
SAGITAL : un environnement d'aide à la conception de grafjets basé sur des méta-modèles

Jean-Marc Faure* — Florent Couffin* — Sandrine Lampérière-Couffin^{*,**}

*Laboratoire d'Ingénierie Intégrée des Systèmes Industriels (LIISI)
CESTI-ISMCM Toulon, Place Georges Pompidou, F-83000 Toulon
{ Jean-Marc.Faure, Florent.Couffin }@toulon.ismcm-cesti.fr

**Laboratoire Universitaire de Recherche en Production Automatisée (LURPA)
ENS Cachan, F-94235 Cachan Cedex
Sandrine.Couffin@lurpa.ens-cachan.fr

RÉSUMÉ. Cet article présente la démarche de conception d'un logiciel d'assistance à la vérification syntaxique et à la simulation de modèles grafjet. Cette démarche consiste à déduire la structure de données et les algorithmes du logiciel d'assistance de deux méta-modèles exprimant respectivement les propriétés structurelles et dynamiques du Grafjet.

ABSTRACT. This article deals with the design of a software developed for grafjet syntactic verification and simulation. In this approach, both data structure and algorithms of the software are derived from two meta-models. The first one expresses Grafjet structural properties; the second one its dynamic properties.

MOTS-CLÉS : Grafjet, Environnement de conception, Méta-modélisation, Simulation, Vérification de propriétés.

KEY WORDS: Grafjet, Design environment, Meta-modelling, Simulation, Properties checking.

1. Introduction

La conception d'un grafcet destiné à une application industrielle requiert de plus en plus l'utilisation d'un outil logiciel d'assistance. Ce logiciel a pour but d'aider le concepteur à éditer un modèle ainsi qu'à vérifier que celui-ci est conforme à l'outil de modélisation Grafcet et répond aux contraintes de l'application.

La vérification d'un grafcet comporte deux aspects : une vérification syntaxique et une vérification dynamique. La vérification syntaxique d'un grafcet concerne les propriétés structurelles telles que le respect de l'alternance étape - transition, le respect de la hiérarchie de forçage, ... Quant à la vérification dynamique du grafcet, elle concerne l'analyse du comportement du modèle en réponse à des séquences d'entrées et la preuve de propriétés telles que réinitialisabilité, stabilité, absence de blocage, ...

Afin de vérifier les propriétés d'un modèle, il importe que celui-ci soit pourvu d'une syntaxe et d'une sémantique suffisamment rigoureuses. De nombreux travaux ont montré que les techniques de méta-modélisation offrent des solutions permettant une définition non-ambigüe de ces deux éléments [MOR 91], [STE 93], [REV 95]. Dans le cas du Grafcet [IEC 88], il s'agit d'exprimer à la fois ses aspects statique (définition des concepts et des liens entre ces concepts) et dynamique (règles d'évolution et modèle temporel associé).

En se basant sur nos travaux de définition de méta-modèles statique [COU 97b] et dynamique [LAM 98a] du Grafcet, nous avons donc entrepris le développement d'un environnement d'aide à la conception de grafcets, nommé SAGITAL (Synthèse et Analyse de Grafcets InTégrées et Assistées par Logiciel). L'objectif à long terme est de fournir une assistance durant toute la phase de conception d'un grafcet. Dans l'état actuel du développement de ce logiciel seules les fonctions de vérification de la cohérence syntaxique du modèle produit et de simulation de son comportement dynamique sont implantées. Dans la suite de l'article, nous présentons tout d'abord la démarche générale de conception du logiciel puis nous montrons comment l'utilisation de méta-modèles statique et dynamique du Grafcet permet la réalisation de ces fonctions.

2. Démarche de conception du logiciel SAGITAL

Cette démarche (Figure 1) est similaire à celles employées en génie logiciel, et notamment pour la conception des systèmes d'information, qui consistent à dériver de modèles conceptuels les données et les algorithmes implantés dans un logiciel. Dans notre cas, ce sont les méta-modèles du Grafcet qui servent de point de départ à la construction du logiciel.

