Evaluation de la validité de la simulation dans le cadre du développement des systèmes embarqués

Vincent Albert

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *Université Toulouse III - Paul Sabatier*
Discipline ou spécialité : *Systèmes Informatiques*

Présentée et soutenue par *Vincent ALBERT*
Le *30 Septembre 2009*

Titre : *Evaluation de la Validité de la Simulation dans le Cadre du Développement des Systèmes Embarqués*

JURY
Michel Gourgand - Président du jury
Norbert Giambiasi - Rapporteur
Jean-Pierre Bourey - Rapporteur
Alexandre Nketsa - Examinateur
Mario Paludetto - Examinateur
Hessam Sarjoughian - Examinateur
Jean Casteres - Examinateur
René Jacquart - Examinateur

Ecole doctorale : *EDSYS*
Unité de recherche : *LAAS-CNRS*
Directeur(s) de Thèse : *Alexandre Nketsa*

À celui qui ne m'a pas attendu... et c'est pas juste

Remerciements

Cette thèse fût une expérience enrichissante sur le plan humain. Embûches, parfois embuscades, se sont trouvées sur mon chemin. L'avoir vécu à fond est peu dire, y avoir pris goût, c'est certain. La simple présence des gens d'ici, m'a aidée à trouver le bout du chemin.

Ma première pensée revient à mes parents, mon frère et ma soeur. M'inspirant curiosités, tempérence et autres vertus morales, le mérite de m'y avoir fait goûter leur revient.

J'exprime ma plus profonde gratitude à Alexandre Nketsa et Mario Paludetto pour la confiance et le soutien permanent qu'ils m'ont porté.

Je remercie très particulièrement Christel Seguin, Jean Casteres et René Jacquart. Je suis ravi d'avoir eu la chance de collaborer avec eux et de l'intérêt qu'ils ont porté à l'égard de mon travail.

Je remercie Jean-Pierre Bourey et Norbert Giambiasi d'avoir accepté de rapporter ce travail ainsi que Michel Gourgand et Hessam Sarjoughian pour leurs conseils et d'avoir accepté de participer au jury de soutenance.

Je remercie Jean-Jacques, Patrice, Dominique et Florent qui, outre m'avoir tant fait rire, m'ont toujours soutenu et encouragé.

Je remercie mes amis du laboratoire. Mehdi l'insatiable, ma source de formalisation mathématique, de rires, parfois d'indignation. Xavier, l'homme aux milles fientes de l'esprit qui vole. Fabien, celui qui ne trouve son chemin qu'au clair de lune et qui, comme punition, aperçoit l'aurore avant les autres hommes. Monsieur Bemke, celui qui possède d'innombrables talents dont celui de n'avoir quelque envie d'abaisser les autres hommes. Louise, celle qui, très justement, sait faire vibrer ses cordes afin de nous permettre de gravir des sommets. Francescou, qui lui aussi fait vibrer des cordes (mais pas les mêmes), celui qui maniait le poum-tchak avec délectation pour m'accompagner dans mes élucubrations.

Enfin je remercie tout mes amis qui me remplissent de bonheur depuis tant d'années, Tonton, Tonia, Papacouer, Peck, Philibierre, Swesh, Sharky, Jc, Hemo, Pichou, Seb, Guitou, Tophe et Tof, You, Pinju, Kev, Contin, Bharat, Anto, Ninou, Mavros.

Table des matières

Table des figures

Introduction générale

Même pris dans sa plus grande généralité, le terme *système* désigne un ensemble d'unités réelles ou abstraites qui interagissent afin de servir un objectif commun (Wikipédia). Parfois l'environnement du système désigne tout ce qui n'a aucune relation avec lui, parfois il représente, au contraire, l'univers dans lequel il est plongé et par conséquent ses interactions possibles. Les limites du système considéré fixent ses relations. Cette définition très générale s'applique évidemment à l'ingénierie des systèmes et donc à ceux que l'homme conçoit et réalise pour son profit et son bien être et qu'il est amenés, par voie de conséquence, à piloter, surveiller et maintenir.

Dans chaque discipline, des modèles sont réalisés dans le but d'expérimenter et de prédire, de tenter d'approcher et d'universaliser, les concepts d'un système. Sociologues, biologistes, écologistes et ingénieurs ont tous recours à ce processus intellectuel de modélisation visant à définir ce qui constitue et caractérise un système et comprendre comment il évolue et comment il se comporte.

L'avènement des outils de calcul numérique dans les années 40 et leur développement exponentiel depuis l'apparition des circuits intégrés durant les années 60, a donné une importance considérable aux études sur les systèmes ayant pour support, tant par leur conception que par leur mise en oeuvre, les ordinateurs ou, d'une façon plus large, les processeurs numériques.

Cela s'accompagne d'un effort de recherche lui aussi considérable tant théorique qu'à caractère appliqué dans le but de disposer d'outils grâce auxquels concevoir un système devient une activité dont le résultat garantit à l'homme une qualité de service et une sécurité prédéfinie.

Le secteur d'ingénierie de l'avionneur Airbus a décidé de renforcer sa façon de développer de nouveaux projets en concentrant ses efforts sur une utilisation maximale de la Modélisation et de la Simulation (M&S) pour étudier le spectre le plus large possible des caractéristiques de l'avion. L'évaluation de la validité attachée à une simulation devient alors une base essentielle de résultats qui garantit à l'ingénieur la qualité de la conception des systèmes.

La simulation est déjà très largement utilisée dans les processus d'ingénierie comme moyen d'aide à la décision. L'établissement d'une estimation de la validité

(correction, fidélité, maturité, représentativité) est obligatoire afin de formellement évaluer le niveau de confiance que l'on peut accorder à une simulation au regard de son environnement et de ses objectifs d'utilisation. La question de la quantité de fidélité nécessaire à un besoin et, manifestement, l'effort requis pour atteindre ce niveau de fidélité sont les principaux enjeux des futurs développements de l'avionneur.

Si nous considèrons qu'une simulation est un modèle, ou un ensemble de modèles, soumis à un environnement d'exécution qui anime ce modèle dans le temps, nous distinguons la validité du modèle et la correction de l'environnement d'exécution, i.e. le simulateur. L'environnement d'execution ajoute des contraintes additionnelles et des erreurs d'implémentation à la validité du modèle. Cet environnement est constitué de systèmes informatisés complexes constitués eux-mêmes de composants (modèles de simulation et calculateurs réels) et d'une infrastructure (ordonnanceurs temps réel, processeurs numériques, interface eléctroniques, et moyens de communication), chacun ayant sa part de contribution sur le niveau de validité de la simulation.

À ce jour, aucun niveau de confiance d'une simulation n'est formellement défini et exprimé comme une approche synthétique, approuvée et standardisée. Ceci est manifestement due à l'apparition récente de la discipline. De nombreux tests de validation sont réalisés à tous les niveaux d'intégration des simulations (tests de non régression, recoupements d'essai en vol, tests exhaustifs de systèmes), chaque branche d'ingénierie ayant ses propres moyens et outils, mais aucune stratégie de validation cohérente n'est définie.

Les stratégies de validation des simulations vont pourtant bon train depuis le début du siècle. Boeing investit considérablement dans la Modélisation et la Simulation (M&S). Il contrôle visiblement diverses initiatives et divers groupes de standardisation traitant ces aspects. Ces groupes de travaux incluent le Simulation Fidelity Group [Roza 99]. Le US Department of Defense (USDoD) et le US Department of the Navy se s'ont chacun doté d'une organisation officielle, le Defense Modelling and Simulation Office (DMSO) et le Navy Modeling and Simulation Management Office. Ils ont élaboré un standard de Vérification, Validation et Accréditation (VV&A) d'une M&S [DMSO 96] [NAVY 04]. De même du coté de l'US Air Force, pour qui la RAND (Research and Development) Corporation a développé un standard pour mesurer les implications de la validation de modélisation multi-résolution et multi-perspectives (MRMPM) [Bigelow 03]. Au Sandia National laboratories

nous parlons d'"estimation totale de l'incertitude d'une M&S" [Rutherford 00]. En France, la Direction Générale de l'Aviation Civile (DGAC) soutien l'International Test Operations Procedure (ITOP) [ITOP 04].

Au même titre que le CMMI [SEI 06] dans l'ingénierie logicielle ou que l'EIA-632 [EIA 99] et l'INCOSE [INCOSE 04] dans le domaine de l'ingénierie système, ces organisations tentent de mettre en place des procédures générales pour plannifier, implémenter et documenter les efforts de VV&A liés aux modèles et aux simulations. Le défi est toujours le même : avoir des processus cohérents et traçables pour diminuer les incertitudes, les risques et, bien sûr, les coûts.

Nous avons constaté que ces stratégies et procédures de VV&A de M&S sont très orientées processus. Notre souhait et notre volonté, contrairement à ces standards émergeants, est de nous focaliser sur les caractéristiques du produit plutôt que sur les aspects du processus de développement du produit, les deux étant bien sûr indispensables et complémentaires. Les besoins d'utilisation de standard d'ingénierie système le prouvent. Si nous pouvons plus facilement aborder ce problème sur le produit, plutôt que sur le processus, c'est bien évidemment parce que nous ne traitons pas des mêmes systèmes que le USDoD par exemple, qui eux utilisent la simulation pour l'élaboration de missiles de défense ou d'entrainement au combat... Le caractère émergeant de ces systèmes rend plus difficile encore la tâche de l'évaluation de la validité de M&S.

Pour notre part, nous étudions la M&S utilisées dans le cadre du développement des systèmes avioniques embarqués. Dans ce contexte, un système est un élément physique constitué de deux composantes : l'équipement avionique proprement dit (ou "end system" dans le jargon de l'EIA-632) et un ensemble de systèmes qui permettent la conception, la production, la vérification, l'exploitation, la maintenance et le recyclage de l'équipement ("enabling system").

À chacun de ces systèmes nous pouvons associer une liste de composants appelée "composition du système" qui sont des objets physiques. Nous pouvons également leur associer un environnement. Un même objet ne peut pas appartenir simultanément à la composition et à l'environnement du système. Par contre, au cours du temps, il peut passer de l'environnement à la composition ou vice versa. En ce qui concerne un équipement avionique, il est toujours possible d'en donner la composition en termes de ports, de processeurs, de mémoires, de bus, de composants

spécialisés, FPGA, ASIC ou composants analogiques.

Ces systèmes sont caractérisés par une nécessaire réactivité et une relation au temps (qui peuvent être très variables selon le domaine) qui imposent généralement une capacité à mémoriser le comportement passé afin d'élaborer le comportement présent, voire de prédire l'évolution future.

Donc un système a des propriétés structurelles qui se rapportent à des aspects statiques (composition, liaisons, masse, dimensions, forme géométrique), et des propriétés comportementales qui se rapportent aux aspects dynamiques (processus qui se déroulent au sein du système, états, modes, actions sur les objets de l'environnement, réactions aux sollicitations des objets de l'environnement).

Pour être efficace, nous devons nous approprier des concepts formels. La théorie générale des systèmes [Bertalanffy 72] peut y répondre. La théorie des systèmes distingue la structure (constitution interne d'un système) du comportement (ses manifestations externes). Elle définit les modèles comme causaux (dont les sorties sont la conséquence d'une entrée) et déterministes (à une entrée donnée ne peut correspondre qu'une seule sortie). Cette théorie sert de base à bon nombre de formalismes tels les équations différentielles, les automates à états finis, les réseaux de Petri... Le concept de décomposition d'un système en sous-composant est également très présent dans la théorie des systèmes. Le caractère ouvert de la théorie, qui laisse apparaître une logique de système globale unique, peut permettre d'élaborer des concepts commun à des systèmes pluridisciplinaires qui collaborent pour réaliser une fonction commune. C'est bien ainsi dans un avion.

La théorie de la Modélisation et de la Simulation, basée sur la théorie des systèmes, tente, quant à elle, à fournir un cadre méthodologique pour la modélisation des systèmes. Contrairement à la théorie des systèmes qui ne vise aucune discipline en particulier, la théorie de la M&S revêt un caractère plus "informatique". Dans ces début [Zeigler 84], cette théorie trace et caractérise les concepts intrinsèques de la modélisation et de la simulation en terme de description, simplification, validation, abstraction, simulation et exploration d'un modèle. Ensuite [Zeigler 00b], B.P Zeigler se concentre sur deux thèmes majeurs : l'intégration des paradigmes de modélisation (discrets et continus) et l'appui de la simulation distribuée sur la coexistance de modèles décrits sous différents formalismes.

Cependant ce qui nous intéressent le plus dans cette théorie, dans le cadre de

notre étude, ce sont les relations entre les constituants d'une activité de M&S. La relation entre un système réel et une représentation de ce système, la relation entre les formalismes, la relation entre un modèle et le simulateur, sont autant de verrous signalés que de problèmes à résoudre. Nous verrons également que la notion de cadre expérimental, permettant de spécifier les conditions d'expérimentation ou d'observation d'un système, est le fondement de notre approche.

La première phase de notre travail a été de nous familiariser avec la vision qu'ont, d'une simulation, les utilisateurs et les développeurs chez Airbus. Comment est-elle utilisée, quand, à quelles fins. Comment perçoivent-ils les notions de validité et de qualité d'une simulation. Ces questions nous ont permis de caractériser le problème posé.

Chaque chapitre de ce manuscrit est composé d'une partie d'état de l'art suivi de la contribution qui en découle.

Le chapitre 1 présente la simulation dans l'ingénierie système. Nous introduisons, dans un premier temps, l'ingénierie système et les processus qui la caractérisent. Ceci nous permettra de spécifier : quel est le périmètre de notre étude, pour quelles étapes d'ingénierie notre approche est censée s'appliquer et quels sont, de manière génerale, les objectifs d'utilisation d'une simulation lors de ces étapes. Dans un second temps, nous définissons les objectifs de l'étude, à partir de notre enquête chez l'industriel. Nous avons ainsi défini les qualités perçues d'une M&S selon trois méta-critères génériques : la correction, la validité et l'efficacité. Nous verrons ce qui se cache derrière ces critères, nous verrons aussi que nous réduisons le périmètre de l'étude au problème de la validité.

Nous ne pouvions pas aborder le problème de l'évaluation de la validité d'une simulation sans étudier les approches proposées par les standards de VV&A. Le chapitre 2 présente un état de l'art sur les concepts et le vocabulaire associés à la modélisation, la simulation, l'abstraction, la vérification, la validation et l'accréditation, le modèle conceptuel et les différents constituants d'une M&S. Cet état de l'art est suivi du cadre méthodologique d'élaboration de modèles et de simulation issue de la théorie de M&S. La seconde partie du chapitre 2 présente le contexte industriel dans un contexte d'ingénierie système. Enfin, nous verrons ce qui caractérise une M&S chez Airbus et comment nous pouvons nous approprier les concepts introduits dans la première partie afin de caractériser le problème de la validité d'une

simulation.

Quotidiennement, les ingénieurs utilisent des abstractions pour modéliser les systèmes. Le meilleur moyen d'évaluer la distance entre un modèle et le système qu'il représente est de tracer les abstractions et les hypothèses qui ont été faites lors de l'élaboration du modèle. Le meilleur moyen de tracer ces abstractions est d'établir une taxonomie. La section état de l'art présente les taxonomies existantes. B.P. Zeigler en a proposé une dans la première édition de la théorie de la M&S, bien qu'elle n'était pas identifiée en tant que telle. Nous étudions également celle de F.K Frantz [Frantz 95], et nous proposons une ébauche de celle de D.S. Weld [Weld 92] dans le domaine de l'Intelligence Artificielle, qui a été et restera pour nous une grande source d'inspiration. Notre contribution consiste, en seconde section, de construire, à partir des pratiques courantes chez Airbus, notre propre taxonomie. Le but ultime est de proposer une hiérarchie de modèles plus ou moins abstraits et d'identifier par des propriétés ce qui caractérise un modèle à chaque niveau de la hiérarchie.

En considerant que la taxonomie d'abstraction de modèles est le langage commun entre utilisateur et développeur d'une simulation, par laquelle un utilisateur définit les abstractions acceptables pour satisfaire un besoin et par laquelle le développeur définit les limitations de sa simulation, notre approche d'évaluation de la validité est basée sur l'applicabilité d'un objectif d'utilisation avec le domaine d'usage d'une simulation. Pour évaluer cette applicabilité, nous nous sommes naturellement tournés vers les techniques d'analyse de signature du génie logiciel. La partie état de l'art du chapitre 4 présente les concepts formels d'ingénierie des composants et de compatibilité qui y sont associés. Cette section est suivie du formalisme de spécification des systèmes issu de la théorie de M&S. La seconde section présente la formalisation de nos propriétés sur ce formalisme de spécification.

Nous ne pouvions terminer cette étude sans tenter d'illustrer les concepts qui y ont été abordés. Le chapitre 5 a pour objectif de montrer comment nos concepts doivent s'intégrer dans un processus d'ingénierie système. La première section présente les différents produits de l'activité de conception d'un système. Puis, nous proposons une méthodologie d'évaluation basée sur les standards d'ingénierie et outillée par le langage SysMl. La deuxième section illustre les concepts d'évaluation de la validité et la méthodologie sous-jacente à travers un cas d'application proposé

par Airbus.

Nous terminons ce manuscrit par une conclusion générale. Nous proposons une synthèse de notre contribution et son intérêt. Ceci nous permet d'ouvrir les portes vers d'autres perspectives et vers tous les travaux qui n'ont pu être traités dans cette étude.

Chapitre 1

Contexte et objectifs de l'étude

Sommaire

1.1 La simulation dans l'Ingénierie Système

1.1.1 Ingénierie Système

1.1.1.1 Processus

Selon l'EIA-632 [EIA 99] un système est *un ensemble de produits nécessaires à la réalisation d'un objectif ou d'une fonction. Un système est composé de produits nécessaires à la réalisation de la fonction opérationnelle ("end product") et de produits nécessaires à sa réalisation, sa mise en service, son maintien et son retrait de service ("enabling products").*

Depuis plus de 30 ans, Airbus Industrie a développé un savoir-faire dans la conception des Systèmes orienté sur une maîtrise de plus en plus accrue des activités

de spécification, de conception et d'analyse. Cette maîtrise est motivée par une volonté d'accroissement de la qualité du système, des produits qui le composent et des services qu'il assure, de réduction des cycles de développement et de favoriser la réutilisation. Pour la conception des systèmes embarqués, l'utilisation de méthodes formelles et le codage automatique de spécification sont une avancée considérable allant dans ce sens. L'A380 confirme la nécessité de bonnes pratiques et d'approches intégrées compte tenu de :

- ses dimensions,
- l'intégration plus forte de l'avion dans son environnement (système de gestion du trafic aérien, système de maintenance des compagnies aériennes, téléphonie, internet...),
- l'intégration plus poussée des systèmes de l'avion en particulier avec l'apparition de l'avionique modulaire et des bus multiplexés,
- la prise en compte d'exigences de maturité des systèmes de plus en plus sévères notamment dès leur entrée en service,
- l'importance des enjeux techniques, industriels et financiers posés.

Selon l'AFIS [AFIS] l'Ingénierie Système est *une démarche méthodologique générale qui englobe l'ensemble des activités adéquates pour concevoir, faire évoluer et vérifier un système apportant une solution économique et performante aux besoins d'un client tout en satisfaisant l'ensemble des parties prenantes.*

Plus précisément, l'Ingénierie Système peut se définir comme [AFIS] un processus coopératif et interdisciplinaire de résolution de problème :

- s'appuyant sur les connaissances, méthodes et techniques issues de la science et de l'expérience,
- mis en oeuvre pour définir, faire évoluer et vérifier la définition d'un système (ensemble organisé de matériels, logiciels, compétences humaines et processus en interaction),
- apportant une solution à un besoin opérationnel identifié conformément à des critères d'efficacité mesurables,
- qui satisfasse aux attentes et contraintes de l'ensemble de ses parties prenantes et soit acceptable pour l'environnement,
- en cherchant à équilibrer et optimiser sous tous les aspects l'économie globale de la solution sur l'ensemble du cycle de vie du système.

Les activités de base de l'ingénierie systèmes sont :

1. définir les objectifs du système,
2. mettre en place la fonctionnalité du système,
3. établir les exigences qui doivent être satisfaites,
4. élaborer une conception et les concepts d'opérations,
5. définir des contraintes de coût,
6. valider que ces contraintes satisfont l'utilisateur,
7. vérifiez que ces contraintes répondent aux exigences,
8. itérer et appliquer le processus aux sous-systèmes.

Ces tâches sont mises en oeuvre par le processus illustré par la figure 1.1. Les tâches énumérées ci-dessus sont réalisées dans le bloc "System Design". Cette étape implique un processus de définition des exigences de performance et fonctionnelles et un processus de définition de ces exigences qui traduit ces exigences en solutions techniques et opérationnelles. Les processus transverses de gestion technique et évaluations techniques sont effectués continuellement tout au long du processus de développement.

Le processus d'Ingénierie Système est utilisé par itération à chaque étape du cycle de développement afin de générer des descriptions plus détaillées du système. Ces descriptions comprennent ce qui doit être atteint (exigences fonctionnelles et concepts des opérations), comment cela doit être atteint (exigences de performance), la manière dont cela est atteint (conception), et les résultats d'analyse et de tests sur la capacité de satisfaire les exigences (vérification) et de satisfaire l'utilisateur (validation). Le processus d'Ingénierie Système conduit à des descriptions plus détaillées pour chaque phase du cycle de développement. Chaque phase commence à partir d'un niveau supérieur de description du système et décompose les exigences du niveau supérieur en exigences de niveau inférieur pour chaque fonction.

1.1.1.2 Vérification et validation

Les activités de vérification et de validation (V&V) d'un systèmes sont très similaires mais traitent de problèmes différents. La vérification consiste à s'assurer que le système, ses éléments, ses interfaces et les produits intermédiaires satisfont

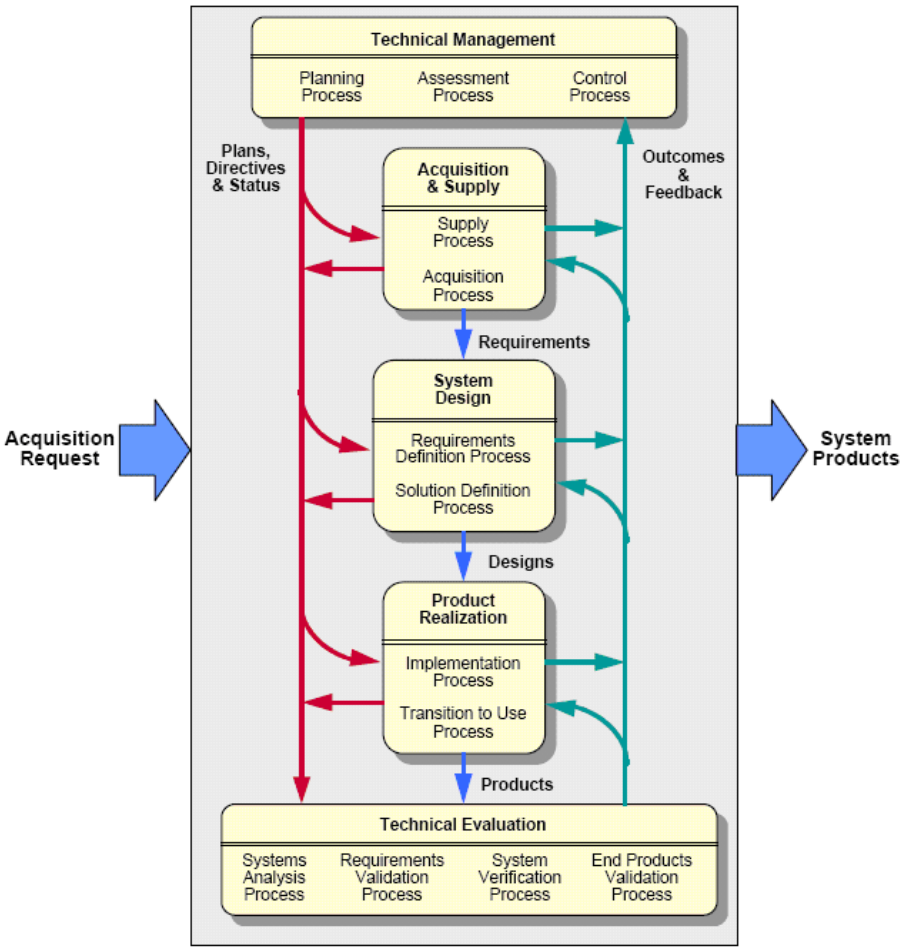

FIGURE 1.1 – Processus d'Ingénierie Système (Source : EIA-632)

leurs exigences. La validation consiste à s'assurer que le système satisfait les besoins de l'utilisateur [EIA 99]. En d'autres termes la vérification veille à ce que le produit soit construit correctement et la validation veille à ce que le bon produit soit construit.

Vérification La vérification est défini par l'EIA-632 comme *la confirmation, par examination et collecte d'évidences objectives, que les exigences à partir desquelles un "end product" est construit, codé, ou assemblé, sont satisfaites.*

La vérification est l'ensemble des tâches, des actions et des activités réalisées pour évaluer l'efficacité des solutions proposées (personnes, produits et processus)

et pour mesurer la conformité aux exigences. Les méthodes de vérification sont diverses : simulation, démonstration, test et inspection. La fonction principale de la vérification est de déterminer que les spécifications, conceptions, processus et produits du systèmes sont conformes aux exigences de haut niveau qui dictent les attentes de l'utilisateur en terme de fonctionnalités, performances et caractéristiques du système et que les processus utilisés respectent le plan de gestion d'ingénierie du système. Comme les descriptions du système sont itérativement allouées, spécifiées, conçues, simulées et testées, une séquence hiérarchique de spécifications, de conceptions et de plans de test adaptée à chaque phase du processus de développement, est produite. La vérification fait partie intégrante du processus de développement incrémental. La fonction secondaire de la vérification est de déterminer, à chaque phase du développement et par des moyens adaptés, que les représentations du système et des sous-systèmes sont conformes aux spécifications et exigences en vigueur à la phase précédente.

Validation La validation est défini par l'EIA-632 comme *la confirmation par examination et collecte d'évidences objectives qu'un "end product", ou l'aggrégation d'"end products", fonctionne comme attendu par le client dans son environnement opérationnel.*

Le matériel et le logiciel sont validées au niveau de l'intégration du système. La validation est l'action de déterminer que le système fait tout ce qu'il est censé faire. La validation est souvent réalisée par une tierce personne autre que le développeur et l'utilisateur. En générale elle est effectuée dans l'environnement opérationnel du système ou une simulation de cet environnement opérationnel. La "validation des exigences" est conduite pour fournir, au plus tôt, l'assurance que celles-ci sont les "bonnes" et qu'elles mèneront le processus de développement à une conclusion qui satisfait l'utilisateur. La validation des exigences est souvent basée sur l'analyse des exigences, l'évaluation de prototypes, des simulations de modèles et de scénarios et retour d'expérience. Les méthodes de validation sont similaires aux méthodes de vérification tel que le test, l'analyse, l'inspection, la démonstration ou la simulation. Les objets de la validation sont les éléments de conceptions, les prototypes et les produits finaux, ainsi que la documentation qui décrit le système. La validation fait partie intégrante du processus de développement incrémental.

1.1.2 Utilisation de la M&S dans l'Ingénierie Système

La définition d'un modèle selon le DoD [USDoD 93] est *une représentation physique, mathématique ou logique d'un système, d'une entité, d'un phénomène, ou d'un processus.*

Souvent, par abus de langage, le terme de modèle est utilisé pour désigner un langage ou un formalisme. Par exemple, nous disons que les réseaux de Petri et UML sont des modèles. Le terme de modèle est employé ici comme une représentation d'un système réel. Nous parlons de modèles de systèmes.

Dans [REVVA 04], un modèle est *une approximation, une représentation, une idéalisation de certain aspect de la structure, du comportement, de la fonction, ou d'autre caractéristiques d'un processus, d'un concept ou d'un système du monde réel.* Un modèle est une représentation simplifiée du système réel. Cette simplification est inhérente à des approximations ou à une idéalisation (perception) du système réel. Elle peut porter sur divers aspects et donc selon différents points de vue.

La modélisation est l'activité de construction d'un modèle. Elle peut être utilisée pour représenter :

- le système en cours de développement, que nous appelons par la suite **système d'intérêt**,
- l'environnement dans lequel le système opére.

L'objectif d'une modélisation est d'obtenir des informations à propos du système avant que des ressources conséquentes ne soient engagées dans la conception, le développement, la construction ou le test du système. Ainsi, le développement et l'utilisation d'un modèle ne doivent pas consommer de ressources qui excèdent la valeur de l'information obtenue (équilibre coût/bénéfice). Si une solution analytique du modèle n'est pas faisable due aux limitations du formalisme de modélisation ou à la complexité du modèle, on peut exécuter le modèle dans le temps, et ainsi tirer des conclusions à propos du système réel en observant le comportement dynamique du modèle. Dans ce contexte, simulation signifie expérimentation avec un modèle. Le DoD définit la simulation comme *une méthode pour exécuter un modèle dans le temps. Une simulation est aussi une technique pour le test, l'analyse ou l'entrainement dans laquelle des systèmes réels ou des modèles qui représentent ces systèmes réels sont utilisés.* Dans [REVVA 04], les auteurs définissent une simulation comme

un modèle qui se comporte comme un système donné lorsqu'il est soumis à un ensemble de stimuli.

La Modélisation et la Simulation (M&S) est utilisée à toutes les phases du cycle de vie du système :

- Stratégies d'aquisition : comparer des concepts/solutions candidates pour des problèmes ou des missions émergentes et déterminer quels concepts et options offrent le meilleur retour sur investissement (Cost and Operational Effectiveness Analyses (COEA) et Acquisition Master Plans).
- Développement des concepts : permet la consolidation des concepts avant l'engagement du développement du système. Les questions posées à ce niveau s'adressent à la satisfaction des exigences, l'amélioration des performances et de la fiabilité et la réduction des coûts.
- Prédiction de la performance : aide à l'identification des exigences qui ne sont pas satisfaites ou qui seront difficiles à satisfaire et le risque associé à la non satisfaction d'une exigence.
- **Conception** : aide à la définition et au développement du système. La M&S est utilisée ici pour prédire et valider le comportement du système et ainsi faciliter le passage d'une étape du cycle du développement à une autre jusqu'à la production.
- **Validation des tests** : permet de planifier des tests, d'identifier des problèmes à priori et de minimiser les surprises. Le gain se traduit ici principalement par l'optimisation des moyens de test (réduction des coûts et du temps d'occupation du moyen).
- Support opérationnel : aide au diagnostic, à la réparation et à l'investigation d'améliorations et de mises à jour conçues après la mise en service du système.
- Entrainement :

Nous verrons que le périmètre de notre étude concerne l'évaluation des simulations utilisées pour les phases de conception et de validation des tests.

Chez Airbus Industrie, il y a une tendance sans cesse croissante à utiliser la M&S dans le cycle de vie des avions. En fait, il s'agit d'une évolution naturelle et souhaitable d'une large utilisation de la M&S au cours du développement des nouveaux appareils. Le département *Simulation* d'Airbus soulève la question de l'efficacité des résultats de simulation utilisée pour le développement des systèmes

embarqués. De telles questions sont motivées par le souhait :

1. d'améliorer le niveau de confiance qui peut être accordé à l'utilisation d'une simulation pour la V&V des systèmes,
2. de réduire les tests sur les moyens physiques et d'optimiser l'utilisation de la simulation en choisissant le meilleur moyen en fonction des objectifs de tests,
3. d'identifier une façon cohérente et continue d'améliorer les produits de simulation en maîtrisant leur qualité et leur coût.

1.2 Objectif de l'étude

Nous appelons **produit de simulation** l'ensemble constitué : des objectifs, i.e. un plan de V&V d'un système d'intérêt ; un modèle du système d'intérêt ; des modèles d'environnements, i.e. les modèles des systèmes en interaction avec le système d'intérêt ; des scénarios d'expérimentation, qui constituent ce qu'on appelle le cadre expérimental ; un modèle exécutable, i.e. un programme informatique ; une plateforme d'exécution, i.e. un ordinateur (ou un ensemble d'ordinateurs distribués sur un réseau) et un système d'opération. Une définition détaillée puis formelle de ces éléments seront données par la suite. De la même manière que l'on réalise des activités de V&V d'un système, on peut se poser la question de la correction et de la validité d'un produit de M&S utilisé dans le cadre de la V&V d'un système. Nous verrons que la V&V d'un produit de M&S implique des considérations un peu différentes de celles proposées par l'Ingénierie Système.

En considérant une utilisation toujours grandissante de la M&S durant le développement de nouveaux programmes avions, l'objectif de cette étude est de statuer sur un niveau formel de la validité d'un produit de M&S et d'établir l'effort nécessaire pour atteindre un niveau de validité requis (figure 1.2). Le niveau de validité doit être vu comme tous les critères de correction, fidélité, représentativité, exactitude, précision... L'effort doit être vu comme toutes les actions liées à la diminution des risques, du coût et de temps de cycle de développement.

Dans ce contexte, les défis posés par cette problématique sont :

1. Définir des exigences de validité et de réalisme pour une simulation avant son développement,

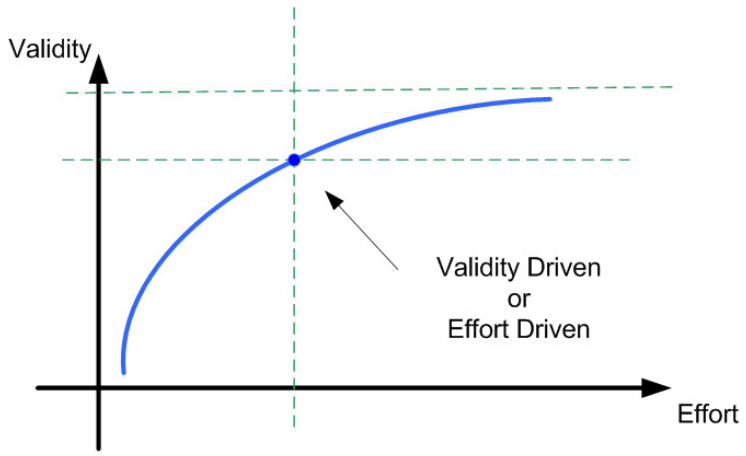

FIGURE 1.2 – Validité vs effort pour établir le moyen de simulation le plus adéquat

2. D'évaluer ce degré de réalisme et de validité tout le long du développement du produit de simulation,
3. Diminuer les risques liés à l'utilisation de la simulation et de gérer les efforts pour atteindre le degré de validité,
4. D'assurer une traçabilité entre un système et le produit de simulation qui le représente face aux exigences de validité.

La difficulté essentielle dans cette étude est d'identifier et de spécifier son périmètre global. Il s'avère que le problème de V&V des produits de simulation, n'est pas un domaine scientifique très mature. La première tâche est donc de spécifier tous les concepts liés à la correction et à la validité dans le domaine de simulation des systèmes et d'identifier ceux qui existent chez Airbus Industrie. À partir des études préliminaires que nous avons conduites, nous avons pu voir que l'évaluation de la validité d'une simulation implique les divers objets d'un produit de M&S (modèle executable, objectifs, cadre expérimental...), et plus particulièrement la relation entre ces objets et que ces objets existent chez Airbus Industrie. Lorsque les objets d'intérêt pour l'évaluation ont été identifiés, nous avons également identifié les qualités requises d'un produit de M&S. Comme l'illustre la figure 1.3, divers acteurs contribuent à l'utilisation des simulations et chacun a une attente différente de la simulation. Nous avons relevé trois critères de haut niveau qui découlent d'une hiérarchisation de la V&V de produits de simulation en niveaux de perception de

qualité de la simulation de la part de ses acteurs [Albert 07a] :

1. Au niveau développeur de la simulation : correction du moyen de simulation par rapport à sa spécification,
2. Au niveau utilisateur de la simulation : validité de l'approche par rapport au problème,
3. Au niveau projet : efficacité de la résolution du problème par l'approche proposée par rapport à l'organisation.

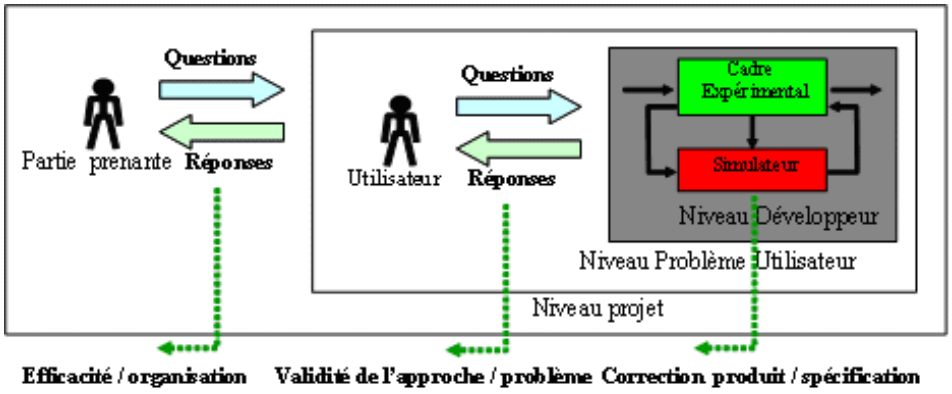

FIGURE 1.3 – Les acteurs de la simulation et leurs attentes

1.2.1 Correction d'une simulation

La correction d'un produit de simulation est le résultat de l'activité de vérification. Ce critère statue sur la conformité du simulateur par rapport à ses spécifications sur la preuve que le simulateur est dégagé de toutes erreurs d'exécution (free of run-time error). Ce critère s'adresse aux modèles et à la plateforme d'exécution.

Au niveau des modèles, il s'agit de s'assurer que ceux-ci satisfont les contraintes héritées de la physique. Par exemple, s'assurer que la masse est toujours positive ou nulle. Ces règles sont plus ou moins simples et plus ou moins connues. Elles méritent d'être suscitées par les experts du domaine afin que les développeurs du modèle puissent les vérifier plus systématiquement.

Au niveau de la plateforme d'exécution, il s'agit de s'assurer que celle-ci n'introduit pas de biais dans l'observation des comportements qui sont encodés par les

modèles. Cela signifie que la plateforme d'exécution doit respecter la sémantique opérationnelle des modèles. Par exemple, l'exécution d'un modèle peut être distribuée sur diverses machines. Ces machines ne partagent pas la même horloge alors que le modèle fut conçu autour d'une horloge globale. Il s'agit donc de vérifier que les contraintes de temps entre les éléments du modèle sont préservées par l'implémentation. Cela signifie également que les calculs numériques doivent être réalisés avec une certaine exactitude en fonction des méthodes de résolution numérique utilisées dans le modèle. Enfin, cela signifie que la plateforme d'exécution doit être suffisamment robuste (pas de "crash" de division par zéro, dépassements,...).

