CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Scientific Publications of the University of Toulouse II Le Mirail

# Caractérisation de la reconfiguration dynamique des architectures logicielles par les grammaires de graphe

Cédric Eichler, Ismael Bouassida Rodriguez, Khalil Drira, Thierry Monteil, Patricia Stolf

**To cite this version:**

Cédric Eichler, Ismael Bouassida Rodriguez, Khalil Drira, Thierry Monteil, Patricia Stolf. Caractérisation de la reconfiguration dynamique des architectures logicielles par les grammaires de graphe. Conférence sur les Architectures Logicielles (CAL 2012), May 2012, Montpellier, France. pp.58-68, 2012. <hal-01228323>

**HAL Id: hal-01228323**
**https://hal.archives-ouvertes.fr/hal-01228323**

Submitted on 12 Nov 2015

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Caractérisation de la reconfiguration dynamique des architectures logicielles par les grammaires de graphe

Cédric Eichler
CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
IRIT ; 118 Route de Narbonne, F-31062 Toulouse, France
Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France
ceichler@laas.fr

Ismael Bouassida Rodriguez
ReDCAD ; ENIS, B.P. 1173, 3038 Sfax, Tunisia
Université de Sfax ; PB 1173 Sfax, Tunisie
bouassida@redcad.org

Khalil Drira
CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France
khalil@laas.fr

Thierry Monteil
CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France
monteil@laas.fr

Patricia Stolf
IRIT ; 118 Route de Narbonne, F-31062 Toulouse, France
Université Paul Sabatier, F-31062 Toulouse, France
stolf@irit.fr

## ABSTRACT

L'adaptabilité automatique d'une application est devenue impérative dans un contexte dynamique de systèmes complexes fortement distribués et contenant un grand nombre de composants. La résolution de cette problématique peut être obtenue grâce à l'adaptation structurelle impliquant une reconfiguration de la structure architecturale de l'application. Nous présentons dans cet article un formalisme permettant, en utilisant les grammaires de graphes, de caractériser le style architectural d'une application et les transformations applicables lors de son exécution.

## Keywords

adaptation structurelle, grammaires de graphe, reconfiguration

## 1. INTRODUCTION

L'utilisation de graphes permet de décrire formellement un large panel de structures de manière simple et efficace. Dans le formalisme Abstract Component Graph (ACG) décrit dans [7] que nous traitons ici, le graphe est constitué d'un ensemble de nœuds marqués, et d'un ensemble d'arcs marqués et orientés qui relient une paire de nœuds. Les nœuds du graphe représentent les composants logiciels, et les arcs représentent les liens entre ces composants. Ces liens peuvent représenter des relations de communication, de contrôle ou de composition. La modification de l'architecture est spécifiée par une règle de transformation de graphe qui conditionne cette évolution. Une telle représentation permet de spécifier les propriétés et les contraintes intrinsèques d'un système, offrant ainsi des moyens formels de vérification et de raisonnement sur celles-ci [10, 4].

C'est dans le contexte des architectures logicielles dynamiques que nous situons notre travail. Ce genre d'architectures correspond à des applications dont les composants sont créés, interconnectés, ou supprimés pendant l'exécution. Ce caractère dynamique répond à des contraintes liées à des variations de capacités de communication, de calcul et d'énergie, ainsi que des évolutions dans la nature des activités qu'elles soutiennent. Le changement peut également résulter de la mobilité des utilisateurs. Le caractère dynamique des architectures implique des difficultés supplémentaires pour la description. Pour les architectures statiques, cette description est réalisée via la déclaration des instances des composants, et de liens d'interconnexion. Cette approche paraît inappropriée puisque la structure même de l'architecture change. De nombreux travaux ont été réalisés afin de décrire les architectures dynamiques, et ont donné naissance à plusieurs approches [8, 1]. Afin de garantir la fiabilité des mises à jour de l'architecture nous aurons recours à des techniques formelles. En particulier, les grammaires de graphes représentent un moyen avec un pouvoir expressif assez puissant pour spécifier les aspects statiques et dynamiques des architectures.

Nous retrouvons dans la littérature différents travaux basés sur les graphes qui traitent des architectures logicielles. Nous pouvons citer principalement les travaux de Le Metayer

[9] qui constituent probablement les premiers travaux de description basés sur les graphes. Le modèle décrit par Le Métayer est structuré en deux niveaux, le premier décrit l'architecture sous la forme d'un graphe, et le second décrit les styles architecturaux par une grammaire de graphes. Le changement de l'architecture est décrit par des règles de transformation de graphes. Le modèle définit une approche formelle basée sur un algorithme de vérification permettant de vérifier a priori et de manière statique la consistance des règles de l'évolution de l'architecture. Cette approche permet de prouver que les contraintes considérées par la description de l'architecture sont préservées par ces règles.

Dans cet article, nous présentons des formalismes permettant à la fois de décrire un style architectural, les instances de ce dernier, et leur évolution structurelle. Ceux-ci seront appliqués à un cas d'étude, les opérations d'intervention d'urgence. Le reste de l'article est organisé comme suit : la section 2 présente le modèle des règles de transformation de graphe. La section 3 définit notre cas d'étude. La section 4 décrit la caractérisation d'un style architectural par une grammaire de graphe (GG). La section 5 traite de la définition d'un ensemble de règles de transformation permettant de modéliser le caractère dynamique des applications considérées.

## 2. RÈGLES DE RÉÉCRITURE DE GRAPHE

Dans cette section, nous abordons le problème des arcs suspendus intrinsèque à la transformation de graphe. Nous présentons diverses approches de modélisation de règles de transformation de graphe, avant de caractériser notre modèle.

### 2.1 Approche basique des transformations de graphes, problème des arcs suspendus

L'approche la plus simple pour transformer un graphe hôte G en un graphe G' est de remplacer un sous-graphe L de G, en un graphe fils R. Dans ce cas, une règle de réécriture est décrite par la paire de graphes (L,R). Son application à un graphe G est subordonnée à l'existence d'au moins une occurrence de L dans G et a pour conséquence la suppression d'une occurrence de L du graphe G et son remplacement par une copie (isomorphe) de R. Cette suppression peut toutefois entraîner l'apparition d'arcs sans nœud de départ ou d'arrivée ou sans aucun des deux.
On parle dans ce cas d'arcs suspendus. Deux approches résolvant cette problématique seront présentées par la suite, chacune impliquant des choix différents concernant la caractérisation des règles de réécriture et le traitement du cas des arcs suspendus [5].

### 2.2 L'approche Simple Push Out

Dans l'approche SPO, une règle de réécriture est comme précédemment décrite par une paire de graphes (L,R). Cette règle est applicable à un graphe G s'il y a au moins une occurrence de L dans G. L'application de la règle implique la suppression du graphe correspondant à Del = (L\(L∩R)), l'effacement des arcs suspendus apparus lors de l'étape précédente, puis le rajout du graphe correspondant à Add = (R\(L∩ R)).

### 2.3 L'approche Double Push Out

L'approche DPO considère une structure plus riche. Une règle est spécifiée par un 3-uplet (L,K,R) où L et R gardent la même signification que dans l'approche SPO. K permet de déterminer clairement la partie à préserver après l'application de la règle. Pour que la règle soit applicable, on précise en sus de la présence de L dans G une contrainte supplémentaire appelée condition de suspension. Celle-ci stipule que la règle ne peut être appliquée uniquement si cela n'entraîne pas l'apparition d'arc(s) suspendu(s). Si ces deux conditions sont remplies, l'application de la règle implique la suppression du graphe correspondant à Del = (L\K) et le rajout d'une copie du graphe Add = (R\K).

### 2.4 Condition d'application négative

Pour déterminer l'applicabilité d'une règle de transformation, les approches présentées précédemment se basent, entre autres, sur l'existence d'une occurrence du sous graphe L dans le graphe hôte G. Dans certains cas, il est nécessaire de pouvoir exprimer des conditions supplémentaires permettant de spécifier des conditions relatives à l'absence d'une occurrence d'un certain sous graphe dans G. Ces dernières sont appelées restrictions ou conditions d'application négatives (Negative Application Conditions ou NACs).

