYCLE [AFTER 6.9s BEFORE 7.1s]

FONCTION 2 : "Essuyer la lunette arrière lorsque l'essuyage avant et la marche arrière sont activés"

PRECONDITIONS:

cle_de_contact_sur_accessoire

AND marche_arriere_activee

AND essuyage_avant_active

EFFETS:

CYCLE balayage_aller

SEQ balayage_retour

SEQ (balais_arretes AND balais_arriere_en_position_repos)

SEQ NEW_CYCLE [AFTER 6.9s BEFORE 7.1s]

4.6 Vers une stratégie multi-modèle de prédiction des tests avec des modes de défaut

Dans cette section, nous discutons de l'approche de représentation multi-modèle que nous avons proposée dans ce chapitre. Ensuite, nous introduisons la manière dont nous utilisons ces modèles pour la prédiction de tests à des niveaux de connaisance différents.

4.6.1 Synthèse de l'approche de modélisation proposée

Notion de composant logiciel Dans ce chapitre, nous avons défini le composant logiciel comme une entité pouvant avoir une représentation structurelle et interagissant avec des composants physiques. Cette représentation est une vue très abstraite de ce qu'est le logiciel de commande, d'une part, mais surtout de ce qu'est l'électronique sous-jacente. Cependant, elle est suffisante compte-tenu des points d'observation disponibles et de l'objectif de diagnostic. Par exemple, le signal de commande de l'interrupteur `K1` (figure 4.7) permettant d'activer l'actionneur a été modélisé par un signal de type booléen car il est mesurable (via l'« outil de diagnostic ») alors que le signal électrique de commande sous-jacent ne l'est pas et `K1` n'est pas un composant remplaçable : l'action réparatrice en cas défaut de ce composant est de remplacer le calculateur `ECU_hab`.

Comportement vs Processus fonctionnel La description d'une entité fonctionnelle par ses pré-conditions et ses effets amène à décrire des propriétés qui sont une abstraction du comportement par instantiation des variables d'état du système étudié sur des valeurs (ou plages de valeurs) particulières : c'est ce qui est appelé état partiel dans l'approche FR[4] (section 3.3.1).

De plus, la notion de causalité est déjà présente dans la définition même de ces composants du fait de leur représentation entrée / sortie. Par exemple, nous avons donné une représentation comportementale du composant logiciel `Detect_af` (figure 4.6) sous la forme d'un automate discret (figure 4.12) qui a été directement déduit de ses propriétés fonctionnelles (section 4.4.2). Nous avons donc supposé que les propriétés du processus de donnée `Pdat_12` représentait le comportement du composant `Detect_af`. Ce type d'assimilation, connaissance fonctionnelle vers connaissance comportementale, est au coeur de la modélisation (comportementale) des systèmes hybrides.

Pour les composants physiques, la distinction est plus évidente à établir car ce problème a été complètement traité dans le cadre de l'approche multi-modèle de Chittaro [Chittaro *et al.*, 1993; Thétiot *et al.*, 1998]. Le passage de la notion de comportement à la notion de fonction se traduit par une abstraction causale des lois de la physique et une abstraction qualitative des variables continues pour définir les propriétés des processus fonctionnels. C'est pourquoi la distinction entre modèle comportemental et modèle fonctionnel est peu claire pour les composants logiciels pour lesquels le comportement modélisé par des automates de Moore se traduit directement en logique propositionnelle.

[4] D'un point de vue formel, on passe d'une représentations numérique à une représentation logique

Un point qui peut permettre de distinguer comportement d'un composant et modèle fonctionnel est la représentation du temps. Comme nous avons pu le voir dans ce chapitre, la construction du modèle fonctionnel d'un système séquentiel passe par l'abstraction qualitative de l'évolution d'un état défini sur un temps continu (ou continu discrétisé) en la définition de séquences d'épisodes qui peuvent avoir une certaine durée.

Téléologie et symptômes fonctionnels Le modèle téléologique, tel que nous l'avons décrit dans ce chapitre représente des fonctions comme une réalisation attendue par l'utilisateur (ou prévu par le concepteur) d'un système. Le cadre de représentation sémantique proposé permet d'abstraire l'évolution de variables du système par une évolution perceptible par l'homme (mouvement de balais).

Une défaillance de fonction, peut se traduire par une perte totale des effets devant être observés mais aussi par l'observation d'effets qui ne sont pas ceux qui étaient attendus. Par exemple, un défaut de rupture (circuit ouvert) du composant `Fil_2` (figure 4.3) provoquera la perte de l'information de fermeture du capteur de position repos du module `EVAR`. Lorsque l'utilisateur demandera une activation du processus essuyage arrière, il observera un balais en mouvement pendant une durée correspondant au seuil de protection *T_protec* modélisé sur l'automate de commande (figure 4.11) puis un arrêt de celui-ci dans une position intermédiaire. L'activation du processus par l'utilisateur donne donc bien un effet mais pas celui attendu.

L'analyse fonctionnelle de l'influence des différents défauts sur un système apporte donc une information sur les différents symptômes issus des effets des fonctions du système. Par la suite, nous les appelons « symptômes fonctionnels » et nous les considérons comme des tests à part entière.

4.6.2 Plan de raisonnement multi-modèle pour la génération de tests

Par la suite, nous considérons que la connaissance structurelle et comportementale des systèmes mécatroniques que nous étudions est disponible. Comme précisé dans ce chapitre, la structure de la partie logicielle du système et le comportement des composants logiciels est déduite des spécifications fonctionnelles, seule connaissance disponible.

La description de la connaissance fonctionnelle d'un système se limite à la description de ses processus globaux que nous attachons directement aux variables permettant de décrire les pré-conditions et les effets.

Notre objectif est d'utiliser ce cadre, que nous schématisons sur la figure 4.16 pour résoudre le problème de séquencement des tests et d'y inclure les symptômes fonctionnels sous forme de tests. Nous constatons que certaines cellules de la grille restent vides par rapport à la figure 4.1. Le passage des cellules structurelles et comportementales vers les cellules téléologiques correspondant aux fonctions du système peut en effet se faire en suivant différents chemins. Comme nous n'utilisons pas les processus fonctionnels dans le raisonnement de diagnostic qui suit, nous ne les avons pas représentés. Ainsi, la prédiction du résultat des tests sous l'occurrence de défauts se fera de la manière suivante :

1. Le système est simulé dynamiquement à partir de son modèle structurel et comporte-

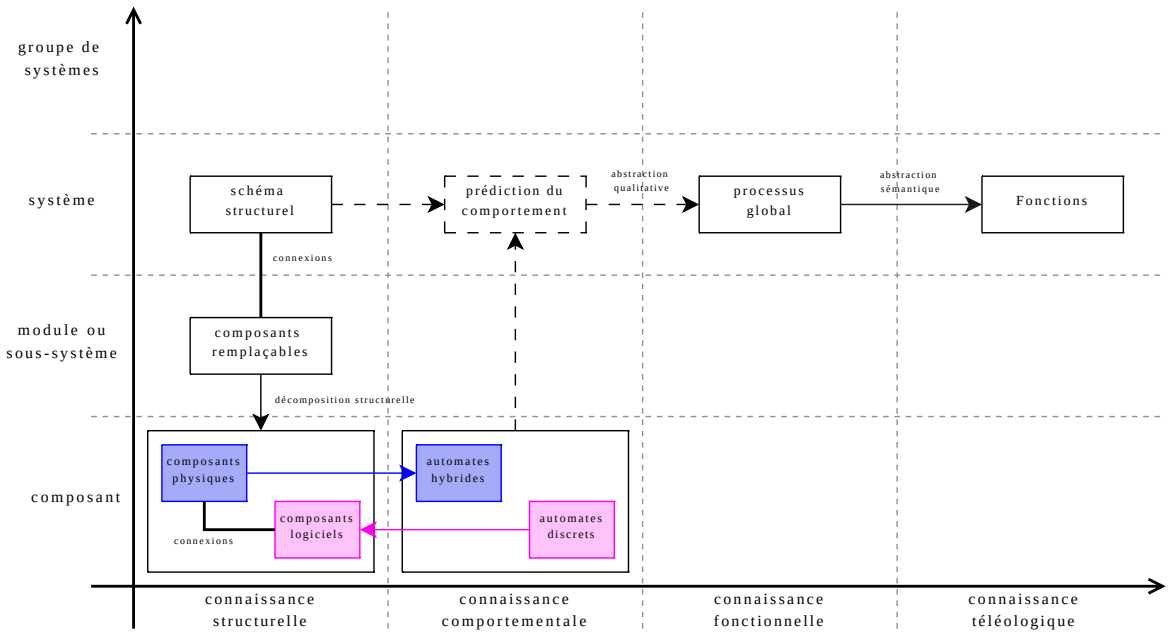


Figure 4.16 — Prédiction multi-modèle de tests

mental. Le modèle comportemental inclut notamment des modes de défauts anticipés.

2. Les variables observables sont définies au niveau des ports logiciels et matériels et leur valeur est donc prédite numériquement par la simulation. Certaines, comme celles intervenant sur les effets des fonctions font l'objet d'une abstraction qualitative puis sémantique pour la prédiction de symptômes fonctionnels sous l'occurrence des défauts anticipés.
3. Le résultat de l'étape de prédiction des tests est donné par une matrice de signature de défauts incluant des tests de types hétérogènes.

5 Prédiction

L'objectif de ce chapitre est de présenter la méthode de prédiction des tests que nous avons développée en plaçant notamment le « Test Sequencing Problem » (section 2.3.2) dans la stratégie multi-modèle présentée dans le chapitre précédent (section 4.6). La prédiction des tests est basée sur la simulation numérique et hybride des systèmes mécatroniques à diagnostiquer. Dans un premier temps, nous présentons les modes de défaut anticipés, et plus particulièrement ceux portant sur les composants logiciels. Ensuite, nous définissons les différents tests utilisés et nous montrons comment nous intégrons les différents moyens d'observation disponibles en garage dans le cadre de la modélisation proposée. Par rapport à la méthode précédente développée à Actia (AGENDA)[Faure, 2001; Olive, 2003], ces tests donnent lieu à de nouvelles modalités appelées « courbes de référence » et « symptômes fonctionnels ». Enfin, nous présentons comment la matrice de signatures des défauts issue de cette étape de prédiction est construite.

5.1 Anticipation des défauts

Comme dans la méthode AGENDA, notre approche de diagnostic est basée sur la définition d'un dictionnaire de défauts anticipés. Chaque défaut anticipé est supposé permanent, unique et entraîne généralement la défaillance d'une fonction du système avec apparition de symptômes. Nous distinguons deux classes de défauts : les défauts logiciels et les défauts matériels.

5.1.1 Modes de défauts matériels

La définition des modes de défaut pour un composant matériel consiste à compléter l'automate hybride décrivant son comportement nominal par des modes de défaut connus. Cette représentation rejoint celle proposée dans [Olive, 2003] mais celle-ci se limitait à l'anticipation de défauts sur des circuits résistifs. Nous montrons qu'il est aussi possible d'anticiper des défauts sur des composants d'un domaine de la physique différent.

Défauts électriques

Les défauts électriques peuvent porter sur les paramètres des composants, sur des commutations ou sur la structure du système. Nous ne définissons que brièvement ces défauts

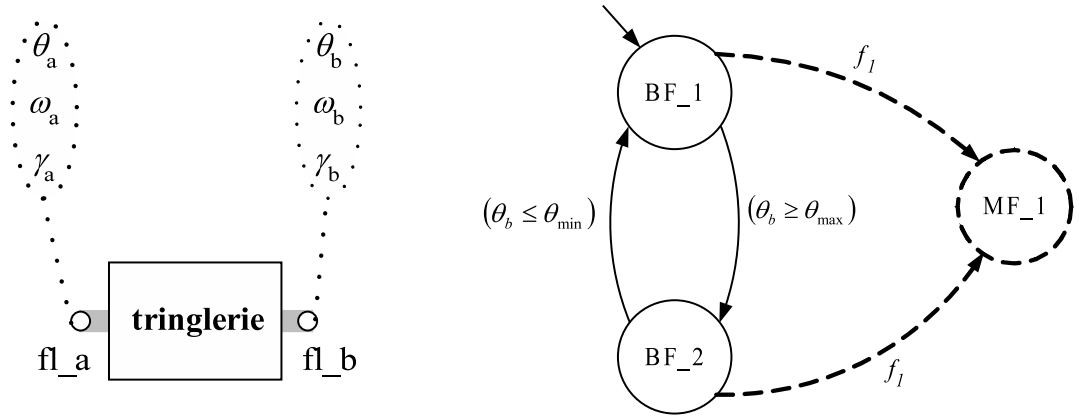


Figure 5.1 — Modèle comportemental du composant `tringlerie` avec modes de défauts

car ils ont déjà été largement traités dans les trois thèses qui ont précédé nos travaux [Duffaut, 1994; Faure, 2001; Olive, 2003].

Défauts paramétriques Les défauts paramétriques correspondent au cas de rupture d'un composant (circuit ouvert), modélisé par un paramètre de conductance nulle et au cas de fusion d'un composant (court-circuit), modélisé par un paramètre de résistance nulle.

Défauts de commutation Un défaut de commutation porte sur un composant de type « interrupteur ». Il peut correspondre à un blocage de celui-ci à l'ouverture (« Collé Ouvert ») ou à la fermeture (« Collé Fermé ») quelque soit la commande. Notons que dans [Olive, 2003] ces défauts étaient appelés « défauts de transition » mais nous préférons leur attribuer à chacun un mode de défaut particulier.

Défauts structurels Les défauts structurels sont ceux qui modifient les connexions entre composants. Ils peuvent correspondre à un circuit ouvert (« rupture » d'une connexion) ou à un court-circuit (ajout d'une contrainte de type égalité de potentiel). Concernant les courts-circuits structurels, nous supposons que ceux-ci interviennent au niveau des ports des composants de type « fil électrique ».

Exemple de défaut mécanique

Pour les composants mécaniques, il est aussi possible d'anticiper des défauts de paramètres. Par exemple, le modèle de comportement nominal du composant `tringlerie` donné dans le chapitre précédent (figure 4.9) peut être augmenté du mode de défaut MF_1 correspondant à un blocage mécanique (figure 5.1). Ce blocage peut être modélisé par un paramètre de frottement R_f qui devient infini dans ce mode.

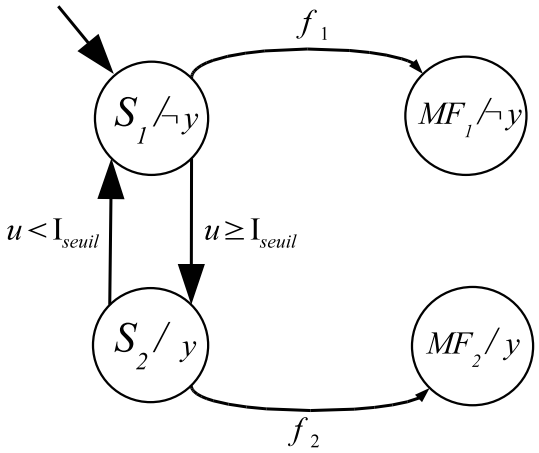


Figure 5.2 — Anticipation de défauts sur le composant `Detect_AF`

5.1.2 Modes de défaut des composants logiciels

Le problème d'anticipation de modes de défaut sur des composants logiciels embarqués est traité par certains travaux mais essentiellement pour des tâches de vérification en conception. L'objectif est de tester si les logiciels qui ont été implémentés sont corrects vis-à-vis des spécifications. A notre connaissance, deux types de défauts sont généralement modélisés : les défauts de transition [Esser et Struss, 2007] et les défauts de signal d'entrée ou de sortie [Godesken, 1999].

Défauts de signaux entrée / sortie Dans [Godesken, 1999], ces défauts sont anticipés pour traiter des erreurs de connexion entre la partie logicielle et la partie matérielle d'un système embarqué. Dans le cadre du diagnostic hors-ligne, cela peut se traduire par le blocage d'un signal de type booléen à « vrai » ou « faux ».

Par exemple, pour le système essuie-vitre, l'automate de comportement du composant `Detect_AF` (figure 4.6) permettant de détecter la présence du balai en position repos peut être augmenté de deux états de défaut (figure 5.2) : l'état de défaut MF_1 correspond au cas où la sortie u de ce composant reste collée à faux et l'état MF_2 correspond au cas où u reste collé à « vrai ».

Défauts de transition Dans [Esser et Struss, 2007], les défauts de transition anticipés correspondent à des transitions oubliées ou mal aiguillées lors de la phase de codage informatique du logiciel. Dans le cadre du diagnostic, il n'existe pas de réparation associée à ce type de défaut car nous supposons que les erreurs de programmation sont traitées en amont de la phase de diagnostic. En effet, lorsque le garagiste réceptionne un véhicule défaillant il vérifie généralement si les logiciels embarqués sur les calculateurs susceptibles d'être concernés par cette défaillance sont à jour. Si ce n'est pas le cas, il télécharge les dernières mise-à-jour avant de démarrer une session de diagnostic.

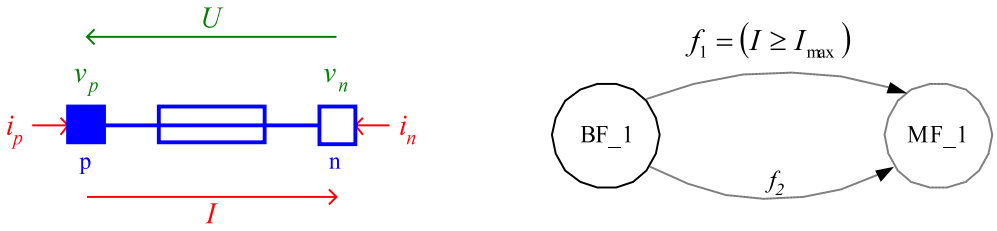


Figure 5.3 — Modes de défauts du fusible

5.1.3 Cas des défauts cascadés

Pour certains composants, il peut exister des modes hybrides non réversibles (impliquent donc une rupture du composant) qui sont activés lors du dépassement d'un seuil sur une variable physique. Le fusible en est l'exemple le plus courant.

Celui-ci peut entrer en mode de défaut de type « circuit ouvert » selon deux événements différents représentés par f_1 et f_2 sur la figure 5.3.

- La condition f_2 représente le cas où l'occurrence du mode de défaut MF_1 est dû à une rupture spontanée de ce composant pour des raisons d'usure. Le défaut n'est donc pas induit par l'occurrence d'un autre défaut sur le système.
- La condition f_1 représente le cas où l'occurrence du mode de défaut MF_1 du fusible est induit par l'occurrence d'un défaut (court-circuit) sur une autre composant du système et ayant provoqué un niveau de courant électrique anormal dans le circuit protégé par le fusible. Nous parlons dans ce cas de défaut cascadé. Nous supposons que la rupture cascadée d'un fusible est instantanée et que le mode MF_1 est déjà présent lorsque le véhicule est réceptionné par le garagiste. Ainsi, tous les défauts anticipés pouvant impliquer cette rupture doivent alors être associés au défaut de rupture du fusible dans la matrice de signature des défauts. Notons que ce défaut correspond à un cas de défaut multiple anticipé. Sa probabilité d'occurrence est celle correspondant au défaut responsable de la rupture du fusible.

5.1.4 Notion de réparation

Soit Ψ le système à diagnostiquer et défini comme un ensemble de n_ψ modules remplaçables $\Psi_q, q \in \{1, ..., n_\psi\}$. Pour chaque module Ψ_q, soit n_F^q le nombre total des modes de défauts anticipés sur l'ensemble des composants élémentaires dont est constitué Ψ_q.

Une réparation est définie dans ce manuscrit comme l'ensemble des modules (composants remplaçables) que le garagiste doit remplacer ou vérifier à la fin de la session de diagnostic afin de rétablir le bon fonctionnement du système. Nous définissons alors l'ensemble $\mathcal{R}$ des $n_R + 1$ réparations possibles $r_0, 0 \in \{0, ..., n_R\}$. Nous distinguons trois types de réparation :

- *Ne rien changer* : lorsque qu'il n'y pas de présence de défaut et que le système fonctionne normalement. Cela correspond, par convention à la réparation r_0
- *Changer le composant remplaçable* : lorsqu'un composant élémentaire est défaillant, le garagiste doit remplacer le module (ou composant) remplaçable qui contient le com-

posant élémentaire.
- *Changement du fusible et du composant remplaçable* : dans le cas de défauts cascadés impliquant un fusible, le garagiste doit remplacer le fusible et le composant dont le défaut (court-circuit) a impliqué sa fusion.

La notion de réparation est utilisée pendant le séquencement des tests en considérant que où si l'ensemble d'ambiguïté associé un noeud de l'arbre de diagnostic correspond à une réparation alors ce noeud n'est pas développé.

5.2 Présentation des moyens de test considérés

Dans cette section, nous présentons l'ensemble des outils de test que nous considérons dans le processus de diagnostic. Ces outils se décomposent en trois grandes classes : les instruments de mesures physiques, les services de diagnostic des calculateurs et les symptômes fonctionnels.

5.2.1 Instruments de mesures physiques

Les instruments de mesures physiques permettent de visualiser directement l'évolution de variables physiques au niveau des ports des composants matériels. Les mesures les plus courantes sont les mesures portant sur des variables électriques. Dans ce manuscrit, nous considérons que le garagiste dispose d'une carte de mesure lui permettant d'utiliser son « outil de diagnostic » comme un oscilloscope ou comme un multimètre numérique. Dans le cadre de la prédiction dynamique et multi-modèle que nous proposons nous privilégions les mesures de tension électrique et certaines mesures de courant (réalisées au moyen d'une pince ampère-métrique). Les mesures de résistance équivalente ne sont pas traitées dans ce manuscrit. Celles-ci nécessitent en fait des déconnexions de faisceaux électriques qui ne sont pas pratiques à gérer avec le simulateur que nous utilisons (chapitre 7). Elles font l'objet d'une prédiction spécifique traitée dans AGENDA [Faure, 2001].

Les mesures électriques sont généralement les plus coûteuses à réaliser car elles nécessitent, d'une part, un certain nombre d'opérations mécaniques pour accéder aux connecteurs au niveau desquels elles sont effectuées, et d'autre part, l'installation de dispositifs de dérivation pour accéder aux points de potentiels électriques.

5.2.2 Services de diagnostic des calculateurs

Comme nous l'avons défini dans le premier chapitre (section 1.2.3), les calculateurs électroniques implémentent un certain nombre de services de diagnostic qui sont intéressants à exploiter lors de la phase de localisation des défauts car ceux-ci apportent des tests peu coûteux à réaliser et ne nécessitent généralement pas d'opérations de démontage.

Utilisation du « test actionneur » Le « test actionneur » permet de piloter directement la sortie d'un calculateur avec l'outil de diagnostic. Comme il s'agit d'une configuration d'activation différente d'un système nous l'interprétons comme la réalisation d'une fonction

spécifique. Cette fonction n'est pas spécifiée à la conception du véhicule mais lors de la conception de l'outil de diagnostic. Lorsque ce service de diagnostic est disponible, il peu se révéler très intéressant à exploiter dans la mesure où il permet de tester individuellement et de manière dynamique un sous-système (le circuit électrique alimentant l'actionneur).

Dans l'exemple du système essuie-vitre, le signal de sortie *cmd_evar* du composant logiciel `Soft_etat_CEAR` est disponible à une activation directe lors d'une session de diagnostic. L'effet devant être observé lors de la réalisation de ce test est 2 cycles de balayage (2 mouvements aller-retour). Cela correspond donc à la réalisation d'une fonction supplémentaire du système essuie-vitre décrite de la manière suivante :

FONCTION 3 : "Essuyer_la_lunette_arrière_en_test_actionneur"

PRECONDITIONS:

cle_de_contact_sur_accessoire

AND demande_test_actionneur

EFFETS:

balayage_aller

SEQ balayage_retour

SEQ balayage_aller

SEQ balayage_retour

SEQ balais_arretes AND balais_arriere_en_position_repos

Cas des « mesures paramètres » Les « mesures paramètres » sont considérées comme des mesures réalisées au niveau des signaux entrée / sortie des composants logiciels. Ces ports constituent donc des points d'observation au même titre que les ports électriques.

Codes défauts Lorsqu'un code défaut est présent dans la mémoire d'un calculateur (section 1.1.3), il est possible d'utiliser cette information comme un test. En effet, si les conditions d'activation d'un code défaut sont connues, il est possible d'intégrer cette information dans les modèles des composants logiciels. Tous les codes défaut peuvent alors être exprimés dans la matrice de signature de défauts en fonction des défauts anticipés lors de l'étape de prédiction des tests.

5.2.3 Les symptômes fonctionnels

Les symptômes fonctionnels portent sur les effets des fonctions d'un système. Ils correspondent à des observations perceptives qui ne nécessitent pas l'utilisation d'appareils de mesure ; ils représentent donc des tests très peu coûteux à réaliser. Actuellement, ces symptômes sont peu exploités, ils ne servent en général qu'à identifier une fonction défaillante. Le fait de les intégrer comme des tests permet de relier, lors de l'étape de prédiction, un dictionnaire de défauts anticipés avec un dictionnaire de symptômes fonctionnels possibles.

5.3 Problème de prédiction des tests

Dans cette section nous définissons la structure d'un test pour un diagnostic multi-modèle. Nous montrons notamment comment ont été intégrés les différents moyens de test dont dispose le garagiste.

5.3.1 Définition de la notion de test

Soit l'ensemble fini $\mathcal{F}$ de n_F+1 défauts, $\mathcal{F} = \{f_i, i = 0, ..., N_F\}$, dans lequel f_0 se rapporte par convention au comportement nominal et les $f_i, (1 \leq i \leq N_F)$ constituent les défauts possibles du système.

Soit un ensemble fini $CONF$ de N_{conf} configurations de test du système tel que $CONF = \{CONF_k, k = 1...N_{conf}\}$. Ces configurations définissent les différents états dans lesquels peuvent être placés les entrées du système (interrupteurs de commandes) et les connecteurs électriques (connectés/déconnectés) pour réaliser des tests. Pour une prédiction dynamique, nous ne définissons que des configurations de commande ; les configurations structurelles correspondent au cas très particulier des tests de résistance équivalente que nous ne considérons pas dans ce manuscrit.

Soit un ensemble fini OBS de variables observables du système. Ces variables observables portent sur les variables physiques ou logicielles circulant à travers les ports physiques ou logiciels disponibles à la mesure.

Nous définissons aussi l'ensemble fini $OBS^k \subseteq OBS$ des N^k_{OBS} variables qui sont observables lorsque le système est dans la configuration $CONF_k$ tel que $OBS^k = \{OBS^k_l, l = 1...N^k_{OBS}\}$. Ainsi, $OBS = \bigcup_{k=1}^{N_{conf}} OBS^k$.

Structure d'un test Chaque test $s_j, j = 1, ..., N_s$ de l'ensemble des tests $\mathcal{S}$ du « Test Sequencing Problem » est défini par le 6-uplet

$$s_j =< (CONF_k)_j, (OBS^k_l)_j, T_j, LEV_j, INST_j, MOD^j >$$

avec :

- $(CONF_k)_j \in CONF$: la configuration dans lequel le système doit être placé pour réaliser le test s_j.
- $(OBS^k_l)_j \in OBS^k$: la variable qui doit être observée et qui est disponible dans la configuration $CONF_k$.
- T_j : la durée pendant laquelle l'observation doit être effectuée. Dans le cadre pratique, nous supposons que ce paramètre est le même pour tous les tests portant sur un système donné.
- LEV_j : précise le type de connaissance dans lequel le test doit être exprimé, c'est à dire comportemental, fonctionnel ou téléologique.
- $INST_j$: l'instrumentation à utiliser et les différents dispositifs à installer pour réaliser le test (outil de diagnostic, dérivateur + carte de mesure...).
- MOD^j : l'ensemble des résultats possibles du test s_j appelés *modalités*.

