D'un JSON-based Domain Specific Language vers une spécification Parallel-DEVS pour les jumeaux numériques des systèmes de production

Gaston Batchoudi, Eric Ramat

D'un *JSON-based Domain Specific Language* vers une spécification Parallel-DEVS pour les jumeaux numériques des systèmes de production

Gaston Batchoudi[1], Eric Ramat[1]

Université Littoral Côte d'Opale - LISIC EA 4491, F-62228 Calais, France
{gaston.batchoudi,eric.ramat}@univ-littoral.fr

Mots-clés : *Model-to-Model, Model Driven Software Developpement, Parallel-DEVS, Jumeau numérique, Système de production.*

1 Introduction

Le domaine de la modélisation et simulation a connu d'importantes avancées au cours des dernières décennies. Les logiciels de simulation tels que Flexim [1], Witness [2] permettent, de manière efficace et fiable, la modélisation des systèmes de production, mais leur coût élevé rend leur acquisition difficile pour les PME et les pénalisent au sein d'une industrie en pleine mutation vers la technologie du jumeau numérique. Un jumeau numérique est un modèle numérique qui reconstitue fidèlement un objet, une opération ou un système afin de concevoir, d'expérimenter ou de contrôler. Face à ces défis, il s'avère important de développer de nouvelles solutions logicielles basées sur des formalismes de modélisation et des langages de type DSL (*Domain Specific Language*) suivant une démarche de conception orientée MDE (*Model Driven Engineering* - [1]) qui intègre non seulement les concepts des logiciels existants mais aussi les exigences d'une industrie plus responsable.

Un DSL est un langage de programmation spécifiquement conçu pour décrire des systèmes d'un domaine donné. Ils sont souvent plus expressif et permettent d'exprimer des concepts du domaine de manière plus concise et naturelle que les langages généraux. Leur haut niveau d'abstraction leur permet d'abstraire souvent les détails complexes du domaine et permet de construire des applications qui simplifient le développement de logiciel. Par exemple, [2] utilise le JSON-DSL qui utilise la syntaxe JSON (*JavaScript Object Notation*) pour le développement d'applications web. Les DSLs ont aussi accéléré l'émergence de nouvelles approches dans l'ingénierie logicielle, notamment les MDSD (*Model Driven Software Development*) qui facilitent la transformation de ces modèles abstraits en code source ou en d'autres modèles de niveau d'abstraction inférieur. [3] propose un DSL pour la modélisation cognitive et des règles pour transformer les modèles de ce DSL en composants DEVS.

L'évolution des systèmes manufacturiers, de leur modélisation à leur simulation, reflète une transformation significative dans le domaine de la fabrication. De la production artisanale à l'industrie 4.0, l'usine a connu une transition passant par les systèmes dédiés (DMS - *Dedicated Manufacturing System*), les systèmes de production flexibles (FMS - *Flexible Manufacturing System*), les systèmes de production reconfigurables (RMS - *Reconfigurable Manufacturing System*) qui associe de plus en plus la technologie de jumeaux numériques. Le formalisme de modélisation et de simulation que nous avons retenu est *Parallel-DEVS* qui est une extension du formalisme DEVS (*Discrete EVent Specification*) initié par P. Zeigler [5] et qui offre un cadre formel pour modéliser et analyser les systèmes à événements discrets à travers des structures mathématiques rigoureuses. L'association de DEVS à d'autres paradigmes tels que

1. https://www.flexsim.com/fr/
2. https://www.lanner.com/fr-fr/technologie/witness-simulation-software.html

le MDE dans lequel "tout est modèle", et les DSLs constitue une puissante approche facilitant le développement de modèle.

Dans le cadre de cet article, nous proposons une démarche de transformation d'un JSON-DSL dédié aux systèmes manufacturiers, dans le but de construire son modèle Parallel-DEVS. Cette transformation vise à établir un processus méthodologique et automatisé pour convertir les spécifications détaillées d'un système, exprimées à travers un DSL en format JSON, en un modèle Parallel-DEVS.

