Actes des onzièmes journées nationales du

Groupement De Recherche CNRS du Génie de la Programmation et du Logiciel

Laboratoire IRIT – ENSEEIHT

11 au 14 juin 2019

Editeurs : Yamine Aït-Ameur
Aurélie Hurault
Pierre-Etienne Moreau

Table des matières

Préface

C'est avec grand plaisir que je vous accueille pour les onzièmes journées nationales du GDR GPL (Génie de la Programmation et du Logiciel) à l'ENSEEIHT et à l'IRIT.

Les missions principales du GDR GPL sont l'animation scientifique de la communauté et la promotion de nos disciplines, notamment en direction des jeunes chercheurs, mais également en direction des mondes académique et socio-économique. Cette animation scientifique est d'abord le fruit des efforts de nos groupes de travail, actions transverses et de l'Ecole des Jeunes Chercheurs en Programmation.

Le GDR GPL est maintenant dans sa onzième année d'activité. Les journées nationales sont un temps fort de l'activité de notre GDR, l'occasion pour toute la communauté d'échanger et de s'enrichir des derniers travaux présentés. Plusieurs événements scientifiques sont co-localisés avec ces journées nationales : la 8ème édition de la Conférence en IngénieriE du Logiciel (CIEL 2019) ainsi que la 18ème édition de l'atelier francophone sur les Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL 2019).

Ces journées sont une vitrine où chaque groupe de travail donne un aperçu de ses recherches. Un peu plus de vingt présentations ont ainsi été sélectionnées par les responsables des groupes de travail. Comme les années précédentes, nous avons demandé aux groupes de travail de nous proposer, en règle générale, des présentations qui avaient déjà fait l'objet d'une sélection dans une conférence nationale ou internationale ; ceci nous garantit la qualité du programme. Cette année, plusieurs sessions sont communes à plusieurs groupes de travail.

Cinq conférenciers nous ont fait l'honneur d'accepter notre invitation. Il s'agit de Romain Rouvoy (Université de Lille), de Alan Schmitt (Inria), Camille Fayollas, Olivier Flebus et Hugues Bonnin (Continental Digital Services).

Le GDR GPL a à cœur de mettre à l'honneur les jeunes chercheurs. C'est pourquoi nous décernerons un prix de thèse pour la septième année consécutive. Nous aurons le plaisir de remettre le prix de thèse GPL à Martin Clochard pour sa thèse intitulée *Méthodes et outils pour la spécification et la preuve de propriétés difficiles de programmes séquentiels*, ainsi qu'un accessit à Thomas Durieux pour sa thèse intitulée *From Runtime Failures to Patches : Study of Patch*. Le jury chargé de sélectionner le lauréat a été présidé par Pascalle Le Gall, que je remercie tout particulièrement, ainsi que l'ensemble des membres du jury.

Avant de clôturer cette préface, je tiens à remercier tous ceux qui ont contribué à l'organisation de ces journées nationales : les responsables de groupes de travail, les membres du comité de direction du GDR GPL et tout particulièrement le comité d'organisation de ces journées nationales présidé par Yamine Aït-Ameur et Aurélie Hurault. Je remercie chaleureusement l'ensemble des collègues Toulousains qui n'ont pas ménagé leurs efforts pour nous accueillir dans les meilleures conditions.

Pierre-Etienne Moreau
Directeur du GDR Génie de la Programmation et du Logiciel

Comités

Comité de programme des journées nationales

Le comité de programme des journées nationales 2019 est composé par les membres du comité de direction du GDR GPL et les responsables de groupes de travail.

Pierre-Etienne Moreau (président), LORIA, Université de Lorraine

Yamine Aït-Ameur, IRIT, ENSEEIHT
Nicolas Anquetil, CRIStAL, Université de Lille
Xavier Blanc, LaBRI, Université de Bordeaux, IUF
Mireille Blay-Fornarino, I3S, Université Nice-Sophia-Antipolis
Mathias Bourgoin, LIFO, Université d'Orléans
Julien Brunel, ONERA
Eric Cariou, LIUPPA, Université de Pau et des pays de l'Adour
Sylvain Conchon, LRI, Université Paris-Sud
Khalil Drira, LAAS, CNRS
Catherine Dubois, Samovar, ENSIIE
Anne Etien, CRIStAL, Université de Lille
Jean-Rémy Falleri, LABRI, ENSEIRB-MATMECA
Jean-Christophe Filliatre, LRI, CNRS
Aurélie Hurault, IRIT, ENSEEIHT
Alain Giorgetti, FEMTO-ST, Université de Franche-Comté
Laure Gonnord, LIP (ENS Lyon), Université Lyon 1
Akram Idani, LIG, Université Joseph Fourier
Loic Jezequel, LS2N, Université de Nantes
Nikolai Kosmatov, CEA-LIST
Régine Laleau, LACL, Université de Paris-Est Créteil
Yves Ledru, LIG, Université Joseph Fourier
Pascale Le Gall, MAS, Centrale Paris
Kevin Martin, Lab-STICC, Université de Bretagne-Sud
Clémentine Nebut, LIRMM, Université de Montpellier
Flavio Oquendo, IRISA, Université de Rennes
Fabrice Rastello, INRIA, ENS Lyon
Camille Salinesi, CRI, Université Paris 1 Panthéon-Sorbone
Julien Tesson, LACL, Université de Paris-Est Créteil
Christelle Urtado, Mines d'Alès
Virginie Wiels, ONERA

Comité scientifique du GDR GPL

Franck Barbier (LIUPPA, Pau)
Pierre Casteran (LABRI, Bordeaux)
Pierre Cointe (LINA, Nantes)
Roberto Di Cosmo (PPS, Paris VII)
Christophe Dony (LIRMM, Montpellier)
Laurence Duchien (CRIStAL, Lille)
Stéphane Ducasse (INRIA, Lille)
Marie-Claude Gaudel (LRI, Orsay)
Jean-Louis Giavitto (IRCAMS, Paris)
Yann-Gaël Guéhéneuc (Polytech, Montréal)
Gaétan Hains (LACL, Créteil)
Nicolas Halbwachs (Verimag, Grenoble)
Olivier Hermant (Mines Paris)
Valérie Issarny (INRIA, Rocquencourt)
Jean-Marc Jézéquel (IRISA, Rennes)
Dominique Méry (LORIA, Nancy)
Marc Pouzet (DI ENS, Paris)
Christel Seguin (ONERA, Toulouse)

Comité d'organisation

Yamine Aït-Ameur, IRIT - ENSEEIHT
Aurélie Hurault, IRIT - ENSEEIHT

Lynda Ait Oubelli
Sarah Benyagoub
Nasrine Damouche
Nassima Djema
Guillaume Dupont
Alexis Maffart

Conférenciers invités

Quels défis pour le développement durable des logiciels ?

Auteur : Romain Rouvoy (Université de Lille / Inria / IUF)

Résumé :
Face à la multiplication des services en ligne qui envahissent notre quotidien, la question du développement durable se pose plus que jamais pour le génie logiciel. En effet, le déploiement massif de services numériques a contribué à l'explosion de la consommation énergétiques des centres de données au cours des dernières années avec des prévisions ciblant 20% de la consommation mondiale à l'horizon 2025. Sans négliger les nombreux enjeux que posent cette évolution des usages, la réduction de l'empreinte énergétique des services logiciels revêt néanmoins un défi particulièrement critique au regard de la disponibilité limitée des ressources de notre planète. Le caractère pluri-disciplinaire de cette thématique de recherche requiert un effort coordonné dans de nombreux domaines afin de pouvoir obtenir des gains significatifs. Au cours de cette présentation, je m'efforcerai donc d'illustrer les différentes dimensions de cette thématique de recherche au travers des travaux que nous avons engagé dans l'équipe Spirals depuis 2010.

Biographie :
Romain Rouvoy est Professeur d'Informatique au sein de la Faculté des Sciences et Technologies de l'Université de Lille. Il est également membre de l'équipe-projet Inria Spirals et membre Junior de l'IUF. Ses thématiques de recherche se situent au carrefour du génie logiciel et des systèmes répartis. Il y explore notamment les problématiques du développement durable des logiciels et du respect de l'intimité des usagers.

Sémantiques Formelles de JavaScript

Auteur : Alan Schmitt (Inria)

Résumé :
Pourquoi formaliser un langage de programmation ? Parce que cela permet de mieux le définir, afin que ses utilisateurs puissent programmer plus rigoureusement. Parce que cela rend également possible la vérification formelle de propriétés de programmes, par exemple avec des assistants de preuves. Depuis 2013, nous formalisons le langage JavaScript. Dans cet exposé, nous décrirons les différentes approches que nous avons suivies, nous montrerons les outils que nous avons développés, nous aborderons nos relations avec le comité de standardisation, et enfin nous parlerons des nouveaux défis scientifiques issus de ces travaux.

Biographie :
Ancien élève de l'École Polytechnique, Alan Schmitt est chercheur à l'Inria depuis 2004, tout d'abord à Grenoble, puis à Rennes depuis 2011. Sa contribution scientifique concerne quatre projet principaux : l'expressivité et les équivalences pour les calculs de processus, la programmation bidirectionnelle, les sémantiques formelles des langages de programmation et les analyses statiques certifiées. Il a reçu en 2015 le Most Influential Paper Award de la conférence POPL pour un papier de 2005 dont il est co-auteur.

Enjeux du développement de services pour les véhicules connectés

Auteur : Camille Fayollas, Olivier Flebus, Hugues Bonnin (Continental Digital Services France)

Résumé :
En connectant massivement les véhicules, leurs capteurs et systèmes, à une plateforme de services, le projet eHorizon de Continental permet l'émergence d'applications automobiles innovantes embarquées en matière de sécurité, d'efficacité énergétique et de confort. Le développement de tels services et applications entraine l'utilisation de technologies et de pratiques telles que l'informatique nuagique, les approches centrées sur les données, l'intelligence artificielle, les méthodes agiles, ... Celles-ci sortent du cadre des pratiques de développement pour les systèmes embarqués critiques et leur utilisation, pour des services automobiles, implique des enjeux que nous introduirons dans cette présentation.

Session du groupe de travail IE

Ingénierie des Exigences

Anatomie des exigences logicielles

Jean-Michel Bruel Sophie Ebersold Florian Galinier

IRIT, Toulouse
bruel@irit.fr, ebersold@univ-tlse2.fr, florian.galinier@irit.fr

Résumé

Un système informatique, comme n'importe quelle autre construction d'ingénierie, existe pour satisfaire certains objectifs, aussi appelés ses exigences. Le génie logiciel a produit de nombreuses évidences sur le fait que les systèmes dépendent principalement de la qualité de ces exigences. Il a aussi permis de réaliser que les exigences étaient elles-mêmes des éléments logiciels de premier plan, au même titre que les tests ou les méthodologies. Et à ce titre les exigences méritent les mêmes bonnes pratiques, approches rigoureuses et études théoriques. Cette présentation vise à introduire des efforts que nous menons dans cette direction, sous la forme d'un ensemble de catégories d'exigences ainsi qu'un ensemble de relations entre exigences, qui visent à servir de cadre théorique et unifié permettant de mettre en oeuvre de telles études.

Semios Suite: a software to improve requirements quality

Manel Mezghanni

Semios, Toulouse
m.mezghanni@semiosapp.com

Résumé

Semios Suite[1] includes *Semios for Requirements* which is a software for detecting errors in specifications from the conception phase. It aims to control specifications quality and reduce management cost. The core of the semantic engine of this tool is based on NLP techniques and works directly with requirements engineering domains tools like IBM DOORS, IBM Doors Next Generation, MS Word and MS Excel. Semios Suite includes also *Semios For Similarity*, a tool based on AI technology to identify potential defects in terms of redundancy or inconsistency in specifications. It includes also *Semios For Acronyms*, a tool that aims to identify acronyms without definitions, acronyms with a definition and acronyms with several definitions.

[1] https://www.semiosapp.com/

FORMOD : un outil d'ingénierie des exigences basé sur la fédération de modèles

Régine Laleau

LACL, Université Paris-Est Créteil,
laleau@u-pec.fr

Résumé

En ingénierie des exigences (IE) pour les systèmes critiques, au moins deux types de modèles sont nécessaires. D'une part les exigences doivent être exprimées dans un langage qui puisse être compris par les différentes parties-prenantes qui participent à la construction du modèle d'exigences dans un but de validation. Dans la plupart des méthodes d'IE ce langage est graphique. D'autre part, il faut un langage formel pour vérifier les exigences critiques. Le projet ANR FORMOSE[1] a pour objectif de construire une méthode formelle d'IE orientée modèles pour des systèmes critiques. Cette méthode, appelée SysML/KAOS, permet de décrire graphiquement les exigences à différents niveaux d'abstraction. A partir d'un modèle de buts, une spécification formelle Event-B est alors dérivée. Pour maintenir la cohérence entre les deux modèles graphique et formel, nous avons utilisé une technique de fédération de modèles qui consiste à définir un méta-modèle pour chacun des langages puis un modèle d'alignement qui permet de spécifier des liens dynamiques entre les deux meta-modèles. L'approche est supportée par l'outil FORMOD, lui-même implémentée sur la plateforme Openflexo.

[1] http://formose.lacl.fr/index.html

Session commune aux groupes de travail Compilation et LaHMA

Compilation — Langages et Modèles de Haut-niveau pour la programmation parallèle, distribuée, de grilles de calcul et Applications

A model-based certification approach for multi/many-core embedded systems

Pierre Bieber, Frédéric Boniol, Youcef Bouchebaba, Julien Brunel
Claire Pagetti, Olivier Poitou, Thomas Polacsek, Luca Santinelli
Nathanaël Sensfelder

ONERA-Toulouse, France

Abstract

This article presents the first year results of the PHY-LOG project that aims at providing a model-based certification framework for aeronautics systems designers when developing multi/many-core-based architectures. After a brief reminder of the certification objectives, we present an overview of the PHYLOG approach. We then detail two points of the methodology: (1) a certification oriented meta-model for multi-and many-core platforms, and (2) an interference prediction method based on this meta-model. This method is then illustrated on a sub-part of the many-core Kalray MPPA R-256.

Primitives de tenseur pour les réseaux de neurones

Thibaut TACHON

LIFO Université Orléans - Huawei

Résumé

Nous présentons HTL (High-level tensor language), un nouveau langage spécifique au domaine (DSL) pour la programmation d'opération parallèles sur des tenseurs. Il permet un développement sûr et productif des réseaux de neurones avec la génération de code vers diverse plateformes matérielles ainsi que l'entrainement des réseaux de neurones basé sur la même sémantique du langage. Les avantages d'HTL sur les DSL existants pour les réseaux de neurones sont son petit ensemble de primitives parallèles, son design minimal du langage source, sa sémantique purement fonctionnelle et son analyse statique permettant la génération de code parallèle efficace.

Compiling for notifying memories: issues and challenges

Kevin J.M. Martin

Univ. Bretagne-Sud, CNRS UMR 6285, Lab-STICC, Lorient

Abstract

More than 62% of the entire measured system energy is spent on moving data between memory and the computation units. The memory hierarchy as an enabler to rise up against the memory wall, goes along with useless and (energy) wasteful memory accesses. This is specifically true when memory is used as synchronization means between processes through polling. The notifying memories concept suggests to monitor the content the memory and to notify the concerned processor when needed [1]. It stands as an alternative between polling and interrupt. This paper presents a case study for data-flow applications, and highlights the issues and challenges for a compilation tool to make use of this new memory organization.

References

[1] K. J. M. Martin, M. Rizk, M. J. Sepulveda, and J.-P. Diguet. Notifying memories: A case-study on data-flow applications with NoC interfaces implementation. In *53rd DAC Conf.*, New York, NY, USA, 2016. ACM.

Session du groupe de travail IDM

Ingénierie Dirigée par les Modèles

Unified LTL Verification and Embedded Execution of UML Models

Valentin Besnard[1], Matthias Brun[1], Frédéric Jouault[1], Ciprian Teodorov[2], and Philippe Dhaussy[2]

[1]ERIS, ESEO-TECH, Angers, France
prenom.nom@eseo.fr
[2]Lab-STICC UMR CNRS 6285, ENSTA Bretagne, Brest, France
prenom.nom@ensta-bretagne.fr

https://hal-univ-tlse3.archives-ouvertes.fr/ESEO-TECH/hal-01887948v1

Abstract

The increasing complexity of embedded systems leads to uncertain behaviors, security flaws, and design mistakes. With model-based engineering, early diagnosis of such issues is made possible by verification tools working on design models. However, three severe drawbacks remain to be fixed. First, transforming design models into executable code creates a semantic gap between models and code. Furthermore, for formal verification, a second transformation (towards a formal language) is generally required, which complicates the diagnosis process. Finally, an equivalence relation between verified formal models and deployed code should be built, proven, and maintained. To tackle these issues, we introduce a UML interpreter that fulfills multiple purposes: simulation, formal verification, and execution on both desktop computer and bare-metal embedded target. Using a single interpreter for all these activities ensures operational semantics consistency. We illustrate our approach on a level crossing example, showing verification of LTL properties on a desktop computer, as well as execution on a stm32 embedded target.

***Keywords*—** UML Execution, Model Interpretation, LTL Model-Checking, Embedded Systems

Efficient use of local energy: An activity oriented modeling to guide Demand Side Management

Alexandre Rio, Yoann Maurel, Olivier Barais, Yoran Bugni

Efficient use of local energy: An activity oriented modeling to guide Demand Side Management

Alexandre Rio
Groupe OKWind, Univ Rennes 1,
Inria, Irisa
Vitré, France
alexandre.rio@okwind.fr

Yoann Maurel, Olivier Barais
Univ Rennes 1, Inria, Irisa
Rennes, France
first.lastname@irisa.fr

Yoran Bugni
Groupe OKWind
Vitré, France
yoran.bugni@okwind.fr

ABSTRACT

Self-consumption of renewable energies is defined as electricity that is produced from renewable energy sources, not injected to the distribution or transmission grid or instantaneously withdrawn from the grid and consumed by the owner of the power production unit or by associates directly contracted to the producer. Designing solutions in favor of self-consumption for small industries or city districts is challenging. It consists in designing an energy production system made of solar panels, wind turbines, batteries that fit the annual weather prediction and the industrial or human activity. In this context, this paper reports the context of this business domain, its challenges, and the application of modeling that leads to a solution. Through this article, we highlight the essentials of a domain specific modeling language designed to let domain experts run their own simulations, we compare with existing practices that exist in such a company and we discuss the benefits and the limits of the use of modeling in such context.

KEYWORDS

Simulation, What-if Scenario Analysis, Case study

1 INTRODUCTION

An interesting aspect of renewable energies is that they can be produced locally, close to the consumers, thus considerably reducing infrastructures and distribution costs [17]. These developments have led to significant gains that make credible the use of these energies on a daily basis as the main source of energy for a large number of industrial and agricultural activities. The **autonomy** of sites with micro-generation capabilities is then greatly increased by self-consumption of locally produced energy. **Self-consumption** of renewable energies is defined as electricity that is produced from renewable energy sources, not injected to the distribution or transmission grid or instantaneously withdrawn from the grid and consumed by the owner of the power production unit or by associates directly contracted to the producer. Self-consumption is seen as a solution to improve the use of renewable energy and decreases energy-network costs. Logically, the higher the self-consumption, the more the investments in solar panels or wind turbines will be worth the expense — feeding production surplus into the grid presupposes suitable collection infrastructures and is not always possible nor desirable.

One striking challenge to self-consumption of renewable energies is the disparities between power generation from solar panels or wind turbines and the actual demand. Efficient use of renewable energy requires the adaptation of consumption practices. These practices and the current organization of activities have been indeed strongly influenced by the traditional energy production model. In city districts, for instance, most of the power production takes place when residents are not at home, pursuing their profession or other daily life activities[17]. Of course, this situation and the possible adaptations of activities are very different depending on whether one considers residential, industrial, services or agricultural areas. This paper focuses on the two latter.

Commercial enterprises could, due to the alignment of working hours with power production from solar panels or wind turbines, achieve higher rates of self-consumption depending on the type of enterprise. **Demand Side Management (DSM)**[19] refers to a set of measures to optimize the use of energy including Demand Response (DR) and storage strategies. DR represents the practice of managing electricity demand in a way that peak energy use is shifted to off-peak periods enabling higher rates of self-consumption. With electricity storage and DR, rates of self consumption can be raised.

On of the keys is thus to **align production and consumption** either by planning processes differently or by relying on storage capabilities. Autonomy and self-consumption are intertwined: achieving high levels of autonomy with renewable energies has a negative impact on self-consumption if no storage is available. Designing a system that enables almost 80% or even 100% self-consumption and a high autonomy of 50% to 70% is, therefore, a complex problem when competitiveness in electricity costs prevails. It requires a) to dimension the nominal power to be produced according to the weather and the characteristics of production facilities (e.g, solar panels or wind turbines), b) to assess the maximum volume of energy to store to absorb peaks, c) to infer consumption of activities in the business processes, and d) to identify the ones that can (and are worth to) be shifted. An expert designing such energy management system will likely be interested in how to size energy production facilities and storage. What trade-off between autonomy and self-consumption have to be done to ensure an acceptable price level? Which regions are best suited for the implementation of such a solution? And, which process reorganizations allow the optimal use of the energy produced?

Building a model of some real-world phenomenon seems to be an intrinsic human endeavor to understand and predict occurrences in the world [4]. If lots of approaches discuss the benefits of Model-Driven Engineering (MDE) approach to help experts in designing simulators, *this paper proposes an industrial case study for representing industrial process and energy management constraints.*

This experience report discusses the benefits and current limitations of MDE to build such a simulator. We go in-depth in the

domain specific language that has been designed to provide abstractions to an expert to define its installation and its business process. We also compare this approach regarding the practices of a local company that design such system for medium size industry. We validate our approach in showing the design of such a DSL and discussing how we improve the accuracy of the simulation compared to the state of the practices in this company. Next, we discuss some findings regarding MDE technologies.

The rest of this paper is organized as follows. Section 2 presents the concrete case study and provides a journey in a renewable energy engineer's day. Section 3 illustrates our approach and introduces the domain specific language and the associated simulator that has been designed to assist a renewable energy engineer. Section 4 provides the evaluation of this work. Section 5 discusses related work. Section 6 discusses some findings regarding MDE technologies and highlights future work.

2 CASE STUDY

This work is carried out as part of a collaboration between OKWind[1] and researchers in computer science. OKWind's focus is to bring energy production and consumption closer together without creating new infrastructures. The company is specialized in the production of renewable sources of energy. It proposes to deploy self-production units directly where the consumption is done. It has developed expertise in vertical-axis wind turbines, photo-voltaic trackers, and heat pump. For example, the company produces several models of double-sided solar trackers ranging from 75 to 117 square meters capable of producing 25 to 40MWh/year or 17 to 23kW in peak conditions.

The company mainly targets businesses and has accumulated a great deal of expertise in the instrumentation of agricultural and industrial sites, as well as service companies. It has a network of several hundred customers which allows it to study in detail the uses of its products. Experiments are currently underway at a number of sites to study how the reorganization of activities can improve self-consumption. This is conducted in synergy with the site managers. OKWind's long-term objective is to be able to achieve an autonomy of around 70% by offering competitive energy rates compared to traditional operators. To achieve this goal, the company intends to design a system that integrates its business experience and as well as site operators to tackle the following questions.

(1) **How to size local renewable energy production units to meet a site's energy consumption?** This study breaks down into the review of several factors for a given site: What is the desired level of autonomy? Are the weather conditions more favorable to wind or solar? Would it make sense to use a combination of multiple energy sources over long periods of time? Does consumption occur when energy is produced or is it necessary to install batteries to store surplus production? Are there any specific tariff policies to integrate, for example, feed-in tariffs? In particular, energy cost constraints must be taken into account: self-consumption must be maximized if return on investment is the main priority.

(2) **Which organization of activities enables the best autonomy and self-consumption?** Shifting processes, however, is conditioned by many business constraints: some activities cannot be performed at night, are intertwined, or have a significant impact on employees or animals' health. The determination of the best organizations is generally done through a complex iterative task which requires a good knowledge and modeling of the processes. The idea is to build a system that provides sufficient guidance to operators to allow them to gradually optimize their practice.

(3) **Which region would be the most interesting for the expansion of its business?** On the basis of meteorological data, business knowledge and the results of previous projects, one may want to estimate whether an equivalent site in a different region could benefit from renewable energy production. If a simulation tool is available to estimate a site on the basis of meteorological knowledge and description of activities, it is then possible to estimate the ecological viability of such a project.

These questions, and the variability of the factors to be taken into account, naturally led us to use a model-driven approach to conduct (What-If Scenario Analysis)[2].

3 MODEL

In this section we present the model on which our Energy Management System (EMS) is based. The goal is to provide a Domain Specific Language (DSL) that enables experts to quickly integrate their knowledge and algorithms, and to provide a library of reusable components and algorithms. This library can be improved by extending existing components or adding new ones. Flexibility and extensibility is mandatory as the number of machines, producers, and algorithms is important. Some of these components will also allow to play back historical data, which is a common use for sizing purposes. Using a DSL and components that clearly separate the different concerns avoids code redundancies and facilitates the work of domain experts [13].

The first intent of our DSL is to let an expert model a site. The sites considered are always structured around five components: the **grid** that provides traditional energy, a set of local **production means** (generally renewable), **storage/batteries** to meet needs in periods of low production and finally a set of **processes** that use

[1] http://www.okwind.fr/

[2] What-if scenario analysis (WISA) is a business planning and modeling technique used to yield various projections for some outcome based on selectively changing inputs.

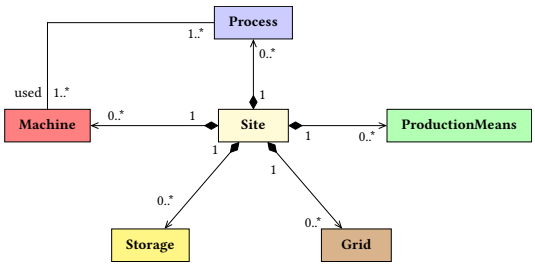


Figure 1: Simplified version of our metamodel

machines. As we are mainly interested in industrial/agricultural sites we use Machine rather than Appliance – a term that often comes up in literature

Figure 1 shows our simplified model capturing these aspects. Some information, such as the name of the modeled site or physical location (GPS coordinates) are common to all objects, hence we have a central class named `Site` to hold every aspect of our regulation system.

We will see that we dedicate an important part to the modeling of processes and the use of machines. In general many models only consider fine-grained consumers such as machines. This makes recommendations difficult because a machine can be used by different activities at different times of the day. Our aim here is to be able to link consumption to business activities in order to guide users in planning their activities.

The storages and the grid have a particular role here: they can both absorb surpluses of local energy production, and meet needs when local energy production is not sufficient. Their presence is optional, but to be able to operate 24 hours a site must generally have one of the two. It could be noted here that we have deliberately chosen to represent them separately in this first version of the model, but it would be quite possible to consider them as a special case of *ProductionsMeans* and *Machine*. This choice, which in practice has a measured impact on implementation, comes partly from the fact that we wanted to be able to use these parts independently.

The rest of this section describes in depth the five main parts of our language introduced in Figure 1. It ends in showing how we handle flexibility in activity modeling.

3.1 Production

Our work is mainly motivated and constrained by production aspects. As a result of the intermittent nature of renewable energy production and its high dependence on external (mainly meteorological) conditions, sites need to be constantly adapted. For that reason, we need to define a GPS coordinate at site level so that each module can access it.

Figure 2 shows a significant portion of the concepts we use for production modeling. *ProductionsMeans* indirectly extends the *ISimulatorPlugin* interface that gives the possibility to start and discover the different components of our project. This is the case for most of the components of our model (*Machine*, *Grid*, *Storage*). It is mainly a utility interface.

ProductionMeans also extends the *IPricedEnergy* interface. This allows assigning a price per kWh to the energy produced locally. This part is important since it is related to the return on investment. There are several ways to infer this cost by taking into account the lifetime of the equipment and its cost of use. A rather rudimentary way is to use a fixed cost estimated on the basis of accumulated experience. That is what we are doing at the moment, but it is possible to change this behavior and implement more advanced techniques and make the price evolve dynamically if necessary.

ProductionMeans is then specialized by implementations such as turbines or solar trackers. In the figure, we have chosen to present solar trackers in more detail. Solar production is affected by multiple factors:

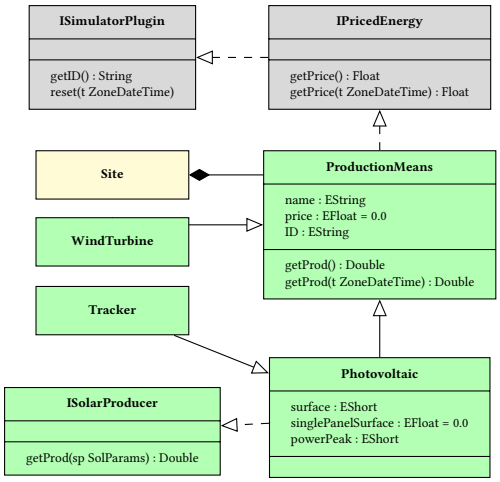


Figure 2: Production related meta classes

- Direct Normal Irradiation (DNI): beam directed directly to the panel surface,
- Reflected Normal Irradiation (RNI): beam reflected by the ground reaching the panel surface,
- Diffuse Normal Irradiation (DfNI): beam reflected on the atmosphere reaching the panel surface,
- Global Normal Irradiation (GNI): the sum of the above three type of beams.

Multiple factors can impact the power production, such as the ground albedo impacting the RNI. All these factors make the power production hard to compute. Commercial services provide solar irradiation data accessible from a web service using. To get reliable results the web service is expecting: a GPS coordinate, information about the type of solar cells, the ground albedo etc. We have written components to interact seamlessly with them.

Considering a GNI value from a third-party service we can apply formulas such as: $Tracker_P = GNI \times \frac{Panel_{pp}}{Panel_S \times 1000} \times 60 \times granularity \times Tracker_S$

- Where GNI is expressed in Wh/m^2
- $Tracker_P$ power of the solar tracker expressed in Watt.
- $Panel_{pp}$ the power peak of a single panel
- $Panel_S$ surface of a single panel
- $Tracker_S$ total surface of the solar tracker

This formula is typical of the one we must regularly integrate into our components. Once the component is implemented, the expert can configure an instance in the DSL with the required parameters or in some cases modifies the formula. We do not expect any advanced programming skills from a domain expert.

By extending the *Photovoltaic* class that already provides a production estimate for a fixed panel, it is possible to implement more complex panels such as the two-axis trackers we mainly use. This allows us to refine the forecasting algorithms that require grasping the specifics of these trackers. Thus, depending on the sites, it will be possible to estimate the production of different types of solar panels and to make an informed choice.

3.2 Grid

Although optional, the grid is often essential for the consumption of most sites. Indeed, often to reach a reasonable level of autonomy at times when local production is low, it would require batteries of large capacities therefore expensive. It is sometimes more profitable in these cases to rely on the energy supplied by the grid. The integration of the grid into a model is interesting because pricing policies are varied. It is therefore, necessary to provide a flexible model that can be adapted to the specificities of each site and region.

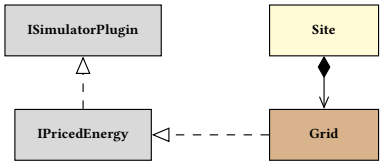


Figure 3: Grid related meta classes

Figure 3 shows the main classes for the grid. *IPricedEnergy* and *ISumulatorPlugin* are extended for the same reasons as for production. We often want to consider the maximum power intensity we can draw from the grid as well as the price per kWh for a given time. This price is fixed by the customer's contract and may vary considerably depending on the time of day and the season. For example, a two-tiered tariff can easily be implemented by adjusting the result depending on the parameter t of the method *getPrice*. This allows to model a basic residential energy contract with expensive peak hours between 8 a.m and 7 p.m. On this basis, we could approximate energy costs on the basis of a consumption history and evaluate the benefits of shifting these consumptions to times when energy is affordable.

3.3 Energy storage

Storage is an alternative and a complement to the grid when it comes to meeting demands outside local production periods. It is an easy way to perform peak-shaving and valley-filling techniques. The energy excess is stored in the battery during peak hours and then discharged when the local production is insufficient. Figure 4 shows the implementation of a class allowing the simulation of different types of batteries. The expert can configure this component or extend it to refine predictive behaviors if necessary.We model specifications such as the capacity in Wh and various level of usage such as depth of discharge and a safety cap, both expressed in percent. Meaning that we don't want the battery to be under a certain level of energy to protect its integrity A performance level is used to represent the loss of energy due to various effects such as Joule effect or transformation (i.e. AC to DC then DC to AC).