Expression multi-modèle d'un test La définition de la notion de test précédente donne la possibilité d'exprimer des tests suivant différent types de connaissance. Suivant le type LEV_j choisi pour exprimer un test s_j, la structure de la modalité MOD^j est différente :

- au niveau comportemental (LEV_j='behavior'), la structure d'une modalité est une série temporelle dont la longueur est fixée par la durée d'observation T_j et la période d'échantillonage (pas de simulation). Cette série temporelle est appelée *courbe de référence*.
- au niveau fonctionnel (LEV_j='functional'), la structure d'une modalité est une séquence d'épisodes qualitatifs (cf. section 4.4.2).
- au niveau téléologique (LEV_j='teleological'), la structure d'une modalité est une séquence d'épisodes sémantiques. Cette séquence constitue un *syptôme fonctionnel*.

5.3.2 Problème de génération de la matrice de signature des défauts

Le processus de prédiction dynamique des tests prend l'ensemble des modes de défaut en entrée et prédit pour chacun d'eux la valeur des tests. Quatre étapes composent ce processus :

Simulation des tests : la première étape consiste à simuler le système de manière itérative pour chaque configuration et chaque mode de défaut. Ainsi, une simulation est réalisée pour chaque couple $(CONF_k, f_i)$ tel que $1 \leq k \leq N_{conf}$ et $1 \leq i \leq N_F$, ce qui fait $N_F \times N_{conf}$ simulations à réaliser. Les résultats des simulations sont stockés dans une matrice **SIM** de dimension $N_s \times N_F$ dont chaque élément $SIM_{i,j}$ représente une série temporelle, résultat de la simulation portant sur la variable d'observation du test s_j sous l'occurrence du défaut f_i.

Anticipation des défauts cascadés A la fin de chaque simulation, le mode de défaut final de chaque fusible présent dans le système doit être vérifié. Si un défaut anticipé sur un composant du système a impliqué la rupture d'un fusible, alors ce défaut doit être associé au défaut de circuit ouvert du fusible et les autres configurations doivent être à nouveau simulées avec ce défaut. En effet, si le fusible a fondu, il est en mode de défaut pour toutes les configurations du système.

Construction des « courbes de référence » : la deuxième étape consiste à construire les *modalités*, sous forme de *courbes de références*, pour les test s_j dont le type est LEV_j='behavior'. Ces modalités sont construites par une méthode de classification qui utilise comme critère la mesure de similarité DTW (Dynamic Time Warping) [Sakoe et Chiba, 1978; Salvador et Chan, 2007; Keogh et Pazzani, 2001] pour former des classes d'équivalence de signaux pour un test donné (cf. section 5.4.1).

Génération des symptômes fonctionnels : les symptômes fonctionnels sont des tests de type LEV_j='teleological' et portent sur des variables qui sont simulées à partir des connaissances structurelles et comportementales (cf. section 5.4.2).

5.4 Construction des modalités

L'objectif de cette section est de présenter comment les modalités des tests sont construites à partir des résultats des simulations stockés dans la matrice **SIM**. La notion de modalité est à considérer comme une étiquette symbolique que nous attribuons à un signal comme une appartenance à une classe de résultat possible pour un test. Lors de la phase de séquencement des test, le moteur de diagnostic ne manipule donc plus des signaux mais des étiquettes symboliques. Dans un premier temps, nous proposons l'utilisation de la mesure de similarité DTW qui permet de comparer les signaux d'un même test deux à deux pour former des modalités de type « courbes de références ». Ensuite, nous présentons une méthode pour générer les symptômes fonctionnels.

5.4.1 Classification de signaux avec la distance DTW

Pour construire un dictionnaire de « courbes de référence » par test, c'est à dire construire l'ensemble MOD^j pour chaque test s_j, il est nécessaire de comparer les signaux deux à deux pour chaque colonne de la matrice de résultats **SIM** et former ainsi des classes d'équivalences. Cette comparaison doit tenir compte de la précision de mesure disponible dans le contexte réel du garage qui est actuellement assez grossière. Pour la mesure d'un signal, cette précision porte sur deux aspects : l'amplitude et le temps.

- *Précision sur l'amplitude* : la précision sur l'amplitude peut être limitée par le bruit contenu dans le signal mesuré, la qualité d'affichage de la courbe sur l'écran de l'outil de diagnostic et l'appréciation visuelle du garagiste. Par exemple, pour une mesure de tension électrique réalisée avec un affichage en amplitude compris entre -20 volts et 20 volts, une précision raisonnable à considérer pour ce type de mesure serait de ± 1 volts.
- *Précision sur le temps* : la précision sur la mesure du temps en garage peut dépendre du décalage temporel entre le début d'enregistrement d'une mesure et la sélection d'une configuration du système pour réaliser le test, de la capacité de l'outil de mesure à échantillonner un signal et de la qualité d'affichage du signal. Par exemple, il est possible de visualiser graphiquemment des « mesures paramètre » mais la période d'échantillonnage de ce type de mesure est assez grande car elle est fortement limitée par le débit de transmission des données disponible entre le calculateur interrogé et l'outil de diagnostic.

Une simple comparaison de deux séries temporelles point par point par une distance euclidienne n'était donc pas satisfaisante car cette méthode ne permet pas de tenir compte d'une tolérance sur le décalage temporel des données contenues dans ces séries. Par exemple, si nous souhaitons tolérer la déformation temporelle entre les signaux X et Y de la figure 5.5, la distance euclidienne ne permet pas de le faire.

DTW (Dynamic Time Warping) est un algorithme qui permet de trouver le recalage temporel optimal entre deux séries temporelles [Sakoe et Chiba, 1978; Salvador et Chan, 2007; Keogh et Pazzani, 2001]. Il est souvent utilisé pour déterminer des similarités entre deux séries temporelles et fait l'objet de nombreuses applications dans le domaine de la reconnaissance de la parole [Poli *et al.*, 2007]. Nous présentons dans un premier temps la forme

classique de l'algorithme puis nous ajoutons des tolérances sur l'amplitude et le décalage temporel afin de tenir compte de l'appareil de mesure.

5.4.1.1 Algorithme DTW classique

Formulation du problème Soit deux séries temporelles X et Y de longueurs respectives N et M qui sont représentées sous forme de deux vecteurs de données :

$$X = [x_1, x_2, ..., x_i, ..., x_N] \qquad \text{et} \qquad Y = [y_1, y_2, ..., y_j, ..., y_M]$$

Pour recaler temporellement les deux sequences de données en utilisant DTW, nous construisons une matrice $\mathbf{D}$ de dimension $N \times M$ pour laquelle chaque élément d_{ij} correspond à la distance locale $d(x_i, y_j)$ entre deux données x_i et y_j. Dans notre cas, nous utilisons la distance euclidienne telle que $d(x_i, y_j) = |x_i - y_j|$. Chaque coordonnée (i, j) de $\mathbf{D}$ correspond alors à un recalage possible entre les données x_i et y_j.

Un chemin de recalage W entre les deux séquences X et Y est un séquence de cordonnées d'éléments contigus de la matrice $\mathbf{D}$. Ainsi, nous avons :

$$W = [w_1, w_2, ..., w_k, ..., w_L] \qquad \text{tel que} \qquad max(N, M) \leq L < N + M$$

où L est la longeur du chemin de recalage W et le $k^{\text{ème}}$ élément de ce chemin est $w_k = (i, j)$ avec i et j des indices des séquences respectives X et Y. La construction de W doit respecter les trois contraintes suivantes :

- ***Conditions limites*** : le chemin W doit commencer au début et à la fin des deux séquences X et Y. Cela se traduit par les deux contraintes $w_1 = (1, 1)$ et $w_L = (N, M)$.
- ***Continuité*** : pour $w_k = (i, j)$ alors $w_{k+1} = (i', j')$ tel que $i - i' \leq 1$ et $j - j' \leq 1$.
- ***Monotonie*** : pour $w_k = (i, j)$ alors $w_{k+1} = (i', j')$ tel que $i - i' \geq 0$ et $j - j' \geq 0$.

Les propriétés de continuité et de monotonie impliquent que si $w_k = (i, j)$, alors il existe trois valeurs possibles pour w_{k+1} qui sont $(i, j + 1)$, $(i + 1, j)$ ou $(i + 1, j + 1)$ (cf. figure 5.4).

Le chemin de recalage optimal est le chemin qui minimise la somme des distances cumulées pour aller du point de recalage initial $w_1 = (1, 1)$ de la matrice $\mathbf{D}$ au point final $w_L = (N, M)$. Ce minimum définit alors la distance (coût) de recalage DTW :

$$DTW = min\left(\sum_{k=1}^{L} d(w_k)\right)$$

Par la suite, nous nous intéressons plus particulièrement au calcul de la distance DTW car c'est elle qui va nous servir de critère de comparaison entre deux signaux. Cette distance devra être presque nulle pour décider de l'équivalence entre deux séries temporelles.

Résolution du problème classique Le calcul de la distance DTW revient à utiliser une approche de programmation dynamique. Elle consiste à définir une matrice de coût Δ de dimension $N \times M$ appelée matrice des distances cumulées. Chaque élément $\delta_{i,j}$ de cette matrice est la distance minimale de recalage entre les séries temporelles $X' = [x_1, x_2, ..., x_i]$

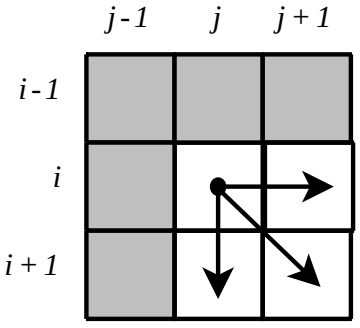


Figure 5.4 — Contraintes de continuité et de monotonie de DTW

et $Y' = [y_1, y_2, ..., y_j]$. La valeur $\delta_{N,M}$ correspond alors à la valeur de la distance minimale DTW recherchée. Nous obtenons ainsi l'algorithme 5.1 permettant de calculer cette distance.

Algorithme 5.1 — Calcul classique de la distance DTW minimal

Entrées : X,Y
Sorties : DTW,$\mathbf{D}$,$\mathbf{\Delta}$

% X : séquence de données de taille N
% Y : séquence de données de taille M
% DTW : distance minimale de recalage
% $\mathbf{D}$ est la matrice des distances locales de taille $N \times M$
% $\mathbf{\Delta}$ est la matrice des distances cumulées $N \times M$
% Calcul de la matrice des distances locales
$\mathbf{D} = \Big[d(i,j) = |x_i - y_j|\Big]_{\substack{1 \leq i \leq N \\ 1 \leq j \leq M}}$
% Initialisation de la matrice des distances cumulées
$\mathbf{\Delta} = \Big[\delta(i,j) = 0\Big]_{\substack{1 \leq i \leq N \\ 1 \leq j \leq M}}$
% Calcul de la première ligne de Δ
pour $i = 2 \ldots N$ **faire** $\delta(i,1) = d(i,1) + \delta(i-1,1)$
% Calcul de la première colonne de Δ
pour $j = 2 \ldots M$ **faire** $\delta(j,1) = d(j,1) + \delta(j-1,1)$
% Calcul des autres valeurs de Δ **pour** $i = 2 \ldots N$ **faire**
  pour $j = 2 \ldots M$ **faire**
    $$\delta(i,j) = d(i,j) + \min(\delta(i-1,j), \delta(i,j-1), \delta(i-1,j-1))$$
  fin
fin
% Valeur de la distance de recalage minimale
retourner $DTW = \delta(N,M)$

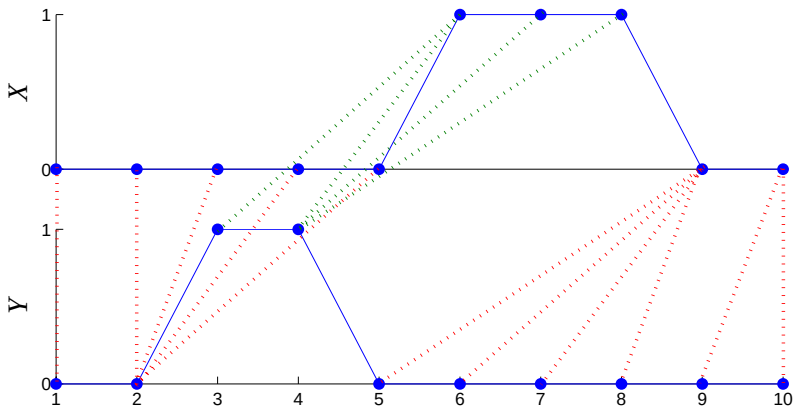


Figure 5.5 — Exemples de signaux pour illustrer DTW

	1		…		j		…		10
1	0	0	1	1	0	0	0	0	0	0
	0	0	1	1	0	0	0	0	0	0
⋮	0	0	1	1	0	0	0	0	0	0
	0	0	1	1	0	0	0	0	0	0
	0	0	1	1	0	0	0	0	0	0
i	1	1	0	0	1	1	1	1	1	1
	1	1	0	0	1	1	1	1	1	1
⋮	1	1	0	0	1	1	1	1	1	1
	0	0	1	1	0	0	0	0	0	0
10	0	0	1	1	0	0	0	0	0	0

Tableau 5.1 — Matrice des distances locales

Exemple simple Afin d'illustrer l'algorithme DTW que nous venons de présenter, nous l'appliquons sur deux séquences de données simples (figure 5.5) de même longueur. Soit X et Y de longueurs $N = 10$ tels que :

$$X = [0, 0, 0, 0, 0, 1, 1, 1, 0, 0] \qquad \text{et} \qquad Y = [0, 0, 1, 1, 0, 0, 0, 0, 0, 0]$$

La tableau 5.1 donne la matrice $\mathbf{D}$ des distances locales entre X et Y et le tableau 5.2 donne la matrice des distances cumulées $\mathbf{\Delta}$ pour cet exemple. La distance de recalage minimal obtenu est $DTX = 0$. Ce résultat est logique car X et Y ont la même amplitude et nous n'avons, pour le moment, donné aucune limite de déformation temporelle des signaux.

5.4.1.2 Prise en compte de l'incertitude de mesure dans DTW

Nous ajoutons à l'algorithme DTW classique des tolérances sur l'amplitude, afin de prendre en compte l'imprécision de la mesure et sur la déformation temporelle afin d'imposer des limites de décalage entre deux séries temporelles.

	1		...		j		...			10
1	0	0	1	2	2	2	2	2	2	2
	0	0	1	2	2	2	2	2	2	2
⋮	0	0	1	2	2	2	2	2	2	2
	0	0	1	2	2	2	2	2	2	2
	0	0	1	2	2	2	2	2	2	2
i	1	1	0	0	1	2	3	3	3	3
	2	2	0	0	1	2	3	4	4	4
⋮	3	3	0	0	1	2	3	4	5	5
	3	3	1	1	0	0	0	0	0	0
10	3	3	2	2	0	0	0	0	0	0

Tableau 5.2 — Matrice des distances cumulées

Imprécision sur l'amplitude Dans le cadre multi-modèle que nous proposons, la prédiction est dynamique et les simulations sont numériques. L'imprécision de mesure est donc ajoutée a posteriori. Une méthode très simple est d'intégrer cette tolérance dans l'algorithme DTW lors du calcul de la matrice des distances locales **D**. Ainsi, si la distance euclidienne entre deux données x_j et y_j est inférieur à une tolérance λ alors elle est considérée comme nulle.

Nous verrons dans le chapitre 7 que le langage de modélisation Modelica permet notamment de typer les variables physiques en fonction de leur nature. Les tolérances λ peuvent donc être fixées en fonction de ce typage.

Utilisation d'une contrainte temporelle Nous avons vu dans la section précédente que l'algorithme DTW dans sa forme classique permet d'absorber des dilatations, compression ou décalages temporels possibles entre deux signaux. Cependant, pour que ces déformations ne soient pas démesurées, il est possible d'ajouter une contrainte afin de les restreindre. La contrainte que nous utilisons est appelée la bande de Sakoe-Chiba [Sakoe et Chiba, 1978] et se formule de la manière suivante :

Bande de Sakoe-Chiba : La recherche du chemin de recalage entre les séquences de données $X = [x_i]_{1 \leq i \leq N}$ et $Y = [y_j]_{1 \leq j \leq M}$ peut être restreinte à la bande (ou fenêtre) caractérisée par la contrainte $|i - (N/M/j))| < R$ avec $R \in \mathbb{N}$, la largeur de la bande.

Le calcul de la matrice Δ des distances cumulées est donc restreinte à cette bande. Les ligne 9 à 20 de l'algorithme DTW classique (algorithme 5.1) sont alors changées par les instructions donnée par l'algorithme 5.2.

Si nous reprenons l'exemple précédent, et que nous appliquons la contrainte de Sakoe-Chiba avec une largeur de bande $R = 3$, nous obtenons la matrice donnée figure 5.3. La distance minimale de recalage qui résulte de cette contrainte est alors $DTW = 3$. Les deux signaux X et Y sont alors différents avec cette contrainte temporelle.

Algorithme 5.2 — Modification de DTW classique avec la contrainte de Sakoe-Chiba

$\mathbf{\Delta} = \left[\delta(i,j) = \text{infinity}\right]_{1 \leq i \leq N, 1 \leq j \leq M}$
pour $i = 2 \ldots R$ **faire** $\delta(i,1) = d(i,1) + \delta(i-1,1)$
pour $j = 2 \ldots R$ **faire** $\delta(j,1) = d(j,1) + \delta(j-1,1)$
% Calcul des autres valeurs de Δ **pour** $i = 2 \ldots N$ **faire**
  pour $j = 2 \ldots M$ **faire**
    si $|i - (N/M/j))| < R$ **alors**
      $\delta(i,j) = d(i,j) + \min(\delta(i-1,j), \delta(i,j-1), \delta(i-1,j-1))$
    fin
  fin
fin
% Valeur de la distance de recalage minimale
retourner $DTW = \delta(N,M)$

	1		…		j		…			10
1	0	0	1	-	-	-	-	-	-	-
	0	0	1	2	-	-	-	-	-	-
⋮	0	0	1	2	2	-	-	-	-	-
	-	0	1	2	2	2	-	-	-	-
	-	-	1	2	2	2	2	-	-	-
i	-	-	-	1	2	3	3	3	-	-
	-	-	-	-	2	3	4	4	4	-
⋮	-	-	-	-	-	3	4	5	5	5
	-	-	-	-	-	-	3	3	3	3
10	-	-	-	-	-	-	-	3	3	3

Tableau 5.3 **—** Matrice des distances cumulées avec la contrainte de Sakoe-Chiba

5.4.2 Génération des symptômes fonctionnels

Dans le chapitre précédent, nous avons présenté l'abstraction d'une série temporelle en une séquence d'épisodes qualitatifs (section 4.4) puis en une séquence d'épisodes sémantiques représentant un symptôme (section 4.5). Le problème de génération automatique des séquences d'épisodes qualitatifs est traité par un certain nombre de travaux dans le domaine du diagnostic mais essentiellement pour des systèmes continus [Ayrolles *et al.*, 1995; Melendez et Colomer, 2001; Gentil *et al.*, 2005]. Les algorithmes existants analysent les tendances (dérivées premières) et courbures (dérivées secondes) des signaux mesurés pour en déduire des épisodes.

Notre objectif d'abstraction est moins fin que ce qui est proposé par ces travaux car nous souhaitons faire abstraction des phénomènes transitoires pouvant exister dans le mode continu d'un système. De plus, les signaux générés par la simulation sont en général non dérivables car ils contiennent des points de commutation. Ce sont en général ces commutations qui délimitent les épisodes qualitatifs que nous voulons construire.

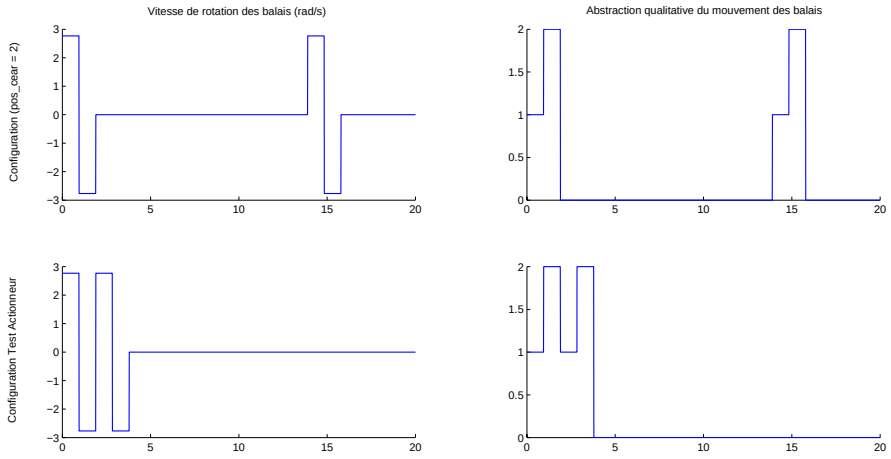


Figure 5.6 — Abstraction de la vitesse de rotation du balai essui-vitre dans les configurations commande utilisateur ($poc\_cear = 2$) et test actionneur

Nous utilisons alors une méthode très simple qui consiste à utiliser des seuils directement dans la simulation afin d'abstraire les différentes valeurs qualitatives prises par une variable physique au cours du temps. Par exemple, il est possible de mettre des seuils directement sur la variable w_b du composant `tringlerie` pour générer les trois valeurs qualitatives possibles de cette variable qui sont $\{0, > 0, < 0\}$. Sur la figure 5.6, nous présentons le résultat de cette abstraction dans les deux contextes d'activation qui sont le test actionneur et l'activation par l'utilisateur et correspondant respectivement aux fonctions `FONCTION_3` (section 5.2.2) et `FONCTION_1` (section 4.5) du système essuie-vitre. Les valeurs $\{0, 1, 2\}$ générées par la simulation correspondent aux valeurs $\{0, > 0, < 0\}$.

Ainsi, il est possibles de générer des series temporelles qui ne prennent que des valeurs entières pour les tests qui sont de type LEV_j='teleological'. Ces séries peuvent ensuite faire l'objet d'une classification avec l'algorithme DTW. Ensuite, nous attachons manuellement un symptôme fonctionnel à chaque classe formée pour ce type de test.

5.5 Conclusion

Dans ce chapitre, nous avons présenté le processus de prédiction des tests que nous proposons dans un cadre multi-modèle. L'application sur le système essuie-vitre est discutée dans le chapitre 7. Le choix d'une simulation numérique est lié au fait que nous utilisons dans l'application finale un langage de modélisation standard qui est Modelica. Ce cadre est suffisant pour l'anticipation de cas de défauts extrêmes mais il ne permet pas actuellement de prendre en compte les cas de défauts de déviation de paramètre que nous ne traitons pas et qui nécessiteraient une simulation qualitative difficile à gérer avec l'outil de simulation considéré. Notons que ce problème avait été partiellement pris en compte dans [Olive, 2003] pour des réseaux résistifs par une méthode qualitative à base de cohérence utilisée de manière complémentaire à la méthode AGENDA. La méthode qualitative était exécutée lorsque le parcours de l'arbre de diagnostic généré par AGENDA échouait.

6 Séquencement des tests

L'objectif de ce chapitre est de présenter la phase de séquencement des tests. Celle-ci prend en entrée la matrice de signature des défauts **A** procurant le résultat théorique de chacun des tests sous l'occurrence de chaque défaut anticipé. Nous ajoutons par ailleurs les notions de coût dynamique pour les tests et de probabilité pour l'occurrence des défauts. Comme nous l'avons défini dans le chapitre 2, le problème consiste à construire une séquence de tests qui est capable d'isoler chacun des défauts en minimisant le coût J de l'arbre de diagnostic que nous rappelons par l'equation (6.1) où $d_{ij} = 1$ si le test s_j est utilisé dans la séquence de tests menant à l'identification du défaut f_i et $d_{ij} = 0$ sinon.

$$J = \sum_{i=0}^{n_F} p_i \times \left(\sum_{j=1}^{n_S} d_{ij} \times c_j \right) \tag{6.1}$$

Nous présentons les deux alternatives de séquencement que nous avons évoquées dans le chapitre 2 :

- La première stratégie consiste à choisir, à chaque étape la session de diagnostic, le meilleur prochain test en fonction du contexte courant. Elle apporte alors une solution locale au problème de séquencement des tests (cf. section 2.3.2).
- La seconde stratégie, celle développée dans AGENDA, consiste à générer un arbre de diagnostic optimal par un algorithme AO* en minimisant le coût J. Elle apporte alors une solution globale au problème de séquencement des tests.

Bien que nous privilégions le caractère interactif et dynamique de la première stratégie, nous présentons également celle basée sur un algorithme AO* afin de comparer les deux approches dans le chapitre suivant. Nous proposons notamment deux heuristiques prenant en compte des coûts dynamiques de tests pour la génération des arbres de diagnostic. Par ailleurs, nous nous plaçons dans le cas de tests multivalués exclusifs, contrairement à AGENDA qui, par le cadre de prédiction ensembliste des tests, devait traiter le cas de tests multivalués non exclusifs.

6.1 Analyse de la matrice de signature des défauts

Dans cette section, nous présentons deux processus de traitement de la matrice des signatures de défauts permettant, d'une part, d'éliminer les tests non informatifs (réduction

du nombre de colonnes de **A**) et d'autre part de réaliser un analyse discriminante de l'ensemble des défauts (analyse des lignes de **A**).

6.1.1 Tests non informatifs

Lorsqu'un test s_j n'a qu'une seule modalité possible (i.e. le résultat du test est le même quelque soit le défaut anticipé), alors il ne peut pas séparer l'ensemble $\mathcal{F}$ des $n_F + 1$ défauts anticipés $f_i, i \in \{0, ..., n_F\}$. Dans ce cas, le test s_j est dit *non informatif* et il est supprimé de l'ensemble des tests prédits $\mathcal{S}$. Dans la matrice de signature des défauts **A**, cela se traduit par la suppression de la colonne relative au test s_j. Ainsi, un traitement itératif permet de supprimer tous les tests non informatifs de $\mathcal{S}$, ce qui revient à supprimer toutes les colonnes correspondantes de **A**.

6.1.2 Analyse discriminante

Deux défauts f_{i_1} et f_{i_2} de $\mathcal{F}$ sont non discriminables si $\forall j \in \{1, ..., n_s\}$, $a_{i_1,j} = a_{i_2,j}$, c'est-à-dire si les lignes de la matrice de signature des défauts **A** indexées par i_1 et i_2 sont identiques. Ainsi, il est possible de construire les sous ensembles de $\mathcal{F}$ des défauts non discriminables. Ces sous ensembles forment alors des feuilles de l'arbre de diagnostic.

Cependant, comme l'objectif du diagnostic est d'isoler le composant remplaçable défaillant et pas nécessairement d'identifier le mode de défaut présent, la notion de sous ensemble de défauts non discriminables est à rapprocher de la notion de réparation définie dans le chapitre précédent (section 5.1.4). Ainsi, si tous les défauts d'un sous-sensemble de défauts non-discriminables de $\mathcal{F}$ correspondent à la même réparation, alors il n'y a pas de problème vis-à-vis de l'objectif de diagnostic.

6.2 Coûts des tests et probabilités de défauts

Dans cette section, nous présentons comment les coûts dynamiques des tests et les probabilités de défaut requis par le problème de séquencement des tests sont gérés pendant la phase de diagnostic.