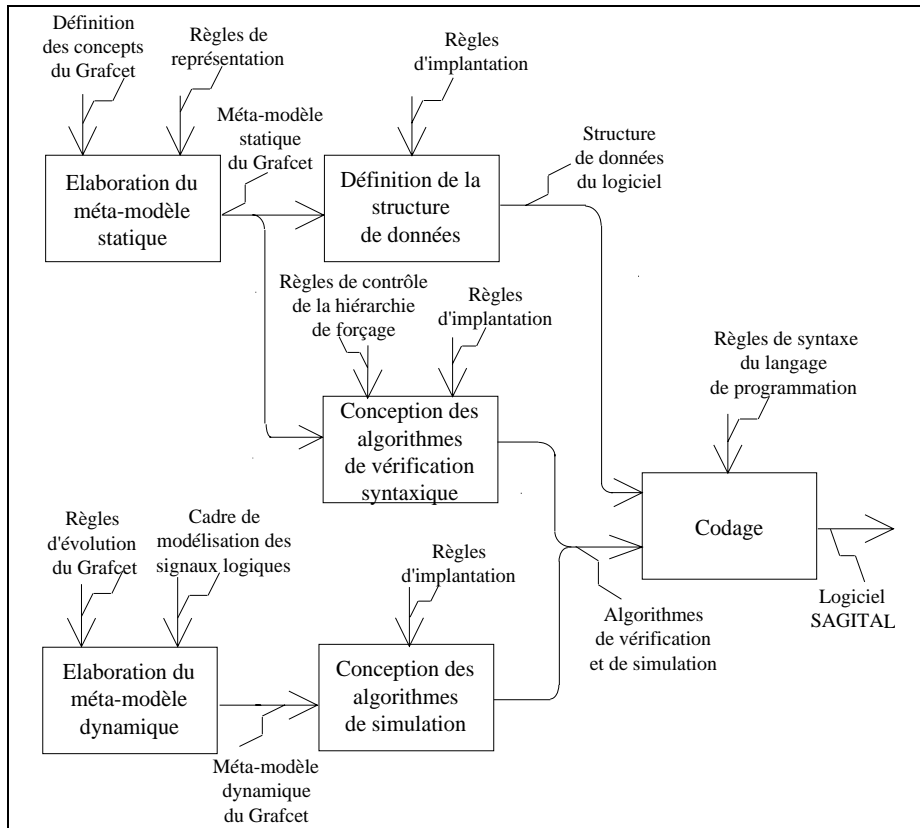


Figure 1. Démarche de conception

Plus précisément, nous avons tout d’abord développé un méta-modèle statique du Grafcet, dans le cadre de travaux visant à l’intégration des activités de conception des systèmes automatisés de production [COU 97a]. La structure de données du logiciel SAGITAL est dérivée de ce méta-modèle décrivant les propriétés structurelles du Grafcet. La vérification complète de la syntaxe d’un modèle grafcet nécessite la définition d’algorithmes exploitant cette structure de données ; ils sont déduits de traitements opérant également sur le méta-modèle statique.

Nous avons ensuite, dans le cadre de recherches sur la vérification/validation du Grafcet [LAM 98a], formalisé les règles d’évolution du Grafcet sous la forme d’un méta-modèle dynamique. Les algorithmes pour la simulation d’un modèle grafcet sont déduits de ce méta-modèle.

De plus, il convient de remarquer que la structure de données et les algorithmes décrivant l’évolution d’un grafcet pourront être utilisés pour la vérification dynamique et la validation d’un modèle dans une version ultérieure du logiciel.

Afin de décrire avec précision les aspects statique et dynamique du Grafcet, il

importe d'exprimer les deux méta-modèles à l'aide de formalismes suffisamment rigoureux et possédant un bon pouvoir d'expression. Nous avons pour notre part choisi :

- un formalisme de type entité-relation enrichi de concepts issus de l'approche objet [COU 97a], tels que les classes de relation d'identification relative et de généralisation/spécialisation pour le méta-modèle statique ;

- une représentation des signaux logiques basée sur un modèle du temps continu à dates discrètes et une algèbre sur le corps de Galois d'ordre 2 pour le méta-modèle dynamique. Cette représentation est basée sur la Théorie du Signal Hyperfini [FRA 97], dans laquelle on adopte cette représentation du temps et où les variables sont des signaux prenant leurs valeurs dans un corps de Galois d'ordre 2^n , avec n entier positif. Le Grafcet se limitant à la description de systèmes logiques, n vaut 1 dans notre étude.