Besides the charge level, a common way of maximizing the lifetime of a battery is to limit the number of charge cycle performed daily. The behavior can be specified in the implementation or use default ones based on the *battery_type* attribute, 1 cycle per day for lithium batteries and no limits for fuel cell batteries. Generally, the cycle limit is based on the wear attributes impacting the battery capacity. The maximum power allowed during charge and discharge of the battery can either be defined by its type, for example, a 1 C

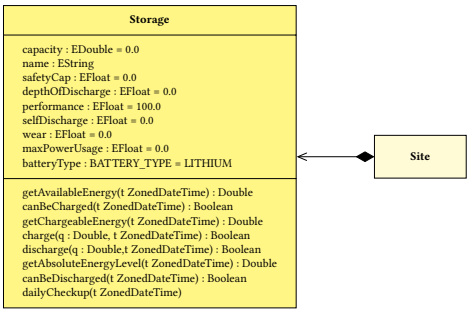


Figure 4: Storage meta class

rate[3] cycle for a lithium battery or defined by an attribute expressed in W.

3.4 Machines

Machines are the counterpart of producers. A balance must be maintained between their consumption and production, whether the latter is local or drawn from the grid or storages. In practice, it uses common concepts : as for *Grid* and *ProductionMeans*, the *Machine* class extend *ISimulatorPlugin* and *IPricedEnergy* classes. The relationships between machines and processes are shown in Figures 6. *Simplemachine* consists in a nominal power representing the power needed to this machine whenever it is used.

Our system provides multiple ways of modeling devices: either by using historical data, physical or statistical model or by introducing new simple devices by providing only the nominal power. Moreover, complex machine can be seen as a composition of much simpler machines. The system supports the implementation of composite machine by composing different simple activities as described in the following section. These components can then be shared, extended, and reused.

Extending the model to implement complex behaviors is straightforward. Let's take the example of an expert that wants to introduce stochastic behavior to bring dynamicity in the simulation. The idea is to simulate a device whose power oscillates from -5% to +5% around its nominal power. Listing 1 shows an implementation.

```
public class LightBulb extends MachineImpl {
  private final int NOMINAL_POWER = 50;
  public LightBulb() {this.setID("plugin.MySpecificLightBulb");}
  @Override
  public float getConsumption(ZonedDateTime t) {
    return NOMINAL_POWER + NOMINAL_POWER* ThreadLocalRandom.current
        ().nextInt(-5, 6);
  }
}
```

Listing 1: Extension of an machine through a Java plugin

The expert can simply define its behavior in plain old Java by extending the default *MachineImpl* abstract class. During the simulation, the parameter of time is provided as a parameter. Using a General Purpose Language such as Java offers a lot of freedom in the configuration of the behavior: using complex math library as well as network library to query external services. A machine has a unique name that is used for discovery and instantiation in the

[3]For a battery of 10Ah, a 1 C rate is equivalent at charging and discharging during an hour at 10A.

Domain Specific Language (see Listing 2). The DSL allows experts with little programming skills to use the components easily.

```
machine light_bulb as "plugin.MySpecificLightBulb"
```

Listing 2: External plugin used as internally defined machines.

Most industrial machines use three-phase electric power, this means that to properly monitor the consumption of such machine three sensors are needed and three data log files are produced. A machine can, therefore, reference other machines as depicted in listing 3.

```
device {
  machine A_phase1 as "plugin.timedcsvconsumer"
  machine A_phase2 as "plugin.timedcsvconsumer"
  machine A_phase3 as "plugin.timedcsvconsumer"
  machine A {composed of (A_phase1,A_phase2, A_phase3)}
}
```

Listing 3: Three-phase electric power example

3.5 Activity modeling

As we have already stressed, the modeling of business processes is a key to provide effective recommendations to users. Modeling machines alone is often not enough because they are frequently used for multiple activities. The idea is to induce behaviors that promote self-consumption by aligning energy demanding activities and local production periods. If this idea is difficult to implement for residential activities which are relatively unforeseeable as Bourgeois[3] pointed out, it applies well to the industrial and agricultural world where the processes are known, industrialized, reproducible and planned several months in advance. Furthermore, consumption scales and potential gains are much higher than in the residential sector, encouraging stakeholders to participate proactively. Finally, it is easier to reuse the experience gained on a type of business to simulate the operation of similar ones.

Figure 6 details the model developed to represent business processes and their relationship with energy-using machines. `Process` is the main class and holds both scheduling description and flexibility. Flexibility is captured through `Reduction` and `Shift` classes and is discussed further in Section 3.6. Based on this model, it is possible for our EMS to apply scheduling algorithms to minimize locally produced energy excess and grid usage so as to raise the site autonomy. Such system is very close to a Cyber-Physical System that integrates embedded sensors to monitor various metrics such as the power consumption of physical devices. A study by [11] showed that classic scheduling algorithms in operating systems can be reused in our case.

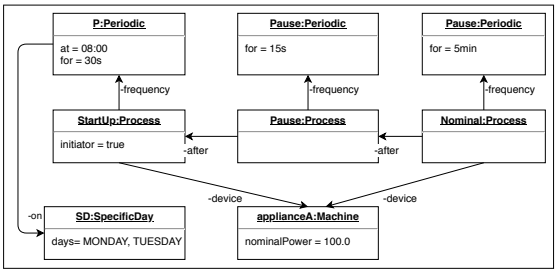


Figure 5: machine behavior modeled with Process

A process is linked to none, one or multiples `Machines` (see Section 3.4): we consider a machine to be active if one of its activities is active and logically it is not possible for two processes to use the same machine at the same time. Process planning is very classic in its representation. We made it similar to the one expressed in the company's or farm's task schedule: a list of tasks with a start date time and a duration or an end time. This allows to easily connect company calendars and the information reported by our system: we can, for example, evaluate the energy consumption of meetings or rooms such as conference rooms by knowing the activities that take place there.

As stated before industrial processes are predictable mostly due to the constant frequency of some activities, to that extent we propose to link a `Frequency` to an activity. A frequency can specify a particular recurrence of events: *every day* or *every Monday at 08:00 for 4 hours* for instance. For more complex activities such as a sanitary emptying or holidays, activities are incorporated into periods of a given duration called `Window`. For fine grain control, one can also link activities with pauses between them as shown in Figure 5. With these concepts, it is possible to model an activity running every day during day time using a specific machine *A*.

It is also possible to model various mode of a machine. For instance, we could define an activity *StartUp* using a device *machineA* working at 100W for 30 s, pausing for 15 s then working again for 5min, on Monday on Tuesday at 8 am. This example is instantiated as an object diagram in Figure5 using `Process` linked as dependencies, which means that a process must first be completed before a new process can start.

3.6 Modeling Flexibility

The language is thus expressive enough to represent how the activities follow each other and deduce their consumption by observing the consumption of the machines. This can help replay complex scenarios involving power consumption and help to size of production means. However, modeling the sequence of activities in time is not sufficient to take measures: it is also necessary to be able to represent the level of flexibility offered to reduce their consumption or shift them. Optimizing self-consumption by modifying processes can be done mainly in two ways: shifting activities to times when energy is produced and less expensive or reducing the intensity of activities where possible – it may be possible to reduce the light intensity of lighting in certain rooms for instance.

The adaptation of activities cannot be done suddenly without a good understanding of the users, their process and their constraint. This requires the cooperation of the users. It is necessary to understand how the activities are carried out at the present time and to understand the elements of variability: is it possible to reduce the intensity of a machine? what is the impact on comfort and the acceptable level of comfort? To what extent is it possible to move some activities and over what time range? Starting from activity modeling, see Section 3.5, we provide a way of capturing the set of possible actions to take.

For instance, ventilation systems or cleaning devices can operate at a lower nominal power but need more time to be as efficient.

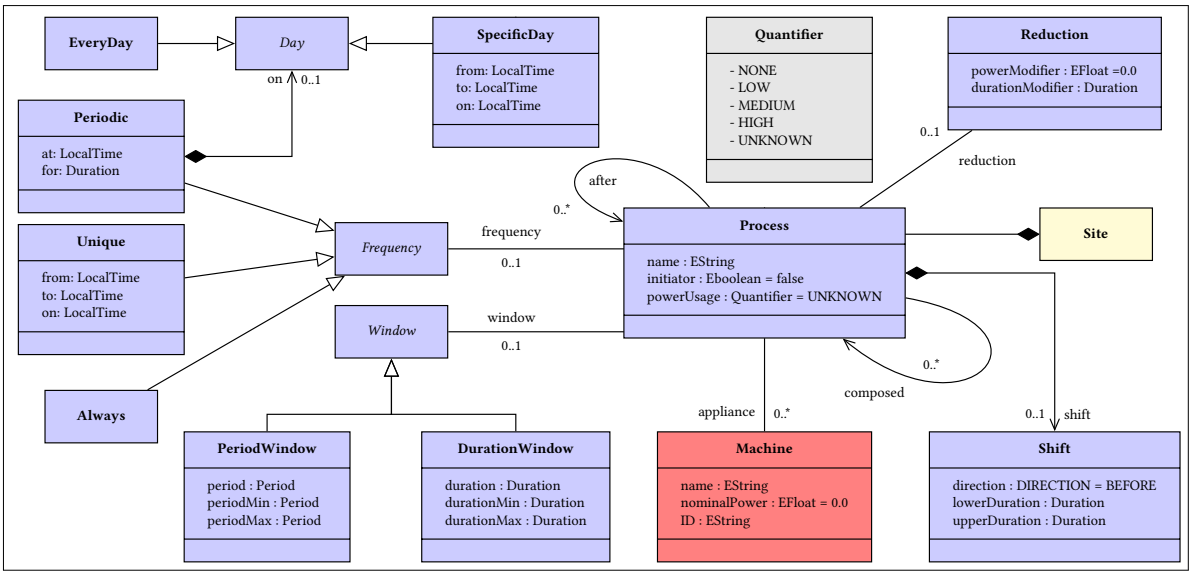


Figure 6: Activities meta classes

An expert could know how to accurately measure the impact of a power reduction and use this information to model more precisely its activities.

Example 3.1. An activity could last 2 hours at nominal power but with a 20 percent power reduction last it would 30 more minutes. This information could then be used by a regulation system to perform recommendation in order to improve the global industrial site autonomy.

Another variability aspect we want to address is the flexibility in schedules. Unlike factories, primary sector industries such as animal husbandries are more flexible and most constraints come from the animal well being.

We noticed multiple times employees arriving at work early in the morning, turning on machines and either doing something else first that requires less energy consuming, that could have been done later on. This highlights the need of an energy-aware monitor, able to know the flexibility of every, or at least most consuming, activities of an industry.

What we want is to distinguish activities that have to be done at an exact moment from those more flexible.

Example 3.2. The cleaning of the room can be done at anytime in the morning.

Example 3.2 can be modeled as a Process running at 10:00 in the morning with a possible Shift of 2 hours both before and after. This means that the default start time of this task is 10 am but if it can be beneficial for the global energy management of the site we allow a time shift of two hours. The earliest start time becomes 8 am and the latest start time becomes 12 am. This only gives information to know how much an activity can be moved, not if it is beneficial or profitable to do it.

In our approach this set of information is modeled by the *Reduction* and *Shift* classes. Our EMS is thus able to make the right recommendations.

4 IMPLEMENTATION AND VALIDATION

We first implemented the metamodel presented in Section 3 using the Eclipse Modeling Framework (EMF). One of its benefits is the graphical view on the metamodel allowing to include collaborators and experts in the loop and show them how their field of expertise has been modeled along with the other aspects of the energy regulation.

We use this model to generate an Xtext grammar to model equipments installed in a given site. This corresponding DSL is used in our simulator to play various scenarios or replay logged data. This model-first basic Xtext grammar has then been modified to better suit our needs. Especially:

- every value has its unit specified right after, this avoids confusion, i.e batteries capacity is often expressed in Ah but also in kWh,
- some key concepts have been renamed to fit expert habits and vocabulary.

Figure 7 illustrates the architecture of our simulator running in standalone based on a site description written in our DSL. Experts can write DSL files with the help of an eclipse editor plugin. Our simulator discretises time on the smallest time step possible, usually a 1 minute time step. Based on that time step the simulator will ask each energy producer about its current production, each energy consumer about its current consumption.

Every simulation performed stores the step by step results in a time-series oriented database named InfluxDB[4]. The simulator also

[4]https://www.influxdata.com/

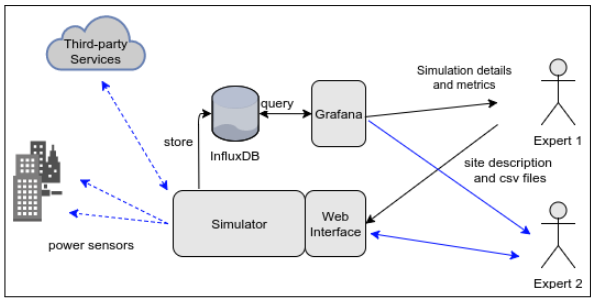


Figure 7: Simulator architecture and use cases

creates a Grafana[5] dashboard per simulation configured to easily navigate through the stored data. These dashboards allow us to easily compare simulations and input parameters and their impact on output metrics: global autonomy, reduction of grid usage etc. Figure 7 depicts the simulator architecture and its various components. In this figure, expert 1 does not know the DSL. He provides through a web interface a form describing an industrial site and CSV files for the consumption and production. The simulator runs it, stores the results in the database and returns metrics and a link pointing to the Grafana dashboard illustrating these results. Expert 2 knows the DSL and writes her site description directly in the DSL, references physical power sensors remotely accessible. She can have process organization recommendations on the web interface and energy monitoring information on the Grafana interface.

Our simulator can be used to answer various questions an expert can have.

4.1 Q1, Sizing production and storage

Our first type of question is **what type of energy producers and batteries should be installed to meet a site's need**. For this question, we assume we have logged a significant amount of data to represent our consumption profile. We will generate variants of site descriptions to simulation various "what if" scenarios. Variants can be generated either manually or using a form on the web interface of the simulator. Our base will be to use the global consumption logged as a CSV file over the year of 2017. Since we just want to properly size our production and storage means we only need the global consumption of the site. To have a base of production we will use production data logged on a geographically close site, production with no data are demonstrated in Section 4.3.

[5]https://grafana.com/

```
Site demo_site {
from 2017-01-01 to 2018-01-01
device {
Appliance globcons_sensor as "plugin.timedcsvconsumer"
}
activities {
Process wholeBusiness {
device (globcons_sensor)
frequency Always
}
}
production { Producer prod_case1 as "plugin.timedcsvproducer" }
grid {Grid edf as "edf.bleu-tempo"}
}
```

Listing 4: Description of our site

```
Battery battery_pack {
  capacity 30000.0 Wh
  safetyCap 10.0 %
  depthOfDischarge 10.0 %
  wear 2.0 %/year
  performance 90.0 %
  type LITHIUM
}
```

Listing 5: Example of lithium battery modeling

A base of simulation can be expressed in our DSL as shown in Listing 4. This simulation describes one energy consumer and one producer, both using an external plugin to use a CSV file. This will tell the simulator to look for a file `globcons_sensor.csv` and `prod_case1.csv`

A lithium battery of 30kWh is modeled in Listing 5. For each simulation, we want to look at the total energy production and the share that was used locally, either to directly feed the local consumption or to charge the batteries.

As the cost of Lithium batteries is high, we try to minimize their capacity and maximize their lifetime. To do so Lithium batteries are modeled to only allow one cycle per day. A full cycle is going from 100% capacity to the depth of discharge then charge back to full capacity. Based on this usage we consider a wear per year slowly reducing the maximum capacity.

When using batteries the simulator will use the energy excess of the day to charge the batteries and discharge the battery to shave consumption peaks during the day and reduce the grid usage in the night. If the grid tariff depends on the time of the day, high-cost periods will be favored to use the batteries in discharge.

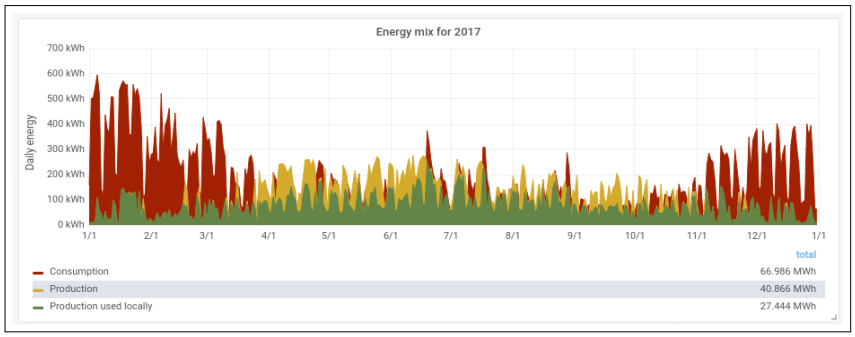
Figure 8a shows the energy mix, in particular, the consumption, in red on the graph, much more important in winter while the production is more important during the summer season. It is also in summer that we have the more energy excess even with the usage of the battery. Figure 8b shows battery usage to help experts estimate if its capacity is appropriate, considering its cost and evaluating its Return On Investment (ROI).

4.2 Q2, Activity organization recommendation

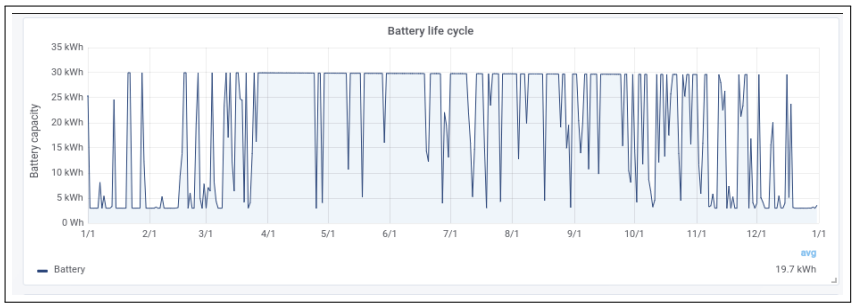
As stated before most industries have organized their process based on what they thought logic, only taking into account the grid flat price or peak and off-peak prices. We want to provide a tool to account local energy availability to best organize processes based on expert knowledge of the process and its implication on productivity, outcome or animal well being. We want to answer the question: **which organization of activities enables the best autonomy and self-consumption?**. In Listing 6, we describe three phases in

```
1  process morning {
2  device (multiperiods_m1)
3  frequency Periodic at 11:00 for 12 h on EveryDay
4  },
5  process day {
6  device (multiperiods_m1)
7  frequency Periodic at 07:00 for 1 h on EveryDay
8  shift direction AFTER lowerDuration 0 h upperDuration 5 h
9  },
10 process evening {
11 device (multiperiods_m2)
12 frequency Periodic at 19:00 for 2 h on EveryDay
13 shift direction AFTER lowerDuration 0 h upperDuration 3 h
14 }
```

Listing 6: Example of processes with flexibility



(a) Yearly energy mix of solar production, winter consumption (red) are the most important, due to ventilation and cooling. Most of the production occurs in summer, producing important energy excess (yellow). Here we can see that optimization is required either by reorganizing activities or introducing new type of producer (wind turbine)



(b) Yearly battery usage, because of the long series of energy excess the battery is under used from April to Octobre

Figure 8: Visualization of the Case 2 simulation, from January 1st 2017 to December 31st 2017

a day using two devices. Lines 8 and 13 allow the process to start from 0 to 3 or 5 hours after the initial start time, this possible shift is illustrated in Figure 9. Because of the local energy available the morning phase has been shifted to begin when energy is available while the evening one has been left untouched because they would have been no benefit of shifting it. For now, the simulator looks for shift flexibility and tries to use them to put a maximum number of processes under the production curve. Our idea is to let experts write their own scheduling algorithm to try them out and see which one is most effective for a particular industry type. We are planning on implementing Bin Packing: First Fit and Best Fit algorithms as Sendama[21] *et al.* did and bottom left decreasing height packing from Ranjan[20] *et al.* in order to benchmark which algorithm gives the best results considering a particular business sector.

Figure 9 shows that for this day shifting the morning process is beneficial for the self-consumption rate. The decision to shift or not a process is based on the context of the regulation system, for instance, local production state, storage rate etc.

4.3 Q3, Geographic comparison

Last type of question we address in this paper is a "what-for" question. **Which region is most interesting for the expansion of a business?** The underlying question is to know which region has the weather better suited for my activity. Of course sunny regions produce more solar energy but bigger production systems mean bigger inverter, cable section and in fine more indirect costs. Producing less and yet enough for a process can be more cost-effective. Some services such as PVGis[6] can help size a photovoltaic installation by providing an estimated monthly production based on more than 20 years of historical data. The pitfall is that monthly data can't help us properly size a system: we want to be able to align the consumption on the production and to know the ratio of energy directly used from production and the quantity taken from the grid. Thus, we need more accurate data, with a granularity of a point per minute or every five minutes.

Commercial third-party services provide irradiation data for a particular GPS coordinate and given some solar cell specifications. They can take into account distant relief casting shadows at a certain time of the day or period of the year. We have integrated their API to query their server for a given site GPS coordinate giving our production model as a parameter.

Listing 7 shows how we specify the GPS coordinate in our DSL and a 110m^2 two-axis solar tracker. Each panel produces 310 Watt peak. Other parameters, such as tilt and azimuth constraints are left to their defaults value by our proxy program. Data are retrieved with a granularity of 1 point per minute, which is enough to perform simulation as previously done in Question 1 and 2.

To answer our "what-for" question we need to generate variants of our DSL with a different GPS coordinate for each variant.

[6] http://photovoltaic-software.com/pvgis.php

8

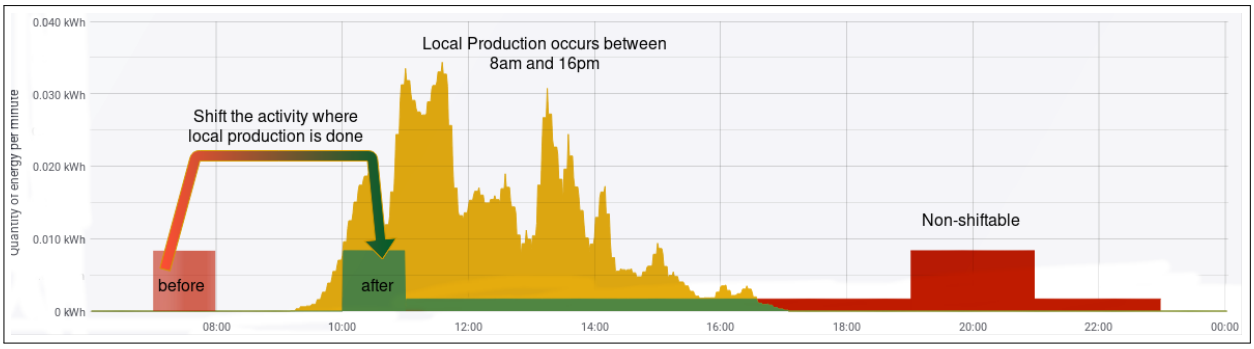


Figure 9: Question 2 : Shifting an activity to a production period when possible increase our self-consumption rate from 9% to 20% for this specific day

```
coordinates 48.8588377 N 2.2770206 E
...
Tracker t1 {
    surface 110 m^2
    powerPeak 310 Wp
    albedo 0.8
    singlePanelSurface 1.6667 m^2
}
```

Listing 7: Generic solar tracker at Paris, France coordinates

Figure 10 shows an estimation of the production in 14 variants in France. For each region, we have a fine grain simulated production an expert can then use to size his production installation or optimize his activities. In this particular case, if we want the city with the biggest production we can choose *Bourges*, *Laguiole* or *Montpellier*. We could also have estimated the best geographic installation for battery life management, based on the consecutive number of "good" or "bad" weather day.

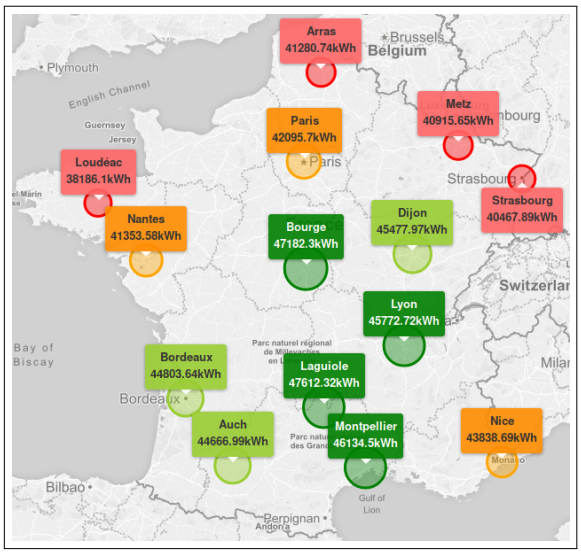


Figure 10: Simulated production taking into account relief shadows for 14 different cities in France : greenest points represent the sites where the estimated production is best.

5 RELATED WORK

Individually, every component of a smart grid has been studied thoughtfully. This goes from local production means, in particular, the most common ones such as solar photovoltaic systems and wind turbine to Energy Management Systems.

Solar energy is the most common renewable energy because of its almost constant production during the day. Furthermore, modern techniques such as Machine Learning[22] or Support Vector Machine[2][15] are becoming more and more reliable and are giving results at a shorter term than it used to. Moreover, their granularity becomes finer. Other external varying constraints are impacting regulation systems: the wind impacting local production, grid prices etc. Scheduling activities have been done under these multiple variable factors. Forecasting and learning techniques are important to take more reliable decisions. Predictions could either be focused on the energy price and site own consumption history[12] or human activity.

People model machine consumption with state machines[10] in order to optimize the various states of a single appliance to optimize its energy efficiency considering a particular task. Synchronization between machines has been modeled using petri net[8][9]. A common way of aligning consumption on production is to use an energy storage unit as a buffer such as batteries[1]. We believe that to be effective a regulation system must consider all the aspect above. Some approaches try to combine some of them but never address what we consider the whole picture.

Vivekananthan[24] et al. propose a Home Energy Management System able to control appliances depending on a variable energy price to reduce energy costs. Because of the domestic sector and its uncertainty the activity cannot be modeled. Chen[6] et al. propose a similar system. To limit this uncertainty, one idea is to address a group of apartments [7], this also has the benefits of mutualizing the investment costs. All these approaches lack the activity modeling aspect, important to better understand the consumption: how it will behave in upcoming days and how it can be adapted.

Some approaches, such as Langer[14] et al. or Onar[18] et al. require too much detailed specifications about the generation units or the consuming devices. This prevents coarse-grained modeling or prototyping and can be too complicated for a non-expert. Focusing on market energy price hides local aspect and local return on

9

investment, this kind of approach cannot be used in our context because of the local production and storage impacting the real cost of the consumed energy.

On another side, Sendama[21] et al. propose a discrete Bin Packing allocation algorithm taking into account local production and batteries. They validate it on their Home Energy Management System. Other algorithms from real-time systems exist[5], like Earliest Deadline First, and are good candidates for scheduling industrial activities. We would like to integrate some of them as modules.

Finally, graphical user interfaces such as Tiree Energy Pulse[23] tells us about the importance of including the end user in the loop and how displaying the live state of a system can provide an Eco-Feedback by reconfiguring rules defined by experts to better suit their habits and benefit the whole system. Bourgeois[3] et al. work focuses on domestic demand-shifting and human integration.

In our opinion, no real solution has a global vision of an industrial site in all its aspects to achieve efficient energy regulation with properly sized energy sources and energy storage units.

6 LESSON LEARNED AND PERSPECTIVES

In this paper, we focused on designing an integrated tool-suite to assist the engineers in dimensioning an Energy Management System (EMS) for an isolated site to reduce the construction of new network infrastructure and reduce its dependence on the grid. We advocate that the MDE is a very good candidate to integrate the various technological and business knowledge on the renewable energy production and consumption forecasting techniques, the planning of processes, energy costs, grid, and batteries. This model allows an engineer to address various concerns: sizing of production units, adapting of storage resources, choice of production technologies, choice of geographical location, or even and above all recommendations for process reorganization. The purposes of our EMS are quite different from grid-centered approaches or larger grain approaches to energy distribution. Here our long-term goal is not to complete the grid occasionally but to do without as much as possible. That is why we are focusing on renewable energies and modeling of single sites at the moment. A good understanding of a site provides huge savings levers on consumption from the grid.

As we pointed out, if there are many approaches to building such system, to the best of our knowledge, this model is one of the very few that focuses specifically on one site and not on the grid as a whole and covers so many aspects of modeling. In particular, we believe that modeling activities are essential to make good forecasts and recommendations. The model is also very extensible, which enables the integration of the various techniques produced by the literature. It is thus very easy to integrate new means of production and their forecasting techniques, new types of activities or new appliances for instance.

After highlighting the different aspects of the model and our system, we discussed how the first version of our model could answer simple simulation questions: how to size a site, or install production equipment, and which activities could be moved to maximize the use of local production means. These are the very questions that a company like OKWind, a specialist in the field, is regularly addressing. The implementation was done with very classical and well-known tools such as EMF. The system was built in a little more than 9 months, from a skills upgrade on the business problem to technologies integration and packaging for end users. It results in the end 4500 loc of Java, 1300 loc of Lua and 78kloc of generated files. The great interest of our tool is that it enables to simulate easily a very wide range of situations and thus allows to determine quickly the best options. If we compare with the company's past practices, engineers mainly used homemade excel sheets and R script. Many scenarios discussed in the paper were not possible. Sharing information among experts was very difficult. Detecting errors in site modeling was challenging. Building this domain specific language and its associated simulator saves lots of time and produces more precise results compared to the traditional manual approaches used before. Within the framework of this experiment which was conducted on real historical data, we were able to show that it was easy to integrate the various algorithms used by the company and that it was easy to extend them. The results obtained are very comparable to those obtained by OKWind experts but in much shorter times. The tool itself is an improvement on those available to the company and should make it possible to assist their experts in studies on a wider scale.

The system is, therefore, promising and the modeling approach is valid. Nevertheless, the implementation of a model-driven approach has not been without challenges or lessons learned. The first surprise was the lack of native support within a framework as EMF for the notion of measurement units. It is easy to specify its own data-types but we expected to find either reusable libraries to incorporate the international system of Units as data-types within a new meta-model or a native support within EMF. In the same vein, no native support makes it possible to simplify the management of time-series data. Again, redeveloping this support is tedious, it surprised us not to find support within a commonly used modeling framework like EMF. On these two points, it could be great if the community provides mechanisms to improve the reuse when designing a new metamodel. It means an algebra to import parts of a meta-model or a grammar but also a marketplace to automatically search and import libraries of metamodel such as maven central. The last challenge we identify for the modeling community is the co-evolution of meta-model and concrete syntax. While Xtext provides an excellent approach for building DSL, the joint evolution of meta-model and concrete syntax remains complex and tedious.

We are now conducting an experiment at several sites to see how adapting activities can improve production equipment profitability. This experience over a long period should provide us with relevant feedback on what can and cannot be requested from a site operator. This should allow us to use our tool not only to simulate upstream but also to make observations and recommendations on a weekly basis. At the same time, we also want to compare similar sites to see which variables are relevant for a particular business. Our intuition is that it is possible to define, for certain specific domains or profiles, parameters that will assist the configuration of the model and refine the recommendations and simulations: the number of animals on a farm is an example. Finally, we would like to explore how this model can be used as a basis for artificial intelligence algorithms to manage real-time operations. All the information described is indeed very complementary to the type of information that a continuous management system such as those proposed by a company like Ubiant [16] would seek.

REFERENCES

[1] H. Alharbi and K. Bhattacharya. 2018. Stochastic Optimal Planning of Battery Energy Storage Systems for Isolated Microgrids. *IEEE Transactions on Sustainable Energy* 9, 1 (Jan. 2018), 211–227. https://doi.org/10.1109/TSTE.2017.2724514

[2] Kuk Yeol Bae, Han Seung Jang, and Dan Keun Sung. 2016. Hourly Solar Irradiance Prediction Based on Support Vector Machine and Its Error Analysis. *IEEE Transactions on Power Systems* (2016). toread.

[3] Jacky Bourgeois. 2016. *Interactive Demand-Shifting in the Context of Domestic Micro-Generation.* Ph.D. Dissertation. cited.

[4] Jean-Michel Bruel, Benoit Combemale, Ileana Ober, and Hélène Raynal. 2015. MDE in Practice for Computational Science. In *INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE*. Reykjavík, Iceland. https://hal.inria.fr/hal-01141393 cited.

[5] Davide Caprino, Marco L. Della Vedova, and Tullio Facchinetti. 2014. Peak shaving through real-time scheduling of household appliances. *Energy and Buildings* 75 (2014), 133–148. https://doi.org/10.1016/j.enbuild.2014.02.013 cited.

[6] Shengbo Chen, Ness B. Shroff, and Prasun Sinha. 2013. Heterogeneous Delay Tolerant Task Scheduling and Energy Management in the Smart Grid with Renewable Energy. *IEEE Journal on Selected Areas in Communications* 31, 7 (jul 2013), 1258–1267. https://doi.org/10.1109/JSAC.2013.130709 cited.