6.2.1 Evaluation dynamique du coût des tests

L'évaluation des coûts c_j de chacun des tests s_j est évalué à chaque étape de la session de diagnostic comme étant la somme de trois composantes : le coût intrinsèque c_j^{cst}, le coût de configuration c_j^{conf} et le coût de l'instrumentation c_j^{inst} :

$$c_j = c_j^{cst} + c_j^{conf} + c_j^{inst} \tag{6.2}$$

Le coût de configuration est une valeur dynamique correspondant au coût de réglage de la configuration $CONF_j$ du test s_j en fonction de la configuration courante (celle correspondant au test précédent). S'il existe N_{conf} configurations de tests possibles, les coûts des

transitions entre ces configurations peuvent être stockés dans une matrice $\mathbf{\Phi}$ de dimension $N_{conf} \times N_{conf}$ dont chaque élément $\phi_{l,k}, (l,k) \in \{1, ..., N_{conf}\}^2$ représente le coût induit pour passer d'une configuration de test $CONF_l$ à une nouvelle configuration $CONF_k$.

Le coût de l'instrumentation c_j^{inst} est une valeur dynamique correspondant au coût induit par l'installation de l'instrumentation $INST_j$ nécessaire à la réalisation du test s_j. Ce coût inclut le coût correspondant à l'opération d'installation du dispositif de mesure et celui correspondant aux opérations de démontage nécessaires pour accéder au composant (connecteur) au niveau duquel la mesure doit être effectuée.

Le coût intrinsèque c_j^{int} est une valeur constante représentant le coût de l'opération requise pour réaliser le test s_j une fois que le système est dans la configuration $CONF_j$ et que l'instrumentation $INST_j$ est installée.

Dans [Faure, 2001] et [Olive, 2003], un modèle topologique était proposé pour représenter la manière dont le système à diagnostiquer devait être démonté. Ce modèle se présentait sous la forme d'un arbre d'assemblage et faisait l'hypothèse que le démontage était l'opération inverse de cet arbre pour estimer les coûts d'accessibilité des points de mesure. Cependant, comme c'était déjà le cas pour ces travaux précédents, nous n'avons pas les données suffisantes pour mettre en œuvre ce type de modèle.

Nous avons donc préféré une approche plus simple utilisant des ordres de grandeurs pour chaque coût c_j^{inst}. Le réglage initial de ces coûts est laissé à l'appréciation de l'expert de maintenance suivant une échelle comprise dans l'intervalle c_j^{inst}. Pendant le séquencement des tests ce coût est pris en compte dans le coût dynamique c_j du test s_j seulement lorsque l'instrumentation $INST_j$ doit être installée pour la première fois, il devient nul pour les étapes suivantes de la session de diagnostic. Ainsi, lorsqu'une instrumentation est installée, elle le reste durant toute la session de diagnostic.

6.2.2 Probabilités des défauts

Les probabilités d'occurrence des défauts sont utilisées par de nombreuses méthodes de diagnostic car elles permettent de focaliser le diagnostic sur les défauts les plus probables d'abord. Celles-ci avaient notamment été utilisées dans la méthode AGENDA. Dans cette section, nous présentons comment sont saisies les probablités a priori d'occurence des défauts et comment celles-ci évoluent après chaque test réalisé.

Probabilités a priori des défauts

Dans le second chapitre, nous avons vu que le problème de séquencement des tests (section 2.3.2) requiert la définition d'un ensemble $\mathcal{P}$ de probabilités d'occurrence a priori $p_i, i \in \{0, ..., n_F\}$ des défauts anticipés $f_i, i \in \{0, ..., n_F\}$ telles que, sous l'hypothèse de défaut unique, $\sum_{i=0}^{n_F} p_i = 1$.

Dans la réalité les probabilités d'occurrenece a priori des défauts ne sont pas disponibles de manière précise car elles ne peuvent pas être figées en conception. Ces données évoluent

en fait au cours du cycle de vie d'un véhicule et des différents retours d'expérience des services de maintenance des constructeurs. Par ailleurs, nous n'avons pas nécessairement besoin de valeurs précises mais plutôt de savoir que certains défauts sont plus probables que d'autres.

Ainsi, dans le prototype final, nous proposons une saisie par ordres de grandeurs des probabilités a priori qui sont ensuite converties sous forme de probabilités quantitatives. Il est également possible de tenir compte d'un retour d'information des services de maintenance (notion de « boucle de retour ») indiquant que certains défauts sont plus récurrents que d'autres.

Evolution des probabilités au cours d'une session de diagnostic

Au début d'une session de diagnostic, lorsqu'aucun test n'a encore été effectué, les probabilités des défauts sont initialisées par les probabilités a priori. Ces probabilités doivent ensuite être mises à jour après chaque réalisation d'un test. Ainsi, si à une étape donnée de la session de diagnostic, il reste un ensemble $\mathcal{F}' \subset \mathcal{F}$ de $n_{F'}$ défauts $f_i, i \in \{1, ..., n_{F'}\}$ à discriminer et que le test s_j ayant n_m^j modalités possibles ($MOD_k^j, k \in \{1, ..., n_m^j\}$) doit être réalisé, alors les deux probabilités suivantes doivent être calculées :

- la probabilité $P(s_j = MOD_k^j)$ que le résultat du test s_j à réaliser soit égal à la modalité MOD_k^j
- la probabilité d'occurrence $P(f_i|s_j = MOD_k^j)$ du défaut f_i sachant que le test s_j a été effectué et que son résultat est la modalité MOD_k^j. Celle-ci est calculée par la règle de Bayes.

Les équations permettant de les calculer sont les suivantes :

$$\begin{cases} P(s_j = MOD_k^j) & = & \sum_{i=1}^{n_{F'}} P(s_j = MOD_k^j|f_i) \times P_c(f_i) \\ P(f_i|s_j = MOD_k^j) & = & \dfrac{P(s_j = MOD_k^j|f_i) \times P_c(f_i)}{P(s_j = MOD_k^j)} \end{cases} \tag{6.3}$$

avec $P_c(f_i)$ la probabilité courante d'occurrence du défaut f_i avant la réalisation du test s_j et $P(s_j = MOD_k^j|f_i)$ égal à 0 ou 1 pour des tests exclusifs[1] (pour un test donné, il n'existe qu'une seule modalité par défaut anticipé).

6.3 Stratégie du prochain meilleur test

La stratégie du prochain meilleur test est une stratégie locale qui permet de réaliser un séquencement dynamique des tests où, à chaque étape, l'algorithme recherche le prochain meilleur test à proposer à l'opérateur en fonction du contexte courant, seule l'étape suivante est explorée (« one step look ahead strategy »).

[1]Notons que dans AGENDA [Faure, 2001; Olive, 2003] la probabilité $P(s_j = MOD_k^j|f_i)$ pouvait être comprise ente 0 et 1 car les tests étaient non exclusifs

L'approche la plus courante est celle de la prochaine meilleure mesure proposée par [de Kleer et Williams, 1987; de Kleer *et al.*, 1992b] et qui maximisee le gain d'information (entropie des probabilités) des différent diagnostics. Dans ce critère, aucun coût de test n'est pris en compte. Afin de les intégrer, nous utilisons le critère de gain d'information par unité de coût (équation (2.2)) que nous appliquons à des tests multivalués.

Si à une étape donnée de la session de diagnostic, il reste un ensemble $\mathcal{F}' \subset \mathcal{F}$ de $n_{F'}$ défauts $f_i, i \in \{1, ..., n_{F'}\}$ à discriminer, alors le prochain meilleur test s_j, ayant n_m^j modalités ($MOD_k^j, k \in \{1, ..., n_m^j\}$), est celui qui maximise le gain d'information $IG(s_j)$. Le calcul de $IG(s_j)$ est donné par l'équation (6.4) et le maximum K recherché est donné par l'équation (6.5).

$$IG(s_j) = -\sum_{k=1}^{n_m^j} P(s_j = MOD_k^j) \log(P(s_j = MOD_k^j)) \tag{6.4}$$

$$K = \max_j \left(\frac{IG(s_j)}{c_j} \right) \tag{6.5}$$

avec $P(s_j = MOD_k^j)$, la probabilité que le résultat du test s_j à réaliser soit égal à la modalité MOD_k^j et calculé par l'équation (6.3). c_j est le coût dynamique du test s_j calculé par l'équation (6.2). Les résultats obtenus par cette stratégie sont discutés dans le prochain chapitre.

6.4 Génération d'un arbre de diagnostic optimal

Comme nous l'avons précisé dans le chapitre 2 de ce manuscrit, la méthode AGENDA utilisait une solution globale au problème de séquencement des tests qui consistait à générer, à partir de la matrice des signatures des défauts, un arbre de diagnostic optimal, c'est-à-dire qui minimise la fonction objectif J donnée par l'équation (2.1). Un algorithme AO*, qui est un algorithme basé heuristique, était utilisé car cette méthode est particulièrement adaptée pour la recherche d'une solution optimale dans un arbre de recherche ET/OU.

6.4.1 Présentation de l'algorithme AO*

On appelle arbre de recherche ET/OU implicite l'arbre qui représente toutes les solutions possibles d'un problème partant des éléments de base de ce problème. Dans le cas du problème de séquencement des tests, les éléments de base sont l'ensemble des défauts $\mathcal{F}$, l'ensemble des tests $\mathcal{S}$ et la matrice de signatures des défauts $\mathbf{A}$. Les solutions possibles sont tous les arbres de diagnostic T qui permettent de discriminer l'ensemble des défauts $\mathcal{F}$ en utilisant un sous-ensemble des tests de $\mathcal{S}$.

L'idée de l'algorithme AO* est de ne développer de manière itérative que les parties de l'arbre de recherche implicite, correspondant à la solution la plus intéressante du problème, en accord avec la fonction objectif à optimiser (equation (2.1)). Ce sous-arbre de l'arbre de recherche ET/OU implicite est sélectionné selon l'évaluation heuristique la plus pertinente

et est appelé l'arbre explicite. Plus l'évaluation heuristique est pertinente, moins l'arbre explicite contient de noeuds. En outre, l'admissibilité de l'heuristique garantit que la solution trouvée est optimale. Celle-ci peut être trouvée à l'intérieur de l'arbre de recherche ET/OU explicite. Par conséquent, l'arbre de diagnostic optimal T^* est un sous-arbre de l'arbre de recherche ET/OU explicite développé par l'algorithme AO*.

A chaque nœud OU N^O de l'arbre de diagnostic, représentant un ensemble d'ambiguïté, nous calculons la valeur estimée courante $F(N^O)$ du coût optimal $J^*(N^O)$ du sous arbre optimal $T^*(N^O)$ partant de ce nœud OU.

Initialisation

A l'initalisation de l'algorithme AO*, l'arbre de recherche ET/OU explicite n'est composé que de sa racine appelée N_R^O qui est un noeud OU représentant l'ensemble total $\mathcal{F}$ des n_F+1 défauts $f_i, i \in \{0, ..., n_F\}$ (i.e. ensemble d'ambiguïté total). La probabilité d'occurrence de chaque défaut f_i au nœud N_R^O, notée $P_{N_R^O}(f_i)$, correspond aux probabilités d'occurrence a priori $p_i, i \in \{0, ..., n_F\}$.

Soit T_a^* l'arbre de diagnostic optimal courant et L^* l'ensemble des feuilles dévelopables de T_a^* (i.e. nœuds OU ne correspondant ni à des défauts isolés, ni à une réparation). T_c^* et L^* sont alors réduits à N_R^O lors de l'initialisation de l'algorithme. Comme N_R^O est une feuille de l'arbre courant, le coût courant $F(N_R^O)$ est égal à la valeur de l'heuristique $H(N_R^O)$ au niveau du nœud N_R^O.

Traitement itératif

A chaque étape de l'algorithme AO*, nous retirons un nœud OU N^O de l'ensemble L^* courant afin de le développer (figure 6.1).

Les n_A nœuds ET $N_j^A, j \in \{1, ..., n_A\}$ fils du nœud OU N^O sont ensuite créés. Chaque N_j^A se rapporte à un test $s_j, j \in \{1, ..., n_s\}$ de l'ensemble des tests pouvant être réalisés sur le système. Soit alors le coût dynamique c_j du test s_j (calculé suivant le mode opératoire donné en section 6.2.1) et n_m^j, le nombre de ses modalités $MOD_k^j, k \in \{1, ..., n_m^j\}$.

Les n_m^j nœuds OU $N_{j,k}^O, k \in \{1, ..., n_m^j\}$, fils de chaque nœud N_j^A sont ensuite créés. Chaque nœud $N_{j,k}^O$ correspond à l'ensemble d'ambiguité résultant de l'ensemble d'ambiguité N^O sachant que la $k^{\text{ème}}$ modalité a été observée pour le test s_j. Pour chaque nœud $N_{j,k}^O$ créé, le coût estimé $F(N_{j,k}^O)$ est égal à la valeur de l'heuristique $H(N_{j,k}^O)$.

Soit $P_N(f_i)$ la probabilité d'occurrence du défaut f_i au nœud N^O et $P(f_i|s_j = MOD_k^j)$ la probabilité conditionnelle que la modalité MOD_k^j soit observée pour le test s_j si le défaut f_i est présent. Pour chaque nœud N_j^A correspondant au test s_j, la probabilité d'occurrence $P_{N_{j,k}^O}(f_i)$ de chaque défaut f_i du nouvel ensemble d'ambiguïté représenté par un nœud $N_{j,k}^O$ créé peut être calculé par les équations (6.3). La première équation correspond à l'évaluation de la probabilité $P_{N_j^A}(s_j = MOD_k^j)$ que la modalité MOD_k^j soit observée pour le test s_j au nœud N_j^A.

Enfin, un traitement récursif est effectué en partant du nœud N^O et en remontant jusqu'au

nœud racine N_R^O. Ce traitement consiste à mettre à jour successivement les coûts F des nœuds rencontrés et à mettre à jour le marquage des nœuds ET correspondant à l'arbre de diagnostic optimal courant T_a^*. Soit alors N_c le nœud courant sur lequel ce traitement est effectué.

- *Si N_c est un nœud OU*
  Soit n_c le nombre de nœuds ET fils de N^c appelés $N_j^c, j \in \{1, ..., n_c\}$. Au plus un de ces nœuds est marqué. Le marquage de ce nœud ET est retiré.
  Le coût $F(N_c)$ est calculé par l'équation (6.6) et le $j^{\text{ème}}$ nœud pour lequel ce minimum est atteint est marqué.

$$F(N_c) = \min_{j=1}^{n_c} F(N_j^c) \tag{6.6}$$

- *Si N_c est un nœud ET*
  Soit n_c le nombre de nœuds OU fils de N^c appelés $N_k^c, k \in \{1, ..., n_c\}$
  Comme N_c correspond à un test s_j ayant un coût c_j et n_m^j modalités, alors les nœuds $N_k^c, k \in \{1, ..., n_c\}$ se rapportent aux n_m^j modalités du test s_j et $n_c = n_m^j$.
  La probabilité d'atteindre le nœud N_k^c en partant du nœud N_c, notée $P_{N_c}(N_k^c)$, correspond à la probabilité d'observer la modalité MOD_k^j pour le test s_j au nœud N_c, calculée suivant l'équation (6.3). Ainsi, pour le nœud courant N_c, le coût $F(N_c)$ est alors calculé par l'équation (6.7).

$$F(N_c) = c_j + \sum_{k=1}^{n_M^j} P(N_k^c) \times F(N_k^c) \tag{6.7}$$

L'ensemble L^* est mis à jour suivant le marquage de l'arbre de diagnostic optimal courant T_a^*.

Condition d'arrêt

L'algorithme AO* s'arrête lorsque l'ensemble $L^* = \{\emptyset\}$. Alors, l'arbre de diagnotic optimal courant T_a^* est l'arbre de diagnostic optimal T^* recherché dont le coût optimal au nœud racine N_R^O est $J^*(N_R^O) = F(N_R^O)$.

Notion d'heuristique

Comme nous avons pu le voir, l'algorithme AO* utilise une fonction d'évaluation heuristique (FEH), H pour orienter la recherche à chaque itération. Pour un noeud OU donné, une FEH est une fonction qui donne une estimation de la valeur de la fonction objectif J^* du sous-arbre optimal de diagnostic ayant ce noeud OU comme racine.

Soit N^O un noeud OU, et $T_{N^O}^*$ le sous-arbre de diagnostic optimal ayant N^O comme racine. Considérons alors $J^*(N^O)$ le coût exact de $T_{N^O}^*$ et $H(N^O)$ le coût estimé de $T_{N^O}^*$. Il a été démontré dans [Mahanti et Bagchi, 1985] que si, pour chaque noeud OU N^O, $H(N^O) \leq J^*(N^O)$, alors l'algorithme converge vers un arbre de diagnostic optimal T^*. Dans ce cas, la fonction d'évaluation heuristique est dite admissible.

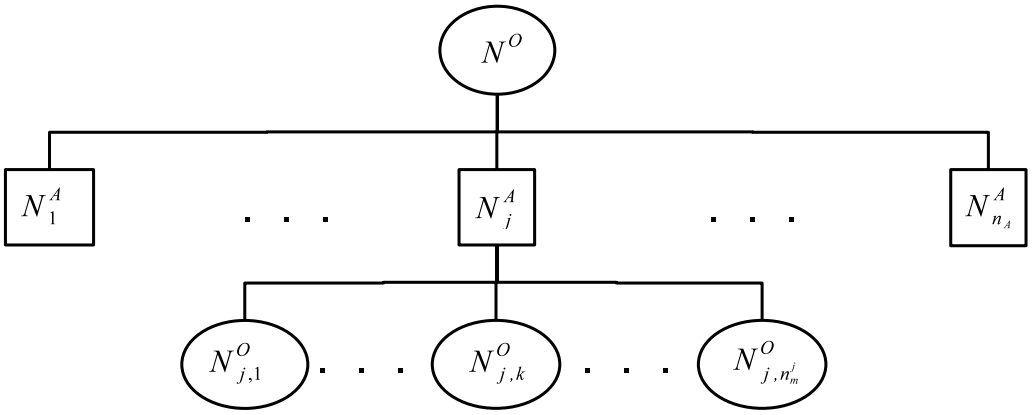


Figure 6.1 — Développement d'un nœud OU et de tous ses fils ET

L'idée fondamentale d'un algorithme de recherche utilisant une FEH est de tenter de se rapprocher à chaque itération de l'objectif en privilégiant les possibilités qui en sont directement plus proches compte tenu de la valeur de l'heuristique. Toutes les possibilités qui ne semblent pas se rapprocher de l'objectif sont mises de côté, mais pas supprimées. Elles sont simplement mises dans une liste de possibilités à explorer si jamais la solution actuellement développée ne s'avère pas la meilleure. Dans l'algorithme AO*, le marquage des nœuds permet de retourner en arrière et d'aller explorer d'autres nœuds dans le cas où la direction de recherche donnée par la FEH ne s'avère finalement pas la meilleure.

Par ailleurs, la vitesse de convergence de l'algorithme AO* est liée à la qualité de la FEH H. Plus $H(N^O)$ est proche de $J^*(N^O)$ sur l'ensemble des noeuds OU N, plus petit sera le nombre de noeuds OU développés inutilement pendant l'exécution de l'algorithme (i.e. noeuds OU qui n'apparaîtront pas dans l'arbre de diagnostic optimal définitif T^*).

Par la suite, nous présentons quatre heuristiques, les deux premières étant reprises des travaux de [Faure, 2001] et [Olive, 2003]. La première (FEH_1) est l'heuristique dite de Pattipati se base sur la génération d'arbres optimaux pour des tests binaires. La seconde, appelée FEH_2, était proposée dans [Faure, 2001] pour étendre l'heuristique HEF_1 à des tests multivalués, exclusifs et à coûts fixes. Les deux suivantes, appelées FEH_{Dyn1} et FEH_{Dyn2}, sont deux extensions de FEH_1 que nous proposons pour introduire la notion de coût dynamique des tests.

6.4.2 Heuristique de Pattipati pour des tests binaires

L'heuristique de Pattipati, notée FEH_1, est une heuristique qui permet de générer des arbres de diagnosic optimaux à partir du problème de séquencement des tests défini par un ensemble $\mathcal{F}$ de n_F défauts f_i de probabilités d'occurrence a priori p_i et un ensemble $\mathcal{S}$ de n_S tests exlusifs binaires avec des coûts c_j pouvant être différents.

La résolution de ce problème se réalise à partir d'un algorithme AO* utilisant une heuristique, dont l'admissibilité est démontrée dans [Faure, 2001], qui est calculée en deux étapes :

1. La première étape consiste à générer un arbre de diagnostic optimal T_0^* à partir de l'algorithme de Huffman pour un problème plus simple dans lequel nous supposons que

tous les tests permettant de discriminer les défauts sont disponibles et qu'ils sont à coût constant égal à 1. Le coût J_0^* de l'arbre obtenu est appelé longueur moyenne du code de Huffman.

2. La deuxième étape consiste à calculer l'heuristique FEH_1 à partir de J_0^* et en prenant en compte des coûts de tests différents.

Algorithme de Huffman

Définition 2 (Condition d'ordre sur les noeuds). *Soit T un arbre et A et B deux noeuds distincts situés sur deux niveaux a et b de T, respectivement, $a, b \in \{0, ..., d\}$ où d est la profondeur de T. Soit A_i et B_j avec $i \in \{1, ..., n_A\}$ et $j \in \{1, ..., n_B\}$ les n_A et n_B fils de A et B, respectivement.*

Soit x_N le sous-ensemble d'ambiguïté associé au noeud N de T et $P(x_N)$ sa probabilité d'occurrence a priori. Pour chaque ensemble d'ambiguïté x, la probabilité $P(x)$ est calculée comme la somme des probabilités des défauts qui constituent x.

Un arbre T vérifie la condition d'ordre sur les noeuds si et seulement si toutes les paires (A, B) de noeuds vérifient les équations (6.8).

$$\begin{cases} (a < b) & \Rightarrow \quad P(x_A) \geq P(x_B) \\ (a = b) \wedge (P(x_A) \geq P(x_B)) & \Rightarrow \quad P(x_{A_i}) \geq P(x_{B_j}) \\ \forall i, j \in \{1, ..., n_A\} \times \{1, ..., n_B\} & \end{cases} \tag{6.8}$$

Cette condition est notée NOC (Node Ordering Condition) dans la suite de ce manuscrit.

Considérons alors Π_0 le problème de séquencement des tests défini par un ensemble F de n_F défauts f_i ayant comme probabilité d'occurrence a priori p_i et un ensemble S de n_S tests exclusifs binaires s_j avec un coût c_j constant égal à 1.

Proposition 1 (Optimalité dans le cas de la NOC). *Etant donné T, un arbre de diagnostic proposé pour le problème de séquencement des tests Π_0, si T vérifie la NOC alors T est optimal.*

Cette proposition est démontrée dans [Faure, 2001].

Algorithme 6.1 — Algorithme d'Huffman

```
1  i ← n_F ;
2  Créer n_F feuilles correspondant aux n_F défauts ;
3  tant que (i > 1) faire
4  |  Ordonner les i défauts par probabilité décroissante ;
5  |  p_{i-1} ← p_{i-1} + p_i (i.e. les 2 plus petites probabilités) ;
6  |  Créer un nouveau noeud f_{i-1} père de f_i et de l'ancien noeud f_{i-1} ;
7  |  Retirer le défaut f_i de l'ensemble des défauts ;
8  |  i ← i − 1 ;
9  fin
10 retourner l'arbre optimal créé T_0^*
```

Proposition 2 (Optimalité de Huffman). *Dans le cas de tests exclusifs binaires, sous l'hypothèse de coûts de tests unitaires, l'algorithme de Huffman construit un arbre de diagnostic binaire optimal par rapport à la fonction objectif J.*

Cette proposition est démontrée dans [Faure, 2001].

Calcul de l'heuristique de Pattipati

Ainsi, l'algorithme de Huffman construit un arbre de diagnostic optimal T_0^* composé d'un sous-ensemble de tests s_j exclusifs binaires de coût $c_j = 1$.

L'heuristique de Pattipati FEH_1 prend en compte des coûts de tests différents telle que sa valeur H soit toujours inférieure à la valeur exacte du coût de l'arbre J^*, i.e. FEH_1 est admissible.

Soit N^O un noeud OU et $J_0^* = J(T_0^*)$ la valeur de la fonction objectif de l'arbre de diagnostic optimal T_0^* obtenu par l'algorithme de Huffman pour l'ensemble des défauts qui est contenu dans N sous l'hypothèse de coûts de tests unitaires. Les n_S tests disponibles sont ordonnés par coûts croissants tels que

$$0 < c_1 \leq ... \leq c_{n_S}$$

La valeur H de l'heuristique proposée par Pattipati est alors exprimée par l'équation 6.9 avec $\lfloor J_0^* \rfloor$ la partie entière de J_0^*.

$$H(N^O) = \sum_{j=1}^{\lfloor J_0^* \rfloor} c_j + \left((J_0^* - \lfloor J_0^* \rfloor) \times c_{\lfloor J_0^* \rfloor + 1} \right) \tag{6.9}$$

6.4.3 Extension à des tests multivalués à coûts fixes

L'hypothèse de tests multivalués rend la discrimination d'un ensemble de défauts par un ensemble de tests plus facile que par des tests binaires. En effet, l'arbre de diagnostic optimal obtenu par l'algorithme de Huffman sous l'hypothèse de tests binaires est plus profond que celui obtenu avec des tests multivalués.

Il a été démontré dans [Faure, 2001] que considérer les tests qui ont le plus grand nombre de modalités d'abord (cf. définition 3) dans l'arbre de Huffman, minimise la profondeur de cet arbre et préserve la propriété d'admissibilité de l'heuristique.

Définition 3 (Condition du plus grand nombre de modalités d'abord (GFC)). *Soit T un arbre de diagnostic ayant une profonfeur d. La condition du plus grand nombre de modalités d'abord est vérifiée si et seulement si chaque noeud du $k^{ième}$ niveau de T a un nombre de fils égal au $k^{ième}$ plus grand nombre de modalités disponibles tel que $k \in \{0, ..., d\}$.*

Soit T_0^* l'arbre de diagnostic optimal obtenu par l'algorithme de Huffman pour un ensemble $\mathcal{F}$ de défauts et un ensemble $\mathcal{S}$ de tests exclusifs binaires, sous l'hypothèse de coût

des tests unitaire. L'extension à des tests multivalués de l'algorithme modifie T_0^* afin d'obtenir un arbre de diagnostic optimal T_1^*.

Considérons alors Π_1 le problème de séquencement des tests défini par un ensemble $\mathcal{F}$ de n_F défauts f_i ayant des probabilités d'occurrence a priori p_i et un ensemble $\mathcal{S}$ de n_S tests exclusifs multivalués s_j possédant n_m^j modalités et un coût c_j constant égal à 1. Le sous-ensemble de tests possédant n_m^j modalités est noté $S_{n_m^j}$; $Card(S_{n_m^j}) \geq 1$ dès qu'au moins un test $s_j \in S_{n_m^j}$.

Alors, pour les différentes valeurs n_m^j, supposons que tous les $Card(S_{n_m^j})$ tests permettant de séparer l'ensemble $\mathcal{F}$ en n_m^j sous-ensembles, sont disponibles.

L'algorithme 6.2 permet alors de construire l'arbre T_1^* pour le problème Π_1 à partir de T_0^*. Les nœuds $N_k^j, k \in \{1, ..., n_N^j\}$ correspondent aux n_N^j nœuds du niveau j de l'arbre. L'indice $k = k_{max}^j$ correspond au nœud N_k^j du niveau j qui a la plus grande probabilité.

