Rapport de projet de recherche

Modélisation et analyse des systèmes logiciels hétérogènes

Réalisé par :
Gomes Killian
Ngamije Emmanuel

Encadré par :
Attiogbe Christian

Introduction

Ce module avait pour objectif d'initier les étudiants au travail d'un chercheur en informatique en leurs faisant partager la vie d'une équipe de recherche pendant quelques mois. Nous avons été affectés à l'équipe AeLoS, sur le sujet de "Modélisation et analyse des systèmes logiciels hétérogènes". En d'autres termes, nous avons dû étudier des façons de vérifier que l'interopérabilité entre des systèmes logiciels hétérogènes était possible, et trouver des outils nous permettant d'assurer cette interopérabilité.
Notre sujet portait plus particulièrement sur les systèmes de modélisation SysML et AADL. Nous avons dû nous familiariser avec ces outils dans un premier temps, afin de trouver de quelle façon nous pourrions utiliser des modèles appartenant à chacun des deux systèmes afin de les faire communiquer et fonctionner ensemble.
Dans ce rapport, nous vous présenterons premièrement l'équipe à laquelle nous avons été affectée, avant d'entrer dans le vif du sujet en présentant SysML et AADL, puis nous vous présenterons les pistes suivies et les recherches menées avec notre compréhension du sujet.

Equipe

Nous avons travaillé au sein de l'équipe AeLoS (**A**rchitectures **e**t **Lo**giciels **S**ûrs). Cette équipe fait partie du laboratoire LS2N (**L**aboratoire des **S**ciences du **N**umérique de **N**antes), qui est la fusion de LINA et IRCCyN.
Ce laboratoire est constitué d'environ 400-450 personnes, dont 50 % sont permanents (Enseignants-Chercheurs, Chercheurs), et 50 % sont non-permanents (Doctorants, Post-Doctorants).

Themes (AeLoS) :

- Styles de conception et d'évolution centrés architectures
- Spécification et vérification de composants logiciels, modélisation formelle, vérification, raffinement
- Multi formalisme et analyse multifacette, interopérabilité sémantique
- Modèles sémantiques de la concurrence

Membres (AeLoS) :

AeLos est constitué de 8 Enseignants-Chercheurs : Pascal ANDRE, Christian ATTIOGBE, Gilles ARDOUREL, Benoit DELAHAYE, Arnaud LANOIX, Jean-Marie MOTTU, Mourad OUSSALAH CHABANE et Dalila TAMZALIT. Parmi ceux-ci, nous n'avons interagit qu'avec Monsieur ATTIOGBE par rapport au projet.

Financements :

Le financement pour la recherche s'obtient grâce à l'Institut ou par la réalisation de projets sous contrat avec des entreprises.

Projets :

Il y a 2 Types de projets dans la recherche :
- Les projets industriels, l'équipe travaille avec une entreprise. Ce type de projet est celui qui rapporte le plus de revenus.
- Les projets académiques, l'équipe travaille sur un sujet particulier au sein de son laboratoire.

Ces 2 types de projets sont des projets axés à la fois sur la recherche et le développement.

La recherche fondamentale, c'est à dire principalement axée sur la recherche et beaucoup moins sur le développement, est une clé majeure pour atteindre l'innovation au sens premier du terme.
La recherche appliquée c'est à dire principalement axée sur le développement et moins sur la recherche, s'applique à résoudre un problème concret donné. Ce type de recherche se retrouve généralement lors de travaux effectués pour une entreprise.
L'organisation du travail d'un enseignant-chercheur est la suivante :
- Enseignement
- Recherche
- Administration
- Participer ou assister à des conférences