[7] Gabriele Comodi, Andrea Giantomassi, Marco Severini, Stefano Squartini, Francesco Ferracuti, Alessandro Fonti, Davide Nardi Cesarini, Matteo Morodo, and Fabio Polonara. 2015. Multi-apartment residential microgrid with electrical and thermal storage devices: Experimental analysis and simulation of energy management strategies. *Applied Energy* 137 (2015), 854–866. https://doi.org/10.1016/j.apenergy.2014.07.068 cited.

[8] Anton Dietmair and Alexander Verl. 2008. Energy consumption modeling and optimization for production machines. *IEEE International Conference on Sustainable Energy Technologies* (2008), 574–579. https://doi.org/10.1109/ICSET.2008.4747073 cited.

[9] Anton Dietmair and Alexander Verl. 2009. A generic energy consumption model for decision making and energy efficiency optimisation in manufacturing. *International Journal of Sustainable Engineering* 2, 2 (jun 2009), 123–133. https://doi.org/10.1080/19397030902947041 cited.

[10] Anton Dietmair, Alexander Verl, and Philipp Eberspaecher. 2011. Model-based energy consumption optimisation in manufacturing system and machine control. *International Journal of Manufacturing Research* (2011). http://www.inderscienceonline.com/doi/abs/10.1504/IJMR.2011.043238 cited.

[11] Tullio Facchinetti and Marco L. Della Vedova. 2011. Real-Time Modeling for Direct Load Control in Cyber-Physical Power Systems. *IEEE Transactions on Industrial Informatics* 7, 4 (nov 2011), 689–698. https://doi.org/10.1109/TII.2011.2166787 cited.

[12] Vito Fusco, Ganesh K Venayagamoorthy, Stefano Squartini, and Francesco Piazza. 2016. Smart AMI based demand-response management in a micro-grid environment. In *2016 Clemson University Power Systems Conference (PSC)*. IEEE, 1–8. cited.

[13] Tomaž Kosar, Pablo E Martı, Pablo A Barrientos, Marjan Mernik, et al. 2008. A preliminary study on various implementation approaches of domain-specific language. *Information and software technology* 50, 5 (2008), 390–405.

[14] Tino Langer, Andreas Schlegel, Johannes Stoldt, and Matthias Putz. 2014. A model-based approach to energy-saving manufacturing control strategies. *Procedia CIRP* 15 (2014), 123–128. cited.

[15] Maria Malvoni, Maria Grazia De Giorgi, and Paolo Maria Congedo. 2016. Data on Support Vector Machines (SVM) model to forecast photovoltaic power. *Data in brief* 9 (2016), 13–16.

[16] Sébastien Mazac, Frédéric ARMETTA, and Salima Hassas. 2014. On bootstrapping sensori-motor patterns for a constructivist learning system in continuous environments. In *Alife 14 : Fourteenth International Conference on the Synthesis and Simulation of Living Systems (Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems)*, The MIT Press (Ed.). New York, United States. https://doi.org/10.7551/978-0-262-32621-6-ch028

[17] Daniel E Olivares, Ali Mehrizi-Sani, Amir H Etemadi, Claudio A Cañizares, Reza Iravani, Mehrdad Kazerani, Amir H Hajimiragha, Oriol Gomis-Bellmunt, Maryam Saeedifard, Rodrigo Palma-Behnke, et al. 2014. Trends in microgrid control. *IEEE Transactions on smart grid* 5, 4 (2014), 1905–1919.

[18] Omer C. Onar, Mehmet Uzunoglu, and Mohammad S. Alam. 2008. Modeling, control and simulation of an autonomous wind turbine/photovoltaic/fuel cell/ultra-capacitor hybrid power system. *Journal of Power Sources* 185, 2 (2008), 1273–1283. https://doi.org/10.1016/j.jpowsour.2008.08.083 cited.

[19] Peter Palensky and Dietmar Dietrich. 2011. Demand side management: Demand response, intelligent energy systems, and smart loads. *IEEE transactions on industrial informatics* 7, 3 (2011), 381–388.

[20] Anshu Ranjan, Pramod Khargonekar, and Sartaj Sahni. 2016. Smart grid power scheduling via bottom left decreasing height packing. In *Computers and Communication (ISCC), 2016 IEEE Symposium on*. IEEE, 1128–1133.

[21] N Benjamin Sendama, Mehdi Laraki, Aawatif Hayar, and Yassine Rifi. 2016. New Renewable Energy Allocation Algorithms based on Bin Packing in a Smart Home.. In *SMARTGREENS*. 309–315.

[22] Navin Sharma, Pranshu Sharma, David Irwin, and Prashant Shenoy. 2011. Predicting solar generation from weather forecasts using machine learning. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*. IEEE, 528–533.

[23] Will Simm, Maria Angela Ferrario, Adrian Friday, Peter Newman, Stephen Forshaw, Mike Hazas, and Alan Dix. 2015. Tiree energy pulse: exploring renewable energy forecasts on the edge of the grid. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1965–1974.

[24] Cynthujah Vivekananthan, Yateendra Mishra, and Fangxing Li. 2014. Real-Time Price Based Home Energy Management Scheduler. *IEEE Transactions on Power Systems* (2014). https://doi.org/10.1109/TPWRS.2014.2358684 cited.

Bringing together Capella and AADL Verification Capabilities*

Hui Zhao
Université Côte d'Azur, CNRS, Inria, I3S

Frédéric Mallet
Université Côte d'Azur, CNRS, Inria, I3S

Ludovic Apvrille
LTCI, Telecom ParisTech, Université Paris Saclay

Abstract

The design of Cyber-Physical systems (CPS) demands to combine discrete models of pieces of software (cyber) components with continuous models of physical components. Such heterogeneous systems rely on numerous domains with competencies and expertise that go far beyond traditional software engineering: systems engineering. In this paper, we explore a model-based approach to systems engineering that advocates the composition of several heterogeneous artifacts (called views) into a sound and consistent system model. Rather than trying to build the universal language able to capture all aspects of systems, we rather propose to bring together small subsets of languages to focus on specific analysis capabilities while keeping a global consistency of all these small pieces of languages. We take as an example, an industrial process based on Capella, which provides (among others) a large support for functional analysis from the requirements to the deployment of components. Even though, Capella is already quite expressive, it does not provide a direct support for schedulability analysis. However, AADL is an language also dedicated to system analysis. It focuses on schedulability analysis, but that does not provide direct support for functional analysis. Rather than trying to extend either Capella or AADL into always more expressive languages to add the missing features we rather extract a pertinent subset of both languages to build a view adequate for conducting schedulability analysis of Capella functional models. Our language is generic enough to extract pertinent subsets of languages and combine them to build views for different experts. It also maintains a global consistency between the different views.

1 Introduction

Developers of Cyber-Physical Systems (CPSs) have to deal with different domains, each of them having different characteristics. It is seldom the case that one development platform or a single language can adapt to all aspects with assumption one-size-fits-all. Therefore developers have to rely on domain-specific languages to handle the different domains problem. This results not only in a proliferation of languages but also

*This work has been supported by the French government, through the UCA^{JEDI} Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-15-IDEX-0001.

increases the design complexity of CPS. Meanwhile, the gaps between languages and platforms brought additional problems such as coherency and consistency problems, which are exposed at integration and simulation stages. This also further exacerbates the complexity, make the complexity skyrocketing.

To tackle these problems, engineers need an approach which can efficiently combine already existing languages. The goal is not only to combine the different languages seamlessly but also to benefit from the advantages of each language. Two solutions are possible. (i) A first solution is to continuously integrate the necessary languages into an existing development platform, thus leading to progressively building a comprehensive development platform. However, this could lead to a never ending process resulting in a gigantic framework, which would be difficult to use and maintain. (ii) A second solution is to keep each language (or tool) isolated, and relate some of the elements from each language (sub) meta-model in order to conduct the different analyses offered by each approach (e.g., scheduling analysis, safety analysis). In this second solution, each domain expert can work independently. Yet, since each language has its own characteristics such as syntax and semantics, we have to eliminate the gaps between them and handle the consistency issues. Therefore, our contribution is to propose a formal combining approach to link two modeling languages. To do so, we define how to relate two (sub-)metamodels. Then, for two given models m_1, m_2 conforming each of the two languages, m_2 can be augmented with some of the information of m_1 so as to perform verification on enriched models (e.g., scheduling, timing, safety), and then to trace the verification results back to m_1.

In this paper, we selected SysML and AADL to demonstrate the relevance of our approach. These languages are supported by the tools Capella/Arcadia and OSATE2 [1], respectively. The paper is organized as follows. We present our contribution with first our workflow and then with the formal definition of a set of combination rules and operators. Then, in section 3, we apply these operators on functional and physical views. In section 4, train traction controlling systems are used to demonstrate architecture and scheduling analysis. Section 5 presents the related work. Finally, section **??** concludes the paper and gives our future work.

Some special notes for this paper are: 1) When we mention Arcadia, it means SysML-based methodology, the language is SysML. 2) In sections 2, 3 and 4, all elements on the left of transformation rules belong to metamodels of Arcadia and all elements on the right are from the AADL metamodel. The two metamodels have been imported by default, and we thus omit the prefix (e.g., *MM.Arcadia*.function) for conciseness.

2 Approach

2.1 Workflow

The workflow of our approach is shown as figure 1. On the one hand the Arcadia methodology and Capella modeling platform focus on several high-level phases of engineering and system functional analysis. On the other hand, the AADL focuses on structural modeling to describe the concrete execution behaviors of components.Hence, in this paper, we aim to enrich the Arcadia model with elements of the AADL model for which a relationship to Arcadia has been established. Firstly, we proposed a set of operators to specify the relationships at the M2 level, which contains corresponding

[1] http://osate.org/index.html

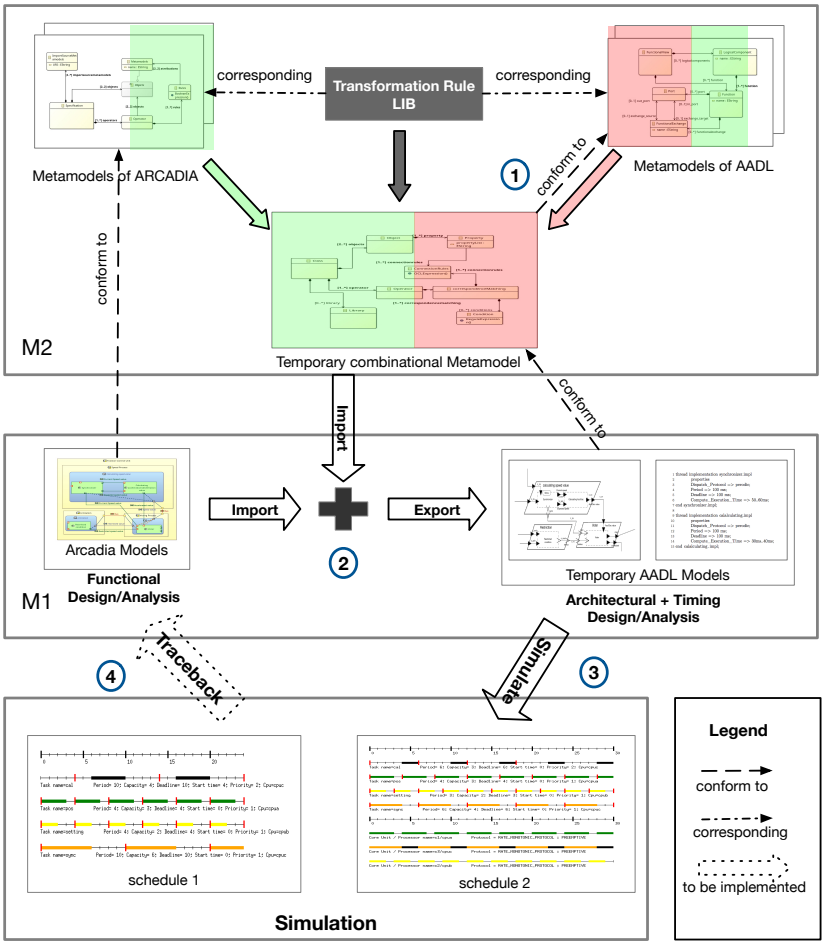


Figure 1: Overview of Workflow

relationships between Arcadia and AADL metamodels and the relationship of other additional attributes. Those relations serve to constitute the transformation system, and the set of all relationships is called Transformation Rule Library (TRL).

In this case, Arcadia and AADL have similar parts but different names (e.g., function in Arcadia and thread in AADL, shown as green part in the figure). To capture and analyze the new features which not exist in Arcadia, Some additional attributes must be added (e.g., period and execution time, shown as red part). Once the metamodels are prepared, we can get a temporary combinational metamodel (TCM) by using TRL at step 1. Moreover, one can use TCM and an existing Arcadia model to export to an AADL model adding new attributes manually at step 2. The new AADl concrete models will be exported into OSATE for further editing. Next, it will be used to perform the scheduling simulation in an analysis tool Cheddar at step 3. The design flaws, conflicts of time and resource would be detected. These results help designers to improve original models at step 4 and fix vulnerabilities or improve the system performances. We expect to be able to automatically trace back to the original model in the near future.

2.2 Operators and Rule expression

According to the workflow in the above subsection, initially, we have a set of metamodels for Arcadia and AADL. We propose a set of operators to manipulate the metamodels, then using these operators to represent operations between metamodels (e.g., transforming, creating, ignoring) in a systematic way. We also provided semantic definitions of operators as well as corresponding writing rules, to help the integration engineer custom-tailor metamodel as they needed. In the following parts of the paper, in order to simply and clearly describe the transformation rules, several operators and their semantics are defined. (see the table 1).

Symbol of operator	**Meaning**
Γ	Transformation Rule
;	End of rule
⇝	Transfer
<>	Parent node
{ }	Attribute
[ ]	Optional element
|	Separation of elements
{ }+	Attribute is to be created
¬	Ignore

Table 1: Symbols of transformation rule expression

2.2.1 Structure of rule

One rule begins with "Γ" and end with ";". What we have to note is each rule can contain one source object which in the left of transfer symbol ⇝ and one or more objects in the right side are target objects. The parent node is enclosed within the angle brackets "<>" (if it has one parent node). it means each rule should write as below:

$$\Gamma < parent > source \rightsquigarrow target;$$

Each part of the object separated by ":". The attribute group is enclosed with parentheses "{ }"; square braces "[ ]" delimit optional elements and the alternatives are separated by a pipe "|". For example, The port has a directional attribute called Direction which could be in or out. Shown as below:

$$P_{ort} : \{Direction[in|out]\}$$

2.2.2 Creating operator

In the case of creating a new attribute, put the name of an attribute in the parentheses with plus "{ }+", is used to present the option is to be created. An example as below, an attribute type of component port will be created with three optional values (data, event, data and event):

$$Port : \{Type[data|event|dataevent]\}+$$

2.2.3 Ignoring operator

For some ignored attributes and objects are denoted with "¬" which is in front of the ignored object.

An example of transformation rule expressions is in listing 1. In particular, the number of attributes of the target object may be greater than the number of attributes of the source object in the case of a new object created.

```
1 ΓP_ort:{Direction[in|out]})⇝ <feature>:Port:{Direction[in|out]}:{Type[data|event|data event]}+;
2 ΓPP ⇝ <feature>:Port:{Direction[in|out]}+:{Type[data|event|data event]}+;
3 ΓP_ort:¬{ordering});
4 ΓEx_fun:{Source}:{Target} ⇝ <connections>:connection:{source}:{target};
```

Listing 1: An example of transformation rules

2.3 Arcadia2AADL model transformation

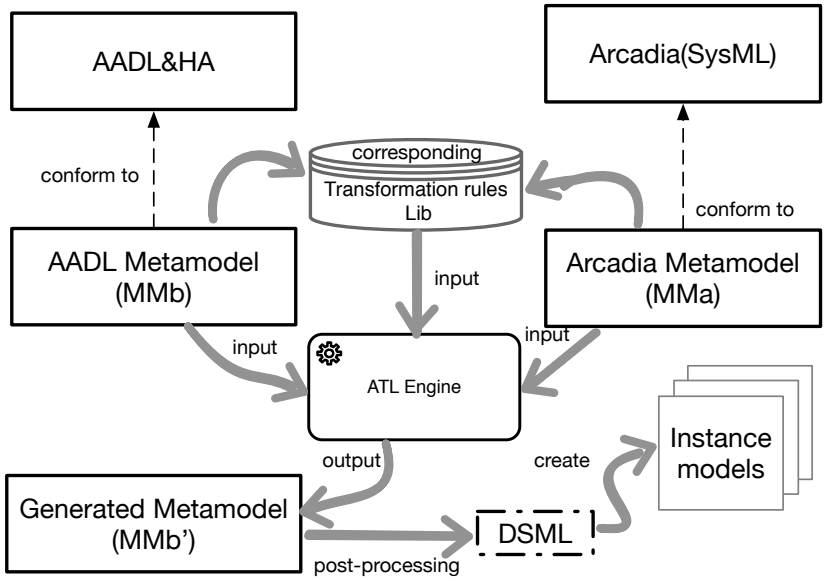


Figure 2: Structure of Arcadia2AADL model transformation

Based on the defined transformation rules, Arcadia2AADL transformation can create a temporary metamodel according to original metamodels and TRL. ATL serves as a transformation engine. For more detailed information about ATL, the reader is referred to [Jouault et al., 2006][Jouault et al., 2008]. The Structure of Arcadia2AADL model transformation is shown in Fig 2. AADL metamodels (denoted as MMb) and Arcadia metamodels (denoted as MMa) are imported (input) into ATL engine, and the engine is going to read transformation rules library and then the transformation engine exports (output) a temporary metamodel (denoted as metamodel MMb') which is a union subset of AADL and Arcadia and conform to AADL metamodel. The generated metamodel is then further post-processed as a DSML (domain-specific modeling language) in *ecore* format which can be used to create instance models. Therefore, all

instance models are conform to this metamodel and will be used for further simulation and analysis.

3 Transformation Rule Library

As we described in the above section, the Transformation Rule Library (TRL) play an important role in the transformation process. Hence, in this section, we present how to construct a TRL from the following three views, functional view and physical view. Each view contains the metamodels which are from the subset of AADL and Arcadia.

3.1 Functional view

3.1.1 Logical components in Arcadia

Logical components in Arcadia contain a set of member elements, such as logical component containers, functions, ports, and functional exchanges. In the Arcadia, Functional diagrams consist of a set of SysML blocks and its interactions, named *Logical components*; The notion of Logical component enables better expression of system engineering semantics compared to SysML, and particularly, reduces the bias towards software. SysML block definition diagrams (BDDs) and internal block diagrams (IBDs) are assigned to different abstract and refined layers, respectively. The definition of a block in SysML can be further detailed by specifying its parts; ports, specifying its interaction points; and connectors, specifying the connections among its parts and ports. This information can also be visualized using logical components in Arcadia. Definition 3.1.1 shows a metamodel of logical components.

Definition 3.1.1. *(Logical Component) A logical component (**LC**) is 5-tuple,*

$$\mathcal{LC} =< C_{omp}, F_{un}, P_{ort}, Ex_{fun}, M_{cf} >$$

where, $C_{omp} = \{\bigcup_{F_{un}}\}$ is a logical component container which contains a set of functional elements.

F_{un} is a finite set of functional block include their name and id attributes. P_{ort} is a finite set of functional ports including directions and allocation attributes. $Ex_{fun} \subseteq P_{ort} \times P_{ort}$ denotes a finite set of functional exchange (connection) between two functional ports, it must be pair, one is source, another is target. $M_{cf} : \Sigma F_{un} \rightarrow C_{omp}$ allocate functions to a logical component container.

In Figure 3, there is a functional instance model of a part of a vehicle traction control unit in ARCADIA as an example. The blue rectangle is named logical component in Arcadia, but we consider it as a function's container, we thus call it *logical component container* C_{omp} in this paper. The green rectangle are functions F_{un} which are contained by C_{omp}. The element M_{cf} has represented this allocation relationship between logical component containers and functions $M_{cf} : \Sigma F_{un} \rightarrow C_{omp}$. The deep green square with the white triangle is the outgoing port (P_{ort}), which connects to an incoming port (P_{ort}) that is drawn as a red square with white triangle and the green line is the functional exchange between two functional ports (Ex_{fun}).

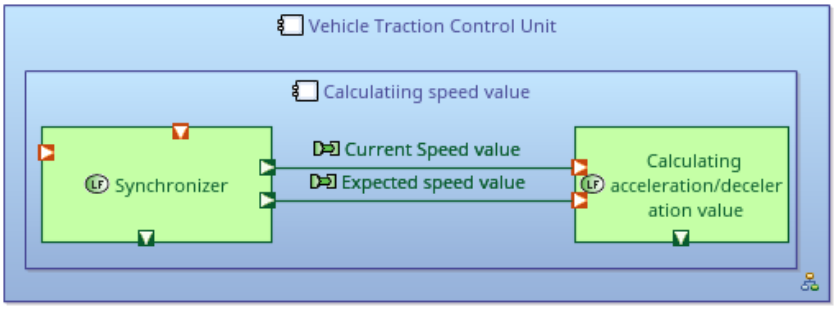


Figure 3: An example of functional view of vehicle traction control unit in ARCADIA

3.1.2 The subset of software in AADL

AADL is able to model a real-time system as a hierarchy of software components, predefined software component types in the category of the components such as thread, thread group, process, data, and subprogram are used to model the software architecture of the system.

Definition 3.1.2. *(Software Composition) A **SC** is a 4-tuple:*

$$\mathcal{SC} =< Type, Port, Connection, Annex >$$

*where Type specifies the type of components (e.g, system, process, thread). Port is a set of communication point of component. Port could be different types such as **data** port, **event** port and **data event** port. And, port can specify the direction such as **in** port, **out** port, **in out** port. Connection is used to connect ports in the direction of data/control flow in uni- or bi-directional.*

3.1.3 Functional elements transformation rules

Table 2 shows the correspondence between AADL and Arcade elements. The Additional attributes column are the attributes to be crated during the transformation. According to this table, we can easily write the transformation rules to transforming Arcadia to AADL on functional parts, denoted $\mathcal{LC} \rightsquigarrow \mathcal{SC} + \mathcal{HA}$. See an example in listing 2.

```
1 ΓC_comp ⇝ Type[ system|process]:{Runtime_Protection[true|false]}+;
2 ΓF_un ⇝ Type[abstract|thread]:{Dispatch\_Protocol[Periodic|Aperiodic|Sporadic|Background|Timed|
      Hybrid]}+;
3 ...
```

Listing 2: Functional elements transformation rules example

3.2 Physical view

3.2.1 Execution Platform in AADL

Processor, memory, device, and bus components are the execution platform components for modeling the hardware part of the system. Ports and port connections are

Arcadia	AADL	Additional attributes
LC container (C_{omp})	System,Process	{Runtime_Protection[true|false]}+
Function (F_{un})	Abstract, Thread	{Dispatch_Protocol[Periodic|Aperiodic|Sporadic|Background|Timed|Hybrid]}+
Port (P_{ort})	Port	{Type[data|event|data event]}+
Functional Exchange (Ex_{fun})	Connection	◯
◯	Annex	Type[abstract| thread]:{annex}+
Physical Node (N_{ode})	Device,Memory,Processor,Bus	{Dispatch_Protocol}+:{Period}:{Deadline}+:{priority}+
Physical Port (PP)	◯	$\neg PP$
Physical Link (PL)	Bus/BusAccess	[{Allowed_Connnection_Type}+:{Allowed_Message_Size}+|{Allowed_Physical_Access}+:{Transmission_Time}+]

Table 2: Functional and Physical elements correspondence table

provided to model the exchange of data and event among components. Functional and non-functional properties like scheduling protocol and execution time of the thread can be specified in components and their interactions.

Definition 3.2.1. *(Execution Platform) A **EP** component is defined as a six tuples:*

$$\mathcal{EP} =< EC, BA, C_{onn} >$$

*where, EC defines the execution component such as processor, memory, bus and device. BA defines the BusAccess which is interactive approach between **bus** component and other execution platform components. $C_{onn} \subseteq EC \times EC$ denotes a finite set of connection between two components via bus device.*

3.2.2 Physical components in Arcadia

The physical component in Arcadia consists of physical Node, Port and Link. The Physical Port and Link correspond to port and bus connection in AADL. There are some choices when the physical Node is translated to AADL such as device, memory, and processor, hence the designer has to point out what type of target component during transformation by using transformation rule express.

Definition 3.2.2. *(Physical Components) A Physical components is 3 tuples,*

$$\mathcal{PC} =< N_{ode}, PP, PL >$$

where, N_{ode} is a execution platform, named node in Arcadia, it could be different type of physical component (e.g, processor, board). PP is the physical component port. PL is physical link, it could be assigned a concrete type such as bus.

Figure 4 is a physical instance model of a part of a vehicle traction control unit in ARCADIA. We can see the yellow parts are the physical node (N_{ode}) and the red line is the physical link (PL) named bus in this case which connects to two physical ports (PP), the small square in dark yellow.

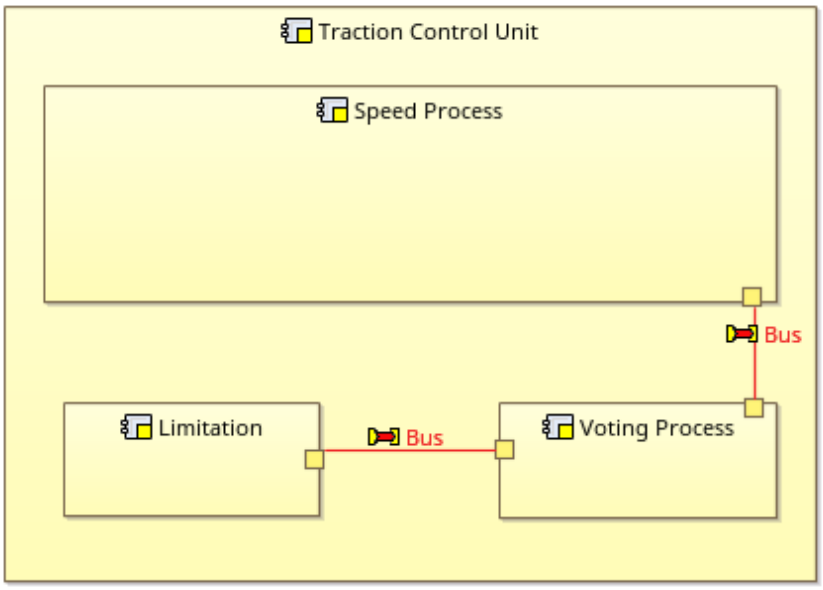


Figure 4: An example of physical view of vehicle traction control unit in ARCADIA

3.2.3 Physical elements transformation rules

According to Table 2, we can easily write the transformation rules for physical elements. Listing 3 shows a part of the code to transform the physical component from Arcadia to AADL.

```
1 ΓN_ode ⇝ [Device|Process|Memory|Bus]:{Dispatch_Protocol}+:{Period}:{Deadline}+:{priority}+;
2 ΓPP ⇝ ¬PP;
3 ΓPL ⇝ Bus/BusAccess:[{Allowed_Connnection_Type}+:{Allowed_Message_Size}+|{
      Allowed_Physical_Access}+:{Transmission_Time}+];
```

Listing 3: Physcial elements transformation rules example

Let us focus on the physical link part (see line 3). The Bus device could be a logical resource or hardware component. Hence, the bus device has different properties depending on the role. When the bus is considered as a logical resource, it contains the properties *Allowed_connection_type and Allowed_Message_Size*. When the bus is hardware, it contains *Allowed_Physical_Access and Transmission_Time*. Therefore, we write the rules that either

$$\{Allowed\_Connnection\_Type\}+ : \{Allowed\_Message\_Size\}+$$

or

$$\{Allowed\_Physical\_Access\}+ : \{Transmission\_Time\}+$$

4 Case Study

To show the efficacy of our approach in transforming and using produced AADL models to analyze the properties, this section presents the experimental results of analyzing the traction controlling unit of railway signaling system. By using our proposed approach, we transfer and extend Arcadia metamodel, and design AADL using OSATE2 with the generated metamodel. once the concrete models have been created, the scheduling property is chosen to show analysis ability through Cheddar tool.

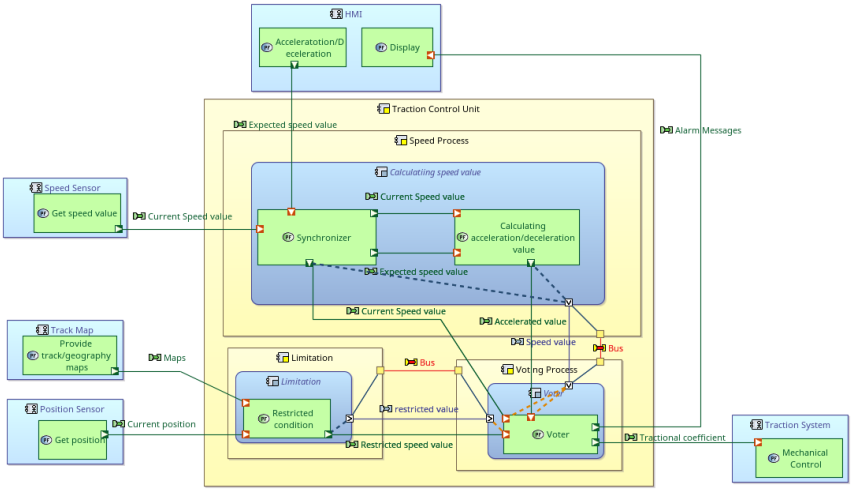


Figure 5: Arcadia model of TCU system

4.1 Train traction controlling system (TCU)

Train movement is the calculation of the speed and distance profiles when a train is traveling from one point to another according to the limitations imposed by the signaling system and traction equipment characteristics. As the train has to follow the track, the movement is also under the constraints of track geometry, and speed restrictions and the calculation becomes position-dependent. The subsystem of calculating the traction effective and speed restrictions is therefore critical to achieving train safe running. Nowadays, Communication based train control (CBTC) is the main method of rail transit (both urban and high-speed train) which adopts wireless local area networks as the bidirectional train-ground communication [Zhu et al., 2009]. To increase the capacity of rail transit lines, many information-based and digital components have been applied for networking, automation and system inter-connection, including general communication technologies, sensor networks, and safety-critical embedded control system.

This paper uses a subsystem which called traction controlling system (TCU) from signaling system of high-speed railways, we use it to illustrate the model transformation from engineering level to detailed architectural level and verified the instance models. The functional modules such as calculation and synchronization will be transformed using our approach, and then non-functional properties such as timing correctness and resource correctness will be verified by schedule analysis tool Ched-

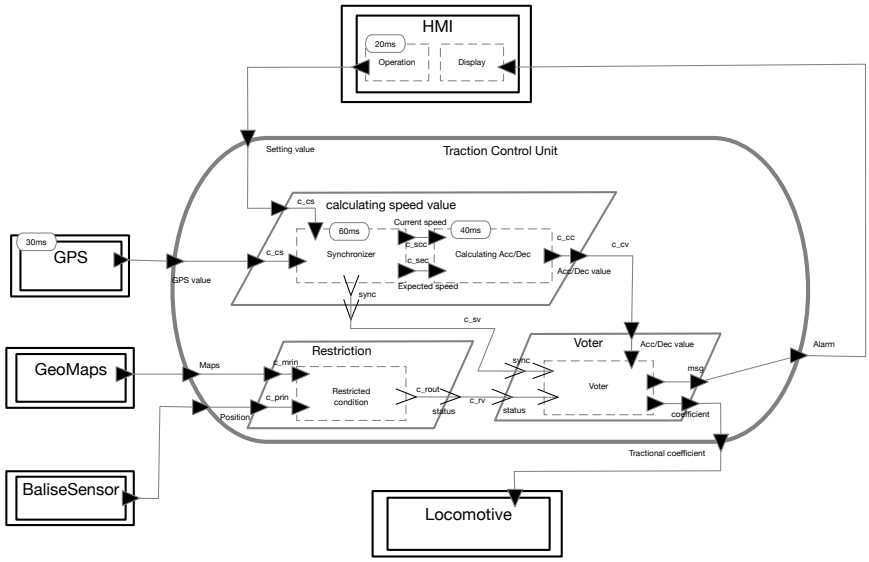


Figure 6: AADL model of TCU system

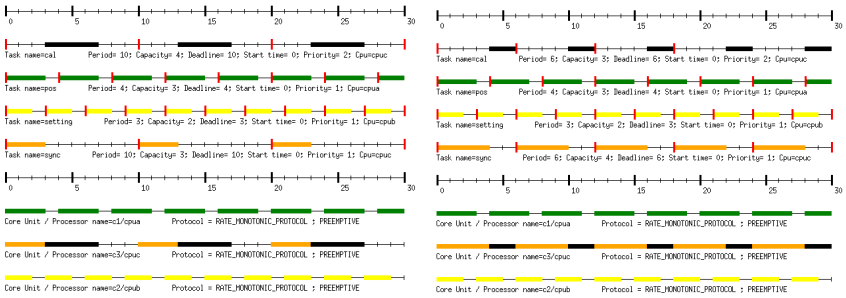
dar [Singhoff et al., 2004].