La valeur H de l'heuristique est ensuite calculée par l'équation (6.9) à partir du coût optimal $J_1^* = J(T_1^*)$ de T_1^* et en ordonnant les tests par ordre de coût croissant.

Algorithme 6.2 — Extension de l'algorithme de Huffman aux tests multivalués

```
1  Ordonner les n_S tests disponibles s_j ayant n_m^j modalités
2  tels que (n_m^1 ≥ ... ≥ n_m^{n_S} ≥ 2) ;
3  j ← 1 ;
4  T_1^* ← T_0^* ;
5  Terminé ← faux ;
6  tant que (Terminé = faux) ET (n_m^j > 2) faire
7      Terminé ← faux ;
8      pour chaque noeud non feuille N_l^{j-1} du niveau (j − 1) faire
9          n_m^N ← 2 ;
10         tant que (n_m^N < n_m^j) ET (∃ au moins un noeud non feuille N_k^j) faire
11             Supprimer N^j_{k_max^j} ;
12             n_m^N ← n_m^N + 1 ;
13         fin
14         si (n_m^N = n_m^j) alors Terminé ← faux ;
15     fin
16     Réordonner le niveau j de T_1^* ;
17     j ← j + 1 ;
18 fin
19 retourner T_1^* ;
```

6.4.4 Cas des tests multivalués à coûts dynamiques

Comme nous l'avons vu précédemment, l'algorithme AO* nécessite le calcul d'un coût dynamique à chaque occurrence d'un test dans l'arbre de recherche ET/OU. Ainsi, pour chaque nœud OU, la configuration de test courante et l'ensemble des instrumentations déjà installées doivent être connus.

Pour ce nœud OU, noté N^O, l'utilisation de l'heuristique FEH_2 donne une estimation $H(N^O)$ du coût $J^*(N^O)$ du sous-arbre optimal $T^*(N^O)$ en se basant uniquement sur la composante intrinsèque c_j^{cst} du coût d'un test s_j. Comme les coûts des tests sont en réalité dynamiques dans l'AO*, cette estimation est toujours inférieure au coût optimal $J^*(N^O)$. L'heuristique FEH_2 est donc admissible pour notre problème de diagnostic (voir démonstration détaillée dans [Faure, 2001]).

La prise en compte des coûts dynamiques de tests dans une fonction d'évaluation heuristique n'est en fait pas évidente. En effet, puisque l'arbre de diagnostic optimal $T^*(N^O)$ n'est pas connu à l'avance, alors les coûts dynamiques des tests contenus dans cet arbre ne le sont pas non plus.

Cependant, suite au cadre simple d'évaluation des coûts dynamiques donné en section 6.2.1, nous proposons deux heuristiques FEH_{Dyn1} et FEH_{Dyn2} basées sur la génération d'un arbre d'Huffman étendu à des tests multivalués (algorithme 6.2) puis sur l'estimation d'un coût dynamique minimal nécessaire pour discriminer l'ensemble des défauts.

Fonction d'évaluation heuristique FEH_{Dyn1}

Soit un nœud OU N^O représentant l'ensemble $\mathcal{F}$ des n_F défauts restant à discriminer $f_i, i \in \{1, ..., n_F\}$ et l'ensemble $\mathcal{S}$ de n_S tests possibles $s_j, j \in \{1, ..., n_S\}$ de coût c_j. L'estimation $H(N^O)$ du coût optimal $J^*(N^O)$ du sous arbre optimal $T^*(N^O)$ par la fonction d'évaluation heuristique FEH_{Dyn1} est réalisée de la manière suivante :

- dans un premier temps, l'algorithme 6.2 permet de calculer la longueur moyenne J_1^* du code d'Huffmann étendu aux tests multivalués. J_1^* donne alors le nombre de tests à utiliser pour le calcul de l'heuristique
- ensuite, les coûts dynamiques c_j des n_s tests de l'ensemble $\mathcal{S}$ sont calculés par l'équation (6.2)
- les n_S tests sont également ordonnés par ordre croissant des coûts intrinsèques c_j^{cst} tel que $0 < c_1^{cst} \leq \ldots \leq c_{n_S}^{cst}$
- l'estimation $H(N^O)$ du coût optimal $J^*(N^O)$ est alors donné par l'équation (6.10)

$$H(N^O) = \min_{j=1}^{n_S}(c_j) + \sum_{j=1}^{\lfloor J_1^* \rfloor - 1} c_j^{cst} + \left((J_1^* - \lfloor J_1^* \rfloor) \times c_{\lfloor J_1^* \rfloor}^{cst} \right) \tag{6.10}$$

avec c_j^{cst} le coût intrinsèque du test s_j de l'ensemble $\mathcal{S}$ des test ordonnés par ordre croissant des coûts privé du test déjà utilisé pour le calcul du premier membre de l'équation 6.10.

Le calcul de la valeur $H(N)$ de l'heuristique FEH_{Dyn1} se base alors sur le même principe que pour l'heuristique FEH_2 à la différence près que le premier coût de test pris en compte dans la formule correspond au coût dynamique minimal parmi les test n_s des tests restant et les coûts suivants correspondent aux $\lfloor J_1^* \rfloor$ coûts intrinsèques minimaux suivants.

Etant donné les conditions dans lesquelles nous nous sommes placés, l'heuristique FEH_{Dyn1} est admissible car nous garantissons une estimation de coût dynamique de test qu'il faudra au moins compter pour discriminer l'ensemble des défauts restants.

Fonction d'évaluation heuristique FEH_{Dyn2}

L'heuristique FEH_{Dyn2} donne une estimation $H(N^O)$ du coût optimal $J^*(N^O)$ du sous arbre optimal $T^*(N^O)$ qui ne prend en compte que les coûts intrinsèques des tests s_j et les coûts d'instrumentation de ces test. L'idée est de déterminer le coût minimal de l'instrumentation à installer pour réaliser les tests permettant de discriminer l'ensemble des défauts restants au niveau du nœud N^O.

1. comme pour l'heuristique FEH_{Dyn1}, la longueur moyenne du code d'huffman étendu J_1^* est préalablement calculée.
2. soit l'ensemble $INST$ des n_{inst} instrumentations $INST_k$, $k \in \{1, ..., n_{inst}\}$ requises par l'ensemble des n_S tests s_j , $j \in \{1, ..., n_S\}$ disponibles au niveau nœud OU N^O.
3. soit l'ensemble $COMB$ des N_{comb} différentes possibilités de combinaisons $COMB_l$, $l \in \{1, ..., N_{comb}\}$ d'instrumentations de $INST$:

$$N_{comb} = \sum_{l=1}^{N_{inst}} C_{N_{inst}}^{l} = \sum_{l=1}^{N_{inst}} \frac{N_{inst}!}{(l)! \times (N_{inst} - l)!} = 2^{N_{inst}} - 1$$

   Par exemple, si $INST = \{I_1, I_2, I_3\}$ alors il existe 7 combinaisons d'instrumentations possibles :

$$COMB = \big\{I_1, I_2, I_3, \{I_1, I_2\}, \{I_1, I_3\}, \{I_2, I_3\}, \{I_1, I_2, I_3\}\big\}$$

4. pour chaque $COMB_l$, $l \in \{1, ..., N_{comb}\}$, le coût c_l^{inst} qui serait induit par l'installation éventuelle des différentes instrumentations incluses dans cette combinaison est calculé. Si une instrumentation est déjà présente sur le système au niveau du nœud N^O, alors son coût d'installation est nul.
5. pour chaque $COMB_l$, $l \in \{1, ..., N_{comb}\}$, nous construisons le sous-ensemble $\mathcal{S}(COMB_l)$ de n_S^l tests s_i^l , $i \in \{1, ..., n_S^l\}$, de coût intrinsèque c_i^l, nécessitant une des instrumentations comprises dans $COMB_l$. Ces tests sont ordonnés par ordre croissant de coût intrinsèque tel que $0 < c_1^l \leq \ldots \leq c_{n_S^l}^l$
6. pour chaque $COMB_l$ tel que $l \in \{1, ..., N_{comb}\}$ et $n_S^l \geq \lfloor J_1^* \rfloor$, le coût $H_l(N^O)$ est évalué suivant l'équation (6.11)

$$H_l(N^O) = c_l^{inst} + \sum_{i=1}^{\lfloor J_1^* \rfloor} c_i^l + \Big((J_1^* - \lfloor J_1^* \rfloor) \times c_{\lfloor J_1^* \rfloor + 1}^l\Big) \tag{6.11}$$

7. la valeur $H(N^O)$ recherchée de l'heuristique FEH_{Dyn2} est alors définie par l'équation (6.12) :

$$H(N^O) = \min_l(H_l(N^O)) \tag{6.12}$$

Comme pour l'heuristique FEH_{Dyn1}, l'admissibilité de FEH_{Dyn2} repose sur celle de Pattipati car nous garantissons une estimation de coût d'instrumentation de test qu'il faudra au moins compter pour réussir à discriminer l'ensemble des défauts restant à discriminer. Notons également que le coût de changement de configuration n'est pas pris en compte dans cette heuristique.

6.5 Discussion

Les deux stratégies de test que nous avons présentées dans cette section on été intégrées dans l'application finale. Cela nous permet de comparer les deux approches dans le chapitre 7.

Par ailleurs, même si nous pensons que la stratégie du prochain meilleur test permet plus de souplesse pour le garagiste, l'utilisation d'arbres de diagnostic reste actuellement un standard en maintenance automobile. Ainsi, nous laissons le choix aux constructeurs automobile de la stratégie à adopter dans leur réseau après-vente.

Troisième partie

Prototype logiciel et validation

7 Mise en œuvre et résultats

Dans ce chapitre, nous présentons le prototype logiciel, appelé MODE-MBR, que nous avons développé afin de mettre en œuvre l'ensemble de la méthode de diagnostic exposée dans ce manuscrit. Ce prototype est basé sur l'utilisation du langage de modélisation Modelica pour décrire les systèmes mécatroniques.

Nous montrons ensuite les résultats obtenus pour le système essuie-vitre arrière présenté dans le chapitre 4 afin notamment de comparer les approches de séquencement par sélection du prochain meilleur test et par arbre de diagnostic optimal. Enfin, nous discutons de la validation du prototype sur des véhicules réels.

7.1 Présentation du protype MODE-MBR

L'arhitecture générale du prototype MODE-MBR est donnée par la figure 7.1. Celui-ci se décompose en deux outils différents : l'atelier auteur et l'outil de séquencement des tests.

7.1.1 L'atelier auteur

L'atelier auteur est un outil destiné à l'expert de diagnostic. Il permet de compiler hors-ligne (prédiction des tests) l'ensemble des données nécessaires au diagnostic d'un système mécatronique embarqué à partir des données d'entrée suivantes :

- les documents de conception du système (schémas électriques, netlists, spécifications fonctionnelles...)
- l'ensemble des modes de défaut anticipés sur les composants du système ainsi que leur probabilité d'occurrence
- l'ensemble des points d'observation (points de potentiel électrique, services de diagnostic des calculateurs...)
- l'ensemble des configurations de test intéressantes pour diagnostiquer le système

Nous avons divisé cet atelier en quatre applications afin de contrôler les différentes étapes du processus de prédiction des tests qui sont la modélisation (environnement « Dymola »), les simulations itératives des tests (application « DDE-Diagolica »), la construction des modalités par l'algorithme DTW (application « modalite ») et la saisie des coûts de test et des probabilités d'occurrence des défauts (application « crosstable »).

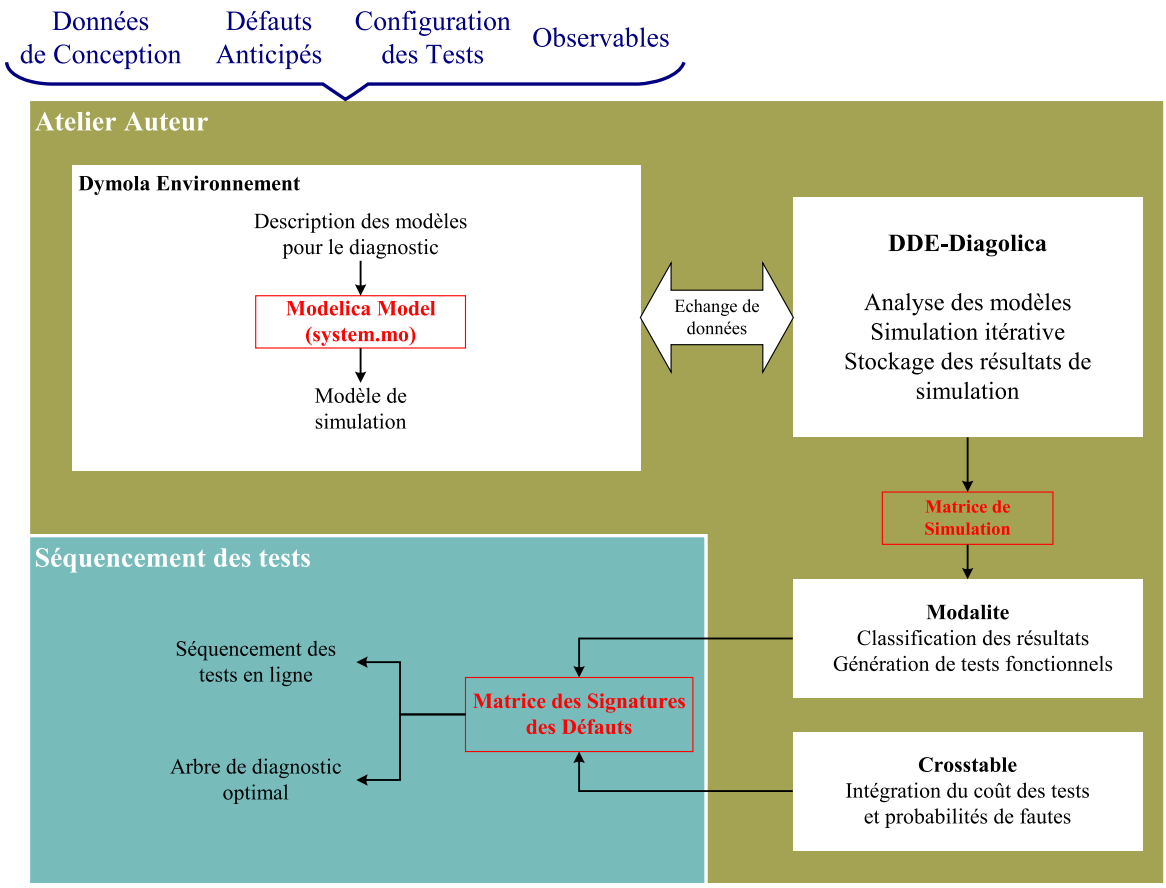


Figure 7.1 — Architecture du prototype logiciel MODE-MBR

7.1.1.1 Modélisation avec l'environnement Dymola

Présentation de Dymola Dymola[1] [Elmqvist *et al.*, 1999] est un environnement commercial de modélisation et de simulation de systèmes multi-physiques hybrides basé sur le langage Modelica [Fritzson, 2003].

Ce langage, qui est un langage orienté objet, est particulièrement adapté à la modélisation structurelle et comportementale de systèmes mécatroniques embarqués. Le standard Modelica est en fait né en 2000 d'un consortium européen, appelé « Modelica Association »[2], visant à rechercher de nouvelles solutions permettant de modéliser et simuler à partir d'un langage de haut niveau des systèmes qui contiennent des composants logiciels et qui mettent en jeu plusieurs domaines de la physique.

Concernant l'environnement de simulation, nous avions initialement choisi l'environnement libre « Open Modelica » [Fritzson *et al.*, 2002] mais la version actuelle ne permet pas encore la gestion des systèmes à événements discrets. De plus, cet outil ne dispose pas d'interface graphique permettant une description modulaire des systèmes. Nous nous sommes alors tournés vers l'environnement Dymola, qui est actuellement le plus avancé en ce qui concerne la prise en compte des librairies standards du langage Modelica.

[1] Dymola est développé et vendu par la société Dynasim, filiale du groupe Dassault Systèmes
[2] www.modelica.org

Notion de composant et de connexion dans Modelica Comme le langage Modelica est un langage orienté objet, il utilise les concepts de *classe* (`class`) et d'*héritage de classe*. Toutefois, ce langage dispose d'une particularité qui est que les classes peuvent être typées afin d'être restreintes à la description d'un type d'objet particulier (composant, connecteur, type de variable...) par l'utilisation d'autres instructions que `class`. Nous ne présentons dans ce paragraphe que les types de classes que nous utilisons :

- L'instruction `model` permet de définir des objets de type composant. Ainsi, les composants matériels et logiciels que nous voulons modéliser correspondent à des classes de ce type.
- L'instruction `connector` permet de définir des objets de type port (physique, signal). Pour définir un signal d'entrée (de sortie), il faut utiliser l'instruction input (output) en complément. Par exemple, les signaux d'entrée et de sortie booléens se définissent de la manière suivante en langage Modelica[3] :

  ```
  connector BooleanInput = input BooleanSignal;
  connector BooleanOutput = output BooleanSignal;
  ```

  La définition d'un port physique consiste à indiquer la variable d'effort (qui est transmise par le port) et la variable de flux (qui traverse le port). Par exemple, la définition d'un port électrique se définit de la manière suivante :

  ```
  connector Pin "Pin of an electrical component"
    SI.Voltage v "Potential at the pin";
    flow SI.Current i "Current flowing into the pin";
  end Pin;
  ```
- L'instruction `type` crée une classe particulière qui permet de définir de nouveaux types de variables. Ainsi, le langage Modelica contient un certain nombre de variables typées en fonction de la nature des quantité physiques auxquelles elles se rapportent (intensité électrique, potentiel électrique...) et de leur unité (volts, ampères...).

Pour décrire les connexions entre composants, afin de construire le modèle structurel du système, il faut utiliser la fonction `connect()` qui permet de relier deux ports de même type entre eux. Par exemple, pour le système essuie-vitre arière (figure 4.3), la connexion entre le port `p_1` du module `ECU_hab` et le port `p_1` du composant `Con_1` s'écrit `connect(ECU_hab.p_1,Con_1.p_1)`.

Bibliothèque de composants élémentaires de diagnostic Pour modéliser les systèmes mécatroniques, nous avons créé une bibliothèque de composants de diagnostic génériques qui peut être étendue en fonction des besoins. Pour chaque composant de cette bibliothèque, nous définissons une variable de mode de diagnostic, appelée *diagmode*, qui est une variable de type énuméré. Cette variable est initialisée par un paramètre modifiable appelé *diagmodeini*.

Par exemple, pour modéliser une résistance électrique (figure 7.2) ayant un mode de bon fonctionnement BF et deux modes de défaut de paramètre CO et CC correspondant respectivements aux défauts de circuit-ouvert et de court-circuit, nous définissons les éléments

[3] L'instruction `SI` se rapporte à la bibliothèque Modelica définissant toutes les variables du système international

suivants :

- comme la résistance est un dipôle électrique, deux ports sont déclarés comme des instances de la classe `Pin` qui de type `connector` comme définie précédemment.
- l'état de la résistance est caractérisée par une variable de type courant i, une variable de tension v et une variable de mode $diagmode$ qui est de type énuméré et dont l'ensemble des valeurs possibles est $\{BF, CO, CC\}$. Comme le type ennuméré n'est pas encore pris en compte dans la version de l'environnement Dymola que nous avons il est simulé avec un package qui utilise une représentation sous forme d'entiers.
- des équations de comportement dont l'activation peut être conditionnée par la valeur de $diagmode$.

Nous obtenons alors la description en langage Modelica suivante :

```
model ResistanceDiag "Resistance avec mode de défauts"
   Modelica.Electrical.Analog.Interfaces.Pin p;
   Modelica.Electrical.Analog.Interfaces.Pin n;
   Modelica.SIunits.Voltage v;
   Modelica.SIunits.Current i;
   parameter Modelica.SIunits.Resistance R=1;
   parameter Modelica.SIunits.Resistance R_CC=0;
   parameter Modelica.SIunits.Conductance G_CO=0;
// Définion du type énuméré Faute.Mode
  package Faute "Enumeration emulation"
      extends Modelica.Icons.Enumeration;
      constant Integer BF = 0 "Nominal Behaviour";
      constant Integer OC = 1 "Open Circuit";
      constant Integer CC = 2 "Short Circuit";
      type Mode
                extends Modelica.Icons.TypeInteger;
      end Mode;
  end Faute;
// Déclaration de la variable de mode et du paramètre de mode initial
  Faute.Mode diagmode;
  parameter Faute.Mode diagmodeini=Faute.BF;
initial equation
    diagmode = diagmodeini;
equation
    // Equations de dipôle vraies dans tous les modes
    v = p.v - n.v;
    i = p.i;
    p.i + n.i = 0;
    // diagmode à l'instant t = diagmode à l'instant t-1
    diagmode = pre(diagmode);
    // Définition des équations comportementales dans chacun des modes
    if (diagmode == Faute.BF) then
        v = R * i;
    elseif (diagmode == Faute.SC) then
        v = R_CC * i;
    else
        v * G_CO = i;
    end if;
end ResistanceDiag;
```

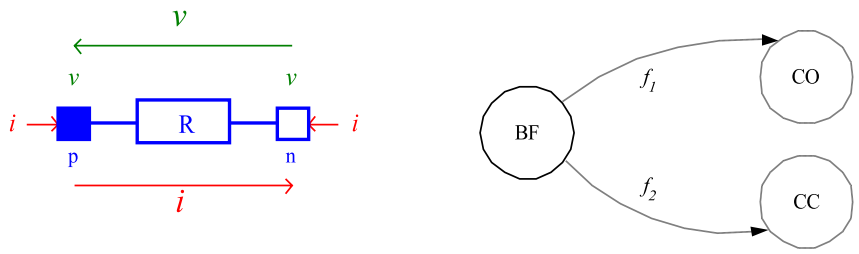


Figure 7.2 — Modèle de la résistance électrique

En langage Modelica, la description des défauts de connexion est reportée sur les connecteurs électriques car le langage ne permet pas de conditionner la fonction `connect()` par l'instruction `if`.

Configurations de test et observables La définition de l'ensemble $CONF$ des configurations de test et de l'ensemble OBS des points d'observation est réalisée directement dans le modèle du système. Cela revient à décrire un objet particulier pour chacun de ces ensembles et de le relier au système étudié; c'est-à-dire de relier les variables définies dans l'objet $CONF$ aux variables de configuration du système et les variables définies dans l'objet OBS aux variables observables du système.

Notion de modèle de simulation Une fois que le modèle d'un système est saisi dans l'environnement Dymola, celui-ci est compilé sous la forme d'un fichier exécutable que nous appelons modèle de simulation. Simuler le système consiste alors à exécuter le modèle de simulation en fonction des paramètres de simulation et d'initialisation contenus dans un fichier texte pris en entrée. Ce fichier texte peut être modifié avec de nouvelles conditions initiales permettant ainsi de réaliser des simulations du système selon différentes conditions sans avoir à recompiler le modèle. Les résultats de chaque simulation peuvent être récupérés dans un fichier texte de sortie.

Interface d'échange de données de Dymola L'environnement Dymola dispose d'une interface d'échange de données lui permettant d'être piloté par des applications extérieures suivant un protocole de communication de type client-serveur. Le protocole qui est utilisé par Dymola est le protocole Dynamic Data Exchange (DDE) de Microsoft Windows. Ainsi, il est possible de développer des applications qui envoient des requêtes de simulation à l'environnement Dymola et qui en analysent la sortie.

7.1.1.2 Simulations itératives avec DDE-Diagolica

« DDE-Diagolica » est une application que nous avons développée afin de réaliser des simulations itératives des systèmes modélisés dans Dymola. Cette application communique

avec l'environnement Dymola, via l'interface DDE présentée, précédemment pour réaliser de manière automatique et séquentielle les opérations suivantes :

1. La première étape consiste à analyser le modèle afin de construire l'ensemble des modes de défauts anticipés $\mathcal{F}$, l'ensemble des configurations de test $CONF$ et l'ensemble des observable OBS. La construction de l'ensemble $\mathcal{F}$ nécessite notamment d'explorer la hiérarchie structurelle du système afin de rechercher l'ensemble des composants pour lesquels une variable de mode de diagnostic ($diagmode$) a été définie.
2. La seconde étape consiste à simuler le système de manière itérative pour chaque configuration et chaque mode de défaut anticipé.
3. Les résultats de chaque simulation sont ensuite stockés dans des fichiers texte.

Une copie-écran de l'application est donnée par la figure 7.7.

7.1.1.3 Application « Modalite »

L'application « Modalite » prend en entrée les résultats des simulations itératives du système et construit automatiquement les modalités (courbes de référence) de test par classification des signaux avec l'algorithme DTW (section 5.4). L'application donne également la possibilité d'intervenir manuellement en fin de classification afin de vérifier le résultat obtenu et de procéder éventuellement à des fusions de modalités jugées différentes par l'algorithme mais qui pourraient être jugées équivalentes par l'expert.

Actuellement, l'étape de génération des symptômes fonctionnels sous forme de propositions sémantiques n'est pas implémentée (cf. section 5.4.2). Les modalités de ces tests restent alors sous la forme de séries temporelles à valeurs entières correspondant aux différents états qualitatifs possibles de la variable observée.

7.1.1.4 Application « Crosstable »

L'application « Crosstable » permet de visualiser la matrice de signatures des défauts puis de saisir les coûts dynamiques de test et les probabilités d'occurrence a priori des défauts. C'est donc dans cette application que nous définissons les instrumentations ainsi que leur coût d'installation. La copie-écran de l'application est donnée par la figure 7.8. Les lignes de la matrice correspondent aux modes de défaut anticipés et les colonnes aux tests prédits. La sélection d'une cellule de cette matrice permet d'observer la modalité correspondante.

7.1.2 Outil de séquencement des tests

L'outil de séquencement des tests est l'application qui est destinée à être embarquée dans l'outil de diagnostic du garagiste. Il implémente les deux stratégies de test présentées dans le chapitre 6. Une copie-écran de l'application est donnée par la figure 7.9. Dans le cadre d'une statégie du prochain meilleur test, tous les tests pouvant être réalisés sont affichés par ordre décroissant du gain d'information par unité de coût. Le premier test proposé est donc

$\mathcal{R}$	Réparation	$\mathcal{F}$	type	Description du défaut
r_0	-	f_0	-	bon fonctionnement
r_1	{ECU_hab.F1}	f_1	HW	ECU_hab.F1 en mode « circuit ouvert »
r_2	{ECU_hab}	f_2	HW	ECU_hab.Rprotec en mode « circuit ouvert »
		f_3	HW	ECU_hab.K1 en mode « reste fermé »
		f_4	HW	ECU_hab.K1 en mode « reste ouvert »
		f_5	SW	ECU_hab.Soft_ECU_hab.cmd_evar collé à 1
		f_6	SW	ECU_hab.Soft_ECU_hab.cmd_evar collé à 0
		f_7	SW	ECU_hab.Soft_ECU_hab.arret_fixe_ar collé à 1
		f_8	SW	ECU_hab.Soft_ECU_hab.arret_fixe_ar collé à 0
r_3	{Fil_1}	f_9	HW	Fil_1 en mode « circuit ouvert »
r_4	{Fil_2}	f_{10}	HW	Fil_2 en mode « circuit ouvert »
r_5	{Fil_3}	f_{11}	HW	Fil_3 en mode « circuit ouvert »
r_6	{ECU_hdc}	f_{12}	SW	Sof_ECU_hdc.etat_cear collé à dmd_ess_ar
		f_{13}	SW	Sof_ECU_hdc.etat_cear collé à dmd_repos_ar
r_7	{EVAR}	f_{14}	HW	EVAR.Wire1 en mode « circuit ouvert »
		f_{15}	HW	EVAR.Wire2 en mode « circuit ouvert »
		f_{16}	HW	ECU_hab.K_af en mode « reste fermé »
		f_{17}	HW	ECU_hab.K_af en mode « reste ouvert »
r_8	{EVAR, ECU_hab.F1}	f_{18}	HW	court-circuit entre ECU_hab.p_1 et ECU_hab.p_2
		f_{19}	HW	court-circuit entre ECU_hab.p_1 et ECU_hab.p_3

Tableau 7.1 — Définition des modes de défauts du système essuie-vitre arrière

celui qui est le plus intéressant ; s'il ne convient pas au garagiste, il est libre d'en choisir un autre. Dans le cas d'une stratégie par arbre de diagnostic optimal, un seul test est proposé à chaque étape.