Ces contraintes peuvent être intégrées dans les approches SPO ou DPO en rajoutant un champ NAC dans la définition de la règle. Elles impactent uniquement sur l'applicabilité de la réécriture et non sur l'étape de transformation proprement dite.

Une règle SPO/NAC aura la structure (L,R,NAC) et sera applicable si les conditions de présence et d'application négative sont remplies. Une règle DPO/NAC aura la structure (L,K,R,NAC) et pourra être appliquée si les conditions de présence, de suspension et d'application négative sont vérifiées.

### 2.5 Notre modèle de transformation de graphe

La méthode utilisée est basée sur l'approche Double PushOut avec conditions d'application négative. Une règle est décrite comme précédemment par un 4-uplet (L,K,R,NACs) où
- L, lié à la partie à retirer, est le sous-graphe à trouver dans le graphe hôte,
- R est lié à la partie à ajouter,
- NACs est un ensemble de graphes spécifiant les conditions d'application négative tel que ∀ NAC ∈ NACs, L ⊂ NAC,
- K est contenu dans L et spécifie la partie invariante de la règle, que nous noterons Inv,
- (R\K) est le sous-graphe à ajouter dans le graphe hôte, cette partie est appelée Add.
- (L\K) est le sous-graphe à supprimer dans le graphe hôte, cette partie est appelée Del.

Ce modèle diffère du formalisme DPO intégrant NAC classique vis à vis de la multiplicité des conditions d'application négative et de la gestion des arcs suspendus qui est la même que dans l'approche SPO (i.e. suppression). Par conséquent, la condition de suspension n'est plus considérée et une règle est applicable sur un graphe G s'il y a une occurrence (isomorphe) de L dans G en l'absence de chacun des élé-

ments de NACs. Son application consiste à retirer la partie Del, à ajouter la partie Add et à effacer les éventuels arcs suspendus.

Par convention un lien labellisé *var* désigne tout lien possible (y compris un lien vide, inexistant) et un attribut constant est désigné entre guillemets.

La figure 1 présente un exemple de l'application d'une règle selon l'approche présentée ci-dessus. La condition d'application négative est automatiquement remplie car vide. De plus, une occurrence de L étant présente dans G1, la règle est applicable à G1. La partie Del est supprimée ce qui entraîne l'apparition d'un unique arc suspendu (qui reliait le nœud 4 au nœud 3). Cet arc est à son tour effacé, puis la partie Add est ajoutée.

# 3. OPÉRATION D'INTERVENTION D'URGENCE

Afin d'appliquer les modèles précédemment décrits, nous allons nous intéresser aux situations d'opérations d'intervention d'urgence (OIU) en nous basant sur les travaux [6] et [2]. Ces situations impliquent des composants architecturaux interagissant de manière organisée et présentant une structure hiérarchique. En cours d'intervention, des participants et des connexions sont susceptibles d'être ajoutés ou supprimés.

## 3.1 Description Générale

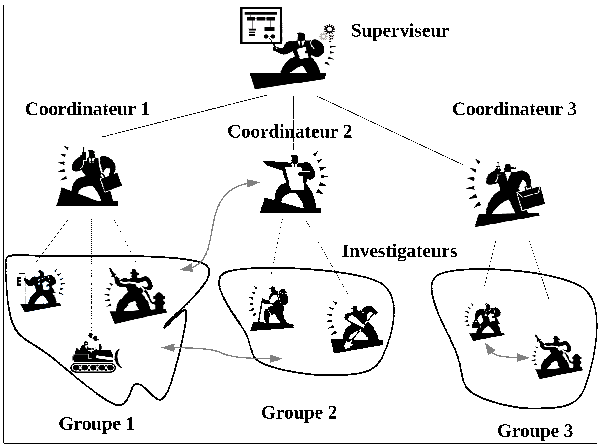


**Figure 2: Opération d'intervention d'urgence**

Les OIU impliquent des groupes structurés de robots ou de personnels militaires coopèrant pour la réalisation d'une mission commune. Les éléments de l'architecture possèdent différents rôles et disposent de ressources inégales en capacités de communication, en CPU et en énergie. Ils sont déployés sur des machines fixes et mobiles et communiquent via des réseaux filaires et sans fil. L'activité comporte deux phases d'exécution correspondant à une phase d'exploration du champ d'investigation et à une phase d'action faisant suite à la découverte d'une situation critique. Les rôles dans l'application et la structure des interactions entre participants évoluent d'une étape à l'autre pour s'adapter à l' évolution des objectifs applicatifs et du contexte d'exécution.

Nous nous intéressons ici au niveau d'abstraction relatif à la couche protocolaire applicative, représenté dans la figure 2. La collaboration est basée sur l'échange de données entre participants, notamment des données d'observation et des données d'analyse, produites périodiquement ou immédiatement après un événement particulier. Une équipe d'intervention d'urgence est ainsi constituée de participants ayant différents rôles : un superviseur de la mission, plusieurs coordinateurs, et plusieurs sections d'investigateurs. Le superviseur de la mission gère l'ensemble des coordinateurs et chaque coordinateur dirige une section d'investigateurs. À chaque rôle correspondent les fonctions suivantes :

- Un superviseur a pour fonction de diriger et d'autoriser les actions qui sont déléguées aux coordinateurs. Le superviseur est l'entité qui supervise toute l'application, il attend des rapports de tous ses coordinateurs qui synthétisent le contexte courant de l'application, et l'informent du déroulement de la mission. Le superviseur est déployé sur une machine fixe, il dispose d'un accès à l'énergie permanent et de capacités de communication et de CPU conséquentes.
- Selon les actions et les objectifs assignés par le superviseur, un coordinateur doit diriger ses investigateurs en leur assignant des tâches à exécuter. Il doit également collecter, interpréter et synthétiser les informations reçues des investigateurs et les diffuser vers le superviseur. Les coordinateurs sont déployés sur des machines mobiles.
- Les investigateurs ont pour fonction d'explorer le champ opérationnel, d'observer, d'analyser et de faire un rapport décrivant la situation aux coordinateurs qui les contrôlent. Ils sont déployés sur des machines mobiles et disposent, donc, de ressources limitées en énergie et en CPU.

Les fonctions assignées aux participants impliquent d'observer en continu (D) le champ d'investigation et de rapporter périodiquement (P) sur ce qui est observé. Les données de retour D sont des données descriptives tandis que les données de retour P sont des données produites, et expriment l'analyse de la situation par un investigateur ou un coordinateur. Le superviseur contrôle l'ensemble de la mission, en décidant des actions à exécuter en fonction des objectifs opérationnels et de l'analyse des retours P transmis par les coordinateurs. Un coordinateur est en charge de la partie de la mission qui lui a été assignée par le superviseur. Il décide localement des actions à exécuter en fonction de l'observation et de l'analyse des données D transmises par les investigateurs. Pour prendre cette décision, le coordinateur peut également utiliser les données P transmises par les investigateurs. Les coordinateurs rapportent l'évolution de la sous mission au superviseur en utilisant des données de retour de type P.

## 3.2 Description des flux de communication de l'opération d'intervention d'urgence

Deux types de flux sont susceptibles d'exister au niveau applicatif du modèle des OIU : les flux de coordination et de coopération.

La coopération a lieu entre différents coordinateurs ou entre investigateurs du même groupe (A2A) ou d'un groupe différent (A2B).