2 D'un JSON-based DSL vers une spécification Parallel-DEVS

L'analyse d'une partie importante des logiciels commerciaux nous a permis d'identifier et de poser les différents concepts constitutifs d'un système de production au sens de sa structure. Ces différents concepts sont modélisés sous forme d'un diagramme de classes UML (voir un extrait Figure 1). Le diagramme est ensuite traduit en langage JSON sous forme d'un schéma.

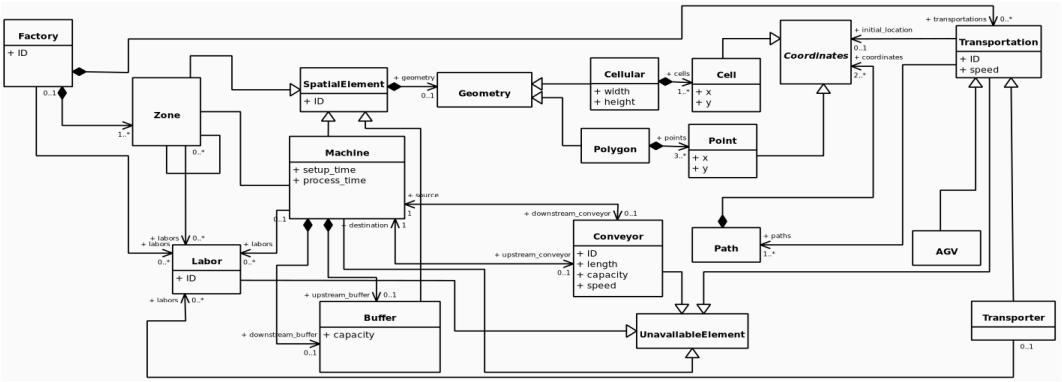

FIG. 1 – Diagramme de classes UML

L'extrait du diagramme de classes montre quelques concepts : *factory*, *zone*, *machine*, *buffer*, *transporter*, *AGV*, ...

Nous introduisons une structure inspirée de celle proposée par Rumbaugh et al. [4], qui décompose un modèle en deux aspects : l'aspect structurel (modèle statique et fonctionnel) qui regroupent la configuration physique et paramétrique du système ainsi que le flux des travaux, et l'aspect dynamique qui contrôle les états du système. Nous adoptons aussi l'approche de [1], qui définit les relations suivantes. *RepresentedBy* désigne la relation de représentation entre un système ou une entité et son modèle. *ConformTo* est utilisé pour indiquer qu'une entité respecte les spécifications d'une autre, dont elle est une instance (*InstanceOf*). *TransformTo* illustre la transformation d'un modèle en un autre, traduisant un homomorphisme entre eux.

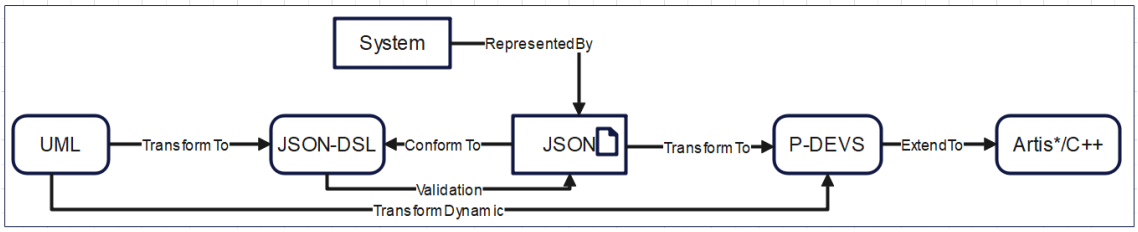

FIG. 2 – Conversion d'un JSON-DSL vers P-DEVS

La procédure suit les étapes suivantes (Figure 2). Dans le premier temps, la modélisation du système donne naissance à un fichier JSON modélisant l'aspect structurel du système. Ensuite, la conformité du contenu du JSON est vérifiée à l'aide d'un schéma JSON qui lui-même est un homomorphisme du diagramme de classes UML. La phase de transformation consiste alors à