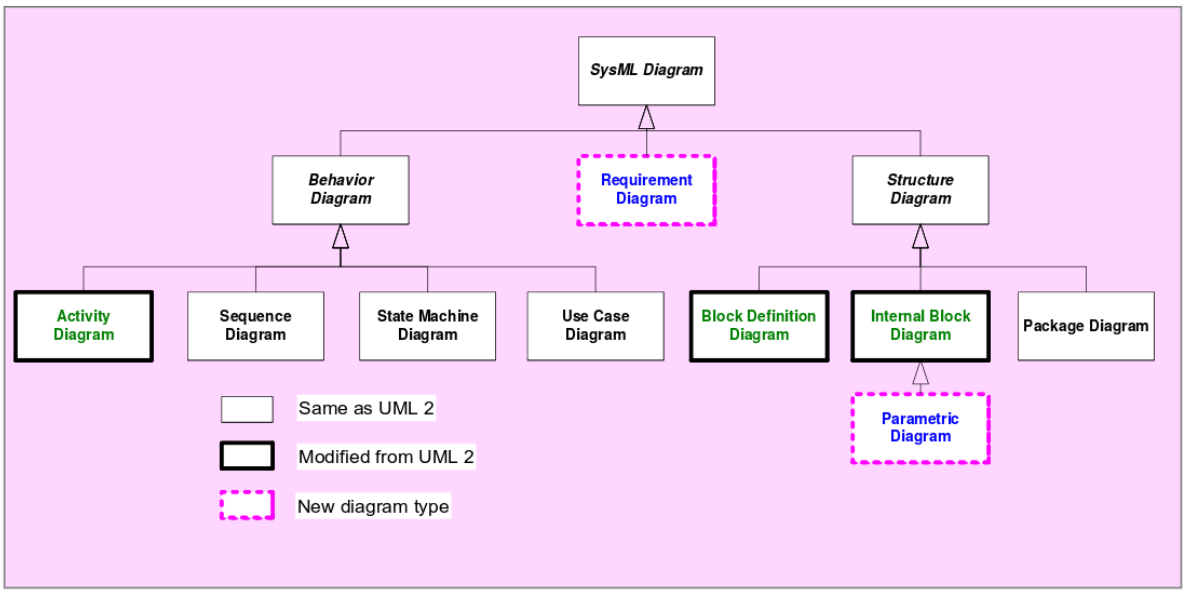
Un enseignant-chercheur travaille très souvent en dehors de se heures de travail.

Familiarisation

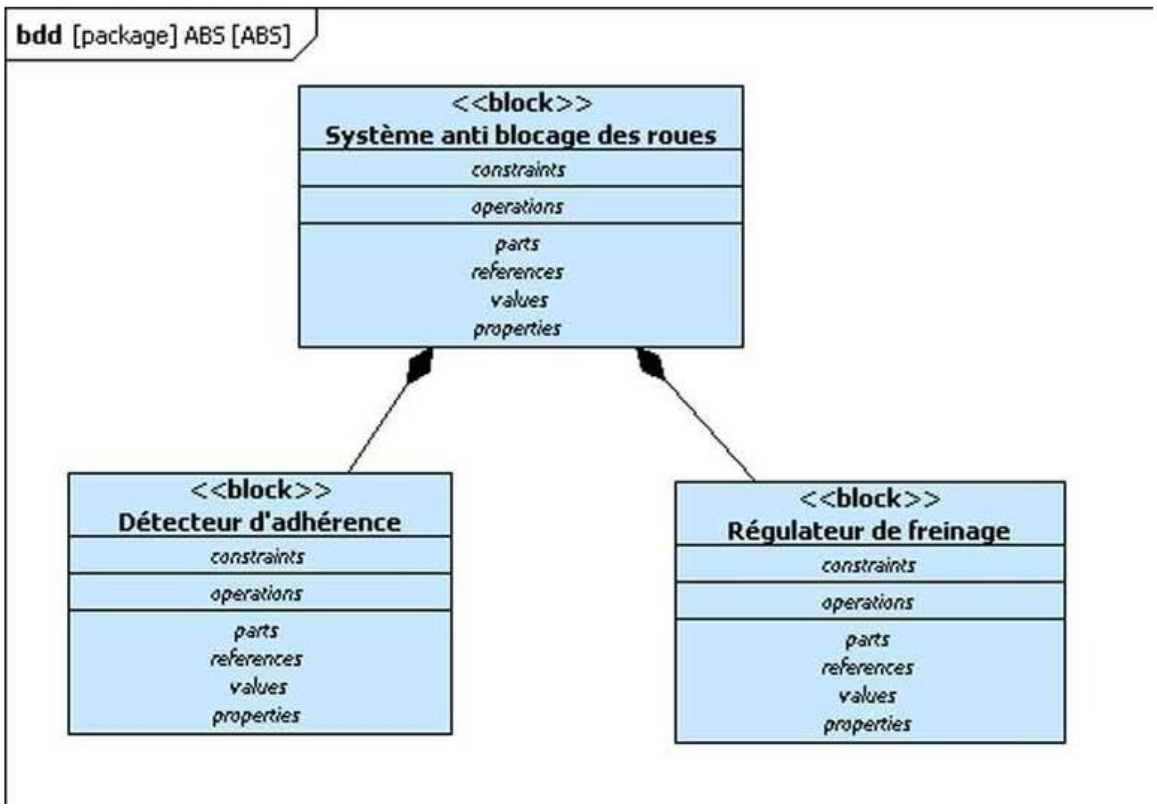
Dans cette partie, nous vous présenterons SysML, et AADL, les deux systèmes de modélisations avec lesquels nous avons dû nous familiariser. Ensuite, nous vous expliquerons la notion de contrat.