La coordination quant à elle s'effectue entre les investigateurs et un ou plusieurs coordinateur(s) ainsi qu'entre les

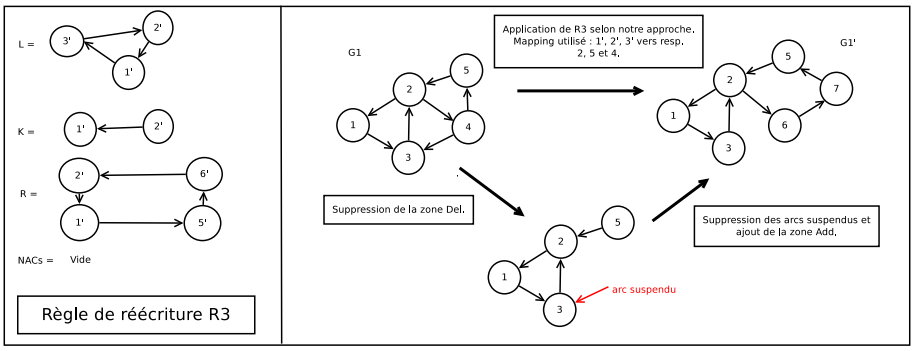


**Figure 1: Application d'une règle de réécriture selon notre approche**

coordinateurs et le superviseur. Des données de différentes natures sont échangées via ces liens :

- information I : Du superviseur vers les coordinateurs et des coordinateurs vers les investigateurs.
- instruction O : Du superviseur vers les coordinateurs et des coordinateurs vers les investigateurs de leurs groupes respectifs.
- observation D : Des investigateurs vers les coordinateurs et, en phase d'action, d'un investigateur vers tous les investigateurs de son groupe.
- rapport P : Des investigateurs vers les coordinateurs et des coordinateurs vers le superviseur.

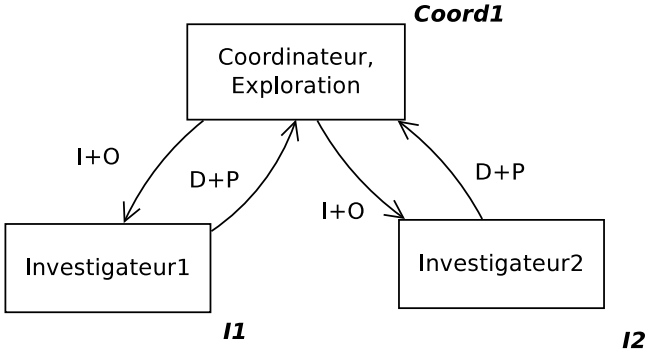


**Figure 3: Situation initiale, groupe minimal en phase d'exploration**

La phase d'exploration se termine quand une situation critique est découverte par un investigateur I1. Cette situation critique implique une reconfiguration de l'architecture et l'application bascule dans une autre phase d'exécution appelée phase d'action. La structure d'un groupe d'investigateurs minimal pour permettre le passage en phase d'action est rappelé dans la figure 3. La reconfiguration touche spécifiquement le coordinateur Coord1 qui contrôle l'investigateur I1 ainsi que tous les investigateurs de Coord1. Les autres coordinateurs et les investigateurs qui leurs sont rattachés ne sont pas affectés par cette reconfiguration. Nous exigeons, pour le passage vers la phase d'action, l'existence d'au moins deux investigateurs. Lors du passage de la phase d'exploration vers la phase d'action, le coordinateur Coord1 ainsi que ses investigateurs réagissent de la manière suivante :

Après avoir découvert la situation critique, I1 continue de remplir les mêmes fonctions que celles qu'il réalisait durant l'étape d'exploration (Observer et Rapporter), mais envoie aussi ses observations D aux autres investigateurs. Il continue d'envoyer les deux types de données (D et P) au coordinateur.

Les autres investigateurs rapportent maintenant seulement les données P au coordinateur ; elles correspondent à des analyses complémentaires des données D transmis par l'investigateur I1. La figure 4 décrit la configuration d'un groupe minimal en phase d'action.

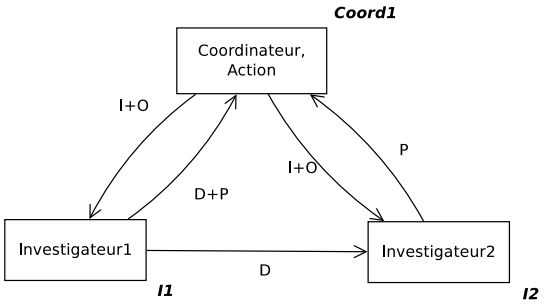


**Figure 4: Groupe minimal après passage en phase d'exploration**

## 3.3 Propriétés architecturales de l'opération d'intervention d'urgence

La description du notre cas d'étude permet de spécifier les propriétés architecturales suivantes, que toute instance consistante doit vérifier :

*Propriété 1.* Une instance de l'architecture ne peut être constituée que de trois types de nœuds : superviseur, coordinateur et investigateur. (p1)

*Propriété 2.* Une instance de l'architecture possède exactement un superviseur. (p2)

*Propriété 3.* Tout coordinateur est contrôlé par un superviseur, c'est à dire que tout coordinateur possède les connexions suivantes : coord $\xrightarrow{\text{“}P\text{”}}$ sup et sup $\xrightarrow{\text{“}I+O\text{”}}$ coord. (p3)

*Propriété 4.* Deux coordinateurs peuvent être liés uniquement par un lien A2A. (p4)

*Propriété 5.* Tout investigateur est contrôlé par un et un seul coordinateur. Une relation de contrôle est symbolisé par un lien $\xrightarrow{\text{“}I+O\text{”}}$. (p5)

*Propriété 6.* Deux investigateurs de groupes différents ne peuvent être liés que par un lien de type A2B. (p6)

*Propriété 7.* Dans un groupe en phase d'exploration, un investigateur envoie des données D et P au coordinateur contrôlant le groupe. (p7)

*Propriété 8.* Dans un groupe en phase d'exploration, deux investigateurs ne peuvent être liés que par un lien de type A2A. (p8)

*Propriété 9.* Tout groupe en phase d'action possède au moins deux investigateurs, dont un unique investigateur critique. Ce dernier envoie des données de type D et P au coordinateur du groupe, et des données de type D à tous les autres investigateurs du groupe. Les autres envoient des données de type D au coordinateur. (p9)

*Propriété 10.* Dans un groupe en phase d'action, un investigateur non-critique ne peut envoyer que des données A2A à un investigateur de son groupe. (p10)

*Propriété 11.* Tout coordinateur ne peut être lié à un investigateur qu'il ne contrôle pas que par une connexion de type I. (p11)

*Propriété 12.* Tout investigateur ne peut être lié à un coordinateur ne le contrôlant pas que par une connexion de type D+P. (p12)

# 4. CARACTÉRISATION DU STYLE ARCHITECTURAL DE L'OPÉRATION D'INTERVENTION D'URGENCE PAR UNE GRAMMAIRE DE GRAPHE

Les GGs s'inspirent des grammaires génératives de Chomsky [3] dans le sens où elles sont également définies par un quadruplet (AX,NT,T,P) où AX représente l'axiome, NT l'ensemble des symboles non terminaux de la grammaire, T l'ensemble de ces symboles terminaux (ici des nœuds de l'arbre) et P l'ensemble des règles de productions. Les règles de productions ne correspondent cependant pas à la définition usuelle des règles de production de grammaires formelles, comme nous l'avons vu dans la section 2.

Un symbole appartenant à NT ou T est noté N(att1,att2,...,attn). Dans les règles de production, ils sont notés $N_i$(att1,att2,...,attn) où i permet d'identifier le nœud. Ces symboles sont partiellement instanciés et, par convention, on notera entre guillemets tout attribut constant.

Par définition, un graphe appartient à un style architectural s'il contient uniquement des nœuds terminaux (éléments de T) et qu'il peut être obtenu à partir de l'axiome AX en appliquant une séquence de règles contenues dans P. Le style architectural des OIU est défini par la GG suivante :

$GG_{OIU}$ = (AX,$NT_{OIU}$,$T_{OIU}$,$P_{OIU}$) avec :
$T_{OIU}$ = {N($X_I$, “Inv”, $N_I$), N($X_{Coord}$, “Coord”, “Action”, $N_{Coord}$), N($X_{Coord}$, “Coord”, “Exploration”, $N_{Coord}$), N($X_{Sup}$, “Sup”)},
$NT_{OIU}$ = {N(“Temp”)}, et
$P_{OIU}$ = {$r_1$,...,$r_{11}$}

Avec :
- $X_y$ l'identifiant du nœud.
- “Inv”, “Coord”, “Sup” le rôle rempli par le nœud.
- “Action” indique que le coordinateur et son groupe d'investigateurs sont en phase d'action.
- N la nature du nœud (robot, pompier...). Ce paramètre sera utilisé pour créer des groupes d'investigateurs homogènes.