1.2.2 Validité d'une simulation

La validité d'un produit de simulation est le résultat de l'activité de validation. Ce critère statue sur la capacité du simulateur et du cadre expérimental à répondre aux questions posées par l'utilisateur pour résoudre ses problèmes. Une attention particulière est portée sur la notion de fidélité qui permet de démontrer que *le modèle et son comportement sont une représentation adequate du système réel et de son comportement à l'égard d'un objectif d'utilisation d'un produit de M&S* [Brade 00].

Le premier aspect de ce critère est de déterminer si le simulateur (modèle exécutable et plateforme d'exécution) permet de réaliser les tests qui sont spécifiés pour atteindre un objectif. Nous parlons communément de cadre expérimental [Zeigler 00b]. Au regard des requêtes du cadre expérimental, cela revient à déterminer si le simulateur offre : (1) les moyens de contrôlabilité permettant de stimuler les modèles et (2) les moyens d'observabilité permettant l'observation des effets des stimuli d'entrée sur le système. Les problèmes de contrôlabilité et d'observabilité consistent à fournir au développeur des modèles d'environnement les spécifications des fonctions ou des cas d'utilisation requis par un cadre expérimental. Ces spécifications définissent les conditions d'interfaçage et les abstractions autorisées. La première difficulté est bien souvent inhérente aux spécifications trop vagues, notamment lorsque le cadre expérimental est peu connu. Nos études préliminaires ont montré que l'utilisateur de la simulation n'est pas capable de définir le cadre expérimental ; il préfèrera se replier sur la formulation "comme dans l'avion". Nous voyons bien là la nécessité d'établir un guide méthodologique ou un cadre de travail facilitant la spécification d'un cadre expérimental. La seconde difficulté est qu'un modèle

d'environnement peut être utilisé dans différentes simulations avec des cadres expérimentaux différents. Par exemple, un même modèle peut être utilisé pour valider deux systèmes d'intérêt différents. Les conditions d'interfaçage ne peuvent pas être constamment imposées par chaque cadre expérimental spécifique. Donc, les développeurs des modèles d'environnement doivent spécifier le domaine d'usage de leur modèle. Le domaine d'usage décrit les capacités, les limitations d'un modèle de simulation. Il y a encore une fois un besoin de clarifier et d'homogénéiser cette notion de domaine d'usage d'un modèle.

Le deuxième aspect concerne la pertinence du choix du modèle au regard des objectifs à atteindre. Considérons que le simulateur permet la stimulation et l'observation plannifié par le cadre expérimental tel que nous l'avons expliqué ci-dessus.

Certains modèles ne décrivent qu'une connaissance partielle du système à un moment donné de son développement et d'autres sont délibéremment simplifiés pour satisfaire plus efficacement les objectifs d'utilisation. Pour chaque modèle simplifié, il faut vérifier la validité de l'abstraction : la simulation d'un modèle abstrait et la simulation d'un modèle de référence ou l'expérimentation du système réel conduisent au même jugement pour l'objectif considéré. De plus lorsque l'on compose des modèles, il faut également vérifier si la composition des abstractions est valide. La pratique la plus commune pour gérer ce problème est de comparer les résultats de l'exécution du modèle abstrait avec ceux du modèle de référence. Ceci implique qu'il existe un modèle de référence. Il serait donc utile de caractériser les procédures d'abstraction valides indépendamment d'un modèle de référence.

Enfin, il faut s'assurer de l'efficacité du cadre expérimental à satisfaire les objectifs d'utilisation. Ceci se traduit par la vérification qu'un cadre expérimental assure une exploration suffisante du modèle. Si ce n'est pas le cas, cela peut signifier soit que les exécutions non explorées ne sont pas pertinentes et donc que le modèle est trop complexe pour l'objectif, soit que les hypothèses du cadre expérimental sont érronées.

1.2.3 Efficacité d'une simulation

L'efficacité d'une simulation est un problème othogonal à la validité et à la correction. Les mesures d'efficacité sont des critères qui permettent de qualifier l'utilisation d'un produit de simulation. Par exemple, ces critères peuvent être le temps

nécessaire au développement et à l'utilisation du produit de simulation ou le coût et la criticité de cette activité de développement pour atteindre la réussite du projet. L'analyse d'efficacité peut permettre de comparer différents produits de simulation valides pour atteindre un objectif d'utilisation, mais ne peut en aucun cas remplacer les analyses précédentes de correction et de validité.

Les mesures qui peuvent permettre de caractériser plus finement les problèmes d'efficacité sont :

- Avoir un modèle à disposition : il est préférable d'avoir un modèle simplifié rapidement plutôt que d'avoir un modèle complexe trop tard. Cependant, réutiliser un modèle détaillé déjà existant peut dans certain cas être plus satisfaisant que de développer un nouveau modèle.
- Intégrer les modèles sur une plateforme d'exécution : l'intégration de modèles trop détaillés rend délicate leur intégration.
- Conception des scénarios de simulation : si la granularité des interfaces du produit de M&S ne sont pas adaptés, la conception des scénarios est complexifiée par l'affectation de valeurs à des variables sans intérêt pour l'objectif d'utilisation.

1.2.4 Conclusion sur les critères

La description d'une qualité d'un produit de simulation en trois méta-critères et les acteurs qui y ont été associés nous ont mené à positionner notre étude essentiellement sur le problème de validité. La correction n'est pas un critère spécifique à la M&S. C'est une activité classique d'ingénierie système ou de génie logiciel. L'efficacité est souvent présentée comme un critère de consolidation de la validité pour une prise de décision argumentée sur l'utilisation d'un produit de simulation.

1.3 Démarche générale d'évaluation

Dans un grand nombre d'organisations, la documentation n'est pas considérée comme une partie de l'activité de conception mais plutôt comme une tâche additionnelle, quelque peu désagréable, qui doit être remplie pour des raisons contractuelles ou pour répondre à des normes. Une partie de l'évaluation repose sur les constituants de la simulation et leurs relations. Les aspects des constituants sont variés, ce qui

rend difficile l'évaluation de leur compatibilité. Et, parce que les produits de simulation sont des produits complexes, ils nécessitent un effort de documentation. Si le processus de validité implique la collecte d'information afin d'améliorer la documentation d'une simulation, il est impératif de réutiliser et d'adapter les documents ou formalismes existant et d'automatiser autant que possible le processus d'extraction d'information à partir des documents existants, plutôt que d'allourdir le processus de développement et introduire de nouveaux documents ou formalismes spécifiques à l'évaluation des produits de simulation. Puisque notre étude s'inscrit dans le cadre de l'ingénierie des systèmes, le processus d'évaluation doit s'intégrer dans ce cadre, et donc se servir des documents issus des standards d'ingénierie système pour la collecte d'information et la production des documents.

Le meilleur moyen de garder un certain degré d'automatisation est de travailler avec des concepts formels. Il est nécessaire de définir des formulaires de documentation, basés sur des formalismes pour décrire les constituants de M&S. Ces formalismes joueront le rôle de médiateur de la mise en correspondance des constituants supportant ainsi le processus de V&V de la M&S. Nous proposons la construction de formats standards de description des constituants d'une M&S et des critères d'acceptation afin de clairement et structurellement spécifier les concepts liés à l'évaluation et sous-jacents à chaque constituant. Il s'agit ensuite de garnir ces fiches standards par extraction et/ou collecte d'information à partir de documents existants et de l'analyse des acteurs de la M&S.

La théorie de la M&S [Zeigler 00b] présente des concepts avancés sur les constituants d'une simulation et leurs relations. Elle propose notamment une spécification formelle du cadre expérimental. Elle permet également de fournir une description d'un modèle tout en se demandant si ce modèle reproduit correctement le comportement dynamique du système selon un point de vue donné offrant ainsi un cadre aux problèmes d'abstractions et de validité d'une abstraction. Ce cadre formel est supporté par le formalisme DEVS (Discrete Event System Specification) utilisé comme langage de spécification indépendant d'une plateforme d'exécution. Ainsi, nous proposons d'utiliser la théorie de la M&S pour la formalisation des constituants. Les critères de compatibilité seront établis sur les artefacts de ce formalisme.

Chapitre 2

Caractérisation du problème

Sommaire

L'objectif de ce chapitre est de caractériser le problème et de préciser l'objectif de la contribution d'un point de vue scientifique et "standard". Le problème de l'évaluation d'une application de simulation est un problème complexe qui est encore discuté dans les différentes communautés de M&S. Nous avons présenté la problématique dans le chapitre précédent. À partir d'une présentation des travaux issus de différentes communautés nous tentons de définir une notion de validité et de fixer ainsi le périmètre de notre contribution. Dans un premier temps, nous développons les définitions que nous avons adoptées pour les concepts de système, de modèle, de simulation, d'abstraction et de validation, tout en nous appuyant sur les étapes de construction d'un produit de M&S. Ensuite, une description des concepts techniques de Vérification, Validation et Accréditation (VV&A) est présentée afin

de positionner notre approche et de s'appuyer sur une terminologie standard. Nous consacrons une section particulière à la théorie de la M&S qui offre un cadre formel d'élaboration de modèle et de simulation. La troisième section présente le contexte industriel et projete les concepts introduits précédemment sur l'utilisation de la M&S dans le cadre du développement des systèmes embarqués. Enfin, nous proposons une approche pour l'évaluation de la validité d'un produit de M&S.

2.1 État de l'art en matière de V&V de M&S

2.1.1 M&S : Une abstraction de la réalité

La problématique du sujet traité dans cette étude prend directement sa source dans le concept même de *modélisation*. En général, un *modèle* est défini comme étant une représentation (parmi d'autres) simplifiée de la réalité. Cette représentation est considérée comme une abstraction de la réalité. La notion de modèle que nous manipulons dans la communauté de M&S est tout à fait comparable à la notion de *théorie* telle qu'elle est définie dans la méthode scientifique [Godfrey-Smith 03]. Une théorie est une explication de la manière dont un ensemble de faits réels interagissent entre eux. Elle est capable de prédire les observations de ces mêmes faits et elle est capable d'être testée au travers des expérimentations ou autrement d'être falsifiée à travers des observations empiriques.

Donc un modèle prend sa source et vit au travers des évidences expérimentales. À partir d'observations précédentes, et donc de par son expérience, le scientifique établit une théorie ou enrichit/corrige une théorie déjà existante, dans le but ultime de généraliser/universaliser un ensemble de suppositions. Pour construire un modèle, les ingénieurs formulent des hypothèses spécifiques comme des représentations d'un système réel, et conçoivent des expériences qui testent l'exactitude de ces hypothèses. Ces étapes sont répétées de manière à approcher le plus possible la réalité. L'abstraction est alors directement liée à la perception qu'a le modélisateur de la réalité, et de sa connaissance du monde réel. Ainsi, une modèlisation est un ensemble de théories formulées (ou raffinées) à partir d'observations du monde réel et d'hypothèses sur ce monde réel. La simulation est l'expérimentation du modèle dans le but de valider ou réfuter les théories formulées.

2.1.1.1 Le point de vue de F.K. Frantz

Le concept d'abstraction de modèle est décrit par F.K. Frantz [Frantz 95] comme une méthode permettant de réduire la complexité d'un modèle tout en conservant la validité des résultats de simulation à l'égard des questions auxquelles la simulation est censée apporter des réponses. Un ensemble de techniques a été développé par les communautés de M&S et de l'intelligence artificielle pour simplifier un modèle. L'auteur propose une taxonomie de ces techniques que nous présentons dans le chapitre suivant. Donc, s'il existe des hypothèses liées à la perception et la connaissance qu'un développeur de modèle a de la réalité, il existe aussi des hypothèses qui permettent d'abstraire volontairement ce modèle afin de faciliter sa compréhension et diminuer le coût de son implémentation, pour un objectif d'utilisation donné. Pour l'auteur, *une abstraction est valide si elle préserve la validité des résultats de la simulation, qui sont dépendants des questions posées*. F.K. Frantz décrit le processus de construction d'une simulation sous forme de deux transformations successives (figure 2.1) :

- le passage du monde réel vers un modèle conceptuel par un mécanisme d'abstraction,
- le passage du modèle conceptuel vers un modèle de simulation par un mécanisme d'implémentation.

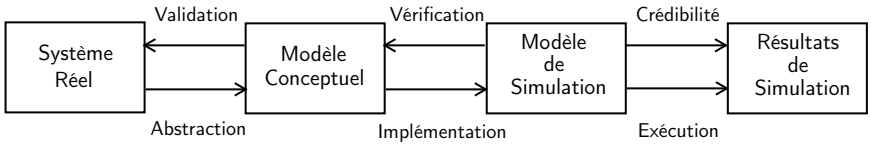

FIGURE 2.1 – Processus de M&S

Le modèle conceptuel est vu comme la façon de représenter le système réel. C'est à travers le modèle conceptuel que sont définis les facteurs qui influencent le comportement du système, les comportements du système qui doivent être inclus dans le modèle, et la représentation de ces comportements en partant du principe que :

- un modèle est toujours construit afin d'être utilisé pour répondre à certaines questions à propos du monde réel,

– les données disponibles pour décrire le monde réel sont variées,
– les ressources de développement et d'implémentation disponibles pour construire, valider et utiliser le modéle doivent être prises en compte dans la modélisation.

La construction du modèle conceptuel est une abstraction puisque le modèle conceptuel est une représentation simplifiée du système du monde réel. Le processus qui détermine si le modèle conceptuel est une représentation exacte du système du monde réel est appelé validation. L'auteur souligne le fait que si l'abstraction est une mise en correspondance entre le modèle conceptuel et le système du monde réel, elle peut tout aussi bien mettre en correspondance deux modèles conceptuels différents. Le modèle conceptuel est ensuite transformé en un modèle exécutable, appelé modèle de simulation, caractérisé par un ensemble d'instructions interprétables par une plateforme d'exécution. La vérification consiste à s'assurer que le modèle de simulation est une description correcte du modèle conceptuel. Notons que le passage du modèle conceptuel vers le modèle de simulation peut être l'objet de plusieurs transformations. Enfin les résultats de simulation sont obtenus par exécution du modèle de simulation. L'accréditation est l'activité qui permet de déterminer si les résultats de simulation doivent être utilisés par rapport à l'impact de cette utilisation (i.e. conséquences indésirables associées à des décisions erronées).

2.1.1.2 Le point de vue de R.G. Sargent

La figure 2.2 élaborée par R.G. Sargent [Sargent 05] illustre la relation entre les constituants d'une application de M&S. Elle inclut également les activités majeures de V&V et les relations entre le monde réel et le monde de simulation.

Le monde réel est constitué du système réel (problem entity) qui devient, par expérimentation une source de données observables (system data). Par un mécanisme d'abstraction sur le système et à partir d'hypothèses sur les données observables, un premier pas est mis dans le monde de simulation (system theories). A partir d'objectifs d'utilisation et des théories du système, un modèle conceptuel est construit. Puis par des étapes successives de spécification et d'implémentation un modèle de simulation est produit. Par expérimentation/utilisation du modèle de simulation, des résultats de simulation sont produits. Conformément aux définitions données en section suivante, le passage d'une étape vers une autre est évalué par des activités de vérification. La validation a trois aspects : la validation du modèle conceptuel, la

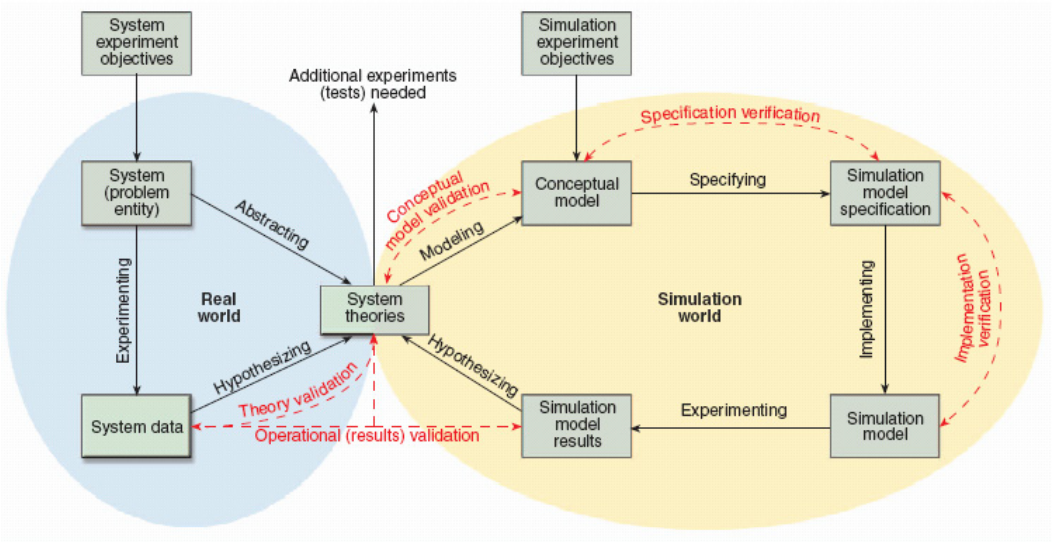

FIGURE 2.2 – Relation entre monde réel et monde de simulation et V&V

validation de la théorie et la validation opérationnelle.

La validation du modèle conceptuel est toujours liée à un objectif d'utilisation. Elle consiste en une évaluation a priori de la "validité" du modèle conceptuel.

La validation opérationnelle est réalisée par comparaison entre les résultats de simulation et les données d'expérimentation du système réel. Elle n'est alors possible que lorsque le système réel est disponible. La technique qui consiste à comparer les données d'expérimentation du système réel aux résultats de simulation est communément appelée "face validation" [DMSO 01].

La validation de la théorie consiste à valider ou réfuter un ensemble d'hypothèses à travers une expérimentation du système réel. C'est le principe de la méthode scientifique. De nombreux travaux plus philosophiques abordent ce problème comme ceux de Karl Popper. Des méthodes comme la méthode hypothético-déductive théorisée par Roger Bacon visent à déterminer la validité d'hypothèses formulées. Cet aspect de la validation est extrêmement important mais il est en dehors du périmètre de notre étude car comme la validation opérationnelle, la validation de la théorie se fait par rapport aux données réelles. Nous considérons tout au long de ce mémoire la validation du modèle conceptuel.

2.1.2 Vérification, Validation et Accréditation d'une M&S

Alors que le monde de l'ingénierie logiciel devenait une discipline à part entière, le besoin d'avoir des processus formels, cohérents, traçables de production de logiciel a été identifié pour des raisons évidentes de diminution des coûts et de détection d'erreurs et d'incohérences au plus tôt. Des modèles de référence et de bonnes pratiques ont alors vu le jour comme le "Capability Maturity Model Integration" (CMMI) [SEI 06], l'EIA632 [EIA 99] et l'ISO 15288 [ISO 03]. L'émergence d'UML [Rumbaugh 99] a significativement accéléré l'apparition de tels processus. Le même besoin s'est fait ressentir en M&S. La VV&A, comparable à ces standards internationaux dans le domaine de la simulation, a contribué à une évolution allant dans le même sens.

Rappelons que la V&V est un ensemble de techniques et d'actions appliquées à un produit et à tous les produits intermédiaires issus du cycle de développement afin de s'assurer qu'ils sont conformes à un ensemble de dispositions pré-établies. La validation système consiste à apporter la preuve que le système fonctionne comme attendu par le client dans son environnement opérationnel. La vérification consiste à s'assurer que le système est conforme à ses spécifications. La V&V a été très favorablement accueillie par les communautés de M&S aussi bien qu'en ingénierie système. Mais, dans le contexte présenté dans la section précédente, le besoin est quelque peu différent. L'objectif étant de construire un niveau de confiance dans une application de M&S, il faut que ces techniques et actions permettent de définir si une simulation peut être utilisée dans une situation donnée et que sa "credibilité" soit établie en évaluant son degré de "représentativité" par rapport au système réel pour un objectif d'utilisation donné. La décision d'utiliser une application de M&S dépendra donc de son domaine d'emploi, de sa correction, de sa précision, de son exactitude, et bien sûr, de son utilisabilité dans un contexte donné. La vérification, validation et accréditation d'une application de M&S sont trois processus distincts, mais non indépendants, pour collecter des évidences permettant d'évaluer de tels critères de "qualité" d'une application de M&S.

Il y a une variété de définitions formelles pour les termes VV&A [Brade 00] [ITOP 04] [REVVA 04]. En général, le sens qui leur est attribué est [USDoD 93] :

– Vérification : le processus permettant de déterminer si l'implémentation d'un

modèle et les données qui lui sont associées sont conformes à la description conceptuel du développeur et à ses spécifications.

– Validation : le processus permettant de déterminer si un modèle et les données qui lui sont associées fournissent une représentation exacte du monde réel dans un contexte d'utilisation donné du modèle.
– Accréditation : la certification officielle qu'un modèle, une simulation, une fédération de modèles et de simulations et les données qui leur sont associées sont acceptables pour un objectif donné.

Ces définitions sont aisément retenues grâce aux questions suivantes : Vérification - Ai-je construit le modèle correctement ?, validation - Ai-je construit le bon modèle ? et accréditation - Puis-je utiliser le modèle ?

2.1.2.1 Vérification

Le sens attribué au processus de vérification tel qu'il est défini par le DoD est le même que celui utilisé dans l'ingénierie système et l'ingénierie du logiciel. Ce processus vise à assurer que le produit final est conforme à sa spécification. Plus généralement, la vérification s'applique à chaque produit intermédiaire d'une phase spécifique du cycle de vie du produit. Ce dernier doit être conforme au référentiel de la phase précédente. La définition de D. Brade va dans ce sens et associe explicitement l'activité de vérification à l'examination de la *correction* d'un modèle [Brade 00] : *la vérification d'un modèle est le processus qui cherche à démontrer que le modèle est correctement représenté et a été transformé correctement d'une forme de représentation à une autre, selon toutes les règles de transformation et de représentation, les exigences, et les contraintes.* L'auteur décompose la correction en deux sous-critères, la *cohérence* - modèle correctement décrit sous toutes ses formes de représentation et la *complétude* - tout ce qui est décrit dans une forme de représentation l'est aussi celle issue de la transformation. Dans [REVVA 04] la vérification est *le processus, utilisé pour construire, sous un ensemble de contraintes de temps et de coûts une preuve justifiée de la correction d'un modèle.* Les développeurs d'une simulation se tournent typiquement vers les méthodologies de l'ingénierie logiciel pour guider la vérification d'une simulation. Ces méthodologies impliquent divers tests logiciel, tel que l'analyse logique, l'analyse des interfaces et l'analyse des contraintes des programmes [Whitner 89] [Oberkampf 04].

2.1.2.2 Validation

Alors que la validation définie par les communautés d'ingénierie système et du logiciel vise à s'assurer que le produit répond bien au besoin de l'utilisateur, la tradition des communautés de M&S associe la validation à l'activité qui cherche à assurer que le produit (de M&S) permettra d'atteindre l'objectif d'utilisation **et** que ce produit a un comportement suffisamment proche du système qu'il représente pour cet objectif d'utilisation. Ainsi, en plus du traditionnel concept de "satisfaction du besoin utilisateur", une notion de *fidélité* [DMSO 00a] a été introduite par la communauté de M&S. Cette notion de fidélité est largement discutée au sein de la communauté. Dans [Brade 00] l'auteur parle de "suitability", terme ensuite décomposé en "capability", "fidelity" et "accuracy" et définit la validation comme *le processus de démontrer que le modèle et son comportement sont une représentation adequate ("suitable") du système réel et de son comportement selon un objectif d'utilisation de la simulation*. Le *Simulation Interoperability Organisation Office* [DMSO 00b] propose une liste de caractéristiques d'une simulation liées à la fidélité : accuracy, capacity, error, fitness, precision, resolution, sensitivity, tolerance, validity. Les diverses techniques de validation de M&S sont listées dans [DMSO 00c], [Whitner 89].

2.1.2.3 Accréditation

L'accréditation quant à elle est une activité spécifique à la M&S et hautement normative. En plus des résultats de V&V produits par les activités précédentes, elle prend en compte des aspects de gestion de projet comme le coût et le risque associés à l'utilisation du produit de M&S, pour prendre la décision finale et officielle que ce produit peut être utilisé avec confiance et crédibilité.

2.1.3 Le modèle conceptuel

Le modèle conceptuel est le premier élément de conception de l'application de M&S. Il est, pour F.K. Frantz, directement issu du mécanisme d'abstraction de la réalité et prend en compte les objectifs d'utilisation de la simulation et les hypothèses de modélisation. Si nous nous reportons au paradigme développé par R.G. Sargent, il prend également en compte les théories du systèmes (algorithmes, équations...). Dans de nombreux travaux, le modèle conceptuel est vu comme le pont, l'élément

de communication entre l'utilisateur et le développeur de la simulation. Il est le point d'entrée pour le développement du produit et doit être accrédité par chacun, utilisateur et développeur. Son développement et sa validation sont donc une étape très importante du développement de l'application de M&S. L'évaluation du modèle conceptuel porte sur l'adéquation entre le besoin exprimé par l'utilisateur et les possibilités du développeur de répondre à ce besoin, compte tenu des ressources de développement (disponibilité des données) et d'implémentation (contraintes liées à la plateforme d'exécution) dont il est tributaire. Nous parlerons communément d'exigences de M&S. Les exigences de M&S sont des exigences spécifiques exprimées par l'utilisateur. Ces exigences spécifiques ne sont pas des exigences systèmes mais des exigences plus détaillées, dérivées des exigences systèmes : les fonctionnalités, les conditions d'utilisation, les contraintes et hypothèses qui caractérisent l'objectif d'utilisation.

2.1.3.1 Alors en quoi consiste le modèle conceptuel ?

Le rôle du modèle conceptuel dans le développement d'une M&S tel qu'il est décrit dans le paradigme de R.G. Sargent ou F.K. Frantz est accepté par les standards de VV&A du DMSO et par la communauté y compris par D.K. Pace dont le travail traite du contenu du modèle conceptuel [Pace 99] d'une application de M&S.

Selon la figure 2.3, le modèle conceptuel transforme les exigences de M&S en une spécification du produit de M&S. D.K. Pace propose un modèle conceptuel en trois parties :

- le contexte de la M&S,
- le concept de la M&S,
- les éléments de la M&S.

Le contexte de la M&S définit le domaine du problème auquel l'utilisateur est confronté pour atteindre ses objectifs d'utilisation. Il décrit la nature du problème et l'ensemble des entités qui sont concernées par l'application de M&S. Suivant le niveau où l'on se place et le domaine, ces entités peuvent être des fonctionnalités, des algorithmes, des composants physiques ou logiques, des processus. Les exigences liées au niveau de fidélité nécessaire pour produire les résultats attendus doivent aussi être décrits dans le contexte de la simulation.

Le concept de la M&S permet de décrire comment les exigences de M&S décrites

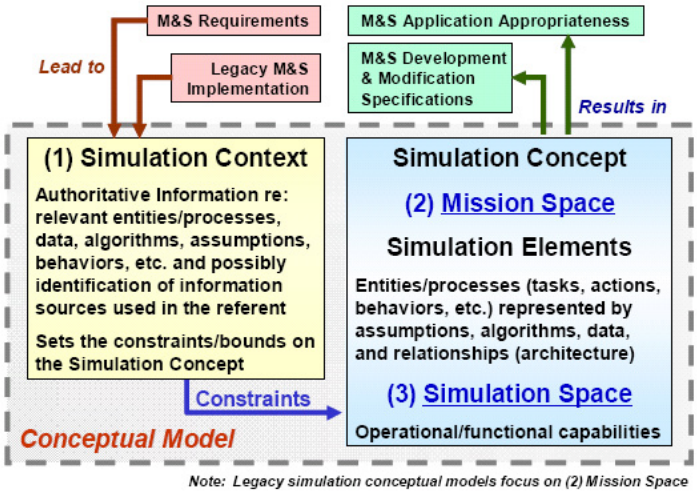

FIGURE 2.3 – Les constituents du modèle conceptuel

dans le contexte de la simulation sont transformées en spécification détaillée de la simulation. Il permet également de décrire comment procéder pour atteindre les objectifs d'utilisation et de justifier en quoi une simulation peut permettre d'atteindre ces objectifs. Par exemple, le concept de la simulation spécifie tout les éléments nécessaires pour acheminer les données d'entrée vers le système d'intérêt et par quoi ou comment ses données sont générées (un senseur, une équation, un ensemble de valeurs dans le temps, le taux de raffraichissement des données...).

Le concept de la simulation est séparé en deux parties :

– L'espace de la mission qui concerne les aspects représentationnels et qui inclut les éléments de la simulation.

– L'espace de la simulation qui concerne les aspects de contrôle de la simulation.

Les éléments de la M&S sont la collection d'informations décrivant les concepts pour une entité, un processus, qui identifient et définissent les états possibles, les tâches, les événements, les comportements, les paramêtres, les caractéristiques et les attributs des éléments décrits dans l'espace de la mission.

Les informations typiques d'un élément de simulation sont [USDoD 93] :

– entité, processus ou définition d'une collection,

– hypothèses à propos des limitations et des contraintes sur l'élément,

– les algorithmes et les pédigrées d'un algorithme,

– les données et leur histoire,

– les relations avec d'autres éléments de la simulation,
– les interactions avec d'autres éléments de la simulation,

2.1.3.2 Les exigences de M&S

Les exigences de M&S sont définies par le DoD [DMSO 04] comme *une collection de fonctionnalités, de représentations, de conditions, de contraintes et d'hypothèses qui définissent les besoins d'utilisation d'une simulation*. Les exigences de M&S sont un élément prépondérant du modèle conceptuel. C'est pour satisfaire ces exigences qu'une application de M&S est construite. Ces exigences spécifient l'ensemble des capacités souhaitées par l'utilisateur pour satisfaire son objectif d'utilisation. Les exigences dites de représentation décrivent les propriétés, les comportements des "choses" du monde réel que l'application de M&S doit avoir. Elles décrivent aussi les propriétés souhaitées en termes de fidélité. Ces exigences définissent la résolution, l'exactitude et la confiance en cette exactitude necessaires à chaque élément afin de satisfaire le besoin de l'utilisateur. D'autres exigences dites d'implémentation concernent des préoccupations sur l'environnement d'exécution (e.g. vitesse d'exécution des modèles).

L'analyse des exigences de M&S mène à l'établissement de critères et de métriques, établies par l'utilisateur, permettant d'établir un niveau de fidélité attendu. Ces critéres sont appelés critères de validation ou d'acceptabilité. Ils doivent être non ambigüs et, bien entendu, évaluables. La valeur des critères peut être collectée à partir de données de test, de résultats de validation, ou plus qualitativement par le jugement de l'expert (Subject Matter Expert) [Pace 02]. C'est le niveau d'exigence dans l'évaluation des critères qui déterminera l'effort nécessaire et le risque associé à l'accréditation de l'application de M&S. Un critère de niveau trop élevé augmente le coût et l'effort de validation. Un critère de niveau trop bas augmente le risque associé à l'utilisation de l'application de M&S. Ainsi, s'assurer que les valeurs des critères sont convenablement établies est la considération première de l'effort de V&V à fournir.

2.2 Cadre méthodologique d'élaboration de modèles et de simulations

2.2.1 Théorie de la M&S

L'élaboration d'un modèle conceptuel se fait à travers des abstractions. Les abstractions utilisées sont variées et elles dépendent directement du point de vue sous lequel le système du monde réel est étudié. La validité d'un modèle conceptuel est alors évaluée selon un point de vue donné. B.P. Zeigler [Zeigler 00b] a proposé, dans la théorie de M&S, un moyen de fournir une description d'un modèle tout en se demandant si ce modèle reproduit correctement le comportement dynamique du système selon un point de vue donné.

Le principe fondamental du processus de M&S décrit par B.P. Zeigler est la séparation du modèle et du simulateur. Le même principe est préconisé par l'approche MDA (Model Driven Architecture) proposée et soutenue par l'OMG [OMG 01]. Les entités de base du processus de M&S sont le système, le modèle et le simulateur (figure 2.4.

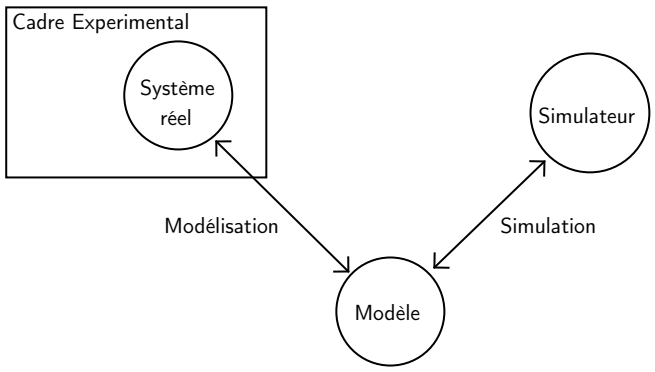

FIGURE 2.4 – Processus de M&S et ces entités

Le système est l'élément réel ou virtuel qui est utilisé comme une source de données observables et assujetti à une modélisation. Le modèle, également appelé substitut du système, est une représentation du système. C'est typiquement un ensemble d'instructions, de régles, d'équations ou de contraintes responsables de son comportement. Le simulateur est un système informatique permettant d'exécuter le modèle et de générer son comportement à partir des instructions du modèle et des

entrées qui sont injectées. Cet ensemble est réorganisé de manière à faire intervenir le concept de cadre expérimental. Le cadre expérimental est une spécification des conditions sous lesquelles un système est observé ou expérimenté. On peut le voir comme un système qui interagit avec le système d'intérêt pour obtenir les données d'intérêt sous des conditions données. C'est une formulation opérationnelle des objectifs qui motive le développement de l'application de M&S. Il peut donc y avoir plusieurs cadres expérimentaux pour un même système et un même cadre expérimental peut être appliqué à plusieurs systèmes. En effet, nous pouvons avoir divers objectifs ou points de vue ou avoir un même objectif qui motive la modélisation de différents systèmes.

Un cadre expérimental est formé de trois composants : un *générateur* qui injecte un ensemble de vecteurs d'entrée au système, un *accepteur* qui sélectionne les données d'intérêt du système en surveillant si les conditions expérimentales souhaitées sont respectées, un *transducteur* qui observe et analyse les vecteurs de sorties du système.

Rappelons que le cadre expérimental transforme les objectifs, qui permettent de focaliser le développement du modèle sur un point de vue particulier, en des conditions d'expérimentation précises. Ainsi, un modèle doit être valide pour un système dans un cadre expérimental donné. Une formulation opérationnelle des objectifs est faite par une mise en correspondance entre les *variables observées* (les entrées et les sorties du système) en des mesures qui permettent de déterminer que le système accompli effectivement une fonction donnée. Ces mesures sont appelées les *mesures des résultats*. La mise en correspondance entre variables observées et mesures des résultats est réalisée par le transducteur.

La relation entre un modèle, un système et un cadre expérimental est la modélisation. La validité d'un modèle est le fondement même de toute modélisation. La validité est définie par une valuation, le degré avec lequel un modèle représente fidèlement un système dans un cadre experimental d'intérêt. La relation entre un modèle et un simulateur est la simulation. Le concept de base de cette relation et la correction du simulateur. Un simulateur simule correctement un modèle si, étant donné un état du modèle et un vecteur d'entrée, il fournit un vecteur de sortie prédéfini (ou attendu, avec des tolérances prédéfinies). Cette relation réfère au principe de séparation des préoccupations entre la conception du modèle et son implémentation.

Hormis les deux relations de base présentées ci-dessus, B.P. Zeigler a introduit deux autres relations fondamentales pour le cadre de notre étude : la modélisation comme une simplification valide (abstraction valide dans la terminologie de F.K Frantz) et la relation d'applicabilité d'un modèle à un cadre expérimental.

Une modélisation "réussie" peut être vue comme une simplification valide. Il est nécessaire de simplifier, ou de réduire la compléxité d'un modèle, pour que celui-ci puisse être exécuté sur un simulateur vu comme une ressource de calcul limitée. Une *relation de préservation* ou *morphisme d'un système* établit une correspondance entre un système "concret" et un système "abstrait" ou simplifié. Le modèle abstrait étant le substitut du modèle concret. B.P. Zeigler utilise les notions de *base model* et *lumped model*. Le modèle concret est ici un modèle plus capable dans le sens qu'il est utilisable pour un plus grand nombre de cadres expérimentaux. Mais pour un cadre expérimental donné le modèle abstrait peut être aussi capable que le modèle concret. Il est important de retenir que pour un cadre expérimental donné, un modèle abstrait doit être aussi valide que le modèle concret.

La relation d'applicabilité détermine si un cadre expérimental peut être appliqué à un modèle. Cette relation est très importante puisqu'elle permet de dire si un objectif d'utilisation peut être atteint avec une application de M&S particulière. Il n'y a que les modèles qui peuvent mettre en oeuvre des conditions d'expérimentation requises par un cadre expérimental qui permet d'atteindre des objectifs, qui ont une chance de fournir des résultats de simulation valides. L'auteur définit aussi une relation de dérivabilité entre des cadres expérimentaux. Cette relation traduit le degré avec lequel un cadre expérimental définit des conditions plus restrictives qu'un autre (par exemple, il permet moins d'observations). La figure 2.5 ci-dessous illustre les relations de morphisme, d'applicabilité et de dérivabilité. Le cadre expérimental CE4, peu restrictif, s'applique aux modèles M1, M2 et M3. Le modèle M4 est trop abstrait pour réaliser CE4 ainsi que les cadres expérimentaux CE1, CE2, CE3 plus restrictifs que CE4. CE2 est applicable à M1. CE3 qui est moins restrictif que CE2 l'est donc aussi. Dans ce cas on peut dire que M1 est trop concret ou complexe pour le cadre expérimental donné mais pas moins valide. Aucun modèle ne permet d'accomoder CE1.