SysML

SysML est l'acronyme de *Systems Modeling Language*, soit en français par Langage de Modélisation de Systèmes.
SysML est à l'ingénierie des systèmes complexes et/ou hétérogènes ce que UML est à l'informatique. Il y a d'ailleurs de grandes similarités entre ces deux langages de modélisations. Ses motivations sont de permettre à des acteurs de corps de métiers différents de collaborer autour d'un modèle commun pour définir un système.



La conception de système donne souvent lieu à une accumulation de documentations qui doivent toutes être croisées et mises à jour pour maintenir la cohérence et respecter les spécifications du système.
La principale différence réside dans le fait que SysML utilise des « Block » quand UML utilise des « Class ». On retrouve dans cette représentation les principes d'UML mais adapté à un système qui combine de la mécanique, de la physique, de l'électronique et de la programmation.



SysML est fait pour :

- Spécifier les systèmes.
- Analyser la structure et le fonctionnement des systèmes.
- Décrire les systèmes et concevoir des systèmes composés de sous-systèmes.
- Vérifier et valider la faisabilité d'un système avant sa réalisation.

SysML intègre :

- Les composants physiques de toutes technologies.
- Les programmes.
- Les données et les énergies.
- Les personnes.
- Les procédures et flux divers.

AADL

Architecture Analysis and Design Language (AADL) est un langage de description d'architecture système destiné aux systèmes embarqués (contextes : automobile, aéronautique et spatial notamment). AADL décrit plusieurs composants qui modélisent une partie du système. Certains composants sont matériels (bus, processor, memory ...), d'autres logiciels (process, thread, subprogram, ...). Chaque composant peut avoir des propriétés (champ properties), et peut contenir des sous-composants (un processus - process - pouvant par exemple contenir plusieurs fils d'exécution - threads). On peut également décrire plusieurs machines et les relier entre elles pour simuler des connexions réseaux entre elles.

Caractéristiques générales

AADL contient plusieurs types de composants concrets :

- Catégorie (Thread, data, bus, processor, etc)
- Type : définition des interfaces (ports)
- Implémentation : définition de la structure interne des composants (sous--composant, code, spécifications Comportementale, etc)

Des propriétés (prédéfinies ou définies par le concepteur) peuvent être associés à chaque élément de modélisation (composants, ports, connections, etc). Les éléments peuvent être spécifiés dans n'importe quel ordre car c'est un langage descriptif.