Les productions de cette grammaire sont spécifiés ci-après. On peut d'ores et déjà noter au vu de $T_{OIU}$ que p1 est remplie. Dans le cas présent, la GG ne comporte qu'un seul élément non terminal qui peut être considéré comme un “flag” ou marqueur signifiant l'état “en cours de construction” d'une configuration. Pour justifier la stabilité des propriétés architecturales pour les règles de production (hors initialisation et terminaison) on considérera qu'elles seront appliquées à des configurations consistantes au nœud *Temp* près.

*Création d'un superviseur.* Cette règle de production ($r_1$) est la seule applicable à l'axiome, assurant la validité de p2. L'unicité est assurée par le fait que $r_1$ ne peut être appliquée qu'une fois.

$r_1$=(L={AX} ;
K={} ;
R={N1($X_{Sup}$ ,“Sup”), N2(“Temp”)} ;
NACs=∅)

*Ajout d'un coordinateur en phase d'exploration.* Cette règle de production ($r_2$) ajoute un coordinateur en phase d'exploration en le liant au superviseur de manière conforme à p3.

$r_2$=(L={N1($X_{Sup}$,“Sup”), N2(“Temp”)} ;
K={N1($X_{Sup}$,“Sup”), N2(“Temp”)} ;
R\K={N3($X_{Coord}$,“Coord”,“Exploration”,$N_{Coord}$),
N3 $\xrightarrow{\text{“}P\text{”}}$N1, N1 $\xrightarrow{\text{“}I+O\text{”}}$N3} ;
NACs=∅)

*Ajout d'un coordinateur en phase d'action.* La règle $R_3$ entraîne l'ajout d'un groupe d'investigateur minimal en phase d'action. Cet ajout est fait conformément aux propriétés 3, 5 et 9.

$r_3$=(L={N1($X_{Sup}$ ,“Sup” ), N2(“Temp”)} ;
K={N1($X_{Sup}$,“Sup”), N2(“Temp”)} ;
R\K={N3($X_{Coord}$,“Coord”,“Action”,N),
N4($X_{I1}$ ,“Inv”,N), N5($X_{I2}$ “Inv”,N),
N3 $\xrightarrow{\text{“}P\text{”}}$N1, N1 $\xrightarrow{\text{“}I+O\text{”}}$N3,
N4 $\xrightarrow{\text{“}D+P\text{”}}$N3, N3 $\xrightarrow{\text{“}I+O\text{”}}$N4,
N5 $\xrightarrow{\text{“}P\text{”}}$N3, N3 $\xrightarrow{\text{“}I+O\text{”}}$N5, N4 $\xrightarrow{\text{“}D\text{”}}$N5 } ;
NACs=∅)

*Ajout d'un investigateur dans un groupe en phase d'exploration.* La règle de production $r_4$ est appliqué lorsqu'un investigateur rejoint un groupe en phase d'exploration, se liant au coordinateur selon les propriétés 5 et 7.

$r_4$=(L={N1($X_{Coord}$,“Coord”,“Exploration”,N),
N2(“Temp”)} ;
K={N1($X_{Coord}$,“Coord”,“Exploration”,N),
N2(“Temp”)} ;
R\K={N3($X_I$ ,“Inv”,N), N3 $\xrightarrow{\text{“}D+P\text{”}}$N1,
N1 $\xrightarrow{\text{“}I+O\text{”}}$N3} ;
NACs=∅)

*Ajout d'un investigateur dans un groupe en phase d'action.* Cette règle de production ($r_5$) permet à un investigateur de participer à la résolution d'une action critique, se liant au coordinateur du groupe et à l'investigateur critique comme décrit dans les propriétés 5 et 9.

$r_5$=(L={N1($X_{Coord}$,“Coord”,“Action”,N), N2(“Temp”),
N3($X_{I1}$,“Inv”,N), N3 $\xrightarrow{\text{“}D+P\text{”}}$N1, N1 $\xrightarrow{\text{“}I+O\text{”}}$N3} ;
K={N1($X_{Coord}$,“Coord”,“Action”,N), N2(“Temp”),
N3($X_{I1}$,“Inv”,N), N3 $\xrightarrow{\text{“}D+P\text{”}}$N1, N1 $\xrightarrow{\text{“}I+O\text{”}}$N3} ;
R\K={N4($X_{I2}$,“Inv”,N), N4 $\xrightarrow{\text{“}P\text{”}}$N1, N1 $\xrightarrow{\text{“}I+O\text{”}}$N4,
N3 $\xrightarrow{\text{“}D\text{”}}$N4} ;
NACs=∅)

*Liens entre coordinateurs.* Dans la règle $r_6$, le lien variable est considéré comme valide puisque les règles sont appliquées à des instances consistantes au nœud *Temp* près. L'ajout du lien A2A entre deux coordinateurs est conforme à p4.

$r_6$ = (L={N1($X_{Coord1}$,“Coord”,$Phase_{Coord1}$,$N_{Coord1}$),
N2($X_{Coord2}$,“Coord”,$Phase_{Coord2}$,$N_{Coord2}$),
N2 $\xrightarrow{var}$N1, N3(“Temp”)} ;
K={N1($X_{Coord1}$,“Coord”,$Phase_{Coord1}$,$N_{Coord1}$),
N2($X_{Coord2}$,“Coord”,$Phase_{Coord2}$,$N_{Coord2}$),
N2 $\xrightarrow{var}$N1,N3(“Temp”)} ;
R\K={N1 $\xrightarrow{\text{“}A2A\text{”}}$N2} ;
NACs=∅)

*Liens entre investigateurs A2A.* Via la règle de production $r_7$, l'ajout d'une connexion A2A entre deux investigateurs d'un même groupe est bien en accord avec les propriétés 8 et 10. En effet, l'investigateur N1 n'envoie pas de donnés à N2, or, la configuration étant supposée valide, il ne peut être critique, l'établissement du lien A2A ne peut donc violer ces deux propriétés.

$r_7$=(L={N1($X_I$,“Inv”,N),N2($X_{I2}$,“Inv”,N),N2 $\xrightarrow{var}$N1,
N3(“Temp”)} ;
K={N1($X_I$,“Inv”,N),N2($X_{I2}$,“Inv”,N),N2 $\xrightarrow{var}$N1,
N3(“Temp”)} ;
R\K={N1 $\xrightarrow{\text{“}A2A\text{”}}$N2} ;
NACs=∅)

*Liens entre investigateurs A2B.* En appliquant la règle de production $r_8$, deux investigateurs de groupes différents (on s'assure que N1 ≠ 2) peuvent être connectés par un lien A2B quelque soit la situation, cf p6.

$r_8$=(L={N1($X_I$ ,“Inv”,“N1”),N2($X_{I2}$,“Inv”,“N2”),
N2 $\xrightarrow{var}$N1,N3(“Temp”)} ;
K={N1($X_I$,“Inv”,“N1”), N2($X_{I2}$,“Inv”,“N2”),
N2 $\xrightarrow{var}$N1,N3(“Temp”)} ;
R\K={N1 $\xrightarrow{\text{“}A2B\text{”}}$N2} ;
NACs=∅)

*Liens d'un investigateur vers un coordinateur n'étant pas le chef de son groupe.* En vérifiant que N1 ≠ N2, le coordinateur ne contrôle pas l'investigateur, et la connexion est conforme à p12. La règle de production $r_9$, assure le lien d'un investigateur vers un coordinateur qui n'est pas le chef de son groupe.

$r_9$=(L={N1($X_{Coord}$,“Coord”,$Phase_{Coord}$,“N1”),
N2($X_I$,“Inv”,“N2”),N1 $\xrightarrow{var}$N2,N3(“Temp”)} ;
K={N1($X_{Coord}$,“Coord”,$Phase_{Coord}$”,“N1”),
N1 $\xrightarrow{var}$N2,N3(“Temp”)} ;
R\K={ N2 $\xrightarrow{\text{“}D+P\text{”}}$ $N1$} ;
NACs=∅)

*Liens d'un coordinateur vers un investigateur qu'il ne contrôle pas.* En vérifiant que N1 ≠ N2, le coordinateur ne contrôle pas l'investigateur, et la connexion est conforme à p11. La règle de production $r_{10}$ assure le lien d'un coordi-

nateur vers un investigateur qu'il ne contrôle pas.