First, we start with component functional views and physical view analysis by designing system models in Arcadia (shown in figure 5). The functions of the traction control system are to collect the external data by sensors such as a speed sensor. The data from Balise sensors is used to determinate the track block, and then it is going to seek the speed restriction conditions by matching accurate positioning (if the track blocks are divided fine enough) and digital geometric maps data. Meanwhile, calculating speed unit received the speed data from GPS and speed control commands from HMI (Human-Machine Interface) periodically. GPS data provides speed value periodically, and HMI data send the operation command (e.g., expected speed value), then the calculating unit has to output an acceleration value and export to the locomotive mechanical system. Although they are periodic, the external data do not always arrive on time due to transmission delay or jitter. Therefore, we should use a synchronizer to make sure they are synchronized. Otherwise, the result would be wrong with asynchronous data. Similarly, to ensure the correctness of the command of acceleration (or deceleration), we applied a voting mechanism which can ensure the result is correct as much as possible. The voter must have the synchronized signal and restriction condition to dedicate to output the acceleration coefficient request to the locomotive system. The AADL diagram shown as figure 6.

4.2 Model transformation

Using the Arcadia2AADL tool, the metamodel of the TCU system in Capella is translated into the corresponding AADL metamodel with the rules and approach which describes in section 3. For instance, on the one hand, the function class is translated into the thread in AADL. To analyze the timing properties, several attributes also have been added such as protocol type, deadline, execution time, period.

On the other hand, the physical part element Node translates to the processor in this



(a) Schedule 1 with idle time (b) Schedule 2 with compact time

Figure 7: Simulation results of task schedule

case. Differ from simple physical Node in Arcadia; the processor element attaches rich properties such as scheduling protocol (scheduler type), process execution time. The allocation relationships on both physical and functional parts are translated into AADL as well.

4.3 Schedule verification

The external data and internal process work sequentially is an essential safety requirement of the system, and each task should be scheduled properly. However, in real-world, the risk of communication quality and rationality of scheduling must be taken into account. Therefore, the schedule verification is a way to evaluate system timing property. An Ada framework called Cheddar which provides tools to check if a real-time application meets its temporal constraints. The framework is based on the real-time scheduling theory and is mostly written for educational purposes [Marcé et al., 2005].

```
thread implementation synchronizer.impl
          properties
          Dispatch_Protocol => perodic;
          Period => 100 ms;
          Deadline => 100 ms;
          Compute_Execution_Time => 50..60ms;
end synchronizer.impl;

thread implementation calalculating.impl
          properties
          Dispatch_Protocol => perodic;
          Period => 100 ms;
          Deadline => 100 ms;
          Compute_Execution_Time => 30ms..40ms;
end calalculating.impl;

thread implementation gps.position
          properties
          Dispatch_Protocol => perodic;
          Period => 40 ms;
          Deadline => 40 ms;
          Compute_Execution_Time => 30ms..40ms;
end gps.position;

```

```
25 thread implementation HMI.setting
26          properties
27          Dispatch_Protocol => perodic;
28          Period => 30 ms;
29          Deadline => 30 ms;
30          Compute_Execution_Time => 20ms..30ms;
31 end HMI.setting;
```

Listing 4: Setting of scheduling properties

Listing 4 shows a set of 4 periodic tasks (cal, pos, sync and setting) of TCU respectively defined by the periods 100, 100, 40 and 30, the capacities 60, 40, 30 and 20, and the deadlines 100, 100, 40 and 30. These tasks are scheduled with a preemptive Rate Monotonic scheduler (the task with the lowest period is the task with the highest priority).

For a given task set, if a scheduling simulation displayed XML results in the Cheddar. One can find the concurrency cases or idle periods (see left of figure 7, comprise the software part and physical device part). People change the parameters directly and reload simulation; a feasible solution can be applied instead. After tuning, finally, the appropriate setting has displayed as in right of figure 7. According to this simulation result, people can correct the properties value in AADL, thereby ensure the correctness of system behavior timing properties.

5 Related work

A considerable number of studies have been proposed on extending UML-like profile to AADL and model transformation methods. This section provides a brief introduction to these works.

An approach for translating UML/MARTE detailed design into AADL design has proposed by Brun et al. [Brun et al., 2008]. Their work focuses on the transformation of the thread execution and communication semantics and does not cover the transformation of the embedded system component, such as device parts. Similarly, in [Turki et al., 2010], Turki et al. proposed a methodology for mapping MARTE model elements to AADL component. They focus on the issues related to modeling architecture, and the syntactic differences between AADL and MARTE are well handled by the transformation rules provided by ATL tool, yet they did not consider issues related to the mapping of MARTE properties to AADL property. In [Ouni et al., 2016], Ouni et al. presented an approach for transformation of Capella to AADL models target to cover the various levels of abstraction, they take into account the system behavior and the hardware/software mapping. However, the formal definition and rigorous syntactic of transformation rules are missed.

The scientists have proposed some specific methods to weave the models as well as metamodels formally such as [Jezequel, 2008], Degueule has proposed Melange, a language dedicated to merging languages [Degueule et al., 2015], and similar works like [Ramos et al., 2007]. However, the structural properties are not supported.

Behjati et al. describe how they combined SysML and AADL in [Behjati et al., 2011] and provided a common modeling language (in the form of the ExSAM profile) for specifying embedded systems at different abstraction levels. De Saqui-Sannes et al. [De Saqui-Sannes and Hugues, 2012] presented an MBE with TTool and AADL at the software level and demonstrated with flight management system. Both of their works do not provide the description in a formal way.

Compared with current studies, the approach proposed in this paper has the following features:

1. Arcadia is chosen as the transformation source. Arcadia provides a broad view of system engineering as well as refined functional and physical views.
2. A proper subset of AADL has been chosen as the transformation target including functional software composition, execution platform. We use it to describe continuous behaviors of Cyber-Physical System.
3. All of the transformations is considered at metamodel level, and then a generated synthesized metamodel can be used to create concrete AADL models for further analysis.
4. Transformation rules are formally defined, and then it is readable by human and easier to verify the correctness of transformation.

6 Conclusions and future work

This paper describes a collaborative design approach between system engineering methodology Arcadia (based on SysML) and architectural design language AADL using transformation at metamodel level. We first present our approach and implementation procedures using ATL. Then, we give a formal description of the key modeling elements of Arcadia and AADL, respectively. Then translation rules from these Arcadia metamodels to AADL are formally defined. Finally, a case study of train traction controlling system is used to demonstrate the transformation from engineering concerned design into an architectural refinement design which can be further analyzed by scheduling properties. In our future work, we will study the translation rules for more elements of Arcadia and also for comprehensive SysML elements, even for others UML-like profiles such as MARTE. At the same time, we will continue to explore the AADL and its annex to support more analysis and formal verification of system design. Besides, the safety-critical systems have become a trend in industrial files. We will study the extension of AADL with verification of safety properties with transformation methodology. For further discussions on this topic, please refer to full text [Zhao et al., 2019].

References

[Behjati et al., 2011] Behjati, R., Yue, T., Nejati, S., Briand, L., and Selic, B. (2011). Extending SysML with AADL concepts for comprehensive system architecture modeling. In *European Conference on Modelling Foundations and Applications*, pages 236–252. Springer.

[Brun et al., 2008] Brun, M., Vergnaud, T., Faugere, M., and Delatour, J. (2008). From UML to AADL: an Explicit Execution Semantics Modelling with MARTE. In *ERTS 2008*.

[De Saqui-Sannes and Hugues, 2012] De Saqui-Sannes, P. and Hugues, J. (2012). Combining SysML and AADL for the design, validation and implementation of critical systems. In *ERTS2 2012*, page 117.

[Degueule et al., 2015] Degueule, T., Combemale, B., Blouin, A., Barais, O., and Jezequel, J.-M. (2015). Melange: A meta-language for modular and reusable development of dsls. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering*, pages 25–36. ACM, ACM.

[Jezequel, 2008] Jezequel, J.-M. (2008). Model driven design and aspect weaving. *Software and Systems Modeling*, 7(2):209–218.

[Jouault et al., 2008] Jouault, F., Allilaire, F., Bézivin, J., and Kurtev, I. (2008). ATL: A model transformation tool. *Science of Computer Programming*, 72(1-2):31–39.

[Jouault et al., 2006] Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I., and Valduriez, P. (2006). *ATL: a QVT-like transformation language*. a QVT-like transformation language. ACM, New York, USA.

[Marcé et al., 2005] Marcé, L., Singhoff, F., Legrand, J., and Nana, L. (2005). Scheduling and Memory Requirements Analysis with AADL. In *Proceedings of the 2005 Annual ACM SIGAda International Conference on Ada*, pages 1–10, New York, NY, USA. ACM.

[Ouni et al., 2016] Ouni, B., Gaufillet, P., Jenn, E., and Hugues, J. (2016). Model Driven Engineering with Capella and AADL. page 0.

[Ramos et al., 2007] Ramos, R., Barais, O., and Jezequel, J.-M. (2007). Matching model-snippets. In *International Conference on Model Driven Engineering Languages and Systems*, pages 121–135. Springer.

[Singhoff et al., 2004] Singhoff, F., Legrand, J., Nana, L., and Marcé, L. (2004). Cheddar - a flexible real time scheduling framework. *SIGAda*, pages 1–8.

[Turki et al., 2010] Turki, S., Senn, E., and Blouin, D. (2010). Mapping the MARTE UML profile to AADL. In *Proceedings of the 3rd International Workshop on Model Based Architecting and Construction of Embedded Systems (ACES-MB 2010)*, pages 11–20. Citeseer.

[Zhao et al., 2019] Zhao, H., Apvrille, L., and Mallet., F. (2019). Meta-models combination for reusing verification techniques. In *Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD,*, pages 39–50. INSTICC, SciTePress.

[Zhu et al., 2009] Zhu, L., Zhang, Y., Ning, B., and Jiang, H. (2009). Train-ground communication in CBTC based on 802.11 b: Design and performance research. In *Communications and Mobile Computing, 2009. CMC'09. WRI International Conference on*, pages 368–372. IEEE.

Session commune aux groupes de travail MFDL, AFSEC et à AFADL

Méthodes Formelles dans le Développement Logiciel — Approches Formelles des Systèmes Embarqués Communicants — Approches Formelles dans l'Assistance au Développement de Logiciels

Une analyse de la propriété fondamentale de vivacité du protocole Chord*

Julien Brunel, David Chemouil et Jeanne Tawa
ONERA DTIS & Université fédérale de Toulouse
F-31055 Toulouse
prénom.nom@onera.fr

28 mars 2019

Résumé

Chord est un protocole qui fournit une table de hachage distribuée évolutive sur un réseau peer-to-peer sous-jacent et qui constitue une cible intéressante pour spécification et la vérification formelles. Les travaux antérieurs ont principalement porté sur des preuves automatiques des propriétés de *sûreté* ou des preuves manuelles de la correction complète du protocole (une propriété de *vivacité*). Dans cet article, nous rendons compte de l'analyse automatique de la correction de Chord au moyen du langage Electrum (développé dans des travaux antérieurs) sur de petites instances de réseau. En particulier, nous avons pu trouver différents cas problématiques dans des travaux antérieurs et montrer que le protocole n'était pas correct tel que décrit alors. Nous avons résolu tous ces problèmes et fourni un version du protocole pour laquelle nous n'avons pas trouvé de contre-exemple en utilisant notre méthode.

1 Introduction

Les systèmes peer-to-peer sont des systèmes distribués sans organisation hiérarchique ou contrôle centralisé. Chord [LNBK02] est l'un des systèmes peer-to-peer les plus populaires. C'est un protocole et algorithme fournissant une table de hachage distribuée (DHT). Une DHT stocke des paires clé-valeur en attribuant des clés à différents nœuds du réseau. Chord traite de la localistion efficace et robuste de données dans un tel réseau. Quand Chord a été initialement présenté, trois qualités principales ont été soulignées : sa simplicité, sa performance et sa correction. Bien que les deux premières affirmations soient valides, démontrer la correction de

*Résumé de [BCT18]. Travaux partiellement financés par le Fonds Européen de Développment Régional (FEDER) à travers le Programme Opérationnel pour la Compétitivité et l'Internationalisation (COMPETE2020), au travers de la *Fundação para a Ciência e a Tecnologia* (FCT) dans le cadre du projet POCI-01-0145-FEDER-016826 et dans le cadre du projet FORMEDICIS de l'Agence Nationale de la Recherche (ANR-16-CE25-0007).

Chord se révèle être une tâche difficile, comme le montrent les nombreux travaux de P. Zave, dont le plus récent est [Zav17].

Dans Chord, chaque nœud dispose d'un identifiant et peut atteindre d'autres nœuds au moyen de pointeurs sur eux. Les nœuds et leurs pointeurs forment une topologie qui est essentielle pour assurer la localisation correcte des données dans le réseau. Comme des nœuds peuvent rejoindre ou quitter le réseau (ou échouer) à tout moment, la topologie est en constante évolution. Un aspect fondamental du protocole réside dans la définition d'opérations de *maintenance* en charge de la réparation de la topologie du réseau, de sorte à ce que les données stockées dans n'importe quel nœud continuent à être accessibles depuis tout autre noeud, malgré les pannes, les arrivées et les départs.

Ainsi, la correction de Chord concerne la topologie du réseau. En fait, les nœuds et leurs pointeurs "successeur" doivent former un *anneau*, de telle sorte que chaque nœud soit accessible depuis n'importe quel autre nœud. Puisque les nœuds peuvent rejoindre et quitter le réseau, la topologie en anneau ne peut pas toujours être assurée. C'est pourquoi la propriété de correction de Chord est exprimée comme suit : *si, à partir d'un instant donné, il n'y a plus d'arrivée, départ ou échec, le réseau est alors assuré de revenir à terme dans une topologie en anneau et d'y rester.* La correction de Chord ne concerne donc pas seulement la structure du réseau, mais aussi son évolution temporelle : il s'agit en fait d'une propriété de *vivacité*. Cette double nature constitue l'une des raisons de la difficulté à démontrer la correction.

Nous avons utilisé Electrum [1] [MBC+16] pour analyser Chord. Electrum est un langage de spécification basé sur la logique temporelle linéaire du premier ordre, dans lequel les propriétés structurelles et temporelles peuvent facilement être définies et vérifiées. Il est inspiré d'Alloy [Jac12] pour ses aspects structurels et de la logique temporelle linéaire (LTL) pour ses concepts temporels.

2 Le protocole Chord

Dans un réseau Chord, chaque nœud a un identifiant (le *hash* à m bits de son adresse IP). Les couples de clés et des données associées sont stockées dans les nœuds. Chaque nœud maintient une *liste de successeurs*, des pointeurs sur d'autres nœuds. Nous appelons le premier élément de cette liste le *successeur*. L'objectif de disposer d'une liste de successeurs au lieu d'un seul est d'être robuste aux pannes : si un noeud quitte le réseau, son prédécesseur a, normalement, toujours des successeurs dans le réseau. En outre, chaque nœud a également un pointeur sur son *prédécesseur*, nécessaire pour l'exécution des opérations de maintenance.

Quand un réseau est structuré en anneau vis-à-vis de la relation induite par les pointeurs *successeurs* et lorsque l'ordre de identifiants est conforme à l'ordre des pointeurs *successor*, alors chaque nœud est accessible depuis n'importe quel autre nœud, *i.e.* toutes les données sont accessibles depuis n'importe quel noeud. Nous disons qu'un tel réseau est dans un état *idéal*.

1. Cf. https://haslab.github.io/Electrum.

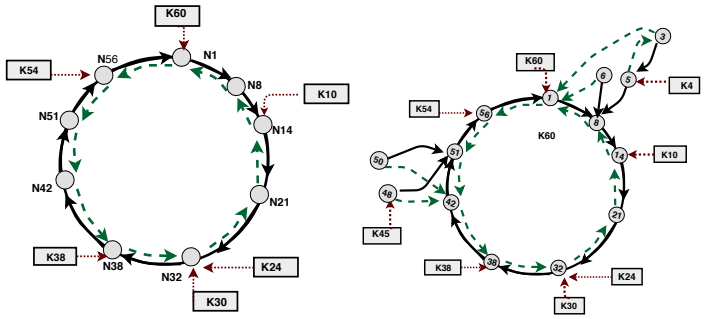


FIGURE 1 – Réseaux idéal et valide (pointeurs “successeur” en gras, pointeurs “prédécesseur” en tireté, et associations clé/données en pointillé).

La structure d’anneau ne peut pas être assurée en permanence. Par exemple, les nœuds rejoignant un anneau créent un *appendice*. Les opérations de maintenance visent alors à revenir à une structure en anneau.

Les auteurs de Chord ont décrit des propriétés du réseau qui assurent la transmission correcte des données [LNBK02]. Ils définissent notamment l’état *idéal*, déjà introduit informellement, et un état imparfait dans une certaine mesure, qu’on dit *valide* [Zav17] (cf. fig. 2).

3 Spécification et vérification

Nous avons spécifié le protocole Chord au moyen du langage Electrum [2]. Notre modèle est inspiré de celui, en Alloy, de P. Zave [Zav17], mais il en diffère grandement sur la forme en raison d’un idiome de modélisation différent. Le modèle traite par ailleurs de Chord avec des nœuds à deux successeurs seulement. De manière plus importante, l’emploi d’Electrum (avec une extension [BCC+18] pour spécifier les événements plus simplement), en particulier de sa couche de logique temporelle, permet de construire un modèle plus abstrait, ne nécessitant pas de spécifier d’invariants pour l’analyse (la recherche d’invariants a été une tâche ardue relatée par P. Zave dans diverses publications). Elle a permis de trouver rapidement plusieurs petites erreurs dans la modélisation de P. Zave et a par ailleurs permis de traiter la propriété de vivacité, tandis que la modélisation en Alloy ne permettait de traiter automatiquement que les propriétés de sûreté.

Nous avons aussi trouvé un problème plus important qui a nécessité l’extension du modèle au moyen d’un nouvel événement (par rapport à celui de P. Zave) mais qui, finalement, correspondait à une proposition des auteurs de Chord dans un de leurs articles. Enfin nous avons dégagé des propriétés d’équité raisonnables nécessaires à l’établissement de la correction.

2. Le modèle complet est disponible ici : https://doi.org/10.5281/zenodo.1322052.

La vérification a été effectué sur un PC sous GNU/Linux équipé d'un processeur Intel Xeon E5-2699 fournissant 512 Go de RAM. En fonction des analyses à effectuer, nous nous sommes basés sur les *model-checkers* bornés et non-bornés proposés dans Electrum AnalyZer. Le premier repose sur une traduction en problème SAT mise en œuvre par Electrum lui-même ou sur un algorithme fourni par nuXmv [CCD+14]. Le second repose sur un algorithme de nuXmv.

Nous d'abord vérifié que notre modèle est cohérent (il admet une instance) et que toutes les branches d'action sont réalisables. Toutes ces analyses ont abouti positivement en moins de 10 s. en utilisant le mode BMC d'Electrum Analyzer. La correction, elle, peut être vérifiée par l'analyseur borné pour des réseaux avec 4-6 nœuds et un horizon de 10 pas (4 nœuds pour un horizon de 15), tandis que l'analyseur non-borné donne un résultat pour les réseaux de 4 membres seulement (le tout avec un *time-out* de 5 h.).

Références

[BCC+18] Julien Brunel, David Chemouil, Alcino Cunha, Thomas Hujsa, Nuno Macedo, and Jeanne Tawa. Proposition of an action layer for electrum. In *Abstract State Machines, Alloy, B, TLA, VDM, and Z*, pages 397–402. Springer, 2018.

[BCT18] Julien Brunel, David Chemouil, and Jeanne Tawa. Analyzing the Fundamental Liveness Property of the Chord Protocol. In *Formal Methods in Computer-Aided Design*, Austin (USA), October 2018.

[CCD+14] Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. The nuxmv symbolic model checker. In *Computer Aided Verification - 26th International Conference, CAV 2014.*, pages 334–342, 2014.

[Jac12] Daniel Jackson. *Software Abstractions : logic, language, and analysis*. MIT press, 2012.

[LNBK02] David Liben-Nowell, Hari Balakrishnan, and David Karger. Analysis of the evolution of peer-to-peer systems. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 233–242. ACM, 2002.

[MBC+16] Nuno Macedo, Julien Brunel, David Chemouil, Alcino Cunha, and Denis Kuperberg. Lightweight Specification and Analysis of Dynamic Systems with Rich Configurations. In *Foundations of Software Engineering*, 2016.

[Zav17] Pamela Zave. Reasoning about identifier spaces : How to make chord correct. *IEEE Transactions on Software Engineering*, 43(12) :1144–1156, Dec 2017.

Approches au Développement Formel de Systèmes Hybrides Basées sur la Preuve : Logique Dynamique et Event-B

G. Dupont, Y. Aït-Ameur, M. Pantel, N. K. Singh
INPT-ENSEEIHT/IRIT, University of Toulouse, France
{guillaume.dupont,yamine,marc.pantel,nsingh}@enseeiht.fr

Résumé

Les systèmes hybrides sont la fusion d'un programme discret avec un environnement analogique. Ils occupent aujourd'hui une part importante de notre environnement et réalisent des tâches très diverses, qui sont parfois critiques. La formalisation et la preuve de tels systèmes est un problème important qui doit être étudié si l'on veut instiguer de la confiance en ces systèmes auprès de ses utilisateurs. Dans ce papier, nous nous intéressons à deux approches *correctes par construction* et basées sur la preuve. Event-B [1] – avec Rodin [2] – ainsi que la logique différentielle dynamique [8] – avec KeYmaera [9] – sont appliquées à une même étude de cas puis comparées. De cette étude nous tirons ensuite une approche générique à la modélisation des systèmes hybrides avec Event-B.

1 Étude de Cas

Nous nous intéressons à une étude de cas développée par Quesel et al. dans [9] : une voiture roule à la position p et avec une vitesse et une accélération v et a respectivement. Le but est de concevoir un contrôleur qui fait freiner automatiquement la voiture avant un panneau stop, qui se trouve à la position SP.

L'accélération est le seul paramètre sur lequel le contrôleur peut agir. Pour des raisons de simplification, elle change de manière discrète et peut prendre la valeur 0 (vitesse stable), $A > 0$ (vitesse en augmentation) et $-b$, $b > 0$ étant la force de freinage de la voiture. La vitesse de la voiture est par ailleurs physiquement comprise entre 0 et V_{max}.

Tant que la voiture est suffisamment loin de l'objectif, l'utilisateur peut changer d'accélération à loisir ; mais lorsque la voiture s'approche de SP, le contrôleur prend le relais et amorce le freinage du véhicule. La propriété de sûreté du système est donc la suivante : $\forall t \in \mathbb{R}^+ \cdot p(t) < SP$.

Le système obéit à l'équation différentielle $\dot{v}(t) = a, \dot{p}(t) = v(t)$.

2 Approche avec d$\mathcal{L}$/KeYmaera

Quesel et al. développent dans [9] un programme hybride répondant à l'étude de cas (c.f. : Figure 1).

Le principe général de ce programme hybride est de jouer alternativement et indéfiniment le contrôleur ($ctrl$) et le système sous contrôle ($plant$). Le contrôleur se contente de faire le choix de l'accélération de la voiture en fonction de la distance à SP. La partie processus continu implémente simplement

les équations différentielles données dans la Section 1. Ces équations sont données avec un *domaine d'évolution* (après le symbole &) qui dénote dans quel domaine le système est sensé évoluer.

Dans le cas présent, le domaine est divisé en deux : un domaine où $safe$ est vrai et un autre où il est faux ; cela permet de signifier que dans le cas où le système entre dans une zone non sûre le contrôleur prend la main sur le système continu.

$$
\begin{array}{lcll}
init & \rightarrow & [(ctrl; plant)^*]\,(req) & (1)\\
init & \equiv & v \geq 0 \wedge A > 0 \wedge B > 0 \wedge safe & (2)\\
safe & \equiv & p + \frac{v^2}{2B} < SP & (3)\\
ctrl & \equiv & (?safe; a := A) \cup (?v = 0; a := 0) \cup (a := -B) & (4)\\
plant & \equiv & (p' = v, v' = a \& v \geq 0 \wedge p + \frac{v^2}{2B} \leq SP) & (5)\\
 & & \cup (p' = v, v' = a \& v \geq 0 \wedge p + \frac{v^2}{2B} \geq SP) & \\
req & \equiv & p \leq SP & (6)
\end{array}
$$

FIGURE 1 – Hybrid program for the self-driven car

Pour prouver que l'invariant de sûreté (req) est respecté par le système, KeYmaera propose de prouver la ligne (1), à savoir *si les conditions initiales ($init$) sont vraies alors quelle que soit l'exécution de ce dernier, req demeure vrai*. Cette quantification sur les exécutions du système est exprimée à l'aide la modalité $[\cdot]$.

3 Approche avec Event-B/Rodin

Dans cette section, nous donnons les grandes lignes de l'approche que nous avons déployée pour modéliser cette étude de cas.

3.1 Extension de la Méthode

La méthode Event-B est basée sur la théorie des ensembles et la logique du premier ordre. Bien que très expressif, ce cadre est assez limitant dès lors qu'il faut manipuler des concepts tels que les réels et la continuité. Afin d'aider à la définition d'extensions à Event-B, [3, 4] ont introduit la notion de *théorie*, un regroupement de types, axiomes, opérateurs et théorèmes qui peuvent être utilisés dans divers modèles Event-B.

Notre travail s'appuie grandement sur ce mécanisme, et nous avons défini à travers lui tous les concepts dont nous avions besoin : continuité, équations différentielles, solvabilité, etc.

3.2 Principe

Observons qu'un système hybride se compose de deux parties : un contrôleur (programme discret que nous voulons écrire) et un processus continu (ou *plant*, le phénomène physique que l'on veut contrôler). Nous décomposons le système en 4 catégories de comportements :

- les **transitions** : changement interne du contrôleur, décision de l'utilisateur
- les **détections** (*sensing*) : changement du contrôleur dû à l'évolution du processus continu
- les **commandes** (*actuation*) : action du contrôleur sur le processus continu
- l'**environnement** : changement du processus continu dû à des perturbations issues de l'environnement (par exemple : vent, pluie, etc.)

De cette observation on déduit un *modèle générique* Event-B qui se compose d'un événement pour chaque catégorie de comportement. À cela s'ajoute une modélisation explicite de l'état discret (en tant que variable discrète) et de l'état continu (en tant que fonction du temps).

Le temps est également modélisé sous la forme d'une variable t en lecture seule, dont le passage est simulé par un événement particulier, `Progress`, qui s'assure que le temps est globalement croissant.

Ce modèle générique s'instancie à l'aide du raffinement. Le raffinement de donnée est utilisé pour concrétiser les états continus et discrets, et le raffinement d'événements est utilisé pour construire les événements réels du système à partir des événements génériques.

3.3 Sémantique

La sémantique classique d'Event-B consiste à envisager les modèles comme des systèmes de transition en entrelaçant les événements qui le composent. Chaque événement est instantané et fait progresser le système d'un état à un autre.

Mais cette sémantique se révèle insuffisante pour traiter des systèmes continus, qui ont une forte composante temporelle. Aussi, nous étendons cette sémantique par l'ajout d'un type d'événement particulier : les événements continus. Contrairement à leurs pendants discrets (dont l'exécution est absolument instantanée), les événements continus s'exécutent pendant un temps prolongé.

Pour être précis, les événement continus concernent le système sous contrôle. Il s'exécutent tant que le contrôleur n'a pas d'événement discret à faire. Dès lors qu'un événement discret est activé, le contrôleur réalise cet événement immédiatement (préemption) avant de revenir à un nouvel événement continu (dépendant probablement du nouvel état du contrôleur).

Les événements de commande (*actuation*) et d'environnement (et *a fortiori* tout événement qui modifie l'état continu) sont des événements continus.

4 Analyse

L'approche de Platzer s'attache à modéliser le système à un niveau fonctionnel proche du programme contrôleur lui-même. La structure même de l'approche rend la décomposition et la construction étape par étape assez difficile, même si des formes de raffinement semblent voir le jour [7].

La puissance de $\mathrm{d}\mathcal{L}$ et des programmes hybrides en général réside plus dans le système de preuve qui l'accompagne que dans les modèles qu'ils permettent d'écrire. En particulier, un certain nombre de concepts sont occultés des programmes hybrides (typiquement, les problèmes de solvabilité des équations différentielles) et sont en fait manipulés en arrière-plan de certaines règles d'inférence.

De son côté, Event-B, de par l'extrême simplicité de son système formel (théorie des ensembles et logique du premier ordre) offre une très grande expressivité, ce qui permet de gérer au niveau du modèle les spécificités du monde continu. Cependant, la manipulation de structures complexes (en particulier au travers du plug-in Theory) rend les preuves beaucoup plus complexes.

Grâce au raffinement qui est au coeur de la méthode, il est possible d'utiliser Event-B pour formaliser des systèmes hybrides à un haut niveau d'abstraction, ce qui permet d'explorer divers schémas d'implémentation (différents types de discrétisation, schémas à contrôleurs multiples, etc.).

5 Conclusion et Travaux Futurs

L'approche présentée a été appliquée avec succès à deux études de cas dans [6] et [5]. Les modèles complets de ces deux études de cas sont disponibles à l'adresse `https://www.irit.fr/~Guillaume.Dupont/models.php`.

Le côté générique de l'approche permet d'explorer divers possibilités quant à la conception de systèmes hybrides. En particulier, différents patrons de conception de contrôleurs peuvent être exprimés :

un ou plusieurs contrôleurs, un ou plusieurs systèmes, types spécifiques d'équations différentielles (linéaires, autonomes, etc.) et ainsi de suite.

Le coeur de la méthode étant le raffinement, il est également possible d'envisager d'autres types de raffinement lié au monde continu, par exemple au niveau de la discrétisation.

Remerciements

Ce travail a été financé par l'ANR-17-CE25-0005 (projet DISCONT `http://discont.loria.fr`) de l'Agence Nationale de la Recherche (ANR).

Références

[1] Abrial, J.R. : Modeling in Event-B : System and Software Engineering. Cambridge University Press, New York, NY, USA, 1st edn. (2010)

[2] Abrial, J.R., Butler, M., Hallerstede, S., Hoàng, T.S., Mehta, F., Voisin, L. : Rodin : an open toolset for modelling and reasoning in Event-B. International Journal on Software Tools for Technology Transfer 12(6), 447–466 (2010)

[3] Abrial, J.R., Butler, M., Hallerstede, S., Leuschel, M., Schmalz, M., Voisin, L. : Proposals for mathematical extensions for Event-B. Tech. rep. (2009)

[4] Butler, M., Maamria, I. : Practical theory extension in Event-B. In : Liu, Z., Woodcock, J., Zhu, H. (eds.) Theories of Programming and Formal Methods, Lecture Notes in Computer Science, vol. 8051, pp. 67–81. Springer Berlin Heidelberg (2013)

[5] Dupont, G., Aït-Ameur, Y., Pantel, M., Singh, N.K. : Hybrid systems and event-b : A formal approach to signalised left-turn assist. In : New Trends in Model and Data Engineering. pp. 153–158. Springer International Publishing (2018)

[6] Dupont, G., Aït-Ameur, Y., Pantel, M., Singh, N.K. : Proof-based approach to hybrid systems development : Dynamic logic and event-b. In : Butler, M., Raschke, A., Hoang, T.S., Reichl, K. (eds.) Abstract State Machines, Alloy, B, TLA, VDM, and Z. pp. 155–170. Springer International Publishing, Cham (2018)

[7] Loos, S.M., Platzer, A. : Differential refinement logic. In : Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 505–514. LICS '16, ACM, New York, NY, USA (2016)

[8] Platzer, A. : Differential dynamic logic for hybrid systems. Journal of Automated Reasoning 41(2), 143–189 (2008)

[9] Quesel, J.D., Mitsch, S., Loos, S., Aréchiga, N., Platzer, A. : How to model and prove hybrid systems with keymaera : a tutorial on safety. International Journal on Software Tools for Technology Transfer 18(1), 67–91 (2016)

Une approche basée sur la séparation des préoccupations pour modéliser et vérifier les règles de signalisation d'un système ferroviaire

Yves Ledru[1,2], Akram Idani[1,2], Rahma Ben Ayed[2], Abderrahim Ait Wakrime[2], Philippe Bon[2,3]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France
{yves.ledru,akram.idani}@imag.fr
[2] Institut de Recherche Technologique Railenium, F-59300, Famars, France
{abderrahim.ait-wakrime,rahma.ben-ayed}@railenium.eu
[3] Univ Lille Nord de France, IFSTTAR, COSYS, ESTAS, 59666 Villeneuve d'Ascq Cedex, France
philippe.bon@ifsttar.fr

Cet article est un résumé étendu de l'article *"A separation of concerns approach for the verified modelling of railway signalling rules"* [9], accepté pour publication à la conférence RssRail 2019 (International Conference on Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification).