L'utilisation conjointe de ces deux formalismes permet de satisfaire aux exigences fixées.

3. Apport d'un méta-modèle statique à la vérification syntaxique de grafquets

3.1. Principes retenus

Nous considérons que le concepteur d'un grafcet est libre du choix de sa démarche et que l'aide apportée par l'outil logiciel consiste uniquement à vérifier que le modèle produit est syntaxiquement conforme au modèle Grafcet. Cette vérification est possible à deux niveaux :

- en ligne ; il s'agit alors de vérifier que chacune des actions du concepteur est licite (respect de l'alternance étape-transition, de la syntaxe des éléments et des structures de base du Grafcet)

- globale ou a posteriori ; on vérifie alors la syntaxe de l'ensemble du modèle considéré comme achevé.

La répartition entre ces deux modes de vérification dépend de choix méthodologiques. Tous deux nécessitent par contre la définition rigoureuse d'un méta-modèle statique du Grafcet que nous présentons ci-après.

3.2. Présentation du méta-modèle statique

Ce méta-modèle [COU 97b] regroupe l'ensemble des concepts du Grafcet : Grafcet Global, Grafcet Partiel, Etape, Transition, ..., représentés par des classes d'entité, et comporte deux parties inter-reliées. La première partie décrit les règles d'identification des concepts en introduisant des relations hiérarchiques de type composé-composant (un Grafcet Global est constitué de Grafquets Partiels, lui-même constitué d'étapes et de transitions, par exemple) qui sont décrites au moyen de

classes de relation d'identification relative. La deuxième partie de ce méta-modèle décrit les autres relations entre les concepts à l'aide de classes de relation simple et de contraintes sur rôles, sur relations ou explicites. La figure 2 montre un extrait de la deuxième partie de ce méta-modèle ; par souci de concision, les abréviations suivantes sont utilisées pour désigner les classes d'entité :

GC	Grafcet Connexe	ME	Macro-Etape
ET	ETape	TR	TRansition
AC	ACtion	RE	REceptivité
VE	Variable Externe (au modèle)		