$r_{10}$=(L={N1($X_{Coord}$,"Coord",$Phase_{Coord}$,"N1"),
N2($X_I$,"Inv","N2"),N2$\xrightarrow{var}$N1,N3("Temp")};
K={N1($X_{Coord}$,"Coord",$Phase_{Coord}$,"N1"),N2$\xrightarrow{var}$N1,
N3("Temp")};
R\K={ N1 $\xrightarrow{\text{"}I\text{"}}$ $N2$};
NACs=$\emptyset$)

*Suppression du nœud non terminal.* La terminaison ne fait que supprimer le nœud temporaire à travers $r_{11}$, et conserve donc toutes les propriétés architecturales.

$r_{11}$=(L={N("Temp")},
K={};
R={};
NACs=$\emptyset$)

# 5. MODÉLISATION DU CARACTÈRE DYNAMIQUE DES APPLICATIONS CONSIDÉRÉES

Le caractère dynamique d'une application, en particulier ses propriétés d'adaptation structurelle, est modélisé par une grammaire de graphe réduite au triplet (NT,T,P) où NT, T et P ont le même sens que précédemment.

Les règles de reconfiguration ou règles de dynamisme contenues dans P modélisent alors toutes les reconfigurations architecturales possibles lors de l'exécution de l'application et uniquement celles-ci. Elles forment un ensemble de descripteurs formels de reconfigurations élémentaires et décrivent ainsi les propriétés de dynamisme du système. Ces règles se décomposent en deux catégories : règles "basiques" déduites des règles de génération et règles spécifiques au déroulement de l'application. Dans le cas des OIU, ces règles spécifiques permettent de passer d'une phase à l'autre.

Le dynamisme des OIU est représenté par le triplet suivant :
$GG_{OIU,reconfiguration}$=($NT_{OIU,reconfiguration}$,
$T_{OIU,reconfiguration}$,$P_{OIU,reconfiguration}$) avec :
$T_{OIU,reconfiguration}$ = $T_{OIU}$,
$NT_{OIU,reconfiguration}$={N("Temp"),N("TempAE"),
N("TempEA")} et
$P_{OIU,reconfiguration}$ = {$d_1$,...,$d_{24}$ }.

Cette grammaire de transformation de graphe comprend des nœuds non terminaux nécessaires lors du passage de l'exploration à l'action et inversement, comme nous le verrons par la suite.

## 5.1 Règles basiques

Les règles de transformation basiques liées au modèle de reconfiguration sont construites à partir du modèle de génération. Pour une GG de génération n'ayant qu'un seul élément temporaire n'étant utilisé que comme un "flag", pour chaque règle r $\in P_{generation}$ autre que l'initialisation et la terminaison, une règle équivalente d' $\in P_{reconfiguration}$ et réciproque d" $\in P_{reconfiguration}$ sont définies comme suit :

- Inv(d') = Inv(p)\N("Temp"), Add(d') = Add(p), Del(d') = Del(p) et NACs(d') = NACs(p) $\cup$ $NT_{reconfiguration}$.
- Add(d") = Del(p) et Del(d") = Add(p). Les conditions d'applications (et donc Inv(d") et Nac(d")) de la règle dépendent du style architectural. On souhaite en particulier que d" soit au moins applicable directement après l'applicaction de r suivie de l'application de la règle de terminaison. Dans le cas de la suppression d'un lien ou d'un nœud optionnel, NACs(p") = $NT_{reconfiguration}$ et Inv(p") = Inv(p)\N("Temp").

Cette notion de règle équivalente et réciproque découle du fait que si un composant logiciel ou un lien est ajoutable lors de la génération, on souhaite pouvoir l'ajouter ou le supprimer lors de la reconfiguration. Nous donnerons ci dessous quelques exemples de règles de reconfiguration pour les OIU.

Reprenons la définition de la règle de production $r_2$ précédemment définie décrivant l'ajout d'un coordinateur en phase d'exploration. $d_1$ et $d_2$ règles de dynamisme respectivement équivalente à et réciproque de $r_2$ sont alors construites tel qu'expliqué précédemment.

*Équivalence, ajout d'un coordinateur en phase d'exploration.* La règle de reconfiguration $q_1$ décrit l'ajout d'un coordinateur en phase d'exploration en le liant au superviseur de manière conforme à la propriété 3. Elle vérifie Inv($d_1$) = Inv($r_2$)\N("Temp"), Add($d_1$) = Add($r_2$) et Del($d_1$) = Del($r_2$). On simplifie NACs($d_1$) car la présence d'un nœud N("TempAE") ou N("TempEA") implique la présence d'un nœud N("Temp").

$d_1$=(L={N1($X_{Sup}$,"Sup")};
K={N1($X_{Sup}$,"Sup")};
R\K={N3($X_{Coord}$,"Coord","Exploration","$N_{Coord}$"),
N3 $\xrightarrow{\text{"}P\text{"}}$N1,N1 $\xrightarrow{\text{"}I+O\text{"}}$N3};
NACs={NAC1};
NAC1\L={N2("Temp")})

*Réciproque, suppression d'un coordinateur en phase d'exploration.* $d_2$ implique la suppression d'un coordinateur en phase d'exploration. Elle vérifie Add($d_2$) = Del($r_2$) et Del($d_2$) = Add($r_2$). Son applicabilité est subordonnée à l'absence d'investigateurs sous contrôle dudit coordinateur, afin de conserver la propriété 5, d'où la définition de NACs($d_2$) intégrant NAC1. On note que si le coordinateur a été ajouté juste avant la terminaison, il n'a pas d'investigateurs sous ses ordres, et que $d_2$ est bien applicable.

$d_2$=(L={N1($X_{Coord}$,"Coord","Exploration",N)};
K={};
R={};
NAC={NAC1,NAC2};
NAC1\L={N2($X_I$,"Inv",N),N1$\xrightarrow{\text{"}I+O\text{"}}$N2,
N2$\xrightarrow{\text{"}D+P\text{"}}$N1};
NAC2\L={N3("Temp")})

Prenons à présent un exemple de lien optionnel, comme un lien entre coordinateurs, dont l'ajout est décrit par la règle $r_6$. $d_9$ et $d_{10}$ règles de dynamisme respectivement équivalente à et réciproque de $r_6$ sont alors construites tel qu'expliqué précédemment.

*Équivalente, ajout d'un lien entre coordinateurs.* La règle de production $d_9$ entraîne l'ajout d'un lien A2A entre deux coordinateurs. Celui-ci est conforme à la propriété 4 et peut toujours être effectué. Elle vérifie $Inv(d_9) = Inv(r_6)\backslash N(\text{“Temp”})$, $Add(d_9) = Add(r_6)$ et $Del(d_9) = Del(r_6)$.

$d_9$=(L={N1($X_{Coord1}$,“Coord”,$Phase_{Coord1}$,$N_{Coord1}$),
N2($X_{Coord2}$,“Coord”,$Phase_{Coord2}$,$N_{Coord2}$),
N2$\xrightarrow{var}$N1};
K={N1($X_{Coord1}$,“Coord”,$Phase_{Coord1}$,$N_{Coord1}$),
N2($X_{Coord2}$,“Coord”,$Phase_{Coord2}$,$N_{Coord2}$),
N2$\xrightarrow{var}$N1};
R\K={N1 $\xrightarrow{\text{“A2A”}}$ N2};
NACs={NAC1};
NAC1\L={N3(“Temp”)})

*Réciproque, suppression d'un lien entre coordinateurs.* À tout moment, un lien A2A peut être rompu via la règle de production $d_{10}$. Elle est telle que $Add(d_{10}) = Del(r_6)$ et Del(d") = $Add(r_6)$. En sus, considérant que nous sommes dans le cas de la suppression d'un lien optionnel, $NACs(d_{10}) = NT_{reconfiguration}$ et $Inv(d_{10}) = Inv(r_6)\backslash N(\text{“Temp”})$. $NACs(d_{10})$ est ici simplifié pour les raisons antérieurement exposées.