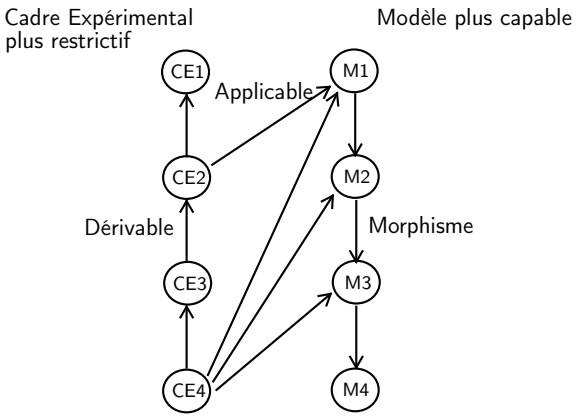

FIGURE 2.5 – Relations de morphisme, applicabilité et dérivabilité

2.3 Description du contexte industriel

Le contexte de l'étude est l'évaluation de la simulation utilisée pour le développement d'un système complexe, e.g. l'avion. Un système complexe est constitué de divers composants ayant leur propre nature (hydraulique, électrique, information...), fonction et comportement. C'est de l'interaction entre ses composants qu'émerge la fonction du système qui les aggrège, appelé le supersystème. Nous parlerons communément de système de systèmes. Maier [Maier 96] propose une discussion argumentée des caractéristiques d'un système de systèmes. Les systèmes considérés dans l'étude sont les systèmes embarqués. Un système embarqué est un calculateur spécifiquement conçu pour réaliser une ou plusieurs fonctions. Le terme de système embarqué désigne aussi bien le matériel que le logiciel utilisé. Chez Airbus Industrie, le terme d'équipement est communément utilisé pour désigner la partie matérielle du système.

Selon la classification des systèmes proposée dans [Zeigler 00b] il y a trois types de systèmes :

– Les systèmes continus : ils ont un espace d'états continu et un avancement continu du temps. Les entrées et les sorties de ces systèmes sont continues. C'est à dire qu'elles peuvent prendre une infinité de valeurs (tant sur l'axe de l'état que sur l'axe du temps). Par exemple la variation de température atmosphérique est un système continu. Traditionnellement un système continu est représenté par un ensemble d'équations différentielles et executé sur un

intégrateur numérique.

- Les systèmes à temps discret : ils opèrent sur une base de temps discrète et peuvent avoir un espace d'état soit continu soit discret. Par exemple, les lois des commandes de vol sont un système à temps discret. Dans un système à temps discret, à chaque instant t, l'état du système à l'instant $t+1$ est calculé en fonction de l'état du système à l'instant t.
- Les systèmes à évènements discrets : ils ont une base de temps continue et un espace d'état continu ou discret. Dans un système à évènements discrets, l'état suivant du système est calculé seulement lorsque un événement a lieu. Un tel événement peut être externe (il provient de l'environnement). Le temps est continu dans le sens qu'un évènement peut avoir lieu à n'importe quel instant t qui n'est pas connu à l'avance et qui ne coïncide pas forcément avec le tick d'une horloge. Le Flight Warning System et le Air Traffic Control System sont des systèmes à évenements discrets.

Selon cette classification seuls les systèmes à évenements discrets et les systèmes à temps discret sont considérés dans cette étude.

Il est important de souligner que l'exécution d'un système à évènements discrets peut avoir lieu autant avec une base de temps continue qu'avec une base de temps discrète. Dans ce dernier cas, à chaque instant la valeur du temps est comparée avec la valeur du temps d'occurence du prochain évènement. Si ces deux valeurs sont égales, le système évolue à l'état correspondant.

	Systèmes continus	Systèmes à temps discret	Systèmes à évènements discrets
Etats	Continus	Discrets ou continus	Discrets ou continus
Temps	Continu	Discret	Continu

2.3.1 Utilisation de la simulation dans le développement de l'avion

L'utilisation de la simulation est applicable à toutes les étapes du processus de développement de l'avion. Définir une stratégie de M&S est une tâche délicate compte tenu des multiples dimensions à prendre en compte :

- La dimension temps : les objectifs d'utilisation sont différents suivant l'étape

du cycle de développement.

- La dimension produit : un système peut être modélisé à différents niveaux hiérarchiques (avion, système, équipement).
- La dimension activité : parfois les activités de développement réalisées par les concepteurs systèmes nécessitent des modèles particuliers du système qu'ils ont en charge mais aussi des modéles d'autres systèmes qui sont connectés au système qui relève de leur responsabilité.
- La dimension discipline : pour un même système, différents modèles sont développés pour analyser des propriétés physiques particulières. Un avion est constitué de disciplines hétérogènes : hydraulique, mécanique, électrique, information...
- La dimension technologique : les modèles de systèmes et leurs simulations sont réalisés avec des outils logiciels variés s'exécutant sur des plateformes différentes.
- La dimension entreprise étendue : les systèmes et leur développement sont réalisés par des équipes différentes dans différents pays.
- La dimension confidentialité : certains systèmes sont développés par d'autres entreprises. Dans un soucis de confidentialité, des modèles de ces systèmes sont souvent fournis sous forme de boite noire (code compilé).

La figure 2.6 illustre la relation entre le développement de l'avion et le développement des systèmes et de leurs équipements pour le développement d'un nouvel avion. Nous appelons équipement le calculateur physique qui permet l'execution du système. Le développement des systèmes est initié à partir du moment où les concepts du produit sont figés (niveau de développement M5) et prendra fin peu avant la certification (M12). Entre ces deux jalons, les activités de développement sont variés :

- M4-M5 : consolidation des concepts, activités permettant d'atteindre une compréhension commune de l'accomplissement techniques des exigences. A l'issue de ces activités, la définition des exigences des différents systèmes de l'avion est produite (System Requirements Document).
- M5-M7 : conceptions détaillées. Finalisation de la spécification de l'avion, des définitions physiques de l'avion jusqu'aux composants élémentaires.
- M7-M9 : manufacture des parties de l'avion. Développement des équipements.

– M9-M10 : Processus d'assemblage final.
– M10-M11 : Préparation du premier vol, execution des tests fonctionnels et, tests au sol.
– M11-M12 : Accomplissement de la certification.

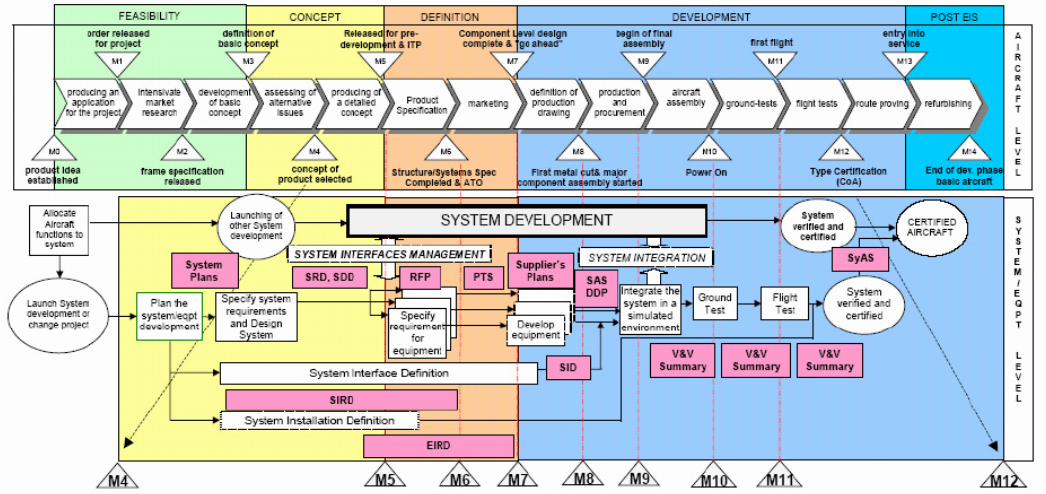

FIGURE 2.6 – Relations entre le développement de l'avion et les systèmes de l'avion

La figure 2.7 illustre les différents simulateurs utilisés le long du cycle de développement de l'avion. Nous pouvons voir que chaque simulateur est spécifique à une classe d'activité, une étape du développement et concerne différents niveaux :

– Simulateur de recherche : de niveau avion, il est applicable dans les étapes amont du développement de l'avion (M3-M5), les simulateurs de recherche permettent de tester et de comparer des solutions alternatives en termes de concepts et de technologies.
– Simulateur de bureau : de niveau système, il est applicable durant toute la phase de spécification d'un système (M5-M7). Son utilisation est principalement liée à de l'analyse du système afin d'aider au passage des exigences vers la spécification.
– Simulateur de développement : de niveau avion, il est applicable durant la phase de développement de l'avion (M7-M11). Son utilisation est dédiée à la consolidation des choix techniques, au test des interfaces cockpit, des commandes de vols et de la qualité des lois de fonctionnement et de commande.

– Banc d'intégration système : de niveau système, il est applicable après la manufacture de l'équipement réel (M8). Il est alors dédié au test unitaire d'intégration des systèmes sur équipement réel.
– Simulateur d'intégration : de niveau avion, il est applicable après le commencement de l'assemblage final (M9). Il est dédié au test d'intégration entre les systèmes et à la validation de l'architecture de l'avion avant les tests en vol.
– Simulateur d'entrainement : dédié à l'entrainement des pilotes, il doit être disponible avant l'entrée en service de l'avion (M13).

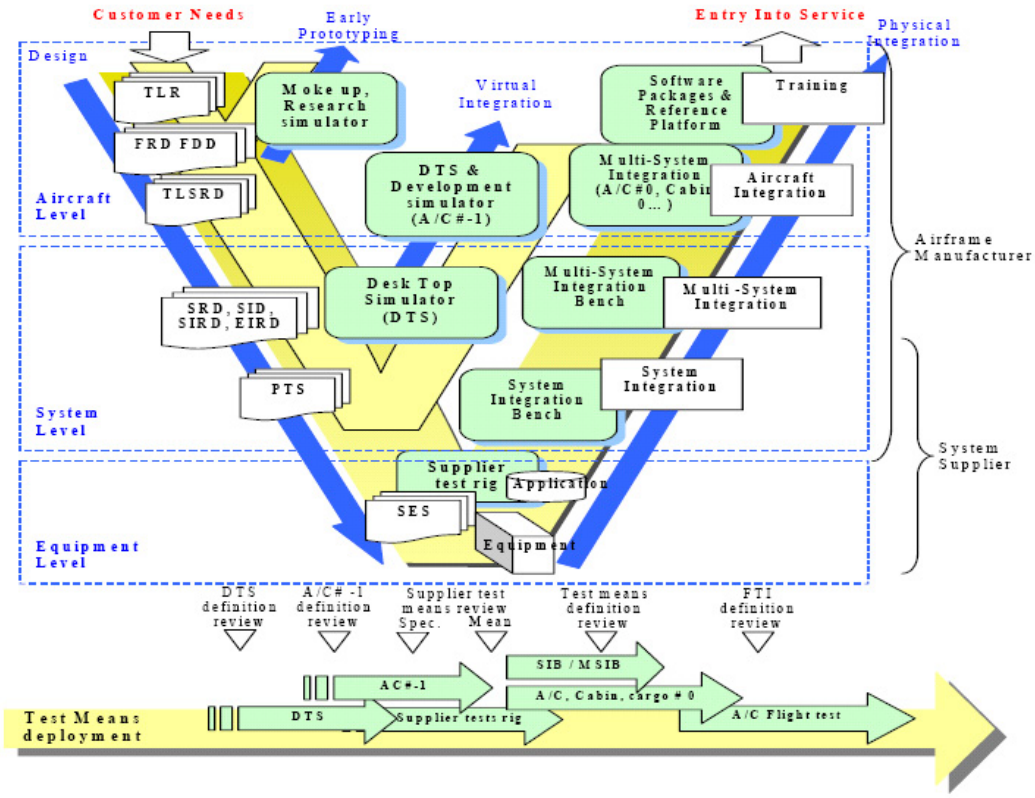

FIGURE 2.7 – Relations entre le développement de l'avion et les simulateurs

2.3.2 Périmètre de l'étude

On distingue deux grands types de simulateurs. Les simulateurs destinés à l'entrainement des pilotes dit simulateur d'entrainement, et ceux destinés à la validation du développement des systèmes de l'avion dit simulateur d'étude et que nous venons

de présenter. Le simulateur d'entrainement est disponible un peu avant l'entrée en service de l'avion, donc lorsque tous les systèmes réels ont été développés. La validité de la simulation peut alors être évaluée par recoupement des résultats obtenus par simulation et des valeurs tirées de l'avion réel. Les modèles sont mis à jour en conséquence. Bien qu'un grand nombre d'applications issues des simulateurs d'étude finissent tôt ou tard dans un simulateur d'entrainement, la validité des simulateurs d'entrainement n'entre pas dans le cadre de notre étude.

Nous choisissons également de ne considérer que les simulations utilisées pour la spécification et la conception des systèmes. Avant le niveau M5 les concepts ne sont pas consolidés et donc les exigences systèmes ne sont pas encore figées. Nous pensons que le problème d'évaluation des applications de M&S à ce niveau est encore un autre problème et que les critères et mesures que nous proposons ne peuvent pas s'y appliquer. Ainsi, nous excluons de l'étude les simulateurs de recherche.

Nous limitons donc le périmètre de notre étude entre les niveaux M5 et M12. Ceci inclut les simulateurs de bureau, les simulateurs de développement, les bancs d'intégration système et les simulateurs d'intégration. Tel que nous l'avons précisé dans le chapitre précédent (section 1.1.2), ces simulations sont utilisées pendant les phases de conception et de validation des tests. Les principaux avantages d'utilisation de la simulation dans ces phases sont :

- Au niveau système : la capacité de prédire et/ou de valider le comportement du système au plus tôt dans le cycle de développement et de vérifier l'intégration d'un système avant son intégration sur l'équipement réel.
- Au niveau avion : la capacité de vérifier l'intégration entre les systèmes avant leur intégration physique.

Ainsi la simulation permet :

- D'améliorer la robustesse et la maturité des concepts d'un système.
- D'anticiper l'intégration des systèmes.
- D'accélerer/augmenter les itérations et permettre ainsi de meilleurs choix de conception.
- De réduire le nombre de tests physiques.

En termes de types de modèles, nous ne considérons que la modèlisation utilisée pour représenter les systèmes embarqués, qu'elle soit logique ou physique. Nous excluons de l'étude tout système n'étant pas exécutable par un calculateur, comme

par exemple, l'environnement dans lequel l'avion évolue (vent, turbulence, température...). La restriction de l'étude aux systèmes à temps discret et à évènements discrets se justifie ainsi.

2.4 Formulation du problème

2.4.1 Caractérisation d'une application de M&S

Comme nous l'avons dit dans le chapitre précédent, nos études préliminaires nous ont permis de voir que les objets impliqués dans l'évaluation de M&S existent chez Airbus Industrie. Nous proposons une définition informelle de ces objets afin de caractériser un produit de M&S [Albert 07a].

2.4.1.1 Modèles du système d'intérêt

Ces modèles sont les modèles du système sous test. C'est sur ce système que porte les objectifs d'utilisation puisqu'il traduit ce que les ingénieurs veulent experimenter par simulation à un moment donné du processus de développement. Citons divers exemples : modèles Saber d'un système hydraulique ; modèles Simulink des commandes de vol ; modèles des commandes de vol, i.e. modèles Scade [ESTEREL 08] implémenter sur un banc d'intégration système. Leur rôle est de capturer un ensemble de résultats permettant d'apporter des réponses à leur activité de conception. Tant que le système d'intérêt n'a pas d'autres réalités, ces modèles sont considérés comme la définiton de référence du système.

2.4.1.2 Les objectifs d'utilisation

Une application de M&S est utilisée pour atteindre des objectifs d'utilisation. Dans le contexte de l'étude une application de M&S est utilisée pour assister les activités de V&V des systèmes avion durant le développement de l'avion. Donc dans ce cas particulier, un objectif d'utilisation se réfère à :

- un système d'intérêt, le système sujet de l'activité de V&V.
- un ensemble d'hypothèses sur l'environnement du système d'intérêt, les systèmes en interface avec le système d'intérêt.
- un ensemble de questions à propos du système d'intérêt.

– un ensemble d'exigences à propos de la validité attendue des résultats (exigences de M&S).

2.4.1.3 Le cadre expérimental

Le cadre expérimental traduit les conditions précises d'expérimentation du modèle du système d'intérêt pour atteindre l'objectif. Il consiste en :

– Un ensemble de points de contrôle et d'observation.
– Un ensemble de scénarios :
  – Un générateur, un ensemble de séquences de valeurs d'entrée qui sont injectées sur les points de contrôle durant la simulation.
  – Un transducteur, un ensemble de séquences de valeurs de sortie qui sont observées sur les points d'observation durant la simulation.
– Un accepteur, un ensemble d'observateurs qui établissent des questions à propos d'une relation entre les entrées et les sorties.
– Un ensemble de résultats qui doivent permettre de répondre aux questions définies par les objectifs.

2.4.1.4 Le modèle exécutable

Un modèle exécutable de simulation consiste en un ensemble de modèles et d'une plateforme d'exécution. Les modèles peuvent être écrits dans un langage de programmation générique (C, FORTRAN...) ou peuvent être exprimés avec des langages (modèles) plus spécifiques à un domaine (Simulink, Scade...). La plateforme d'exécution fournie le moyen de générer le comportement des modèles. Une plateforme d'exécution est généralement constituée d'un ordinateur (ou d'un ensemble d'ordinateurs) ou d'un calculteur (PC...) et d'un système d'exploitation (Windows, Linux...). La plateforme d'exécution fournit egalement les moyens d'instrumentation permettant d'exciter les entrées des modèles (bouttons, joystick, scripts...) et d'observer les sorties des modèles (écrans, fichiers d'enregistrement...). C'est le modèle exécutable qui sera soumis aux scénarios spécifiés par le cadre expérimental.

2.4.2 Caractérisation de la validité d'une M&S

Nous proposons une définition informelle de la validité d'une M&S à travers la relation entre les constituants d'un produit de simulation [Albert 07b].

L'avion est un système constitué d'un ensemble de sous-systèmes, eux mêmes pouvant être décomposés en un ensemble de sous-systèmes. Le processus de développement de l'avion est alors décomposé en un ensemble de processus de développement de sous-systèmes. Les exigences du niveau supérieur sont décomposées en exigences de niveau inférieur pour chaque fonction ou sous-système. Les exigences sont ensuite traduites jusqu'à obtention du produit final ("end product"). Puis, en remontant la hiérarchie de description de l'avion, les produits finaux, chacun responsable d'une fonction de l'avion, sont intégrés afin de satisfaire les exigences décrites au plus haut niveau de la hiérarchie.

Les systèmes "primaires" (ATA) d'un avion (lois de pilotage, alarmes, fuel, hydraulique, communication, navigation, moteurs...) et leur sous-systèmes (sous-ATA) doivent échanger des données de manière à réaliser leurs fonctions respectives. Un sous-système est alors conçu pour évoluer dans un environnement ou un contexte donné, autrement dit dans un cadre expérimental de référence qui est dans notre cas, l'avion réel.

La construction d'un modèle se fait à travers des abstractions. Au fur et à mesure du développement du système, ces modèles deviennent de plus en plus concrets. Nous considérons donc un ensemble de modèles d'un même système, hiérarchisés par une relation de morphisme. Dans cette hiérarchie, un modèle de haut niveau est un modèle plus abstrait que le modèle de bas niveau, dans le sens qu'il est utilisable pour un plus petit cadre expérimental. Au plus bas niveau de la hiérarchie se trouve le système physique, composé d'un ensemble d'applications logicielles, réparties sur un ensemble de calculateurs réels, distribués à travers un réseau et autres éléments physiques. Au niveau le plus abstrait, il y a l'idée. Entre ces deux niveaux extrêmes, une multitude de modèles sous différentes formes et différentes configurations voient le jour : conceptuelles (exigences), formelles (Matlab, Scade, Saber), exécutable (C, C++). Ces modèles sont éxecutés sur des plateformes différentes.

À un instant donné, le développement d'un système nécessite le lancement d'un projet de M&S. L'objectif d'utilisation définit le système d'intérêt, i.e. une partie

isolée du système, modélisé pour un plan de V&V spécifique. Il s'agit ensuite de dériver de cet objectif d'utilisation, une spécification des modèles en interface avec le système d'intérêt au niveau d'abstraction nécessaire et suffisant. La définition complète et détaillée d'un modèle en interface avec le système d'intérêt n'est pas toujours la meilleur solution. Donc des modèles plus abstraits peuvent être utilisés pour permettre des stimulations ou des observations adéquates du système d'intérêt.

Ces systèmes en interface étant eux-mêmes en cours de développement, une représentation suffisement détaillée n'est pas toujours disponible au moment où elle est nécessaire. Que l'abstraction soit souhaitée ou non, des classes d'abstractions doivent être identifiées permettant ensuite de définir les circonstances sous lesquelles un modèle est une représentation valide du système réel, i.e. *le domaine d'usage du modèle*. La description de telles propriétés permet d'assurer une cohérence et une traçabilité entre tout les modèles d'un même système et ainsi faciliter la mise à jour des modèles.

Puisqu'une M&S dépend entièrement d'un objectif d'emploi et que la validité d'un modèle ne peut être mesurée que pour un contexte donné, c'est à travers la notion de cadre expérimental que doit être conduite l'évaluation. Nous définissons deux cadres expérimentaux :

- *SOU* : le Simulation Objectives of Use (SOU), le cadre expérimental qui décrit la façon dont l'expérimentation du système d'intérêt sera conduite. Cette spécification sera couplée au modèle exécutable pour engendrer les résultats attendus. Ce cadre expérimental doit être enrichi pour spécifier les abstractions acceptables.
- *SDU* : le Simulation Domain of Use (SDU) le cadre expérimental qui décrit le domaine d'usage d'un modèle, i.e. ses propriétés, ses limitations.

À travers un processus de vérification et de mise en correspondance entre ces deux cadres expérimentaux, nous proposons de vérifier la compatibilité du scénario avec le modèle. Une simulation est valide si un *SOU* est applicable à un *SDU*. La notion d'applicabilité est utilisée ici en référence à celle donnée par B.P. Zeigler en section 2.2.1

Cette approche permet d'évaluer la validité :

- A priori : définir le *SDU* nécessaire et suffisant pour satisfaire un *SOU* donné.
- A posteriori : définir si un *SDU* déjà développé permet de satisfaire le *SOU*.

Chapitre 3

Propriétés de caractérisation du niveau d'abstraction d'une M&S

Sommaire

L'objectif de ce chapitre est d'identifier les techniques d'abstraction utilisées dans le cadre du développement d'applications de M&S de systèmes embarqués. Dans la première partie nous présentons la synthèse de techniques d'abstraction de modèle trouvés dans la littérature. Bien que la communauté de M&S aborde les notions d'abstraction, nous avons également étudié les techniques utilisées dans le domaine de l'intelligence artificielle et du raisonnement qualitatif [Weld 92] [Falkenhainer 91] [Iwasaki 93] [Kuipers 90] [Rickel 94]. Puis nous proposons une taxonomie adaptée à notre contexte. Cette taxonomie est orientée de telle manière que des règles de compatibilité entre les éléments de cette taxonomie puissent être élaborés par la suite. C'est à partir de ces éléments que nous mettrons en place des

critères de validité afin de définir le modèle nécessaire et suffisant pour un objectif d'utilisation donné.

3.1 État de l'art en matière d'abstraction

Nous avons étudié les travaux dont l'objectif est de documenter des relations d'abstraction entre modèles. La proposition de taxonomie d'abstraction initiale est proposée par P. Fiswick, mais celle-ci touche un périmètre trop large par rapport à nos préoccupations. Nous présentons les taxonomies de B.P Zeigler [Zeigler 84] et de F.K Frantz [Frantz 95] qui concernent les types de procédures d'abstraction utilisées. Ainsi que celle de l'article de D.S. Weld [Weld 92] qui concernent les propriétés d'abstraction de modèles.

3.1.1 Taxonomie d'abstraction de D.S. Weld

La taxonomie de D.S. Weld est basée sur quatre dimensions :

- le "scope" est l'étendue du phénomène que décrit le modèle.
- le "domain of applicability" est caractérisé en terme de contraintes sur les valeurs des paramètres exogènes.
- l'"accuracy" est la distance calculée sur les résultats.
- la "resolution" est la capacité de discrimination du modèle.

Cette proposition a une valeur pragmatique car les différentes "dimensions" de modélisation constituent, d'après Weld, une partition. Cette notion de partition est importante. En fait, elle signifie que ce qui est démontré sur l'une des dimensions est établi independamment des autres.

3.1.2 Taxonomie des technique d'abstraction de B.P. Zeigler

B.P Zeigler définie l'abstraction comme *une méthode ou un algorihtme appliqué à un modèle pour réduire sa complexité tout en préservant sa validité dans un cadre expérimental donné*. En partant du principe que la compléxité est mesurée en termes de temps et d'espace requis pour simuler le modèle, l'auteur rapporte cette complexité à la *taille* et à la *résolution* d'un modèle. La taille fait référence à l'étendue du système que le modèle représente et la résolution fait référence au nombre de variables dans le système et à leur précision ou granularité. La taille

est mesurée à partir du nombre de composants dans le modèle et la résolution à partir du nombre d'états par composant. À ce couple est ajouté l'interaction entre les composants qui permet une meilleure gestion de la complexité.

Méthode de simplification	Description	Propriétés affectées
Aggrégation	Combiner des groupes de composants en un composant unique qui représente le comportement conjoint lorsqu'il interagit avec d'autres groupes	Taille et résolution
Omission	Laisser de côté un composant, une variable ou une interaction	Taille, résolution et interaction
Linéarisation	Représenter le comportement au voisinage d'un point à l'aide d'une relation linéaire	Interactions
Déterministe vers stochastique	Remplacement d'une ou plusieurs variables définies de manière déterministes par des variables aléatoires	Interactions
Stochastique vers déterministe	Remplacement d'une ou plusieurs variables définies de manière aléatoires par des variables déterministes	Interactions
Transformation de formalisme	Mise en correspondance entre un formalisme et un autre (système d'équations différentielles vers un modèle à évènements discret)	

Les méthodes d'abstraction incluent diverses approches de simplifications. Celles proposées par B.P. Zeigler sont présentées dans le tableau ci-dessus. Ces méthodes peuvent affecter soit la taille du modèle soit la résolution. La réduction des interactions doit être mesurée par la compléxité calculatoire des fonctions qui implémentent ces interactions. Ces méthodes d'abstraction sont basées sur la notion de morphisme

qui d'un modèle concret, peut amener vers un modèle abstrait valide. Dans la terminologie de l'auteur un composant est un élément de base d'une unité plus complexe, typiquement le sous-système d'un système. Fondamentalement, un modèle est spécifié par un ensemble de variables choisies pour décrire un système et par un ensemble d'interactions entre ces variables.

3.1.3 Taxonomie des techniques d'abstraction de F.K. Frantz

La taxonomie proposée par F.K. Frantz [Frantz 95] répertorie les techniques d'abstractions de modèles à travers trois grands thèmes : *abstraction des limites d'un modèle*, *abstraction du comportement* et *changement de forme du modèle*.

3.1.3.1 Abstraction des limites du modèle

Les abstractions des limites du modèle concernent les frontières entre le système et son environnement, à savoir ce qui est inclu et exclu du modèle. Elles consistent à modifier l'espace des variables d'entrées du modèle. Ce type d'abstraction se réalise à travers différentes procédures de simplification : limiter le domaine des valeurs, sélectionner des variables selon leur influence et éliminer des paramètres.

Limiter le domaine des valeurs Cette technique consiste à limiter le domaine de valeurs d'une variable d'entrée. Par exemple, un modèle peut être limité à un sous-ensemble de valeurs tel que ce modèle ne s'applique qu'à des températures de l'eau comprise entre 0 et 100 degrés Celsius. Les calculs liés à la transition de l'eau vers un état solide ou de gaz peuvent être éliminés du modèle pour un objectif d'utilisation donné.

Sélectionner des variables selon leur influence Cette procédure de simplification utilise les relations entre les variables observées par la simulation pour trouver le modèle le plus simple. Les abstractions de modèle sont définies en classant les variables d'un modèle en trois classes : exogène, dépendante et sans intérêt. Cette approche décrite par Rickel et Porter [Rickel 94] vise à définir l'abstraction pour laquelle le nombre de variables exogènes est minimal.

Eliminer des paramètres Cette procédure d'abstraction détermine quels paramètres du modèle peuvent être éliminés, produisant ainsi un modèle plus simple au niveau calculatoire, et qui fournit des résultats approximatifs avec une tolérance nécessaire pour atteindre l'objectif de la simulation. Une méthode traditionnelle de l'analyse de l'impact de telles abstractions sur l'exactitude des résultats est l'analyse de sensibilité. Nayak [Nayak 96] propose une approche alternative qui consiste à étudier les relations causales entre les variables.

La pré-définition d'une hiérarchie de modèle est une autre procédure de simplification qui utilise aussi l'élimination de paramètres. Dans la pré-définition d'une hiérarchie de modèles, les relations entre les modèles sont définies comme des hypothèses explicites. Par exemple choisir une loi "idéale" pour représenter un phénomène tout en sachant que cette loi néglige certains facteurs. La loi des gaz parfaits $P = nRT/V$ est un modèle approché de l'équation de Van der Waals $P = \frac{RT}{V-B} - \frac{a}{V^2}$ [Weld 92]. Les graphes de modèles [Addanki 91] et la modélisation compositionnelle [Falkenhainer 91] sont des exemples de hiérarchies pré-définies de modèles.

3.1.3.2 Abstraction du comportement du modèle

D'après la classification de F.K Frantz, nous pouvons abstraire le comportement d'un système à travers l'aggrégation de quatre aspects : les états, le temps, les entités et les fonctions. Cela consiste à grouper un ensemble d'éléments (états, fonctions ou entités) en un élément unique. Les éléments aggrégés ne sont alors plus discernables puisqu'ils sont encapsulés par l'élément de plus haut niveau. Ainsi le modèle abstrait a moins d'éléments que le modèle de base.

Aggrégation d'états Trois approches spécifiques permettent de déterminer comment aggréger des états : l'aggrégation du comportement, la décomposition causale et la représentation numérique.

Aggrégation du comportement Le principe de l'aggrégation du comportement d'un modèle est de grouper les états du système pour lesquels leur distinction n'a aucun intérêt pour les exigences de la simulation. Si le système est représenté par un graphe d'états-transitions, lorsque nous avons un sous-graphe avec une seule entrée et une seule sortie, alors ce sous-graphe est un candidat pour le représenter

par un seul état. Ainsi, il n'est plus nécessaire de calculer les états du sous-graphe.

Les réseaux de Petri hiérarchiques [Valette 79] utilisent l'aggrégation du comportement d'un modèle par aggrégation d'états. La définition des réseaux de Petri hiérarchiques (HRdP) dit qu'il est possible de remplacer une place d'un reseau P par un autre reseau P', de maniere a permettre une description et une analyse par raffinements successifs d'une structure de contrôle. Il est dit alors que le reseau P' est le raffinement du reseau P par substitution d'une place. Ce sous-reseau P' est ainsi appelé un bloc avec une place initiale p_{ini} et une place finale p_{fin}. La figure 3.1 illustre un exemple de substitution d'une place. La place p_3 du reseau de la figure a. est substituée par le bloc (reseau) de la figure b. Le resultat (la mise a plat) de cette substitution est le reseau P'' de la figure c. Les places $p_{1'}, p_{2'}, p_{3'}, p_{4'}, p_{5'} et p_{6'}$ ont été aggrégés en la place p_3 [Albert 05].

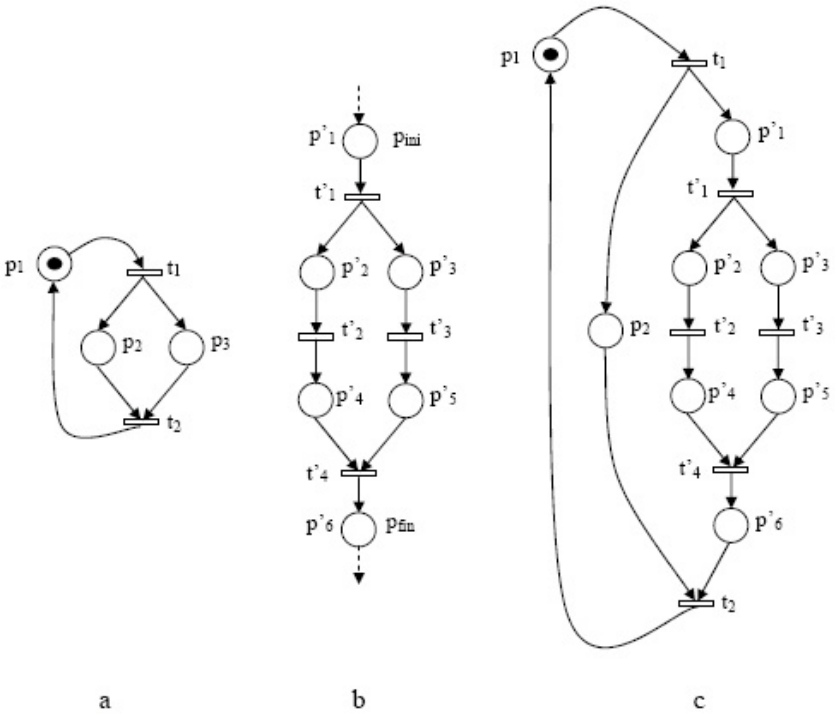

FIGURE 3.1 – Exemple de substitution d'une place par un bloc

Décomposition causale Le principe de la décomposition causale [Kuipers 90] est de découper un modèle en composants faiblement connectés, afin d'exécuter

chaque composant séparément à condition de spécifier les contraintes comportementales entre les composants. Deux composants sans relations causales peuvent être exécutés en parallèle.

La recherche d'invariants de place ou de transition permettent une décomposition structurelle d'un réseau de Petri décrivant le comportement d'un système. Chaque invariant devient un composant de ce système avec des entrées/sorties et des contraintes d'évolution, i.e. communications/synchronisations avec les autres composants.

Représentation numérique Le passage vers une représentation numérique d'une variable continue est un procédure d'aggrégation d'état. Par exemple, remplacer des nombres en virgule flottante par nombre ayant la même valeur entière est une aggrégation d'état.

Aggrégation d'entités Ce que Frantz appelle l'aggrégation d'entités c'est lorsqu'un objet de plus haut niveau est utilisé pour représenter une collection d'objets de plus bas niveau qui peuvent être considérés comme équivalent, e.g. parce que l'objet de plus haut niveau exerce une fonction similaire. Considérons par exemple, le système de commande d'injection de fuel dans les moteurs d'un avion. L'injection est contrôlée par un ensemble de valves en sortie des réservoirs de l'avion. Chacun des réservoirs est disposé à un endroit précis de l'avion. Un modèle peut avoir chaque valve représentée par une entité séparée ou par une seule et même entité avec un débit de sortie corrélativement plus important, i.e. la somme des débits sortant de chaque valve. Dans ce cas, les entités de bas niveau ont la même fonction, et l'entité aggrégée est modélisée afin de réaliser la fonction correspondant à celle du niveau inférieur. Cependant, dans certain cas, cette aggrégation n'est pas permise. Par exemple, elle ne permet pas d'observer le centre de gravité de l'avion en fonction de la consommation de fuel dans chacun des réservoirs. Si l'objectif est d'observer l'évolution du poids de l'avion en fonction du débit sortant alors l'aggrégation est permise.

Aggrégation de fonctions Alors que l'aggrégation d'entité implique un changement des entités du modèle, l'aggrégation de fonctions ne changent pas les entités

qui sont représentées, mais aggrége les fonctions que ces entités réalisent. L'abstraction par aggrégation de fonctions s'apparente à celui du concept de hiérarchie par composition/décomposition. La figure 3.2 illustre ce concept. Le système de train d'atterrissage est un système vital de l'avion. Ce système est en interaction avec d'autres systèmes de l'avion. L'aggrégation de fonction définit la hiérarchie entre système et avion. Au niveau système, nous pouvons voir le train d'atterrissage comme un ensemble d'entités interagissant entre elles et avec l'environnement du système à travers leur entrées et sorties respectives. Au niveau avion, ce système est une entité qui réagit à des évènements d'entrée I provenant d'autres systèmes avion et répond par des évènements de sortie O. À ce niveau, les constituants du système de train d'atterrissage existent toujours, mais les détails d'implémentation du système et/ou de ses constituants ne sont pas disponibles. Donc seules leurs fonctions sont essentielles, la structure est sous-jacente.

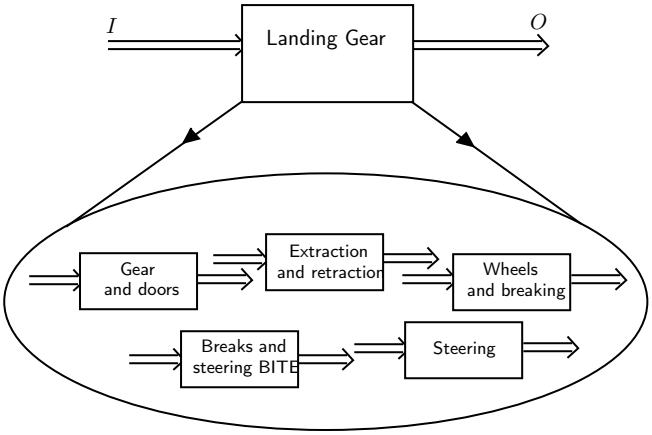

FIGURE 3.2 – Hiérarchie par aggrégation de fonctions

Aggrégation temporelle L'aggrégation temporelle consiste à changer la granularité du pas de temps. Cette abstraction peut être réalisée soit en accroissant le pas de temps soit en considérant que des évènements qui apparraissent très proches dans le temps soient considérés comme simultanés. Ce type d'abstraction a été longuement étudié en intelligence artificielle [Iwasaki 93].

3.1.3.3 Abstraction de la "forme" du modèle

La troisième catégorie d'abstraction concerne les techniques qui simplifient les relations entre les entrées et les sorties. L'ensemble des entrées et des sorties sont les mêmes dans le modèle abstrait et le modèle concret, contrairement à la première catégorie d'abstraction vue en 3.1.3.1. Les entités, les états et la base de temps sont les mêmes, contrairement à la deuxième catégorie 3.1.3.2. Les abstractions de cette catégorie consistent à utiliser des moyens moins nécessiteux en ressources calculatoires (mémoire et temps) pour calculer la valeur des paramètres. Cette catégorie utilise des techniques comme les LUT (look-up table), la distribution de probabilité (remplacement d'une ou plusieurs variables définies de manière déterministes par des variables aléatoires), les fonctions linéaires d'interpolation...

3.2 Proposition d'une taxonomie d'abstraction pour le domaine des systèmes embarqués

Nous proposons une taxonomie des propriétés d'abstractions de modèles pour le domaine des systèmes embarqués. Deux approches sont possibles pour établir une taxonomie. La première consiste à adopter celle de D.S. Weld axés sur la documentation des relations d'abstraction entre modèles et de la métrique. La deuxième consiste à documentation les procédures d'abstraction comme celle de F.K. Frantz et B.P. Zeigler. Nous adoptons la première alternative. Car c'est sur les propriétés formelles des modèles et sur leur métrique que nous elaborons par la suite la mise en correspondance entre un objectif d'utilisation et le domaine d'usage d'une simulation. Ces approches de mises en correspondance entre composants sont présentées dans le chapitre suivant. Signalons cependant que les deux taxonomies ne sont pas incompatibles. En effet, afin d'assurer une cohérence et une traçabilité entre tous les modèles d'un même système, il faut également documenter les procédures d'abstraction et identifier les propriétés des modèles qu'elles affectent.