Les systèmes ferroviaires sont des systèmes critiques dont la sûreté a été étudiée depuis de nombreuses années. La sûreté résulte d'une combinaison de dispositifs physiques (freins, signaux,...), de règles (par exemple des règles de signalisation) et d'une coopération entre agents humains (conducteurs de trains, agent de circulation,...). Des technologies nouvelles sont déployées pour améliorer ces systèmes ferroviaires. Par exemple, GNSS (Global Navigation Satellite System) utilise le système GPS pour connaître la position du train. De nouvelles normes européennes, comme ERTMS/ETCS (European Rail Traffic Management System/European Train Control System) ont été proposées pour remplacer les systèmes de signalisation existants. Dans ce contexte, de nouvelles règles de signalisation doivent être conçues pour prendre en compte ces nouveaux équipements. Ces règles ont un caractère critique qui requiert des activités de vérification et de validation pour garantir leur sûreté.

Les méthodes formelles, et en particulier la méthode B [1], sont utilisées dans ce domaine depuis plus de 25 ans. L'utilisation de la méthode B pour des études de cas industrielles s'est soldée par plusieurs réussites remarquables [12], parmi lesquelles, le développement du métro Météor à Paris [3] ou la modernisation du métro de New York [13]. Au fil des années, une communauté scientifique s'est bâtie et a grandi autour de projets européens comme FMERail, ou de conférences comme RSSRail [7]. En 2018, la conférence ABZ [6], qui réunit notamment la communauté de la méthode B, a proposé à ses participants de traiter un cas d'étude qui modélisait ERTMS/ETCS level 3. Dans [2], nous avons utilisé une combinaison d'UML et d'Event-B pour cette étude de cas.

Dans [4, 5] nous avons proposé de structurer les spécifications B de règles de signalisation en nous inspirant de la description des systèmes d'information sécurisés en SecureUML [11]. Nous structurons notre modèle en un premier modèle qui décrit le comportement du système ferroviaire en l'absence de règles de circulation, c'est-à-dire

un système qui est essentiellement régi par les lois de la physique : les trains doivent suivre les rails et sont arrêtés en cas d'accident. Un deuxième modèle décrit les règles de signalisation qui régissent le comportement des trains et des agents. Dans ce modèle, les trains doivent respecter les signaux et on peut démontrer que les règles garantissent l'absence d'accidents. Ce modèle est proche de la définition d'une politique de contrôle d'accès car il accorde à des agents, humains ou logiciels, des permissions d'accéder aux objets du diagramme de classes.

Cette approche promeut une séparation des préoccupations qui traite d'une part les objets qui constituent le système et d'autre part les règles qui permettent aux agents de les manipuler. L'outil B4MSecure [8] est utilisé en support de cette approche. Il permet de traduire des diagrammes SecureUML en spécifications B. Nous utilisons également l'outil ProB [10] pour animer et valider la spécification [4] et le démonstrateur de l'atelier B pour prouver les obligations de preuve liées aux invariants qui garantissent la sûreté ferroviaire [5].

Dans [4], nous avons appliqué cette approche pour décrire les échanges d'informations entre un train et le centre de contrôle. Dans cet article, nous complétons cette approche sur les points suivants:

1. *Méthode formelle "légère"*. Notre approche passe par plusieurs étapes de vérification pour vérifier des propriétés d'atteignabilité, ou garantir la préservation d'invariant (absence d'accidents). Nous utilisons le model-checker ProB pour automatiser ces vérifications. Il permet de vérifier les propriétés d'atteignabilité de notre modèle en quelques minutes. Par contre, la vérification de la préservation d'invariant peut prendre plusieurs heures.
2. *Prise en compte des erreurs humaines.* Un système ferroviaire est piloté par des agents humains (conducteurs de train, agent de circulation). Dès lors, cela rend possible des erreurs humaines, par exemple quand un agent est fatigué. Ces erreurs humaines constituent des violations des règles. Dans cet article, nous prenons en compte ces comportements en les modélisant et en évaluant leurs conséquences. Il est également possible de modéliser des contre-mesures pour empêcher ces erreurs de se produire ou limiter leurs conséquences.
3. Nos travaux précédents [5] ont étudié les protocoles qui régissent les échanges entre le train et l'agent de circulation. Dans cet article, nous prenons en compte un système plus large qui comprend notamment la topologie des voies. De plus, nos activités de vérification font appel au model-checking, alors que dans nos travaux précédents, nous faisions appel à la preuve ou à l'animation.

Remerciements Ce travail est financé par le projet NExTRegio de l'IRT Railenium. Les auteurs remercient SNCF Réseau pour son soutien. Nous remercions également German Vega pour le support de B4MSecure.

References

1. Abrial, J.R.: The B-book: assigning programs to meanings. Cambridge University Press (1996)

2. Ait Wakrime, A., Ben Ayed, R., Collart Dutilleul, S., Ledru, Y., Idani, A.: Formalizing railway signaling system ERTMS/ETCS using UML/Event-B. In: MEDI 2018. pp. 321–330 (2018). https://doi.org/10.1007/978-3-030-00856-7_21
3. Behm, P., Benoit, P., Faivre, A., Meynadier, J.M.: Météor: A successful application of B in a large project. In: FM'99. LNCS, vol. 1708, pp. 369–387. Springer (1999). https://doi.org/10.1007/BFb0053352
4. Ben Ayed, R., Collart Dutilleul, S., Bon, P., Idani, A., Ledru, Y.: B formal validation of ERTMS/ETCS railway operating rules. In: ABZ 2014. LNCS, vol. 8477, pp. 124–129. Springer (2014). https://doi.org/10.1007/978-3-662-43652-3_10
5. Ben Ayed, R., Collart Dutilleul, S., Bon, P., Ledru, Y., Idani, A.: Formalismes basés sur les rôles pour la modélisation et la validation des règles d'exploitation ferroviaires. Technique et Science Informatiques **34**(5), 495–521 (2015). https://doi.org/10.3166/tsi.34.495-521
6. Butler, M.J., Raschke, A., Hoang, T.S., Reichl, K.: Abstract State Machines, Alloy, B, TLA, VDM, and Z - 6th Int. Conference, ABZ 2018, LNCS, vol. 10817. Springer (2018). https://doi.org/10.1007/978-3-319-91271-4
7. Fantechi, A., Lecomte, T., Romanovsky, A.: Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification - 2nd Int. Conf., RSSRail 2017, LNCS, vol. 10598. Springer (2017). https://doi.org/10.1007/978-3-319-68499-4
8. Idani, A., Ledru, Y.: B for modeling secure information systems - the B4MSecure platform. In: ICFEM 2015. LNCS, vol. 9407, pp. 312–318. Springer (2015). https://doi.org/10.1007/978-3-319-25423-4_20
9. Ledru, Y., Idani, A., Ben Ayed, R., Ait Wakrime, A., Bon, P.: A separation of concerns approach for the verified modelling of railway signalling rules. In: International Conference on Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification - RssRail 2019. Lille, France (Jun 2019), http://hal.univ-grenoble-alpes.fr/hal-02043174
10. Leuschel, M., Butler, M.J.: ProB: an automated analysis toolset for the B method. STTT **10**(2), 185–203 (2008). https://doi.org/10.1007/s10009-007-0063-9
11. Lodderstedt, T., Basin, D.A., Doser, J.: SecureUML: A UML-based modeling language for model-driven security. In: UML 2002, LNCS 2460. Springer (2002). https://doi.org/10.1007/3-540-45800-X_33
12. Sabatier, D.: Using formal proof and B method at system level for industrial projects. In: RSSRail 2016. LNCS, vol. 9707, pp. 20–31 (2016). https://doi.org/10.1007/978-3-319-33951-1_2
13. Sabatier, D., Burdy, L., Requet, A., Guéry, J.: Formal proofs for the NYCT line 7 (flushing) modernization project. In: ABZ 2012. LNCS, vol. 7316, pp. 369–372. Springer (2012). https://doi.org/10.1007/978-3-642-30885-7_34

Session commune au groupe de travail LTP et à AFADL

Langages, Types et Preuves — Approches Formelles dans l'Assistance au Développement de Logiciels

On est tranquilles pour longtemps – les reçus-temps en logique de séparation

François Pottier
francois.pottier@inria.fr

Au cours de cet exposé, je rappellerai d'abord brièvement comment la logique de séparation, étendue avec une notion simple de "crédit-temps", permet de vérifier à la fois la correction fonctionnelle et la complexité en temps d'un programme. Puis je présenterai une notion duale de "reçu-temps". Je montrerai qu'elle permet d'argumenter formellement que certains événements indésirables, comme le dépassement de capacité d'un compteur entier, ne peuvent pas survenir avant un temps très grand. En d'autres termes, certains programmes sont "sûrs pour longtemps".

Primitive Floats in Coq

Guillaume Bertholon Érik Martin-Dorel Pierre Roux

Abstract

Some mathematical proofs involve intensive computations, for instance: the four-color theorem, Hales' theorem on sphere packing (formerly known as the Kepler conjecture) or interval arithmetic. For numerical computations, floating-point arithmetic enjoys widespread usage thanks to its efficiency, despite the introduction of rounding errors.

Formal guaranties can be obtained on floating-point algorithms based on the IEEE 754 standard, which precisely specifies floating-point arithmetic and its rounding modes, and a proof assistant such as Coq, that enjoys efficient computation capabilities. Coq offers machine integers, however floating-point arithmetic still needed to be emulated using these integers.

A modified version of Coq is presented that enables using the machine floating-point operators. The main obstacles to such an implementation and its soundness are discussed. Benchmarks show potential performance gains from two to three orders of magnitude.

1 Motivation

The proof of some mathematical facts can involve a numerical computation in such a way that trusting the proof requires trusting the numerical computation itself. Thus, being able to efficiently perform this kind of proofs inside a proof assistant eventually means that the tool must offer efficient numerical computation capabilities.

Floating-point arithmetic is widely used in particular for its efficiency thanks to its hardware implementation. Although it does not generally give exact results, introducing rounding errors, rigorous proofs can still be obtained by bounding the accumulated errors. There is thus a clear interest in providing an efficient and sound access to the processor floating-point operators inside a proof assistant such as Coq.

1.1 Proofs Involving Numerical Computations

We give below a few examples of proofs involving floating-point computations.

As a first example, consider the proof that a given real number $a \in \mathbb{R}$ is nonnegative. One can exhibit another real number r such that $a = r^2$ and apply a lemma stating that all squares of real numbers are nonnegative. Typically, one could use the square root $\sqrt{a}$.

A similar method can be applied to prove that a matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite[1] as one can exhibit R such that[2] $A = R^T R$. Such a matrix can

[1] A matrix $A \in \mathbb{R}^{n \times n}$ is said positive semidefinite when for all $x \in \mathbb{R}^n$, $x^T A\, x \geq 0$.

[2] Since, when $A = R^T R$, one gets $x^T A\, x = x^T \left(R^T R\right) x = (Rx)^T (Rx) = \|Rx\|^2 \geq 0$.

be computed using an algorithm called Cholesky decomposition. The algorithm succeeds, taking neither square roots of negative numbers nor divisions by zero, whenever A is positive definite[3].

When executed with floating-point arithmetic, the exact equality $A = R^T R$ is lost but it remains possible to bound the accumulated rounding errors in the Cholesky decomposition such that the following theorem holds under mild conditions.

Theorem (Corollary 2.4 in [5]) *For $A \in R^{n\times n}$, defining $c := \frac{(n+1)\epsilon}{1-2(n+1)\epsilon}\,\mathrm{tr}(A) + 4n\,(2(n+1) + \max_i A_{i,i})\,\eta$, if the floating-point Cholesky decomposition succeeds on $A - c\,I$, then A is positive definite. ϵ and η are tiny constants given by the floating-point format used.*

A formal proof in Coq of this theorem can be found in a previous work [4]. Thus, an efficient implementation of floating-point arithmetic inside the proof assistant leads to efficient proofs of matrix positive definiteness. This can have multiple applications, such as proving that polynomials are nonnegative by expressing them as sums of squares [3] which can be used in a proof of the Kepler conjecture [1].

Interval arithmetic constitutes another example of proofs involving numerical computations. Sound enclosing intervals can be easily computed in floating-point arithmetic using directed roundings, towards $\pm\infty$ for lower or upper bounds. The Coq.Interval library [2] implements interval arithmetic and could benefit from efficient floating-point arithmetic.

More generally, there are many results on rigorous numerical methods [6] that could see efficient formal implementations provided efficient floating-point arithmetic is available inside proof assistants.

1.2 Objectives

The Coq proof assistant has built-in support for computation, which can be used within proofs, and recent progress have been done to provide efficient integer computation (relying on 63-bit machine integers).

The overall goal of this work is to implement efficient floating-point computation in Coq, relying directly on machine `binary64` floats, instead of emulating floats with pairs of integers. Experimentally, that latter emulation in Coq incurs a slowdown of about three orders of magnitude with respect to an equivalent implementation written in OCaml.

1.3 Outline

After providing some background on proof-by-reflection and floating-point arithmetic, this talk will focus on the implementation itself, with the interface that it exposes, several design choices or technicalities and some pitfalls to avoid. Finally some benchmarks will be presented.

The implementation is available in the `primitive-floats` branch of the following repository: `https://github.com/validsdp/coq/tree/primitive-floats`

[3] A matrix $A \in \mathbb{R}^{n\times n}$ is said positive definite when for all $x \in \mathbb{R}^n \setminus \{0\}$, $x^T A\,x > 0$.

References

[1] Victor Magron, Xavier Allamigeon, Stéphane Gaubert, and Benjamin Werner. Formal proofs for nonlinear optimization. *Journal of Formalized Reasoning*, 8(1):1–24, 2015.

[2] Érik Martin-Dorel and Guillaume Melquiond. Proving tight bounds on univariate expressions with elementary functions in Coq. *Journal of Automated Reasoning*, 57(3):187–217, October 2016.

[3] Érik Martin-Dorel and Pierre Roux. A reflexive tactic for polynomial positivity using numerical solvers and floating-point computations. In Yves Bertot and Viktor Vafeiadis, editors, *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2017, Paris, France, January 16-17, 2017*, pages 90–99. ACM, 2017.

[4] Pierre Roux. Formal Proofs of Rounding Error Bounds - With Application to an Automatic Positive Definiteness Check. *J. Autom. Reasoning*, 57(2):135–156, 2016.

[5] Siegfried M. Rump. Verification of positive definiteness. *BIT Numerical Mathematics*, 46:433–452, 2006.

[6] Siegfried M. Rump. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 19:287–449, 2010.

Formalisation en Coq des erreurs d'arrondi de méthodes de Runge-Kutta pour les systèmes matriciels

Florian Faissole

Inria, Université Paris-Saclay, F-91120 Palaiseau
LRI, CNRS & Univ. Paris-Sud, F-91405 Orsay
Email : florian.faissole@inria.fr

Résumé

Bien que les équations différentielles ordinaires soient omniprésentes dans la modélisation de systèmes physiques ou biologiques, leur résolution exacte est parfois fastidieuse voire impossible. L'utilisation de méthodes numériques, comme les méthodes de Runge-Kutta, permet d'obtenir des solutions approchées. Nous exhibons et formalisons en Coq des bornes sur les erreurs d'arrondi induites par l'implémentation en arithmétique à virgule flottante de méthodes de Runge-Kutta appliquées à des systèmes linéaires matriciels en tenant compte d'éventuels dépassements de capacité inférieurs.

1 Introduction

Les équations différentielles ordinaires modélisent de nombreux phénomènes physiques, biologiques ou économiques. Il n'existe cependant pas toujours de méthode directe pour les résoudre. Des méthodes numériques itératives ont été mises au point pour résoudre ces problèmes de façon approchée. L'essentiel des travaux scientifiques autour des méthodes numériques consiste à construire des méthodes relativement peu coûteuses permettant d'obtenir une approximation précise de la solution exacte et qui puissent s'appliquer à une classe importante de problèmes. Parmi ces méthodes de résolution, les méthodes de Runge-Kutta sont parmi les plus fréquemment utilisées (voir section 3). Les domaines d'application des équations différentielles étant parfois critiques (*e.g.* en aéronautique ou pour la modélisation de comportements robotiques), leur résolution est un objet d'étude intéressant pour des travaux de vérification formelle [13, 14, 5].

L'implémentation de ces méthodes en arithmétique à virgule flottante provoque des erreurs d'arrondi pouvant s'accumuler au cours des itérations. Ces erreurs étant dans la plupart des cas négligeables face aux erreurs de méthode, elles ont peu été étudiées par les numériciens. Des travaux ont néanmoins proposé des méthodes pour majorer les erreurs d'arrondi par des approches probabilistes tirant parti d'éventuelles compensations d'erreurs [10] ou en utilisant des techniques à base d'intervalles [8, 1] ou de modèles de Taylor [3, 17]. Cependant, les bornes exhibées ne tiennent compte ni des caractéristiques mathématiques détaillées des méthodes de Runge-Kutta, ni de la forme de l'implémentation choisie. Nous proposons une approche à grains fins (*i.e.* en décomposant l'analyse d'erreur au niveau des opérations élémentaires) qui tire parti de ces caractéristiques, ce qui rapproche nos travaux des résultats présentés par Fousse [9] pour les méthodes d'intégration numérique

ou par Boldo [4] qui borne et formalise en Coq les erreurs d'arrondi induites par un schéma de résolution de l'équation des ondes. Roux [18] propose une analyse d'erreur d'arrondi en Coq pour plusieurs algorithmes numériques impliquant des opérations matricielles, comme la décomposition de Cholesky. Cette formalisation repose sur une spécification de l'arithmétique à virgule flottante définie *via* un type `Record` et particulièrement adaptée à l'analyse d'erreur. Cette spécification est par ailleurs satisfaite par le format *binary64* de Flocq [7], une bibliothèque Coq d'arithmétique des ordinateurs.

Boldo, Faissole et Chapoutot [6] ont exhibé (sans garantie formelle) des bornes sur les erreurs d'arrondi induites par l'implémentation de méthodes de Runge-Kutta appliquées à des systèmes linéaires unidimensionnels (systèmes de la forme $\dot{y} = \lambda y$ où $\lambda \in \mathbb{R}$). Dans cet article, nous généralisons cette approche au cas des systèmes linéaires multidimensionnels, où λ est remplacé par une matrice carrée A à coefficients réels. Les opérations matricielles étant plus complexes que les opérations sur les nombres réels, les résultats prennent en compte un nombre plus important de paramètres, à commencer par la dimension du système. Nous proposons de plus une formalisation de ces résultats dans l'assistant de preuves Coq [1] en combinant la bibliothèque Flocq [7] et les matrices de la bibliothèque MathComp [15]. Nous tenons compte d'éventuels dépassements graduels de capacité inférieurs (*underflows*), dont la contribution impacte les bornes d'erreurs exhibées. En revanche, nous ne tenons pas compte des dépassements de capacité supérieurs (*overflows*). Notre approche diffère légèrement de celle adoptée par Roux [18] car nous utilisons directement la formalisation des formats de nombres flottants de la bibliothèque Flocq, sans passer par une spécification dédiée de l'arithmétique à virgule flottante.

La section 2 présente quelques rappels d'arithmétique à virgule flottante. La section 3 est dédiée à la présentation du problème et de notre méthodologie (basée sur la distinction entre erreurs locales et globales). Dans la section 4, nous présentons les éléments de base de la formalisation Coq. Dans les sections 5 et 6, nous bornons respectivement les erreurs locales et globales des méthodes de Runge-Kutta avant de conclure en section 7.

2 Prérequis d'arithmétique à virgule flottante

2.1 Formats de représentation des nombres flottants

Les formats de représentation des nombres à virgule flottante ainsi que les opérations sur ces nombres sont régis par la norme IEEE-754 [12, §3]. En analyse d'erreur, une définition relativement haut niveau (sans décrire la représentation bit à bit des nombres flottants) est suffisante. Un format flottant $\mathbb{F}$ est représenté par un tuple $(\beta, p, e_{\min}, e_{\max})$ où l'entier $\beta \geqslant 2$ est la base, $p \in \mathbb{N}$ est la précision, $e_{\min} \in \mathbb{Z}$ et $e_{\max} \in \mathbb{Z}$ sont les valeurs minimales et maximales de l'exposant. Un nombre flottant dans $\mathbb{F}$ est soit une valeur exceptionnelle parmi $+\infty$, $-\infty$ ou NaN, soit une valeur égale à $\pm d_0.d_1 \cdots d_{p-1} \times \beta^e$ (avec d_i des chiffres dans la base β) et tel que $e_{\min} \leq e \leq e_{\max}$. Dans ce qui suit, $\beta = 2$.

La Figure 1 présente la répartition des nombres flottants sur l'axe réel. Les nombres flottants dits normalisés vérifient $d_0 = 1$ et $e_{\min} \leq e \leq e_{\max}$. Le plus petit nombre flottant normalisé positif est $\xi = 2^{e_{\min}}$ et le plus grand nombre flottant normalisé est $\Omega = (2 - 2^{1-p})2^{e_{\max}}$. Lorsqu'un nombre flottant est plus petit que ξ, il est dit dénormalisé et on parle de dépassement de capacité inférieur (*underflow*). Nous avons alors $e = e_{\min}$ et $d_0 = 0$. On note $\eta = 2^{e_{\min}-p+1}$ le plus petit nombre flottant dénormalisé positif. Lorsque le résultat

1. Les fichiers sont disponibles en ligne : `https://www.lri.fr/~faissole/CoqRK`

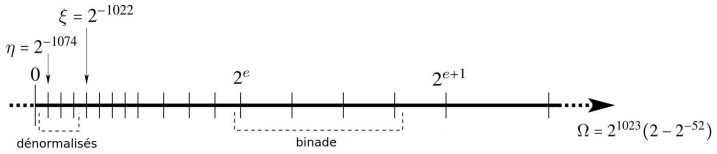


FIGURE 1 – Représentation des nombres flottants (format *binary64*) sur l'axe réel

d'une opération flottante n'est pas exactement représentable dans le format considéré, il est nécessaire d'arrondir le résultat vers un nombre flottant. Plusieurs modes d'arrondi sont définis dans la norme IEEE-754 [12], dont l'arrondi au plus proche. Lorsque le résultat est le point-milieu entre deux flottants consécutifs, une régle de bris d'égalité (*tie-breaking rule*) permet de choisir vers lequel arrondir, par défaut le nombre flottant pair est choisi.

Dans cet article, nous considérons un format flottant $\mathbb{F}$ prenant en compte les *underflows* (sauf mention du contraire). Nous considérons un mode d'arrondi au plus proche avec une règle de bris d'égalité quelconque, noté $\circ$. Nous noterons $\oplus$, $\ominus$, $\otimes$ et $\oslash$ les opérations arrondies correspondant aux opérations usuelles $+, -, \times$ et $/$, $\circ[\cdots]$ signifie que toutes les opérations à l'intérieur des crochets sont arrondies. Pour une même opération, le parenthésage à droite est implicite en l'absence de parenthèses, ainsi $\circ[ab + c] = (a \otimes b) \oplus c$ et $\circ[a + b + c] = a \oplus (b \oplus c)$. La convention s'étend aux opérations entre matrices et vecteurs. En particulier, pour $v \in \mathbb{R}^n$ un vecteur de taille n et $A, B \in \mathbb{R}^{n\times n}$ des matrices carrées de taille n, $\circ[A \times v]$ et $\circ[A \times B]$ font respectivement référence au produit de A par v et au produit de A par B en évaluant les sommations de la droite vers la gauche.

2.2 Fondements de l'analyse d'erreurs d'arrondi

Les analyses d'erreur que nous présentons sont basées sur quelques définitions et résultats fondamentaux (voir Higham pour une présentation détaillée de ces résultats [11, §3]). Nous commençons par présenter un résultat bornant l'erreur relative commise lors de l'arrondi d'un nombre réel de valeur absolue supérieure à ξ : si $x \in \mathbb{R}$ est tel que $\xi \leqslant |x|$, alors $\frac{|\circ(x)-x|}{|x|} \leqslant u = \frac{1}{2}\beta^{1-p}$. La quantité u, fondamentale en analyse d'erreur, est appelée unité d'arrondi. Par exemple, dans le format double précision *binary64*, $u = 2^{-53}$. Lorsque le nombre réel x est plus petit que ξ, l'erreur relative commise en arrondissant x peut devenir très grande, mais nous pouvons prouver que l'erreur absolue est bornée par $\frac{\eta}{2}$. Nous pouvons en déduire ce que Higham appelle modèle standard de l'arithmétique à virgule flottante [11, §2.2.], *i.e.* pour $x, y \in \mathbb{F}$, $\Diamond \in \{+, -, \times, /\}$, il existe $\delta, \varepsilon \in \mathbb{R}$ tels que $\circ(x \Diamond y) = (x \Diamond y)(1 + \varepsilon) + \delta$, $|\varepsilon| \leqslant u$, $|\delta| \leqslant \frac{\eta}{2}$ et $\varepsilon\delta = 0$. Si $\Diamond \in \{+, -\}, \delta = 0$.

Nous utiliserons régulièrement la notation $\gamma_d = \frac{du}{1-du}$ ($d \in \mathbb{N}$, $du < 1$) [11, §3.1.].

2.3 Flocq : bibliothèque d'arithmétique des ordinateurs en Coq

Pour formaliser nos résultats, nous utilisons Flocq, une bibliothèque Coq d'arithmétique des ordinateurs développée par Boldo et Melquiond [7] comportant une représentation abstraite des nombres flottants : les nombres dans un format donné sont des sous-ensembles des nombres réels `R` de la bibliothèque standard de la forme $m \cdot \beta^e$ où m et e sont des entiers. Par exemple, le format `FLX` correspond aux nombres flottants sans bornes sur les exposants : seule la précision p est prise en compte, avec la condition $|m| < \beta^p$. Le format `FLT` prend en compte les dépassements graduels de capacité inférieurs et est paramétré par

p et $e_{\min}$, les nombres flottants dans ce format vérifiant $|m| < \beta^p$ et $e \geq e_{\min}$. L'ensemble des résultats présentés dans cet article sont valides pour le format `FLX`. Nous avons ensuite étendu l'ensemble des résultats génériques pour le format `FLT`, *i.e.* l'ensemble des résultats à l'exception de l'instanciation aux méthodes d'Euler et de RK2.

3 Présentation du problème et méthodologie

Les méthodes de Runge-Kutta sont itératives. Elles consistent à discrétiser un intervalle de temps $[0, t_N]$ en un ensemble de points $t_0, \ldots, t_N$ et à construire itérativement les solutions approchées $y_0, \ldots, y_N$ en ces points. Ce sont des méthodes explicites à un pas constant h : la solution au temps $t_{n+1} = t_n + h$ est obtenue à partir de la solution au temps t_n.

Les systèmes linéaires multidimensionnels sont parmi les plus fréquemment rencontrés dans les domaines liés à des problèmes physiques. Les équations différentielles associées sont de la forme $\dot{y} = Ay \quad$ avec $\quad y \in \mathbb{R}^d \quad$ et $\quad A \in \mathbb{R}^{d\times d}$. L'application d'une méthode de Runge-Kutta explicite sur ce système linéaire conduit à une relation de récurrence de la forme $y_{n+1} = R(hA)y_n$ avec R un polynôme en hA (de la forme $\sum_i \alpha_i (hA)^i$, $\alpha_i \in \mathbb{R}$). Prenons l'exemple de deux méthodes classiques, les méthodes d'Euler et de RK2. Les polynômes associés à ces méthodes sont $R_{\text{Euler}}(hA) = (I + hA)$ et $R_{\text{RK2}}(hA) = (I + hA + 0{,}5h^2A^2)$. Le polynôme R est appelé fonction de stabilité linéaire pour les méthodes de Runge-Kutta car, dans le cas des systèmes linéaires, la condition $|||R(hA)|||_\infty < 1$ caractérise une classe de couples système-méthode dit stables (voir section 4.1 pour une définition de la norme vectorielle $\|\cdot\|_\infty$ et de sa norme matricielle subordonnée $|||\cdot|||_\infty$). La fonction R est purement mathématique et ne caractérise pas l'implémentation des méthodes en arithmétique à virgule flottante, qui suppose le choix d'un algorithme, noté $\widetilde{R}$ et qui dépend de trois paramètres : le pas h (considéré comme représentable dans le format flottant, contrairement à ce qui est fait dans [6]), une matrice $\widetilde{A}$ correspondant à la matrice "arrondie de A" (matrice obtenue après arrondi de chaque coefficient de A) ainsi que la solution $\widetilde{y_n}$ calculée à l'itération précédente (lors du calcul de $\widetilde{y_{n+1}}$). Ainsi, une implémentation de la méthode de Runge-Kutta est de la forme : $\widetilde{y_0} = \circ(y_0)$ et pour tout n, $\widetilde{y_{n+1}} = \widetilde{R}(h, \widetilde{A}, \widetilde{y_n})$. À un même polynôme R correspondent plusieurs implémentations $\widetilde{R}$ possibles [2]. Dans cet article, nous nous intéressons à une évaluation "à la Horner" des méthodes numériques (décrite ci-dessous pour Euler et RK2 respectivement) :

$$\widetilde{y_{n+1}} = \circ\left[\widetilde{y_n} + (h\widetilde{A})\widetilde{y_n}\right], \qquad \widetilde{y_{n+1}} = \circ\left[\widetilde{y_n} + (h\widetilde{A})\left(\widetilde{y_n} + (\tfrac{h}{2}\widetilde{A})\widetilde{y_n}\right)\right].$$

Nous souhaitons borner l'erreur d'arrondi globale induite par l'implémentation d'une méthode de Runge-Kutta après n itérations, *i.e.* $E_n = \|\widetilde{y_n} - y_n\|_\infty$. Afin d'y parvenir, nous commençons par étudier les erreurs dites locales. L'erreur locale commise à l'itération n ne quantifie que les erreurs commises par les calculs de l'itération courante et peut être définie par $\varepsilon_0 = \|\widetilde{y_0} - y_0\|_\infty$ (que nous supposons connue) et pour tout $n \in \mathbb{N}, \quad \varepsilon_{n+1} = \|\widetilde{R}(h, \widetilde{A}, \widetilde{y_n}) - R(hA)\widetilde{y_n}\|_\infty$. Si les résultats ont du être adaptés, la méthodologie est similaire à celle utilisée dans le cas unidimensionnel [6]. Elle consiste dans un premier temps à construire une borne sur les erreurs locales relatives par applications mécaniques et consécutives du Lemme 1 (voir section 5). Ensuite, cette borne sur les erreurs locales est utilisée pour borner l'erreur globale par application directe du Théorème 2 (voir section 6).

2. En effet, les opérations flottantes ne sont pas associatives et dépendent de l'ordre d'évaluation. Par ailleurs, les calculs peuvent être décomposés, factorisés, etc.

4 Éléments de base de la formalisation

4.1 Vecteurs et matrices

Les systèmes étudiés font intervenir des vecteurs et des matrices à coefficients réels. Nous utilisons essentiellement des vecteurs colonnes de taille d (dont le type Coq est noté `cV_d`) et des matrices carrées de taille d (de type `M_d`).