$d_{10}$=(L={N1($X_{Coord1}$,“Coord”,$Phase_{Coord1}$,$N_{Coord1}$),
N2($X_{Coord2}$,“Coord”,$Phase_{Coord2}$,$N_{Coord2}$),
N1$\xrightarrow{\text{“A2A”}}$N2,N2$\xrightarrow{var}$N1};
K={N1($X_{Coord1}$,“Coord”,$Phase_{Coord1}$,$N_{Coord1}$),
N2($X_{Coord2}$,“Coord”,$Phase_{Coord2}$,$N_{Coord2}$),
N2$\xrightarrow{var}$ N1};
R\K={};
NACs={NAC1};
NAC1\L={N3(“Temp”)})

Ainsi, toutes les règles de reconfiguration basiques peuvent être construites à partir des règles de génération. Afin de modéliser plus finement le déroulement réel de l'application, il est parfois intéressant d'introduire des règles spécifiques.

## 5.2 Règles spécifiques

Dans le cas des OIU, les règles spécifiques décrivent le changement de phase d'un groupe d'investigation. Ce changement ce déroule comme suit :

- Lorsqu'un coordinateur passe d'une phase à l'autre, un composant non terminal est introduit. On vérifie de plus avant d'appliquer cette règle qu'il était auparavant absent.
- Les règles gérant le changement de connexion induit pour un investigateur nécessite la présence du composant non terminal
- Lorsque tous les investigateurs ont des connexions conformes (présence d'un composant non terminal et absence d'un investigateur ayant une connexion non conforme), le composant temporaire est supprimé.

En outre, on interdit toute application d'un autre règle de reconfiguration tant que le passage d'une phase à l'autre n'est pas fini (tant que le composant non terminal est présent). Lors d'un changement de phase, le coordinateur concerné cesse toute communication avec les autres coordinateurs ou les investigateurs n'appartenant pas à son groupe. De même, lors du retour en phase d'exploration, l'investigateur critique termine toute connexion autre qu'avec son coordinateur attitré.

*Initialisation E→A.* L'application de cette règle ($d_{19}$) marque l'initialisation du changement de phase d'un groupe d'investigateurs de l'exploration vers l'action.

$d_{19}$=(L={N1($X_{Sup}$,“Sup”),
N2($X_{Coord}$,“Coord”,“Exploration”,N),
N2 $\xrightarrow{\text{“P”}}$N1,N1 $\xrightarrow{\text{“I+O”}}$N2,
N3($X_{I1}$,“Inv”,N),N4($X_{I2}$,“Inv”,N),
N3 $\xrightarrow{\text{“D+P”}}$N2,N4 $\xrightarrow{\text{“D+P”}}$N2,N2 $\xrightarrow{\text{“I+O”}}$N3,
N2 $\xrightarrow{\text{“I+O”}}$N4,N4 $\xrightarrow{var}$N3,N3 $\xrightarrow{var}$N4};
K={N1($X_{Sup}$,“Sup”),
N2($X_{Coord}$,“Coord”,“Exploration”,N),
N3($X_{I1}$,“Inv”,N),N4($X_{I2}$,“Inv”,N),
N4 $\xrightarrow{var}$N3};
R\K={N5($X_{Coord}$,“Coord”,“Action”,N),N5 $\xrightarrow{\text{“P”}}$N1,
N1 $\xrightarrow{\text{“I+O”}}$N5, N4 $\xrightarrow{\text{“P”}}$N5, N5 $\xrightarrow{\text{“I+O”}}$N4,
N3 $\xrightarrow{\text{“D+P”}}$N5, N5 $\xrightarrow{\text{“I+O”}}$N3, N3 $\xrightarrow{\text{“D”}}$N4,
N6(“Temp”), N7(“TempEA”)};
NACs={NAC1};
NAC1\L={N6(“Temp”)})

Ces investigateurs forment avec un nouveau nœud, représentant le nouvel état du coordinateur et ajouté conformément à la propriété 3, un groupe minimal en phase d'action (en accord avec la propriété 9). Ces investigateurs rompent le contact avec le nœud représentant l'ancien état du coordinateur afin de respecter la propriété 5.

*Initialisation A→E.* On vérifie que le groupe comporte au moins deux investigateurs. L'application de cette règle ($d_{20}$) marque l'initialisation du changement de phase d'un groupe d'investigateurs de l'action vers l'exploration. Un nouveau nœud, représentant le nouvel état du coordinateur est ajouté conformément à la propriété 3. L'investigateur critique est lié à ce dernier en suivant les propriétés 5 et 7.

$d_{20}$=(L={N1($X_{Sup}$,“Sup”),N2($X_{Coord}$,“Coord”,“Action”,N),
N2 $\xrightarrow{\text{“P”}}$N1,N1 $\xrightarrow{\text{“I+O”}}$N2,N3($X_{I1}$,“Inv”,N),
N3 $\xrightarrow{\text{“D+P”}}$N2,N2 $\xrightarrow{\text{“I+O”}}$N3} ;
K={N1($X_{Sup}$,“Sup”),N2($X_{Coord}$,“Coord”,“Action”,N),
N3($X_{I1}$,“Inv”,N)} ;
R\K={N4($X_{Coord}$,“Coord”,“Exploration”,N),
N4 $\xrightarrow{\text{“P”}}$N1,N1 $\xrightarrow{\text{“I+O”}}$N4,N3 $\xrightarrow{\text{“D+P”}}$N4,
N4 $\xrightarrow{\text{“I+O”}}$N3,N6(“Temp”),N7(“TempAE”)} ;
NACs={NAC1} ;
NAC1\L={N6(“Temp”)})

*Traitement d'un investigateur E→A.* Cette règle ($d_{21}$) décrit le traitement d'un investigateur lors du de phase d'un groupe d'investigateurs de l'exploration vers l'action. Une fois l'initialisation effectuée, chaque investigateur sera intégré de manière valide au groupe en phase d'action. Pour ce faire, l'investigateur rompt le contact avec le nœud représentant l'ancien état du coordinateur pour établir de nouvelles connexions avec le nœud symbolisant le nouvel état du coordinateur et son investigateur critique conformément aux propriétés 5 et 9.

$d_{21}$=(L={N1($X_{Coord}$,“Coord”,“Exploration”,N),
N2($X_{Coord}$,“Coord”,“Action”,N),
N3($X_{I1}$,“Inv”,N),N4($X_{I2}$,“Inv”,N),
N3 $\xrightarrow{\text{“D+P”}}$N2,N2 $\xrightarrow{\text{“I+O”}}$N3,N5(“TempEA”),
N4 $\xrightarrow{\text{“D+P”}}$N1,N1 $\xrightarrow{\text{“I+O”}}$N4,
N4 $\xrightarrow{var}$N3,N3 $\xrightarrow{var}$N4} ;
K={N1($X_{Coord}$,“Coord”,“Exploration”,N),
N2($X_{Coord}$,“Coord”,“Action”,N),
N3($X_{I1}$,“Inv”,N),N4($X_{I2}$,“Inv”,N),N4 $\xrightarrow{var}$N3,
N3 $\xrightarrow{\text{“D+P”}}$ $N2$, N2 $\xrightarrow{\text{“I+O”}}$ $N3$,N5(“TempEA”) } ;
R\K={N4 $\xrightarrow{\text{“P”}}$ $N2$,N2 $\xrightarrow{\text{“I+O”}}$N4,N3 $\xrightarrow{\text{“D”}}$N4} ;
NACs=∅)

*Traitement d'un investigateur A→E.* Cette règle ($d_{22}$) décrit le traitement d'un investigateur lors du de phase d'un groupe d'investigateurs de l'action vers l'exploration. Une fois l'initialisation effectuée, chaque investigateur sera intégré de manière valide au groupe en phase d'exploration. Pour ce faire, l'investigateur rompt le contact avec le nœud représentant l'ancien état du coordinateur pour établir de nouvelles connexions avec le nœud symbolisant le nouvel état du coordinateur et son investigateur critique conformément aux propriétés 5 et 7.