Comme dans la proposition de D.S. Weld, nous faisons un effort particulier pour garder un maximum d'indépendance entre les éléments de la taxonomie. Nous partons de concepts simples, trivialement calculables (mise en correspondance de noms, de types, de signatures,...), jusqu'à des concepts plus complexes (mise en cor-

respondance de comportements). Cette approche nous permet de traiter une partie du problème concernat l'effort nécessaire pour justifier d'une validité d'une simulation. Une règle de compatibilité complexe nécessite un effort de documentation plus important. D'un autre coté, l'évaluation de la validité à travers une règle de compatibilité complexe, augmente la confiance ou l'efficacité de la mesure de validité.

Notre proposition suppose une définition syntaxique et sémantique à quatre dimensions :

- l'architecture, qui inclut le périmètre de la simulation et sa taxonomie,
- les données, qui inclut les types de données, leur domaine, précison, résolution,...,
- les calculs, qui adresse la mesure d'une distance sur les comportements,
- le temps, qui traite les problèmes d'horloge du simulateur.

Dans la suite de ce chapitre, nous présentons la ou les techniques d'abstraction qui affectent chaque dimension. Chaque technique d'abstraction est illustrée d'un exemple. Cela nous a mené à la définition de propriétés "mesurables" pour caractériser les dimensions.

3.2.1 Dimension architecture

La dimension architecture adresse les problèmes de périmètre de la simulation et comment les composants inclus dans la simulation décrivent le système (boites noires, boîtes blanches). Ces propriétés nous permettrons de résoudre les problèmes de connexion entre un scénario et une simulation en terme de disponibilité des paramètres de la simulation aux frontières de la simulation.

3.2.1.1 Techniques d'abstraction sur l'architecture

Les abstractions sur l'architecture référent aux techniques d'abstraction des limites du modèle et l'aggrégation de fonctions de F.K Frantz ainsi qu'à la technique d'omission de B.P. Zeigler. Les propriétés que nous proposons sur l'architecture d'une simulation sont affectées par toutes les techniques d'abstraction qui consistent à omettre ou aggréger des éléments, des fonctions du système. Lorsque des fonctionnalités du système sont omises, le périmètre de la simulation doit être redéfini ainsi que l'ensemble minimum des objets nécessaires pour atteindre les objectifs d'utili-

sation.

Exemple d'omission et d'abstraction des limites du modèle Le système Air-Traffic Control (ATC) consiste en un ensemble d'applications offrant différents services de communication entre le cockpit et le sol :

- Les applications ACARS (Aircraft Communication Addressing and Reporting System) incluent les fonctions CPDLC (Controller Pilot Data-link Communications), AFN (ATS Facilities Notification), et ADS (Automatic Dependence Surveillance).
- Les applications ATN incluent les fonctions CPDLC, CMA (Context Management Application), FIS (Flight Information Service)et ADS (Automatic Dependence Surveillance).
- Les applications ARINC 623 [ARINC 05] incluent les fonctions DC (Departure Clearance), OC (Oceanic Clearance) and ATIS (Air Traffic Information Service).

Ces services sont offerts à l'équipage à travers le CDS (Common and Display System). Le CDS est composé de divers équipements (unités d'affichage, keyboard, cursor control unit) et d'un ensemble de pages HMI (NOTIFICATION, CONNECTION STATUS, REQUEST, POSITION REPORT, MODIFY...).

Supposons que le système d'intérêt consiste en l'une des fonctions de l'ATC, e.g. l'AFN. Toutes les autres fonctions sont en dehors du périmètre de l'objectif d'utilisation. De plus, seule la page HMI du CDS dédiée à la fonction AFN, i.e. la page NOTIFICATION, devrait être modélisée.

Exemple d'aggrégation de fonction Cette technique est couramment utilisée pour réduire la complexité (en terme de quantité d'informations) d'un modèle en encapsulant les objets qui le constituent. Les détails d'implémentation du modèle sont ainsi "cachés". Le contexte industriel de notre étude génère aussi ce type d'abstraction. Elle est également utilisée pour des raisons de propriétés intellectuelles. Les systèmes d'un avion sont développés par diverses entreprises (e.g. le Flight Management System par Thalès, l'ATC par Honeywell...). Airbus Industrie est responsable de la définition du besoin de tels systèmes puis de leur intégration. De fait, un système ou un modèle est souvent fourni sous forme de code compilé. Seule une vue

boîte noire de ces systèmes est alors disponible. Les détails d'implémentation ne sont pas fournis, ce qui rend inaccessible un certain nombre de données ou de paramètres.

3.2.1.2 Propriétés sur l'architecture

La dimension architecture sera caractérisée par deux propriétés : le périmètre et la topologie.

Périmètre D.S. Weld définit le périmètre d'un modèle par *l'étendue du système que le modèle décrit. Un modèle a un périmètre plus grand qu'un autre s'il décrit plus largement le système. Changer le périmètre d'un modèle est équivalent à redéfinir les frontières entre le système (décrit par le modèle) et son environnement. Donc, implicitement, le choix du périmètre du modèle inclut la sélection des paramètres exogènes et la détermination de leurs valeurs.*

Un paramètre exogène est un paramètre externe dont le système dépend. Un paramètre endogène est un paramètre interne au système. Pour illustrer cette propriété du périmètre, considérons un système A (figure 3.3a) avec un ensemble de paramètres endogènes représentés par des croix et un ensemble de paramètres exogènes représentés par des cercles. Par réduction du périmètre du modèle du système A, nous obtenons un système B (figure 3.3b) avec un nouvel ensemble (plus petit) de paramètre endogènes et un nouvel ensemble (plus grand) de paramètres exogènes.

FIGURE 3.3 – Modèles avec des périmètres différents

Les nouveaux paramètres exogènes (cercles noirs) deviennent alors des paramètres contrôlables et/ou observables par l'utilisateur de la simulation afin de mener convenablement son expérimentation. Ils appartiennent donc, désormais, au cadre expérimental. Nous définissons alors le périmètre d'une application de M&S par l'ensemble des paramètres exogènes au système d'intérêt.

Illustrons la caractérisation de cette propriété sur un exemple. Le système global est un véhicule dont la vitesse est adaptée en fonction de la distance qui le sépare du véhicule qui le précède. Supposons que le régulateur de vitesse du véhicule [Schulz 97] soit le système d'intérêt. Alors, les paramètres exogènes du régulateur sont les informations extérieures au régulateur, nécessaires à l'étude, qui sont fournies par les composants de l'environnement de ce régulateur et qui appartiennent donc au cadre expérimental de ce produit de simulation. Certains de ces éléments peuvent être abstraits ou simplifiés si leurs contributions ne sont pas pertinentes pour le régulateur. Le cadre expérimental consiste alors en des stimuli d'entrée pour la vitesse du véhicule d'intérêt *speedV1*, la distance *distance* et la vitesse du véhicule précédent *speedV2*. L'utilisateur de la simulation fait l'hypothèse que la forme aérodynamique du véhicule n'est pas essentielle. L'utilisateur souhaite observer la position de l'accélérateur *throttle*. La figure 3.4 illustre la frontière entre le cadre expérimental et le modèle du système d'intérêt pour une application de M&S du régulateur de vitesse.

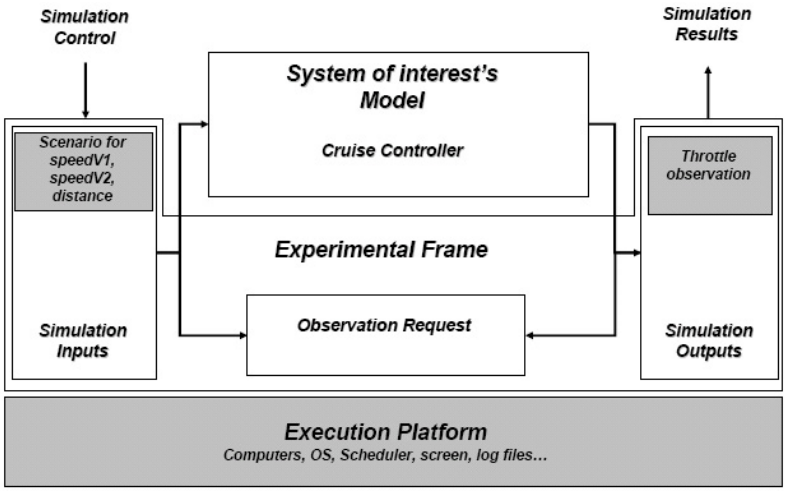

FIGURE 3.4 – Relations entre le modèle du système d'intérêt et le cadre expérimental pour la validation d'un régulateur de vitesse

Imaginons maintenant que ce nous n'étudions plus le régulateur propement dit, mais son intégration dans le véhicule. Dans ce cas, *speedV1* devient un paramètre endogène au système d'intérêt. On peut alors imaginer que ce n'est plus la vitesse du véhicule qui est injectée dans la simulation mais d'autres éléments de contrôle

(logique ou physique) comme les pédales d'accelérateur et de frein. De nouvelles frontières entre le cadre expérimental et le modèle du système d'intérêt sont alors-définies (figure 3.5).

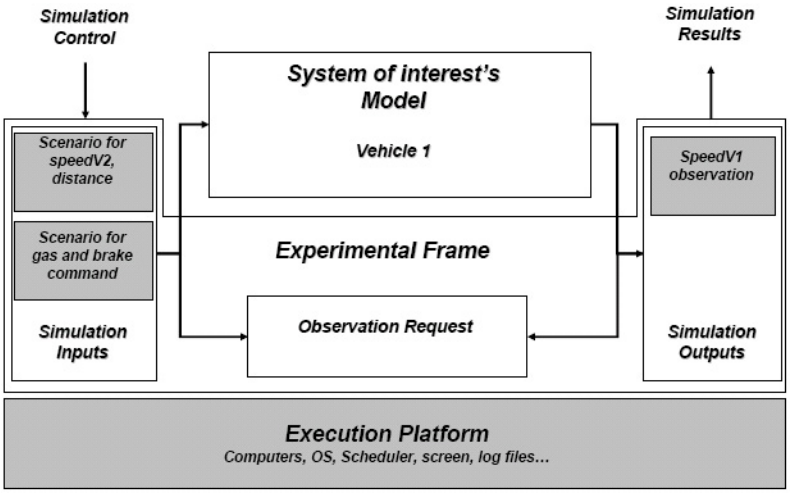

FIGURE 3.5 – Relations entre le modèle du système d'intérêt et le cadre expérimental pour la validation d'un véhicule

Topologie Le concept de topologie permet de répondre au problème de disponibilité des paramètres aux frontières du produit de M&S. Dans certains, cas un scénario peut nécessiter des entrées ou des sorties spécifiques à l'expérimentation, à savoir qu'elles ne sont pas forcément des entrées/sorties du système réel. La topologie d'un produit de M&S nécessite de considérer celles-ci comme un ensemble de composants qui intéragissent à travers des ports d'entrées/sorties, i.e. des points de contrôle et d'observation spécifiques. Les entrées et les sorties (les paramètres exogènes) de ces composants sont accessibles à travers ces ports d'entrée/sortie. Dans l'exemple du véhicule figure 3.5, le scénario nécessite que la simulation offre un point d'observation de la vitesse du véhicule. Ce point d'observation peut être capturé en différents points. Sur le système réel la vitesse du véhicule peut être observée sur le compteur de vitesse. Nous pouvons imaginer que l'objectif d'utilisation nécessite d'observer l'état de la variable *speedV1* à entrée du régulateur de vitesse. Dans ce cas, la simulation doit implémenter un moyen d'observer cette variable. Le composant *véhicule* doit alors avoir un port de sortie spécifique à l'observation de cet état.

3.2.2 Dimension donnée

La dimension donnée traite le problème de compatibilité des types de données d'une simulation. D'après son périmètre et sa topologie, un produit de M&S consiste en un ensemble de composants interconnectés par des ports d'entrée/sortie et un environnement qui définit les limites du système d'intérêt. Nous considérons que les composants envoient et reçoivent des messages, respectivement de et sur leurs ports via des connecteurs et que les données sont encapsulées. La dimension donnée vise à contraindre le type de la donnée. L'évaluation de la validité repose sur des règles de compatibilité de type. Par exemple, si un composant doit recevoir des données codées par des entiers, et qu'un autre lui envoie une chaîne de caractères, le premier composant ne fonctionnera pas correctement.

Une simulation manipule un nombre varié de types de donnée, à différents niveaux d'abstraction. Ceux-ci incluent les types associés à un langage de programmation et à un paradigme de modélisation :

1. les types mêmes : bool, integer, char, float
2. leur format : mots A429 [ARINC 95], mots AFDX (Avionics Full DupleX)
3. leur type abstrait, qui spécifie le concept associé à la donnée : altitude, vitesse, commande d'accélération
4. leur "types mathématiques", plus spécifiques liés à des propriétés qui rentrent directement dans les techniques d'interprétation abstraite [Cousot 92], e.g. précision, résolution, domaine.

3.2.2.1 Techniques d'abstraction sur les données

La caractérisation de propriétés sur les données concerne les procédures de coercition de type et les techniques d'abstraction de F.K. Frantz de limitation du domaine de valeurs et aggrégation des états.

Exemple de coercition de type Il est fréquent en M&S de changer le type d'une donnée. Cette technique est très connue dans le domaine de l'informatique sous le nom de conversion de type. En langage C, la conversion d'une variable de type entier vers un type réel est un exemple de ce genre de conversion. La coercition de type est aussi utilisée à un niveau plus abstrait. Dans les phases de modélisation nous

retrouverons les types des quantités physiques dont les plus simples sont les unités du système international (mètre, litre, pied). Il est possible d'utiliser différentes valeurs pour exprimer la même grandeur. Par exemple une altitude peut s'exprimer soit en mètres soit en pieds ou même en niveau de vol (1 Flight Level = 100 mètres). La quantité d'information est la même mais le composant qui fournit une altitude en mètres n'est pas sémantiquement connectable avec un composant qui attend une altitude exprimée en niveau de vol.

Exemple d'aggrégation d'état Le concept d'aggrégation d'états correspond à la définition de B.P. Zeigler [Zeigler 00b] appelée coercition du domaine des valeurs d'une variable pour laquelle *des variables d'un modèle abstrait peuvent décrire relativement moins de conditions (ou d'états) que celle que l'on peut trouver dans un modèle concret*. Les états (ou conditions) ont été aggrégés.

Nous retrouvons ce type de techniques dans l'interprétation abstraite. En informatique, l'interprétation abstraite [Cousot 92] est une théorie d'approximation de la sémantique de programmes basée sur les fonctions monotones pour des ensembles ordonnés, en particulier les treillis (en anglais : lattice). Considérons la valeur '2'. Toutes les propriétés listées à droite de la figure 3.6 sont des abstractions par aggrégation d'état de la valeur '2'. Dans un treillis, le passage d'un élément aval vers un élément amont dénote une perte d'information.

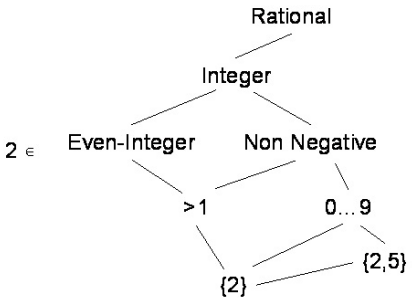

FIGURE 3.6 – Un treillis d'abstraction de la valeur '2'

Dans le même ordre d'idée, la communauté du raisonnement qualitatif a exploré des alternatives dans la représentation des paramètres continus allant de l'algèbre des signes aux réels [Forbus 96] [Trave-Massuyes 03]. La plus faible résolution d'une représentation est l'abstraction par l'état, qui représente une quantité par

le simple choix entre "normale" ou non. Puis, l'algèbre des signes représente un paramètre continu par un "-", un "+" ou un "0". Puis, une valeur continue peut être représentée via des espaces de quantités. Cela consiste à identifier pour chaque variable un ensemble de valeurs remarquables qui partitionnent le domaine de la variable en espaces de quantités. Par exemple, une température d'un fluide peut être représentée par un ensemble de valeurs qualitatives ordonnées tel que :

$$\{zero,]zero, freezing[, freezing,]freezing, boiling[, boiling,]boiling, +\infty[\}$$

Ces techniques d'abstraction sont utilisées dans la conception des systèmes. Par exemple, la notion de phases de vol est une représentation abstraite de l'altitude communément utilisée par les systèmes d'un avion (figure 3.7). Au plus haut niveau de la description, l'altitude est représentée par deux valeurs "en vol" et "au sol". Le type phase de vol permet de donner un peu plus d'information sur l'altitude : une altitude sera représentée par les valeurs "PREFLIGHT", "TAXI OUT", "CLIMB", "CRUISE"... et nous savons que la phase '6' ("CLIMB") correspond à une altitude comprise entre 0 et $400ft$.

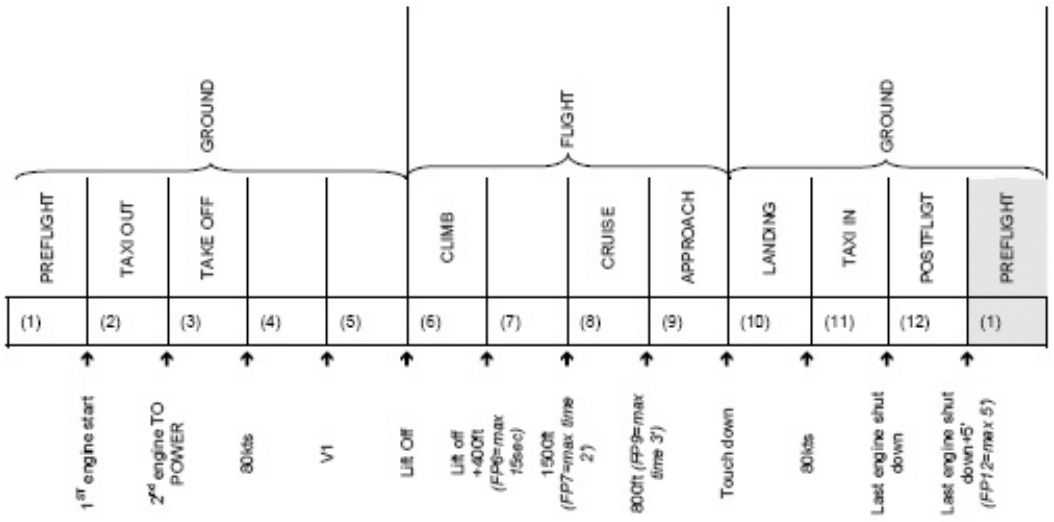

FIGURE 3.7 – Exemple de spécification de phases de vol

La quantisation est également une technique d'abstraction utilisée dans le cadre du développement d'un modèle qui aggrège les états. L'encodage d'une donnée dans

un référentiel digital induit inévitablement des approximations. La quantisation est le processus d'approximation d'un ensemble de valeurs continues (ou un large ensemble de valeurs discrètes) par un ensemble plus petit de symboles discrets ou de valeurs entières. Un signal continu peut prendre un nombre infini de valeurs, avec idéalement une exactitude infinie, alors que l'exactitude d'un signal digital est dépendant de la résolution de la quantisation. Ce type d'approximation survient car il est impossible de représenter tous les nombres réels avec une machine à états finis. Les calculs sont réalisés sur un nombre fini de chiffres (arithmetic IEEE). Par exemple la valeur réel de $\ln 2$ est 0.69314718055994530941.... Une représentation finie arrondie au 3ème chiffre après la virgule de $\ln 2$ est 0.693.

3.2.2.2 Propriétés sur les données

Ces techniques d'abstraction sur les données nous ont amené à la définition de propriétés mathématiques. Elles ont consisté à préciser la qualité d'une donnée par les éléments suivants :

- Une unité
- Un domaine : Le domaine de valeurs que peut prendre une donnée. Par exemple un modèle peut uniquement accepter une tension si elle est comprise dans l'intervalle $[-10V; +10V]$ ou la surface d'une valve comprise dans l'intervalle $[0; S_{max}]$.
- Une résolution : la distance entre deux valeurs successives d'une donnée. Par exemple une tension limitée au domaine $[-10V; +10V]$ avec une résolution de $5V$ ne peut prendre que les valeurs $-10V, -5V, 0V, 5V, 10V$.

La figure 3.8 ci-dessous illustre les notions de domaine et de résolution d'une donnée.

L'interprétation abstraite et la quantisation augmente le nombre de valeurs que peut prendre une donnée. Ce type d'abstraction est prise en compte par la propriété de *précision* (figure 3.8). La précision d'une donnée permet de spécifier la distance entre une valeur physique et sa valeur à un plus haut niveau de représentation. Une métrique permettant de mesurer la précision est le nombre de valeurs possibles que peut prendre un échantillon. Par exemple l'encodage d'un signal discret en un signal digital sur 16 bits peut prendre une des 65.536 valeurs possibles par échantillon.

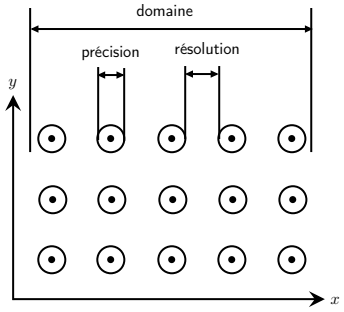

FIGURE 3.8 – Le triplet domaine - résolution - précision

3.2.3 Dimension calcul

La dimension calcul concerne la relation d'entrée/sortie d'un modèle. Alors que le domaine des données concerne le type des données, nous étudions dans le domaine des calculs la complexité d'une relation entre les entrées et les sorties d'un modèle. Nous parlerons plus généralement d'"accuracy" et nous faisons référence au concept du même nom chez D.S. Weld.

L'exemple qui suit illustre la différence entre l'exactitude d'un calcul et la précision d'une donnée. Prenons un modèle qui donne la position d'un avion selon un point donné. La position exacte de l'avion par rapport à ce point est $D = 100m$. Considérons que le modèle donne une valeur de $98m$. Si la précision du modèle est supposée être infiniment élevée (erreur de précision est 0), on peut dire que le modèle ne doit pas être utilisé. Supposons maintenant que le modèle donne une valeur de D avec un erreur de précision égale à $2m$ ($[98, 102]$), le modèle est utilisable puisque la valeur vraie est dans l'intervalle. Nous justifions ainsi la séparation des dimensions donnée et calcul afin de tracer les sources d'incertitudes (manque de précision ou d'exactitude).

3.2.3.1 Exemple de techniques d'abstraction sur les calculs

Les abstractions sur les calculs s'appuient sur les techniques d'abstraction de changement de forme du modèle de F.K. Frantz. Nous classons ces techniques en deux grandes classes d'abstraction qui génèrent des approximations sur les calculs. La première appelée *simplification*, a pour utilité de remplacer une relation fonctionnelle complexe entre des variables internes d'un modèle par une fonction plus simple

mais moins exacte. La deuxième classe d'abstraction concerne les approximations liées aux méthodes d'analyse numérique.

Simplification Ce type d'approximation peut être de nature variée. Certaines consistent à omettre des paramètres qui ne sont pas majeurs pour un objectif d'utilisation. C'est l'exemple de l'équation de Van der Waals qui est un modèle plus exact que la loi des gaz parfaits. Ou encore, le calcul d'un débit de fuel pourrait ignorer la pression exercée en sortie du réservoir. Le calcul de la pression peut aussi, au lieu de dépendre de la quantité de liquide dans le reservoir, être remplacé par une constante. Ce type d'hypothèse est présenté par Iwasaki [Iwasaki 93] sous le nom d'*exogénisation*. L'exogénisation consiste à remplacer une équation dynamique par une constante (à variable exogène). Cette approximation est vue par Iwasaki comme une abstraction de l'échelle de temps. En effet, certaines variables changent si lentement, en réponse à certains évènements qu'elle peuvent être vues comme indépendantes d'autres variables. L'équation correspondant à une telle variable peut être remplacée par une constante. Par contraste aux variables qui réagissent trop lentement à un changement d'autres variables, certaines variables agissent si rapidement pour atteindre leur état d'équilibre qu'elle peuvent être vues comme instantanées. Cette opération est appelée l'équilibration [Iwasaki 93]. Prenons par exemple, le cas d'un modèle permettant la commande d'ouverture et de fermeture d'une valve dans un système de fuel. Ce modèle donne deux variables de sorties : *Valve_Open_Signal* et *Valve_Shut_Signal*. L'ouverture et la fermeture de la valve nécessitent un certain temps *Valve_Time*. Quand la valve est en position ouverte, la variable *Valve_Open_Signal* est affectée à "TRUE". Quand la valve est en position fermée, la variable *Valve_Shut_Signal* est affectée à "TRUE" (figure 3.9). A tout autre instant (pour une durée égale à *Valve_Time*), *Valve_Open_Signal* et *Valve_Shut_Signal* sont affectées à "FALSE". Il y a alors une équilibration de la valeur de la position de la valve. L'exactitude du débit de fuel fourni aux moteurs est, bien entendue, affectée par cette approximation puisque la valve est modélisée soit ouverte soit fermée alors qu'en réalité elle est de résistance (à l'écoulement) variable et continue.

En utilisant l'exogénisation ou l'équilibration, un modèle de base est simplifié en remplacant une équation dynamique par une équation d'équilibre ou par une

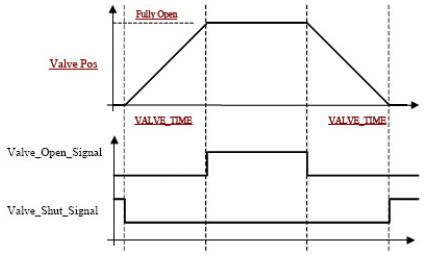

FIGURE 3.9 – Position d'une valve

constante. Il est alors possible de produire des modèles différents allant d'un modèle complètement dynamique à un modèle statique en fonction de la granularité temporelle nécessaire à l'objectif d'utilisation.

Les Look-up tables (LUT) sont également très largement utilisées par les développeurs de modèles. Une LUT est une structure de donnée, généralement une table, utilisée pour remplacer un calcul par un ensemble d'opérations de consultation. La valeur d'entrée est convertie en un index de la table et la valeur de sortie est déterminée à partir de la valeur indéxée. Ce type d'abstraction offre un gain important en vitesse de calcul. Cependant, une table nécessite une grande quantité d'espace mémoire. Et plus le nombre de point (d'index) est grand, plus l'espace mémoire nécessaire est important et l'exactitude du calcul est élevée.

Il existe une multitude d'autres exemples utilisés par les dévelopeurs de modèle de système avion qui consistent à simplifier un calcul : la supposition qu'un réservoir est modélisé comme un récipent de capacité infinie, l'utilisation d'un modèle d'une planète sans relief pour le calcul d'une altitude ou d'une planète sphérique et non éllipsoidale pour le calcul d'une position (latitude/longitude), etc.

Analyse numérique La seconde classe d'hypothèses qui engendrent des approximations sont celles dues à une analyse numérique d'un modèle continu. Nous entendons, par analyse numérique, toutes les méthodes permettant l'élaboration d'algorithmes pour la résolution numérique de problèmes mathématiques continus généralement représentés par l'algèbre linéaire ou les équations différentielles. Ce type de problème nécessite des méthodes itératives permettant de donner une solution approximative (ex. méthodes d'approximation des intégrales, le théorème du point fixe, méthode d'approximation itérative des solutions d'équations fonctionnelles).

3.2.3.2 Propriétés sur les calcul

Relations d'entrée/sortie Pour rester dans l'idée de définir des propriétés de plus en plus complexes, nous proposons d'aborder la dimension calcul par une propriété qui consiste à lister les correspondances entre les entrées et les sorties du modèle. Cette propriété permet de s'assurer le plus simplement possible qu'une dépendance fonctionnelle entre un ensemble de stimuli et un ensemble d'observateurs est représentée dans le modèle. L'idée de cette propriété vient des mesures de testabilité pour les logiciels flots de données [Nguyen 02]. La testabilité d'un système se base sur la contrôlabilité et l'observabilité des composants du système. La contrôlabilité d'un composant est la facilité d'amener des données de l'entrée du système jusqu'à l'entrée de ce composant. L'observabilité d'un composant est la facilité de propager des données de la sortie de ce composant jusqu'à la sortie du système. Ces mesures sont basées sur des modèles qui décrivent les flots de données entre les composants d'un système.

Exactitude et précision mathématique d'un calcul Nous avons vu qu'il existe deux grandes classes d'approximation : celles liées à des hypothèses de simplification d'un calcul et celles liées à l'analyse numérique. Nous proposons donc deux propriétés permettant de documenter et tracer ces approximations : *l'exactitude* et *la précision mathématique du calcul* correspondant respectivement aux erreurs associées aux deux classes d'approximation citées ci-dessus. Nous dirons que l'exactitude est la distance entre une solution approximée d'un calcul et la solution exacte. L'exactitude d'un calcul est le résultat d'approximations de modélisation telle que l'utilisation de LUT (Look-Up Table), d'une gravité constante, d'un modèle de terre plate pour le calcul d'une altitude, d'une terre sphérique pour le calcul d'une position, d'équilibration, d'exogénisation. La précision mathématique d'un calcul mesure les erreurs engendrées et propagées par une méthode d'analyse numérique. Notre proposition consiste à dire qu'il y a deux types d'erreurs qui agissent sur la précision mathématique : les erreurs de quantisation et les erreurs de discrétisation. L'erreur de quantisation est la distance entre une valeur analogique et une valeur digital quantizée alors que l'erreur de discretisation est l'erreur résultant du fait qu'une fonction d'une variable continue est représenté dans un référentiel digital par un nombre fini d'évaluations. La quantisation approxime une variable d'une

représentation infinie vers une représentation finie (ex. $\ln 2 = 0.693$). La discrétisation approxime une variable pouvant prendre une infinité de valeurs en un ensemble fini de valeurs, par exemple à un nombre fini de points parmi son domaine, cas de l'échantillonage par exemple.

Nous considérons que l'erreur de quantisation est la cause de deux phénomènes : l'arrondi et la troncature. Les erreurs d'arrondi surviennent parce qu'il est impossible de représenter tous les nombres réels sur une machine à états finis. Les erreurs de troncature se produisent lorsqu'une méthode itérative est terminée et que la solution approximée diffère de la solution exacte. Considérons par exemple le problème de résoudre numériquement $3x^3 + 4 = 28$. Avec une méthode directe la solution est $x = 2$. Si l'on applique une méthode itérative (ex. la méthode de dichotomie) à $f(x) = 3x^3 + 4$ avec les valeurs intiales $a = 0$, $b = 3$, $f(a) = 4$ et $f(b) = 85$, le tableau ci-dessous montre qu'après trois itérations la solution est entre 1.875 et 2.0625. Il y a donc une erreur de troncation de 0.125.

a	b	mid	$f(mid)$
0	3	1.5	14.125
1.5	3	2.25	38.17...
1.5	2.25	1.875	23.77...
1.875	2.25	2.0625	30.32...

Nous pensons qu'une erreur d'approximation peut être tracée et quantifiée. Par exemple, l'erreur d'approximation maximal produite par l'utilisation d'une LUT est la différence maximal entre deux valeurs de sorties de la table. Une approximation linéaire est une approximation d'une fontion générale qui utilise une fonction linéaire, e.g. la tangeante. Des méthodes mathématiques permettent d'évaluer les erreur d'approximation introduite par une linéarisation, e.g. théorème de Taylor.

Interaction dynamique Nous proposons une dernière propriété sur la dimension calcul qui consiste à définir une signature comportementale d'un modèle. La signature comportementale permet de capturer l'interaction dynamique des composants, e.g. l'ordre relatif d'appel des fonctions d'un modèle. Nous proposons grâce à cette propriété de documenter l'ensemble des traces acceptées par un modèle et les traces des scénarios d'utilisation de ce modèle qui peut être vu comme un plan de test. Cela

permet d'identifier les abstractions qui ont été faites sur les interactions du modèles et de vérifier que les fonctions de transition et de sortie d'un modèle (section 4.1.2.1 sont conformes à celles nécessaires à l'objectif d'utilisation.

3.2.4 Temps

Dans la taxonomie de F.K Frantz, l'aggrégation temporelle discute de la réduction de la granularité de l'avancement du temps par l'utilisation d'un avancement du temps dirigé soit par une horloge, soit par les évènements. Ces modalités de gestion du temps conduisent, en principe, à des simulation différentes. Mais par rapport au problème qui nous concerne, le concept de compatibilité du temps réfère à l'alignement d'horloge d'un scénario sur celle d'un modèle.

Pour éviter les confusions lorsque nous parlons du temps, il est important de distinguer trois concepts de temps différents. Le premier, appelé temps de "l'horloge murale", est le temps du monde réel. Il ne peut pas être arrété et on ne peut pas agir dessus. La façon dont le temps est modélisé est appelé le temps mathématique. Par exemple, dans une simulation d'un vol Paris-Orlando, le temps s'étend de 12.45 à 22.45 dans l'après-midi du 27 février 2009. Puisque nous avons exclu de l'étude les systèmes continus, nous parlerons de temps mathématique discrétisé. Le temps mathématique discrétisé est une sous-classe du temps mathématique qui est continu et infiniment précis. Le temps du simulateur est le temps tel qu'il est représenté dans le simulateur. Dans le vol Paris-Orlando, le temps du simulateur peut être représenté comme une variable à virgule flottante et précision double qui peut prendre des valeurs dans l'intervalle $[0.0, 10.0]$, où l'unité du temps du simulateur correspond à une heure de temps. Nous n'adressons pas de propriétés liées à la performance du simulateur, i.e. le temps nécessaire pour résoudre un problème calculatoire. Par exemple le simulateur du vol Paris-Orlando peut nécéssiter d'une heure et quinze minutes pour s'exécuter. Dans ce cas, nous avons un scénario de dix heures exécuté par le simulateur en une heure et quinze minutes. Le temps d'exécution d'un simulateur se mesure avec le temps de l'horloge murale.

Le temps mathématique discrétisé est une variable interne du modèle, c'est un type de donnée particulier, mais qui n'est pas spécialement distinct de tous les autres. Les propriétés liées au temps mathématique discrétisé sont donc traitées avec les concepts de précision, de domaine et de résolution. Les problèmes de pas

de discrétisation ont été traités par la dimension calcul.

Dans le cas du temps du simulateur, le problème de compatibilité relève du possible ordonnancement d'un modèle par et pour le scénario.

3.2.5 Récapitulatif de la taxonomie

La figure 3.10 récapitule l'ensemble des propriétés d'abstraction d'une simulation. Les procédures d'abstractions (élimination de variables, aggrégation d'états...) et les conséquences qu'elles ont sur les propriétés d'abstraction n'apparaissent pas dans ce schéma.

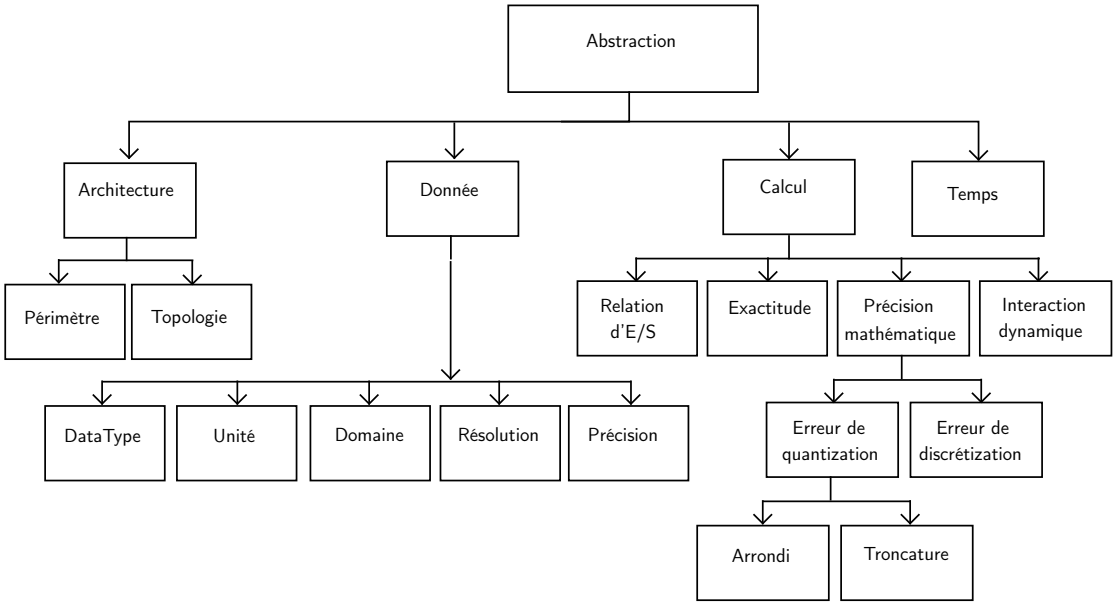

FIGURE 3.10 – Taxonomie d'abstraction de modèle

Chapitre 4

Propriétés et critères formels d'évaluation

Sommaire

L'étude de la validité d'une simulation repose sur la compatibilité entre le cadre expérimental lié à l'expérimentation d'un système d'intérêt et le domaine d'usage d'un modèle. Pour obtenir une documentation formelle de ces "objets", nous nous basons sur la structure algébrique d'un système. Nous utilisons la notion de composant et de mise en correspondance entre composants afin de vérifier l'applicabilité d'un cadre expérimental à un domaine d'usage. La première section de ce chapitre présente les approches de composition et de mise en correspondance de composants dans le domaine de l'ingénierie logicielle et de la M&S. Puis, nous présentons la structure algébrique développée dans la théorie de la M&S. La seconde section définit formellement la mise en correspondance entre un cadre expérimental et un domaine d'usage. Cette mise en correspondance est établie sur les éléments de la structure algébrique.

4.1 État de l'art en vue d'une formalisation du problème

4.1.1 Ingénierie basée composant

4.1.1.1 Mise en correspondance de composants dans le domaine du logiciel

Le principe de l'ingénierie logicielle basée composants (CBSE) consiste à organiser un système logiciel qui satisfait des exigences données en utilisant des composants. Un composant logiciel est [Szyperski 02] *une unité de composition avec des interfaces contractuelles spécifiques et des dépendances contextuelles explicites. Un composant logiciel peut être déployé de façon autonome puis soumis à une composition par des tiers.* A travers ses interfaces le composant est connecté à son environnement. L'objectif des approches formelles de l'ingénierie logicielle basée composants est de spécifier les propriétés d'un composant sur ses interfaces. Cela consiste à enrichir les interfaces en fournissant une *signature* et une *spécification* du composant [Zaremski 97]. La signature décrit une information sur le type du composant. Elle joue le même rôle qu'une signature de type en langage de programmation. Elle définit par exemple le type d'entrée/sortie d'une fonction ou d'une méthode. Ceci inclut le nom de la fonction et le nombre de paramètres et éventuellement le type de donnée retourné par la fonction. La signature couvre les aspects statiques du composant.