7.2 Prédiction des tests pour le système essuie-vitre arrière

Dans cette section, nous appliquons le processus de prédiction au système essuie-vitre arrière présenté dans le chapitre 4. L'objectif est notamment de définir les différents ensembles du problème de séquencement des tests relatifs à ce système.

7.2.1 Modes de défaut anticipés et ensemble des réparations

Dans le tableau 7.1, nous définissons l'ensemble $\mathcal{F}$ des modes de défaut anticipés pour le système essuie-vitre arrière ainsi que l'ensemble $\mathcal{R}$ des réparations associées à ces défauts (en terme de composant à remplacer). $\mathcal{F}$ contient alors 20 éléments $f_i, i \in \{0, ..., 19\}$ et l'ensemble $\mathcal{R}$ contient 9 éléments $r_k, k \in \{0, ..., 8\}$. La colonne « type » du tableau indique si le mode de défaut porte sur un composant logiciel (SW) ou matériel (HW).

7.2.2 Définition des tests réalisables et de leur coût

Configurations de test Pour ce système, nous définissons un ensemble $CONF$ de $n_{CONF} = 3$ configurations de test $CONF_i, i \in \{1, ..., 3\}$ définies de la manière suivante :

- $CONF_1$: configuration correspondant au cas où aucune entrée du système n'est activée. Cette configuration est considérée comme initiale, c'est-à-dire comme étant la configuration courante à l'initialisation du séquencement des tests.
- $CONF_2$: configuration correspondant aux préconditions nécessaires à la réalisation de la fonction 1 du système, intitulée "Essuyer la lunette arrière lors d'une mise sur position 2 du commutateur" (cf. section 4.5).
- $CONF_3$: configuration correspondant aux préconditions nécessaires à la réalisation d'un test actionneur (fonction 3 du système, cf. section 5.2.2)

La matrice $\mathbf{\Phi}$ des coûts de transition entre configurations est de dimension 3×3 et est définie de la manière suivante :

$$\mathbf{\Phi} = \begin{bmatrix} 0 & 0,5 & 1 \\ 0,5 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Instrumentations Nous définissons l'ensemble $INST$ de $N_{inst} = 3$ instrumentations définies de la manière suivante :

- $INST_1$ correspond à l'installation d'un dérivateur de points de potentiel au niveau du connecteur `Con_2`. Son coût d'installation est $c_1^{inst} = 10$.
- $INST_2$ correspond à l'installation d'un dérivateur de points de potentiel au niveau du connecteur `Con_3`. Son coût d'installation est $c_2^{inst} = 10$.
- $INST_3$ correspond à l'outil de diagnostic. Son coût d'installation est $c_3^{inst} = 4$.

.

Nous pouvons noter que les instrumentations $INST_1$ et $INST_2$ ont un coût d'installation qui est maximum car elles nécessitent de réaliser un certain nombre d'opérations de démontage avant d'accéder aux connecteurs au niveau desquels elles doivent être installées.

Observables Nous définissons un ensemble OBS de $N_{obs} = 16$ variables que nous supposons observables dans les trois configurations définies précédemment. L'ensemble total $\mathcal{S}$ des tests disponibles contient alors $N_s = N_{obs} \times N_{conf} = 48$ éléments. Dans le tableau 7.1, nous définissons chaque variable observable pour le système essuie-vitre en précisant le niveau de connaissance dans lequel elles doivent être exprimées (cf section 5.3.1) et les instrumentations à utiliser pour les mesurer. Par ailleurs, les coûts intrinsèques des N_s tests réalisables sur ce système sont considérés comme unitaires.

7.2.3 Simulations

L'étape de prédiction des tests, pour ce système, nécessite de réaliser $N_{sim} = N_F * N_{conf} = 60$ simulations. Chaque simulation est réalisée sur une durée de 20 secondes avec

OBS_j	LEV_j	Instrum.	Définition
$v1$	'behavior'	$INST_1$	potentiel au port `Fil_1.n`
$v2$	'behavior'	$INST_1$	potentiel au port `Fil_2.n`
$v3$	'behavior'	$INST_1$	potentiel au port `Fil_3.n`
$v4$	'behavior'	$INST_2$	potentiel au port `Fil_1.p`
$v5$	'behavior'	$INST_2$	potentiel au port `Fil_2.p`
$U1$	'behavior'	$INST_1$	tension entre `Fil_1.n` et `Fil_2.n`
$U2$	'behavior'	$INST_1$	tension entre `Fil_1.n` et `Fil_3.n`
$U3$	'behavior'	$INST_1$	tension entre `Fil_2.n` et `Fil_3.n`
$i1$	'behavior'	$INST_1$	courant circulant dans `Fil_1`
$i2$	'behavior'	$INST_1$	courant circulant dans `Fil_2`
$i3$	'behavior'	$INST_1$	courant circulant dans `Fil_3`
$etat\_cear$	'behavior'	$INST_3$	signal `Soft_ECU_hdc.etat_cear`
$arret\_fixe\_ar$	'behavior'	$INST_3$	signal `Soft_ECU_hab.arret_fixe_ar`
$etat\_mar$	'behavior'	$INST_3$	signal `Soft_ECU_hab.etat_mar`
cmd_evar	'behavior'	$INST_3$	signal `Soft_ECU_hab.cmd_evar`
$MvtBalais$	'teleological'	-	vitesse de mouvement des balais

Tableau 7.2 — Définition des modes de défauts du système essuie-vitre arrière

un pas de simulation de 0.01 seconde (chaque série temporelle obtenue contient 2001 points). Par ailleurs, nous supposons que les balais sont arrêtés en position repos au début de chaque simulation (nous verrons dans les perspectives de ce manuscrit que cette hypothèse n'est pas toujours vraie dans le cas réel).

Pour un système de cette taille, la réalisation des n_{sim} simulations nécessite environ 4 minutes de calcul machine (pour un PC équipé d'un micro-processeur Pentium D 3 GHz et de 1Go de mémoire vive).

7.2.4 Génération des modalités

Les résultats des simulations sont ensuite traités par l'application « Modalite » afin de construire les modalités des tests. Cette étape, comprenant la classification automatique des signaux et les opérations manuelles de fusion de modalités pour affiner la classification, nécessite de 10 à 15 minutes de travail en fonction du nombre de tests à traiter.

Nous donnons alors en figure 7.3 le résultat de cette classification concernant les trois tests relatifs à l'observation du mouvement balais (correspondant aux trois configurations simulées). Comme l'abstraction sémantique de ces modalités n'est pas encore implémentée, nous les gardons sous la forme de courbes de référence les interprétant de la manière suivante :

- Les modalités $M11$, $M21$ et $M31$ correspondent au symptôme fonctionnel "balais_arretes". Ce symptôme représente un bon fonctionnement en configuration $CONF_1$ et à une défaillance du système pour les configurations $CONF_2$ et $CONF_3$.

- Les modalités $M12$, $M22$ et $M32$ correspondent au symptôme fonctionnel "balayage_continu" qui est une abstraction de l'effet :

  `CYCLE` balayage_aller `SEQ` balayage_retour `NEW_CYCLE`
- Les modalités $M13$ et $M23$ correspondent au symptôme fonctionnel "balayage_intermittent" qui est une abstraction de l'effet :

  `CYCLE` balayage_aller `SEQ` balayage_retour `SEQ` (balais_arretes) `NEW_CYCLE` [`AFTER` 11.9s `BEFORE` 12.1s]
- La modalité $M33$ correspond à l'effet devant être observé en fonctionnement nominal pour un test actionneur. Le symptôme fonctionnel correspondant est alors :

  balayage_aller `SEQ` balayage_retour `SEQ` balayage_aller `SEQ` balayage_retour `SEQ` balais_arretes
- La modalité $M24$ correspond au symptôme fonctionnel :

  balayage_continu [`AFTER` 6.9s `BEFORE` 7.1s] `SEQ` balais_arretes
- La modalité $M34$ correspond au symptôme fonctionnel :

  balayage_continu [`AFTER` 2.4s `BEFORE` 2.6s] `SEQ` balais_arretes
- La modalité $M35$ correspond au symptôme fonctionnel :

  balayage_aller `SEQ` balayage_retour `SEQ` balayage_aller `SEQ` balayage_retour `SEQ` balais_arretes [`AFTER` 11.9s `BEFORE` 12.1s] balayage_aller `SEQ` balayage_retour `SEQ` balais_arretes

7.3 Séquencement des tests

Dans cette section, nous présentons et discutons les résultats obtenus par application des deux stratégies de séquencement des tests sur le système essuie-vitre arrière présenté dans ce manuscrit.

7.3.1 Analyse discriminante

Comme indiqué précédemment, une analyse préliminaire est réalisée afin de déterminer les sous-ensembles de défauts de $\mathcal{F}$ qui ne sont pas discriminables. Pour ce système, il en existe deux qui sont :

- $\{f_3, f_5\}$: ces deux défauts sont associés à la même réparation r_2 (cf. tableau 7.1) correspondant au remplacement du calculateur habitacle `ECU_hab`.
- $\{f_1, f_4, f_6, f_{18}, f_{19}\}$: les composants remplaçables `ECU_hab`, `EVAR` et le fusible `F1` sont concernés par ce sous-ensemble. En cas d'occurrence de l'un de ces défauts, il est donc nécessaire de tester ces composants individuellement, par un test de résistance par exemple, afin d'identifier la réparation à effectuer.

7.3.2 Application de la stratégie du prochain meilleur test

Afin d'analyser les résultats obtenus par l'application de la stratégie du prochain meilleur test sur le système essuie-vitre arrière, nous avons généré un arbre représentant

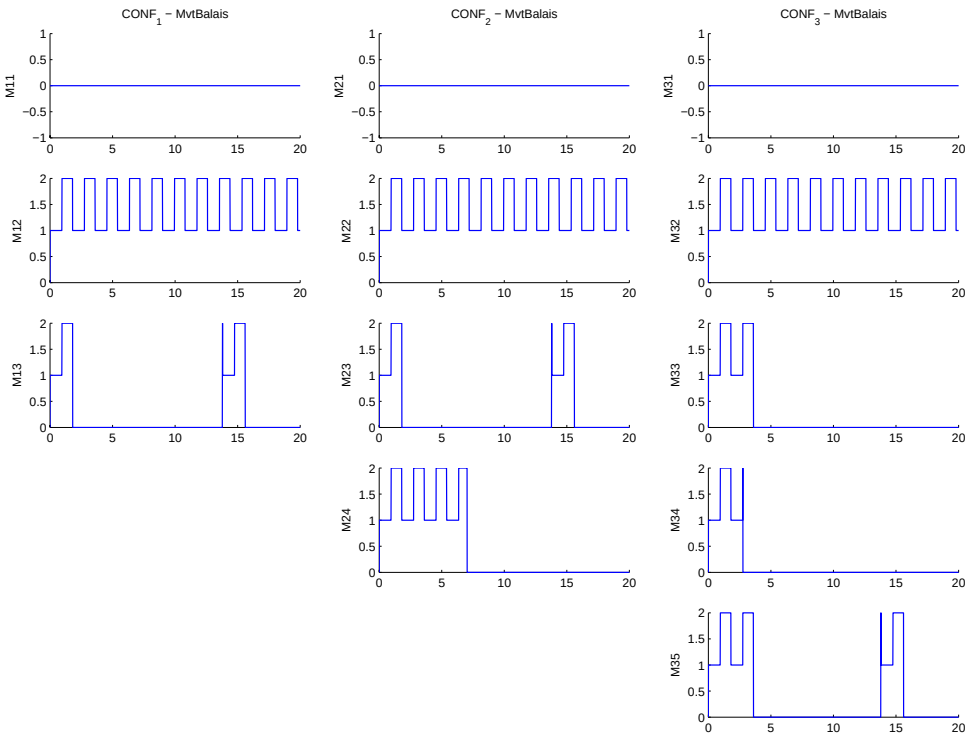


Figure 7.3 — Symptômes fonctionnels correspondant à l'observation du mouvement des balais selon les trois configurations de test

toutes les séquences possibles permettant de discriminer l'ensemble des défauts sous l'hypothèse que, à chaque étape, le test le plus intéressant, au sens du gain d'information par unité de coût est celui qui est effectivement réalisé. Nous donnons une représentation de cet arbre en figure 7.4 sous une forme compacte où les feuilles correspondent aux défauts et les nœuds représentent des tests. Chaque test est représenté par sa configuration, sa variable observable et sont coût dynamique. Le coût de cet arbre, qui n'est pas optimal, est $J = 16,85$. Nous pouvons faire les remarques suivantes :

- de manière générale, les premiers tests qui interviennent dans une séquence donnée correspondent à ceux qui sont les moins coûteux. Les mesures électriques sont toujours proposées à la fin, lorsqu'il n'y a plus d'autres choix. Nous montrons ainsi que la prise en compte du coût d'installation d'une instrumentation influence significativement la séquence des tests.
- certains défauts peuvent être localisés directement par des tests fonctionnels.

7.3.3 Génération de l'arbre de diagnostic optimal

L'arbre optimal, généré par l'algorithme AO*, est donné par la figure 7.5. Son coût est $J^* = 14.35$, c'est-à-dire environ 18% de moins que l'arbre obtenu précédemment (figure 7.4). Nous pouvons faire les remarques suivantes :

heuristique	Nb nœuds OU explorés
FEH_1	606906
FEH_{Dyn1}	213311
FEH_{Dyn2}	218650

Tableau 7.3 — Evaluation des performances des heuristiques pour le système essuie-vitre simplifié

- l'arbre optimal est moins profond que l'arbre précédent
- les sous-ensembles de défauts $\{f_{14}, f_{15}\}$ et $\{f_7, f_8\}$ correspondent à deux feuilles de l'arbre car chacun de ces sous-ensembles correspond à une réparation.
- la réalisation du test électrique représenté par le nœud $\{CONF_3; v5; [11]\}$ est proposée avant le test $\{CONF_3; arret\_fixe\_ar; [5]\}$ qui est moins cher localement car il correspond à une mesure réalisée avec l'outil de diagnostic. Cela vient du fait que l'algorithme AO* a calculé qu'il était plus optimal au sens global de faire une mesure électrique avant.
- une séquence donnée de l'arbre optimal n'est pas nécessairement optimale car l'optimisation se fait sur l'arbre global. Par exemple, le coût de la séquence des tests permettant d'isoler les défauts f_{16} et f_7 est supérieure avec l'arbre optimal (somme des coûts des test de la séquence égal à $27,5$) qu'avec la stratégie du prochain meilleur test (somme des coûts des test de la séquence égal à 28).

Performance des heuristiques Nous testons également les deux heuristiques FEH_{Dyn1} et FEH_{Dyn2}, proposées dans le chapitre précédent (section 6.4.4), prenant en compte le coût dynamique des tests dans l'exécution de l'algorithme AO*. Pour évaluer leur performance et la comparer avec l'heuristique de FEH_1 (qui ne prend en compte que les coûts intrinsèques) nous utilisons comme critère le nombre total de nœuds OU explorés pour parvenir à un arbre de diagnostic optimal du système essuie-vitre. Les résultats sont donnés dans le tableau 7.3. Nous pouvons ainsi remarquer que les heuristiques FEH_{Dyn1} et FEH_{Dyn2} permettent de gagner significativement en performances. Pour cet exemple, FEH_{Dyn1} est meilleure que FEH_{Dyn2}.

7.4 Discussion et essais sur des véhicules réels

Dans ce chapitre, nous avons présenté le prototype logiciel mettant en œuvre l'ensemble de l'approche de diagnostic exposée dans ce manuscrit. Les résultats obtenus montrent que même si la stratégie du prochain meilleur test n'est pas optimale au sens global, les séquences de tests obtenues sont réalistes et pertinentes pour le diagnostic en garage. De plus, comme la stratégie est interactive, elle donne plus d'initiative au garagiste et a donc des chances d'être mieux acceptée dans son environnement de travail.

Cependant, nous pouvons noter que l'introduction des coûts d'installation des instrumentations ajoute un effet d'horizon sur les séquences de test. En effet, comme à chaque étape, le critère du gain d'information par unité de coût n'explore que l'étape suivante, il a

tendance à persister dans l'instrumentation courante et d'y épuiser les tests restants parce qu'ils sont moins chers avant de proposer d'installer une nouvelle instrumentation qui aurait été finalement nécessaire. Ceci est significatif si nous comparons la branche gauche de l'arbre optimal (figure 7.5) avec celle de l'arbre obtenu avec une stratégie locale (figure 7.4). L'arbre optimal propose dès la deuxième étape la mesure du potentiel $v4$ alors que ce même test n'est proposé qu'à la quatrième étape par la stratégie locale. Toutefois, le fait que la stratégie interactive ne propose de réaliser des mesures électriques qu'en dernier recours reste intéressant, de plus le garagiste peu très bien décider de passer directement à des tests électriques s'il le souhaite car l'outil lui en laisse la possibilité.

Essais sur véhicules réels

Comme nous l'avons précisé dans l'introduction de ce manuscrit, l'application a été validée sur des véhicules réels de deux constructeurs différents : PSA et Fiat.

Système essuie-vitre du constructeur PSA Le modèle structurel du système essuie-vitre des véhicules conçus par le constructeur PSA est donné en annexe (figure B.1). Les véhicules Peugeot 1007 et Citroën C4 sont notamment ceux sur lesquels nous avons pu faire des essais. Le modèle réel est en fait très proche de celui présenté auparavant dans ce manuscrit car ce dernier en était fortement inspiré. Pour des soucis de clarté dans le chapitre 4, nous avions négligé deux fils électriques et les comportemement suivants :

- lorsque le volet de coffre est ouvert, l'essuyage arrière est désactivé.
- lorsque la charge de la batterie est insuffisante, le système essuie-vitre est inhibé.
- le processus d'essuyage arrière est activé par le processus de lavage.

Nous avons eu ensuite l'occasion, à plusieurs reprises, de démontrer nos résultats à des équipes de maintenance du constructeur PSA, à des équipementiers et à des garagistes généralistes. La stratégie de test a notamment été appréciée, surtout par le fait de tenir compte d'observations fonctionnelles dans le processus de diagnostic.

Nous avons également testé l'algorithme AO* avec les trois heuristiques FEH_1, FEH_{Dyn1} et FEH_{Dyn2}. Pour ce véhicule, la matrice de signatures contient 22 défauts anticipés et 42 tests prédits et nous disposions de quatre instrumentations, soit une de plus que le système essuie-vitre du manuscrit. Les nombres de nœuds développés pour générer l'arbre optimal est donné dans le tableau 7.4 pour les trois heuristiques. Nous notons cette fois que l'heuristique FEH_{Dyn2} est légèrement meilleure que l'heuristique FEH_{Dyn1} mais ce gain en performance n'est pas significatif. Cette faible différence peut provenir du fait que, pour une instrumentation donnée, nous avons beaucoup de tests à réaliser. Or, si nous calculons la longueur du code d'Huffmann au premier nœud, nous remarquons que celle-ci est faible (inférieure à 2) vis-à-vis du nombre de tests disponibles par instrumentation installée. FEH_{Dyn2} ne nous donne donc pas le gain en performance que nous pensions initialement obtenir, d'autant plus que son calcul est plus complexe que FEH_{Dyn1}.

Système essuie-vitre du véhicule Fiat Grande Punto Le modèle structurel du système essuie-vitre du véhicule Fiat Grande Punto est donné par la figure B.2. Celui-ci est assez dif-

heuristique	Nb nœuds OU explorés
FEH_1	1222746
FEH_{Dyn1}	742646
FEH_{Dyn2}	712774

Tableau 7.4 — Evaluation des performances des heuristiques pour le système essuie-vitre de la citroën C4

férent du précédent car le commutateur essuie-vitre (module `H005`) est connecté par liaison filaire au calculateur habitacle (module `M001_BCM`). Pour ce véhicule, nous avons eu l'occasion de tester notre approche pour deux versions différentes du logiciel embarqué à bord du calculateur habitacle.

La première version du logiciel ne disposait pas d'état de protection du moteur électrique dans son automate de comportement (état S_4 de la figure 4.11) alors que cet état avait été ajouté sur la deuxième version. Nous avons alors démontré, en prenant deux véhicules Grande Punto ayant chacun une version différente du logiciel, que la modification du modèle pour passer d'une version à l'autre était simple. Sachant que l'exécution de la chaîne de compilation des données de diagnostic dans l'atelier auteur ne prend que 20 à 25 minutes. Cette évolution a été faite en direct devant les ingénieurs de Fiat qui on eu ainsi l'occasion d'apprécier cette performance par rapport à une méthode générant manuellement un arbre de diagnostic.

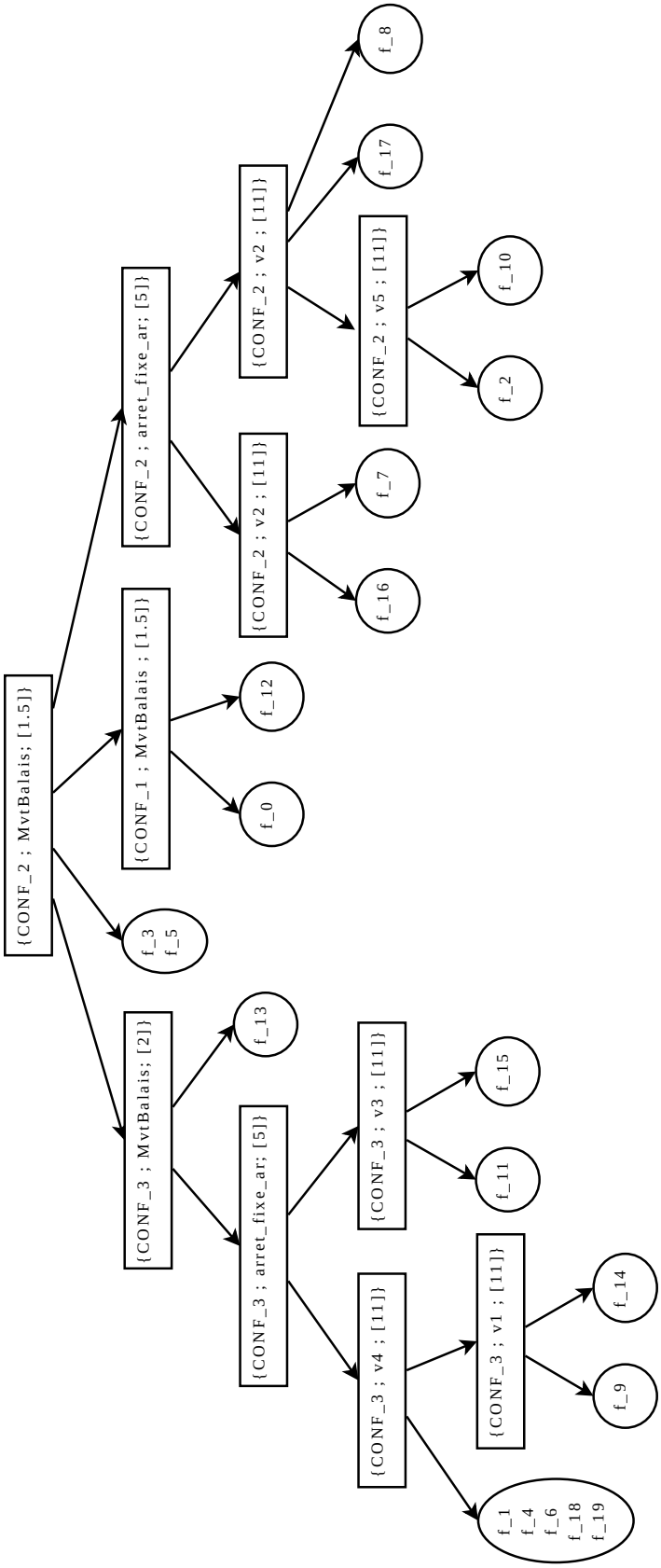


Figure 7.4 — Génération de toutes les possibilités de séquences avec le critère du maximum de gain d'information par unité de coût non optimal $J = 17,02$

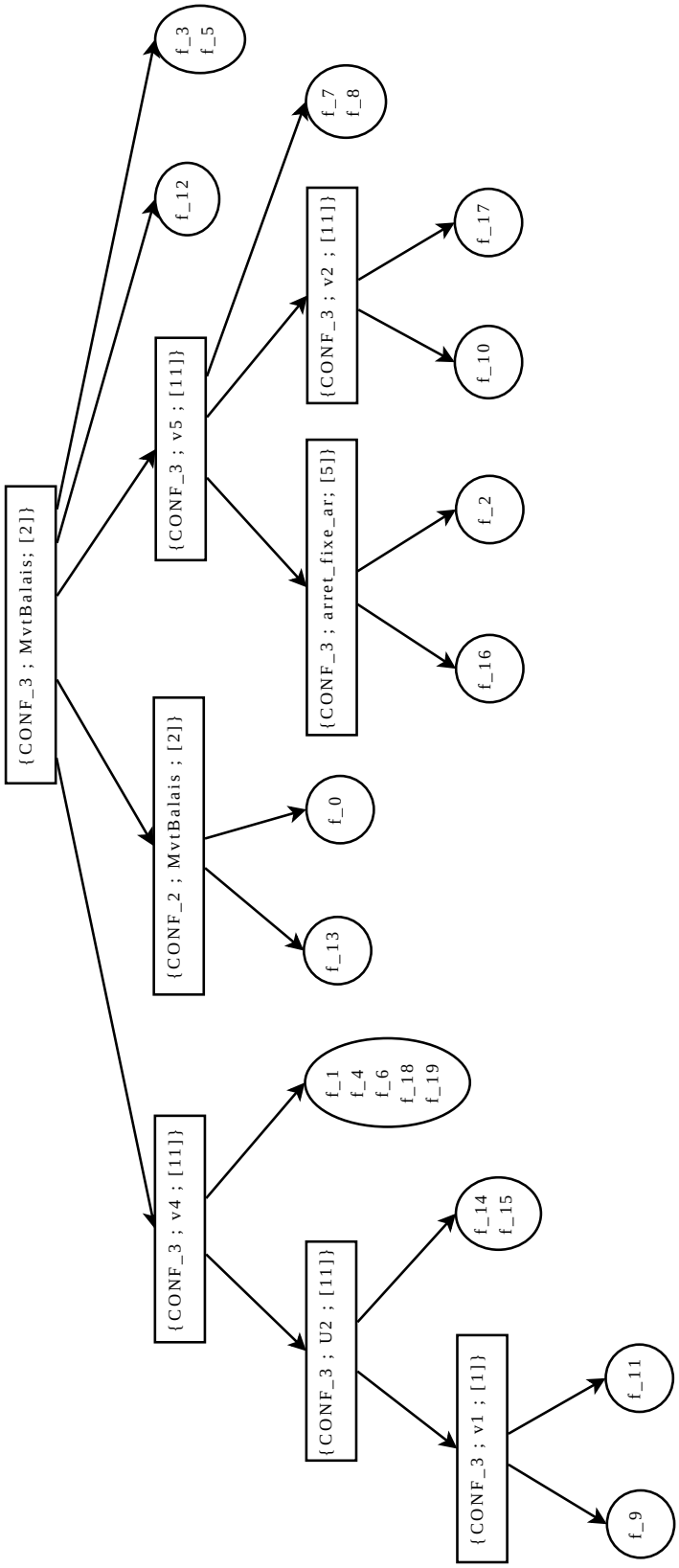


Figure 7.5 — Arbre de diagnostic optimal du système essuie-vitre arrière ($J^* = 14.35$)