$d_{22}$=(L={N1($X_{Coord}$,“Coord”,“Action”,N),
N2($X_{Coord}$,“Coord”,“Exploration”,N),
N3($X_{I1}$,“Inv”,N),N4($X_{I2}$,“Inv”,N),
N5(“TempAE”),N3 $\xrightarrow{\text{“D+P”}}$N2,N2 $\xrightarrow{\text{“I+O”}}$N3,
N4 $\xrightarrow{\text{“P”}}$N1,N1 $\xrightarrow{\text{“I+O”}}$N4,
N3 $\xrightarrow{\text{“D”}}$N4,N4 $\xrightarrow{var}$N3 } ;
K={N1($X_{Coord}$,“Coord”,“Action”,N),
N2($X_{Coord}$,“Coord”,“Exploration”,N),
N3($X_{I1}$,“Inv”,N),N4($X_{I2}$,“Inv”,N),
N5(“TempAE”),N4 $\xrightarrow{var}$N3,
N3 $\xrightarrow{\text{“D+P”}}$N2,N2 $\xrightarrow{\text{“I+O”}}$N3} ;
R\K={N4 $\xrightarrow{\text{“D+P”}}$N2,N2 $\xrightarrow{\text{“I+O”}}$N4} ;
NACs=∅)

*Terminaison E→A.* Lorsque tous les investigateurs ont été traités, le changement de phase est terminé et la règle $d_{23}$ est utilisée. Les nœuds temporaires signalant qu'une telle modification est en cours sont supprimés ainsi que le nœud représentant l'ancien état du coordinateur.

$d_{23}$=(L={N1($X_{Coord}$, “Coord”, “Exploration”, “N”),
N2($X_{Coord}$, “Coord”, “Action”, “N”),
N3(“TempEA”), N4(“Temp”) } ;
K={N2($X_{Coord}$, “Coord”, “Action”, “N”)} ;
R\K={ } ;
NAC={NAC1} ;
NAC1\L={N5($X_{I1}$,“Inv”,N),N1 $\xrightarrow{\text{“I+O”}}$N5,
N5 $\xrightarrow{\text{“D+P”}}$N1})

*Terminaison A→E.* Lorsque tous les investigateurs ont été traités, le changement de phase est terminé et la règle $d_{23}$ est utilisée. Les nœuds temporaires signalant qu'une telle modification est en cours sont supprimés ainsi que le nœud représentant l'ancien état du coordinateur.

$d_{24}$=(L={N1($X_{Coord}$, “Coord”, “Action”, “N”),
N2($X_{Coord}$, “Coord”, “Exploration”, “N”),
N3(“TempAE”), N4(“Temp”) } ;
K={N2($X_{Coord}$, “Coord”, “Exploration”, “N”)} ;
R\K={ } ;
NAC={NAC1} ;
NAC1\L={N5($X_{I1}$,“Inv”,N),N1 $\xrightarrow{\text{“I+O”}}$ $N5$,
N5 $\xrightarrow{\text{“P”}}$N1})

## 5.3 Navigation entre les instances consistantes

Dans le cas d'une GG n'ayant qu'un nœud non terminal de type “flag” et en considérant les règles de reconfiguration basiques comme des descripteurs formels, on cherche à démontrer que si chacune des reconfigurations est effectuée telle que décrite dans la règle correspondante, il est impossible d'atteindre une configuration invalide à partir d'une configuration valide. En sus, on montrera que toute configuration valide est accessible à partir de n'importe quelle configuration valide. Pour ce faire, nous construisons un graphe dont chaque nœud représente une configuration et chaque arc une règle de transformation de graphe.

*Notations.* Soient :

- $GG_{generation}$ = (AX,N("Temp"),T,$P_{generation}$) une GG où N("Temp") est utilisé comme flag.
- $GG_{reconf}$ = ($\emptyset$,T,$P_{reconf}$) où $P_{reconf}$ est l'ensemble des règles de reconfigurations basiques liées aux règles de $P_{generation}$. L'ensemble des non terminaux est vide car une règle basique ne peut introduire de nœud non terminal dans le cadre d'une GG avec un seul élément temporaire utilisé simplement comme simple flag.
- C l'ensemble des graphes consistants pour $GG_{generation}$. Par définition, $\forall c \in C, \exists(r_1,\dots,r_n) \in P^n_{generation}, c = r_{term}(r_1...(r_n(r_{init}(AX)...)$.
- V un ensemble d'arc labellisé de C dans C tel que, $\forall(c,c') \in C^2, \forall r \in P_{generation} - \{p_{init}, p_{termn}\}, (c,c',r) \in V \Rightarrow \exists(r_1,\dots,r_n) \in P^n_{generation}, c = r_{term}(r_1\dots(r_n(p_{init}(AX)\dots) \wedge c' = r_{term}(r(r_1\dots(r_n(r_{init}(AX)\dots)$.
- G = (C, V) le graphe de génération.
- A(c) où c $\in C$ l'ensemble des configurations accessibles par reconfiguration à partir de c. Par définition, $g \in A(c) \Leftrightarrow \exists(d_1,\dots,d_n) \in P^n_{reconf}, g = d_1\dots(d_n(c)\dots)$.
- D(c) un ensemble d'arc labellisé de A(c) dans A(c) tel que, $\forall(g,g') \in (A(c))^2, \forall d \in P_{reconf}, (g,g',d) \in V_r(c) \Rightarrow g' = d(g)$.
- R(c) = (A(c), D(c)).
- $g_0 = d_{term}(d_{init}(AX))$.

*Propriété 2.1.* $\forall(g,g',r) \in V, \forall c \in C, (g,g') \in A(c) \Rightarrow \exists(\bar{d},\tilde{d}) \in P^2_{reconf}, ((g,g',\bar{d}) \in D(c) \wedge (g',g,\tilde{d}) \in D(c))$.

Soient $\bar{d}$ la règle de transformation équivalente à r et $\tilde{d}$ est sa règle réciproque. Par définition, $(g,g',r) \in V \Rightarrow \exists(r_1,\dots,r_n) \in P^n_{generation}, g = r_{term}(r_1\dots(r_n(r_{init}(AX)\dots) \wedge g' = r_{term}(r(r_1\dots(r_n(r_{init}(AX)\dots)$.

Par définition, $L_{\bar{d}}$=L$_r$\N("Temp") et, comme $NT_{reconf}=\emptyset$, $NACs_{\bar{d}}$=$NACs_r$, donc si r est applicable à $(r_1\dots(r_n(r_{init}(AX)\dots)$, $\bar{d}$ est applicable à $r_{term}(r_1\dots(r_n(r_{init}(AX)\dots) =$ g. Son application consiste à retirer Del($\bar{d}$)=Del(r) et ajouter Add($\bar{d}$)=Add(r), d'où $r(r_1\dots(r_n(r_{init}(AX)\dots)$ est le produit de l'union disjointe de $\bar{d}(g)$ et N("Temp"), ce qui implique que g' =$\bar{d}(g)$. Par définition, $(g,g',\bar{d}) \in D(c)$.

Par définition, $\tilde{d}$ est applicable à g'. Son application consiste à retirer Del($\tilde{d}$) qui était la partie ajouté lors de l'application de r et à ajouter Add($\tilde{d}$) qui était le graphe supprimé lors de l'application de r, donc g=$\tilde{d}$(g') et $(g',g,\tilde{d}) \in D(c)$.

*Corollaire 2.1.* L'ensemble des configurations valides est stable pour les règles de reconfiguration basiques. $\forall d \in P_{reconf}, \forall c \in C, ((c,c',v) \in D(c) \implies c' \in C$.