La spécification décrit le comportement dynamique du composant. Plus précisément, elle caractérise sa sémantique plus qu'une simple signature, e.g. l'ordre dans lequel les fonctions du composant doivent être appelées ou le protocole de communication utilisé. La spécification d'un composant peut être réalisée à l'aide de plusieurs méthodes. Une façon commune de décrire la spécification d'un composant est d'utiliser des assertions booléennes appelées "pré" et "postconditions" pour chaque service offert, ainsi que des invariants pour définir des propriétés générales d'utilisation cohérente d'un composant. Une spécification est alors interprétée de la manière suivante : un client peut établir une requête à un fournisseur seulement si le service requis est dans un état qui ne viole pas l'invariant et la précondition associés au service. En retour, le fournisseur garantit que ce qui est spécifié par la postcondition et l'invariant seront respectés. Ainsi, les droits et devoirs du client et du

fournisseurs sont clairement identifiés. Cette méthode provient initialement du language Eiffel [Meyer 88] définie sous le nom de conception par contrats [Mitchell 02] [Beugnard 99] [Carrez 03]. Elle est maintenant utilisée dans de nombreux langages tels que Java et aussi dans UML, notamment à travers le langage OCL [OMG 04]. Dans [Lee 02], les auteurs utilisent les automates à interface [Alfaro 01] pour décrire la spécification d'un composant.

Ensuite la spécification de mise en correspondance ("specification matching") est utilisée afin de définir si deux composants logiciels sont liés syntaxiquement et sémantiquement. Cela permet de résoudre un ensemble de questions diverses telles que [Zaremski 97] :

- Le recouvrement : Comment recouvrir un composant d'une librairie logicielle basé sur sa sémantique plutôt que sur sa structure syntaxique ?
- La réutilisation : Comment adapter un composant d'une librairie logicielle pour convenir aux besoins d'un sous-système donné ?
- La substitution : Quand remplacer un composant par un autre sans affecter le comportement observable d'un système ?
- Le sous-type : Quand un objet est-il un sous-type d'un autre objet ?

A chaque composant est associé une signature C_{sig} et une spécification C_{spec}. Etant donné deux composants, $C = \langle C_{sig}, C_{spec} \rangle$ et $C' = \left\langle C'_{sig}, C'_{spec} \right\rangle$, une mise en correspondance de composants est une fonction (un prédicat) $Match$ [Zaremski 97] :

$$Match : Component, Component \rightarrow Bool$$

$$Match(C, C') = match_{sig}(C_{sig}, C'_{sig}) \wedge match_{spec}(C_{spec}, C'_{spec})$$

Deux composants C et C' sont en correspondance si leurs signatures et leurs spécifications sont mises en correspondance. L'exacte mise en correspondance n'est pas toujours nécessaire. Dans le cas de la substitution, il n'est pas nécessaire que le composant de substitution ait exactement le même comportement que le composant substitué. Il suffit que son comportement soit adapté à un environnement et à des objectifs donnés. Les auteurs ont proposé différents types de mise en correspondance plus ou moins "forte" (exact pre/post match, plug-in match, plug-in postmatch, guarded plug-in match...).

4.1.1.2 Validité d'une composition dans le domaine de la M&S

E.W Weisel, M.D. Petty et R.R. Mielke [Weisel 03] traitent le problème de la préservation de la validité d'une composition de modèles valides. Pour ce faire, ils ont établi une théorie de la composabilité. Ils définissent la composabilité comme *la capacité à sélectionner et assembler des composants de simulation dans diverses combinaisons de simulation de systèmes pour atteindre les exigences d'un utilisateur*. Par similitude à la mise en correspondance de composants logiciels, deux types de composabilité ont été définis : syntaxique et sémantique. La composabilité syntaxique traite la compatibilité au niveau des détails d'implémentation, e.g. au niveau des mécanismes de communication ou aux niveaux des interfaces. La composabilité sémantique est liée à la validité d'une composition, i.e. les modèles qui sont composés communiquent de façon cohérente pour atteindre un objectif donné. Elle traite les problèmes de domaine de validité d'un modèle et de la cohérence des hypothèses de modélisation. Deux composants peuvent être syntaxiquement connectés (une donnée peut passer de l'un vers l'autre) alors que leur composition est sémantiquement invalide.

La théorie de la composabilité traite l'aspect sémantique d'une composition. Elle part du principe, comme l'illustre la figure 4.1, qu'un modèle M est une fonction exécutable f qui affecte à tous les éléments d'entrée du modèle $x \in X$ un élément de sortie $y = f(x)$ unique tel que $y \in Y$.

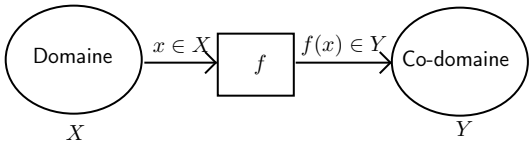

FIGURE 4.1 – Définition formelle d'un modèle

Soit S un ensemble non vide d'états, I l'ensemble des valeurs d'entrée et O l'ensemble des valeurs de sortie tels que S, I et O sont des ensembles de vecteurs finis d'entiers. $X \subseteq (S \times I)$ est appelé le domaine de la fonction et $Y \subseteq (S \times O)$ est appelé le co-domaine de la fonction. La notion de fonction exécutable, tirée de la théorie de la calculabilité, signifie qu'elle est exécutable par un système informatisé (machine de Turing) en un nombre fini de pas.

Ensuite, les auteurs définissent la simulation comme une séquence d'exécutions du modèle, *où la sortie du modèle à chaque pas d'exécution est son entrée pour le pas suivant*. La figure 4.2 illustre cette définition.

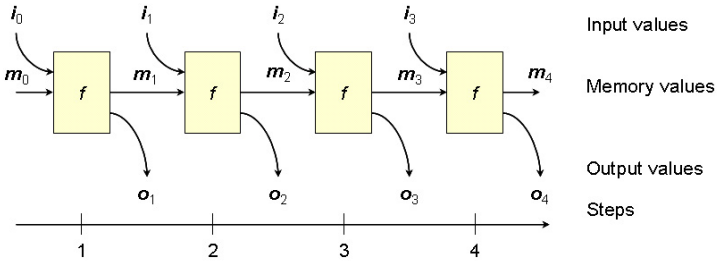

FIGURE 4.2 – Définition d'une simulation

À chaque pas d'exécution n, la fonction calcule la sortie o_n et l'état m_n du modèle à partir de l'état m_{n-1} et de l'entrée externe i_{n-1} au pas d'exécution précédent. Ceci est conforme à la définition d'un modèle énoncée précédemment.

La définition de la validité d'un modèle est conforme à celle que nous avons évoquée au chapitre 2, i.e. *le comportement du modèle est "suffisamment proche" du système qu'il représente*. Formellement, les auteurs utilisent le concept de *bisimulation* pour statuer sur la validité d'un modèle. La bisimulation est le processus de mise en correspondance de trajectoires de deux *systèmes à transitions labélisées* (LTS). Si les deux LTS exhibent les mêmes trajectoires, ils sont équivalents.

Les auteurs ont ensuite étendu les définitions formelles de modèle et de simulation pour adresser les modèles composites. Soit deux modèles $f : A \to B$ et $g : C \to D$, la composition est dénotée $h = f \circ g$. Pour être composable, f et g doivent être compatibles, i.e. le co-domaine de f doit être un sous-ensemble du domaine de g : $f(A) \subseteq C$.

En partant du principe que f est valide pour un sous-ensemble de son domaine A' et que les résultats de f sont valides dans un sous-ensemble de son co-domaine B', une composition $h(a') = g(f(b'))$ est valide si le domaine de g et le co-domaine de f ont une intersection non nulle : $B' \cap C'$ (figure 4.3).

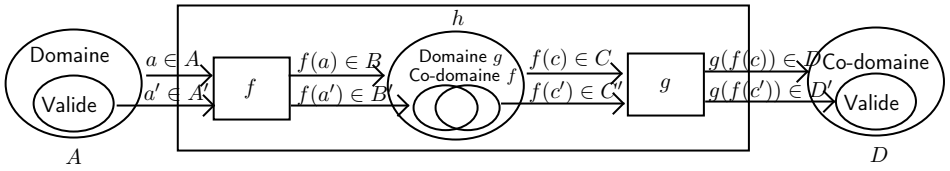

FIGURE 4.3 – Composition valide

4.1.2 Formalismes de spécification des systèmes

La théorie de la M&S de B.P Zeigler [Zeigler 00b] propose au même titre que les travaux de E.W. Weisel, M.D. Petty et R.R Milke, une description formelle d'un modèle et d'une simulation. Cette description formelle repose sur la théorie des systèmes qui est un formalisme rigoureux pour la représentation des systèmes dynamiques. Cette théorie distingue la structure du système (la constitution intérieure du système) de son comportement (sa manifestation extérieure). Le comportement d'entrées/sorties d'un système consiste en une paire d'enregistrements de données collectées à partir d'un monde réel ou d'un modèle. La structure interne d'un système inclut son état et son mécanisme de transition d'états (elle définit comment une entrée transforme un état courant en un état suivant) ainsi que son mécanisme de génération de sorties.

B.P. Zeigler a identifié deux aspects principaux et orthogonaux de cette théorie :

- les formalismes de spécification des systèmes : les types de style de modélisation, tel que continu ou discret, que les modélisateurs peuvent utiliser pour construire leurs modèles.
- les niveaux de spécification du système : les niveaux auxquels nous pouvons décrire comment les systèmes se comportent et les mécanismes qui les font fonctionner.

Un formalisme utilise des abstractions. Si différents formalismes existent, c'est d'ailleurs parce que la complexité des systèmes du monde réel nécessite d'avoir différentes formes d'abstractions, parfois successivement, parfois simultanément. L'idée directrice est de construire une sorte de hiérarchie entre les formalismes où chaque formalisme serait la sous-classe d'un autre formalisme dont le niveau le plus haut est le formalisme de la théorie des systèmes. Dans ce sens, nous pouvons établir une relation, entre les formalismes et les abstractions qu'engendrent leurs utilisations

respectives et montrer quand un type de système peut faire ce que fait un autre système. Un formalisme est une sous-classe d'un autre formalisme si le premier peut être *simulé* par le second. Par exemple B.P. Zeigler a montré qu'un système à temps discret peut être simulé par un système à évènement discret en rendant l'avance du temps dépendante d'une constante. Ramener un formalisme à celui de la théorie des systèmes permet également d'être indépendant du language d'implémentation et de la plateforme d'exécution. Cela répond au problème de la combinaison de différents formalismes pour l'étude des systèmes complexes, aussi appelé multi-modélisation. Cet aspect de relations entre les formalismes de spécification des systèmes est également abordé à l'université de Berkeley dans le projet Ptolemy [Lee 98]. Dans Ptolemy, les formalismes sont appelés, des domaines (e.g. temps-continu, flow de données, évènements discrets, machines à état fini, flow de données synchrones, temps discret, temps périodique...). Chaque domaine utilise un "model of computation", un ensemble de règles et de mécanismes qui gouvernent les interactions entre les composants. Ces "models of computation" sont ce que B.P. Zeigler appelle les simulateurs abstraits.

4.1.2.1 Définition formelle d'un système

Définition 4.1 (Système)**.** Un système est une struture

$$S =< T, X, \Omega, Y, Q, \Delta, \Lambda > \text{ où}$$

- T est la base de temps,
- X est l'ensemble des valeurs d'entrée,
- $\Omega \subseteq (X, T)$ est l'ensemble des segments d'entrées acceptables,
- Y est l'ensemble des valeurs de sortie,
- Q est l'ensemble des états,
- $\Delta : Q \times \Omega \to Q$ est la fonction de transition,
- $\Lambda : Q \times X \to Y$ est la fonction de sortie.

T peut aussi bien appartenir à l'ensemble des réels $\mathbb{R}$, à l'ensemble des entiers $\mathbb{N}$ ou à un facteur c de l'ensemble $\mathbb{N}$ (système échantilloné).

Définition 4.2. Une trajectoire (ou un signal) est un ensemble de valeurs successives dans le temps appliquées à un ensemble, par exemple à l'ensemble d'entrée

X du système. La restriction de cette trajectoire sur un intervalle de temps noté $< t_1, t_2 >$ est appelée un segment, noté $\omega :< t_1, t_2 >\rightarrow X$ ou $\omega_{<t_1,t_2>}$. Si x est une entrée du système, $x \in X$, un segment $\omega_x :< t_1, t_2 >\rightarrow X$ décrit une séquence de valeurs $\omega_x(t) = \omega_x(t_1), \omega_x(t_1 + 1), ..., \omega_x(t_2)$ appliquée à la variable d'entrée x.

Un segment (comme un système) peut :

- être continu sur une base de temps continue,
- représenter une séquence d'évènements ordonnés sur une base de temps continue,
- représenter une séquence de valeurs sur une base de temps discrète (figure 4.4).

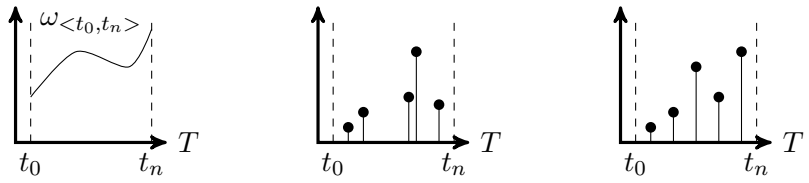

FIGURE 4.4 – Types de segment

L'introduction de la base de temps et de la trajectoire (ou du segment) nous permet de dire qu'un système est une entité caractérisée par un ensemble d'états Q ordonnés sur une base de temps T qui réagit à un ensemble de segments d'entrée ω sur X, provenant de son environnement. L'état suivant du système est déterminé en fonction de l'état présent et d'un segment d'entrée par la fonction de transition Δ. La fonction de sortie Λ définit une trajectoire de sortie ρ sur Y étant donné un état courant et un segment d'entrée. La figure 4.5 illustre cette interprétation.

Remark 4.1. La trajectoire de sortie peut être uniquement fonction de l'état présent (machine de Moore).

Prenons la figure 4.6. Supposons qu'un système soit dans un état $q \in Q$ à l'instant t_1 et qu'un segment d'entrée $\omega :< t_1, t_2 >\rightarrow X$ soit appliqué à l'ensemble d'entrée X du système. Le système réagit à ce segment en appliquant la fonction de transition menant à l'état *final* $\Delta(q, \omega)$ à l'instant t_2. La sortie observée à l'instant t_2 est, quand à elle, définie par la fonction de sortie tel que $y_2 = \Lambda(\Delta(q, \omega), \omega(t_2))$.

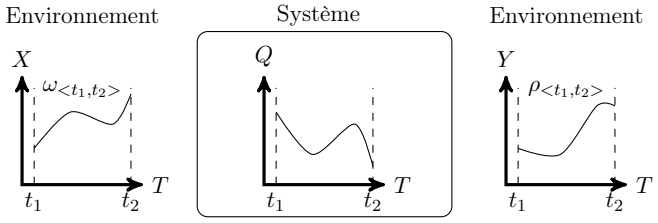

FIGURE 4.5 – L'expérimentation d'un système

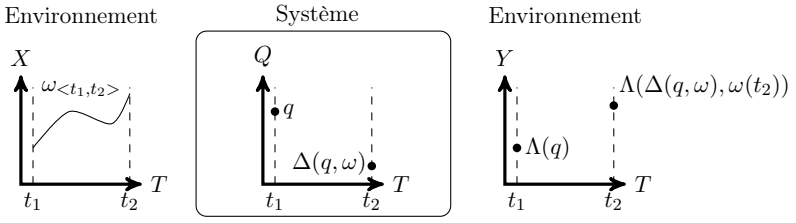

FIGURE 4.6 – Mécanisme dynamique d'un système

4.1.2.2 Hiérarchie de spécification des systèmes

Le deuxième aspect de la théorie de la M&S définit des niveaux de spécification pour lesquels nous pouvons décrire comment un système se comporte et les mécanismes qui régissent ce comportement. Allant du comportement vers la structure, cette hiérarchie de niveaux est proche des concepts de boîte noire/boîte blanche. Le niveau 0 appelé *cadre d'observation*, décrit comment stimuler le système avec des entrées, quelles variables mesurer, et comment les observer à travers une base de temps. Le niveau 1 appelé *relation d'entrée/sortie*, met en relation les entrées du système avec ses sorties et produit ainsi une collection de paires d'entrée/sortie qui caractérise le comportement du système. À ce niveau, une même entrée peut affecter des sorties différentes. La sortie du système n'est donc pas prédictible à partir des observations de ses entrées. Le niveau 2 appelé *fonction d'entrée/sortie* permet de lever cet indéterminisme par la connaissance de l'état initial du système. Etant donné un état donné et une entrée appliquée au système, une sortie unique est produite. Le niveau 3 appelé *transition d'état*, décrit comment les états du système sont affectés par les évènements externes et comment le système répond à ces évènements en sortie. Le niveau 4 appelé *couplage de composant*, décrit un système sous forme

de composants qui interagissent. Comme déjà évoqué dans le chapitre précédent, le passage vers un plus haut niveau d'abstraction (e.g. transition d'état vers fonction d'entrée/sortie) est une abstraction par simplification de la spécification du système. Cette relation est définie par B.P. Zeigler comme une association. Le passage vers un plus haut niveau d'abstraction est appelé "réalisation", le passage vers un plus bas niveau est appelé "implémentation".

La hiérarchie de spécification des systèmes propose différents niveaux pour lesquels nous pouvons décrire comment ils se comportent et les mécanismes qui régissent ces systèmes.

Niveau	Spécification	Structure formelle
0	Cadre d'observation	$\langle T, X, Y \rangle$
1	Relation d'E/S	$\langle T, X, \Omega, Y, R \rangle$
2	Fonction d'E/S	$\langle T, X, \Omega, Y, F \rangle$
3	Transition d'état	$\langle T, X, \Omega, Y, Q, \Delta, \Lambda \rangle$
4	Spécification itérative	$\langle T, X, \Omega_G, Y, Q, \delta, \lambda \rangle$
5	Système structuré	$\langle T, X, \Omega, Y, Q, \Delta, \Lambda \rangle$ avec X, Y, Q, Δ, Λ structurés
6	Système multicomposants	$\langle T, X, \Omega, Y, D, \{M_d = (Q_d, E_d, I_d, \Delta_d, \Lambda_d)\} \rangle$
7	Couplage	$\langle T, XN, YN, D, \{M_d \vert d \in D\}, \{I_d, Z_d \vert d \in D \cup \{N\}\} \rangle$

4.1.2.3 Relation de morphisme

B.P Zeigler a établi des relations qui placent les éléments de deux descriptions d'un système en correspondance et ce à chaque niveau de la hiérarchie de spécification (figure 4.7). Rappelons que cette relation, appelée *relation de préservation* ou *morphisme d'un système*, établit une correspondance entre un système "concret" et un système "abstrait". Considérons, figure 4.8, deux systèmes S et S' tel que S est plus grand que S' en termes de nombre d'états. Il y a eu morphisme entre S et S' par aggrégation d'états. À chaque état du système S' une correspondance est établie, illustrée par les flèches en gras, avec un état ou un ensemble d'états du système S. Cette correspondance est appelée un *homomorphisme* si, lorsque S'

réalise une transition, par example $a \stackrel{\alpha'}{\rightarrow} b$ alors S réalise la transition concernant les états correspondants, i.e. $A \stackrel{\alpha}{\rightarrow} B$. Nous dirons qu'il y a préservation de la fonction de transition. Similairement, un homomorphisme implique la préservation de la fonction de sortie. Cela signifie que la sortie produite par le système S' à un état q, par exemple à l'état b, doit être la même que la sortie produite par le système S à l'état correspondant, ici B. Si deux systèmes sont homomorphiques, alors leurs comportements sont équivalents.

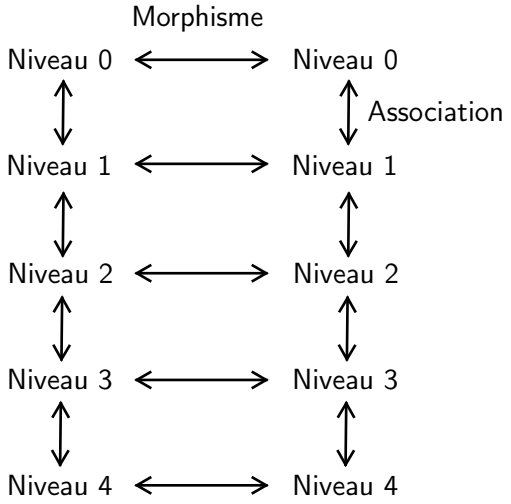

FIGURE 4.7 – Association et morphisme

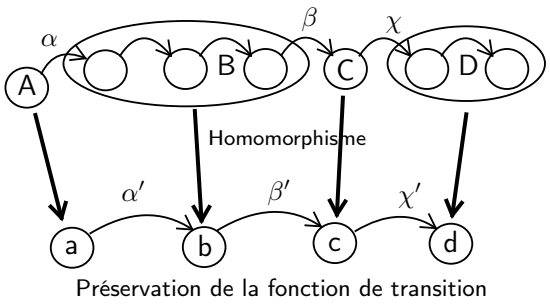

FIGURE 4.8 – Abstraction par morphisme

4.1.2.4 Conclusion sur la théorie de la M&S

À chacune des relations qui ont été définies dans la théorie de M&S (relation entre formalisme, association et morphisme), une notion de validité peut alors être associée. La validité de la représentation d'un système réel, la validité de la description à un niveau de spécification par rapport à un autre niveau et la validité entre un modèle concret et un modèle abstrait. Rappelons que dans le cadre de notre étude, nous ne nous intéressons pas à la validité due au formalisme utilisé puisque le périmètre de l'étude a été réduit aux systèmes à temps discrets et aux systèmes à évènements discrets. Le système réel est donc, dans ce cas précis, déjà soumis aux abstractions dues à la forme du modèle. De plus, nous considérons que c'est l'utilisateur de la simulation qui spécifie à quel niveau de la hiérarchie de spécification se situe son besoin. C'est à travers le niveau de détail de ses exigences de M&S que nous pourrons définir le niveau de spécification auquel nous nous intéressons. Nous proposons donc une hiérarchie de propriétés d'une simulation basée sur la hiérarchie de spécification des systèmes à travers laquelle, l'utilisateur d'une simulation décrit les caractéristiques attendues et les abstractions autorisées de la simulation et le développeur décrit les caractéristiques de la simulation qu'il fournit. Par des techniques de mise en correspondance, nous établissons des critères d'acceptation d'un modèle de simulation pour un objectif donné.

4.2 Formalisation des règles de compatibilité

Nous proposons dans cette section, la formalisation des propriétés d'abstraction de notre taxonomie sur la structure algébrique d'un système. Puis nous proposons des critères d'acceptabilité selon chaque propriétés [Albert 09b] [Albert 09a].

4.2.1 Définition de la mise en correspondance entre un SOU et un SDU

Soit un ensemble de cadres expérimentaux EF et $\mathscr{A}$ un ensemble de valeurs d'attributs (e.g. bool, string...). Une propriété $\mathscr{P}$ est une fonction $\mathscr{P} : EF \rightarrow 2^{\mathscr{A}}$, où $2^{\mathscr{A}}$ est l'ensemble des parties de $\mathscr{A}$. Donc une propriété affecte un attribut spécifique à un cadre expérimental $ef \in EF$, à savoir l'attribut $\mathscr{P}(ef) \subseteq \mathscr{A}$. Par

exemple, une propriété est une fonction $\mathscr{P}$ qui affecte à un cadre expérimental ef des noms de variables d'entrée, e.g. $\mathscr{P}(ef) = altitude, MachNumber$.

Soit $SOU \in EF$ et $SDU \in EF$ deux composants dont la signature et la spécification sont respectivement caractérisées par $\mathscr{P}(SOU)$ et $\mathscr{P}(SDU)$, le problème d'applicabilité d'un SOU à un SDU est défini par :

$$Applicability : \mathscr{P}(SOU), \mathscr{C}, \mathscr{P}(SDU) \rightarrow \{0, 1\}$$

La signification concrète de cette relation peut être énoncée de la façon suivante. Soit une propriété sur un cadre expérimental SOU qui caractérise un ensemble de scénarios de simulation et les abstractions acceptables pour atteindre un objectif d'utilisation, une propriété sur un cadre expérimental SDU qui caractérise le domaine d'usage d'un produit de simulation, et $c \in \mathscr{C}$, un critère d'acceptabilité, alors la relation *Applicability* retourne la valeur "1" si le SOU est applicable à SDU pour ce critère sinon "0".

4.2.2 Critères d'applicabilité

4.2.2.1 Périmètre

Le premier critère d'applicabilité consiste à s'assurer que le périmètre du produit de M&S couvre le périmètre requis par l'objectif d'utilisation de la simulation.

Au plus bas niveau de la hiérarchie de spécification, un cadre expérimental est défini par la structure

$$EF =< T, X_{EF}, Y_{EF} > \text{ où}$$

- T est la base de temps
- X_{EF} est l'ensemble des variables d'entrée
- Y_{EF} est l'ensemble des variables de sortie

Définition 4.3 (Information)**.** Nous définissons un ensemble de type K contenant l'information véhiculée par une variable. Nous considérons que K ne contient que des noms, e.g. $K = \{altitude, speed, energy, power, mach\}$. La fonction $information_i : X_i \cup Y_i \rightarrow K$ associe un élément de K sur les variables d'entrée/sortie d'un composant i. L'operation $information_i(x)$ permet de récupérer l'information associée à la variable $x \in X_i$, e.g. $information_i(x) = altitude$.

Définition 4.4. Le périmètre d'un cadre expérimental EF est défini par l'ensemble des variables d'entrée X_{EF} qui agissent sur le système et l'ensemble des variables de sortie Y_{EF} qui sont contrôlés par le système :

$$scope_{EF} = information_{EF}(X \cup Y)$$

Propriété 4.1 (Compatibilité du périmètre)**.** *SOU est applicable à SDU selon leur périmètre respectif si et seulement si* $scope_{SOU} \subseteq scope_{SDU}$ *au nom de l'information véhiculée par la variable près.*

Exemple Considérons le cadre expérimental d'une expérimentation défini par :

$$SOU = \langle T, X_{SOU}, Y_{SOU} \rangle$$

avec

- $T = c \bullet \mathbb{N}$ par exemple,
- $X_{SOU} = \{x_{SOU1}, x_{SOU2}\}$ l'ensemble des variables observées,
- $Y_{SOU} = \{y_{SOU1}, y_{SOU2}\}$ l'ensemble des stimuli injectés.

Considérons un modèle de simulation dont le domaine d'usage est défini par :

$$SDU = \langle T, X_{SDU}, Y_{SDU} \rangle$$

avec

- $T = c \bullet \mathbb{N}$ par exemple,
- $X_{SDU} = \{x_{SDU1}, x_{SDU2}\}$ l'ensemble des variables d'entrée,
- $Y_{SDU} = \{y_{SDU1}, y_{SDU2}, y_{SDU3}\}$ l'ensemble des variables de sortie.

Le produit de simulation correspondant est présenté figure 4.9a. ci-dessous. Les figures b., c. et d. illustrent les autres configurations possibles et leurs conséquences sur l'applicabilité. Pour des raisons de simplicité, nous considérons que les noms utilisés pour désigner l'information véhiculée par les variables du SOU et du SDU sont identiques, i.e.

$$information(x_{SOU1}) = information(y_{SDU1})$$

$$information(x_{SOU2}) = information(y_{SDU2})$$

$$information(y_{SOU1}) = information(x_{SDU1})$$

$$information(y_{SOU2}) = information(x_{SDU2})$$

Dans le cas de la figure 4.9d., nous considérons que le périmètre du *SOU* et applicable au périmètre du *SDU*. Cependant, il faut s'assurer qu'il n'y ai pas de dépendance entre x_{SDU3} et y_{SDU1} ou y_{SDU2}, auquel cas les résultats de simulation pourraient être biaisés. Les dépendances entre les entrées/sorties d'un composant sont étudiées par la propriété *relation d'E/S* (section 4.2.2.5).

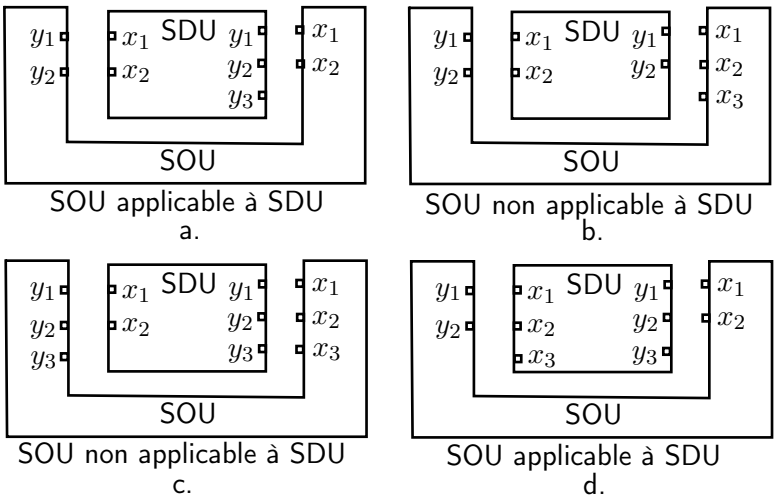

FIGURE 4.9 – Périmètre d'un produit de simulation

4.2.2.2 Topologie

La topologie d'un système est une description sous forme d'une organisation structurée et hiérarchique des composants internes du système (fonctions, composants logiques) et du couplage entre ces composants et l'environnement. Des fonctions de couplage sont utilisées pour décrire comment les composants sont connectés entre eux. Elles spécifient comment les valeurs d'entrée d'un composant sont dérivées des valeurs de sortie du ou des composants qui l'influence. Une fonction de couplage, notée $z_{A \to B}(x, y)$, signifie que A est le composant influenceur et B est le composant influencé. A et B sont connectés par leur port respectif x et y. Donc la valeur de y dépend de x.

Définition 4.5. Le cadre expérimental d'une expérimentation est défini par la structure :

$$SOU =< T, X_{SOU}, Y_{SOU}, D_{SOU}, \{M_d\}, \{I_d\}, \{Z_{SOU\,i\to d}\}, Z_{SOU\to SDU}, Z_{SDU\to SOU} >$$

où

- T est la base de temps,
- X_{SOU} est l'ensemble des variables d'entrée du cadre expérimental,
- Y_{SOU} est l'ensemble des variables de sortie du cadre expérimental,
- D_{SOU} est l'ensemble des composants tel que :

  pour tout $d \in D_{SOU}$,

  $$M_d =< T, X_d, Y_d, >$$

  , pour tout $d \in D_{SOU} \cup \{SOU\}$,
  - $I_d \subseteq D_{SOU} \cup \{SOU\}$ est l'ensemble des composants qui influencent d
  - $Z_{SOU\,i\to d}$ est l'ensemble des fonctions de couplage du cadre expérimental entre un composant $i \in I_d$ et un composant d tel que :
    - si $i = SOU, Z_{SOU\,i\to d} : X_i \to X_d$ est la fonction de couplage d'entrée,
    - si $i \neq SOU$ et $d \neq SOU, Z_{SOU\,i\to d} : Y_i \to X_d$ est la fonction de couplage interne,
    - si $d = SOU, Z_{SOU\,i\to d} : Y_i \to Y_d$ est la fonction de couplage de sortie,
- $Z_{SOU\to SDU} : Y_{SOU} \to X_c$, est la fonction de couplage qui désigne des points de stimulation particuliers sur un composant $c \in D_{SDU} \cup SDU$,
- $Z_{SDU\to SOU} : Y_c \to X_{SOU}$, est la fonction de couplage qui désigne des points d'observation particuliers sur un composant $c \in D_{SDU} \cup SDU$.

Définition 4.6. Le domaine d'usage d'un modèle de simulation est défini par la structure :

$$SDU =< T, X_{SDU}, Y_{SDU}, D_{SDU}, \{M_d\}, \{I_d\}, \{Z_{SDU\,i\to d}\} >$$

où

- T est la base de temps,
- X_{SDU} est l'ensemble des variables d'entrée du modèle de simulation,
- Y_{SDU} est l'ensemble des variables de sortie du modèle de simulation,
- D_{SDU} est l'ensemble des composants tel que :

  pour tout $d \in D_{SDU}$,

  $$M_d =< T, X_d, Y_d >$$

, pour tout $d \in D_{SDU} \cup \{SDU\}$,

- $I_d \subseteq D_{SDU} \cup \{SDU\}$ est l'ensemble des composants qui influence d,
- $Z_{SDU\,i \to d}$ est l'ensemble des fonctions de couplage du produit de simulation entre un composant $i \in I_d$ et un composant d tel que :
  - si $i = SDU, Z_{SDU\,i \to d} : X_i \to X_d$ est la fonction de couplage d'entrée,
  - si $i \neq SDU$ et $d \neq SDU, Z_{SDU\,i \to d} : Y_i \to X_d$ est la fonction de couplage interne,
  - si $d = SDU, Z_{SDU\,i \to d} : Y_i \to Y_d$ est la fonction de couplage de sortie.

Propriété 4.2 (Compatibilité de la topologie)**.** *SOU est applicable à SDU selon leur topologie respective si et seulement si*

$$\textit{pour tout } z \in Z_{SOU \to SDU} : Y_{SOU} \to X_c \textit{ tel que } c \neq SDU$$

$$\exists Z_{SDU\,i \to d} : X_i \to X_d \textit{ tel que } i = SDU \textit{ et } c = d$$

$$\textit{pour tout } z \in Z_{SDU \to SOU} : Y_c \to X_{SOU} \textit{ tel que } c \neq SDU$$

$$\exists Z_{SDU\,i \to d} : Y_i \to Y_d \textit{ tel que } d = SDU \textit{ et } c = i$$

Exemple Soit le produit de simulation illustré par la figure 4.10. Le modèle de simulation SDU est un modèle couplé constitué des composants a et b. $Z_A(x_1) = x_{a1}$, $Z_B(x_2) = x_{a2}$ et $Z_C(x_2) = x_{b2}$ sont les fonctions de couplage d'entrée, $Z_D(y_{a2}) = y_1$ et $Z_E(y_b) = y_2$ sont les fonctions de couplage de sortie et $Z_F(y_{a3}) = x_{b1}$ est la fonction interne du composant SDU. Le cadre expérimental de l'expérimentation SOU est aussi un modèle couplé constitué d'un générateur, d'un accepteur et d'un transducteur. $Z_{i \to d}$ est la fonction de couplage interne, $Z_{d \to SOU}$ est la fonction de couplage de sortie et $Z_{SOU \to d}$ est la fonction de couplage d'entrée du composant SOU. Les périmètres du SOU et du SDU sont compatibles puisque $scope_{SOU} = scope_{SDU}$. Z_G et Z_H sont les fonctions de couplage qui désignent les points de stimulation. Z_I et Z_J sont les fonctions de couplage qui désignent les points d'observation. Dans cet exemple, $Z_G(y_1) = x_1$ tel que $y_1 \in Y_{SOU}$ et $x_1 \in X_c$ avec $c = SDU$. Il n'y a donc pas de problème particulier pour cette fonction de couplage. Même chose pour $Z_H(y_2) = x_2$ tel que $y_2 \in Y_{SOU}$ et $x_2 \in X_c$ avec $c = SDU$ et pour $Z_I(y_1) = x_1$ tel que $y_1 \in Y_c$ et $x_1 \in X_{SOU}$ avec $c = SDU$. Supposons

que $concept(y_{a1}) = concept(y_b) = concept(y_2) = concept(x_2)$ tel que $y_2 \in Y_{SDU}$ et $x_2 \in X_{SOU}$, e.g. une altitude. La topologie du cadre expérimental définit une fonction $Z_J(y_{a1}) = x_2$ (ligne pointillée sur la figure). La topologie du modèle de simulation ne permet pas l'observation de y_2 puisqu'il n'existe pas de fonction de couplage de sortie qui connecte y_{a1} avec y_2. En revanche, d'après le domaine d'usage du modèle de simulation l'altitude peut être observée à partir du composant b.

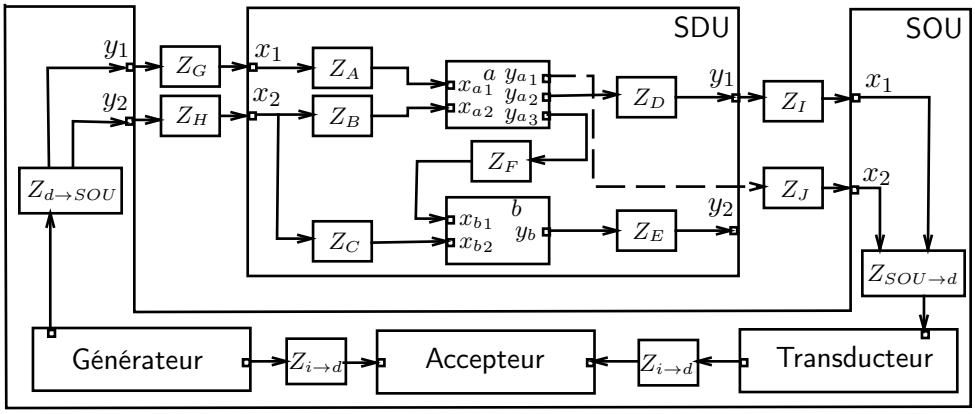

FIGURE 4.10 – Topologie d'un produit de simulation

4.2.2.3 Domaine, Unité, Résolution, Précision

Les propriétés suivantes sont toutes basées sur le principe de signature de type affectée aux variables d'entrée/sortie des composants SOU et SDU. Les critères d'acceptabilité sont ensuite définis par des règles de mise en correspondance entre les signatures.

Définition 4.7 (Unité)**.** Soit un ensemble de type SU qui contient les unités du système international et des unités spécifiques au domaine. Nous considérons que l'ensemble SU ne contient que des noms, e.g. $SU = \{m, km, ft, V, FlightLevel, ^\circ C\}$. La fonction $unit_i : X_i \cup Y_i \to SU$ associe un élément de SU aux variables d'entrée/sortie d'un composant i.