Nous procédons à une analyse par normes des erreurs d'arrondi, ce type d'analyse reposant sur des normes sous-multiplicatives [11, §6]. Les normes vectorielles $\|\cdot\|_1$ et $\|\cdot\|_\infty$ sont particulièrement adaptées à l'analyse d'erreurs et les normes matricielles subordonnées à ces normes sont faciles à exprimer en fonction des coefficients de la matrice. Par ailleurs, nous nous basons sur des résultats de Higham qui sont valables pour ces deux normes [11]. Nous travaillons ici avec la norme $\|\cdot\|_\infty$, définie comme $\|v\|_\infty = \max_i |v_i|$ pour $v \in \mathbb{R}^d$ et dont la norme subordonnée est définie par $|||A|||_\infty = \max_i \sum_j |A_{i,j}|$ pour $A \in \mathbb{R}^{d\times d}$. Ces normes sont formalisées à l'aide de grands opérateurs (*bigops*) de MathComp [2], un mécanisme permettant d'itérer des opérations comme l'addition ou le maximum :

```
Definition vec_norm {d} : cV_d → R := fun v ⇒ \big[Rmax / 0]_(i < d) Rabs (v i).
Definition mat_norm {d} : M_d → R :=
                fun A ⇒ \big[Rmax / 0]_(i < d) (\big[Rplus / 0]_(j < d) Rabs (A i j)).
```

Nous prouvons qu'il s'agit bien de normes et que $|||\cdot|||_\infty$ est sous-multiplicative, *i.e.* pour $v \in \mathbb{R}^d, A \in \mathbb{R}^{d\times d}$, $\|Av\|_\infty \leqslant |||A|||_\infty \|v\|_\infty$ (`mx_vec_norm_submult` en Coq) et pour $A, B \in \mathbb{R}^{d\times d}$, $|||AB|||_\infty \leqslant |||A|||_\infty |||B|||_\infty$ (`mx_norm_submult` en Coq). La sous-multiplicativité permet de démontrer des bornes sur les erreurs d'arrondi des produits matrice-vecteur (`mx_vec_prod_error`) et matrice-matrice (`mx_prod_error`) :

$$\|\circ[A\times v] - Av\|_\infty \leqslant \gamma_d |||A|||_\infty \|v\|_\infty + \frac{d}{2}(1+\gamma_{d-1})\eta.$$
$$|||\circ[A\times B] - AB|||_\infty \leqslant \gamma_d |||A|||_\infty |||B|||_\infty + \frac{d^2}{2}(1+\gamma_{d-1})\eta.$$

Nous prouvons ces propriétés pour des sommations évaluées de droite à gauche. Ces bornes restent cependant vraies quelque soit l'ordre d'évaluation [11, §3.5.].

4.2 Méthodes de Runge-Kutta

Nous définissons un type `Sc` (pour Schéma), dont les éléments définissent des relations de récurrence entre les vecteurs y_n et y_{n+1} et caractérisent de ce fait l'application des méthodes de Runge-Kutta à des systèmes multidimensionnels (de dimension d) :

```
Definition Sc (d : nat) : Type := (R → R) → cV_d → cV_d.
```

Sur la donnée d'une fonction $\mathcal{W} \in \mathbb{R} \to \mathbb{R}$ (par exemple, il peut s'agir d'un arrondi ou de la fonction identité) et d'un vecteur $y_n \in \mathbb{R}^d$, un élément de type `Sc` renvoie un vecteur $y_{n+1} \in \mathbb{R}^d$. On peut évaluer l'application de la méthode numérique `M : Sc` sur n itérations à partir de la condition initiale `y0` (`y0_tilde` étant le vecteur `y0` dont toutes les composantes ont été "arrondies" par la fonction $\mathcal{W}$) :

```
Definition meth_iter (M:Sc) n (y0 : cV_d) (W : R → R) := iter n (M W) y0_tilde
```

Nous définissons formellement les erreurs locales et globales (en valeur absolue) :

```
Definition error_loc (M : Sc) n (y0 : cV_d) (W : R → R) (⋆ε_{n+1} = ‖R̃(h, Ã, ỹ_n) − R(hA)ỹ_n‖_∞ ⋆)
  := vec_norm (M W (meth_iter M n y0 W) − M (fun x ⇒ x) (meth_iter M n y0 W)).
Definition error_glob (M : Sc) n (y0 : cV_d) (W : R → R)                    (⋆E_n = ‖ỹ_n − y_n‖_∞ ⋆)
  := vec_norm (meth_iter M n y0 W − meth_iter M n y0 (fun x ⇒ x)).
```

5 Erreurs locales

Nous démontrons un résultat générique pour borner l'erreur d'arrondi locale commise lors d'une itération de la méthode. Ce résultat permet de traiter toute méthode à un pas explicite, d'ordre quelconque, implémentée suivant une évaluation "à la Horner" (voir Section 3)

Lemme 1. *Soit* $d \in \mathbb{N}^*$, $y \in \mathbb{R}^n$, $C_1, C_2, C_3, D_1, D_3 \in \mathbb{R}_+$, $A_1, A_2, A_3 \in \mathbb{R}^{n\times n}$, $X_1, X_3 \in \mathbb{F}^d$, $\widetilde{A_2} \in \mathbb{F}^{n\times n}$. *Soit* $\rho = C_2 u|||A_3|||_\infty + C_3 u|||A_2|||_\infty + C_2 C_3 u^2$.
Soit $\mathscr{C} = C_1 u + \rho + u\,(|||A_1|||_\infty + C_1 u) + (u + (1+u)\gamma_d)(\rho + |||A_2|||_\infty |||A_3|||_\infty)$.
Soit $\mathscr{D} = (1+u)\left(\frac{d}{2}(1+\gamma_{d-1}) + D_1 + D_3(1+\gamma_d)(C_2 + |||A_2|||_\infty)\right)$.
Supposons que :
- $\|X_1 - A_1 y\|_\infty \leqslant C_1 u \|y\|_\infty + D_1 \eta$.
- $|||\widetilde{A_2} - A_2|||_\infty \leqslant C_2 u$.
- $\|X_3 - A_3 y\|_\infty \leqslant C_3 u \|y\|_\infty + D_3 \eta$.

Alors $\|X_1 \oplus (\widetilde{A_2} \otimes X_3) - (A_1 + A_2 A_3) y\|_\infty \leqslant \mathscr{C}\|y\|_\infty + \mathscr{D}\eta$.

En Coq, le lemme correspondant au Lemme 1 est nommé `build_bound_mult_loc`. La démonstration de ce résultat repose sur une analyse d'erreur en avant relativement naïve. La quantité $X_1 \oplus (\widetilde{A_2} \otimes X_3)$ correspond à un pas de l'évaluation "à la Horner" en arithmétique à virgule flottante (c'est-à-dire $\widetilde{R}(h, \widetilde{A}, \widetilde{y_n})$) et $(A_1 + A_2 A_3)y$ correspond à l'évaluation mathématique $R(hA)\widetilde{y_n}$. Ces évaluations étant construites à partir d'évaluations de Horner plus simples, on peut reconstruire pas à pas une borne sur l'erreur locale de l'expression entière. En l'absence de dépassement de capacité inférieur, $\mathscr{D} = 0$.

Prenons l'exemple de la méthode RK2 (sans prise en compte de l'*underflow*). On instancie le Lemme 1 avec $y = \widetilde{y_n}$, $X_1 = \widetilde{y_n}$, $A_1 = I$, $\widetilde{A_2} = \circ\left[h\widetilde{A}\right]$, $A_2 = hA$, $X_3 = \circ\left[\widetilde{y_n} + \frac{h}{2}\widetilde{A}\widetilde{y_n}\right]$ et $A_3 = I + \frac{hA}{2}$. On peut vérifier que l'expression $\|X_1 \oplus (\widetilde{A_2} \otimes X_3) - (A_1 + A_2 A_3)y\|_\infty$ correspond à l'erreur locale ε_{n+1} pour la méthode RK2. Il faut alors exhiber C_1, C_2 et C_3. Il apparaît de façon évidente que $C_1 = 0$ car $\|X_1 - A_1 y\|_\infty = \|\widetilde{y_n} - I\widetilde{y_n}\|_\infty = 0$. Pour exhiber C_2, on borne $|||h\widetilde{A} - hA|||_\infty$, ce qui est assez naturel par dépliage de la définition de $|||\cdot|||_\infty$ et en appliquant des résultats génériques d'analyse d'erreurs d'arrondi. Enfin, il reste à exhiber C_3, *i.e.* à borner $\|X_3 - A_3 y\|_\infty$. Pour y parvenir, on réapplique le Lemme 1 avec $y = X_1 = X_3 = \widetilde{y_n}$, $A_1 = A_3 = I$, $A_2 = \frac{hA}{2}$ et $\widetilde{A_2} = \circ\left[\frac{h}{2}\widetilde{A}\right]$. Il est trivial d'exhiber C_1 et C_3, C_2 étant obtenu de la même manière qu'à l'étape précédente.

Nous avons utilisé cette méthodologie pour borner les erreurs locales des méthodes d'Euler et de RK2 en *binary64* (en négligeant les *underflows*). Pour la méthode d'Euler :

$$\forall n \in \mathbb{N}^*, \varepsilon_n \leqslant (u + (u + 3{,}12\gamma_d)h|||A|||_\infty)\,\|y_{n-1}\|_\infty. \tag{1}$$

Le lemme Coq associé est nommé `Euler_loc_FLX`. Pour la méthode RK2 :

$$\forall n \in \mathbb{N}^*, \varepsilon_n \leqslant \left(u + (11{,}3u + 2{,}56\gamma_d)(h|||A|||_\infty + h^2|||A|||_\infty^2)\right)\|y_{n-1}\|_\infty. \tag{2}$$

Le lemme Coq associé est nommé `RK2_loc_FLX`. La méthodologie peut paraître fastidieuse mais il est néanmoins possible de partiellement l'automatiser. Tout d'abord, à chaque sous-application du Lemme 1, nous utilisons la tactique `eapply` de Coq, qui va permettre d'inférer automatiquement les valeurs des paramètres y, X_1, A_1, $\widetilde{A_2}$, A_2, X_3 et A_3. Il ne reste alors plus qu'à instancier manuellement les valeurs respectives de C_1, C_2 et C_3 puis à prouver les 3 hypothèses du lemme. Les bornes obtenues s'avèrent relativement lisibles (voir Équations (1) et (2)) et ne font pas apparaître de termes d'ordre supérieur (en u^2, u^3, etc). Nous avons en effet majoré ces termes à chaque étape de la construction de la borne d'erreur en utilisant la tactique automatique `interval` [16], ce qui nécessite des hypothèses sur u : en *binary64*, on a $u^2 \leqslant 2^{-53}u$ mais des hypothèses plus faibles suffisent, *e.g.* $u^2 \leqslant 0{,}01u$.

6 Erreurs globales

Dans de précédents travaux [6], un théorème général permet de construire de façon systématique une borne sur l'erreur globale d'une méthode à partir des bornes exhibées sur les erreurs locales précédentes. Nous généralisons ce résultat au cas multidimensionnel :

Théorème 2. Passage des erreurs locales à l'erreur globale
Soit $C, D \geqslant 0$. *Supposons que pour tout* $n \in \mathbb{N}^*, \varepsilon_n \leqslant C\|\widetilde{y_{n-1}}\|_\infty + D\eta$ *et que* $0 < C + |||R(hA)|||_\infty < 1$ *(condition de stabilité). Alors :*

$$\forall n, E_n \leqslant (C + |||R(hA)|||_\infty)^n \left(\varepsilon_0 + \frac{nC\|y_0\|_\infty}{C + |||R(hA)|||_\infty}\right) + nD\eta.$$

En Coq, le théorème correspondant au Théorème 2 est nommé `error_loc_to_glob`. Si l'énoncé du théorème est très proche de celui démontré dans [6], la démonstration l'est également. En effet, le choix de mener une analyse par normes a permis de généraliser le résultat appliqué aux systèmes scalaires unidimensionnels pour qu'il s'étende aux systèmes linéaires multidimensionnels, *i.e.* matriciels, en remplaçant les valeurs absolues par des normes. À partir des résultats exhibés dans la section 5, nous bornons l'erreur globale correspondant aux méthodes d'Euler et de Runge-Kutta d'ordre 2 par application directe du Théorème 2. Il suffit en effet de remplacer la variable C par la valeur correspondante dans le Théorème 2, *e.g.* $u + (u + 3{,}12\gamma_d)h|||A|||_\infty$ pour la méthode d'Euler.

7 Conclusion et perspectives

Nous avons proposé une méthodologie générique pour borner les erreurs d'arrondi associées à une certaine classe de méthodes numériques en tenant compte des dépassements de capacité inférieurs. Par rapport à nos travaux antérieurs [6], nous proposons une généralisation aux systèmes multidimensionnels *via* une analyse par normes. Cette généralisation est non-triviale car elle repose sur des résultats fondamentaux d'analyse matricielle, mais la méthodologie appliquée reste similaire. Nous instancions nos résultats à deux méthodes classiques (Euler et RK2) et exhibons des bornes qui, dans le cas de méthodes stables, sont quasi-linéaires en le nombre d'itérations. Par ailleurs, nous formalisons l'ensemble du raisonnement dans Coq. La formalisation des résultats génériques représente environ 3300 lignes de Coq en `FLX` et 3600 lignes en `FLT`. L'instanciation aux méthodes d'Euler et de RK2 représente environ 1200 lignes de Coq.

Nous envisageons de factoriser une partie des résultats commune aux formats `FLX` et `FLT`. Flocq contient en effet des résultats liant ces deux formats [7]. De plus, bien que nous nous basions sur la bibliothèque Mathematical Components, la formalisation n'utilise pas les tactiques propres à `ssreflect`, qui permettraient probablement de réduire la taille des démonstrations. Nous pourrions également combiner nos preuves à des définitions et résultats formalisés par Roux dans de précédents travaux [18], comme la notation γ (et ses propriétés), ainsi que certains résultats fondamentaux, *e.g.* une borne sur l'erreur d'arrondi du produit scalaire de deux vecteurs. Une autre perspective pour ce travail est l'extension de la méthodologie à d'autres classes de méthodes, comme les méthodes implicites et multi-pas, ou à d'autres classes de systèmes, comme les systèmes affines ou non-linéaires. Une autre perspective envisagée est la combinaison des erreurs d'arrondi et des erreurs de méthode. Pour évaluer la finesse de la borne obtenue, nous pourrions, comme dans le cas unidimensionnel [6], comparer numériquement la borne d'erreur à l'erreur d'arrondi effectivement commise. Enfin, nous envisageons de vérifier que la borne d'erreur d'arrondi obtenue est négligeable par rapport à l'erreur de méthode effectivement commise.

Références

[1] J. Alexandre dit SANDRETTO et A. CHAPOUTOT : Validated explicit and implicit Runge-Kutta methods. *Reliable Computing*, 22, 2016.

[2] Y. BERTOT, G. GONTHIER, S. O. BIHA et I. PASCA : Canonical big operators. *In International Conference on Theorem Proving in Higher Order Logics*, pages 86–101. Springer, 2008.

[3] M. BERZ et K. MAKINO : Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. *Reliable Computing*, 4(4):361–369, 1998.

[4] S. BOLDO : Floats & Ropes : a case study for formal numerical program verification. *In 36th International Colloquium on Automata, Languages and Programming*, volume 5556 de *LNCS - ARCoSS*, pages 91–102, Rhodos, Greece, juillet 2009. Springer.

[5] S. BOLDO, F. CLÉMENT, J.-C. FILLIÂTRE, M. MAYERO, G. MELQUIOND et P. WEIS : Wave Equation Numerical Resolution : a Comprehensive Mechanized Proof of a C Program. *Journal of Automated Reasoning*, 50(4):423–456, avril 2013.

[6] S. BOLDO, F. FAISSOLE et A. CHAPOUTOT : Round-off Error Analysis of Explicit One-Step Numerical Integration Methods. *In 24th IEEE Symposium on Computer Arithmetic*, pages 82–89, London, United Kingdom, juillet 2017.

[7] S. BOLDO et G. MELQUIOND : *Computer Arithmetic and Formal Proofs*. ISTE Press - Elsevier, décembre 2017.

[8] O. BOUISSOU et M. MARTEL : GRKLib : a guaranteed Runge-Kutta library. *In International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics*, 2006.

[9] L. FOUSSE : Multiple-precision correctly rounded Newton-Cotes quadrature. *ITA*, 41(1):103–121, 2007.

[10] P. HENRICI : *Error propagation for difference methods*. The SIAM series in applied mathematics. John Wiley, 1963.

[11] N. J. HIGHAM : *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd édition, 2002.

[12] IEEE standard for floating-point arithmetic. *IEEE Std 754-2008*, Aug 2008.

[13] F. IMMLER et J. HÖLZL : Numerical analysis of ordinary differential equations in Isabelle/HOL. *In Interactive Theorem Proving*, volume 7406 de *LNCS*, pages 377–392. Springer Berlin Heidelberg, 2012.

[14] F. IMMLER et C. TRAUT : The flow of ODEs : Formalization of Variational Equation and Poincaré Map. *Journal of Automated Reasoning*, 62(2):215–236, Feb 2019.

[15] A. MAHBOUBI et E. TASSI : Mathematical Components. `https://math-comp.github.io/mcb/`, 2017.

[16] G. MELQUIOND : Proving bounds on real-valued functions with computations. *In International Joint Conference on Automated Reasoning, IJCAR 2008*, Sydney, Australia.

[17] M. NEHER, K. R. JACKSON et N. S. NEDIALKOV : On Taylor model based integration of ODEs. *SIAM Journal on Numerical Analysis*, 45(1):236–262, 2007.

[18] P. ROUX : Formal Proofs of Rounding Error Bounds - With Application to an Automatic Positive Definiteness Check. *J. Autom. Reasoning*, 57(2):135–156, 2016.

Session commune au groupe de travail MTV^2 et à AFADL

Méthodes de test pour la validation et la vérification — Approches Formelles dans l'Assistance au Développement de Logiciels

Test d'un robot agricole en simulation

Clément Robert[1], Thierry Sotiropoulos[1], Hélène Waeselynck[1], Jérémie Guiochet [1], and Simon Vernhes[2]

[1]LAAS-CNRS, Université de Toulouse, Toulouse
[1]Naïo Technologies, Toulouse

Résumé

Ce document est le résumé étendu d'un article en cours de soumission à une revue. Le titre de l'article d'origine est : "The virtual lands of Oz : testing an agribot in simulation".

Mots-clés – Etude de cas industrielle, test du logiciel, simulation, génération de mondes, système autonome, robot agricole

Les missions d'un robot autonome sont typiquement testées par des expérimentations sur le terrain. Cette approche est très coûteuse et peut présenter des risques. Afin d'explorer plus de situations opérationnelles à moindre coût et sans encourir de risque, nos travaux développent des tests en simulation : le robot accomplit ses missions dans des mondes virtuels. Cet article présente une étude de cas industrielle sur la faisabilité et l'efficacité d'une telle approche.

Le système étudié est Oz, un robot agricole de désherbage développé par la société Naïo Technologies (voir la figure 1). Son logiciel a été testé dans des champs de légumes virtuels, en utilisant un simulateur 3D basé sur Gazebo. L'étude de cas a permis de se confronter à plusieurs défis : la génération aléatoire d'environnements 3D complexes, la vérification automatisée du comportement observé (oracle de test), et la fidélité imparfaite de la simulation par rapport à un test dans le monde réel. Nous présentons l'approche de test en simulation que nous avons développée, et comparons les résultats avec ceux d'expérimentations sur le terrain.

Les tests en simulation sont générés à partir d'un modèle de mondes qui comprend : (i) une vue structurée des éléments à générer (les rangées de légumes, le terrain 3D, ...) sous forme d'un diagramme UML, (ii) des contraintes sur les valeurs des paramètres de génération de ces éléments, sous forme d'une grammaire attribuée. La génération procède en deux temps. On produit d'abord une configuration valide de valeurs de paramètres (i.e., un mot de la grammaire), puis on utilise ces valeurs pour produire le contenu des mondes, dans un format directement compréhensible par le simulateur.

(a) Experimentation sur le terrain (b) Test dans un champ virtuel

FIGURE 1 – Le robot Oz en action

L'oracle de test est constitué d'un ensemble de détecteurs, chacun ciblant une propriété spécifique. On cherche à détecter des comportements anormaux tels que des collisions, le franchissement d'un seuil de vitesse, des erreurs de perception, ou encore des comportements inappropriés à la phase de mission en cours.

Les résultats de l'étude montrent le fort potentiel du test en simulation pour alléger les expérimentations coûteuses. En effet, les tests en simulation s'avèrent efficaces en dépit d'une reproduction basse fidélité de la physique du robot. Ils révèlent la plupart des fautes trouvées par les expérimentations dans le monde réel, y compris la faute ayant causé la majorité des défaillances sur le terrain. Ils révèlent également une nouvelle faute qui n'avait pas été mise en évidence jusque-là. Par contre, la simulation peut introduire des défaillances parasites qui ne se produiraient pas dans le monde réel.

Tests de politiques d'adaptation pour systèmes cyber-physiques

Jean-Philippe Gros
Institut FEMTO-ST, Univ. Bourgogne Franche-Comté, CNRS
15B avenue des Montboucons, 25030 Besançon, Cedex, France

Abstract

Cet article est dédié à la validation des politiques d'adaptation en utilisant une approche de tests à partir de modèles; à la mise en place d'un verdict basé à la fois sur la vérification à l'exécution et aux propriétés temporelles; à la détection d'inconsistances entre les politiques d'adaptation et les reconfigurations implémentées dans le système. Nous proposons un moyen d'établir un verdict de tests basé sur le respect des politiques d'adaptation ainsi que les mesures de couverture des règles. Ces verdicts nous donnent des informations sur les règles pour détecter d'éventuelles reconfigurations qui n'auraient pas dû avoir lieu, des règles avec une priorité haute qui ne sont jamais déclenchées ou des règles avec une priorité faible qui sont déclenchées trop souvent, des inconsistances dans les règles, ou une mauvaise interprétation des priorités. Les verdicts sont obtenus en analysant les traces d'exécution du système qui est stimulé par un modèle d'usage à transitions probabilistes. Afin d'illustrer notre approche, nous utilisons un système de peloton de voitures autonomes.

1 Introduction

Les systèmes adaptatifs gagnent en importance au fil des années, on les retrouve dans les véhicules intelligents, les infrastructures adaptatives (e.g smartgrids), les robots et de manière générale dans l'industrie. Ces systèmes s'adaptent en fonction des évènements internes et externes qu'ils rencontrent par le biais de reconfigurations dynamiques. Ces reconfigurations dynamiques changent l'architecture de systèmes adaptatifs [3] et sont guidées par des politiques d'adaptation. Une politique d'adaptation a pour rôle de guider le système en indiquant la pertinence de déclencher ses reconfigurations. Une politique d'adaptation est constituée de reconfigurations et d'un ensemble de règles de déclenchement des reconfigurations. Chaque règle étant composée de gardes sur la configuration du système, de priorités à l'activation et de propriétés temporelles. Les priorités, exprimées par de valeurs floues comme dans [4] indiquent la pertinence d'appliquer une reconfiguration.

Lors du développement d'un système adaptatif, il est possible de mal interpréter les choix spécifiés par les politiques d'adaptation. Actuellement, des méthodes de vérification [9] existent afin d'établir qu'un système se comporte correctement du point de vue des propriétés (temporelles) du système. Cependant, il n'existe pas de garantie sur le respect des politiques d'adaptation. Plus précisément, nous souhaitons nous assurer que l'ensemble des politiques d'adaptation ne présente pas d'incohérence. Ensuite, il faut s'assurer que ces politiques d'adaptation soient valides en terme de cohérence et de respect des propriétés fonctionnelles. Dans une autre mesure, il est possible de se convaincre de l'efficacité du système en validant des propriétés non-fonctionnelles. La validation des propriétés non-fonctionnelles consisterait à évaluer les choix des reconfigurations faites en rejouant la même séquence d'évènements. Le fait est que les propriétés et les règles de politiques d'adaptation

ne peuvent évaluer les reconfigurations du système à la volée mais plutôt a posteriori. Afin d'établir de tels verdicts, il est nécessaire d'établir une technique de validation de conformité pour exercer le système à produire des traces pertinentes.

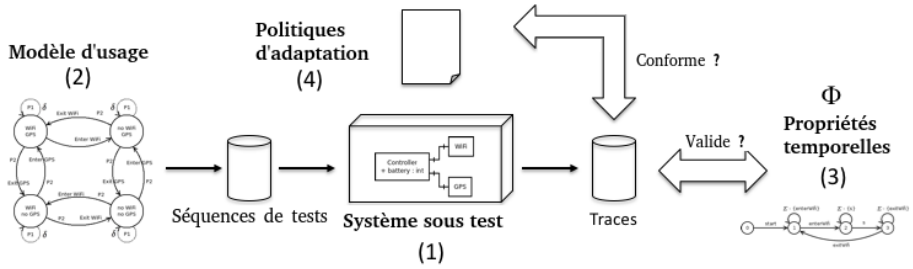


Figure 1: Approche par tests à partir de modèles pour valider les politiques d'adaptation

Afin de satisfaire ces problématiques, nous proposons une approche à partir de modèles dans la Figure. 1, qui se base sur les méthodes proposées dans [1]. Nous proposons de générer des cas de tests tout en établissant une méthode de verdict de test permettant de s'assurer du bon comportement du système sous test.

Les travaux [9] répondent à une première partie de la problématique en vérifiant la validité des propriétés temporelles en analysant rétroactivement les traces produites par le système (3). À partir de ces travaux, nous souhaitons nous assurer que les traces produites par système sous test (1) couvrent suffisamment les comportements possibles du système. Pour cela, nous proposons des critères de couverture qui s'appuient sur les propriétés temporelles (3) et les politiques d'adaptation (4). Les traces utilisées visant à satisfaire les critères de couverture sont obtenues en stimulant le système à l'aide d'un modèle d'usage (2). Ce modèle d'usage génère des séquences d'évènements (ou cas de tests) grâce à un automate probabiliste. Les traces d'exécution sont alors analysées : les propriétés temporelles (3), sont vérifiées pendant l'exécution du système sous test (1). Cette étape s'effectue en utilisant une technique d'analyse du système basée sur les travaux de [7]. Les politiques d'adaptation (4) sont utilisées pour s'assurer que le système effectue les reconfigurations au bon moment. Pour cela, nous vérifions que les reconfigurations souhaitées sont bien déclenchées et qu'aucune reconfiguration non souhaitée n'est déclenchée. Nous nous intéressons également à la cohérence de la politique d'adaptation, pour cela, en se basant sur les critères de couverture, il nous est possible de détecter qu'une règle n'est jamais satisfaite. En outre de ces aspects de cohérence, nous proposons un verdict sur le respect des priorités. Ce verdict permet de détecter lorsqu'une reconfiguration avec une priorité haute n'est pas ou peu déclenchée, ou qu'une reconfiguration avec une faible priorité est trop souvent déclenchée.

Cet article est organisé de la façon suivante : La Section 2 présente le contexte scientifique et définit les notions et notations utilisées dans ce document. Dans la Section 3, nous décrivons notre approche de tests à partir de modèles ainsi que les critères de couverture associés. Nous conclurons et présenterons les futurs axes de recherche de ce travail dans la Section 4.

2 Contexte

Cette section présente le contexte dans lequel nous avons mené nos études. Celles-ci se sont portées sur un système de convoi de véhicules autonomes. Le comportement de ce système est vérifié par des propriétés temporelles et le déclenchement des reconfiguration guidé par des politiques d'adaptation.

2.1 Le convoi de véhicules autonomes

Dans ce système, les véhicules sont organisés en peloton ou individuellement. Dans chaque peloton, on trouve un leader qui se situe à l'avant. Un véhicule seul peut demander à rejoindre un peloton ou créer un nouveau peloton en se groupant avec un autre véhicule seul. Le système est soumis à des évènements internes, en effet, chaque véhicule au sein d'un peloton peut demander à sortir soit parce qu'il a atteint sa destination soit parce qu'il arrive à court d'énergie. Le véhicule désigné leader peut changer soit parce qu'un véhicule a plus d'autonomie que lui soit parce que sa destination est plus éloignée que le véhicule leader actuel. Des évènements externes peuvent avoir lieu, un conducteur peut décider de reprendre la main et quitter le convoi ou un nouveau véhicule peut arriver à portée du peloton pour une éventuelle fusion.

Nous considérons deux séquences différentes comme illustré dans la Figure 2, la séquence d'évènements externes (ou cas de tests) générée par le modèle d'usage et le chemin de reconfiguration (ou traces d'exécution) généré par le système en réponse aux évènements externes. Les séquences et chemins de reconfiguration sont régis par un tic d'horloge, cela signifie que les évènements et les reconfigurations sont échantillonnés selon la même fréquence. Dans le cas d'un convoi de véhicules autonomes, plusieurs évènements peuvent avoir lieu. L'évènement *join* intervient quand des véhicules sont à portée pour se rejoindre, le système peut alors faire rejoindre plusieurs véhicules *acceptJoin* ou ne rien faire *refuseJoin*. Lorsqu'un utilisateur décide volontairement de quitter le peloton, l'évènement *quit* intervient, le système réagit à cet évènement par un (*acceptQuit*). Le système peut également réagir à des évènements internes et choisir de changer de leader avec la reconfiguration (*getRelay*). Si aucune reconfiguration n'a eu lieu entre deux tics d'horloge, on observe une reconfiguration *run* sur le chemin de reconfiguration.

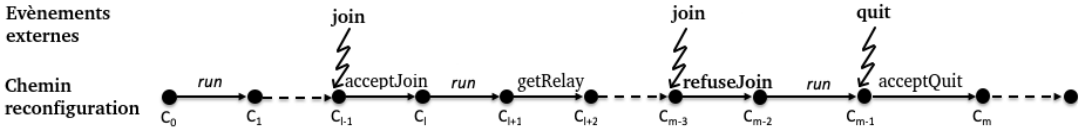


Figure 2: Séquence d'évènements et chemin de reconfiguration

2.2 Propriétés Temporelles

Dans cette section, nous présentons les propriétés temporelles FTPL[1] introduites dans [5]. Le langage FTPL est instancié avec des évènements du système pour exprimer une propriété temporelle. La sémantique de FTPL que nous utilisons se base sur les travaux de [6]. Nous proposons deux exemples liés à notre cas de peloton de véhicules autonomes :

$\varphi 1$: **after** *CreatePlt* **before** *DeletePlt* **always** *PlatoonId.VehicleNb* > 2
Cette propriété s'applique pour chaque convoi de véhicules. Elle vérifie qu'après la création d'un convoi il y a toujours au moins deux véhicules jusqu'à la suppression du peloton.

$\varphi 2$: **after** *Join* **before** *Quit* **always** *VehicleId.Battery* > 5 and *VehicleId.Distance* > 2
Cette propriété s'applique pour chaque *VehicleId*. Elle s'assure qu'après l'entrée du véhicule dans un convoi, les niveaux de batterie et distance restante du véhicule sont supérieurs à respectivement 5% et 2km, jusqu'à la sortie du véhicule.

[1]FTPL vient de la fusion entre TPL (Temporal Pattern Language) et le préfixe 'F' comme 'First order logic'.

Comme nous le montrent ces exemples, une propriété FTPL est constituée entre autres de mots clés temporels (**after, before, always, eventually**), d'évènements (*CreatePl*, *DeletePl*) et de propriétés de configuration (*Distance* > *20*, *Battery* > *5*, *VehicleNb* > *2*). Ces propriétés s'inscrivent dans le contexte global de la route qui inclut les voitures et leurs convois distingués grâce aux identifieurs (*VehicleId* et *PlatoonId*). Ces propriétés assurent à l'exécution que le système se comporte correctement. Toutefois, il est possible que certaines règles soient évaluées à potentiellement vrai (resp. potentiellement faux) dans le cas d'un **always** (resp. **eventually**) qui est évalué à vrai (resp. faux) jusqu'à un instant t mais qui demande d'être évalué à vrai jusqu'à (resp. vrai au moins une fois avant) un instant t'.
Les propriétés sont utilisées pour exprimer des exigences sur le système mais elles seront aussi dans la définition des politiques d'adaptation que nous présentons maintenant.

2.3 Politiques d'adaptation

Des politiques d'adaptation ont été définies dans [2, 4, 6], nous les avons redéfinies afin de les adapter à notre modèle. Dans notre approche, les opérations de reconfiguration sont gardées par des séquences spécifiques d'évènements.
Reprenons notre exemple de convoi de véhicules autonomes, afin de garantir la propriété $\varphi 2$, nous devons réfléchir aux reconfigurations à effectuer. C'est le rôle des politiques d'adaptation et c'est au niveau de leurs règles de reconfiguration que ces reconfigurations sont décidées. Deux exemples de règles de reconfiguration sont données en Figure 3.

```
when (after Join before Quit and VehicleId.battery < 33)
   if (state = leader ) then
        utility of PassRelay is high

when (after Join before Quit and VehicleId.battery > Leader.battery)
   if (state = platooned ) then
        utility of GetRelay is medium
```

Figure 3: Deux règles de reconfiguration du convoi de véhicules autonomes

Ces règles de reconfiguration stipulent, dans un premier temps, la propriété FTPL à valider, elle est décrite après l'utilisation du mot-clé `when` qui délimite l'intervalle d'évènements durant lequel la reconfiguration peut avoir lieu. Dans un second temps, les règles de reconfiguration contiennent une garde sur la configuration du système, qui est décrite après l'utilisation du mot-clé `if` qui définit la configuration courante du système qui peut entraîner une reconfiguration. Pour finir, le nom de la reconfiguration R_N et son utilité F est renseignée sous forme de valeur floue (`high, medium, low`) après le mot clé *utility* afin d'associer une utilité avec un nom d'opération de reconfiguration.
Ces deux exemples s'appliquent aux véhicules et servent à déterminer les moments de passage de relais entre le véhicule leader et un autre véhicule du convoi. Dans le premier cas, la reconfiguration `PassRelay` se déclenche lorsque le leader n'a plus assez de batterie et dans ce cas il doit être rétrogradé. Dans le second cas, la reconfiguration `GetRelay` se déclenche lorsque l'autonomie du véhicule courant est supérieure à celle du leader.
L'implémentation des politiques d'adaptation et plus particulièrement les règles de reconfiguration sont laissées à la discrétion des développeurs qui sont susceptibles de mal interpréter la spécification. Une mauvaise implémentation peut mener le système à se reconfigurer lorsque ce n'est pas nécessaire, ne pas se reconfigurer ou choisir la mauvaise reconfiguration et engendrer une violation de propriété. Le processus de tests décrit dans la prochaine section a pour but de répondre à ce problème en validant que les politiques d'adaptation sont correctement écrites et retranscrites à l'exécution du système.