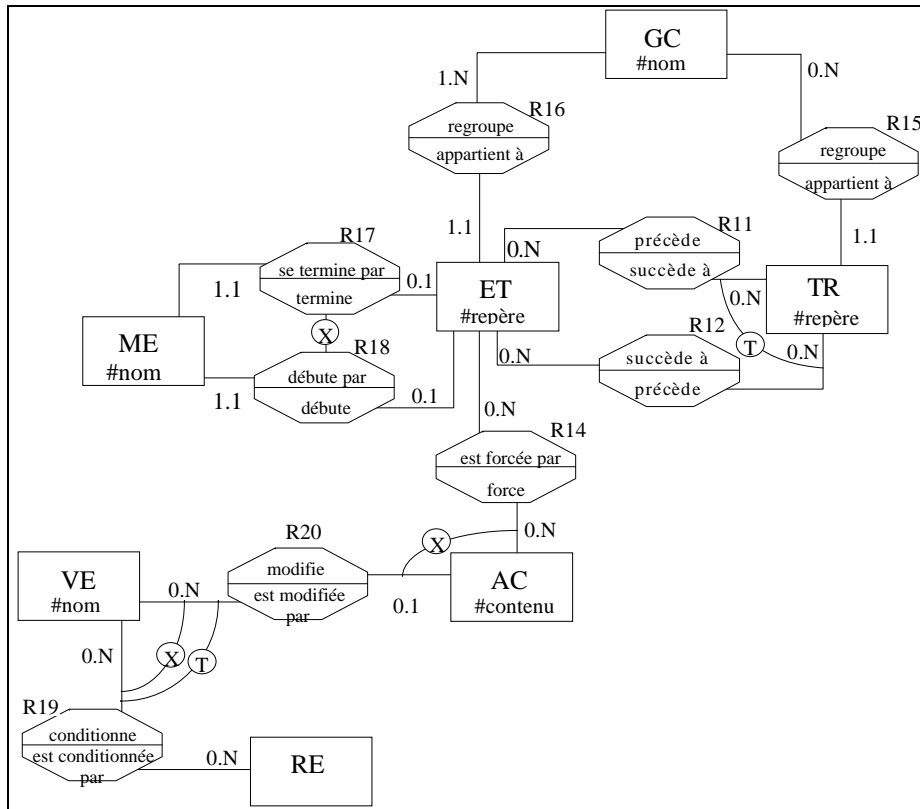


Figure 2. Extrait du méta-modèle statique du Grafcet

3.3. Exemples de vérification

La prise en compte des contraintes exprimées sous forme graphique dans ce méta-modèle permet de vérifier aisément les règles de base du modèle Grafcet. Il est ainsi possible de vérifier que la règle d'alternance étape-transition est respectée (classes de relation R11 et R12), qu'une macro-étape commence et se termine par une et une seule étape (cardinalités des classes de relation R17 et R18) et que ces deux étapes sont différentes (contrainte d'exclusion sur les classes de relation R17 et R18), que le modèle ne comporte pas de transition isolée (cardinalités des classes d'entité ET et TR pour les classes de relation R15 et R16), que toutes les variables d'entrée-sortie du modèle sont utilisées dans les réceptivités ou les actions (contraintes sur rôles d'exclusion et de totalité au niveau des classes de relation R19 et R20), ...

D'autres vérifications plus complexes portant sur l'ensemble du modèle nécessitent l'emploi des contraintes explicites. Nous citerons notamment le fait qu'un ordre de forçage doit être associé à une étape appartenant à un grafcet partiel situé dans le même grafcet global que le grafcet forcé, que toute réceptivité faisant intervenir une variable interne doit être associée à une transition figurant dans le même grafcet global que l'étape dont l'état est caractérisé par cette variable interne. La formalisation de ces contraintes explicites requiert l'introduction d'un langage déclaratif comportant des opérateurs sur classes d'entité et de relation. Il ne nous est pas possible de développer plus ce point dans cet article ; le lecteur intéressé voudra bien se reporter à [COU 97b].

Enfin certaines vérifications nécessitent la définition de traitements sur ce méta-modèle. La vérification du respect de la hiérarchie de forçage [LES 93] en est un exemple.

Nous obtenons donc finalement un modèle grafcet syntaxiquement correct dont il est possible de générer une représentation dans un format neutre, par exemple le format " GR7 " utilisé par l'outil logiciel CADEPA.

4. Apport d'un méta-modèle dynamique à la simulation de grafquets

4.1. Simulation et vérification formelle

Le grafcet conçu étant syntaxiquement correct, il importe ensuite de vérifier ses propriétés dynamiques. Pour ce faire, il est possible de procéder par simulation ou par vérification formelle [DEL 96], [ROU 96]. La simulation, qui consiste à solliciter le modèle grafcet par des séquences d'entrées et à analyser les séquences de sorties produites, a pour inconvénient principal de conduire à une explosion combinatoire du nombre de séquences à générer dans le cas de systèmes complexes. Un second inconvénient, conséquence de l'explosion combinatoire, réside dans le fait que le concepteur doit obligatoirement sélectionner, pour la simulation, un

nombre limité de séquences d'entrées et que la pertinence de son choix n'est pas démontrée a priori.

Malgré ces remarques, la simulation présente cependant l'avantage d'être immédiatement accessible au concepteur, de lui permettre de tester les configurations qu'il juge importantes en termes de vivacité et de sûreté de fonctionnement et peut lui servir à mieux comprendre les causes de certains fonctionnements non désirés détectés par une technique de vérification formelle.

Nous pensons donc que ces deux approches, simulation et vérification formelle, sont complémentaires et doivent être implantées dans un outil d'assistance. Toutes deux nécessitent une modélisation précise des règles d'évolution du grafset que l'on peut exprimer dans un méta-modèle dynamique.

4.2. Présentation du méta-modèle dynamique

Ce méta-modèle [LAM 98a] décrit les règles d'évolution du Grafset, en incluant les ordres de forçage, et les liens de causalité entre les entrées et les sorties du Grafset.