Définition 4.8 (Domaine)**.** Soit un ensemble de type Rg qui contient le domaine de valeurs que peuvent prendre les variables. Par exemple la température de l'air varie entre $[-60^\circ C; 99^\circ C]$, une altitude est limitée par $[-1000ft; +50000ft]$ ou un

numéro de Mach est compris dans l'intervalle $[0.10Mach; 1.0Mach]$. La fonction $range_i : X_i \cup Y_i \rightarrow Rg$ associe un élément de Rg aux variables d'entrée/sortie d'un composant i.

Définition 4.9 (Résolution)**.** Soit un ensemble de type Sp qui contient des résolutions. Rappelons que la résolution est la distance entre deux valeurs successives d'une variable. Par exemple, une température de l'air comprise dans l'intervalle $[-60^{\circ}C; 99^{\circ}C]$, peut, avec une résolution de $0.25^{\circ}C$ ne prendre que les valeurs $-60^{\circ}C, -59.75^{\circ}C, -59.5^{\circ}C, \ldots, 98.5^{\circ}C, 98.75^{\circ}C, 99^{\circ}C$. Une altitude limitée par un domaine de $[-1000ft; +50000ft]$ et définie par une résolution de $1foot$ peut prendre les valeurs $-1000ft - 999ft, -998ft, \ldots, 49999ft, 50000ft$. Un numéro de Mach dont le domaine est $[0.10Mach; 1.0Mach]$ et la résolution est 0.0000625 peut prendre les valeurs $0.1000625, 1.001275, \ldots, 0.9999375, 1.0$. La fonction $resolution_i : X_i \cup Y_i \rightarrow Sp$ associe un élément de Sp aux variables d'entrée/sortie d'un composant i.

Définition 4.10 (Précision)**.** Soit un ensemble de type P qui contient des précisions. Rappelons que la précision est le nombre de valeurs que peut prendre un échantillon. Par exemple une température de l'air qui varie entre -60°C et 99°C, codée sur 9 bits peut prendre une valeur parmi les 512 valeurs possibles par échantillonnage soit une précision de $\frac{(60+99)}{512} \approx 0.31^{\circ}C$. La fonction $precision_i : X_i \cup Y_i \rightarrow P$ associe un élément de P aux variables d'entrée/sortie d'un composant i.

Propriété 4.3 (Compatibilité du type mathématique)**.** *SOU est applicable à SDU selon leurs types mathématiques respectifs si et seulement si*

$$\textit{pour tout } z \in Z_{SOU \rightarrow SDU} : y \rightarrow x \textit{ tel que } y \in Y_{SOU} \textit{ et } x \in X_{SDU} \Rightarrow$$

$$unit_{SOU}(y) = unit_{SDU}(x)$$

$$\wedge range_{SOU}(y) \subset range_{SDU}(x)$$

$$\wedge resolution_{SOU}(y) \subset resolution_{SDU}(x)$$

$$\wedge precision_{SOU}(y) \leq precision_{SDU}(x)$$

$$\textit{pour tout } z \in Z_{SDU \rightarrow SOU} : y \rightarrow x \textit{ tel que } y \in Y_{SDU} \textit{ et } x \in X_{SOU} \Rightarrow$$

$$unit_{SOU}(y) = unit_{SDU}(x)$$

$$\wedge range_{SOU}(y) \supset range_{SDU}(x)$$

$$\wedge resolution_{SOU}(y) \supset resolution_{SDU}(x)$$

$$\wedge precision_{SOU}(y) \geq precision_{SDU}(x)$$

4.2.2.4 Datatype

Lorsque une variable est encodée dans un référentiel digital, des types associés au langage de programmation lui sont affectés tel qu'un entier, une chaîne de caractère, un booléen...

Définition 4.11 (Datatype)**.** Soit un ensemble de type T contenant des datatypes standards et des datatypes spécifiques à un langage de modélisation ou de programmation, e.g. $T = \{int, complex, long, double, AFDX, A429, string\}$. La fonction $datatype_i : X_i \cup Y_i \rightarrow T$ associe un élément de T aux variables d'entrée/sortie d'un composant i.

L'exacte mise en correspondance n'étant pas obligatoire, nous utilisons un treillis de types de donnée [Xiong 02] tel que l'illustre la figure 4.11. Le treillis établit un ordre partiel entre les types de donnée, une relation du haut vers le bas dénote $\sqsubseteq$, une perte d'information. Par exemple un entier codé sur 32 bits peut être converti en un double sans perte d'information alors que le contraire n'est pas vrai (double $\sqsubset$ int).

Propriété 4.4 (Compatibilité du type informatique)**.** *SOU est applicable à SDU selon leur type informatique respectif si et seulement si le type de la variable "influence" est le même ou, à un plus bas niveau dans la lattice, que le type de la variable influencée.*

$$pour\ tout\ z \in Z_{SOU \rightarrow SDU} : y \rightarrow x\ tel\ que\ y \in Y_{SOU}\ et\ x \in X_{SDU}$$

$$\Rightarrow datatype_{SOU}(y) \sqsupseteq datatype_{SDU}(x)$$

$$pour\ tout\ z \in Z_{SDU \rightarrow SOU} : y \rightarrow x\ tel\ que\ y \in Y_{SDU}\ et\ x \in X_{SOU}$$

$$\Rightarrow datatype_{SOU}(y) \sqsubseteq datatype_{SDU}(x)$$

FIGURE 4.11 – Un exemple de treillis de types de donnée

4.2.2.5 Relation d'E/S

La propriété de relation d'E/S consiste à décrire de manière abstraite des flots d'écoulement également appelés chemins d'information [Nguyen 02] formés par la relation causale entre les paramètres. Le but de ce critère de compatibilité est de vérifier que les relations entre un ensemble de stimuli et un ensemble d'observateurs nécessaires à la satisfaction de l'objectif sont représentées par un comportement dans le modèle de simulation utilisé.

Définition 4.12. Le cadre expérimental d'une expérimentation est défini par la structure :

$$SOU =< T, X_{SOU}, Y_{SOU}, D_{SOU}, \{M_d\}, \{I_d\}, \{Z_{SOU\,i\to d}\}, Z_{SOU\to SDU}, Z_{SDU\to SOU}, Z_{SOU} >$$

où

- T,X_{SOU},Y_{SOU},D_{SOU},M_d,I_d,$Z_{SOU\,i\to d}$, $Z_{SOU\to SDU}$ et $Z_{SDU\to SOU}$ sont définis comme précédemment,
- $Z_{SOU} : Y_{SOU} \to X_{SOU}$, est la fonction de couplage qui lie un stimuli à un observateur.

Définition 4.13. Le domaine d'usage d'un modèle de simulation est défini par la structure :

$$SDU =< T, X_{SDU}, Y_{SDU}, D_{SDU}, \{M_d\}, \{I_d\}, \{Z_{SDU\,i\to d}\}, Z_{SDU} >$$

où

- T,X_{SDU},Y_{SDU},D_{SDU},M_d,I_d et $Z_{SDU\,i\to d}$ sont définis comme précédemment,
- $Z_{SDU\,d} : X_d \to Y_d$ est la fonction de couplage qui lie une entrée du composant d à une sortie du composant d.

Propriété 4.5 (Compatibilité des relation d'E/S)**.** *SOU est applicable à SDU selon leur relation d'E/S respective si et seulement si*

$$\textit{pour tout } z \in Z_{SOU} : y \to x \textit{ tel que } y \in Y_{SOU} \textit{ et } x \in X_{SOU}$$

$$\exists z_{SDU} \in Z_{SDU\,d} : z_{SOU\to SDU}(y) \to z_{SDU\to SOU}(x)$$

$$\textit{tel que } d = SDU, z_{SOU\to SDU} \in Z_{SOU\to SDU} \textit{ et } z_{SDU\to SOU} \in Z_{SDU\to SOU}$$

Remark 4.2. D'après la topologie, les fonctions $z_{SOU\to SDU}(y)$ et $z_{SDU\to SOU}(x)$ retournent respectivement la variable d'entrée du modèle de simulation connectée à y_{SOU} et la variable de sortie du modèle de simulation connectée à x_{SOU}.

Exemple Reprenons l'exemple que nous avons présenté dans la section topologie 4.2.2.2. Le domaine d'usage du modèle de simulation est enrichi afin de spécifier les fonctions de couplages d'E/S : $Z_{b1}(x_{b1}) = (y_b)$ et $Z_{b2}(x_{b2}) = (y_b)$, représentées par des lignes sur la figure. Nous considérons que le SOU est applicable au SDU selon leur topologie respective (figure 4.12). Soit un objectif d'utilisation qui consiste à stimuler l'entrée $x_2 \in X_{SDU}$ et à observer la sortie $y_2 \in Y_{SDU}$. Or

$$z_{SOU\to SDU}(y_2) = x_2 \text{ tel que } x_2 \in X_{SDU}$$

$$z_{SOU \to SDU}(x_2) = y_2 \text{ tel que } y_2 \in Y_{SDU}$$

Cet objectif est donc définit par la fonction de couplage suivante :

$$Z_{SOU} = \{z_1(y_2, x_2)\} \text{ tel que } y_2 \in Y_{SOU} \text{ et } x_2 \in X_{SOU}$$

Donc, pour être compatible selon les relations d'E/S, le domaine d'usage du modèle de simulation doit avoir une fonction de couplage d'entrée/sortie :

$$Z_{SDU1}(x_2) = (y_2)$$

Nous utilisons des opérateurs logiques afin de composer des fonctions de couplage. L'opérateur logique "et", noté $\otimes$, désigne une composition série et l'opérateur logique "ou", noté $\oplus$, désigne une composition parallèle. Nous utilisons des dépendances booléenne, i.e. si une fonction de couplage z existe alors $z = true$ sinon $z = false$. Donc étant donné les couplages internes, les couplages de sorties, les couplages d'entrée et les couplages d'entrée/sortie des composants, nous pouvons déterminer les fonctions de couplage d'entrée/sortie d'un modèle de simulation.
$Z_{SDU1} = ((Z_B \otimes Z_{a1} \otimes Z_F \otimes Z_{b1}) \oplus (Z_C \otimes Z_{b2})) \otimes Z_E$
$Z_{SDU1} = ((true \otimes false \otimes true \otimes true) \oplus (true \otimes true) \otimes true$
$Z_{SDU1} = true$

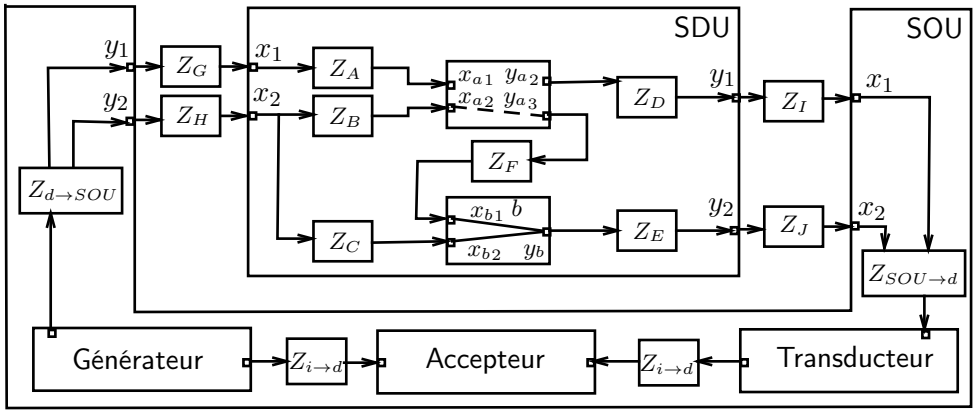

FIGURE 4.12 – Relation d'E/S d'un produit de simulation

4.2.2.6 Interaction dynamique

Définitions préliminaires Nous proposons d'utiliser les IOA (I/O Automata) [Alfaro 01] pour décrire le comportement dynamique des composants.

Définition 4.14 (I/O Automata)**.** Un IOA est un n-uplet $\langle D, I, O, dom, S, \sigma, \Sigma, \delta, S_0 \rangle$ où :

- D est un ensemble de types de donnée (entier, réel, string...),
- I est un ensemble fini de noms qui représente l'ensemble des variables d'entrée,
- O est un ensemble fini de noms qui représente l'ensemble des variables de sorties,
- $dom : I \cup O \rightarrow D$ est la fonction de typage qui alloue un type de donnée à chaque variable,
- S est l'ensemble des états,
- $\sigma : S \rightarrow ((I \cup O) \rightarrow dom(I \cup O))$ est la fonction de configuration i.e. la correspondance des valeurs aux variables typées d'entrée/sortie à chaque état.
- Σ est l'ensemble des noms d'évènements. Ces noms sont les étiquettes des transitions d'état. ε est l'ensemble des évènements non observables.
- $\delta \subset S \times \Sigma \times S$ est la relation de transition, une transition identifie un changement sur les entrées, sur les sorties ou sur les états.
- $S_0 \subset S$ est l'ensemble des états initiaux possibles, les états dans lesquels l'automate peut être si aucune entrée n'a encore été transformée.

$I \cup O \cup \Sigma$ est le vocabulaire utilisé pour définir l'automate. Pour des raisons de simplicité, nous supposons que les noms utilisés dans le vocabulaire sont uniques.

Exemple A titre d'exemple, considérons un système de communication à mémoire partagée dont le comportement est décrit par l'IOA *buffer* figure 4.13. Dans cet exemple, une variable `queue` est définie pour décrire une séquence de messages. Cette variable est déclarée comme un tableau de longueur `len` pouvant contenir des messages m de type id_m avec une valeur initial `empty`. Cet automate est constitué d'un état initial s_0 où la séquence est vide et d'un état s_1 où la séquence est non vide. Des relations de transitions relient ces états. Ces transitions sont déclenchées par deux types d'évènements :

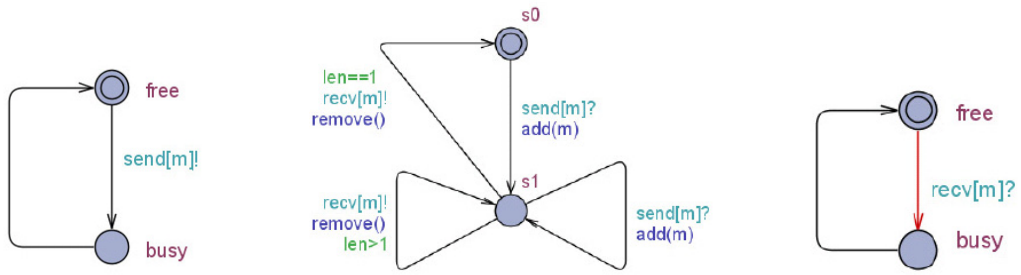

FIGURE 4.13 – Automate d'un système de communication à mémoire partagée

- `send(m)` est un évènement d'entrée, où chaque message est ajouté à la tête de la séquence `queue` grâce à la fonction de configuration `add(m)`,
- `recv(m)` est un évènement de sortie, où chaque message est supprimé de la séquence `queue` grâce à la fonction de configuration `remove()`.

Les fonctions de configuration modifient la valeur de la variable `queue` à chaque état. Par exemple si `len=2`, `queue` peut prendre les valeurs $\{(0,0),(id_{m1},0),(id_{m1},id_{m2}),(id_{m2},0)$.

Dans la théorie des IOA, nous pouvons synchroniser un évènement de sortie d'un IOA avec un évènement d'entrée d'un autre IOA, si ces évènements ont le même noms. Par exemple, si l'évènement de sortie `send(m)` apparaît dans un IOA, alors cet évènement peut être synchronisé avec l'évènement d'entrée `send(m)` d'un autre IOA. Par exemple, la figure 4.13, illustre deux autres IOAs, un *client* et un *serveur* qui échangent des messages via *buffer*.

Définition 4.15 (Restriction d'un automate)**.** La restriction d'un automate $\langle D, I, O, dom, S, \sigma, \Sigma, \delta, S_0\rangle$ à un sous-ensemble $I' \cup O' \cup \Sigma'$ de son vocabulaire est un automate $\langle D, I', O', dom, S, \sigma/I'\cup O', \Sigma', \delta/\Sigma', S_0\rangle$ où les configurations et les transitions restreintes sont définies ci-dessous.

Définition 4.16 (Restriction d'une configuration)**.** La restriction $\sigma/I' \cup O'$ d'une configuration $\sigma : S \to ((I \cup O) \to dom(I \cup O))$ restreinte à un vocabulaire $I' \cup O'$ tel que $I' \cup O' \subset I \cup O$ est la configuration $\sigma'/I' \cup O' : S \to ((I' \cup O') \to dom(I' \cup O'))$ telle que $\sigma/I' \cup O'(s)(x) = \sigma(s)(x)$ pour tout $x \in I' \cup O'$.

Intuitivement, la restriction d'une configuration garde en chaque état la valeur d'un sous-ensemble de variables d'intérêt et laisse de coté les valeurs de toutes les autres variables.

Définition 4.17 (Restriction d'une transition)**.** La restriction δ/Σ' d'une relation de transition $\delta \subset S \times \Sigma \times S$ restreinte à un vocabulaire $\Sigma' \subset \Sigma$ est la relation de transition $\delta/\Sigma' \subset S \times \Sigma' \cup \{\varepsilon\} \times S$ tel que :

- pour tout $e \in \Sigma'$, pour tout $(s, s') \in S \times S$, si $(s, e, s') \in \delta$ alors $(s, e, s') \in \delta/\Sigma'$,
- pour tout $e \in \Sigma - \Sigma'$, pour tout $(s, s') \in S \times S$, si $(s, e, s') \in \delta$ alors $(s, \varepsilon, s') \in \delta/\Sigma'$.

Intuitivement, les restrictions d'une relation de transition garde inchangée la structure d'une relation de transition mais distingue les noms d'un sous-ensemble d'évènements d'intérêt et masque les autres noms avec des évènements non observables ε.

Définition 4.18 (Arbre de calcul)**.** Un arbre de calcul extrait à partir d'un IOA est un arbre constitué de noeuds connectés par des arcs orientés tel que :

- chaque noeud $n_j(s_i)$ est étiqueté par un état s_i de l'automate et la racine de l'arbre est étiqueté par un état initial s_0.
- chaque arc $(n_j(s_i), l_i, n_{j+1}(s_{i+1}))$ qui connecte deux noeuds est étiqueté par un nom d'évènement l_i de l'automate.
- un arc $(n_j(s_i), l_i, n_{j+1}(s_{i+1}))$ est tel que $(s_i\ l_i\ s_{i+1})$ est une transition d'automate.

Définition 4.19 ("Run" (ou trace))**.** Une trace d'un IOA est une séquence (finie ou non) $\sigma(s_0)l_0 \ldots \sigma(s_i)l_i \ldots$ où s_i est un état (s_0 est un état initial) et l_i est un nom d'évènement tel que tout $(s_i\ l_i\ s_{i+1})$ d'une trace est une transition de l'automate. Une trace peut aussi être définie par un chemin extrait de l'arbre de calcul d'un automate.

Définition 4.20 (Restriction d'une trace)**.** Une trace restreinte à un vocabulaire est la trace initiale où la fonction de configuration initiale et la relation de fonction initiale sont remplacées par leurs restrictions respectives.

Définition d'une contrainte d'interfaçage temporelle Une contrainte d'interfaçage temporelle est une propriété d'exécution définie à l'égard d'un vocabulaire donné (noms de variables d'entrée/sortie et d'évènements).

Les propriétés d'invariants sont des contraintes sur la fonction de configuration. Par exemple, considérons que *altitude* soit une variable d'entrée du vocabulaire de

l'automate. La contrainte "la variable *altitude* est positive ou nulle" est un exemple d'invariant. L'automate qui satisfait cet invariant inclu une variable d'entrée entière appelée *altitude* et une fonction de configuration σ telle que, pour tout état s, $\sigma(s)(altitude) \geq 0$.

Les propriétés temporelles sont des contraintes qui lient les configurations et les transitions. Deux sortes de propriétés temporelles doivent être distinguées :

- linéaire versus arbre de calcul : s'adresse aux contraintes sur toutes les exécutions de l'automate versus celles sur l'arbre de calcul extrait de l'automate,
- passé versus futur.

Considérons, par exemple, la variable de sortie booléenne *alarm* du vocabulaire d'un automate. "Si la valeur de *alarm* est vraie dans un état de l'exécution, alors il y a un état précédent dans l'exécution où $altitude \leq 10$" est une propriété à temps linéaire et passé. "Si $altitude \leq 10$ dans un noeud de l'arbre de calcul d'un automate, alors nous devrions pouvoir trouver un chemin et un noeud suivant dans le chemin qui satisfait $alarm = true$" est un exemple de propriété sur l'arbre de calcul et futur.

Les propriétés temporelles avec un vocabulaire qui s'adresse à des variables d'horloge, sont des propriétés temporisées. "Si $altitude \leq 10$ était vrai pendant au moins dix unités de temps alors $alarm = true$" est un exemple de propriété temporisée.

Définition 4.21 (Ensemble d'exécution satisfait par un TIC)**.** Soit un automate A et TIC une contrainte définie sur le vocabulaire incluse dans le vocabulaire de A. Nous notons $\|TIC\|_A$ l'ensemble des exécutions de A qui satisfait le TIC.

Dans le contexte de la validité d'un produit de M&S, les SOU sont les propriétés des exécutions de simulation qui sont requises pour réaliser correctement une expérimentation particulière (la validation d'un système d'intérêt). Le SOU peut être vu comme un plan de test. Nous proprosons de formaliser un SOU par un TIC sur le vocabulaire $I_{SOU} \cup O_{SOU} \cup \Sigma_{SOU}$. Donc les entrées du SOU sont les résultats de simulation que l'utilisateurs veut observer alors que les sorties du SOU sont les stimuli qui sont injecté dans la simulation. Similairement, SDU sont les propriétés assurées par l'exécution de la simulation. SDU peut être considéré comme une spécification détaillée d'une simulation. Nous proposons de formaliser un SDU par un TIC sur un vocabulaire $I_{SDU} \cup O_{SDU} \cup \Sigma_{SDU}$. Les entrées du SDU doivent être alimentées

par les sorties du plan de test alors que les sorties du SDU sont tous les résultats de simulation possibles.

SOU et SDU peuvent être connectés s'ils ont des vocabulaires compatibles :

- $I_{SOU} \subset O_{SDU}$: tous les résultats d'intérêts du plan de test sont fournis par la simulation et la simulation peut fournir plus de résultats que nécessaire.
- $O_{SOU} = I_{SDU}$: toutes les stimulations planifiées par le plan de test peuvent être faites et toutes les entrées nécessaires pour réaliser la simulation sont définies par le plan de test (remarque : le cas $O_{SOU} \subset I_{SDU}$ est résolu par la règle d'applicabilité *relation d'E/S*).
- $\Sigma_{SOU} \subset \Sigma_{SDU}$: tous les évènements d'intérêt du plan de test peuvent être observés durant la simulation mais la simulation permet d'observer plus d'évènements que nécessaire.

La compatibilité des vocabulaires est la première condition nécessaire pour une utilisation de la simulation à l'égard d'un plan d'expérimentation. Clarifions maintenant comment vérifier la compatibilité d'un SOU et d'un SDU par comparaison des traces remplies par les TICs correspondants.

Soit SOU et SDU deux TICs sur les vocabulaires respectifs $V_{SOU} = I_{SOU} \cup O_{SOU} \cup \Sigma_{SOU}$ et $V_{SDU} = I_{SDU} \cup O_{SDU} \cup \Sigma_{SDU}$ tel que ces vocabulaires sont compatibles. SOU et SDU sont complètement compatibles selon leur intéraction dynamique si et seulement si toutes les traces du SDU restreintes au vocabulaire du SOU sont des traces du SOU : $\|SDU\|_{VSDU/VSOU} = \|SOU\|_{VSOU}$.

Intuitivement cela signifie que :

- $\|SDU\|_{VSDU/VSOU} \subset \|SOU\|_{VSOU}$: le comportement du SDU est dans l'enveloppe des comportements significatifs à l'égard du SOU.
- $\|SOU\|_{VSOU} \subset \|SDU\|_{VSDU/VSOU}$: toutes les expérimentations plannifiées par le SOU peuvent être jouées sur le SDU.

Lorsque la première condition est fausse, il y a des exécutions du SDU qui ne sont pas des exécutions du SOU. Nous distinguons deux cas :

1. Il y a un risque sur la couverture de test. Ceci peut être le cas si le SDU considère des variables d'entrées avec d'autres valeurs que celles planifiées par le SOU. Donc le SOU n'explore pas toutes les exécutions de la simulation. Cela signifie que, soit les exécutions non explorées ne sont pas pertinentes pour le plan de test, soit que le SOU n'est pas assez complet.

2. Il y a un biais dans la simulation ou dans la définition de référence. Cela peut être le cas car le SDU considère des variables de sorties avec d'autres valeurs que celles attendues par le SOU. Cela siginifie que, soit les résultats de simulation sont incorrect, soit que des hypothèses du SOU sont fausses.

Lorsque la deuxième condition est fausse, il y a des exécutions envisagées par le SOU qui ne sont pas des exécutions du SDU. La encore, nous distinguons deux cas :

3. Certains tests sont en dehors du domaine d'usage du modèle de simulation. Ceci peut être le cas si le SOU considère des variables de sorties avec plus de valeurs que celles acceptées par le SDU. Donc, le SOU projète d'explorer des exécutions de simulation en dehors du périmètre recommendé par le SDU et les résultats de simulation ne peuvent plus être garantis. Le SDU ou le SOU doit être modifié.
4. Il y a un risque sur la complétude du modèle. Cela peut être le cas si le SOU considère les variables d'entrées avec plus de valeurs que celles fournit par les résultats de simulation. Cela signifie que, soit la simulation permet d'apprendre quelque chose si des incertitudes du SOU sont réduites, soit que la simulation oublie l'implémentation de certains cas.

La figure 4.14 illustre ces quatres cas.

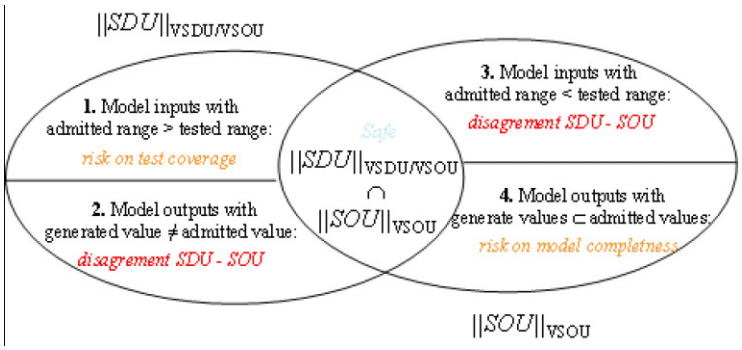

FIGURE 4.14 – Niveau de compatibilité comportemental SOU/SDU

Chapitre 5

Description et illustration d'une méthodologie d'évaluation

Sommaire

5.1 Proposition d'une méthodologie d'évaluation

5.1.1 La démarche d'ingénierie système

Les processus de conception d'un système (bloc "System Design" de la figure 1.1 présenté chapitre 1 (1.1.1.1), sont utilisés afin de transformer les exigences du client (l'acquéreur) en un ensemble de produits réalisables qui satisfont ce client. Cela consiste en deux processus [EIA 99] illustrés sur la figure 5.1 : le processus de définition des exigences et le processus de définition d'une solution.

Le processus de définition des exigences transforme les exigences du client en *exigences techniques du système*. Ces exigences techniques sont ensuite transformées

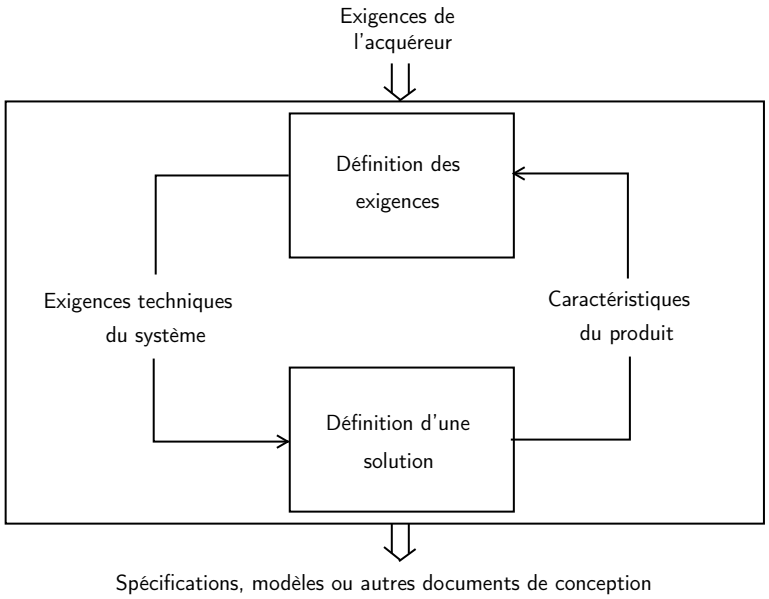

FIGURE 5.1 – Processus de conception d'un système

en un ensemble d'*exigences spécifiées*. Ces exigences prennent la forme de spécifications, de modèles ou autres documents de conception. L'EIA-632 présente les différents processus d'ingénierie système sous forme d'exigences. Ces deux processus sont respectivements raffinées en trois exigences (exigences 14 à 19 [EIA 99]) qui mettent en exergue différents "produits" de représentation du système à différents niveaux 5.2.

Les exigences de l'acquéreur Ces exigences regroupent les exigences du client, de l'utilisateur ou de l'opérateur du système ou d'une portion du système. Elles incluent des exigences de développement, de production, de test, de déploiement, d'entraînement, d'opérations, de maintenance et de la disposition des produits du système.

Les exigences techniques du système Les exigences techniques du système découlent des exigences de l'acquéreur. L'objectif ici est de classer les exigences en exigences opérationnelles (profils d'utilisation) et de performance. Les exigences opérationnelles incluent :

– les évènements auquels le produit doit répondre.

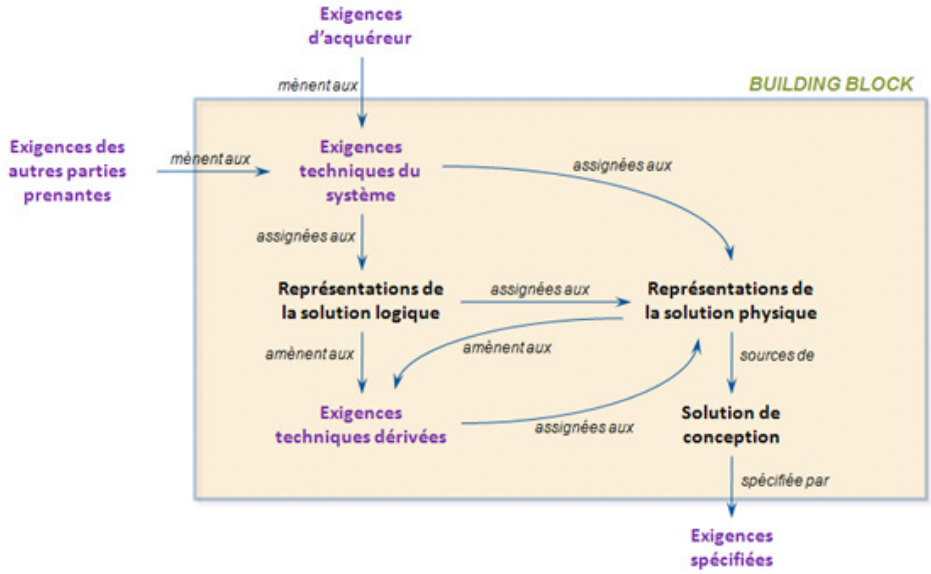

FIGURE 5.2 – "Produits" de conception du système

- ce que le produit doit être capable d'accomplir en réponses à ces évènements pour satisfaire les éxigences identifiées par l'acquéreur.
- les interfaces fonctionnelles et physiques (e.g. mécaniques, électrique, thermique, information), incluant les interactions physiques (e.g. forme, masse) et les flots d'informations.

Les exigences de performances définissent comment (avec quelles "qualités" non-fonctionnelles) les fonctions du système doivent être accomplies. Celles-ci incluent des contraintes d'entreprise, de développement, de production, suppport et maintenance...

Les représentations de la solution logique La solution logique consiste à fournir une définition abstraite du système afin de décrire comment il se comporte. Cette définition peut prendre différentes formes suivant le type d'approche utilisée, e.g. analyse fonctionnelle, analyse orientée-objet, analyse structurée... L'élaboration d'une solution logique inclut l'identification et la définition :

1. des interfaces du système liées à l'organisation et au séquencement logique de celui-ci,
2. des réponses (sorties) d'une sous-fonction, d'un groupe de sous-fonctions, d'ob-

jets, etc..., et des stimuli (entrées) pour chaque profils opérationnels identifiés,

3. des modes et des états du système,
4. de la chronologie associée à la séquence d'exécution des fonctions pour chaque profil opérationnel,
5. des flots de données entre les fonctions, et leurs contrôles d'exécution.

Les représentations de la solution physique La solution physique alloue les éléments de la solution logique à des éléments physiques du système (e.g., capteur, moteur, calculateur, élément de stockage, source d'alimentation...). Cette allocation est précédée par une analyse de la solution logique pour déterminer quels éléments seront implémentés par du logiciel et d'autres par du matériel.

5.1.2 La démarche de développement d'un modèle conceptuel

Rappelons que les propriétés, les méthodes et l'approche que nous proposons doivent s'intégrer dans une démarche d'ingénierie système. La recherche depuis quelques années et les industriels depuis peu, s'intéressent à SysML comme outil de support à l'ingénierie système. SysML tend en fait à couvrir l'ensemble du processus de conception que nous avons présenté ci-dessus. Il utilise pour cela un certain nombre de diagrammes et d'artefacts permettant la définition des exigences techniques, de la solution logique et physique du système sous différentes vues :

- Exigences et leurs relations à d'autres exigences, éléments de conception et cas de test.
- Composition structurelle, interconnexion et classification.
- Comportement basé fonctions, messages et états.
- Contraintes sur les propriétés physiques et de performance.
- Allocations entre le comportement, la structure et les contraintes

Un modèle conceptuel de simulation est décomposé en trois grands ensembles que nous définissons par des `package` SysML (figure 5.3) :

- le `package SOU` concerne la vue utilisateur et décrit les objectifs d'utilisation et le cadre expérimental de l'expérimentation,
- le `package SDU` concerne la vue développeur et décrit le domaine d'usage du modèle de simulation,

– le `package Applicability` qui concerne les règles de compatibilité entre le `SOU` et le `SDU`.

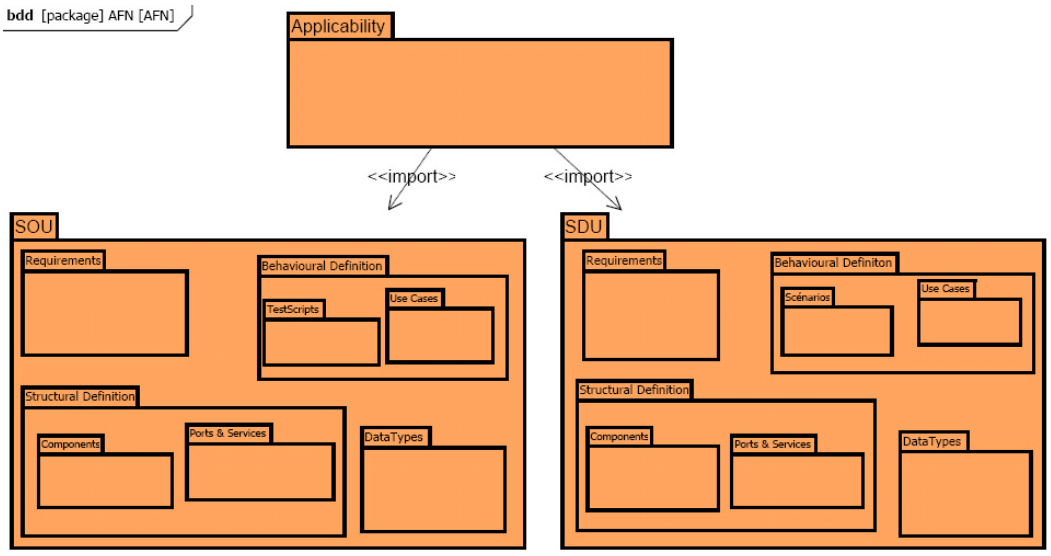

FIGURE 5.3 – Organisation du modèle conceptuel d'un produit de simulation

Les `package SOU` et `SDU` sont constitués de divers `package` :

– un `package Requirements` (normalement décomposés en 4 sous-ensembles : `Functional Requirements`, `Performance Requirements`, `Interface Requirements` et `Environment Requirements`).
– un `package Behavioural Definition` contient des cas d'utilisation et les scénarios du système.
– un `package Structural Definition` décrit le système en termes de composants qui communiquent à travers des ports et utilisent ou réalisent des services.
– un `package DataType` définit une bibliothèque de types de données utilisées par le système. Ces types incluent les types standards informatique (e.g. bool, réel...) et les unités du système international (e.g. mètre, litre...). Ils incluent également des types spécifiques au domaine spécialement conçus pour le besoin de l'étude.

Package Behavioural Definition La définition comportementale décrit la simulation sous forme de cas d'utilisation. Les cas d'utilisation décrivent le comportement

d'un système du point de vue de l'utilisateur. C'est, en quelque sorte, l'image d'une fonctionalité du système, déclenchée en réponse à la stimulation d'un acteur externe. Ils permettent alors de définir les limites du système et les relations entre le système et l'environnement.

Les diagrammes de séquence permettent de représenter les interactions entre des éléments structurels comme une séquence d'échange de messages. Ces interactions peuvent être entre le système et son environnement (e.g. les acteurs) ou entre des composants du système, et ce à n'importe quel niveau de décomposition hierarchique. Un message peut représenter l'invocation d'un service ou l'envoie d'un signal sur le système ou un composant du système. En général, un ou plusieurs diagrammes de séquence sont associés à un cas d'utilisation.

Package Structural Definition Le package `Structural Definition` est constitué de deux sous-packages : `components` et `ports & services`.

Nous utilisons les diagrammes de blocs pour définir les caractéristiques d'un composant de la simulation. Les blocs SysML fournissent un moyen générique pour décrire l'architecture d'un système. Ce sont des unités modulaires incluant les caractéristiques structurelles et comportementales des éléments du système, telles que ses propriétés et ses opérations. À un plus bas niveau de décomposition, un bloc peut être vu comme un ensemble de sous-systèmes, appelés `parts`, qui peuvent à leur tour, être décrits par des blocs. Un port est une propriété d'un bloc qui spécifie un point d'interaction distinct entre ce bloc et son environnement ou ses composants internes. Un port peut optionnellement spécifier les services (`interface`) qu'un bloc fournit ou requiert à ou de son environnement. Les ports sont connectés aux propriétés du bloc par des connecteurs à travers lesquels des requètes peuvent être faites pour invoquer les caractéristiques comportementales d'un bloc.