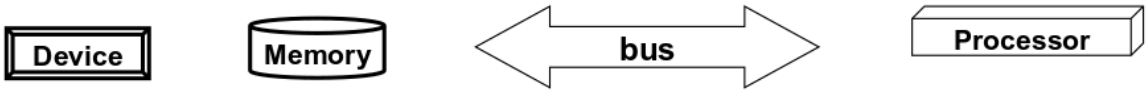
Les composants font partie de plusieurs ensembles de catégories :

- Plate--forme d'exécution (execution platform components)
- Logiciels (software components)
- Systèmes (system composition)

Description de la plate--forme d'exécution

Plusieurs catégories encore une fois :

- Processor : micro--processeur + ordonnanceur
- Memory : disque dur, mémoire vive, etc...
- Bus : réseau, etc…
- Device : composant dont on ignore la structure interne



Un processor modélise l'ensemble "processeur + noyau" contenant entre--autres un ordonnanceur. Un device sert typiquement à modéliser un ensemble "capteur + le pilote de ce capteur".

Description du logiciel

Catégories de la partie logicielle :

- Thread : fil d'exécution (ou thread dans les noyaux)
- Data : structure de données
- Process : processus, un espace mémoire pour l'exécution de threads qu'il contient
- Thread group : crée une hiérarchie dans les threads
- Subprogram : procédure, comme pour les langages de programmation. N'as pas de valeur de retour

Un process doit contenir au moins un thread.

Description système

Permet de structurer la description (matériel et logiciel)
Contient les composants qui peuvent être manipulés de façon indépendante :

- System,
- Processor,
- Memory,
- Device,
- Bus process,
- Data

Les composant : thread, thread group et subprogram ne peuvent pas être manipulés de façon indépendante.

Nous allons maintenant aborder le principe des ports, qui vont permettre de connecter les composants AADL présentés entre eux.

Description des ports

Les ports modélisent les échanges d'information.

Data : transport de données ; comme dans un circuit électronique
Event : émission d'un signal
Event data : signal + transport de données ; comparable à un message

Notion de contrat

Le contrat est une notion assez difficile à appréhender aux premiers abords, nous avons décidé de simplifier son explication en expliquant concrètement l'utilité des contrats.

Un contrat lie les composants de différents systèmes a des obligations pour assurer le fonctionnement entre eux. Grâce à cela, nous pouvons être sûr que les composants peuvent être "composés" et nous obtiendrons un résultat cohérent.

Exemple : Si nous respectons les préconditions sur les paramètres d'entrée de la fonction "f", alors nous serons sûr que les valeurs de retours de f seront cohérentes. De ce fait, si nous avons besoin de ces valeurs de retours pour les passer en paramètres à une fonction "g", et qu'ils vérifient les préconditions de g, alors nous serons sûr que le résultat de la composition (g o f) sera cohérent.