3 Validation de l'implémentation vis à vis des politiques d'adaptation

Dans cette section, nous présentons les contributions apportées lors de la thèse dont l'ensemble est résumé Fig. 1. Les politiques d'adaptation influencent le système, il est donc nécessaire de s'assurer que le système implémente correctement ces politiques d'adaptation. Cela consiste à vérifier de manière indépendante que le système se reconfigure avec les bonnes reconfigurations au bon moment. Le système doit également toujours répondre aux propriétés fonctionnelles. Nous proposons également un moyen de nous assurer de la conformité des priorités avec la spécification.

Une des difficultés de cette approche est que les vérifications de propriétés sont faites a posteriori. En effet, les propriétés temporelles doivent atteindre leur état final pour être évaluées. De la même façon, les reconfigurations déclenchées par le système guidées par les politiques de reconfiguration peuvent mener à une erreur mais après un certain délai. Pour répondre à ces problématiques, nous proposons une analyse à l'exécution des traces produites par le système. Afin de garantir la pertinence de ces traces, nous élaborons des critères de couverture garantissant que le système a bien exploré les propriétés et les politiques d'adaptation. Tout d'abord, nous présentons une méthode de génération de tests à partir d'un modèle d'usage indépendant en simulant l'environnement extérieur au système. Ce modèle a pour but de produire les traces pertinentes pour les critères de couverture définis. Pour finir, nous expliciterons les verdicts de test mis en place et plus précisément les verdicts sur le respect des politiques d'adaptation.

3.1 Critères de couverture

Nos critères de couverture pour les propriétés FTPL s'appuient sur la définition d'un automate associé à ses propriétés.

Definition 1 (Automate de propriétes FTPL)**.** *Soit un automate noté A_φ composé du quadruplet $\langle Q, q_0, Q_f, T\rangle$ avec Q l'ensemble des états, $q_0 \in Q$ est l'état initial, $Q_f \subset Q$ est l'ensemble des états finaux, et $T : Q \times Ev_\varphi \to Q$ est la fonction de transition avec Ev_φ l'ensemble des évènements possibles dans la propriété.*

Nous allons nous appuyer sur les exemples de la Sect. 2.2 et particuliérement la propriété :

after *Join* **before** *Quit* **always** *VehicleId.Battery* > 5 and *VehicleId.Distance* > 2
La création de l'automate visible Fig. 4 est déterminée par les mots-clés temporels et leurs évènements associés. Notre automate est donc constitué des transitions *join* et *quit*. ϵ représente l'ensemble des transitions possibles. L'état final est atteint lorsque le scénario de la propriété a été complété, dans notre exemple lorsqu'un véhicule est entré et sorti du convoi.

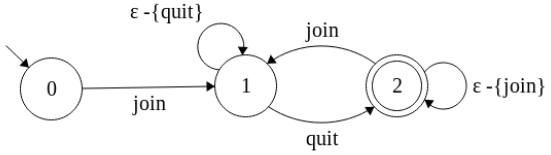


Figure 4: Automate de la propriété φ_1

Definition 2 (Couverture des propriétés FTPL)**.** *Soit une propriété φ et A_φ son automate associé. Un état de l'automate A_φ est dit* activé *si tous ses couples de transitions entrantes et sortantes ont été*

parcourus par le système. Une propriété est alors dite couverte *lorsque tous les états de son automate associé ont été activés.*

Dans notre exemple Fig. 4, pour que A_φ soit *activé*, les trois couples de transitions $0 \xrightarrow{join} 1 \xrightarrow{Quit} 2$, $1 \xrightarrow{Quit} 2 \xrightarrow{Join} 1$ et $2 \xrightarrow{Join} 1 \xrightarrow{Quit} 2$ doivent être sensibilisés.

Definition 3 (Couverture des politiques d'adaptation)**.** *Une règle de politique d'adaptation est dite* activable *si sa garde (partie* `if`*) et sa propriété de déclenchement (partie* `when`*) sont évaluées à vrai et si l'opération de reconfiguration associée est exécutée par le système. Une politique d'adaptation est dite* couverte *lorsque toutes ses règles ont été activées.*

Maintenant que les critères de couverture ont été définis, nous présentons le modèle d'usage permettant de les satisfaire.

3.2 Modèle d'usage et génération de tests

Le modèle d'usage a pour but de produire des séquences d'évènements externes qui exercent le système. Les systèmes adaptatifs réagissent d'une part aux évènements externes et d'autre part aux évolutions de leurs variables internes en accord avec les politiques d'adaptation qui guident à quel moment déclencher les reconfigurations. En d'autres termes, le système se comporte à la manière d'une boîte noire ce qui rend la définition de son modèle impossible. C'est pourquoi nous considérons un modèle d'usage du système sous test. Ce modèle d'usage ne va pas représenter le comportement du système mais simuler l'environnement dans lequel le système évolue. Ce type de modèle a pour but de spécifier les différents évènements qui peuvent avoir lieu dans l'environnement. Pour rappel, nous considérons les systèmes régis par des tics d'horloge, et donc, en plus des évènements externes, nous définissons la notion de *délai*. Cela se traduit par une absence d'évènement externe durant une certaine période, mais de son côté, le système continue d'évoluer et de mettre à jour son état interne. L'automate dans la Figure 5 représentant le modèle d'usage est défini de la même façon qu'un automate de propriété temporelle mais à une différence : la notion de *délai* est présente dans l'automate du modèle alors que l'automate de propriété ne garde que les évènements qui agissent sur sa propriété. L'automate génère pendant l'exécution (en ligne) ce qui permet à l'automate d'être connecté au système. En effet, en plus d'envoyer les séquences d'évènements externes au système, l'automate peut observer les évènements internes du système. Ces concepts sont illustrés Figure 5.

L'exemple des voitures autonomes utilise trois évènements externes marqués par des transitions en

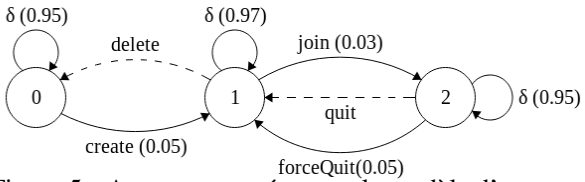


Figure 5: Automate représentant le modèle d'usage

lignes pleines : (*create*) crée un véhicule en l'ajoutant sur la route, (*join*) est déclenché lorsque le véhicule est à portée d'un ou plusieurs véhicules et (*forceQuit*) est déclenché lorsqu'un véhicule veut sortir de son convoi. Les évènements marqués par des transitions en tiret sont des évènements internes observables par le modèle d'usage, ils servent à garantir la cohérence du modèle. En effet, l'évènement (*delete*) est déclenché lorsqu'un véhicule arrive à destination et ne peut avoir lieu avant la création de ce véhicule. De la même façon, un véhicule peut quitter le convoi avec l'évènement

(*quit*) suite à une reconfiguration déclenchée par le système, dans ce cas l'automate d'usage peut à nouveau déclencher l'évènement (*join*).
A noter que l'automate est discret, et donc le *délai* représente une unité de temps. Dans ces conditions, pour générer une séquence d'évènements, nous utilisons un algorithme de type 'Markov random walk' [8] qui consiste à laisser la séquence être générée par parcours aléatoire de l'automate probabiliste. Le processus de génération continue de générer des évènements jusqu'à atteindre les critères de couverture. Avec ces cas de tests, le système est stimulé et peut déclencher des changements internes ou des reconfigurations dans le système. Le système produit une trace de reconfiguration qui est analysée selon le respect des politiques d'adaptation et propriétés FTPL. Cette analyse, les métriques et leurs verdicts sont décrits maintenant.

3.3 Verdicts de tests

L'établissement d'un verdict de test s'appuie sur deux artefacts. La première étape consiste à vérifier les propriétés FTPL. La seconde étape s'occupe de vérifier les conditions d'exécution des reconfigurations. Ce travail se base sur les travaux [7] sur la vérification des propriétés FTPL.

Verdict sur les propriétés FTPL Intuitivement, une propriété FTPL *passe le verdict* pour toute trace de longueur *i*, si l'état à l'indice *i* est l'état final de la propriété FTPL, que la propriété est vérifiée et que tout état à un indice supérieur à *i* ne sera pas dans l'état final de l'automate de la propriété. La propriété est *violée* s'il existe un état ne respectant pas la propriété. Enfin, le verdict est *inconclusif* si aucun état ne parvient à l'état final de ladite propriété.

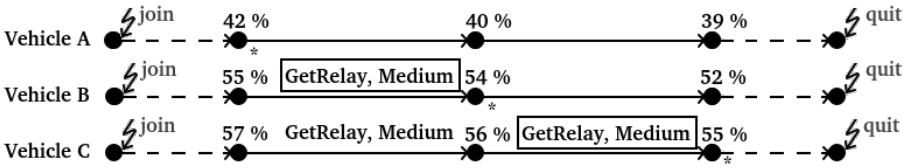


Figure 6: Séquence de reconfigurations observables

Nous définissons à partir des exemples présentés dans la Secion 2.3 notre deuxième partie de verdict basé cette fois sur les politiques d'adaptation en essayant de détecter les reconfigurations non souhaitées.
La Figure 6 montre un exemple de traces produites par trois véhicules au sein d'un même convoi. Les évènement externes *join* et *quit* sont indiqués de la même façon que dans la Figure 2, les reconfigurations internes (*GetRelay*) sont associées à une priorité (*Medium*). À chaque état, le niveau d'énergie restante est indiqué par un pourcentage et le signe $*$ signifie que le véhicule est *leader*. Entre chaque état, les reconfigurations possibles ainsi que leur priorité sont affichées. La reconfiguration choisie par le système est encadrée.
On appelle *eligible* une règle de reconfiguration qui est activable. On appelle *actual* lorsque la reconfiguraton est activée. Le but du verdict étant de vérifier que la reconfiguration *actual* est présente dans la liste des reconfigurations éligibles. Dans notre exemple, GetRelay est *eligible* pour les véhicules B et C donc, on s'attend à voir une telle reconfiguration être activée, contrairement à PassRelay définie Figure 3 qui déclencherait une erreur si elle était activée dans ce cas de figure.
En complément de ce verdict, nous proposons des métriques sur les règles de reconfiguration : $\#elig$ est donné par le nombre de fois qu'elle a été désignée *eligible* et $\#actual$ est donné par le nombre de fois qu'une règle est désignée *actual*. Ces métriques nous permettent d'effectuer des ratios permettant

de comparer le taux de déclenchement. Si une règle à priorité élevée possède un taux de déclenchement plus faible qu'une règle à priorité basse c'est qu'il existe potentiellement une incohérence dans le déclenchement des reconfigurations ou la définition des politiques d'adaptation.

4 Conclusion et axes de recherche

Ce papier a présenté une approche de tests à partir de modèles qui a pour but de vérifier qu'un système adaptatif implémente fidèlement les politiques d'adaptation spécifiées indépendamment. Cette approche se base sur un modèle d'usage générant des séquences de tests permettant de diriger le système sous test. La validation du système repose sur plusieurs verdicts : un sur le respect des propriétés temporelles et l'autre sur l'évaluation de la légitimité des reconfigurations. Étant donné que nos mesures et vérifications sont effectuées à l'exécution, la génération des séquences de tests peut être effectuée avant ou pendant l'exécution. Cette approche a été implémentée sur un programme Java modélisant le convoi de véhicules autonomes décrit dans ce papier. Les résultats obtenus valident, pour cet exemple l'approche décrite. Les travaux futurs se dirigent vers une validation des propriétés non fonctionnelles du système en analysant quels paramètres peuvent être modifiés pour mieux répondre à des questions d'efficacité. Nous préparons des expérimentations sur d'autres cas d'étude.

References

[1] B. Beizer. *Black-box Testing: Techniques for Functional Testing of Software and Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[2] F. Chauvel, O. Barais, N. Plouzeau, I. Borne, and J.-M. Jézéquel. Expression qualitative de politiques d'adaptation pour Fractal. In Y. Aït Ameur, editor, *2ème Conf. sur les Architectures Logicielles (CAL 2008), 3-7 Mars 2008, Montréal, Québec, Canada*, volume RNTI-L-2 of *Revue des Nouvelles Technologies de l'Information*, page 119. Cépaduès-Éditions, 2008.

[3] R. De Lemos, H. Giese, H.A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N.M. Villegas, T. Vogel, et al. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32. Springer, 2013.

[4] J. Dormoy and O. Kouchnarenko. Event-based adaptation policies for Fractal components. In *AICCSA 2010, ACS/IEEE Int. Conf. on Computer Systems and Applications*, pages 1–8, Hammamet, Tunisia, may 2010.

[5] J. Dormoy, O. Kouchnarenko, and A. Lanoix. Using temporal logic for dynamic reconfigurations of components. In L. Barbosa and M. Lumpe, editors, *FACS*, volume 6921 of *LNCS*, pages 200–217. Springer Berlin Heidelberg, 2012.

[6] O. Kouchnarenko and J.-F. Weber. Adapting component-based systems at runtime via policies with temporal patterns. In J. L. Fiadeiro, Z. Liu, and J. Xue, editors, *FACS, 10th Int. Symp. on Formal Aspects of Component Software*, volume 8348 of *LNCS*, pages 234–253. Springer, 2014.

[7] O. Kouchnarenko and J.-F. Weber. Decentralised evaluation of temporal patterns over component-based systems at runtime. In I. Lanese and E. Madelaine, editors, *Formal Aspects of Component Software*, volume 8997 of *LNCS*, pages 108 – 126, Bertinoro, Italy, sep 2015. Springer.

[8] A. Sinclair. *Algorithms for Random Generation and Counting: A Markov Chain Approach*. Birkhauser Verlag, Basel, Switzerland, Switzerland, 1993.

[9] Jean-François Weber. *Guider et contrôler les reconfigurations de systèmes à composants: Reconfigurations dynamiques: modélisation formelle et validation automatique. (Guide and control component systems-based system reconfigurations: Dynamic reconfigurations: formal modelling and automatic validation)*. PhD thesis, University of Franche-Comté, Besançon, France, 2017.

Towards a Test-and-Proof Framework for C11 in Isabelle/HOL

Burkhart Wolff

Univ. Paris-Sud, Laboratoire LRI, UMR8623, Orsay, F-91405, France
Burkhart.Wolff@lri.fr

Abstract

We report on an integration of a novel C11 Frontend into Isabelle/HOL enabling different semantic backends (AutoCorres, Securify, IMP2, Clean, ...). We discuss the challenges of a Generic Framework ranging from IDE to Proof-Support, and show how a small semantic backend can be hooked into our Framework enabling both deductive program verification as well as program-based Test-Generation.

Prix de thèse et Accessits du GDR Génie de la Programmation et du Logiciel

Prix de thèse du GDR GPL
Méthodes et outils pour la spécification et la preuve de propriétés difficiles de programmes séquentiels

Auteur : Martin Clochard (Université Paris-Saclay, Inria, équipe TOCCATA)

Résumé :
Cette thèse se positionne dans le domaine de la vérification déductive de programmes, qui consiste à transformer une propriété à vérifier sur un programme en un énoncé logique, pour ensuite démontrer cet énoncé. La vérification effective d'un programme peut poser de nombreuses difficultés pratiques. En fait, les concepts mis en jeu derrière le programme peuvent suffire à faire obstacle à la vérification. En effet, certains programmes peuvent être assez courts et n'utiliser que des constructions simples, et pourtant s'avérer très difficiles à vérifier.

Cela nous amène à la question suivante : dans le contexte d'un environnement de vérification déductive de programmes basé sur les démonstrateurs automatiques, quelles méthodes appliquer pour réduire l'effort nécessaire à la fois pour spécifier des comportements attendus complexes, ainsi que pour démontrer qu'un programme respecte ces comportements attendus ?

Pour mener notre étude, nous nous sommes placés dans le cadre de l'environnement de vérification déductive de programmes Why3. La vérification de programmes en Why3 est basée sur la génération de conditions de vérification, et l'usage de démonstrateurs externes pour les prouver, que ces démonstrateurs soient automatiques ou interactifs. Nous avons développé plusieurs méthodes, certaines générales et d'autres spécifiques à des classes de programmes, pour réduire l'effort manuel. Nos contributions sont les suivantes. Tout d'abord, nous ajoutons des fonctionnalités à Why3 pour assister le processus de vérification, notamment un mécanisme léger de preuve déclarative basé sur la notion d'indicateurs de coupures. Ensuite, nous présentons une méthode de vérification d'absence de débordement arithmétique pour une classe d'utilisation des entiers difficile à traiter par les méthodes standards. Enfin, nous nous intéressons au développement d'une bibliothèque générique pour la spécification et la preuve de programmes générateurs de code.

Biographie :
Martin Clochard est chercheur post-doctoral à L'ETH Zürich, au sein du groupe "programming methodology". Il travaille sur le développement de méthodes de vérification pour le langage Go, un langage concurrent avec des mécanismes de synchronisation bloquants, en utilisant des méthodes basées sur la logique de séparation. Il a effetué sa thèse sous la direction de Claude Marché et Andrei Paskevich à l'université Paris-Saclay, au sein de l'équipe VALS du LRI.

Accessit
From Runtime Failures to Patches : Study of Patch Generation in Production

Auteur : Thomas Durieux (Inria Lille, Université de Lille)

Résumé :
Les logiciels font dorénavant entièrement partie de notre quotidien et les bugs en font naturellement tout autant. Les développeurs et entreprises consacrent un temps considérable pour essayer des les éradiquer. La création d'un correction prend énormément de temps, non seulement parce qu'il est difficile de créer un bon correctif, mais également parce que cela nécessite l'intervention d'humains. En effet, Un utilisateur doit signaler le bug et le développeur doit le reproduire et le corriger, c'est un processus long et fastidieux. La réparation automatique de bugs est un nouveau champ de recherche qui consiste à proposer automatiquement un correctif et minimiser l'intervention d'un développeur et d'accélérer le déploiement des correctifs en production.

Cependant, les techniques de réparation automatique actuelles exigent toujours une intervention d'un développeur. En effet, ces techniques utilisent les tests comme spécification du logiciel : un test qui échoue, spécifie le bug dans l'application et les tests qui réussissent, spécifient le bon comportement du logiciel. Il faut donc toujours un développeur pour reproduire le bug et le transformer en un test qui l'expose. Cette exigence qui réduit considérablement l'applicabilité de ces techniques.

Dans le cadre de cette thèse, nous proposons une nouvelle approche de génération de correctifs qui s'affranchit de cette exigence. Elle repose sur l'idée de rapprocher la génération automatique de correctifs de l'environnement de production. En effet c'est celui-ci qui contient toutes les données et toutes les interactions humaines qui mènent aux bugs. Dans cette thèse, nous présentons comment exploiter ces données pour détecter les bugs, générer les correctifs et les valider, directement dans l'environnement de production et ce sans l'intervention d'un développeur.

Biographie :
Thomas Durieux est un chercheur postdoctoral à l'institut du génie systèmes et logiciels (INESC-ID) à Lisbonne (Portugal). Il travaille sur la réparation et la localisation automatique des bugs dans les logiciels. Il a effectué sa thèse avec Martin Monperrus (KTH), Lionel Seinturier (Université de Lille) et Youssef Hamadi (Microsoft Research) au sein de l'équipe Spirals d'Inria Lille.

Démonstrations et Posters

SysML/KAOS:
A Formal Requirements Engineering Method

Steve Tueno[1,2], Marc Frappier[1], Régine Laleau[2], Amel Mammar[3]

[1]GRIL – Université de Sherbrooke, Canada
[2]LACL – Université Paris Est Créteil Val de Marne, France
[3]SAMOVAR-CNRS – Télécom SudParis, France

Presentation

SysML/KAOS includes:

1. ***Goal*** [1, 2, 3] and ***Domain*** [4, 5] **Modeling**
2. ***Formal Model* Generation** [6, 7, 8, 9]
3. ***Behavior* Modeling** and **Formal *Verification* and *Validation*** [10, 11, 12]
4. **Back Propagation of *Updates*** [13, 8, 9]

Experiments

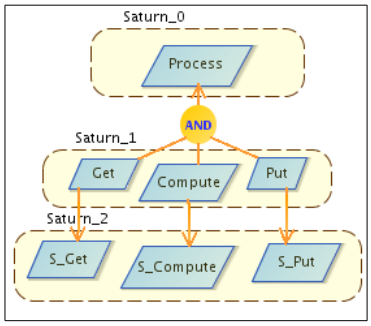


Figure 1: ***Goal Modeling***

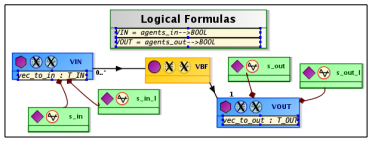


Figure 2: ***Domain Modeling*** [Saturn_2]

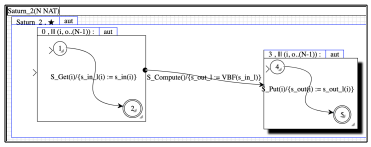


Figure 3: ***Behavior Modeling (ASTD)*** [Saturn_2]

The SysML/KAOS Requirements Engineering Method

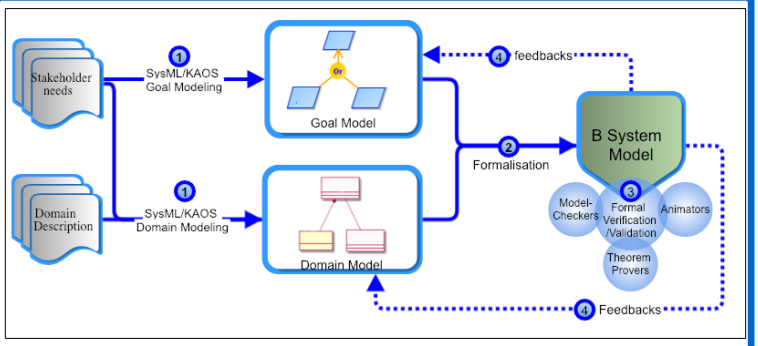


Figure 4: Overview of the specification process [13]

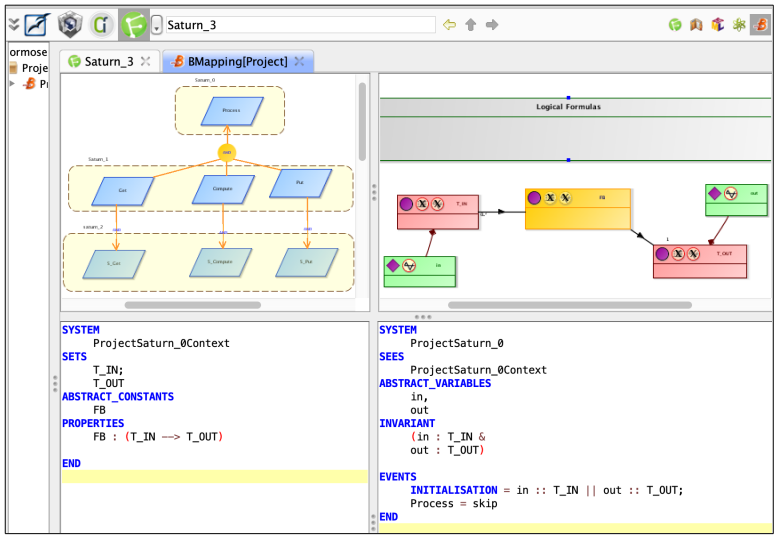


Figure 5: Overview of the SysML/KAOS Modeling of the *Saturn* Protocol [8]

References

[1] Régine Laleau, Farida Semmak, Abderrahman Matoussi, Dorian Petit, Ahmed Hammad, and Bruno Tatibouet.
A first attempt to combine SysML requirements diagrams and B.
Innovations in Systems and Software Engineering, 6(1-2):47–54, 2010.

[2] Christophe Gnaho, Farida Semmak, and Regine Laleau.
An overview of a SysML extension for goal-oriented NFR modelling.
In *RCIS 2013, Paris, France, May 29-31, 2013*, pages 1–2. IEEE, 2013.

[3] Christophe Gnaho, Régine Laleau, Farida Semmak, and Jean-Michel Bruel.
bCMS requirements modelling using SysML/KAOS.

[4] Steve Tueno, Régine Laleau, Amel Mammar, and Marc Frappier.
Towards Using Ontologies for Domain Modeling within the SysML/KAOS Approach.
IEEE proceedings of MoDRE workshop, 25th IEEE International Requirements Engineering Conference.

[5] Steve Tueno, Régine Laleau, Amel Mammar, and Marc Frappier.
The SysML/KAOS Domain Modeling Approach.
ArXiv e-prints, cs.SE, 1710.00903, September 2017.

[6] Abderrahman Matoussi, Frédéric Gervais, and Régine Laleau.
A goal-based approach to guide the design of an abstract Event-B specification.
In *ICECCS 2011*, pages 139–148.

[7] Steve Jeffrey Tueno Fotso, Amel Mammar, Régine Laleau, and Marc Frappier.
Event-b expression and verification of translation rules between sysml/kaos domain models and B system specifications.
In *ABZ*, volume 10817 of *Lecture Notes in Computer Science*, pages 55–70. Springer, 2018.

[8] Steve Tueno, Marc Frappier, Régine Laleau, Amel Mammar, and Hector Ruiz Barradas.
The Generic SysML/KAOS Domain Metamodel.
ArXiv e-prints, cs.SE, 1811.04732, November 2018.

[9] Steve Tueno, Régine Laleau, Amel Mammar, and Marc Frappier.
Formal representation of SysML/KAOS domain models.
ArXiv e-prints, cs.SE, 1712.07406, December 2017.

[10] Thierry Leconte, David Déharbe, Étienne Prun, and Erwan Mottin.
Applying a formal method in industry: A 25-year trajectory.
In *SBMF 2017*, volume 10623 of *Lecture Notes in Computer Science*, pages 70–87. Springer, 2017.

[11] Steve Tueno, Régine Laleau, Amel Mammar, and Marc Frappier.
SysML/KAOS Approach on the Hybrid ERTMS/ETCS Level 3 case study, 2018.

[12] Steve Jeffrey Tueno Fotso, Marc Frappier, Régine Laleau, Amel Mammar, and Michael Leuschel.
Formalisation of SysML/KAOS goal assignments with B system component decompositions.
In *IFM 2018*, volume 11023 of *Lecture Notes in Computer Science*, pages 377–397. Springer, 2018.

[13] Steve Jeffrey Tueno Fotso, Marc Frappier, Regine Laleau, and Amel Mammar.
Back propagating B system updates on SysML/KAOS domain models.
In *ICECCS 2018*, pages 160–169. IEEE, 2018.

[14] ANR-14-CE28-0009.
Formose ANR project, 2017.

Acknowledgements

This work is carried out within the framework of the *FORMOSE* project [14] funded by the French National Research Agency (ANR). It is also partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Contact Information

- **Web:** http://formose.lacl.fr/
- **Email:** steve.tuenofotso@univ-paris-est.fr, laleau@u-pec.fr, Marc.Frappier@USherbrooke.ca, amel.mammar@telecom-sudparis.eu

FROM RESEARCH TO INDUSTRY

cea tech

list

Sail Through Your C Code Either Statically or Dynamically with MetAcsl

Virgile Robles, Nikolai Kosmatov,
Virgile Prevosto, Louis Rilling,
Pascale Le Gall

Problem

Function contracts are not suited to express every property:
- Some properties are **hard to express with contracts** alone
- Some properties span across a l**arge number of functions**

Lack of a high-level specification mechanism amenable to automatic verification and testing in FRAMA-C [1].

Example

Confidentiality-sensitive page management:
- each **memory page** has a **confidentiality level**
- each **user** has a confidentiality level
- a process can only **read/write** a page when allowed by the relative confidentiality levels (see Figure 1, 2)
- these constraints are **pervasive** in the program

Solution

A new specification mechanism: the **Meta-Property**
- a set of **target functions**
- a **context** (strong invariant, writing constraint, …)
- a first order **property** on the memory

A verification mechanism:
- translate meta-properties back to ACSL

with the **MetAcsl** [2] plugin for FRAMA-C (see Figure 3, 4).

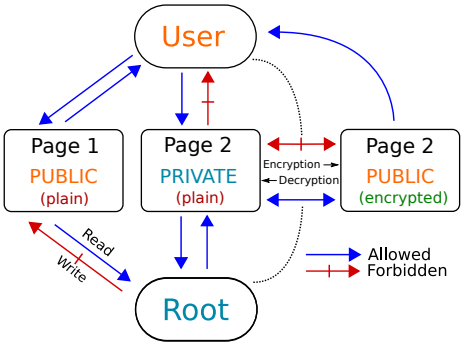


Figure 1: Allowed and forbidden accesses for two agents, two pages and encrypted data

```c
struct Page* page_alloc();
void page_free(struct Page* p);
int page_read(
        struct Page* from,
        char* buffer);
int page_write(
        struct Page* to,
        char* buffer);
int page_encrypt(struct Page* p);
int page_decrypt(struct Page* p);
```

Figure 2: Simplified API of the confidentiality-oriented example

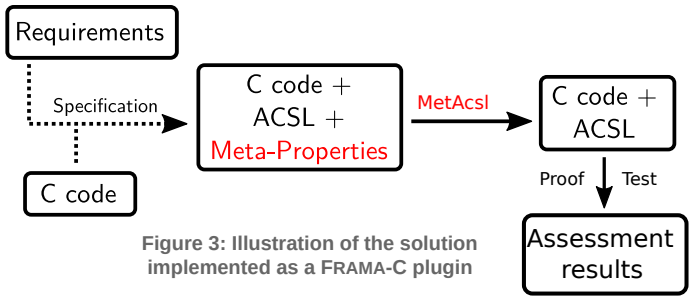


Figure 3: Illustration of the solution implemented as a FRAMA-C plugin

frama C

Software Analyzers

```c
/*@ meta \prop,
        \name(write_confidentiality),
        \targets(\ALL),
        \context(\writing),
        \forall Page* p; page_level(p) > process_level
        || \separated(page_data(p)[i], \written);
*/
/*@ meta \prop,
        \name(cons_proc_level),
        \targets(\diff(\ALL, secure_chg_lvl)),
        \context(\writing),
        \separated(&process_level, \written);
*/
```

```c
int page_write(struct Page* to, char* buffer) {
        for(int i = 0 ; i < page_size(p) ; ++i) {
                /*@ assert write_confidentiality:
                        \forall Page* p;
                        page_level(p) > process_level
                        || \separated(page_data(p)[i], page_data(to)[i]);
                */
                /*@ assert cons_proc_level:
                        \separated(&process_level, page_data(to)[i]);
                */
                page_data(p)[i] = buffer[i];
        }
}

/* Some other functions */

int page_read(struct Page* from, char* buffer) {
        // ...
        /*@ assert write_confidentiality:
                \forall Page* p;
                page_level(p) > process_level
                || \separated(page_data(p)[i], &process_level);
        */
        //@ assert cons_proc_level: \false;
        process_level = 99999;
}
```

Figure 4: Automatic translation of meta-properties with MetAcsl

Contributions

- A specification mechanism, **meta-properties,** to express high-level properties in Frama-C, and several useful extensions.
- A specification **transformation technique**, enabling the use of existing assessment tools on meta-properties:
  - **Static** deductive verification with the WP plugin
  - **Dynamic** assertion checking with the E-ACSL plugin
- A FRAMA-C plugin, **MetAcsl**, implementing this technique **automatically**, making it easy to re-verify properties after a code or specification update.

References

[1] Kirchner, F., Kosmatov, N., Prevosto, V., Signoles, J., Yakobowski, B. **FRAMA-C - A software analysis perspective.** In: Formal Aspects of Computing (2014)
[2] Robles, V., Kosmatov, N., Prevosto, V., Rilling, L., Le Gall, P. **MetAcsl: Specification and Verification of High-Level Properties.** (tool demo paper) In : TACAS (2019)
[3] Robles, V., Kosmatov, N., Prevosto, V., Rilling, L., Le Gall, P. **MetAcsl : spécification et vérification de propriétés de haut niveau** (long abstract of [2], in French). In: AFADL (2019)

Commissariat à l'énergie atomique et aux énergies alternatives
F-91191 Gif-sur-Yvette Cedex

This work was partially supported by the project VESSEDIA, which has received funding from the EU Horizon 2020 research and innovation programme under grant agreement No 731453. The work of the first author was partially funded by a Ph.D. grant of the French Ministry of Defense.