Le package `ports & services` définit les classes qui implémentent ou utilisent les services qu'un bloc fournit ou utilise. Ces classes seront utilisées pour typer les ports des blocs.

Enfin, nous définissons le comportement associé à un bloc à l'aide des IOAs . Le comportement associé à un bloc définit comment le bloc répond à des évènements d'entrée et fournit des sorties. Les formalismes SysML permettant cette déscription sont les diagrammes d'activités et les machines à états. SysML n'inclut pas les

IOAs, nous utilisons Uppaal [Larsen 97], qui nous permet de définir des contraintes d'interfaçage temporelles sur les comportements des blocs, ce que ne permettent pas les diagrammes d'activités et les machines à états SysML.

5.2 Cas d'application

Nous illustrons maintenant, à partir d'un cas d'application, la méthodologie et les concepts que nous avons proposés. En partant des exigences d'un système et du plan de V&V destiné à valider ces exigences, nous allons construire un modèle de l'expérimentation, i.e. le SOU. En partant de la spécification d'un modèle de ce système nous allons construire le SDU. Puis nous évaluons leur compatibilité.

5.2.1 Présentation du système

Le cas d'étude que nous proposons est basé sur l'application AFN (Aircraft Facilities Notification), composant du Air Traffic System (ATC). L'application AFN est le pré-requis pour toutes communications entre le cockpit et les stations sol. Son rôle est de signaler aux centres de contrôle de trafic aérien les adresses et les informations de l'avion concernant les moyens de communications (e.g. numéro de vol, numéro d'identification, code ICAO...).

L'application AFN est constitué de deux procédures (figure 5.4). La procédure de "notification initiale" qui consiste à établir un contact avec une station sol et la procédure de "requète pour notification" qui consiste à transférer le contact établi d'une station sol à une autre. Un plan de vol est élaboré selon un passage sur des stations sol successives permettant une couverture continue des communications. Lorsque cette couverture ne peut plus être assurée par une station sol, le contact est établi avec la prochaine.

Description de la procédure Un message de recommandation de prise de contact FN_CAD est envoyé par une station sol LFDG afin que le système se notifie à une nouvelle station sol LFCQ. Lorsqu'un FN_CAD est reçu, une procédure de "requète pour notification" est engagée et un message de réponse FN_RESP est envoyé à la station sol LFDG amorçant ainsi la procédure. Puis, un message de prise de contact FN_CON est envoyé à la nouvelle station sol. Le système déclenche une temporisa-

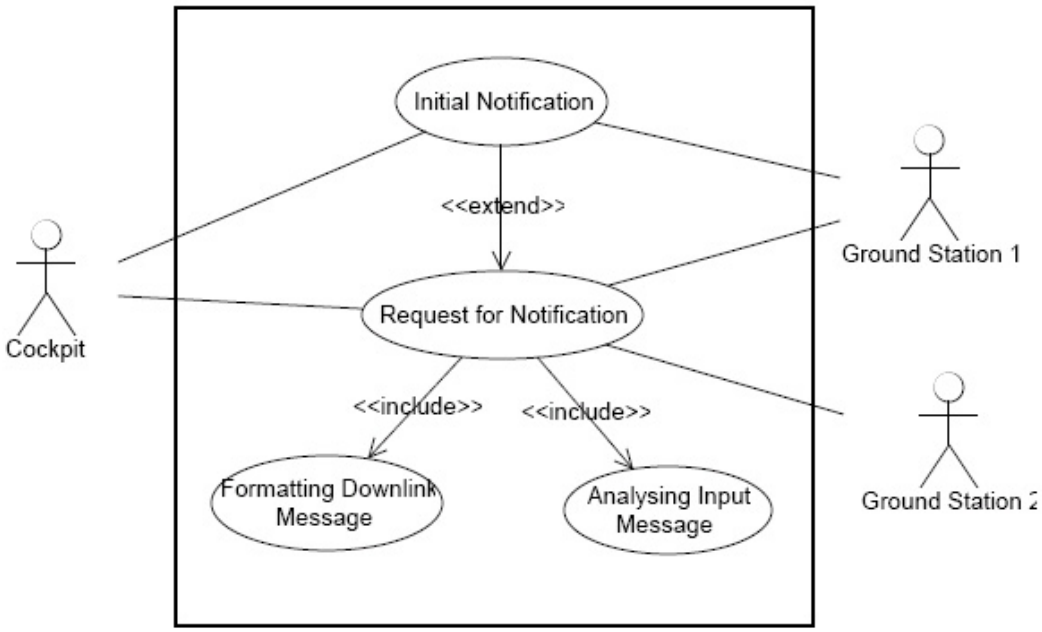

FIGURE 5.4 – Diagramme de cas d'utilisation de l'application AFN

tion ATST de dix minutes dans l'attente d'un accusé de réception FN_AK. Lorsque l'accusé de réception est reçu par le système, le timer est remis à zéro et un message de complétion FN_COMP est envoyé par le système à la première station sol afin de l'informer du résultat de la transaction. La figure 5.5 ci-dessous illustre cette procédure.

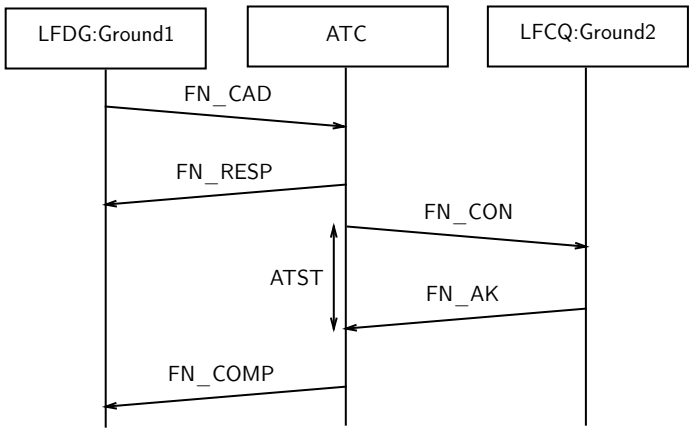

FIGURE 5.5 – Procédure de requète pour notification

5.2.2 Spécification du SOU

Les objectifs d'utilisation de la simulation concernent la procédure de requète pour une notification de l'avion à une station sol.

5.2.2.1 Exigences d'intérêt

Un SOU est construit pour vérifier et valider des exigences du système ATC. Nous considérons les exigences suivantes liée à la fonction AFN :

AFN01 Si l'adresse de la station sol reçue dans un message provenant de celle-ci n'est pas une chaine de 7 caractères, où chaque caractère appartient aux sous-ensembles (A à Z) and (0 à 9), le système doit ignorer la totalité du message.

AFN02 Si le numero de vol reçu dans un message provenant de la station sol n'est pas une chaine de 8 caractères, où chaque caractère appartient aux sous-ensembles (A à Z) and (0 à 9), le système doit ignorer la totalité du message.

AFN03 Lors de la réception d'un message de recommandation de prise de contact valide, le système doit transmettre un message de réponse, puis il doit envoyer un message de prise de contact à la prochaine station sol.

AFN04 L'adresse de la station sol contenue dans le message de réponse doit être la même que celle contenue dans le message de recommandation de prise de contact correspondant.

AFN05 Le numéro de vol contenu dans un message envoyé par le système doit être une chaine de 8 caractères, où chaque caractère appartient aux sous-ensembles (A,Z) et (0,9).

AFN06 Le système doit remplir le champ "position" d'un message avec la position courante de l'avion, exprimée en latitude et longitude avec une résolution d'un dixième de minute sous le format suivant : YDDMMTZDDDMMT, où la direction Y appartient à (N ;S), la direction Z appartient à (E ;W), les degrés DD et DDD appartienent respectivement à l'ensemble (00,90) et (000,180), les minutes MM appartiennent à l'ensemble (00,59), les dixièmes de minutes T appartiennent à l'ensemble (0,9).

AFN07 Lors de la réception d'un accusé de réception provenant de la prochaine station sol, le système doit transmettre un message de complétion à la première station sol.

5.2.2.2 Définition comportementale

Scénarios Nous pouvons, d'ores et déjà, élaborer un (ou un ensemble) de plans de test afin de valider les exigences. Nous utilisons les diagrammes de séquence pour décrire ces plans de test. Ces diagrammes représentent les plans de V&V élaborés sous forme textuelle par le concepteur du système ATC. Le plan de test correspondant à la validation des exigences ci-dessus est donné par le diagramme de la figure 5.6.

Le plan de test est constitué de trois objets : un *système d'intérêt*, soumis aux stimuli du *générateur* et aux observations du *transducteur*. Bien que nous n'ayons ici encore qu'une vision boite noire du système d'intérêt, nous pouvons d'ores et déjà identifier quelques éléments du périmètre requis de celui-ci. Le premier message `LFDG.ContactAdvisory` signifie qu'il faut stimuler l'envoi d'un message de recommandation de prise de contact à partir de la station sol LFDG. Cette information est d'une grande importance puisqu'elle signifie que le système d'intérêt est constitué de la fonction AFN et d'(au moins) une station sol. Ensuite, le plan de test consiste à observer la réception d'un message de réponse sur la station sol LFDG, de récupérer l'adresse de la station sol et le numéro de vol du message de réponse afin de vérifier leur format. Notons que, par la suite, une deuxième station sol LFCQ est nécessaire à la réalisation de cette expérimentation.

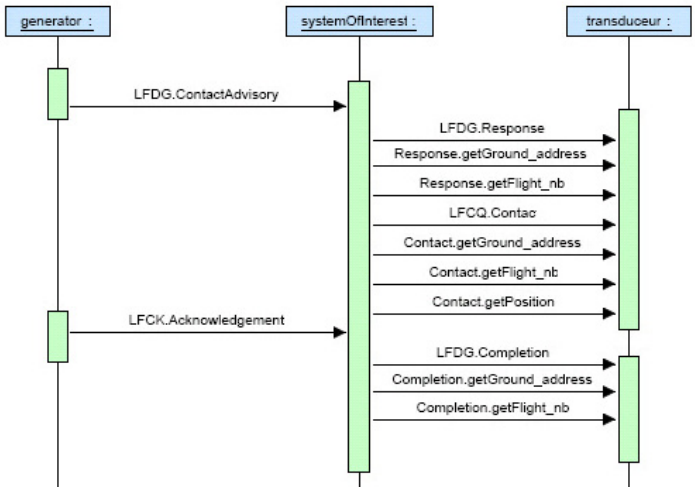

FIGURE 5.6 – Diagramme de séquence du plan de test "Requète pour Notification"

5.2.2.3 Définition structurelle

À partir des plans de test ci-dessus, nous pouvons élaborer la définition structurelle du SOU. L'architecture logique du SOU est constituée de trois composants : un générateur, un transducer et un accepteur.

Ports & Services La première étape de la définition structurelle consiste à définir les classes qui seront utilisées pour typer les ports des composants. Ces classes sont représentées par des blocs SysML. Le diagramme de bloc figure 5.12 présente ces classes. Dans notre cas d'étude nous avons identifié deux services génériques requis par le SOU : un service permettant la génération de message AFN et un service permettant l'observation de message AFN respectivement définis par les `interfaces AfnMessageGeneration` et `AfnMessageObservation`. Il y a deux types `AFNMessage`, les messages provenant du sol vers le système ATC et ceux provenant du système ATC vers le sol, respectivement définis par les blocs `UpLink` et `DownLink`. Un message AFN doit avoir comme attributs un numéro de vol `flight_nb` et une adresse de station sol `ground_address`. Un message `DownLink` doit également contenir un attribut `position` contenant la position de l'avion au moment de l'envoie du message. Les types des attributs sont spécifiés dans le `package DataType` tel que requis par le SOU, e.g. `GroundAddressType` est une chaine de 7 caractères, où chaque caractère appartient aux sous-ensembles (A to Z) and (0 to 9). Les opérations `getGroundAddress()` et `setGroundAddress(aGroundAddress:GroundAddressType)` signifient respectivement qu'il doit être possible de récupérer ou d'assigner la/une l'adresse de la station sol d'un message AFN. Il est important de souligner que seules les informations relatives à la satisfaction de l'objectif d'utilisation doivent être spécifiées, indiquant ainsi les abstractions acceptables. On remarque par exemple que l'objectif d'utilisation ne nécessite pas l'assignement de la position de l'avion dans un message. Cette opération est faite par le système ATC.

Components La définition structurelle du SOU est donnée par le diagramme de bloc SysML ci-dessous (figure 5.8). Le SOU est représenté par un bloc qui possède trois `parts properties` : `generator`, `acceptor` et `transduceur`. Les ports typés fn_resp, fn_comp, fn_con, fn_cad et fn_ak sont affectés au bloc.

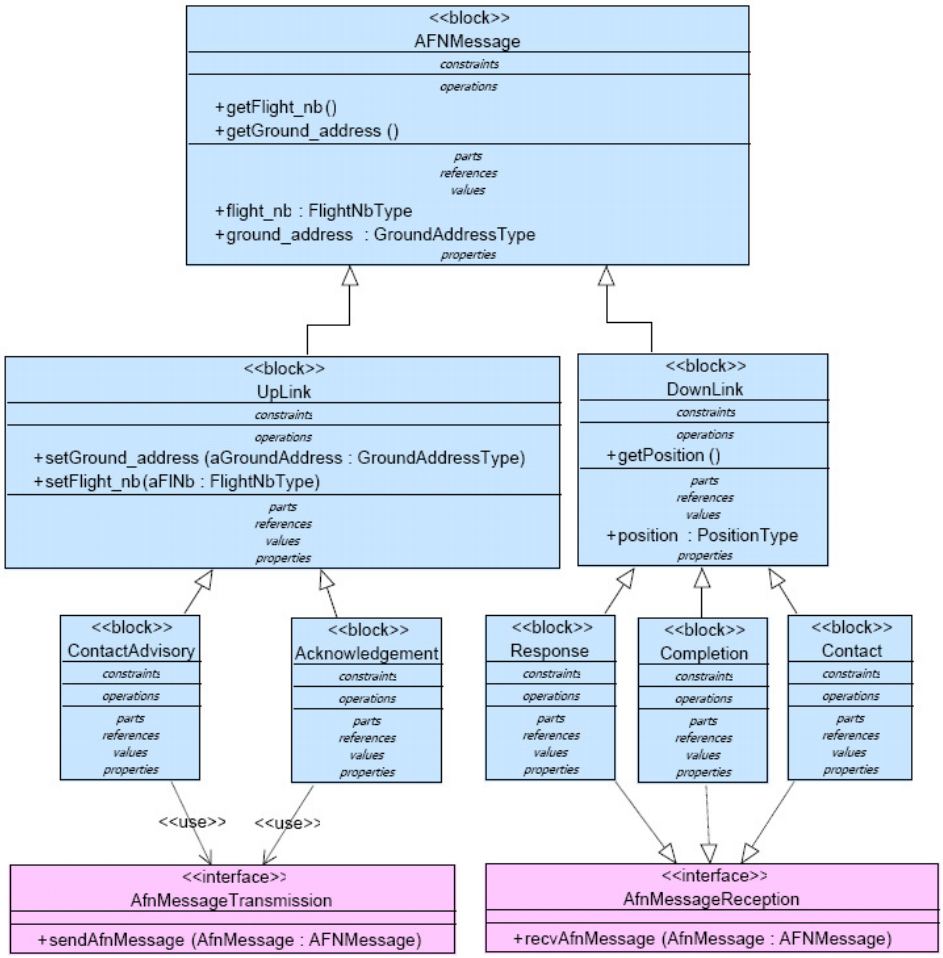

FIGURE 5.7 – Définition des ports et des services du SOU

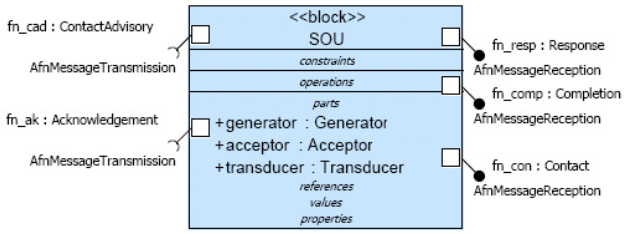

FIGURE 5.8 – Définition structurelle du SOU

Structure interne du composant SOU La structure interne du composant SOU est décrite à l'aide d'un diagramme de bloc interne SysML (figure 5.9. Nous retrouvons les trois composants Generator, Acceptor et Transducer représenté par

des `parts` connecté à travers leurs ports respectifs.

Rappelons que l'accepteur selectionne les données d'intérêt en surveillant si les conditions expérimentales souhaitées sont respéctées (section 2.2.1). À partir de relations établies entre les segments générées par le générateur et de ceux observées par le transducteur, l'accepteur permet de vérifier si les exigences du système d'intérêt sont valides. Par exemple, l'exigence AFN07 met en relation une valeur de la variable `fn_ak` avec une valeur de la variable `fn_comp`, e.g. si `fn_ak` = true alors un jour, `fn_comp` = true. Nous décrivons une manière de spécifier ce type de conditions par la suite.

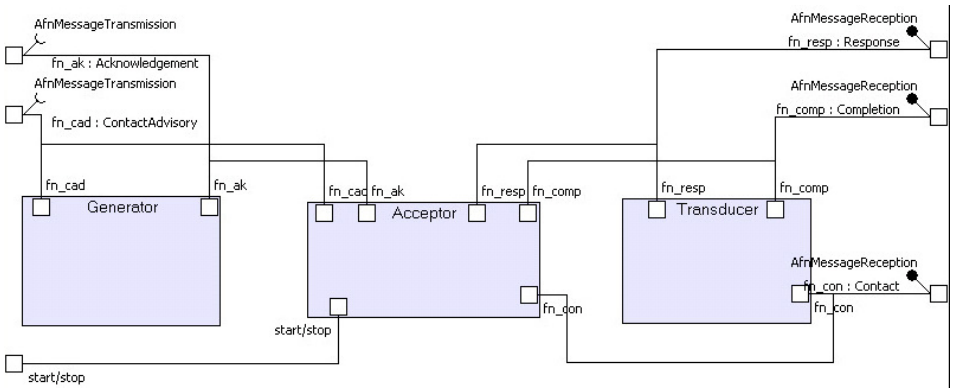

FIGURE 5.9 – Structure interne du SOU

5.2.3 Spécification du SDU

La même démarche est adoptée pour définir un SDU. Plaçons nous dans le cas où un modèle de simulation a été développé. Nous voulons savoir si ce modèle de simulation peut être utilisé pour satisfaire l'objectif d'utilisation décrit dans la section précédente.

5.2.3.1 Définition structurelle

Components L'architecture logique du SOU est constituée de trois composants (figure 5.10 : l'AFN et deux stations sol LDFG et LFCQ. Ce modèle de simulation offre deux ports d'entrée `fn_cad` et `fn_ak` respectivement typés par les classes `ContactAdvisory` et `Acknowledgement` et trois ports de sorties `fn_con`, `fn_resp` et `fn_comp` respectivement typés par les classes `Response`, `Contact` et `Completion`.

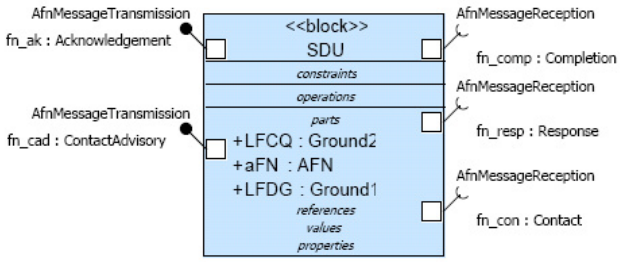

FIGURE 5.10 – Définition structurelle du SDU

Ports & Services Regardons de plus près le diagramme de bloc décrivant les classes qui sont utilisées pour typer les ports du composant SDU (figure 5.11). Nous remarquons que le seul attribut d'un message AFN est sa validité `validity` de type booléen. `validity = true` signifie que le message est au bon format. Ce modèle de simulation ne permet pas donc pas spécifier une adresse de station sol ou un numero de vol. Il permet simplement de dire si le message et valide ou non sans même donner d'information de quel champ (adresse sol ou numéro de vol) rend ce message valide ou non. Il y eu abstraction par interprétation abstraite.

Structure interne du composant SDU La structure interne du composant SDU est décrite à l'aide d'un diagramme de bloc interne SysMl (figure 5.9. Nous retrouvons les trois composants AFN, LFDG et LFCQ représentés par des `parts` connectés à travers leurs ports respectifs.

5.2.4 Validation de la structure et des données

5.2.4.1 Spécification formelle du SOU

Le SOU est défini par la structure :

$$SOU =< X_{SOU}, Y_{SOU}, D_{SOU}, Z_{SOU \to SDU}, Z_{SDU \to SOU} >$$

avec

- $X_{SOU} = \{fn\_resp, fn\_con, fn\_comp\}$
- $Y_{SOU} = \{fn\_cad, fn\_ak\}$
- $D_{SOU} = \{generator, acceptor, transducer\}$
- $Z_{SOU \to SDU}(fn\_ak) = fn\_ak_{LFCQ}$

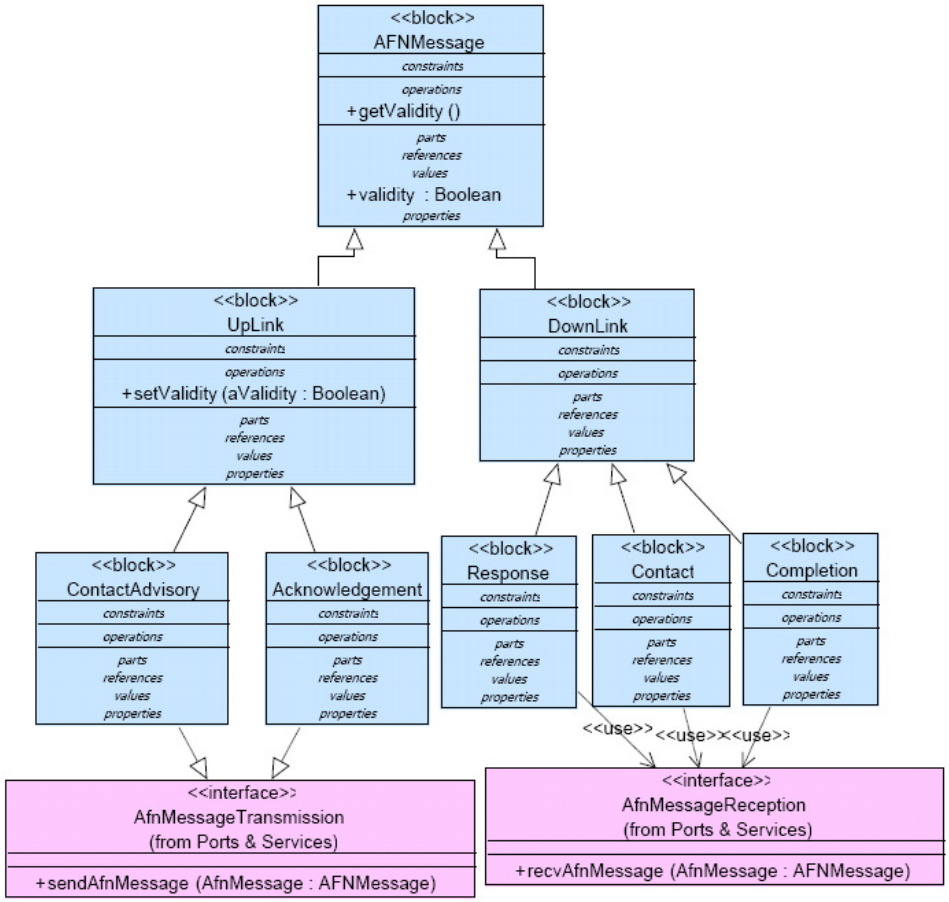

FIGURE 5.11 – Définition des ports et des services du SDU

- $Z_{SOU \to SDU}(fn\_cad) = fn\_cad_{LFDG}$
- $Z_{SDU \to SOU}(fn\_resp_{LFDG}) = fn\_resp$
- $Z_{SDU \to SOU}(fn\_comp_{LFDG}) = fn\_comp$
- $Z_{SDU \to SOU}(fn\_con_{LFCQ}) = fn\_con$

tel que
$$\begin{cases} information_{SOU}(fn\_cad) = ContactAdvisory \\ information_{SOU}(fn\_ak) = Acknowledgement \\ information_{SOU}(fn\_resp) = Response \\ information_{SOU}(fn\_con) = Contact \\ information_{SOU}(fn\_comp) = Completion \end{cases}$$

et tel que
$$\begin{cases} datatype_{SOU}(fn\_cad) = datatype_{SOU}(fn\_ak) = char[15] \\ datatype_{SOU}(fn\_resp) = datatype_{SOU}(fn\_con) = char[28] \\ datatype_{SOU}(fn\_comp) = char[28] \end{cases}$$

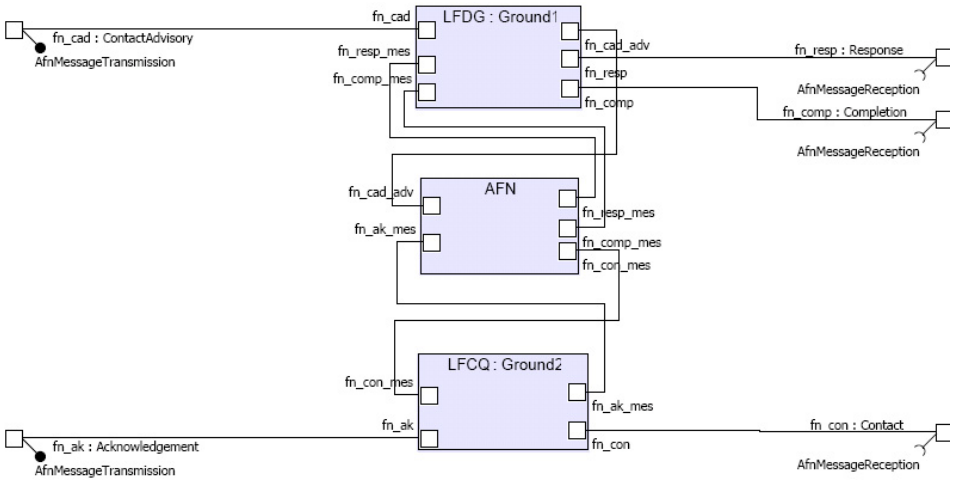

FIGURE 5.12 – Structure interne du SDU

Remark 5.1. `char[15]` correspond à un message constitué des 7 caractères de l'adresse de la station sol et des 8 caractères du numéro de vol. `char[28]` correspond à un message constitué de l'adresse de la station sol, du numéro de vol et des 13 caractères correspondants à la position de l'avion.

5.2.4.2 Spécification formelle du SDU

Le SDU est défini par la structure :

$$SDU =< T, X_{SDU}, Y_{SDU}, D_{SDU}, \{M_d\}, \{I_d\}, \{Z_{SDUi\to d}\} >$$

où

- $X_{SDU} = \{fn\_cad, fn\_ak\}$
- $Y_{SDU} = \{fn\_resp, fn\_con, fn\_comp\}$
- $D_{SDU} = \{LFDG, AFN, LFCQ\}$
- $Z_{SDU\,SDU\to LFDG}(fn\_cad) = fn\_cad_{LFDG}$
- $Z_{SDU\,SDU\to LFCQ}(fn\_ak) = fn\_ak_{LFCQ}$
- $Z_{SDU\,LFDG\to SDU}(fn\_resp_{LFDG}) = fn\_resp$
- $Z_{SDU\,LFDG\to SDU}(fn\_comp_{LFDG}) = fn\_comp$
- $Z_{SDU\,LFCQ\to SDU}(fn\_con_{LFCQ}) = fn\_con$

tel que

$$\begin{cases} information_{SDU}(fn\_cad) = ContactAdvisory \\ information_{SDU}(fn\_ak) = Acknowledgement \\ information_{SDU}(fn\_resp) = Response \\ information_{SDU}(fn\_con) = Contact \\ information_{SDU}(fn\_comp) = Completion \end{cases}$$

$$\text{et tel que } \begin{cases} datatype_{SDU}(fn\_cad) = datatype_{SDU}(fn\_ak) = bool \\ datatype_{SDU}(fn\_resp) = datatype_{SDU}(fn\_con) = bool \\ datatype_{SDU}(fn\_comp) = bool \end{cases}$$

5.2.4.3 Evaluation de la compatibilité

D'après les spécifications du SDU et du SOU nous pouvons dire que $scope_{SOU} = scope_{SDU}$. Rappelons qu'un SOU et un SDU sont compatibles selon leur toplogie respectives si et seulement si :

$$\text{pour tout } z \in Z_{SOU \to SDU} : Y_{SOU} \to X_c \text{ tel que } c \neq SDU$$

$$\exists Z_{SDU\, i \to d} : X_i \to X_d \text{ tel que } i = SDU \text{ et } c = d$$

$$\text{pour tout } z \in Z_{SDU \to SOU} : Y_c \to X_{SOU} \text{ tel que } c \neq SDU$$

$$\exists Z_{SDU\, i \to d} : Y_i \to Y_d \text{ tel que } d = SDU \text{ et } c = i$$

Donc SOU est applicable à SDU selon leurs topologies respectives puisque :

$$\begin{cases} Z_{SOU \to SDU}(fn\_ak) = fn\_ak_{LFCQ} \wedge Z_{SDU\, SDU \to LFCQ}(fn\_ak) = fn\_ak_{LFCQ} \\ Z_{SOU \to SDU}(fn\_cad) = fn\_cad_{LFDG} \wedge Z_{SDU\, SDU \to LFDG}(fn\_cad) = fn\_cad_{LFDG} \\ Z_{SDU \to SOU}(fn\_resp_{LFDG}) = fn\_resp \wedge Z_{SDU\, LFDG \to SDU}(fn\_resp_{LFDG}) = fn\_resp \\ Z_{SDU \to SOU}(fn\_comp_{LFDG}) = fn\_comp \wedge Z_{SDU\, LFDG \to SDU}(fn\_comp_{LFDG}) = fn\_comp \\ Z_{SDU \to SOU}(fn\_con_{LFCQ}) = fn\_con \wedge Z_{SDU\, LFCQ \to SDU}(fn\_con_{LFCQ}) = fn\_con \end{cases}$$

Pour évaluer l'applicabilité du SOU au SDU selon leurs types de données respectifs, nous pouvons construire par exemple la lattice ci dessous (figure 5.13 et formaliser la relation entre les types des variables. Au plus haut niveau d'abstraction (en bas de la lattice) nous n'avons aucune information sur le message ; il peut être valide ou erroné. Un message peut être erroné soit parce que le champ adresse de la station sol est erroné, soit parce que le champ numéro de vol est erroné ou soit parce que les deux sont erronés. Au plus bas niveau d'abstraction (en haut de la lattice)

le message est une chaine de 15 caractères (7 caractères pour l'adresse de la station sol et 8 caractères pour le numéro de vol).

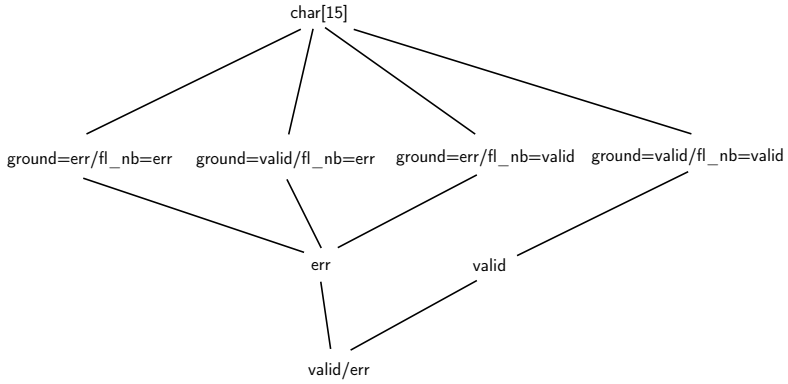

FIGURE 5.13 – Lattice du type d'un message

Rappelons qu'un SOU est applicable à un SDU selon leur type de donnée respectif si et seulement si :

$$\text{pour tout } z \in Z_{SOU \to SDU} : y \to x \text{ tel que } y \in Y_{SOU} \text{ et } x \in X_{SDU}$$
$$\Rightarrow datatype_{SOU}(y) \sqsupseteq datatype_{SDU}(x)$$

$$\text{pour tout } z \in Z_{SDU \to SOU} : y \to x \text{ tel que } y \in Y_{SDU} \text{ et } x \in X_{SOU}$$
$$\Rightarrow datatype_{SOU}(y) \sqsubseteq datatype_{SDU}(x)$$

$$datatype_{SOU}(fn\_cad) \sqsubset datatype_{SDU}(fn\_cad) \Rightarrow incompatible$$
$$datatype_{SOU}(fn\_ak) \sqsubset datatype_{SDU}(fn\_ak) \Rightarrow incompatible$$
$$datatype_{SOU}(fn\_comp) \sqsubset datatype_{SDU}(fn\_comp) \Rightarrow compatible$$
$$datatype_{SOU}(fn\_resp) \sqsubset datatype_{SDU}(fn\_resp) \Rightarrow compatible$$
$$datatype_{SOU}(fn\_con) \sqsubset datatype_{SDU}(fn\_con) \Rightarrow compatible$$

Cette incompatibilité à un impact direct sur les exigences d'utilisation et nécessite alors une révision du SOU. Les exigences peuvent être validées à condition d'être modifiées. C'est le cas des exigences AFN01 et AFN02 qui sont aggrégées en

une seule exigence : *si le message n'est pas valide le système doit ignorer la totalité du message.*

Par la suite nous assumons que les révisions du SOU ont été réalisées afin que le SOU soit complètement compatible avec le SDU : les messages sont de type `bool`.

5.2.5 Validation des calculs

Dans cette section, nous tentons d'illustrer la compatibilité du SOU et du SDU selon leur interaction dynamique. Nous proposons divers scénarios afin de couvrir les différents cas possibles.

5.2.5.1 Définition comportementale formelle des composants du SOU

Le scénario proposé consiste à valider le processus de "Requète pour Notification" dans les conditions nominales. Ce scénario a été elaborée pour valider les exigences du système ATC suivantes :

AFN01 Lors de la réception d'un message de recommandation de prise de contact valide, le système doit transmettre un message de réponse, puis il doit envoyer un message de prise de contact à la prochaine station sol.

AFN02 Lors de la réception d'un accusé de réception provenant de la prochaine station sol, le système doit transmettre un message de complétion à la première station sol.

Scénario 1 "Requète pour Notification" dans les conditions nominales

1. GENERATE : Envoyer un message FN_CAD valide depuis la station sol LFDG.
2. OBSERVE : Réception d'un message AFN_RESP sur la station sol LFDG.
3. OBSERVE : Réception d'un message FN_CON sur la prochaine station sol LFCQ.
4. GENERATE : Envoyer un message FN_AK valide depuis la station sol LFCQ.
5. OBSERVE : Réception d'un message FN_COMP sur la station sol LFDG.

Les composants `generator` et `transducer` sont décrit pas l'IOA de la figure 5.14. Pour des raisons de simplicité, nous avons défini un seul composant `GenTrans` qui réalise à la fois les stimulations et les observations. Les services `AfnMessageTransmission` et `AfnMessageReception` sont implémentés en utilisant des variables partagées, e.g. `fn_cad = true` signifie qu'il y a un message de recommandation de prise

de contact à envoyé. Les fonctions `setCadValidity(bool)` et `setAkValidity(bool)` permettent d'assigner la validité du message.

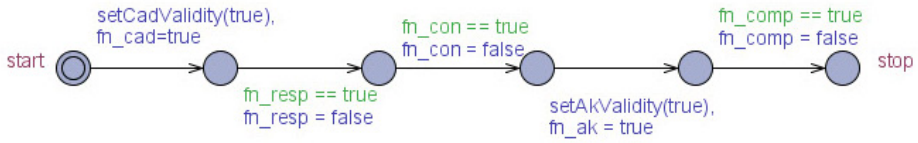

FIGURE 5.14 – IOA du composant GenTrans pour le scénario 1

Le composant `acceptor` est spécifié à l'aide des TICs suivants.

1. Si la propriété `A◊ fn_ak imply fn_comp` est vraie, cela signifie que si un accusé de réception valide est envoyé, alors à un moment donné, un message de complétion sera transmis.
2. Si la propriété `A◊ fn_cad imply fn_resp` est vraie, cela signifie que si un message de recommandation de prise de contact valide est envoyé, alors à un moment donné, un message de réponse sera transmis.
3. Si la propriété `A◊ fn_cad imply fn_con` est vraie, cela signifie que si un message de recommandation de prise de contact valide est envoyé, alors à un moment donné, un message de contact sera transmis.

5.2.5.2 Définition comportementale formelle des composants du SDU

Le SDU est constitué de trois composants LFDG, AFN, et LFCQ (figure 5.15). Nous utilisons les canaux de synchronisation Uppaal pour l'échange des messages entre ces trois composants.

5.2.5.3 Vérification de l'applicabilité du SOU sur le SDU selon leur intéraction dynamique respective

Nous pouvons vérifier dans quel cas 1, 2, 3 ou 4 (4.2.2.6) nous nous trouvons grâce aux propriétés suivantes.

Si la propriété `A◊ GenTrans.stop` est vraie, cela signifie que tous les chemins permettent d'atteindre l'état `stop` du composant GenTrans. Dans ce cas toutes les traces du SDU restreintes au vocabulaire du SOU sont des traces du SOU :

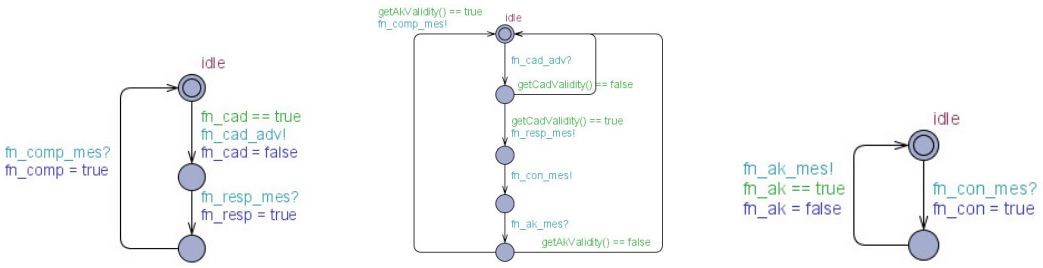

FIGURE 5.15 – IOA du SDU

$\|SDU\|_{VSDU/VSOU} = \|SOU\|_{VSOU}$. Dans cet exemple la propriété est fausse. Ceci est du au fait, par exemple, que le SDU prend en compte le cas où le message de recommandation de prise de contact n'est pas valide.