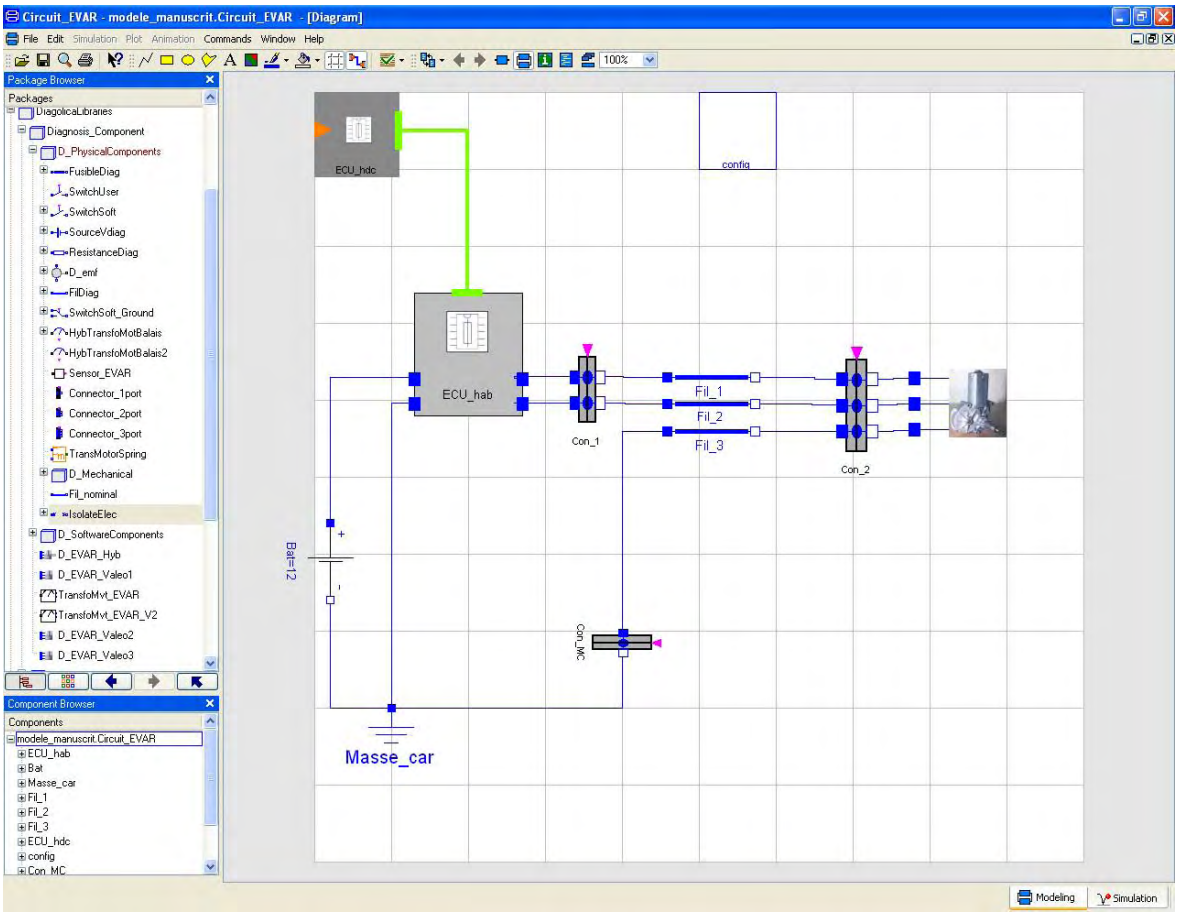


Figure 7.6 — Copie écran de l'environnement Dymola

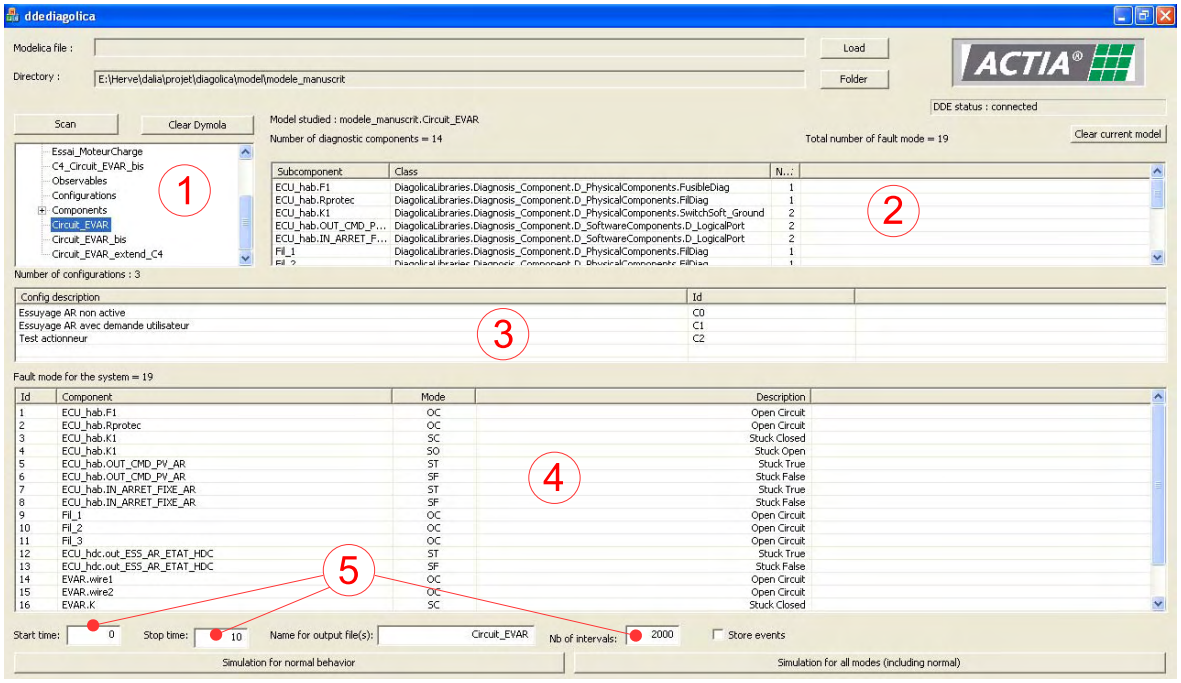


Figure 7.7 — Copie écran de l'application de « DDE-Diagolica ». Les différents champs numérotés correspondent à la légende suivante : (1) cadre de sélection du modèle à analyser puis à simuler, (2) liste des composants pour lesquels des modes de défaut sont définis, (3) ensemble $CONF$ des configurations de test, (4) ensemble $\mathcal{F}$ des défauts anticipés, (5) définition des paramètres de simulation : instants de début et de fin de simulation, nombre de points à calculer

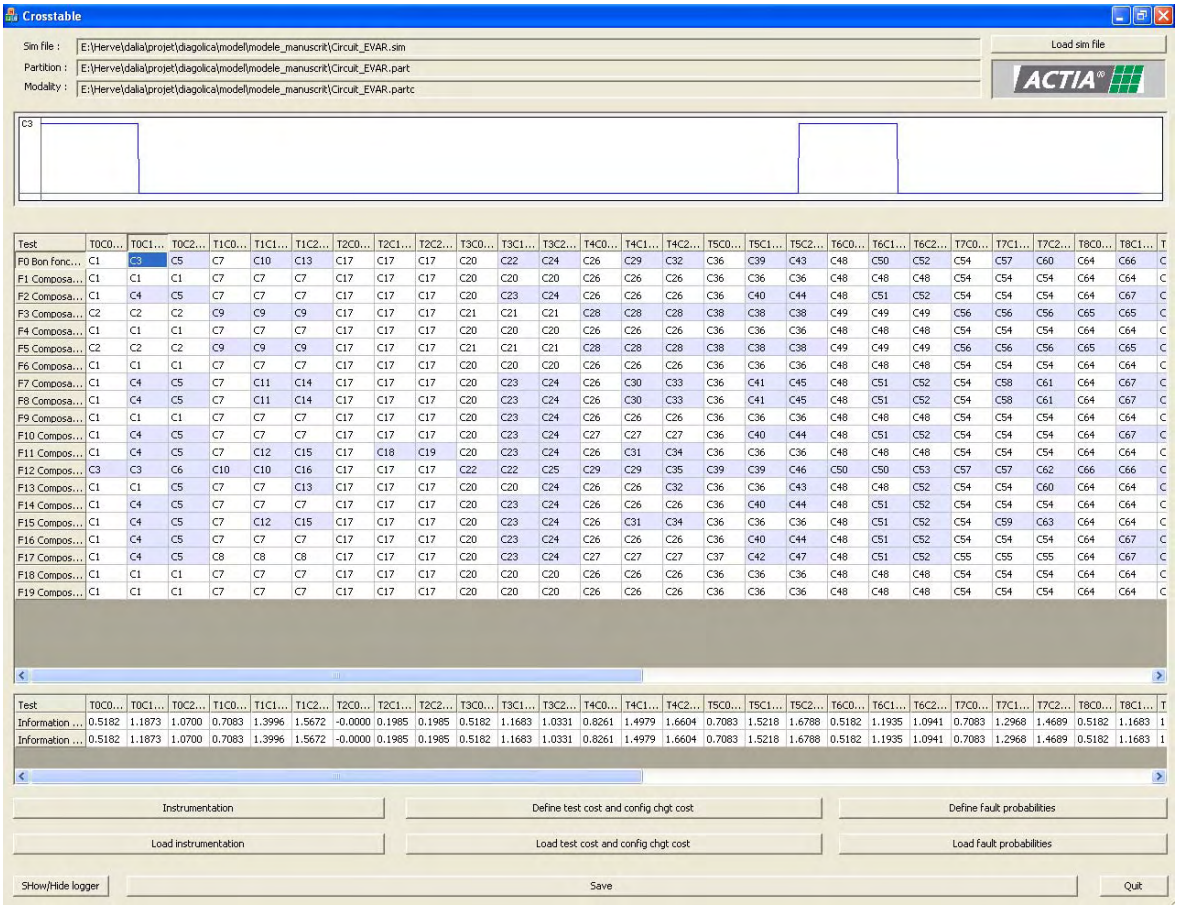


Figure 7.8 — Copie écran de l'application « crosstable »

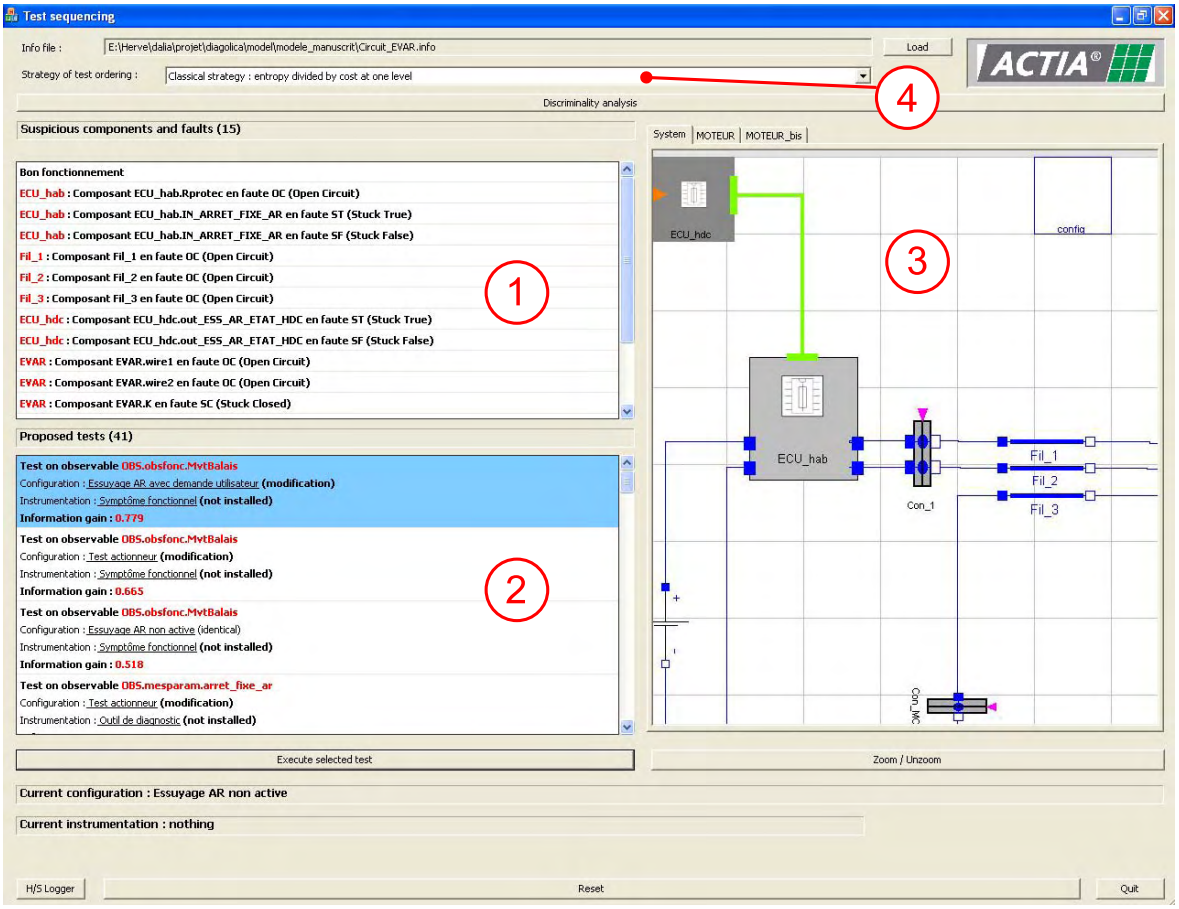


Figure 7.9 — Copie écran de l'application de séquencement dynamique. Les champs numérotés correspondent à la légende suivante : (1) ensemble des défauts restant à discriminer. Si aucun test n'a encore été effectué, ce cadre affiche l'ensemble d'ambiguité total, (2) liste des tests proposés, (3) schéma électrique du système, (4) champ de sélection de la stratégie de séquencement (prochain meilleur test ou arbre de diagnostic).

8 Conclusions et Perspectives

Cette thèse s'intéresse au problème de diagnostic hors-ligne des systèmes mécatroniques embarqués dans le domaine automobile et plus particulièrement à l'étape de localisation hors-ligne des défauts présents sur un véhicule. Notre problème de diagnostic se formule comme un problème de test dans lequel il s'agit de déterminer, à partir de la connaissance de conception du véhicule et de symptômes déjà présents, les meilleures séquences de test que le garagiste doit réaliser afin de localiser le ou les composants défaillants à remplacer. La méthode que nous avons développée est basée sur la définition d'un dictionnaire de défauts et sur des jeux de simulations numériques dynamiques permettant de prédire l'écolution de chaque test disponible sous l'occurrence de chaque défaut anticipé.

Nos travaux font suite à deux précédentes thèses réalisées à ACTIA [Faure, 2001; Olive, 2003] qui avaient notamment abouti à la méthode de diagnostic AGENDA. Cette méthode avait pour caractéristiques essentielles que les modèles utilisés étaient statiques et limités à des circuits résistifs pouvant inclure des composants à plusieurs modes de fonctionnement tels que les interrupteurs et les fusibles. Or, comme nous l'avons défini dans le chapitre 1, les systèmes mécatroniques sont des systèmes dynamiques hybrides constitués de composants physiques à plusieurs modes de fonctionnement dont l'activation est conditionnée par des composants logiciels. Ces systèmes sont difficiles à diagnostiquer car, d'une part, la connaissance disponible pour les modéliser est hétérogène et, d'autre part, les points d'observation sont peu nombreux et parfois très difficile d'accès.

Nous avons alors choisi d'utiliser une représentation multi-modèle (chapitre 4) afin de structurer et de hiérarchiser les différents types de connaissance dont nous disposons sur les systèmes mécatroniques. Le mode de représentation que nous proposons d'utiliser se présente sous la forme d'une grille que nous avons appelée plan de représentation multi-modèle. L'axe horizontal de cette grille permet de réaliser des hiérarchies par agrégations (le plus souvent appelées hiérarchies structurelles) et se divise en quatre niveaux qui sont : le niveau groupe de systèmes (prestations), le niveau système, le niveau module/sous-système et le niveau composants élémentaires. Cette hiérarchie est celle qui est la plus couramment employée par les ingénieurs de conception et est proche de l'architecture matérielle des systèmes réels.

L'axe vertical correspond à la hiérarchie fonctionnelle du système et se divise également en quatre niveaux correspondant au quatre types épistémologiques définis par L. Chittaro [Chittaro *et al.*, 1993] dans l'approche multi-modèle initiale, i.e. la connaissance structurelle,

la connaissance comportementale, la connaissance fonctionnelle et la connaissance téléologique. Cette hiérarchie permet de faire la passerelle entre les fonctions (buts, besoins) du système modélisé et le comportement de chaque composant constituant ce système. Par rapport à l'approche multi-modèle de L. Chittaro, nous avons ajouté un certain nombre de concepts, repris pour certains de la méthode SA-RT, afin de positionner les aspects logiciels dans la hiérarchie. Les principales caractéristiques des différents niveaux de la hiérarchie fonctionnelle peuvent se résumer de la manière suivante :

- La connaissance structurelle décrit la topologie du système par les composants logiciels et matériels qui le composent et les liens qui permettent à ces composants d'interagir au travers de leurs ports. Ces liens sont de nature acausale lorsqu'ils décrivent une connexion entre deux ports physiques de même nature (échange d'une variable d'effort et d'une variable de flux) ou causale lorsqu'ils décrivent une connexion entre un signal d'entrée et un signal de sortie (échange d'information au travers des ports de types signaux).
- La connaissance comportementale d'un système décrit comment les composants se comportent individuellement et interagissent en terme de quantités qui caractérisent leur état et en terme de lois qui régissent l'évolution de cet état au cours du temps. Le comportement d'un composant matériel est décrit par un automate hybride et le comportement d'un composants logiciel par un automate discret.
- La connaissance fonctionnelle réalise une interprétation de la connaissance comportementale en terme de processus fonctionnels organisés suivant un réseau de dépendance (réseau causal). Un processus, pouvant être un processus de données, un processus de contrôle, un processus de stockage ou un processus physique, est caractérisés par un état fonctionnel qui est actif ou inactif en fonction des propriétés qui sont vraies en entrée (pré-conditions) et d'effets observés en sortie. La construction de ce modèle nécessite, d'une part, d'abstraire un comportement décrit de manière quantitative en logique propositionnelle et, d'autre part, de réaliser une interprétation causale des équations physiques afin de construire les processus physiques. Par ailleurs, nous avons introduit la notion d'épisodes qualitatifs afin de représenter des aspects temporels et séquentiels sur les effets.
- La connaissance téléologique spécifie les fonctions du système en terme de buts assignés par son concepteur. Bien que le terme de fonction puisse prêter à confusion avec le modèle fonctionnel, nous l'utilisons à ce niveau de connaissance car il correspond au langage utilisé dans le milieu industriel. L'usage des termes « but » utilisé dans [Chittaro *et al.*, 1993] ou de « méta-fonction » utilisé dans [Kitamura *et al.*, 2002] est en effet moins courant. Comme pour le modèle des processus, nous associons un état fonctionnel à une fonction permettant d'exprimer si une fonction est réalisée ou non. Cet état fonctionnel est caractérisé par des pré-conditions et des effets exprimés sous forme de propositions sémantiques.

La définition de ce cadre multi-modèle nous paraît nécessaire pour structurer une tâche de raisonnement devant utiliser des observations de types différents. Ainsi, un symptôme fonctionnel (symptôme client) porte sur les effets des fonctions du système (connaissance téléologique), une mesure physique se réalise au niveau des ports des composants logiciels et matériels (modèle structurel et comportemental). Les observations qualitatives cor-

respondent quant à elles à des mesures réalisées au niveau structurel mais interprétées de manière qualitative. C'est suivant cette représentation que nous avons défini ensuite notre approche de diagnostic, celle-ci étant divisée en deux étapes qui sont la prédiction des tests (simulation) et le séquencement.

L'étape de prédiction (chapitre 5) consiste à simuler les tests sous l'occurrence de chaque défaut anticipé afin de construire la matrice de signatures des défauts du système. Par rapport à la méthode AGENDA, nous avons étendu la notion de mode de défaut aux composants logiciels et nous avons redéfini la notion de test dans un cadre multi-modèle afin d'intégrer tous les moyens de mesure disponibles en garage tels que les tests fonctionnels, les services de diagnostic des calculateurs électroniques et les mesures physiques. De même nous avons défini de nouvelles modalités qui sont les courbes de référence et les symptômes fonctionnels. Les simulations sont dynamiques et réalisées à partir du modèle structurel et comportemental. Comme les résultats de ces simulations sont des séries temporelles, nous avons proposé l'algorithme DTW pour les regrouper en classes d'équivalence afin de former les modalités des tests. Les modalités correspondant aux tests fonctionnels sont générées par abstraction qualitative puis sémantique des résultats des simulations numériques relatives à ces tests afin de générer l'ensemble des symptômes fonctionnels possibles d'un système.

Le séquencement est ensuite réalisé de manière interactive suivant la stratégie du prochain meilleur test ; le prochain meilleur test étant celui pour lequel le gain d'information par unité de coût est maximum. Le coût des tests est évalué de manière dynamique et dépend de trois paramètres qui sont le coût de réglage de la configuration sur le système, le coût d'installation de l'instrumentation de mesure et le coût intrinsèque du test une fois que le système est dans la bonne configuration et que l'instrumentation est installée. Cette stratégie a été comparée aux arbres de diagnostic optimaux générés par un algorithme AO* (méthode AGENDA) pour lequel nous avons proposé deux nouvelles heuristiques prenant en compte les coût dynamiques des tests. Les résultats obtenus sur le système essuie-vitre ont montré que même si la stratégie du prochain meilleur test n'est pas optimale au sens global elle donne une séquence de tests très pertinente pour un diagnostic en garage. D'une part, la stratégie est interactive et laisse plus d'initiative au garagiste durant la session de diagnostic en lui permettant de « sauter » des tests ou d'en refuser. D'autre part, les séquences de tests proposées sont en accord avec le modèle de diagnostic que peut se faire le garagiste qui est de réaliser d'abord les tests les moins chers avant de réaliser des tests électriques, nécessitant un certain nombre d'opérations de démontage et de remontage.

L'ensemble de cette approche de diagnostic a fait l'objet d'un prototype logiciel (chapitre 7). Ce prototype logiciel se divise en deux parties correspondant à deux métiers différents : l'atelier auteur et l'application de séquencement. L'atelier auteur permet à l'expert de diagnostic de compiler hors-ligne l'ensemble des données de diagnostic (matrice de signatures des défauts, probabilités des défauts, coûts des tests) en ayant le contrôle sur chaque étape du processus de prédiction des tests. Cet atelier est basé sur l'utilisation du langage Modelica pour simuler le système. Ce langage a l'avantage d'être standard et d'avoir une capacité de description qui correspond à la notion de comportement et de structure tels que nous les avons définis dans l'approche multi-modèle. Le choix d'utiliser un tel langage a été essentiellement motivé par le désir d'économiser le coût de développement d'un simulateur qui aurait répondu à 100% à nos besoins. En effet, nous avons vu que dans certains

cas le langage Modelica était peu pratique et nécessitait d'avoir recours à certaines astuces de description (pour la modélisation des défauts de connexion par exemple).

L'outil de séquencement permet de réaliser le séquencement interactif des tests, c'est-à-dire suivant la stratégie du prochain meilleur test. L'algorithme AO* a également été implémenté dans cette application pour la génération d'arbres de diagnostic mais ce n'est pas la stratégie que nous préconisons d'un point de vue applicatif.

Pour conclure, listons les principaux aux avantages de la méthode de diagnostic que nous avons développé :

- La quantité de données devant être embarquée sur l'outil de diagnostic en garage est réduite à la matrice de signature des défauts, des probabilités d'occurrence a priori des défauts et des coûts de réalisation des tests.
- L'utilisation des modèles permet de réduire le coût humain de génération des données de diagnostic par rapport aux arbres de diagnostic construits manuellement par les experts de diagnostic. Cette réduction de coût n'est pas immédiatement perceptible lorsqu'un système mécatronique est modélisé pour la première fois car cette tâche nécessite encore l'intervention d'un expert ayant une bonne connaissance du système. L'amélioration devient vraiment significative lorsqu'il est nécessaire de gérer plusieurs variantes d'une même prestation du véhicule car la modification des modèles est simple à réaliser et ne nécessite ensuite qu'une vingtaine de minutes (par variante) de traitement pour compiler les données de diagnostic.
- La prise en compte des symptômes clients et des services de diagnostic des calculateur comme tests montrent qu'il est possible de diagnostiquer un système en effectuant un nombre d'opérations de démontage et de remontage minimal.
- La prédiction dynamique des tests montre qu'il est essentiel, pour diagnostiquer les systèmes mécatroniques actuels, de tenir compte du fait que la présence d'un défaut peut ne pas modifier en amplitude la valeur d'une variable mesurée mais influer sur la séquence des valeurs prises par cette variable.

Enfin, il existe un certain nombre de perspectives et de limites à notre travail que nous discutons en deux parties : la première partie est consacrée aux perspectives applicatives et industrielles de la méthode et la deuxième partie porte sur les perspectives scientifiques.

Perspectives applicatives et industrielles du module MODE-MBR

Modélisation et perspectives d'application sur d'autres systèmes

Comme nous l'avons indiqué dans le chapitre 7, le prototype MODE-MBR a été présenté, avec le système essuie-vitre arrière comme exemple de démonstration, à différents acteurs du domaine automobile et plus particulièrement aux équipes de maintenance des constructeurs et à des garagistes généralistes.

L'intérêt de la méthode et les résultats obtenus ont certes été reconnus mais la disponibilités des modèles de conception peut être considérée comme un obstacle (du point de

vue d'un constructeur) à tout type d'approche de diagnostic à base de modèles[1]. En effet, les problèmes liés à la maintenance en garage ne sont pas explicitement pris en compte en phase de conception et les équipes d'experts en maintenance des constructeurs n'ont pas toujours un accès direct et automatique aux données de conception du véhicule tels que les spécifications fonctionnelles des calculateurs ou les modèles des capteurs et des actionneurs. Par ailleurs, les constructeurs ne disposent pas encore d'outils pluri-disciplinaires leur permettant de faire des analyses de conception globales de systèmes mécatroniques complets mais ils disposent plutôt d'outils divers et variés spécialisés pour une partie d'un système. Par exemple, les composants logiciels sont testés sans réelle prise en compte de l'environnement qu'ils doivent contrôler ou surveiller. Ce point est un des points qui avait été relevé par le groupe de travail inter GdR AFSEC (cf. section 1.1). Cependant, ces aspects font l'objet de beaucoup de travaux de recherche et seront amenés à évoluer d'ici quelques années. Notons que les langages de modélisation tels que Modelica et VHDL-AMS et leurs évolutions futures font partie de cette mouvance.