Les entrées, sorties et variables d'activité d'étapes sont représentées par des signaux prenant leurs valeurs sur le corps de Galois d'ordre 2. Les règles d'évolution du Grafset sont alors décrites sous la forme d'équations différentielles pouvant se mettre sous la forme suivante, en ne considérant pas le forçage par souci de simplicité (un exemple faisant intervenir du forçage sera traité à la figure 3) :

$$X_i' = X_i \cdot (R_i \# S_i) + S_i$$

avec :

- X_i signal représentant la variable d'activité de l'étape i
- X_i' signal représentant la dérivée à gauche de X_i
- X_i^- signal représentant le décalage vers la droite de X_i
- S_i signal représentant la condition d'activation de l'étape i
- R_i signal représentant la condition de désactivation de l'étape i
- ., # et + opérateurs produit galoisien (ET logique), OU logique et somme galoisienne (OU exclusif logique),

la condition initiale étant fonction de la nature de l'étape i (étape initiale ou non).

Nous adoptons d'autre part, pour exprimer les relations du modèle avec son environnement, une représentation à deux échelles de temps, interne et externe au modèle, et considérons que toutes les évolutions du modèle sont possibles entre deux variations consécutives des entrées et que les sorties ne sont émises que lorsqu'une situation stable est atteinte.

4.3. Simulation

Une phase de simulation consiste à solliciter le grafctet dont on teste les propriétés dynamiques par une séquence d'entrées définie par l'utilisateur et à déduire les réponses du modèle en appliquant ses règles d'évolution propres déduites du méta-modèle dynamique. Lors de cette évaluation, nous adoptons les postulats temporels du Grafctet décrits dans [LHO 97] ; en particulier les sorties associées à des étapes instables ne sont pas émises et les cas d'instabilité totale sont détectés. La définition des séquences d'entrées a nécessité le développement d'un éditeur de chronogrammes, qui est utilisé également pour l'affichage des séquences de sorties. En considérant l'exemple de la figure 3 on trouvera ci-après les équations d'évolution relatives à ce modèle (O_E : origine temps externe ; X_{issf} : état d'activité de l'étape i sans considérer le forçage) ; la figure 4 montre l'exemple d'une séquence d'entrées appliquée à ce modèle et la séquence de sorties calculée par notre outil.