Si la propriété `E◇ GenTrans.stop` est vraie, cela signifie qu'il existe un chemin permettant d'atteindre l'état `stop` du composant GenTrans. Le résultat fournit par le vérificateur est vrai, nous nous trouvons donc dans le cas numéro 1 où le comportement du SOU est dans l'enveloppe du SDU mais le SOU n'explore pas toutes les exécutions du SDU.

Si `A□ GenTrans.stop imply LFCQ.idle && LFDG.idle && AFN.idle` est vraie alors le lorsque le scénario est terminé, la simulation est revenue à son état initial.

La figure 5.16 illustre le résultats des vérifications proposées ci-dessus.

FIGURE 5.16 – Vérification des propriétés du scénarios 1

Conclusion générale

Conclusion

L'objectif de cette étude a été de proposer une approche générale d'évaluation de la validité d'une M&S utilisée dans le cadre du développement des systèmes embarqués. Cette approche s'inscrit dans une démarche visant à améliorer la confiance en l'utilisation d'une simulation dont les résultats sont souvent remis en cause sans justification cohérente. Une contribution de cette étude est de définir un cadre de justification.

À long-terme, l'objectif d'Airbus est de se doter d'une stratégie standardisée et d'un guide de bonnes pratiques, des activités de validation des simulations, disponibles pour tous les produits de simulation dans un domaine technologique donné. Le défi est de mettre à la disposition des utilisateurs de simulation un "label" objectif et formel de validité, d'améliorer la valeur ajoutée de la simulation dans les tâches d'ingénierie et enfin d'estimer et de minimiser les charges de V&V des simulateurs.

Au regard d'un objectif de validation d'un système d'intérêt, une simulation doit être la plus proche possible du système de l'avion qu'elle représente tout en respectant des contraintes de coûts et de délais de mise à disposition. Les simulations doivent être disponibles avant les systèmes eux-mêmes. Si le niveau de validité est trop faible les résultats nécessaires à l'expérience ne peuvent être atteints. Si le niveau de validité est trop élevé, le temps de travail de modélisation et de calcul est inutilement dépensé.

En considérant donc que la validité d'une simulation n'est jamais évaluée dans l'absolu mais toujours au regard d'un objectif d'utilisation, nous avons défini ce problème de la validité comme l'applicabilité d'un objectif à un modèle de simulation.

La solution proposée a été un système d'évaluation de la validité qui s'appuie sur :

- un modèle de description des propriétés des systèmes embarqués et de la simulation, orienté évaluation, i.e. permettant la mise en correspondance des élements du modèle.
- des règles de mise en correspondance formelles entre un objectifs d'utilisation

et le domaine d'usage d'un modèle.

– une méthodologie d'ingénierie permettant l'application des concepts formels et montrant l'intégration de ces concepts dans un processus d'ingénierie système.

Le modèle de description des propriétés, appelé le modèle conceptuel, est le langage unificateur entre utilisateur et développeur de la simulation. Il permet, d'une part, de parler strictement de la même chose lorsque le terme de "validité" est évoqué, et d'autre part, d'évaluer la compatibilité entre un niveau de validité attendu par l'expérience et un niveau de validité fourni par le produit de simulation. Nous avons assimilé le problème de niveau de validité à une hiérarchie d'abstraction de modèle et de simulation. Nous avons donc développé une taxonomie d'abstractions, attachées aux procédures qui génèrent ces abstractions et illustrée par des pratiques courantes du développeur de modèles. Cette taxonomie a pour objectif d'établir le lien entre hypothèses de M&S et niveaux d'abstraction d'un modèle.

Nous avons basé ce modèle sur le concept de cadre expérimental. Ce concept, proposé dans la théorie de la M&S nous a permis de traité cette problématique dans un cadre méthodologique fondé. Il nous a permis, en outre, de palier un problème récurrent chez Airbus selon lequel l'objectif d'utilisation de la simulation n'est souvent pas bien défini. Nous avons montré que le modèle n'est pas l'unique cause d'un éventuel biais d'une simulation mais que l'expérimentation qui est faite avec ce modèle peut aussi avoir été mal définie. Il est important de noter également que le cadre expérimental peut aussi décrire, outre un cadre d'expérimentation et un cadre de validité, un cadre de réutilisation (description du nouveau contexte d'utilisation ou d'expérimentation) et un cadre d'interopérabilité (description de la manière dont le modèle va communiquer). Ainsi l'évaluation de la validité peut être conduite soit :

– sur le développement ab initio d'une simulation nécessaire et suffisante pour satisfaire un objectif d'utilisation,

– sur la réutilisation d'une simulation déjà existante pour satisfaire un objectif d'utilisation.

L'idée de définir des propriétés auquelles nous pouvons attacher des attributs, tel que l'illustre le diagramme Kiviat de la figure 5.17 [Balci 00], afin de permettre à l'utilisateur d'exprimer le niveau de validité requis, au développeur de caractériser le domaine d'usage de la simulation qu'il peut de fournir, puis de vérifier si le second couvre le premier, nous rapidement paru être une approche raisonnable.

Cependant, nous avons été rapidement confrontés au problème du niveau de détails de ces propriétés qui conduit à la question de savoir à quel niveau de décomposition du produit de simulation ces critères doivent adresser. En réalité et idéalement, la réponse serait "à tous". En effet, nous avons pu constater que les abstractions sont faites tout le long du cycle de développement de la simulation.

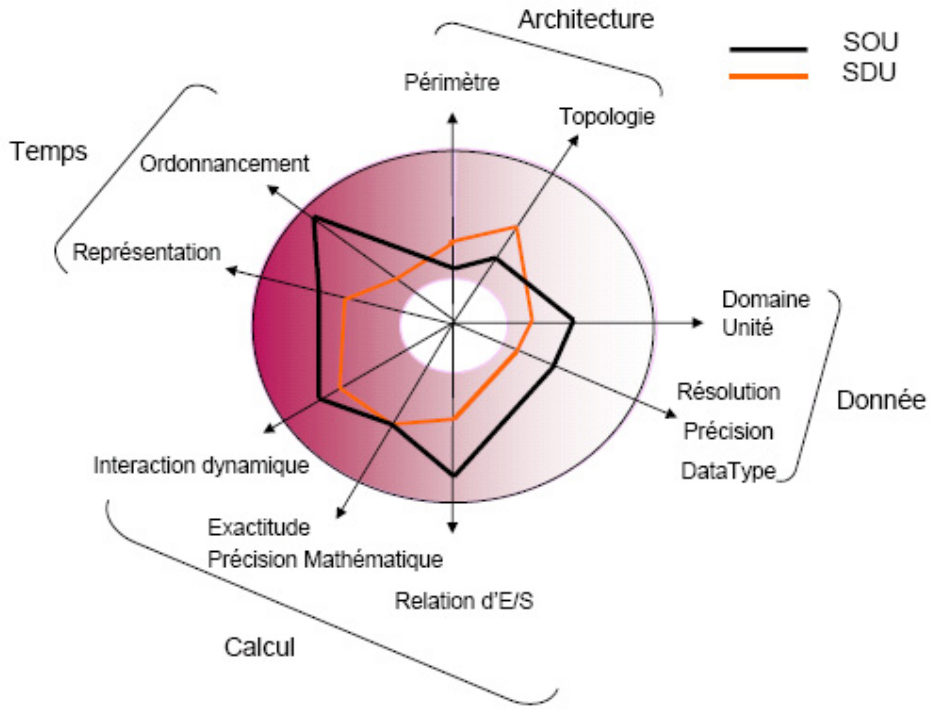

FIGURE 5.17 – Diagramme Kiviat d'une simulation

Dans la phase de développement du modèle conceptuel, les abstractions sont le fruit de la perception que l'ingénieur se fait du système réel. Dans le contexte de l'EIA-632, nous dirions : "Quelles sont les exigences de l'acquéreur qui ne sont pas prises en compte ? Celles qui ont été prises en compte, ont-elles été comprises et sont-elles des exigences techniques réelles ?

Dans la phase de spécification d'un modèle mathématique, i.e les différentes représentation d'un "building block", les abstractions sont dues au paradigme et formalisme de modélisation utilisés et à toutes les contraintes de discrétisation que nous avons largement évoqués par ailleurs.

Enfin, dans la phase d'implémentation d'un modèle informatique, nous retrou-

vons les abstractions dues au contraintes de l'infrastructure de la plateforme d'exécution (ordonnanceurs, communications, processeurs...).

L'idée de proposer des critères selon les quatres domaines, architecture, donnée, calcul et temps, et selon trois couches, conceptuelle, mathématique, informatique (figure 5.18) nous a effleuré l'esprit, mais fût rapidement abandonné. Elle nous a cependant aidés dans notre réflexion en particulier lorsque nous avons tenté de raffiner ces domaines en critères mesurables. Le cas le plus évident est celui concernant l'aspect *donnée*, où, au niveau conceptuel nous pouvons définir l'information véhiculée par la donnée, au niveau mathématique, nous pouvons caractériser la résolution, le domaine, etc... de la donnée et au niveau implémentation, nous pouvons caractériser son type informatique et sa précision. Cependant nous ne pouvions pas tout traiter dans le cadre de cette étude. En revanche, nous pensons que la proprosition de définir des propriétés et des critères selon ces quatres domaines et ces trois couches peuvent servir de base au raffinement d'autres critères plus spécifiques, selon le produit de simulation et la discipline (mécanique, hydraulique, information...).

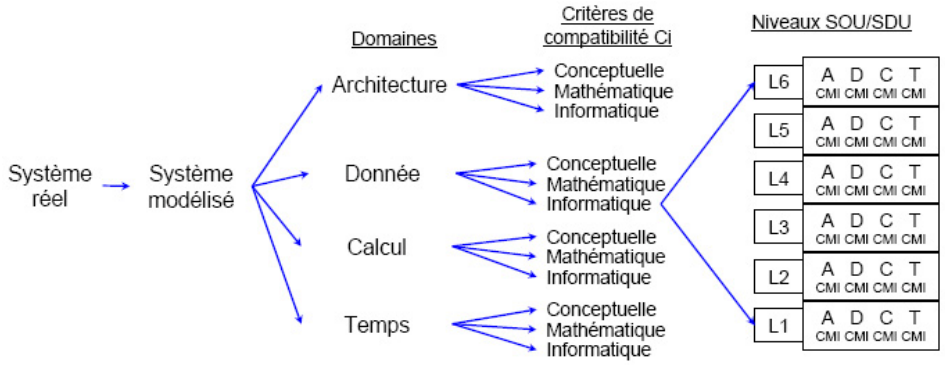

FIGURE 5.18 – Niveau de simulation selon quatres domaines et trois couches

Le problème de la mise en correspondance est fondé sur le principe qu'un objectif d'utilisation et un domaine d'usage d'une simulation sont deux composants, au sens formel du terme, i.e., qui interagissent à travers leurs interfaces et uniquement à travers leurs interfaces. Nous nous sommes tournés vers les techniques de l'ingénierie basée composants, pour itérativement enrichir le concept de cadre expérimental de "concepts symboliques" et couvrir les éléments de la taxonomie d'abstraction.

Puisque la M&S est un des processus de l'ingénierie systèmes, nous avons pro-

posé une méthodologie permettant d'intégrer nos concepts formels à un tel processus afin d'aboutir au modèle conceptuel souhaité. En partant des objectifs d'utilisation, i.e. les exigences d'un système d'intérêt et le plan de V&V destiné à valider les exigences du système d'intérêt, l'utilisateur décrit successivement (la même démarche est appliquée par le développeur de la simulation) :

- un modèle des scénarios de tests via un diagramme de séquence,
- une vue logique de l'expérimentation : ontologie du domaine comme un ensemble de blocs, leurs attributs et leurs relations, i.e. les entrées par lesquelles les blocs sont capables de réagir et les sorties produites par les blocs,
- une vue physique de l'expérimentation : la topologie de l'expérimentation comme un ensemble de composants (générateur, tranducteur et accepteur), de ports, de valeur sur les attributs et de connecteurs,
- une modélisation dynamique : les états et changements d'états possibles de chacun des blocs.

Futurs travaux

Si nous avons évoqué la nécessité d'un domaine *temps* comme composante d'une hiérarchie d'abstraction de modèle et de simulation, nous n'avons que très peu traité ce sujet. De considérer les trois temps, horloge murale, temps mathématique et temps du simulateur est un début, voir un prérequis et, finalement tout à fait applicable à notre concept des trois couches. Comme dans tout autre domaine il faut y apposer des propriétés et des critères. Nous avons bien commencé à réfléchir à la question, mais nous n'avons pas encore formalisé ces propriétés en termes de concepts symboliques, hormis pour le temps mathématique que nous traitons comme une donnée classique. Voici quelques éléments de notre réflexion. Nous pouvons caractériser le temps selon deux propriétés :

1. Comment le temps est-il représenté ? Des exemples d'attributs sont : causal, synchrone et physique/temps-réel. Un temps causal considère l'ordre d'apparition des évènements, e.g. l'évènement A se produit avant l'évènement B. Un temps synchrone offre une notion quantitative du temps mais est représenté à une autre échelle que le temps physique/temps réel. Il a été dilaté ou contracté d'un facteur k. Les simulations dilatées ou contractées sont, malgré

tout, très utilisée sur les simulateurs Airbus. Si le temps de la simulation est physique/temps-réel, cela signifie qu'il est identique à celui du système réel.

2. Quels types de propriétés temporelles est-il possible d'observer ? Des exemples d'attributs sont :
   - Réponse bornée : entre deux éléments observables seulement l'écoulement d'une période maximale de temps est permise.
   - Durée : une propriété doit être valide pour une quantité donnée de temps.
   - Fréquence : un élément observale doit se produire n fois durant m uniités de temps.
   - Temporisation : un signal est déclenché après une quantité donnée de temps.

Nous pensons qu'une taxonomie des abstractions est un excellent moyen, (1) de construire une hiérarchie de modèle et (2) de fournir un langage commun entre utilisateurs et développeurs de la simulation qui ont des préoccupations souvent bien différentes. La formalisation de l'impact d'une hypothèse de modélisation sur les propriétés d'une simulation serait une excellente avancée vers une automatisation de la M&S. Un prochain travail devrait consister à formaliser les procédures d'abstractions sur des concepts formels. Les travaux de B.P Zeigler [Zeigler 00b] peuvent constitués une base de réflexions sur le sujet notamment grâce aux concepts formels d'homomorphisme et de morphisme approximatif que nous avons évoqué. Les travaux de [Sevinc 91] basés sur une définition modifiée d'homomorphisme, décrivent un ensemble de programmes qui génèrent des versions simplifiées de modèles à partir de l'exécution d'une simulation. N'oublions pas les travaux issus du domaine de l'intelligence artificielle [Rickel 94], [Forbus 96], [Iwasaki 93], [Falkenhainer 91] qui peuvent tour à tour concerné la formalisation de chacune des boîtes de notre taxonomie.

L'intégration de la notion de cadre expérimental et de nos concepts formels dans un processus d'ingénierie système constitue une valeur ajoutée pour le développement des systèmes. Cela permet de répondre à une problématique connue qui est d'assurer une traçeabilité entre les différents produits issus des étapes de développement et de réduire la distance entre développement d'un système et développement d'une M&S. Cela permet également d'aller vers une automatisation de la M&S et d'ouvrir la voie à des moyens de vérification formelle. Le domaine des calculs et plus particulièrement l'étude de la compatibilité selon l'interaction dynamique mériterait

d'être approfondie. Le générateur définit l'ordre logique des stimuli injectés dans la simulation. Le transducteur définit l'ordre logique des observations. Nous avons spécifié ces deux composants à l'aide d'automates. L'accepteur, spécifié à l'aide de logique temporelle, consiste à vérifier si une exigence a été vérifiée à partir de paires de stimulus/observation. En ce qui concerne les propriétés liées à l'évaluation de la compatibilité entre le SOU et le SDU, nous avons également défini des propriétés à logique temporelle sur des états du SOU afin de déterminer dans quel cas de compatibilité nous nous trouvions. Si cette approche est cohérente, un effort reste à faire pour trouver des propriétés de compatibilité plus ou moins systématiques.

De nombreux travaux traitent de l'étude de la compatibilité à la fois statique et dynamique de composants. Nous avons détaillé ceux de [Weisel 03] dans le domaine de la M&S et nous avons évoqué ceux de [Meyer 88] [Beugnard 99] et [Carrez 03] dans le domaine du logiciel. D'autres travaux importants, notamment ceux de [Gössler 05] et [Sifakis 05] meritent une attention particulière. Enfin dans [Talcott 98], [Arbab 05], [Lee 02], [Zhou 08] , les auteurs contribuent à ces aspects, en proposant des méthodes d'analyse de signatures et spécifications basées sur la notion d'acteurs qui communiquent à travers des ports typés (types statiques et comportementaux) et dont les concepts sont mise en oeuvre par la plateforme Ptolemy.

Nous remarquons que nous n'avons pas parler de DEVS dans cette étude. DEVS vient plus tard, dès lors que nous avons choisi un paradigme de modélisation. Travailler sur la structure algébrique de la hiérarchie des systèmes nous a permis de rester indépendant du paradigme ouvrant la voie vers l'extension de certaines des propriétés sur, par exemple, les systèmes hybrides. Le cas d'application que nous avons présenté est un système à évènements discrets. Nous aurions pu utiliser dans ce cas DEVS pour la modélisation dynamique. Mais, à notre connaissance, DEVS n'offre pas d'outils de vérification formelle et de preuve formelle, c'est pourquoi nous avons opté pour les automates à interfaces. Notons que des travaux permettent la transformation de modèle DEVS, en langage de spécification formelle permettant la formulation de preuves par logique combinatoire. Dans [Traoré 05], l'auteur propose une transformation de DEVS vers le langage Z.

En revanche DEVS permet une simulation dynamique des modèles grâce au "simulateur abstrait". Nous n'avons traité que de critères statiques et dynamiques

vérifiables statiquement. Nous souhaiterions, par la suite modéliser, la dynamique des SOU et SDU sous forme de composants DEVS afin de vérifier la compatibilité d'un SOU et d'un SDU selon une simulation dynamique. Des outils tel que DEVS-JAVA [Zeigler 00a], Mimosa [Muller 04] ou MatlabDEVS [Deatcu 03] offre de telles possibilités.

Comme nous l'avons précisé plus haut, Ptolemy permet la spécification de systèmes sous la forme de composition d'acteurs. Dans le même esprit que DEVS et son simulateur abstrait, un acteur Ptolemy peut être exécuté grâce aux "model of computation". Il existent de nombreux "model of computation" implémentés dans Ptolemy permettant la modélisation et l'exécution de systèmes sous diverses formes : continu, flots de données synchrones, temps discret, évènements discrets, "periodic time-driven"... Cela montre à quel point il est également primordial de s'intéresser aux relations entre les formalismes de représentations des systèmes qui eux aussi génèrent des abstractions du système réel.

Bibliographie

[Addanki 91] Sanjaya Addanki, Roberto Cremonini & J. Scott Penberthy. *Graphs of models.* Artif. Intell., vol. 51, no. 1-3, pages 145–177, 1991. 51

[AFIS] AFIS. Association Française d'Ingénierie Système. http ://www.afis.fr/. 10

[Albert 05] Vincent Albert, Alexandre Nketsa & Jean-Claude Pascal. *Towards a metal-model based approach for hierarchical Petri net transformations to VHDL.* European Simulation and Modelling Conference, pages 531–536, 2005. 52

[Albert 07a] Vincent Albert, Alexandre Nketsa, Mario Paludetto, Christel Seguin, René Jacquart, Antoine Casta & Guy Laporte. *Simulation Validity Assessment : Problem formulation.* Rapport technique, LAAS-CNRS, 2007. 18, 43

[Albert 07b] Vincent Albert, Alexandre Nketsa & Jean-Claude Pascal. *Criteria and methods to establish the valid interaction of a simulation and its intended purpose.* European Simulation and Modelling Conference, pages 29–36, 2007. 45

[Albert 09a] Vincent Albert, Alexandre Nketsa, Mario Paludetto, Christel Seguin, René Jacquart & Jean Casteres. *Simulation Validity Assessment : simulation objectives of use description guidelines.* Rapport technique, LAAS-CNRS, 2009. 84

[Albert 09b] Vincent Albert, Alexandre Nketsa & Jean-Claude Pascal. *Signature matching applied to simulation/frame duality.* The Fourth International Conference on Systems (ICONS 2009), pages 190–196, 2009. 84

[Alfaro 01] Luca Alfaro & Thomas A. Henzinger. *Interface automata.* Dans Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering (FSE), ACM, pages 109–120. Press, 2001. 75, 96

[Arbab 05] Farhad Arbab. *Abstract behavior types : a foundation model for components and their composition.* Dans Sci. Comput. Program., volume 55, pages 3–52, Amsterdam, The Netherlands, 2005. Elsevier North-Holland, Inc. 131

[ARINC 95] ARINC. *ARINC 429 Mark 33 Digital Information Transfer System.* Aeronautical Radio, Incorporated, 1995. 61

[ARINC 05] ARINC. *ARINC 623-3 Character-Oriented Air Traffic Service (ATS) Applications.* Aeronautical Radio, Incorporated, 2005. 57

[Balci 00] Osman Balci, William F. Ormsby, John T. Carr III & Said D. Saadi. *Planning for verification, validation, and accreditation of modeling and simulation applications.* Dans WSC '00 : Proceedings of the 32nd conference on Winter simulation, pages 829–839, San Diego, CA, USA, 2000. Society for Computer Simulation International. 126

[Bertalanffy 72] Ludvig Von Bertalanffy. The history and status of general systems theory. éditions Wiley-interscience, 1972. in Trends in general systems theory, G. J. Klir. 4

[Beugnard 99] Antoine Beugnard, Jean-Marc Jézéquel, Noël Plouzeau & Damien Watkins. *Making Components Contract Aware.* Dans Computer, volume 32, pages 38–45, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press. 75, 131

[Bigelow 03] James H. Bigelow & Paul K. Davis. *Implications for Model Validation of Multiresolution, Multiperspective Modeling (MRMPM) and Exploratory Analysis.* Research and Development Corporation, 2003. disponible sur http ://handle.dtic.mil/100.2/ADA439815. 2

[Brade 00] Dirk Brade. *VV&A II : enhancing modeling and simulation accreditation by structuring Verification and Validation results.* Dans WSC '00 : Proceedings of the 32nd conference on Winter simulation, pages 840–848, San Diego, CA, USA, 2000. Society for Computer Simulation International. 19, 28, 29, 30

[Carrez 03] Cyril Carrez, Alessandro Fantechi & Elie Najm. *Behavioural Contracts for a Sound Assembly of Components*. Dans In FORTE, pages 36–39. Springer, 2003. 75, 131

[Cousot 92] Patrick Cousot & Rahida Cousot. *Abstract interpretation and application to logic programs*. Dans J. Log. Program., volume 13, pages 103–179, New York, NY, USA, 1992. Elsevier Science Inc. 61, 62

[Deatcu 03] Christina Deatcu. *Development of an Object-Orientated DEVS-Simulator with Matlab*, 2003. 132

[DMSO 96] DMSO. *Verification, Validation and Accreditation* (VV&A) *Recommended Practices Guide* (RPG). Defense Modeling and Simulation Office (DMSO), 1996. disponible sur http ://vva.msco.mil/. 2

[DMSO 00a] DMSO. *Fidelity, RPG Special Topic*. Defense Modeling and Simulation Office (DMSO), 2000. 30

[DMSO 00b] DMSO. *Validation, RPG Reference Document*. Defense Modeling and Simulation Office (DMSO), 2000. disponible sur http ://vva.msco.mil/. 30

[DMSO 00c] DMSO. *V&V Tools, RPG Reference Document*. Defense Modeling and Simulation Office (DMSO), 2000. disponible sur http ://vva.msco.mil/. 30

[DMSO 01] DMSO. *V&V Techniques, RPG Reference Document*. Defense Modeling and Simulation Office (DMSO), 2001. disponible sur http ://vva.msco.mil/. 27

[DMSO 04] DMSO. *Requirements, RPG Reference Document*. Defense Modeling and Simulation Office (DMSO), 2004. disponible sur http ://vva.msco.mil/. 33

[EIA 99] EIA. *Processes for Engineering a System : EIA-632*. EIA Electronics Industries Alliance, Janvier 1999. 3, 9, 12, 28, 103, 104

[ESTEREL 08] ESTEREL. *Efficient Development of Safe Avionics Software with DO-178B Objectives Using SCADE Suite*. Esterel

Technology, 2008. disponible sur http ://www.esterel-technologies.com/files/AeronauticsHandBook-DO178B.pdf. 43

[Falkenhainer 91] Brian Falkenhainer & Kenneth D. Forbus. *Compositional modeling : finding the right model for the job*. Artif. Intell., vol. 51, no. 1-3, pages 95–143, 1991. 47, 51, 130

[Forbus 96] Kenneth D. Forbus. *Qualitative Reasoning*, 1996. 62, 130

[Frantz 95] Frederick K. Frantz. *A taxonomy of model abstraction techniques*. Dans WSC '95 : Proceedings of the 27th conference on Winter simulation, pages 1413–1420, Washington, DC, USA, 1995. IEEE Computer Society. 6, 25, 48, 50

[Godfrey-Smith 03] Peter Godfrey-Smith. Theory and reality : an introduction to the philosophy of science. University of Chicago Press, Chicago, IL, USA, 2003. 24

[Gössler 05] Gregor Gössler & Joseph Sifakis. *Composition for component-based modeling*. Sci. Comput. Program., vol. 55, no. 1-3, pages 161–183, 2005. 131

[INCOSE 04] INCOSE. *System Engineering Handbook : A "What to" Guide for all Practitioners*. International Council on Systems Engineering, 2004. 3

[ISO 03] ISO. *System Life-Cycle Processes*. ISO 15288, Novembre 2003. 28

[ITOP 04] ITOP. *General procedure for planning, implementing, and documenting Verification & Validation efforts relating to Models and Simulations to support the exchange of this information*. International Test Operations Procedure, 2004. 3, 28

[Iwasaki 93] Yumi Iwasaki. *Reasoning with multiple abstraction models*. Recent advances in qualitative physics, pages 67–82, 1993. 47, 54, 66, 130

[Kuipers 90] Benjamin Kuipers. Abstraction by time-scale in qualitative simulation. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. 47, 52

[Larsen 97] Kim G. Larsen, Paul Pettersson & Wang Yi. *UPPAAL in a Nutshell*, 1997. 109

[Lee 98] Edward A. Lee & Alberto Sangiovanni-vincentelli. *A Framework for Comparing Models of Computation.* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 17, pages 1217–1229, 1998. 79

[Lee 02] Edward.A. Lee & Ye Xiong. *Behavioural Types for Component-Based Design.* Technical Memorandum UCB/ERL M02/29, 2002. 75, 131

[Maier 96] Mark W. Maier. *Integrated Modeling : A Unified Approach to System Engineering.* Dans Journal of Systems and Software, 1996. 37

[Meyer 88] Bertrand Meyer. *Eiffel : a language and environment for software engineering.* J. Syst. Softw., vol. 8, no. 3, pages 199–246, 1988. 75, 131

[Mitchell 02] Richard Mitchell, Jim McKim & Bertrand Meyer. Design by contract, by example. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2002. 75

[Muller 04] Jean-Pierre Muller. *Modélisation et Simulation individu-centrées*, 2004. disponible sur http ://lil.univ-littoral.fr/Mimosa/. 132

[NAVY 04] NAVY. *Verification, Validation and Accreditation* (VV&A) *Recommended Practices Guide* (RPG). Navy Modeling and Simulation Management Office, 2004. disponible sur https ://nmso.navy.mil/VVA/tabid/58/Default.aspx. 2

[Nayak 96] P. Pandurang Nayak & Leo Joskowicz. *Efficient compositional modeling for generating causal explanations.* Dans Artif. Intell., volume 83, pages 193–227, Essex, UK, 1996. Elsevier Science Publishers Ltd. 51

[Nguyen 02] Binh T. Nguyen, Michel Delaunay & Chantal Robach. *Testability Analysis for Software Components.* ICSM '02 : Proceedings

of the International Conference on Software Maintenance, page 422, 2002. 68, 93

[Oberkampf 04] William L. Oberkampf, Timothy G. Trucano & Charles Hirsch. *Verification, validation, and predictive capability in computational engineering and physics.* Appl. Mech. Rev., vol. 57, no. 5, pages 345–385, 2004. 29

[OMG 01] OMG. *Model Driven Architecture.* Object Management Group, 2001. disponible sur http ://www.omg.org/mda/. 34

[OMG 04] OMG. *UML 2.0 OCL Specification.* Object Management Group, 2004. disponible sur http ://www.omg.org/docs/ptc/03-10-14.pdf. 75

[Pace 99] Dale K. Pace. *Development and Documentation of a Simulation Conceptual Model.* volume 77, pages 192–210, 1999. 31

[Pace 02] Dale K. Pace & Jack Sheehan. *Subject Matter Expert (SME)/Peer Use in M&S V&V.* Dans in Proc. Workshop on Foundations for Modeling and Simulation (M&S) Verification and Validation (V&V) in the 21st Century, SCS, pages 22–24, 2002. 33

[REVVA 04] REVVA. *VV&A methodological guidelines Reference Manual.* Common Validation, Verification and Accreditation Framework for Simulation, 2004. disponible sur http ://www.revva.eu/. 14, 28, 29

[Rickel 94] Jeff Rickel & Bruce Porter. *Automated modeling for answering prediction questions : selecting the time scale and system boundary.* Dans AAAI'94 : Proceedings of the twelfth national conference on Artificial intelligence (vol. 2), pages 1191–1198, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence. 47, 50, 130

[Roza 99] Manfred Roza, David Gross & Scott Harmon. *Report Out of the Fidelity Experimentation ISG.* Spring Simulation Interoperability Workshop, 1999. 2

[Rumbaugh 99] James Rumbaugh, Ivar Jacobson & Grady Booch, editeurs. The unified modeling language reference manual. Addison-Wesley Longman Ltd., Essex, UK, UK, 1999. 28

[Rutherford 00] Brian M. Rutherford, Kathleen V. Diegert, Kenneth F. Alvin, Sharon M. Deland, Brian M. Rutherford & Kathleen V. Diegert. *Estimation of Total Uncertainty in Modeling and Simulation.* Sandia National laboratories, 2000. 3

[Sargent 05] Robert G. Sargent. *Verification and validation of simulation models.* Dans WSC '05 : Proceedings of the 37th conference on Winter simulation, pages 130–143. Winter Simulation Conference, 2005. 26

[Schulz 97] S. Schulz, J. W. Rozenblit & K. Buchenrieder. *Towards an Application of Model-Based Codesign : An Autonomous, Intelligent Cruise Controller.* Dans Proceedings of the 1997 IEEE Conference and Workshop on Engineering of Computer Based Systems, pages 73–80, 1997. 59

[SEI 06] SEI. *Capability Maturity Models Integration.* Software Engineering Institute, 2006. disponible sur http ://www.sei.cmu.edu/cmmi/. 3, 28

[Sevinc 91] Suleyman Sevinc. *Theories of discrete event model abstraction.* Dans WSC '91 : Proceedings of the 23rd conference on Winter simulation, pages 1115–1119, Washington, DC, USA, 1991. IEEE Computer Society. 130

[Sifakis 05] Joseph Sifakis. *A Framework for Component-based Construction Extended Abstract.* Dans SEFM '05 : Proceedings of the Third IEEE International Conference on Software Engineering and Formal Methods, pages 293–300, Washington, DC, USA, 2005. IEEE Computer Society. 131

[Szyperski 02] Clemens Szyperski. Component software : Beyond object-oriented programming. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002. 74

[Talcott 98] Carolyn L. Talcott. *Composable Semantic Models for Actor Theories.* Dans Higher Order Symbol. Comput., volume 11, pages 281–343, Hingham, MA, USA, 1998. Kluwer Academic Publishers. 131

[Traoré 05] Mamadou K. Traoré. *Combining DEVS and Logic.* 1st OICMS. Clermont-Ferrand, pages 307–317, 2005. 131

[Trave-Massuyes 03] Louise Trave-Massuyes, Liliana Ironi & Philippe Dague. *Mathematical foundations of qualitative reasoning.* Revues Scientifiques, AI Magazine, vol. 24, no. 4, pages 91–106, 2003. 62

[USDoD 93] USDoD. *System Safety program requirements, Appendix A : Guidance for Implementation of System Safety Program.* Department of Defense, 1993. 14, 28, 32

[Valette 79] Robert Valette. *Analysis of Petri nets by stepwise refinement.* J. Comput. Syst. Sci., vol. 18, no. 1, pages 35–46, 1979. 52

[Weisel 03] Eric W. Weisel, Mikel D. Petty & Roland R. Mielke. *Validity of Models and Classes of Models in Semantic Composability.* Proceedings of the Fall 2003 SIW, 2003. 76, 131

[Weld 92] Daniel S. Weld. *Reasoning about model accuracy.* Artif. Intell., vol. 56, no. 2-3, pages 255–300, 1992. 6, 47, 48, 51

[Whitner 89] Richard B. Whitner & Osman Balci. *Guidelines for selecting and using simulation model verification techniques.* Dans WSC '89 : Proceedings of the 21st conference on Winter simulation, pages 559–568, New York, NY, USA, 1989. ACM. 29, 30

[Xiong 02] Yuhong Xiong, Yuhong Xiong & Yuhong Xiong. *An extensible type system for component-based design.* Rapport technique, 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, 2002. 92

[Zaremski 97] Amy Moormann Zaremski & Jeannette M. Wing. *Specification matching of software components.* ACM Trans. Softw. Eng. Methodol., vol. 6, no. 4, pages 333–369, 1997. 74, 75

[Zeigler 84] Bernard P. Zeigler. Theory of modelling and simulation. Krieger Publishing Co., Inc., Melbourne, FL, USA, 1984. 4, 48

[Zeigler 00a] Bernard P. Zeigler. *DEVSJAVA Tucson : Electr. Comput. Eng. Dept.*, 2000. disponible sur http ://www.acims.arizona.edu/SOFTWARE/software.shtml. 132

[Zeigler 00b] Bernard P. Zeigler, Herbert Praehofer & Tag G. Kim. Theory of modelling and simulation. Academic Press, San Diego, California, USA, 2000. 4, 19, 22, 34, 37, 62, 78, 130

[Zhou 08] Ye Zhou & Edward A. Lee. *Causality interfaces for actor networks.* Dans ACM Trans. Embed. Comput. Syst., volume 7, pages 1–35, New York, NY, USA, 2008. ACM. 131

Simulation Validity Assessment for Embedded Systems Development

Abstract : The main objective of this thesis is to propose a general approach for assessing the validity of a Modelling and a Simulation (M&S) used during the development of embedded systems. This approach is an effort to improve confidence in the use of a simulation whose results are often questioned without consistent justification. A contribution of this thesis is to define a framework to establish this justification.

The context of the study is the set of Airbus'simulation products. In relation to a Verification and Validation plan of a system, a simulation must be as close as possible to the system it represents while respecting the constraints of cost and timely availability. The simulations must be available before the systems themselves. If the level of validity is too low, the results required for the experiment can not be reached. If the level of validity is too high, time of modelling and calculation is unnecessarily spent.

Considering that the validity of a simulation is never assessed in isolation but always in relation to a target user, we have defined the problem of validity as the applicability of a simulation objective of use.

We have treated the problem of validity level as a hierarchy of model abstractions. We therefore propose a model for describing properties of abstractions, called the conceptual model, which is the unifying language between user and developer of the simulation. It allows, first, to speak strictly the same thing when the term "validity" is mentioned, and secondly, to assess the compatibility between an expected validity level for the experience and a validity level provided by the simulation.

Then, we have established formal matching rules for mapping between an objective of use and a simulation described respectively as the conceptual model proposed. The problem of mapping is based on the principle that both are components, in the formal sense, i.e., they interact through their interfaces and only through their interfaces. We look at component-based engineering techniques to iteratively enrich the concept of "context validity" of a model by "symbolic concepts" and cover each property of the taxonomy of abstractions.

Since M&S is one process of systems engineering, we proposed a methodology to integrate our formal concepts in such a process to achieve the desired conceptual model. We illustrate this approach on an avionic communication system.

Keywords : Model, Simulation, Validity, Abstraction, Component-based, Formal Matching

Auteur : Vincent ALBERT
Titre : Evaluation de la validité de la simulation dans le cadre du développement des systèmes embarqués
Directeur de thèse : Alexandre NKETSA
Lieu et date de soutenance : Toulouse le 30 septembre 2009

Résumé :
L'objectif de cette étude est de proposer une approche générale d'évaluation de la validité d'une Modélisation et Simulation (M&S) utilisée dans le cadre du développement des systèmes embarqués. Cette approche s'inscrit dans une démarche visant à améliorer la confiance en l'utilisation d'une simulation dont les résultats sont souvent remis en cause sans justification cohérente. Le cadre d'application de l'étude est l'ensemble des produits de simulation d'Airbus. Au regard d'un objectif de validation d'un système, une simulation doit être la plus proche possible du système qu'elle représente. Dans le cycle de développement d'un avion, les simulations doivent être disponibles avant les systèmes eux-mêmes. Si le niveau de validité est trop faible les résultats nécessaires à l'expérience ne peuvent être atteints. Si le niveau de validité est trop élevé, du temps de travail de modélisation et de calcul est inutilement dépensé.
Nous avons assimilé le problème de niveau de validité à une hiérarchie d'abstraction de modèles. Nous proposons un modèle de description des propriétés d'abstractions qui permet de parler strictement des mêmes choses lorsque le terme de "validité" est évoqué et d'évaluer la compatibilité entre un niveau de validité attendu par l'expérience et un niveau de validité fourni par le produit de simulation.
Puis, nous avons établi des règles formelles de mise en correspondance d'un objectif d'utilisation et du domaine d'usage d'un modèle. Le problème de la mise en correspondance est fondé sur le principe qu'un objectif d'utilisation et un domaine d'usage d'une simulation sont deux composants, au sens formel du terme. Nous avons adapté les techniques de l'ingénierie basée composants, pour enrichir, par des techniques itératives, nos deux composants.
Enfin nous avons proposé une méthodologie permettant d'intégrer nos concepts formels au processus d'Ingénierie Systèmes. Nous illustrons cette démarche sur un système de communication avionique.
Mots clés : Modélisation, Simulation, Validité, Abstractions, Cadre experimental

Discipline administrative : Systèmes Informatiques
Intitulé et adresse du laboratoire : Laboratoire dAnalyse et dArchitecture des Systèmes - 7 avenue du Colonel Roche F-31077 Toulouse, France