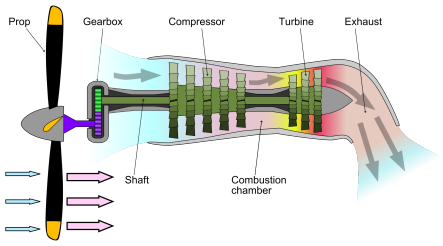
Vessedia DGA CentraleSupélec

Optimind : Towards a Model-Driven Toolbox for Engineers

Adel Ferdjoukh,
{adel.ferdjoukh}@altran.com

Introduction and Problematic

Optimind is a workflow manager for multidisciplinary optimisation. It was developed by Altran Technologies for many years. The main objective of Optimind is to allow for an engineer to easily **combine different domain specific tools as quickly as possible**. The first implementation of Optimind was done in Python in the context of optimisation for turboprop aircraft engines.

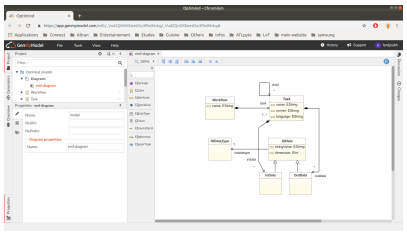


Recently, we started the second phase of the project. The goal is to use Model Driven Engineering to create a generic toolbox for engineers. Many challenges arise:

- Multi-language support
- Deploy on the Cloud
- Allow collaborative work
- Offer design assistance

Collaborative design

Nowadays, running tools on the cloud and collaborative work are very recurring demands from software users. Rich client usage is decreasing day after day.
A possibility is to convert from RCP to **Eclipse RAP** application [1] (Remote Application Platform).
GenMyModel [2] is a promising (commercial) tool that allow collaborative work on modelling in a web browser.



References

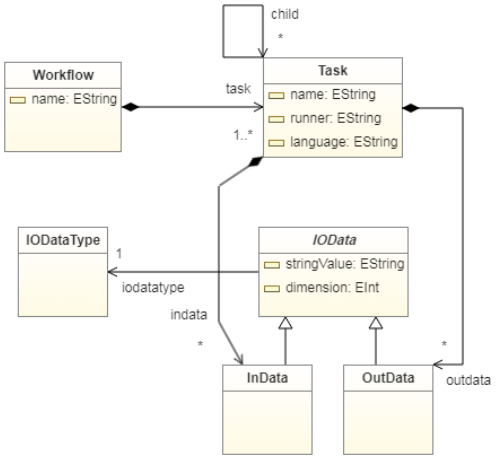
[1] Clausen, M., Hatje, J., Rathlev, J., and Meyer, K. Eclipse rcp on the way to the web. *ICALEPCS 2009–Proceedings* (2009), 884–886.

[2] Dirix, M., Muller, A., and Aranega, V. Genmymodel: an online uml case tool. In *ECOOP* (2013).

[3] Ferdjoukh, A., Baert, A.-E., Chateau, A., Coletta, R., and Nebut, C. A CSP Approach for Metamodel Instantiation. In *IEEE ICTAI* (2013), pp. 1044–1051.

A DSML for workflows

We designed an Ecore meta-model in order to describe the data that our *workflow manager* has to handle. Here we give a simplified version.



MDE and Eclipse RCP (Rich Client Platform) is beneficial for our project :

- Easy peristence of data
- Save dev amount and time
- Quick Editor Creation (Xtext & Sirius

Sirius Xtext

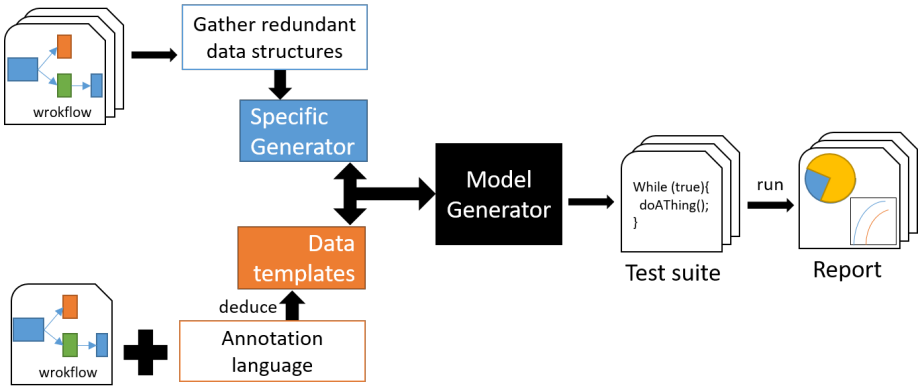
Design assistance and automated tests

In Optimind, workflow designers create generic workflows. Users need to parameter these workflows (creating a runtime instance). It is then crucial for designers to ensure the validity of their workflows. Manual testng of workflow is hard and time-consuming :

- Multiple input data
- Diverse data types
- Size of workflows

For that we propose to use existing model generation tools (such as grimm [3]) and adapt them to generate data to automatically test workflows.

1. Gather data on redundant data structures : Polynomial, String generation, Custom file structure, etc
2. Provide designers an annotation language to create data templates
3. Generate a test suite
4. Run and report



Verification of programs with arrays using Horn Clauses

Julien Braine
ENS de Lyon, France

Ph.D advisors: Laure Gonnord, David Monniaux
julien.braine@ens-lyon.fr

Our Team: CASH

Static Analyses in the team

- Design new low-cost analyses to allow compiler optimizations
- Design safe domain specific languages to avoid programmer bugs
- Design precise, non domain specific, static analysis to ensure correctness of code

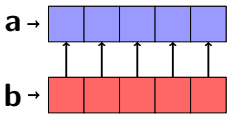
Static analysis to ensure functional correctness of code

Goal: Ensure code correctness $\neq$ finding as many bugs as possible
Setting: Programs have assertions describing the desired behavior
General problem: How to check all possible runs?
My focus: The case of programs with data-structures, especially arrays

Setting: Horn clauses as semantics of a program

Example: array copy program.

```
j = rand()%N;
for(i=0; i<N; i++) {
    a[i] = b[i];
}
assert (b[j] == a[j]);
```



Horn clauses: A logical formula expressing the assertion and the program's semantics.
Shape of Horn clauses:

- Existentially quantified predicates, represent possible values at each program point
- Universally quantified variables to define the transition relation

Result of a Horn clauses solver:

- SAT $\Rightarrow$ Found instanciation for predicate $\Rightarrow$ Program correct
- UNSAT $\Rightarrow$ No possible predicate instantiation $\Rightarrow$ Program is buggy
- Unknown or timeout $\Rightarrow$ Unable to find instanciation or disprove its existence

Translation from programs: Abstracts memory, and specifics of the language

Example:

$$
\begin{array}{lll}
True & \wedge\, j < N \longrightarrow & Start(a,b,N,i,j) \\
Start(a,b,N,i,j) & \longrightarrow & Loop(a,b,N,0,j) \\
Loop(a,b,N,i,j) \quad \wedge\, a' = a[i \leftarrow b[i]] & \wedge\, i < N \longrightarrow & Loop(a',b,N,i+1,j) \\
Loop(a,b,N,i,j) & \wedge\, i \geq N \longrightarrow & Assert(a,b,N,i,j) \\
Assert(a,b,N,i,j) & \wedge\, a[j] \neq b[j] \longrightarrow & False
\end{array}
$$

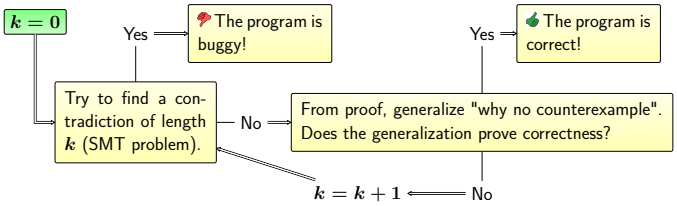
Horn Clauses

- can express the semantics of programs with no information loss
- have clear and easily defined semantics (its a logical formula!)
- have a very simple unified syntax $\Rightarrow$ very good intermediate representation
- tools (such as SeaHorn[1]) can generate Horn clauses from programs (LLVM bytecode)
- have efficient solvers such as Z3[2]

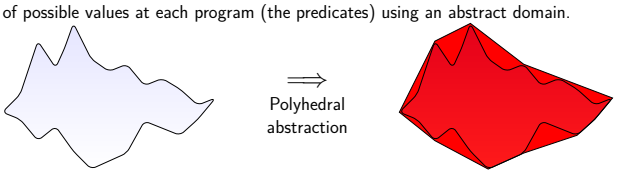
[1] https://seahorn.github.io/
[2] https://github.com/Z3Prover/z3/
[3] https://hal.archives-ouvertes.fr/hal-01162795/document
https://hal.archives-ouvertes.fr/hal-01206882v3/document
[4] https://github.com/vaphor

Related work: Interpolants and Abstract Interpretation to solve Horn Clauses

Interpolants



Abstract Interpretation Abstract Interpretation consists in over-approximating the set of possible values at each program (the predicates) using an abstract domain.



Comparison of these techniques

	Abstract Interpretation	Interpolants
Requires	Abstract domain and abstraction of program's operations	Interpolation technique for the given theory
Soundness	👍	👍
Precision	Fixed by abstract domain	Fixed by underlying logic
Uses assertion	👎	👍
Termination	👍	👎
Predictable failures	👍	👎
Horn Solver	?	Z3
Handles well arrays	👎	👎

PhD intro: Handling arrays in Horn clauses

Problem: Arrays $\Rightarrow$ quantified invariants $\Rightarrow$ no good enough interpolation technique.

Solution: Create a new Horn problem without arrays by using abstract interpretation and solve it with a state of the art solver.

Example:

- **Program:** array copy.
- **Technique:** SAS15-16, Monniaux & Alberti & Gonnord[3]
- **Abstract domain:** Cell abstraction.
  An array a is abstracted by its cells, that is $\{(k, a[k]), k \in \mathbb{N}\}$.
- **Using the abstract domain in Horn clauses (simple version)**
  Replace $P(a, v)$ by $P^{\#}(k, a[k], v)$ in the Horn clauses
- **Fully removing arrays**: no array type in predicates $\Rightarrow$ Apply array axioms $\Rightarrow$ no arrays
- **Solving the abstracted problem**: Launch Z3. *Answer:* SAT in <1s. 👍

Tool: *Vaphor* by Braine & Monniaux & Gonnord[4]

My PhD

In the context of Horn clauses, my goal is:

Improve existing array abstractions	$\longrightarrow$	Extend to other data-structures
Function summaries for scaling	$\longrightarrow$	Implement and test these techniques in a tool (FramaC?)
Implement a verified equivalent of STL (but in C)	$\longrightarrow$	Use this "STL" in verified algorithms

http://perso.ens-lyon.fr/julien.braine/

http://www.inria.fr/ http://www.ens-lyon.eu/ http://www.ens-lyon.fr/LIP/ https://www.universite-lyon.fr/ http://www.cnrs.fr/

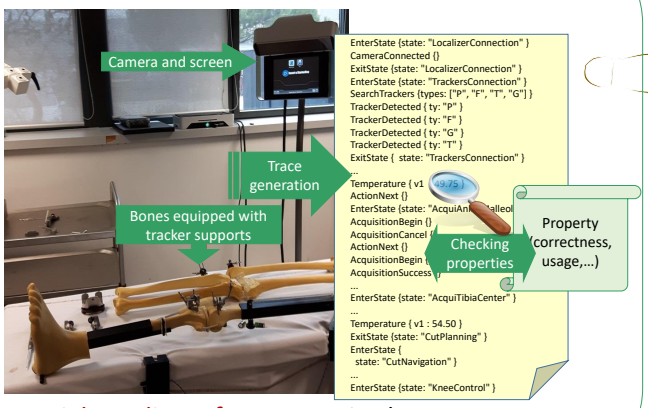
ParTraP : a language and its toolset for the specification of parametric trace properties

Yoann Blein, Ansem Ben Cheikh, Salim Chehida, German Vega, Yves Ledru, Lydie du Bousquet

VASCO team, Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000, Grenoble, France

Yves.Ledru@imag.fr

Context : Computer-aided Surgery



High quality software required
but formal methods are not mandatory!

ParTraP : Parametric Trace Property language

Context and Needs:

- Off-line trace monitoring (traces sent to Blue Ortho after the surgery)
- Temporal properties expressed on parametric events
- Properties valid on a restricted scope of the trace
- An intuitive language for SW Engineers not trained in formal methods

Design choices for the ParTraP language [FormaliSE18]

- Declarative with keyword-oriented syntax
- The use of Dwyer's specification patterns
- Possible inclusion of Python assertions

Sample properties

« The temperature of the camera must stay above 25°C »

```
ValidTemp : absence_of Temp t where t.v1 < 25;
```

Or using Python assertions:

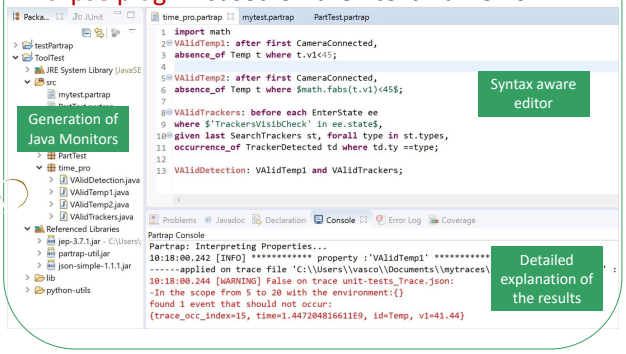
```
import math
ValidTemp : absence_of Temp t where $fabs(t.v1) < 25$;
ValidTemp1 : after first CameraConnected, ValidTemp;
```

« Each EnterState should be followed by an ExitState with the same state parameter »

```
ValidExit : EnterState ee followed_by ExitState xx
            where ee.state == xx.state;
```

ParTraP-IDE [RV18]

Eclipse plug-in based on the Xtext framework



Example and Counter-example generator

Helps software engineers to figure out the meaning of their formulae!

Given a ParTraP property, our prototype tool based on Z3 SMT solver, finds a valuation of the trace which satisfies the property.

For example the following property

```
(assert (afterFirst "CameraConnected"
            (absence_of_where "Temp" t (< (v1 t) 25.0))))
```

Produces the following trace where **Temp** events are absent

```
[{"id": "CameraConnected", "time": 5263, "v1": 2.0},
  {"id": "C", "time": 5264, "v1": 0.0},
  {"id": "CameraConnected", "time": 5853, "v1": 4.0}]
```

Availability

ParTraP-IDE is distributed as open-source at

http://vasco.imag.fr/tools/partrap/

A « lightweight formal method » to fit the needs of software engineers not trained in FM

- Keyword oriented
- Python expressions
- Detailed explanations of results
- Examples generation

[FormaliSE18] Yoann Blein, Yves Ledru, Lydie du Bousquet, Roland Groz:
« *Extending specification patterns for verification of parametric traces* ».
FormaliSE@ICSE 2018: 10-19, ACM, 2018, https://doi.org/10.1145/3193992
[RV18] Ansem Ben Cheikh, Yoann Blein, Salim Chehida, Germán Vega, Yves Ledru, Lydie du Bousquet:
« *An Environment for the ParTraP Trace Property Language (Tool Demonstration)* ».
RV 2018: 437-446, Springer, 2018, https://doi.org/10.1007/978-3-030-03769-7_26

This project is partly funded by the ANR MODMED project (ANR-15-CE25-0010).

Static and Dynamic Green Semantic Model for Software

Adel Noureddine

Université de Pau et des Pays de l'Adour, LIUPPA, Anglet, France
adel.noureddine@univ-pau.fr

Abstract. We propose a semantic model to capture and structure static and dynamic information related to software energy. The model is to be built using source code and project's static information, and by generating a dynamic prediction model for energy consumption with an empirical and statistical approach.

Keywords: Software Engineering · Green Computing · Semantic Modeling

1 Introduction and motivation

Energy demands in information systems are rising, from computers devices to connected devices and software, with a projection of ICT technologies greenhouse gas emissions exceeding 14% of global GHCE in 2040 [1]. These findings show the necessity to reduce and optimize energy consumption in these systems, and many efforts have been made to address energy, with approaches covering two main directions : monitoring or estimating energy consumption [2], or optimizing energy footprint of software, servers or equipment in particular workloads. [3–6]

However, these approaches address energy in individual and separate layers or "silos": energy is managed and optimized for a particular layer in the system (software, servers, equipment, OS, VM, middleware, etc.) or a particular workload. The next big leap in energy efficiency requires addressing energy in a holistic way.

Our proposition is to address this holistic philosophy by proposing a semantic model regrouping static and dynamic software metrics, with an empirical energy and performance prediction model. Our goals are therefore to: 1) propose a semantic model to regroup static and dynamic software metrics, 2) propose performance and energy prediction models for software, and 3) provide a comprehensive software semantic model to be used on runtime for multiple use cases.

Use cases vary from software adaptation and reconfiguration, autonomic deployment of software, to runtime migration based on our performance and energy model. Application domains include reconfiguration and deployment in data centers, servers and virtual machines, adaptations and optimizations in limited resources environments such as internet of things, or economic concerns in industrial environments as in industry 4.0 sectors or in smart homes and smart cities.

2 Green Software Semantic Model

2.1 Why a semantic model?

Images, such as in the JPEG format, includes metadata in a standard specification called EXIF (Exchangeable image file format). This format adds non-visible (as in non-functional requirement) data to an image file to expand knowledge about the image. Users can therefore view the content of the image, while at the same time, this metadata can be used to perform reconfigurations or adaptations (*e.g.*, regroup images based on location, condition printing based on image resolution).

However, software does not include a comprehensive semantic modeling to provide users, on runtime, quality data on software, their usage context, and their performance and energy consumption. Our proposition is therefore to create a semantic model, similar to EXIF, for software.

2.2 Description of the Green Software Semantic Model

Our semantic model for software energy and performance is composed of four main categories of information:

General metadata: This category of metadata includes information about the software, its authors, license information and related general and non-technical data. Typically, this information is provided or generated directly in the source code, in accompanying readme and configuration files. For example, Maven projects include much of this data in a pom.xml file. Many JavaScript projects provide a package.json file with similar information, C/C++ projects often include Makefile and CMake configuration files, or Rust projects use Cargo.toml files. Examples of extracted metadata include software name, description, versions, authors, licenses and copyrights, or developer and usage documentation.

Software quality information: In contrast to the general metadata, software quality information consolidate data about code quality and various software metrics. This information is gathered from the software source code through existing analyzers, such as Frama-C or SonarQube. The data collected is divided into the following categories:

- Structural data: metrics describing how the source code is written, such as the number of lines of code, number of methods/functions, variables, statements, conditions and loops, APIs.
- Quality data: how well the code is written, such as the complexity of its methods/functions, duplication statistics, technical debt or design metrics.
- Test data: how well the code has been tested and covered. Typically, this data includes the number of unit tests, test duration, errors, failures, etc., and coverage metrics.

Software and hardware dependency information: Applications and software components are built and deployed in an ecosystem of dependent components. Therefore, the energy efficiency of software is related to the various software dependencies. For example, a web application dependent on a comprehensive framework such as Spring or Struts frameworks will have a different energy impact than a similar-feature application built with plain Java and JSP files. In addition to software dependencies, we collect in our model information about hardware dependencies. Information such as on which platform the application is built, tested or can run are provided, along with specific hardware or platform requirements. These data can be gathered from metadata in configuration and README files, as with general metadata, but may also be available in the source code or through tools such as SonarQube.

Performance and energy prediction data models: The prediction model is based on an empirical approach, with statistical and regression analysis on large amount of performance and energy consumption data and metrics. Our empirical approach requires large amount of quality data, e.g., performance and energy data structured for each program and source code. We propose to generate this data using controlled experiments in a lab, and industrial experiments in production environments.

References

1. Lotfi Belkhir and Ahmed Elmeligi. Assessing ICT global emissions footprint: Trends to 2040 & recommendations. *Journal of Cleaner Production*, 2018.
2. Adel Noureddine, Romain Rouvoy, and Lionel Seinturier. A review of energy measurement approaches. *Operating Systems Review*, 47(3):42–49, 2013.
3. L. Ardito, G. Procaccianti, M. Torchiano, and A. Vetrò. Understanding green software development: A conceptual framework. *IT Professional*, 17(1):44–50, Jan 2015.
4. Anne-Cecile Orgerie, Marcos Dias de Assuncao, and Laurent Lefevre. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Comput. Surv.*, 46(4):47:1–47:31, March 2014.
5. Jean-Marc Pierson. *Large-scale Distributed Systems and Energy Efficiency: A Holistic View.* John Wiley & Sons, 2015.
6. EA Jagroep. *Green Software Products.* PhD thesis, Utrecht University, 2017.

Défis dans le développement collaboratif et ouvert de l'assistant de preuve Coq et de son écosystème

Théo Zimmermann[1,2] theo@irif.fr

[1] Université de Paris, IRIF, CNRS, F-75013 Paris, France
[2] Inria, équipe-projet πr^2

1 Introduction

Coq [1] est un logiciel libre développé par des chercheurs Inria depuis plus de 35 ans, qui permet d'énoncer formellement des théorèmes mathématiques et d'en vérifier les preuves. Ce logiciel et le langage associé s'appuient sur des décennies de recherche sur la logique d'une part, et les langages de programmation fonctionnels d'autre part. Cependant, à mesure que ce logiciel grandit, en nombre d'utilisateurs, en taille et complexité du code, et que son développement s'ouvre davantage aux contributeurs extérieurs, d'autres types de questions de recherche se posent.

Ces nouvelles questions ont trait notamment à la maintenance et à l'évolution du logiciel, à la facilitation de son installation et de son utilisation, à l'implication des utilisateurs dans le développement open source du logiciel, sa documentation, et son écosystème de paquets.

La thèse que je défends dans mon travail de doctorat est que le succès d'un logiciel mathématique complexe et d'un langage fonctionnel fortement typé tel que Coq ne pourra être effectif que grâce aux avancées que permettront le mariage de la théorie et de la pratique, de la recherche en logique et langages de programmation, avec celle en génie logiciel. Historiquement, cette dernière a davantage été associée aux langages orientés objet, qui en ont beaucoup bénéficié. Dans le même temps, les langages fonctionnels, plus à la pointe au niveau des concepts, et permettant de programmer de manière plus sûre et abstraite, pâtissaient grandement d'un manque d'outillage avancé.

2 Démarche

Mon objet d'étude est un développement logiciel spécifique, et ma démarche consiste à identifier les problèmes concrets dont souffrent les développeurs ou la communauté d'utilisateurs en étant moi-même impliqué au cœur du développement. Une fois les problèmes concrets identifiés, il est nécessaire de prendre du recul afin d'en déterminer la substance. Cette étape d'abstraction permet de raisonner sur une question qui n'est plus spécifique à ce développement logiciel ou cette communauté d'utilisateurs, même si des particularismes peuvent continuer à jouer un rôle dans la mise en place de solutions génériques.

Parfois, les solutions à ces problèmes ont déjà été clairement identifiées et n'ont plus qu'à être appliquées ; parfois la littérature scientifique n'a pas encore abordé le problème en question mais d'autres projets open source ont été confrontés à un problème similaire et ont proposé une solution. Mais l'évolution rapide et continue des processus et outils de développement dans le monde du logiciel libre fait que les problèmes sont parfois nouveaux, ou trop spécifiques, et sont en attente de solutions novatrices. Dans ce qui suit, j'illustre cette démarche sur des exemples concrets.

Mesurer l'impact d'un changement de bug tracker. Le logiciel Coq est aussi ancien que le concept même de logiciel libre. Le dépôt contenant son code source versionné date pour sa part de 1999, année de création du terme « open source ». En 2007, c'est-à-dire juste avant la création de GitHub, le projet avait commencé à utiliser le *bug tracker* Bugzilla, l'un des bug trackers libres les plus connus, utilisé par de nombreux logiciels libres importants. Dix ans plus tard, en 2017, GitHub était devenu la plateforme centrale de développement de Coq. Le système de *pull request* était utilisé, non seulement pour intégrer les contributions extérieures, mais aussi pour tester et valider les changements proposés par les développeurs principaux. Dans ce contexte, l'utilisation de Bugzilla

devenait de plus en plus problématique : nécessité de changer de plateforme pour naviguer entre les tickets et le code, lourdeur, lenteur, et manque de maniabilité comparative, qui rendaient son utilisation plus difficile, à la fois pour les développeurs principaux, et les contributeurs extérieurs.

J'ai tout d'abord émis l'idée d'opérer un basculement vers le bug tracker intégré à GitHub, en proposant un comparatif des fonctionnalités. Puis, j'ai démontré la faisabilité d'une migration des tickets pré-existants en réutilisant et adaptant un outil [2] qui n'était pas conçu pour supporter une migration de l'ampleur de celle de Coq. Cette évaluation de la faisabilité et des bénéfices a permis à l'équipe de développement de valider le changement de bug tracker, que j'ai alors effectué.

Enfin, à l'aide de données de l'activité sur le bug tracker dans une période de plus de 18 mois avant et autant après le changement, et de la méthode de *Regression on Discontinuity*, nouvelle en génie logiciel empirique, j'ai montré [3] l'effet causal de ce changement de bug tracker sur l'activité : les développeurs y sont plus actifs et les non-développeurs sont plus nombreux à participer aux discussions sur les tickets. Ce résultat est basé sur un seul cas d'étude et devrait être répliqué sur d'autres projets, mais il est aussi la première comparaison de bug trackers du genre.

Gérer la documentation des changements dans les nouvelles versions. En 2017, j'ai mis en place un nouveau processus d'intégration des pull requests basé sur le modèle (répandu) du *backporting*, avec un *release manager* qui prend les décisions de backporter ou non chaque changement, alors qu'auparavant les développeurs devaient choisir eux-mêmes pour quelle branche (stable ou *master*) ils préparaient leurs correctifs (les branches stables étant fusionnées plus tard dans *master*). Or, cette simplification du processus a pour conséquence qu'au moment de la préparation et de l'intégration d'un correctif, les développeurs ne savent pas avec certitude dans quelle version celui-ci sera présent en premier, ce qui a généré des incohérences dans le fichier de *changelog*.

La solution que j'ai proposée est inspirée par une idée des développeurs de GitLab [4] pour résoudre les conflits récurrents dans leur changelog. Chaque pull request décrit les changements qu'elle introduit dans un fichier séparé, et c'est au moment de sortir une nouvelle version que la documentation des changements dans cette version est compilée, sur la base du sous-ensemble de ces fichiers se trouvant dans la branche stable correspondant.

Alors que la plupart des recherches sur le backporting ont porté sur l'exemple du noyau Linux, mon analyse détaillée des différentes options méthodologiques adaptées à un projet de taille moyenne, et de leurs limites, devrait pouvoir bénéficier à de nombreux projets.

S'assurer de la maintenance des paquets sur le long terme. La force d'un *framework* ou d'un langage de programmation réside souvent en grande partie dans son écosystème de paquets. Les paquets développés pour Coq ont la double particularité d'avoir fortement besoin d'être maintenus pour rester compatibles avec les évolutions de Coq, et d'être souvent l'œuvre d'étudiants, notamment en thèse qui ne restent pas toujours actifs dans la communauté sur le long terme.

L'initiative *coq-community* [5], que j'ai lancée en m'inspirant d'*elm-community* [6], est une organisation informelle d'utilisateurs dédiée à la maintenance sur le long terme de paquets Coq suscitant l'intérêt. Moins d'un an après son lancement en juillet 2018, elle connait des débuts prometteurs puisqu'elle accueille déjà 16 paquets et projets, maintenus par 11 mainteneurs principaux. Cet effort est aussi une occasion de proposer et de tester des outils et des processus nouveaux dans la maintenance de paquets Coq, et de contribuer à des investigations plus générales sur la comparaison de divers écosystèmes de paquets en fonction de certaines caractéristiques structurantes.

Références

1. The Coq development team: Coq 8.9.0 (2019). https://doi.org/10.5281/zenodo.1003420.
2. Berestovskyy, A.: bugzilla2github, `https://github.com/berestovskyy/bugzilla2github/`.
3. Zimmermann, T., Casanueva Artís, A.: Impact of switching bug trackers: a case study on a medium-sized open source project (2019). HAL preprint hal-01951176.
4. Speicher, R.: How we solved GitLab's CHANGELOG conflict crisis, `https://about.gitlab.com/2018/07/03/solving-gitlabs-changelog-conflict-crisis/`. Last accessed 17 May 2019.
5. Coq-community manifesto, `https://github.com/coq-community/manifesto`.
6. Elm-community homepage, `http://elm-community.org`. Last accessed 17 May 2019.

PRODUCT LINE CONFIGURATIONS GUIDED BY PROCESS TRACES

Houssem CHEMINGUI, Camille SALINESI, Ines GAM,
Raul MAZO, Henda BEN GHEZALA

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE
Centre de Recherche en Informatique

MASS CUSTOMIZATION SOLUTIONS

One to One offer with a maximum of specific requirements.

PRODUCT LINES

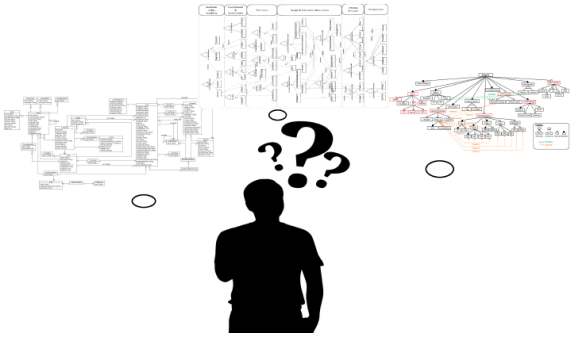
Similar products sharing common characteristics and satisfying particular requirements.

A configuration represents a product of the Line.

PROBLEM STATEMENT

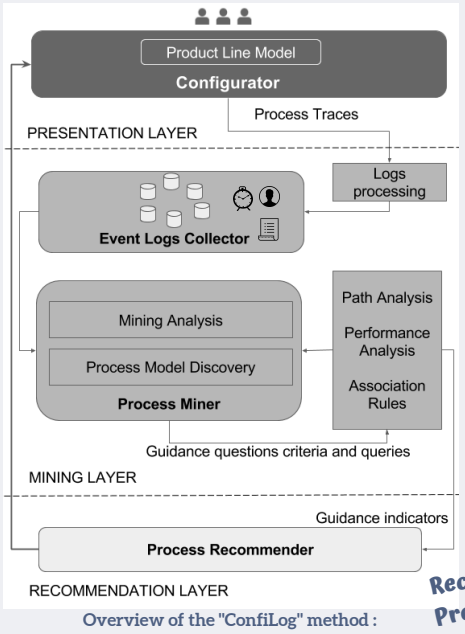
The configuration process becomes quickly complex when considering :

The big number of constrained decisions,
The order of decisions,
The runtime of inappropriate decisions.



How to guide, provide proposals and recommend choices!

PROCESS MINING AS A SOLVING TOOL



Overview of the "ConfiLog" method :
This method is currently under development in the CRI Laboratory

GQM

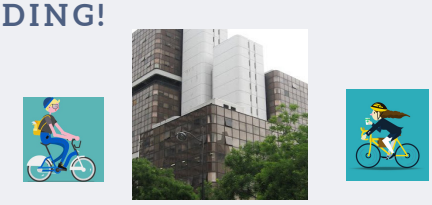
3 Guidance Goals
Rapidity
Flexibility
Customization

25 Guidance Questions

47 Evaluation Metrics

Recommendation of Process Orders and decision values

EXPERIMENTATION IS LOADING!



A bike for each student! Let's Configure!

- Outcome 1 : Collect Real Configuration process Traces
- Outcome 2 : Evaluate the guidance of the "ConfiLog" Configurator.

WHAT WE WILL DO!

Situational Approaches : Goals will be detected and orchestrated automatically according to user partial Configurations.

This work is supported by the French-Tunisian CMCU project 16G/1416 called CONFIGURE.

+33 7 67 92 12 91
Mohamed-Houssem.Chemingui@etu.univ-paris1.fr

Ce document contient les actes des onzièmes journées nationales du Groupement De Recherche CNRS du Génie de la Programmation et du Logiciel (GDR GPL) s'étant déroulées à l'ENSEEIHT – IRIT du 11 au 14 juin 2019.

Les contributions présentées dans ce document ont été sélectionnées par les différents groupes de travail du GDR. Il s'agit de résumés et de nouvelles versions qui correspondent à des travaux qui ont déjà été validés par les comités de programmes d'autres conférences et revues et dont les droits appartiennent exclusivement à leurs auteurs.