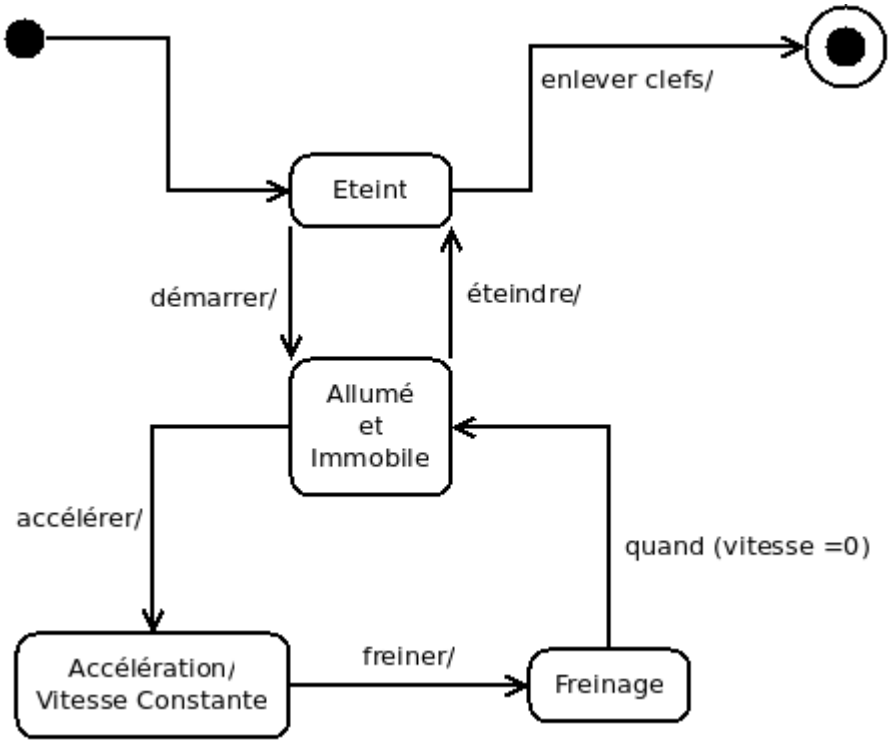
Recherche menée

Nous avons commencé le semestre sur un problème de communication qui a engendré un retard sur le début de nos activités concernant le TER. Le diagramme de Gantt prévisionnel est disponible en annexe, et nous y avons joint le diagramme de Gantt final, qui représente réellement l'avancement des tâches.
La phase de compréhension du sujet a été assez longue, et nous avons dû nous documenter sur les systèmes SysML et AADL en parallèle.

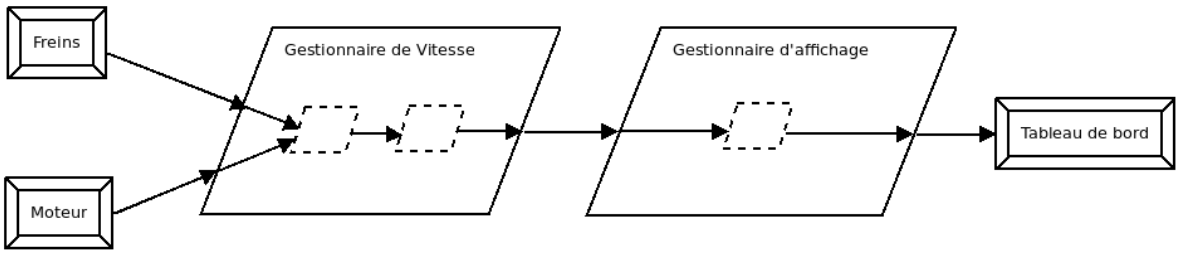
Après nous être documenté, nous avons commencé à essayer de faire des modèles. Il nous a été recommandé de modéliser le fonctionnement d'une voiture, nous nous sommes donc focalisés sur cela.
Dans un premier temps, nous pensions que nous devions faire deux modèles, un en SysML et un en AADL, qui devrait être similaire pour ensuite les comparer. Nous avons donc effectué de nombreux modèles différents en sysml, mais n'avons jamais réussi à en réaliser un similaire en AADL.
Après un rendez-vous avec notre encadrant, nous avons compris qu'il fallait en fait qu'un des deux modèles soit en fait un composant (en quelque sorte) de l'autre. Plus généralement il fallait qu'un modèle ait besoin d'utiliser l'autre. Nous avons alors choisi de modéliser le fonctionnement global d'une voiture avec un automate à état en SysML, et la gestion de la vitesse avec un modèle en AADL.

Nous vous avons joint les diagrammes finaux réalisé par la suite, qui sont ceux sur lesquels nous avons essayé de faire avancer la problématique.

Modèles sémantiques



Automate à état en SysML modélisant le fonctionnement global d'une voiture



Modèle AADL de la gestion et de l'affichage de la vitesse d'une voiture

Le but du projet serait donc maintenant de prouver que l'automate à état pourrait utiliser ce modèle AADL, en particulier le processus de gestion de vitesse, pour gérer les différentes actions.

Nous avons donc essayé par la suite de trouver des éléments qui pourraient nous permettre d'assurer ce fonctionnement, et nous sommes tombés sur des propriétés formelles faisant parti d'un système de technique de vérifications (model-checking).