extraire les informations spécifiées en JSON pour construire des modèles atomiques *Parallel-DEVS*. Lors de cette transformation, les paramètres JSON sont transposés en un ensemble de paramètres P dans le modèle *Parallel-DEVS*, les flux de travaux en ports d'entrées X et de sorties Y, le temps d'avancement t_a du modèle atomique est obtenu à partir des paramètres de temps. La dynamique est complétée à partir du diagramme UML (*TransformDynamic*) d'où sont traduit les états initiaux s_0, les états S, les fonctions de transitions $\delta_{int}, \delta_{ext}, \delta_{conf}$ et la fonction de sortie λ. Ainsi, la nouvelle structure du modèle atomique est donnée par $M =< P, X, Y, S, s_0, \delta_{int}, \delta_{ext}, \delta_{conf}, \lambda, t_a >$. Enfin, le simulateur est obtenu par instanciation (*ExtendTo*) en C++ des différents modèles atomiques *Parallel-DEVS* des composantes du modèle décrit dans le JSON et un graphe de modèles (modèle couplé) est construit en ajoutant et en connectant ces modèles grâce à la classe *FactoryGraphManager* dérivée de la classe *GraphManager* du framework Artis*[3].

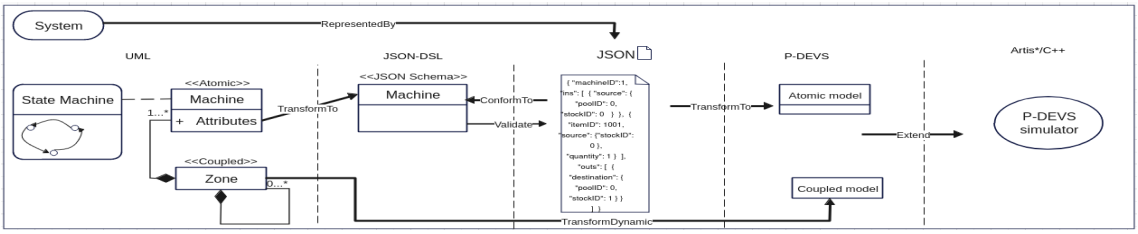

FIG. 3 – Exemple de conversion d'un JSON-DSL vers *Parallel-DEVS*

La Figure 3 présente uniquement les concepts de *machine* comme modèle atomique et de *zone* comme modèle couplé, la procédure est la même pour toutes les entités spécifiées du modèle.

3 Conclusions et perspectives

Le développement des logiciels de simulation des systèmes de production reste un domaine actif à l'ère de l'industrie 4.0. Nous avons proposé une solution JSON-DSL pour convertir une spécification JSON d'un système en un simulateur *Parallel-DEVS*. La suite de ce travail se concentrera sur le couplage du simulateur ainsi construit aux modèles d'optimisation et d'ordonnancement dans le cadre formel DEVS. Nous envisagerons aussi l'intégration et l'échange en temps quasi-réel des données entre le système réel et le simulateur afin de tendre vers les jumeaux numériques.

Références

[1] Jean Bézivin. "On the unification power of models". In : *Software & Systems Modeling* 4.2 (2005), p. 171-188.

[2] Enrique Chavarriaga, Francisco Jurado et Francy D. Rodríguez. "An approach to build JSON-based Domain Specific Languages solutions for web applications". In : *Journal of Computer Languages* 75 (2023), p. 101203. issn : 2590-1184.

[3] Saurabh Mittal et Scott A Douglass. "From domain specific languages to DEVS components : application to cognitive M&S." In : *SpringSim (TMS-DEVS)*. 2011, p. 256-265.

[4] James Rumbaugh et al. *Object-oriented modeling and design*. T. 199. 1. Prentice-hall Englewood Cliffs, NJ, 1991.

[5] Bernard P Zeigler, Herbert Praehofer et Tag Gon Kim. *Theory of modeling and simulation*. Academic press, 2000.

3. `https://gitlab.com/artis-star/artis-star`, un environnement de multi-modélisation et de simulation de systèmes complexes développé au LISIC/ULCO