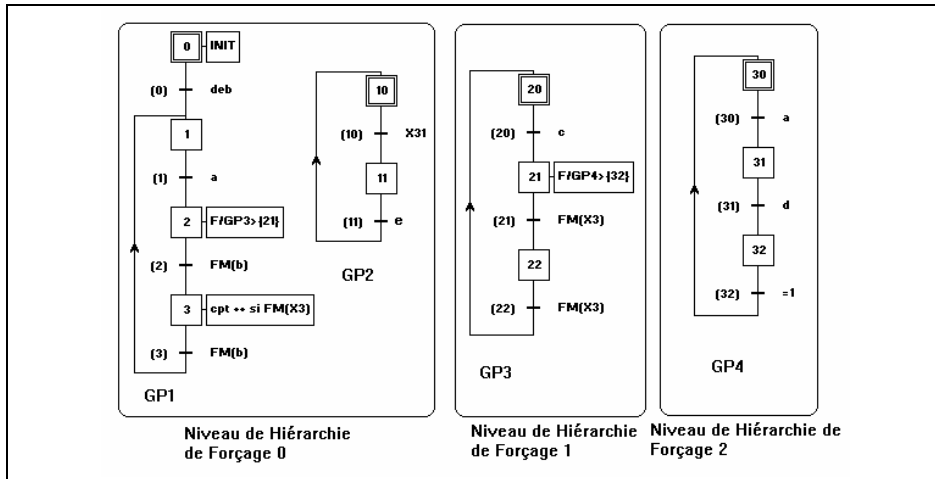


Figure 3. Exemple de modèle grafctet

$$\begin{aligned}
 X'_0 &= deb.X_0 \text{ et } X_0(O_E) = 1 ; & X'_3 &= b.b'(X_2 + X_3) \text{ et } X_3(O_E) = 0 \\
 X'_1 &= a.X_1 + deb.X_0 + b.b'.X_3 \text{ et } X_1(O_E) = 0 & X'_{10} &= e.X_{11} + X_{31}.X_{10} \text{ et } X_{10}(O_E) = 1 \\
 X'_2 &= a.X_1 + b.b'.X_2 \text{ et } X_2(O_E) = 0 & X'_{11} &= e.X_{11} + X_{31}.X_{10} \text{ et } X_{11}(O_E) = 0 \\
 X_{20} &= (1^* + X_2).(X_{20})_{ssf} \text{ avec } (X_{20})'_{ssf} = X_3.(X_3)'.X_{22} + c.(X_{20})_{ssf} \text{ et } (X_{20})_{ssf}(O_E) = 1 \\
 X_{21} &= X_2 \# (X_{21})_{ssf} \text{ avec } (X_{21})'_{ssf} = X_3.(X_3)'.(X_{21})_{ssf} + c.X_{20} \text{ et } (X_{21})_{ssf}(O_E) = 0 \\
 X_{22} &= (1^* + X_2).(X_{22})_{ssf} \text{ avec } (X_{22})'_{ssf} = X_3.(X_3)'.(X_{21} + (X_{22})_{ssf}) \text{ et } (X_{22})_{ssf}(O_E) = 0 \\
 X_{30} &= (1^* + X_{21}).(X_{30})_{ssf} \text{ avec } (X_{30})'_{ssf} = X_{32} + a.(X_{30})_{ssf} \text{ et } (X_{30})_{ssf}(O_E) = 1 \\
 X_{31} &= (1^* + X_{21}).(X_{31})_{ssf} \text{ avec } (X_{31})'_{ssf} = a.X_{30} + d.(X_{31})_{ssf} \text{ et } (X_{31})_{ssf}(O_E) = 0 \\
 X_{32} &= X_{21} \# (X_{32})_{ssf} \text{ avec } (X_{32})'_{ssf} = (X_{32})_{ssf} + d.X_{31} \text{ et } (X_{32})_{ssf}(O_E) = 0
 \end{aligned}$$

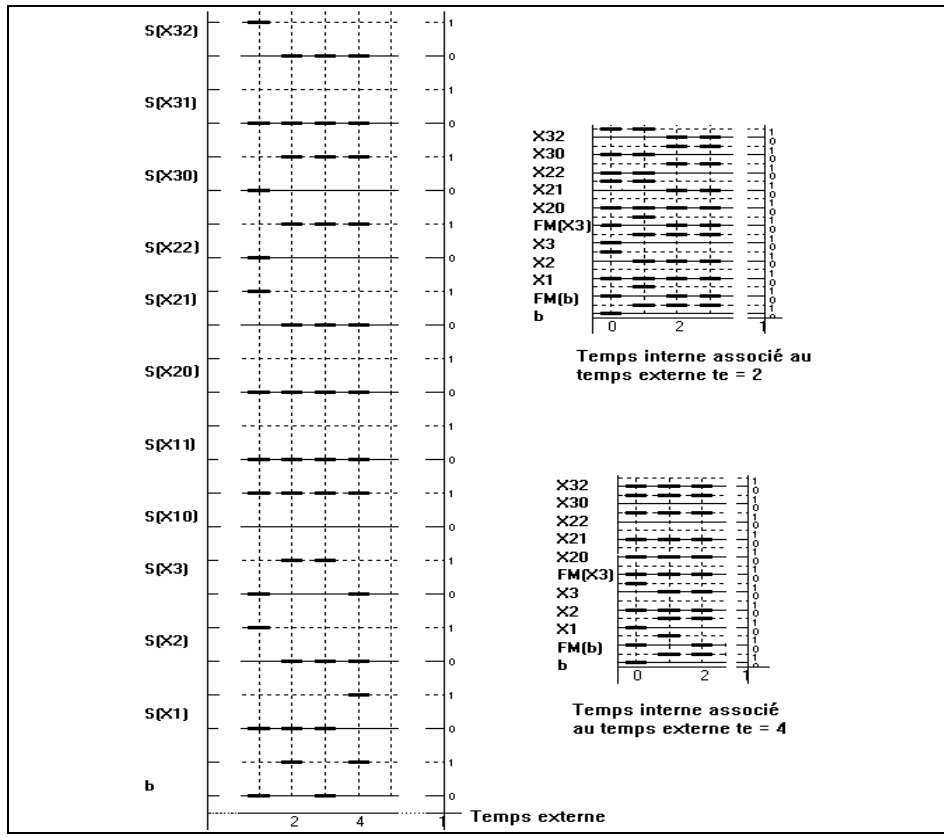


Figure 4. Résultats de simulation sur l'exemple de la figure 3

5. Conclusion

L'ensemble des travaux présentés a donné lieu au développement du logiciel SAGITAL, écrit en C/C++ et fonctionnant sous Windows, permettant l'édition d'un grafset guidée par la syntaxe, la vérification syntaxique de l'ensemble du modèle grafset et la simulation de ses évolutions.