En ce qui concerne notre problème de diagnostic, l'étude d'un nouveau système représente effectivement un travail d'expertise de départ pour obtenir un modèle en langage Modelica car nous n'avons pas toujours de données de conception claires et actualisés. Cependant, ce coût porte surtout sur la première modélisation et faire évoluer les modèles pour une nouvelle variante ou un véhicule conçu par un autre constructeur représente un coût bien inférieur. Nous avons vérifié cet aspect pour le système essuie-vitre arrière que nous avons d'abord modélisé pour le véhicule Citroën C4 puis actualisé pour le véhicule Fiat Grande Punto qui avait une architecture matérielle et des spécifications fonctionnelles légèrement différentes. Maintenant, il pourrait être intéressant de continuer à appliquer l'approche sur d'autres systèmes afin d'en appréhender les performances de manière plus précise. Les pistes d'application sont les suivantes :

- L'ensemble des systèmes à dominante électrique et électromécanique pourraient donner les mêmes types de résultats que le système essuie-vitre, i.e. systèmes d'éclairage, systèmes d'ouverture et de fermeture centralisées, portes électriques ...
- Comme la société Actia est également concepteur d'architectures multiplexées pour véhicules industriels (bus de ville, engins de chantier ...), il serait intéressant d'appliquer la méthode à des systèmes de ces véhicules.
- Actuellement, le système de climatisation automatique du véhicule Peugeot 407 est en cours d'étude à Actia dans le cadre d'un stage en Master Recherche. L'objectif de ce stage est d'explorer les limites de notre approche pour un système mettant en jeu des phénomènes physiques plus complexes. Un des premiers éléments de cette étude que nous pouvons mentionner est que beaucoup de défauts pouvant intervenir sur le circuit frigorifique ou sur le circuit de circulation d'air ne correspondent pas à des défauts extrêmes mais à des défauts de déviation de paramètre nécessitant un cadre de raisonnement de diagnostique que nous n'avons pas abordé (X. Olive [Olive, 2003] avait proposé une méthode basée sur un raisonnement qualitatif dans sa thèse). L'élaboration d'une méthode de diagnostic hors-ligne intégrant des modes de défauts de déviation dans un cadre multi-modèle est un axe de recherche intéressant à explorer.

[1]Cette remarque peut toutefois être nuancée par le fait que, même pour générer un arbre manuellement, un expert de diganostic doit nécessairement disposer d'un modèle, même partiel, du système à diagnostiquer

Intégration du module MBR dans le projet MODE

Dans le premier chapitre de ce manuscrit (section 1.3), nous avons écrit que la méthode de diagnostic hors-ligne à base de modèle s'intégrait dans la stratégie MODE qui est une vision plus globale du diagnostic en garage car elle vise à faire collaborer plusieurs méthodes de diagnostic représentées par les projets OBIR (Ontology Based Information Retrieval), DDP (Diagnostic Distribué Préventif), RDF (Reconnaissance des Formes) et notre approche MBR (Model Based Reasoning) dans un même objectif qui est : « localiser le composant défaillant de la manière la plus efficace possible et le remplacer ». Nous indiquons alors plusieurs pistes permettant d'intégrer le module MBR dans cette démarche :

Pré-localisation d'un sous-système avec DDP Dans le cadre du projet DDP, le diagnostic réalise une détection d'incohérence en surveillant en ligne les messages (événements) qui circulent sur le bus de données puis suspecte un sous système (calculateur électronique et l'ensemble des capteurs et actionneurs qui lui sont associés) comme étant défaillant. Si le défaut détecté est persistant (donc non fugitif), alors il est possible de tenir compte de cette information et de réaliser un diagnostic hors-ligne avec MBR uniquement sur la partie pré-localisée par DDP.

Collaboration avec le module OBIR Dans le projet OBIR, le diagnostic est réalisé par la recherche sémantique d'une fiche d'incident connu dans une base de cas à partir de symptômes clients saisis en langue naturelle. Si aucune fiche n'est trouvée, alors il est possible d'utiliser l'information contenue dans le symptôme-client en entrée du module MBR de deux manières :

- Le symptôme-client peut dans le cas le plus simple donner une information sur la prestation ou même un sous-système du véhicule à suspecter. Dans ce cas le module MBR peut réaliser un diagnostic à partir de cette information.
- Le symptôme-client peut également être analysé de manière plus fine en le rapprochant des symptômes fonctionnels générés par la méthode de prédiction que nous avons développée. S'il y a une correspondance de trouvée, alors le symptôme client correspond à un des tests de la matrice de signature des défauts et peut par conséquent permettre de réduire l'ensemble des défauts à discriminer avant de commencer une session de séquencement de tests. Pour parvenir à cette collaboration, il est nécessaire que les modules OBIR et MBR utilisent la même ontologie pour la représentation des symptômes. Cet aspect sera très certainement exploré dans les futurs travaux de recherche du laboratoire commun Autodiag.

Transfert technologique

Actuellement, les travaux réalisés dans le cadre de cette thèse sont prévus pour être intégrés, en partie, aux solutions de diagnostic développées par Actia à l'horizon de 2009. La première étape de ce transfert sera d'adapter l'algorithme de séquencement des tests à la solution de diagnotic ACTI-DIAG.

Par ailleurs, un dépôt de brevet est encours de rédaction afin de protéger certaines parties de la méthode de diagnostic proposée.

Perspectives Scientifiques du diagnostic hors-ligne

En plus des perspectives d'application de notre approche de diagnostic, nous discutons ci-après de futurs axes de recherche qui pourraient être explorés afin de l'améliorer.

Problème d'initialisation

Pour réaliser la simulation d'un système pour une configuration et un mode de défaut donnés (étape de prédiction), il est nécessaire de définir les états initiaux de tous les composants qui constituent le système afin de définir l'état initial du système. Comme les systèmes que nous étudions sont des systèmes dynamiques hybrides, cela correspond à définir le mode hybride initial de chaque composant constituant le système ainsi que les valeurs initiales des variables continues.

Cet état initial correspond à la situation dans laquelle le système est supposé être à l'instant où le garagiste sélectionne une configuration pour réaliser un test. Or, pour certains composants physiques ayant un modèle dynamique, il est parfois difficile d'anticiper réellement cet état initial qui dépend de l'environnement dans lequel se trouve le véhicule.

Pour illustrer ce problème, revenons à l'exemple du système essuie-vitre arrière. Sur la figure 8.1, nous représentons « à plat » le niveau structurel le plus bas de la partie électromécanique de ce système. Nous rappelons que la commande en fermeture de l'interrupteur `K1` permet d'alimenter le moteur électrique. Le composant `tringlerie` permet de transformer le mouvement de rotation continue en sortie du moteur en mouvement de rotation alternatif afin d'obtenir le mouvement de balayage aller/retour souhaité. Le comportement sous forme d'automate hybride de ce composant avait été donné dans le chapitre 4 (cf. figure 4.9). Par ailleurs, l'interrupteur `K_af` est fermé si le balais essuie-vitre est en position repos.

Dans le chapitre 7 (section 7.2.3), nous avons fait l'hypothèse que quelque soit la simulation effectuée la position initiale du balais correspond toujours à la position repos, i.e. l'interrupteur `K_af` est alors en position fermée. Or, pour certains modes de défaut simulés, nous vérifions que cette hypothèse n'est pas toujours réaliste. Nous présentons un cas illustrant ce problème :

Supposons que nous voulions calculer le résultat de la mesure de la tension U aux bornes de l'interrupteur `K_af` sous l'occurrence du défaut « circuit ouvert » du fusible (claquage du fusible non dû à un défaut cascadé) et lorsque le système est dans la configuration $CONF_2$, correspondant à une demande d'essuyage par l'utilisateur. Pour ce test, deux résultats sont possibles en fonction de la situation selon laquelle le défaut est apparu :

- si le fusible fond lorsque l'actionneur est à l'arrêt et que les balais sont en position repos, alors la valeur de la tension U est une constante égale à 0 volts car l'interrupteur `K_af` est fermé.

– si le fusible fond lorsque l'actionneur est en fonctionnement alors l'occurrence de ce défaut provoque un arrêt des balais dans une position pouvant être différente de la position de repos. Dans ce cas, l'interrupteur `K_af` reste ouvert et la valeur de U est une constante égale à 12 volts.

Ainsi, pour ce test, il existe deux modalités possibles sous l'occurrence du défaut circuit-ouvert du fusible `F1` en fonction des deux cas d'initialisation (balais en position repos ou balais en position intermédiaire). Pour ce système, il serait possible de considérer ces deux cas pour chaque simulation et de comparer les résultats des prédictions obtenues pour tous les tests en fonction de ces deux initialisations différentes. Si le résultat du test est le même sous l'occurrence d'un défaut f_i, alors le test est exclusif pour ce défaut. Sinon, le test devient non exclusif.

Actuellement, cet aspect n'est pas traité dans l'application car la non exclusivité d'un test obligerait à modifier l'algorithme de séquencement qui est basé sur une hypothèse d'exclusivité. De plus, nous ne citons qu'un cas particulier. Une étude plus approfondie de ce problème d'initialisation des simulations des tests serait intéressante à réaliser dans de futurs travaux de recherche afin d'en avoir une caractérisation plus formelle et plus générale qui permettrait de mieux l'appréhender sur d'autres systèmes.

Par ailleurs, la solution que nous proposons ne peut être généralisé à tous les différents cas d'initialisation que nous pourrions avoir. Par exemple, si cette initialisation porte sur une variable physique continue (ex : une température ambiante) il est alors difficile d'anticiper hors-ligne le problème d'initialisation avec l'approche que nous proposons.

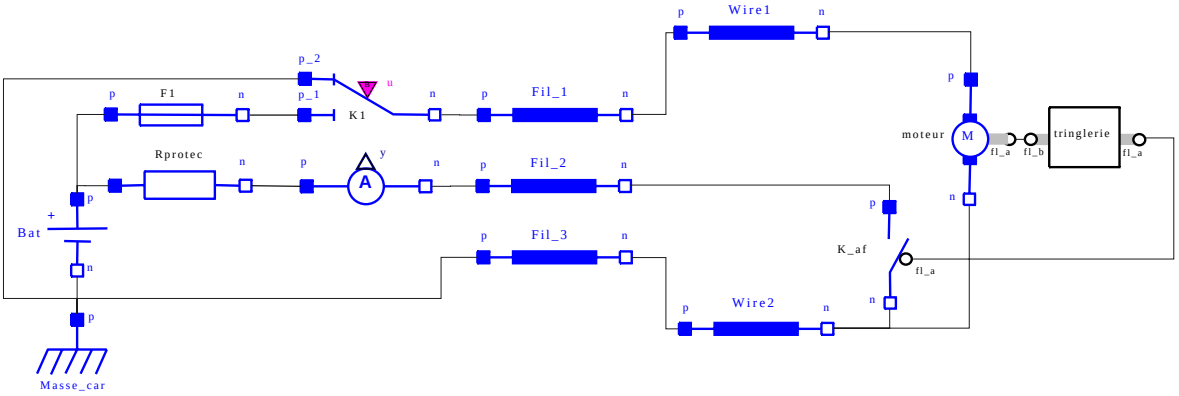


Figure 8.1 — Schéma structurel de la partie électromécanique du système essuie-vitre

Choix des configurations, testabilité

Dans ce manuscrit, les configurations de test que nous simulons correspondent à des cas d'utilisation du système réel, i.e. ceux définis par les pré-conditions des différentes fonctions du système. Pour le système essuie-vitre, qui dispose de peu de configurations de test, cette démarche est satisfaisante. Cependant il pourrait être intéressant d'anticiper plus précisément la testabilité d'un système en déterminant notamment un ensemble de configurations de test minimal permettant d'observer tous les défauts pouvant apparaître sur un système. En effet, il est possible de se retrouver dans les deux cas de figure suivants :

– Si dans la matrice de signatures des défauts le mode de bon fonctionnement ne peut pas être discriminé par rapport à certains modes de défaut, alors les configurations de test sont insuffisantes car le défaut ne peut être observé avec les tests prédits.
– Pour des systèmes plus complexes, comme le système de climatisation automatique, il existe un nombre de configurations de test possibles beaucoup plus important que pour le système essuie-vitre. Dans ce cas, simuler toutes les configurations possibles est certainement inutile car certaines seulement ont un intérêt réel pour le diagnostic.

Dans la communauté du diagnostic logique, ce problème a été formulé dans certains travaux dans lesquels la notion de configuration est appelée « stimuli » [McIlraith et Reiter, 1992; Esser et Struss, 2007]. Il serait intéressant de s'en inspirer pour évaluer efficacement la testabilité des systèmes que nous avons à diagnostiquer. Les travaux de [Bayoudh *et al.*, 2008b] sur le diagnostic actif des systèmes hybrides peuvent également apporter des éléments de réponse.

Construction automatique des processus physiques et explication causale

Nous avons vu dans le chapitre 3 que la constuction d'une hiérachie fonctionnelle consistait à réaliser une interprétation causale du comportment afin d'en déduire les principaux processus mis en jeu pour la réalisation des fonctions (buts) du système. La causalité est en fait un concept très utilisé dans la communauté de l'intelligence artificielle et notamment dans le domaine du raisonnement qualitatif [Travé-Massuyès et Dague, 2003; Gentil et Montmain, 2004] où elle permet des tâches d'explication et de prédiction des comportements des systèmes physiques. Ce concept provient de l'idée que l'être humain a souvent recours à des explications en termes causaux. Il est alors utilisé par les ingénieurs pour appréhender comment un système physique fonctionne (ou ne fonctionne pas comme il le devrait) dans des tâches de raisonnement comme la conception, le diagnostic [Darwiche et Pearl, 1994; Darwiche, 1995] ou la supervision [Berruet *et al.*, 2002; Sekhri *et al.*, 2002].

Dans l'approche multi-modèle développée dans [Zouaoui *et al.*, 1999; Thétiot, 1999] (qui est une extension aux Bond Graphs causaux de l'approche de L. Chittaro), les processus physiques sont liés au comportement par leurs co-fonctions correspondant à des chemins causaux suivis dans le Bond Graph causal. Dans le cadre de nos travaux, nous nous sommes demandés s'il n'était pas possible de réaliser cette même abstraction mais à partir d'un modèle causal obtenu par analyse structurelle du modèle comportemental du système suivant l'approche proposée dans [Travé-Massuyès et Pons, 1997]. Cette étude prospective a notamment fait l'objet d'un article publié en 2006 à SAFEPROCESS et DX ([Ressencourt *et al.*, 2006]). Nous n'avons pas poursuivi dans cette direction car, d'une part, les systèmes visés dans notre approche sont physiquement trop simples pour justifier de l'utilité du modèle causal (peu de phénomènes physiques en interaction). Cependant de futurs travaux de recherche pourraient reprendre cette étude pour les raisons suivantes :
– réaliser un raisonnement qualitatif afin de traiter les défauts de déviation de paramètres non traités dans ce manuscrit.
– modéliser des systèmes physiques du véhicule plus complexes et à haut niveau. i.e. climatisation automatique, circuit hydraulique ...
– retourner des explications au garagiste lors du séquencement des tests. Cet aspect

pourrait être essentiel d'un point de vue cognitif pour l'acceptation de l'outil par le garagiste.

Bibliographie

A. ABU-HANNA et W. JANSWEIJER : Modeling domain knowledge using explicit conceptualization. *IEEE Expert : Intelligent Systems and Their Applications*, 09(5):53–64, 1994. ISSN 0885-9000.

J. ARMENGOL, L. TRAVÉ-MASSUYÈS, J.L. de la ROSA et J. VEHÍ : On modal interval analysis for envelope determination within the ca–en qualitative simulator. *Dans Seventh Information Processing and Management of Uncertainty in Knowledge-Based Systems Conference (IPMU'98)*, pages 110–117, Paris, France, 1998.

K. AUTIO et R. REITER : Structural Abstraction in Model-Based diagnosis. *Dans 13th European Conference on Artificial Intelligence ECAI-98*, pages 269–273, Brighton, UK, 1998.

Monet AUTOMOTIVE : Automotive technological roadmap. Deliverable A5, Network of Excellence on Model-based Systems and Qualitative Reasoning (project MONET2), November 2004. http ://monet.aber.ac.uk.

L. AYROLLES, R. FAIVRE et F. GUERRIN : Time abstraction and quantitative/qualitative interpretation of multiple dynamic processes. *Dans 9th International Workshop on Qualitative Reasoning (QR-95)*, Amsterdam, Nederland, 1995.

M. BASSEVILLE, A. BENVENISTE, G. MOUSTAKIDES et A. ROUGEE : Optimal sensor location for detecting changes in dynamical behavior. *IEEE Transactions on Automatic Control*, pages 1067–1075, 1987. ISSN 0018-9286.

M. BAYOUDH, L. TRAVÉ-MASSUYÈS et Xavier OLIVE : Hybrid systems diagnosability by abstracting faulty continuous dynamics. *Dans Proceedings of the17th International Workshop on Principles of Diagnosis (DX-06)*, pages 9–15, Burgos, Spain, 2006.

M. BAYOUDH, L. TRAVÉ-MASSUYÈS et Xavier OLIVE : State tracking in the hybrid space. *Dans Proceedings of the 18th International Workshop on Principles of Diagnosis (DX-07)*, pages 221–228, Nashville, TN, USA, 2007.

M. BAYOUDH, L. TRAVÉ-MASSUYÈS et Xavier OLIVE : Hybrid systems diagnosis by coupling continuous and discrete event techniques. *Dans 17th IFAC World Congress, Séoul*, Seoul, Korea, 2008a.

M. BAYOUDH, L. TRAVÉ-MASSUYÈS et Xavier OLIVE : Towards active diagnosis of hybrid systems. *Dans Proceedings of the 19th International Workshop on Principles of Diagnosis (DX-08)*, Blue Montains, Australia, 2008b.

J. BELL : *Interpretation of simulation for model-based design analysis of engineered systems*. Thèse de doctorat, University of Wales, Aberystwyth, UK, 2006.

J. BELL et N. SNOOKE : Describing System Functions that Depend on Intermittent and Sequential Behavior. *Dans 18th International Workshop on Qualitative Reasoning QR'04*, pages 51–57, Evanston, (USA), 2004.

J. BELL, N. SNOOKE et C. PRICE : Functional Decomposition for Interpretation of Model-Based Simulation. *Dans 19th International Workshop on Qualitative Reasoning QR'05*, pages 192–198, Austria, 2005.

P. BERRUET, E. CRAYE et A.K.A. TOGUYÉNI : Modèles et algorithmes pour la surveillance-supervision. *Dans* E. NIEL et E. CRAYE, éditeurs : *Maîtrise des risques et sureté de fonctionnement des systèmes de productions - Traité IC2*, chapitre 6, pages 145–194. Ed. Hermès, 2002.

B. CHANDRASEKARAN : Functional representation : A brief historical perspective. *Applied Artificial Intelligence*, 8(2):173–197, 1994a.

B. CHANDRASEKARAN : Functional representation and causal processes. *Advances in Computers*, 38:73–143, 1994b.

B. CHANDRASEKARAN, A. K. GOEL et Y. IWASAKI : Functional representation as design rationale. *IEEE Computer*, 26(1):48–56, 1993.

B. CHANDRASEKARAN et J.R. JOSEPHSON : Representing function as effect. *Dans 5th International Workshop on Advances in Functional Modeling of Complex Technical Systems*, Paris, France, 1997.

L. CHITTARO, G. GUIDA, C. TASSO et E. TOPPANO : Functional and Teleological knowledge in the multimodeling approach for reasoning about physical systems : A case study in diagnosis. *IEEE Transactions on Systems, Man and Cybernetics*, 23(6):1718–1751, 1993.

L. CHITTARO et R. RANON : Hierarchical Model-Based Diagnosis Based on Structural Abstraction. *Advanced Engineering Informatics*, 155(1-2):147–182, 2004.

V. COCQUEMPOT, T. El MEZAYANII et M. STAROSWIECKI : Hybrid dynamical system monitoring using structured analytical redundancy relations. *Dans 17ème IMACS World Congress*, Paris, France, 2005.

M.O. CORDIER, P. DAGUE, F. LEVY, J. MONTMAIN, M. STAROSWIECKI et L. TRAVÉ-MASSUYÈS : Conflicts versus analytical redundancy relations. a comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives. *IEEE Transactions on Systems, Man and Cybernetics Part B*, 34(5):2163–2177, 2004.

P. DAGUE : Théorie logique du diagnostic à base de modèles. *Dans* B. DUBUISSON, éditeur : *Diagnostic, intelligence artificielle et reconnaissance de forme*, chapitre 1, pages 17–105. Hermès Science Europe Ltd, Paris, France, 2001. ISBN 2-7462-0249-2.

P. DAGUE et L. TRAVÉ-MASSUYÈS : Raisonnement causal en physique qualiative. *Dans* L. TRAVÉ-MASSUYÈS et P. DAGUE, éditeurs : *Modèles et raisonnements qualitatifs*, chapitre 7, pages 207–268. Hermès Science Europe Ltd, Paris, France, 2003. ISBN 2-7462-0744-3.

A. DARWICHE : Model-based diagnosis using causal networks. *Dans 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 211–219, Montréal, Québec, Canada, 1995.

A. DARWICHE et J. PEARL : Symbolic causal networks. *Dans 12th national conference on artificial intelligence (AAAI-94)*, pages 238–244, Seattle, Washington, USA, 1994.

J. de KLEER, A. K. MACKWORTH et R. REITER : Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2-3):197–222, 1992a.

J. de KLEER, O. RAIMAN et M. SHIRLEY : One step lookahead is pretty good. pages 138–142, 1992b.

J. de KLEER et B. C. WILLIAMS : Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

J. de KLEER et B. C. WILLIAMS : Diagnosis with behavioral modes. *Dans 11th International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1324–1330, 1989.

B. DUBUISSON, éditeur. *Automatique et Statistiques pour le Diagnostic*. Hermès, Traités IC2, Paris,France, 2001. ISBN 2-7462-0248-4.

O. DUFFAUT : *Problématique Multi-Modèle pour la génération d'Arbres de Tests : Application au Domaine de l'Automobile*. Thèse de doctorat, ENSAE Toulouse, 1994.

J. P. ELLOY : Une spécification « complète » préalable à la sûreté de fonctionnement ? Journée Enjeux de l'Automatique Embarquée de l'AFSEC, 2006.

H. ELMQVIST, D. BRUCK et M. OTTER : *Dymola - Dynamic Modeling Laboratory. User's Manual*. Dynasim AB, 1999.

E. A. EMERSON : Temporal and modal logic. pages 995–1072. MIT Press, Cambridge, MA, USA, 1990. ISBN 0-444-88074-7.

M. ESSER et P. STRUSS : Fault-model-based test generation for embedded software. *Dans IJCAI*, pages 342–347, 2007.

P. P. FAURE : *An Interval Model-Based Approach for Optimal Diagnosis Tree Generation : Application to the Automotive Domain*. Thèse de doctorat, LAAS-CNRS, 2001.

P. P. FAURE, L. TRAVÉ-MASSUYÈS et H. POULARD : An Interval Model-Based Approach for Optimal Diagnosis Tree Generation. *Dans 10th International Workshop on Principles of Diagnosis (DX-99)*, pages 78–89, Loch Awe, Scotland, 1999.

K. D. FORBUS : Qualitative process theory. *Artificial Intelligence*, 24(1-3):85–168, 1984. ISSN 0004-3702.

P. M. FRANK : Advanced fault detection and isolation schemes using nonlinear and robust observers. *Dans 10th IFAC Congress*, volume 3, pages 63–68, Munich, Germany, 1987.

P.M. FRANK : Analytical and qualitative model-based fault diagnosis, a survey and some new results. *European Journal of Control*, 2:6–28, 1996.

P. FRITZSON : *Principles of Object-Oriented Modeling and Simulation with Modelica*. Wiley-IEEE Computer Society Pr, 2003. ISBN 0471471631.

P. FRITZSON, P. ARONSSON, P. BUNUS, V. ENGELSON, L. SALDAMLI, H. JOHANSSON et A. KARSTRÖM : The open source modelica project. *Dans 2th International Modelica Conference*, pages 297–306, Germany, 2002.

S. GENTIL, C. GARCIA-BELTRAN et S. CHARBONNIER : Semi-qualitative temporal episodes prognosis for process supervision. *Dans 16th IFAC World Congress*, page 6p., Prague, République Tchèque, 2005.

S. GENTIL et J. MONTMAIN : Hierarchical representation of complex systems for supporting human decision making. *Advanced Engineering Informatics*, 18(3):143–159, 2004.

J. J. GERTLER : *Fault detection and diagnosis in engineering systems*. CRC Press, 1998. ISBN 0-8247-9427-3.

J. C. GODESKEN : Fault models for embedded systems. *Dans Proceedings of the 10th IFIP Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME-99)*, pages 354–359, London, UK, 1999.

A. K. GOEL et B. CHANDRASEKARAN : Functional representation of designs and redesign problem solving. *Dans International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1388–1394, Detroit, Michigan, USA, 1989.

W. HAMSCHER, L. CONSOLE et J. de KLEER, éditeurs. *Readings in model-based diagnosis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992. ISBN 1-55860-249-6.

D. HAREL et A. NAAMAD : The STATEMATE semantics of statecharts. *ACM Transactions on Software Engineering and Methodology*, 5(4):293–333, 1996.

D. J. HATLEY et I. A. PIRBHAI : *Strategies for real-time system specification*. Dorset House Publishing Co., Inc., New York, NY, USA, 1987. ISBN 0-932633-04-8.

J.P. HAUTIER, J. FAUCHER et J.P. CARON : Le graphe informationnel causal : Un outil pour analyser, comprendre, représenter. *Dans Jourées 3EI*, Cachan, France, 1999.

R. HAWKINS, J. STICKLEN, J. K. MCDOWELL, T. HILL et R. BOYER : Function-based modeling and troubleshooting. *Applied Artificial Intelligence*, 8(2):285–302, 1994.

N. HUGHES, E. CHOU, C. PRICE et M. LEE : Automating mechanical fmea using functional models. *Dans Proc. of the 12th International Florida Artificial Inteligence Conference (FLAIRS'99*, pages 394–398, Orlando, Florida, USA, 1999.

J. E. HUNT, D. R. PUGH et Chris J. PRICE : Failure mode effects analysis : A practical application of functional modeling. *Applied Artificial Intelligence*, 9(1):33–44, 1995.

R. ISERMANN et P. BALLÉ : Trends in the application of model-based fault detection and diagnosis of technical process. *Control Engineering Practice*, 5:709–719, 1997.

Y. IWASAKI et B. CHANDRASEKARAN : Design verification through function- and behavior-oriented representations : Bridging the gap between function and behavior. Rapport technique, Knowledge System Laboratory, Stanford, California, USA, 1992.

R. A. JOHNSON : An Information Theory Approach to Diagnosis. *Dans Proceedings Reliability Quality and Control*, pages 102–109, 1960.

E. KEOGH et M. PAZZANI : Derivative dynamic time warping. *Dans 1st SIAM International Conference on Data Mining*, page 11p, 2001.

A. M. KEUNEKE : Device representation-the significance of functional knowledge. *IEEE Expert : Intelligent Systems and Their Applications*, 6(2):22–25, 1991. ISSN 0885-9000.