En effet, par définition, $c \in C \implies \exists(r_1,\dots,r_n) \in P^n_{generation}, g = r_{term}(r_1\dots(r_n(r_{init}(AX)\dots)$. Par disjonction de cas,

- v est l'équivalent de la règle $r_i$. c'=$p_{term}(r_i(r_1\dots(r_n(r_{init}(AX)\dots)$, d'où $c' \in C$.
- v est la réciproque de la règle $r_i$. $\exists k \in [|1,n|], r_k = r_i$. Par conséquent $c' = p_{term}(r_1\dots r_{k-1}(r_{k+1}(\dots r_n(r_{init}(AX)\dots)$ et g'$\in C$.

*Propriété 2.2.* Le graphe de reconfiguration est indépendant de la configuration consistante initiale. $\forall(c,c') \in C^2, R(c) = R(c')$

En effet, soit c $\in C$. Par définition de C, $\exists r_1...r_n \in P^n_{generation}, c = r_{term}(r_n...(r_1(r_{init}(AX)...)$.
D'après la propriété 2.1, $\exists(\bar{d_1},\dots,\bar{d_n}) \in P^n_{reconf}, c = \bar{d_n}(...(\bar{d_1}((g0)...)$ et $\exists\tilde{d_1}...\tilde{d_n} \in P^n_{reconf}, g0 = \tilde{d_n}(\dots(\tilde{d_1}((c)\dots)$.
En outre, par définition de R(c), $\forall c' \in C, c' \in R(c) \Rightarrow R(c') \subseteq R(g)$. S'il existe un chemin entre c et c' et un chemin entre c' et c" alors il existe un chemin entre c et c".
Selon ce qui précède, $(R(c) \subseteq R(g0)) \wedge (R(g0) \subseteq R(c))$, d'où $R(c) = R(g0)$.

Finalement, $\forall c \in C, R(c) = R(g0)$. Par transitivité, on obtient la propriété 2.2.
Par la suite, on notera R, D et A.

*Propriété 2.3.* L'ensemble des configurations consistantes est exactement l'ensemble des nœuds du graphe de reconfiguration. $C = A$
En effet, $\forall c \in C, c \in A(c) = A$, d'où $C \subseteq A$.
Supposons que $A \notin C$. Soit (c,c') $\in A^2, c \notin C \wedge c' \in C$. Par définition de A, $\exists(d_1,\dots,d_n) \in P^n_{reconf}, c = d_n(\dots(d_1((c')\dots)$.
Soit $\bar{c} \in A \cap C, \exists i \in [|0,n-1|], \bar{c} = d_i(...d_1(c')...) \wedge c = d_n(...d_{i+1}(\bar{c})...) \wedge (\forall j \in [|i+1,n|], d_j(...d_{i+1}(\bar{c})...) \notin C$.
D'après le corollaire 2.1, $d_{i+1}(\bar{c}) \in C$, ce qui est une contradiction, par conséquent $A \subseteq C$.

In fine, $C = A$.

*Propriété 2.4.* R est fortement connexe.

En effet, soit (c,c') $\in A^2$. D'après la propriété précédente, (c,c') $\in C^2$. D'après la propriété 2.1, $\exists(\bar{d_1},...,\bar{d_n}) \in P^n_{reconf}, c = \bar{v_n}(...(\bar{v_1}((g0)...)$ et $\exists(\tilde{d_1},...,\tilde{d_n}) \in P^n_{reconf}, g0 = \tilde{d_n}(...(\tilde{d_1}((c)...)$. De même $\exists\bar{w_1}...\bar{w_m} \in P^m_{reconf}, c' = \bar{w_m}(...(\bar{w_1}((g0)...)$ et $\exists\tilde{w_1}...\tilde{w_m} \in (P_{reconf})^m, g0 = \tilde{w_m}(...(\tilde{w_1}((c')...)$.

Finalement, on obtient $\forall(c,c') \in A^2, (\exists \bar{d_1}...\bar{d_n}, \tilde{d_1}...\tilde{d_n} \in P^{2n}_{reconf}) \wedge (\exists \bar{w_1}...\bar{w_m}, \tilde{w_1}...\tilde{w_m} \in (P_{r,a})^{2m}), c = \bar{d_n}(...(\bar{d_1}((\tilde{w_m}(...(\tilde{w_1}((c')...))...) \wedge c' = \bar{w_1}(...(\bar{w_m}((\tilde{d_n}(...(\tilde{d_1}((c)...))...)$. R est donc fortement connexe.

Ainsi, on garantit que l'ensemble des instances valides d'une architecture logicielle définie par une GG utilisant un seul nœud non terminal en tant que flag est stable pour les règles basiques construites à partir de cette GG. En sus, elles permettent de passer de n'importe quelle configuration valide à n'importe quelle autre. Il faut ensuite prouver la stabilité des règles spécifiques introduites pour l'application, en utilisant par exemple les concepts définis dans [GDD06]. Si ces règles introduisent des éléments temporaires, la propriété 2.3 devient alors "l'ensemble des nœuds terminaux de A est égal à C".

## 6. CONCLUSIONS

Dans cet article, nous avons présenté une approche de modélisation d'architectures logicielles dynamiques. Nous avons tout d'abord défini un formalisme permettant de modéliser des règles de transformations de graphe pouvant prendre en compte de multiples conditions d'application négative. En nous basant sur un exemple, nous proposons par la suite l'utilisation de grammaires de graphes afin de définir un style architectural et de caractériser ses instances valides, ainsi qu'un modèle permettant la prise en compte du caractère dynamique de ce dernier.

Nous fournissons également un approche qui, dans certains cas, permet de construire semi-automatiquement des règles de reconfiguration. Nous garantissons la stabilité de l'ensemble des configurations consistantes pour ces règles. En outre, n'importe quelle configuration valide est ainsi accessible à partir de n'importe quelle autre.

Nos travaux s'articulant autour de GG particulières, nous nous intéresserons dans le futur à l'étude du cas général. Nous nous proposons également de fournir prochainement un framework permettant la génération automatique de GG et de l'ensemble des instances consistantes du style architectural ainsi décrit. Cet ensemble pourra ainsi être exploité dans le cadre de la prise de décision de reconfiguration lors de l'exécution de l'application considérée. Ces travaux seront utilisés dans le cadre de la modélisation de l'architecture logicielle du projet "thinking global Service for persOnnal comPuter"*.

## 7. REFERENCES


[1] R. Allen and D. Garlan. A formal basis for architectural connection. *ACM Trans. Softw. Eng. Methodol.*, 6 :213–249, July 1997.

[2] I. Bouassida-Rodriguez. *Gestion dynamique des architectures logicielles pour les systèmes communicants collaboratifs*. PhD thesis, INSA, 2011.

[3] N. Chomsky. Three models for the description of language. *Information Theory, IEEE Transactions on*, 2(3) :113–124, 1956.

[4] H. Ehrig. Tutorial introduction to the algebraic approach of graph grammars. In *Proceedings of the 3rd International Workshop on Graph-Grammars and Their Application to Computer Science*, pages 3–14, London, UK, 1987. Springer-Verlag.

[5] H. Ehrig, M. Korff, and M. Lowe. Tutorial introduction to the algebraic approach of graph grammars based on double and single pushouts. In H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Graph Grammars and Their Application to Computer Science*, volume 532 of *Lecture Notes in Computer Science*, pages 24–37. Springer Berlin / Heidelberg, 1991. 10.1007/BFb0017375.

[6] K. Guennoun. *Architectures Dynamiques dans le Contexte des Applications à Base de Composants et Orientés Services*. PhD thesis, EDSYS, 2007.

[7] K. Guennoun, K. Drira, and M. Diaz. Adressing the conformity of dynamic and distributed architecture management protocols. In *Proc. 6th IEEE International Symposium and School on Advanced Distributed Systems*, Guadalajara, Mexico, 2006.

[8] J. Magee, N. Dulay, S. Eisenbach, and J. Kramer. Specifying distributed software architectures. In W. Schäfer and P. Botella, editors, *Software Engineering ESEC '95*, volume 989 of *Lecture Notes in Computer Science*, pages 137–153. Springer Berlin / Heidelberg, 1995. 10.1007/3-540-60406-5-12.

[9] D. L. Metayer. Describing software architecture styles using graph grammars. *IEEE Transactions on Software Engineering*, 24 :521–533, 1998.

[10] G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1 : Foundations*. World Scientific, 1997.