De nombreux travaux de développement restent à réaliser afin de disposer d'un logiciel possédant toutes les fonctionnalités souhaitables. Nous envisageons notamment l'adjonction de modules logiciels permettant la vérification des propriétés dynamiques d'un grafset et sa validation en intégrant dans le logiciel les travaux présentés dans [LAM 98b]. Le lien avec des outils de programmation d'automates, bien que partiellement réalisé par utilisation de la représentation dans un format neutre comme indiqué au paragraphe 3.3, doit également être développé.

Nous pensons en tout cas que la structure de notre logiciel, basée sur des méta-modèles bien formalisés, facilitera ces extensions et permettra d'autre part une intégration aisée avec d'autres outils logiciels de conception d'automatismes.

6. Bibliographie

- [COU 97a] COUFFIN F., “Modèle de données de référence et processus de spécialisation pour l’intégration des activités de conception en génie automatique”, Thèse de doctorat, Ecole Normale Supérieure de Cachan.
- [COU 97b] COUFFIN F., LAMPERIERE S., FAURE J.M., “Contribution to the Grafcet formalisation : a static meta-model proposition”, *Journal Européen des Systèmes Automatisés (JESA)*, vol. 31, n°4/1997, pp. 645-667.
- [DEL 96] DE LOOR P., “Du TTM/RTTL pour la validation des systèmes commandés par Grafcet”, Thèse de doctorat, Université de Reims Champagne Ardennes.
- [FRA 97] FRACHET J.P., LAMPERIERE S., FAURE J.M., “Modelling discrete events systems behaviour using the hyperfinite signal”, *Journal Européen des Systèmes Automatisés (JESA)*, vol. 31, n°3/1997, pp. 453-470.
- [IEC 88] *Preparation of function charts for control systems*, International Standard IEC 848, 1988.
- [LAM 98a] LAMPERIERE-COUFFIN S., “De la vérification de cahier des charges des systèmes à événements discrets à la validation des spécifications décrites en Grafcet”, Thèse de doctorat, Ecole Normale Supérieure de Cachan.
- [LAM 98b] LAMPERIERE S., FAURE J.M., COUFFIN F., “Preuve de propriétés dynamiques de grafquets à partir de leur description structurelle”, *3^{ème} Conférence Internationale ADPM’98 "Les systèmes dynamiques hybrides"*, Reims, France, 19-20 mars 1998.
- [LES 93] LESAGE J.J., ROUSSEL J.M., “Hierarchical approach to GRAFCET using forcing order”, *Revue Automatique Productique Informatique Industrielle*, vol. 27, n°1, pp. 127-141, février 1993.
- [LHO 97] LHOSTE P., ET AL., “Comportement temporel du GRAFCET”, *Journal Européen des Systèmes Automatisés (JESA)*, vol. 31, n°4/1997, pp. 695-711.
- [MOR 91] MOREJON J., OUDRHIRI R., DE GAUDEMONT M., NEGROS P., “GraphOR : A Meta Design Tool”, *Entity-Relationship Approach Conference : The Core of Conceptual Modelling*. pp.43-56, H. Kangassalo Editor, Elsevier Science Publishers B.V. (North-Holland), 1991.
- [REV 95] REVAULT N., SAHRAOUI H. A., BLAIN G., PERROT J.F., “A MetaModeling technique : the METAGEN system”, *TOOLS EUROPE’95 Conference: TOOLS 16*, Prentice Hall. pp. 127-139, Versailles, France, mars 1995.
- [ROU 96] ROUSSEL J.M. ET LESAGE J.J., “Validation and verification of grafquets using finite state machine”, *Conférence IMACS-IEEE Computational Engineering in Systems Applications (CESA’96), Symposium on Discrete Events and Manufacturing Systems*, Lille, France, 9-12 juillet 1996.
- [STE 93] STEELE P. M., ZASLAVSKY A. B., “The Role of Meta Models in Federating System Modelling Techniques”, *12th international conference Entity-Relationship Approach*, Springer-Verlag, pp. 315-326, Arlington, USA, december 1993.