P. KHAYATI : *Système de diagnostic automatique à base de reconnaissance statistique de forme :application à l'automobile*. Thèse de doctorat, LAAS-CNRS, 2003.

M. KINNAERT : Fault diagnosis based on analytical models for linear and nonlinear systems, a tutorial. *Dans 5th IFAC Symposium on Fault detection, Supervision and Safety of Technical Processes (SAFEPROCESS-03)*, pages 37–51, Washington DC, USA, 2003.

Y. KITAMURA, T. SANO, K. NAMBA et R. MIZOGUCHI : A Functional Concept Ontology and Its Application to Automatic Identification of Functional Structures. *Advanced Engineering Informatics*, 16(2):145–163, 2002.

M. LIND : Making sense of the abstraction hierarchy. *Dans Conference on Cognitive Science Approaches to Process Control (CSAPC-99)*, pages 195–200, Villeneuve d'Asc, France, 1999.

Morten LIND : Modeling goals and functions of complex industrial plants. *Applied Artificial Intelligence*, 8(2):259–283, 1994.

K. LUNDE, R. LUNDE et B. MÜNKER : Model-based failure analysis with rodon. *Dans 4th International Conference on the Prestigious Applications of Intelligent Systems of ECAI (PAIS'2006)*, pages 647–652, Italy, 2006.

A. MAHANTI et A. BAGCHI : And/or graph heuristic search methods. *J. ACM*, 32(1):28–51, 1985.

D. MAQUIN, V. COCQUEMPOT, J. P. CASSAR, M. STAROSWIECKI et J. RAGOT : Generation of analytical redundancy relations for fdi purposes. *Dans IFAC Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives (SDEMPED-97)*, page 8p., 1997.

D. MAQUIN, B. MARX et J. RAGOT : Surveillance et diagnostic. *Dans* R. HUSSON, éditeur : *Automatique : du cahier des charges à la réalisation de systèmes*, chapitre 16, page 51p. Sciences Sup, Dunod, Paris, France, 2007. ISBN 978-2-10-050397-1.

S. A. MCILRAITH et R. REITER : On tests for hypothetical reasoning. pages 89–96. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992. ISBN 1-55860-249-6.

S. A. MCILRAITH et R. B. SCHERL : What sensing tells us : Towards a formal theory of testing for dynamical systems. *Dans National Conference on Artificial Intelligence (AAAI-00)*, pages 483–490, Austin, TX, USA, 2000.

J. MELENDEZ et J. COLOMER : Episodes representation for supervision. application to diagnosis of a level control system. *Dans Workshop on Principles of Diagnosis DX'01*, Sansicario,Italy, 2001.

M. MODARRES et S.W. CHEHON : Function-centered modelling of engineering systems using the goal tree-success tree technique and functional primitives. *Reliability Engineering and Systems Safety*, 64:181–200, 1999.

MONET2 : Model based systems in automotive domains : Applications and trends. Rapport technique, Monet2 - Automotive Task Group, 2003. URL `http://monet.aber.ac.uk:8080/monet/docs/tg_minutes_and_reports/automotive/a1_report.pdf`.

P. J. MOSTERMAN, G. BISWAS et S. NARASIMHAN : Measurement selection and diagnosability of complex physical systems. *Dans 8th International Workshop on Principles of Diagnosis (DX-97)*, pages 79–86, Le Mont-Saint-Michel, France, 1997.

S. NARASIMHAN, P. MOSTERMAN et G. BISWAS : A systematic analysis of measurement selection algorithms for fault isolation in dynamic systems. *Dans Working Notes of the 8th International Workshop on Principles of Diagnosis, DX-98*, pages 94–101, Cape Cod, MA, USA, 1998.

N.A.SNOOKE et J.BELL : Abstracting automotive system models from component-based simulation with multi level behaviour. *Dans Proc. of the 16th International Workshop on Qualitative Reasoning, QR'02*, pages 151–160, Barcelona,Spain, 2002.

M. NYBERG : Criterions for detectability and strong detectability of faults in linear systems. *International Journal of Control*, 75:190–501, 2002.

X. OLIVE : *Approche Intégrée à Base de Modèles pour le Diagnostic Hors-Ligne et la Conception : Application au Domaine de l'Automobile*. Thèse de doctorat, LAAS-CNRS, 2003.

K. R. PATTIPATI et M. G. ALEXANDRIDIS : Application of Heuristic Search and Information Theory to Sequential Fault Diagnosis. *IEEE Transactions on Systems, Man and Cybernetics*, 20(4):872–887, juillet 1990.

K. R. PATTIPATI et M. DONTAMSETTY : On a Generalized Test Sequencing Problem. *IEEE Transactions on Systems, Man and Cybernetics*, 22(2):392–396, mars 1992.

R.J. PATTON et J. CHEN : A review of parity space approaches to fault diagnosis. *Dans 1st IFAC Symposium on Fault detection, Supervision and Safety of Technical Processes (SAFEPROCESS-91)*, 1991.

H.M. PAYNTER : *Analysis and Design of Engineering Systems*. MIT Press, Cambridge, USA, 1961.

C. PICARDI, L. CONSOLE, F. BERGER, J. BREEMAN, T. KANAKIS, J. MOELANDS, S. COLLAS, E. ARBARETIER, N. De DOMENICO, E. GIRARDELLI, O. DRESSLER, P. STRUSS et B. ZILBERMANN : Autas : A tool for supporting fmeca generation in aeronautic systems. *Dans 16th Eureopean Conference on Artificial Intelligence (ECAI-2004)*, pages 750–754, 2004.

G. POLI, J.F. MARI, J.H. SAITO et A.L.M. LEVADA : Voice command recognition with dynamic time warping (dtw) using graphics processing units (gpu) with compute unified device architecture (cuda). *Dans 19th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2007)*, pages 19–25, Rio Grande do Sul, Brésil, 2007.

H. POULARD : Improvement to the baarycentric correction procedure. *Dans Proc. of WCNN 95, volume 1*, pages 710–713, Washington DC, 1995.

H. POULARD : *Statistiques et réseaux de neurones pour un système de diagnostic :application au diagnostic de pannes automobiles*. Thèse de doctorat, LAAS-CNRS, 1996.

C. PRICE : Autosteve : electrical design analysis. *IEE Colloquium on Applications of Model-Based Reasoning*, pages 59–63, 1997.

C. PRICE, D. R. PUGH, M. S. WILSON et N. SNOOKE : The flame system : Automating electrical failure mode effects analysis (fmea). *Dans Annual Reliability and Maintainability Symposium*, Washington D.C., 1995.

C. J. PRICE, N. SNOOKE, D. R. PUGH, J. E. HUNT et M. S. WILSON : Combining functional and structural reasoning for safety analysis of electrical designs. *Knowl. Eng. Rev.*, 12(3):pp 271–287, 1997. ISSN 0269-8889.

J. RAGOT et D. MAQUIN : Génération d'indicateurs de défauts à base d'observateurs. *Dans* B. DUBUISSON, éditeur : *Automatique : du cahier des charges à la réalisation de systèmes*, chapitre 3, page 51p. Hermès Science Publications, Traités IC2, Paris, France, 2001. ISBN 2-7462-0248-4.

J. RASMUSSEN : *Information Processing and Human-Machine Interaction : An Approach to Cognitive Engineering*. Elsevier Science Inc., New York, NY, USA, 1986. ISBN 0444009876.

R. REITER : A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

H. RESENCOURT : Approche multi-modèle et hiérarchique pour le diagnostic hors-ligne des systèmes embarqués ; application à l'automobile. 7ème Congrès des Doctorants de l'Ecole Doctorale Systèmes (EDSYS), 2006.

H. RESSENCOURT, L. TRAVÉ-MASSUYÈS, H. POULARD et J. THOMAS : Model based testing of automotive functions with mode-mbr for fault localization in the garages. Rapport technique, LAAS-CNRS, 2008.

H. RESSENCOURT, L. TRAVÉ-MASSUYÈS et J. THOMAS : Hierarchical modelling and diagnosis for embedded systems. *Dans 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS'2006)*, pages 553–558, Beijing, China, 2006. Also

in *17th International Workshop on Principles of Diagnosis (DX'06)*, Burgos, Spain, 2006, pages 235–242.

Axel REYMONET, Nathalie AUSSENAC-GILLES et Jérôme THOMAS : Tâche, domaine et application : influences sur le processus de modélisation de connaissances. *Dans Actes des 17e journées francophones d'ingénierie des connaissances*, Juin 2006.

Axel REYMONET, Jérôme THOMAS et Nathalie AUSSENAC-GILLES : Modélisation de Ressources Termino-Ontologiques en OWL. *Dans Actes des 18e journées francophones d'ingénierie des connaissances*, July 2007a.

Axel REYMONET, Jerome THOMAS et Nathalie AUSSENAC-GILLES : Modelling ontological and terminological resources in owl dl. *Dans Proceedings of ISWC '07 workshop "From Text to Knowledge : The Lexicon/Ontology Interface" (OntoLex '07)*, November 2007b.

M. SACHENBACHER, A. MALIK et P. STRUSS : From electrics to emissions : Experiences in applying model-based diagnosis to real problems in real cars. *Dans Working Papers of the 9th International Workshop on Principles of Diagnosis (DX-98)*, pages 246–253, Cape Cod, MA, USA, 1998. Also in *Working Papers of the ECAI-98 Workshop on Model-based Systems and Qualitative Reasoning*, Brighton, UK, 1998.

M. SACHENBACHER, P. STRUSS et C. M. CALÉN : A prototype for model-based on-board diagnosis of automotive systems. *AI Communication*, 13(2):83–98, 2000.

H. SAKOE et S. CHIBA : Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

S. SALVADOR et P. CHAN : Fastdtw : Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.*, 11(5):561–580, 2007.

W. SEIBOLD : A model based diagnosis and simulation system in practical use, the concept of rodon. *Dans 5th International Workshop on Principles of Diagnosis (DX-94)*, New Paltz, NY,USA, 1994.

L. SEKHRI, A.K.A. TOGUYENI et E. CRAYE : A relational based approach for analysing functional graphs of automated production systems. *Dans IEEE International Conference on Systems, Man and Cybernetics*, pages 55–59, 2002.

N. SINGH : Saturn : an automatic test generation system for digital circuits. pages 348–353, 1992.

N. A. SNOOKE et C. J. PRICE : Hierarchical functional reasoning. *Knowledge-Based Sysems*, 11 (5-6):301–309, 1998.

S. SOLDANI, M. COMBACAU, A. SUBIAS et J. THOMAS : Intermittent fault diagnosis : a diagnoser derived from the normal behavior. *Dans 1st IFAC Workshop on Dependable Control of Discrete Systems, DCDS'07*, pages 261–266, Cachan, Paris (France), Juin 2007a. Also in *18th International Workshop on Principles of Diagnosis, DX'07*, pages 391–396, Nashville, Tenessee (USA), (mai 2007).

S. SOLDANI, M. COMBACAU, A. SUBIAS et J. THOMAS : On-board diagnosis system for intermittent fault : Application in automotive industry. *Dans 7th IFAC International Conference on Fieldbuses and Networks in industrial and Embedded Systems, FET'07*, pages 151–158, Toulouse (France), Novembre 2007b.

S. SOLDANI, M. COMBACAU, J. THOMAS et A. SUBIAS : Intermittent fault detection through message exchanges : a coherence based approach. *Dans 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS'2006*, pages 1549–1554, Beijing (Chine), Août 2006. Also in *17th International Workshop on Principles of Diagnosis, DX'06*, pages 251–256, Burgos (Espagne), (Juin 2006).

M. STAROSWIECKI et P. DECLERCK : Analytical redundancy in non-linear interconnected systems by means of structural analysis. *Dans IFAC Symposium on Advanced Information Processing in Automatic Control, (AIPAC'89)*, pages 23–27, Nancy, France, 1989.

Jon STICKLEN et B. CHANDRASEKARAN : Integrating classification-based complied level reasoning with function-based deep level reasoning. *Applied Artificial Intelligence*, 3(2-3):275–304, 1989.

Peter STRUSS et Oskar DRESSLER : "physical negation" integrating fault models into the general diagnostic engine. *Dans 11th International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1318–1323, 1989.

V. SUMBUGAMOORTHY et B. CHANDRASEKARAN : Functional representations of devices and compilation of diagnostic problem-solving systems. *Dans* J. L. KOLODNER et C. K. RIESBECK, éditeurs : *Experience Memory and Reasoning*, chapitre 4, pages 47–73. Lawrence Erlbaum Associates, 1986. ISBN 0898596440.

R. THÉTIOT, F. ZOUAOUI, M. DUMAS, P DAGUE et T. RENAUD : Automatic construction of processes from bond graph representation. *Dans Proc. of the International Workshop on Qualitative Reasoning QR'98*, pages 131–136, Cape Code, (USA), 1998.

J. THOMAS, N. AUSSENAC-GILLES, C. CHABAUD, M. COMBACAU, O. DUFFAUT, C. DUMAZEAU, B. JAMES, H. POULARD, H. RESSENCOURT, A. REYMONET, S. SOLDANI, J.L. SOUBIE, A. SUBIAS et L. TRAVÉ-MASSUYÈS : Heterogeneous knowledge based diagnosis. Rapport LAAS N°08477, Soumis à AI Communications, 2008.

R. THÉTIOT : *Utilisation de l'approche Multi-Modèles pour l'aide au diagnostic d'installations industrielles*. Thèse de doctorat, Université d'Evry Val d'Essonne, Evry, France, 1999.

L. TRAVÉ-MASSUYÈS et P. DAGUE : *Modèles et raisonnement qualitatifs*. Hermès, Traité IC2 Information, Commande, Communications, Paris, 2003. ISBN 2-7462-0744-3.

L. TRAVÉ-MASSUYÈS et R. PONS : Causal ordering for multiple mode systems. *Dans 11th International Workshop on Qualitative Resoning (QR-97)*, pages 203–214, Cortona, Italie, 1997.

L. TRAVÉ-MASSUYÈS, J. THOMAS, X. OLIVE et H. POULARD : Pour une automobile plus sûre et plus fiable. fiabilité prédictive des assemblages mécatroniques. diagnostic automobile. Rapport technique, LAAS-CNRS, 2004.

L. TRAVÉ-MASSUYÈS, T. ESCOBET et X. OLIVE : Diagnosability analysis based on component-supported analytical redundancy relations. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 36(6):1146–1160, 2006.

P. T. WARD et S. J. MELLOR : *Structured Development for Real-Time Systems*. Prentice Hall Professional Technical Reference, 1991. ISBN 0138546541.

D. S. WELD et J. de KLEER, éditeurs. *Readings in qualitative reasoning about physical systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1-55860-095-7.

J. ZAYTOON, éditeur. *Systèmes dynamiques hybrides*. Hermès, Traités IC2, Paris,France, 2001.

F. ZOUAOUI, R. THÉTIOT et M. DUMAS : Behavioral interpretation according to multimodeling representation. *Applied Intelligence*, 10(2-3):211–224, 1999. ISSN 0924-669X.

Liste des figures

A Légende du modèle structurel

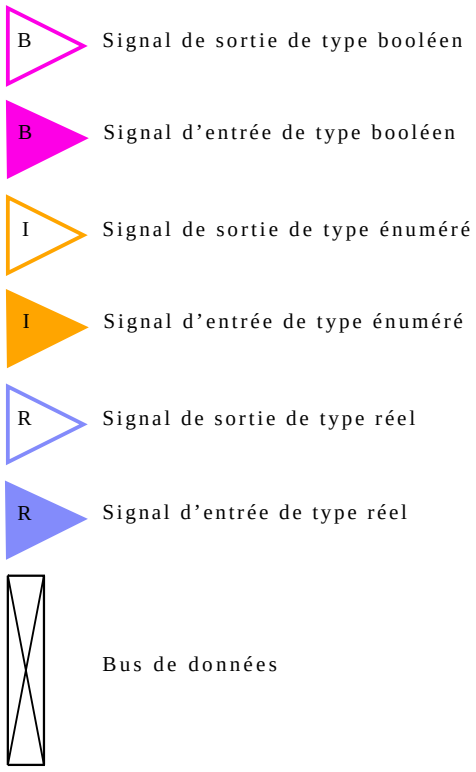


Figure A.1 — Légende des ports du type signal

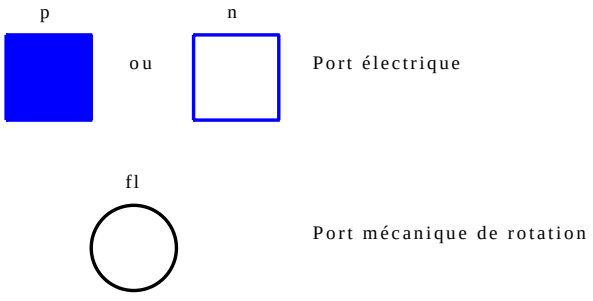


Figure A.2 — Légende des ports physiques

B Modèle du système essuie-vitre arrière pour deux véhicules réels

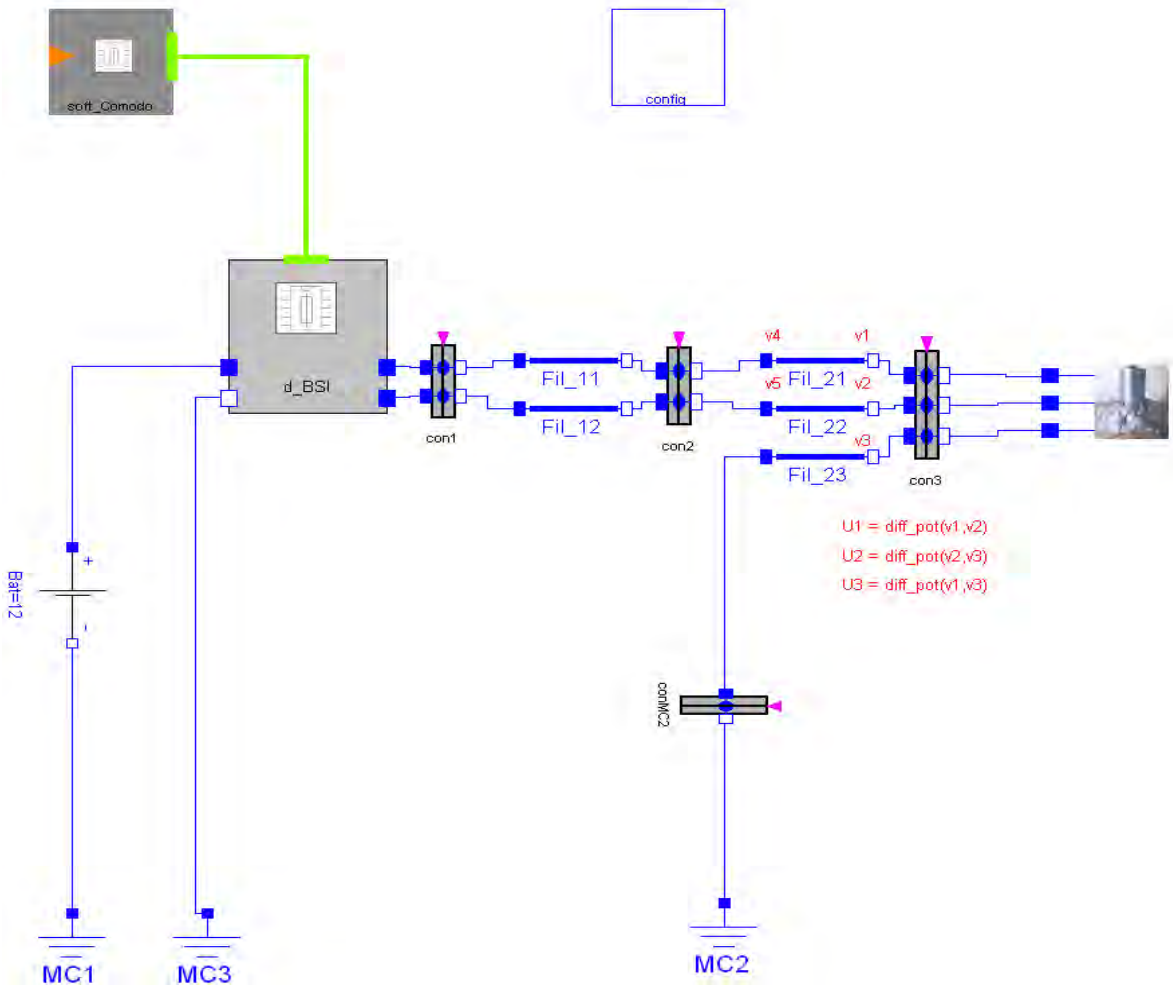


Figure B.1 — Modèle structurel du système essuie-vitre arrière du constructeur PSA. Essais effectués sur Peugeot 1007 et Citroën C4

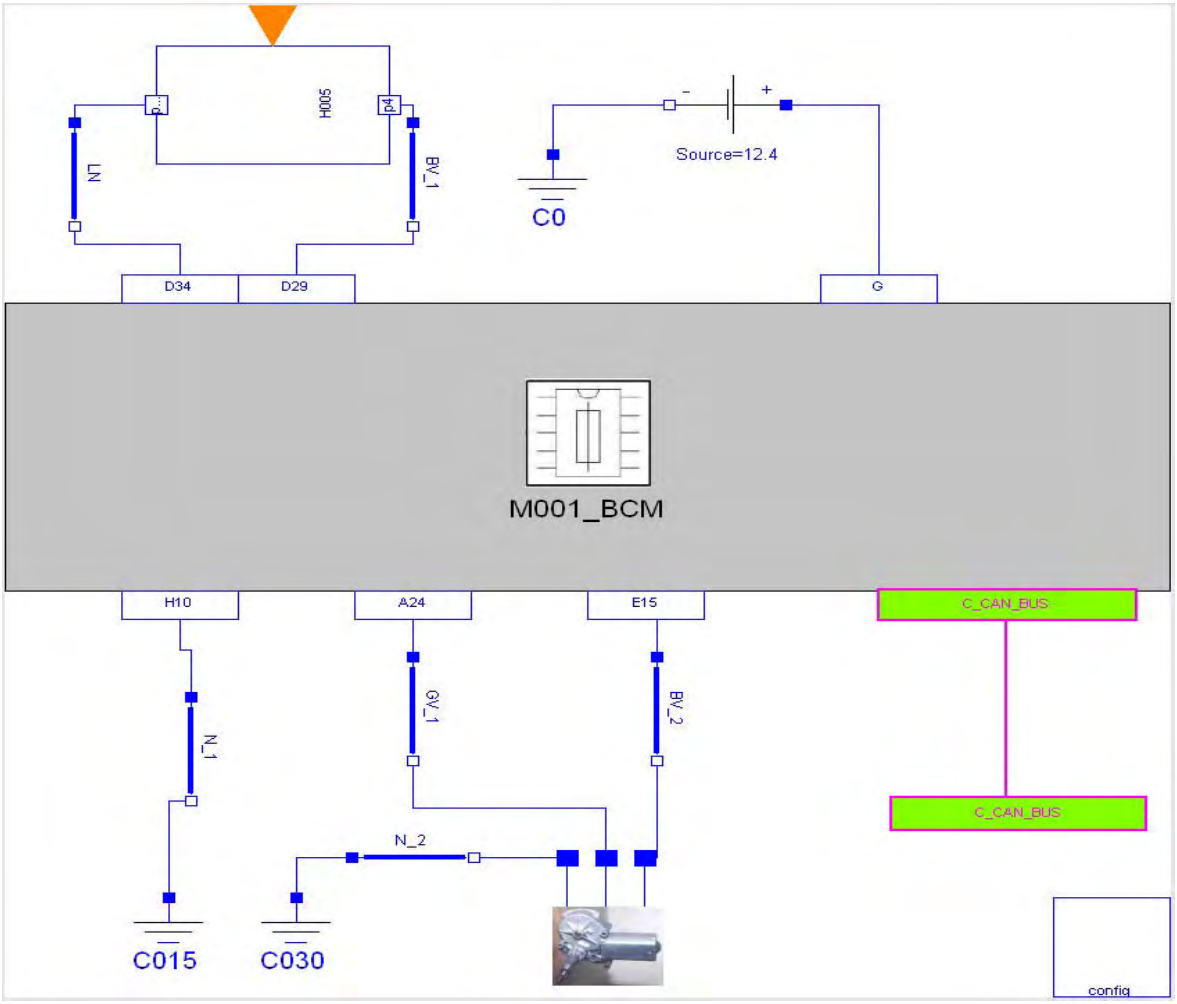


Figure B.2 — Modèle structurel du système essuie-vitre arrière du véhicule Fiat Grande Punto

Hervé Ressencourt

Diagnostic hors-ligne à base de modèles : approche multi-modèle pour la génération automatique de séquences de tests

Directrice de thèse :
Louise Travé-Massuyès, LAAS-CNRS
Université de Toulouse

Résumé

Le travail de cette thèse s'intéresse au problème du diagnostic débarqué dans le domaine automobile. Il a consisté à proposer et à mettre en œuvre une méthode opérationnelle à base de modèles qui détermine les meilleures séquences de tests que doit réaliser le garagiste afin de localiser un composant défaillant sur un véhicule. Nous proposons une approche de représentation multi-modèle des systèmes mécatronique afin de prendre en compte la complexité fonctionnelle des architectures embarquées actuelles et de relier des symptômes fonctionnels de haut niveau à un ensemble de défauts portant sur des composants matériels et logiciels. Le séquencement des tests est réalisé à partir d'un critère du prochain meilleur test. Cette stratégie interactive laisse l'initiative à l'opérateur humain d'accepter ou de refuser le test proposé. Un prototype logiciel a été développé et testé avec succès sur la fonction essuyage arrière de véhicules réels. Cette thèse a été réalisée dans le cadre d'une convention CIFRE entre le LAAS-CNRS et la société ACTIA ainsi que dans le cadre du Laboratoire Commun Autodiag (LAAS, IRIT, ACTIA) dont l'objectif est de développer de nouvelles méthodes de diagnostic pour le domaine automobile.

Mots-clés : diagnostic à base de modèles, diagnostic hors-ligne, approche multi-modèle, séquencement de tests, systèmes mécatroniques embarqués

Laboratoire d'Analyse et d'Architecture des Systèmes - UPR 8001
7, avenue du Colonel Roche, 31077 Toulouse Cedex 4

Hervé Ressencourt

Off-Board Model Based Diagnosis : Multi-model Approach for the automatic generation of test sequences. Application to the automotive domain

Supervisor :
Louise Travé-Massuyès, LAAS-CNRS
Université Paul Sabatier

Abstract

This thesis deals with the problem of off-board diagnosis in the automotive domain. The work has consisted in proposing and implementing an operational model based approach that determines the best sequences of tests to be performed by the garage mechanic to localise a faulty component on a vehicle. A multi-model approach is proposed for the description of mechatronic systems, which allows us to handle the functional complexity of embedded systems and to match functional symptoms with a set of faults on hardware / software components. The test sequencing problem is approached along a next best test strategy based on a local heuristic. This strategy enables an interactive diagnostic session, allowing more flexibility and leaving with the human operator the initiative to accept or reject the proposed test. A software prototype has been developed and tested on the rear wiper system of real vehicles. This thesis, supported by a CIFRE grant, is the result of collaboration between the company ACTIA and the research center LAAS-CNRS in the framework of the common laboratory Autodiag (LAAS, IRIT, ACTIA) which aims at developing new methods for diagnosis in the automotive domain.

Key words : model based diagnosis, off-board diagnosis, multi-model approach, test sequencing, embedded mechatronic systems

Laboratoire d'Analyse et d'Architecture des Systèmes - UPR 8001
7, avenue du Colonel Roche, 31077 Toulouse Cedex 4