Nous nous sommes dit que les propriétés, que nous vous présentons par la suite, faisaient possiblement parti du genre de vérifications que nous aurions eu à effectuer afin d'assurer la cohérence et le fonctionnement d'une composition de modèles hétérogènes, notamment avec notre modèle d'automate à états.

Propriétés à démontrer

Dans un premier temps, nous avons les propriétés non temporisées :

- Propriété d'atteignabilité (Reachability) : exprime l'accessibilité d'un état.
- Propriété de sûreté (Safety) : indique, sous certaines conditions, l'impossibilité d'atteindre un état non désiré.
- Propriété de vivacité (Liveness) : exprime qu'un état, sous certaines conditions, sera fatalement atteint.
- Propriété d'absence de blocage dit "mort" (No deadlock) : énonce que le système n'arrivera pas dans un état où il lui sera impossible d'évoluer.
- Propriété d'absence de blocage dit "vivant" (No livelock) : L'absence de livelock est une propriété similaire au deadlock qui exprime que le système ne se trouvera jamais dans une situation où il restera bloqué dans une boucle infinie.
- Propriété d'équité (Fairness) : indique qu'un évènement pourra ou ne pourra pas être atteint un nombre infini de fois.

Puis, il existe aussi des propriétés temporisées :

- Propriété de délai maximum ("promptness") : exprime un délai maximal entre un événement et sa réaction.
- Propriété de ponctualité ("punctuality") : exprime une durée exacte entre deux évènements.
- Propriété de périodicité ("periodicity") : indique qu'un événement se produira à intervalle régulier.
- Propriété de délai borné ("interval delay") : indique qu'un événement peut se produire après un autre dans un intervalle de temps précis.
- Propriété de délai minimal ("minimal delay") : exprime un délai minimal entre un événement et sa réaction.

Il existe des langages formels de spécification qui permettent d'exprimer ces types de propriétés : ce sont les logiques temporelles.
Parmi ces logiques, nous avons les logiques LTL, et la CTL. Nous nous sommes documentés sur les logiques et voici leur fonctionnement :

Logique CTL :

On raisonne sur la validité des propriétés à chaque états en faisant de la propagation de propriétés, selon des règles définies, aux états suivants. La résolution de la validité de la propriété se fait par résolution d'un système d'équations de propriétés.

Logique LTL :

Cette méthode se base sur la théorie des langages. Elle part du principe que les automates génèrent un langage spécifique à leur fonctionnement et qui peut s'exprimer sous forme d'expressions (ω−expression). On utilise alors des automates spécifiques que l'on peut générer à partir de la négation de la propriété formelle à respecter, ce qui lui permettra de détecter des langages qui violent cette propriété. Et on le synchronise avec le système à tester. Si le système viole la propriété alors l'automate de "*Büchi*" le dira.

En informatique théorique, un automate de Büchi est un ω-automate ou automate fini opérant sur des mots infinis, avec une condition d'acceptation particulière : une trace (ou calcul ou chemin infini) est réussie si et seulement si elle passe un nombre infini de fois par au moins un état acceptant. Un mot infini est accepté s'il est l'étiquette d'un calcul réussi.

Nous avons donc supposé qu'il y aurait probablement de quoi creuser dans ces voies là, mais la date butoir arrivant, nous n'avons pas eu le temps d'avancer nos recherches assez pour déterminer la validité ou non de cette piste.

Conclusion

Ce "stage" de TER (Travail d'Étude et de Recherche) nous a permis d'étendre nos connaissances sur milieu de la recherche. Les détails apportés par notre encadrant sur le fonctionnement d'une équipe de recherche nous ont intéressé. Malheureusement, nous avons mal abordé le semestre en accumulant le retard et nous avons dû faire le choix de laisser de côté le TER au profit d'autres projets et des examens, choix motivé par la difficulté du sujet et le manque de résultats durant les recherches.
Nous assumons la responsabilité de ce choix et sommes déçus de ne pas nous être plus investi dans un stage qui aurait pu nous apporter plus d'expérience sur le domaine de la recherche.