Vers un Modèle de Déploiement à base de Bigraphes

Nadira Benlahrache*
Faiza Belala*, Chafia Bouanaka*, Malika Benammar**

* LIRE, Université Mentouri de Constantine, Algérie
belalafaiza@hotmail.com
**Département d'Informatique, Université de Batna, Algérie
m_benammar@ekit.com

Résumé : Le déploiement constitue une phase importante dans le cycle de vie d'un logiciel, souvent construite de façon ad-hoc. Les préoccupations des architectes aujourd'hui sont communes, et s'articulent autour de la définition d'un processus de déploiement générique permettant d'assembler et de distribuer correctement des applications logicielles quelle que soit leur technologie d'implémentation. Cet article propose un cadre formel, à base des bigraphes, permettant la modélisation de l'opération de déploiement d'une application dans un environnement cible. D'une part, un méta modèle à base des systèmes réactifs bigraphiques (BRS) est utilisé pour définir formellement un style architectural regroupant une famille d'architectures ayant un ensemble de propriétés communes. D'autre part, la description formelle de la plate-forme d'exécution chargée d'accueillir les composants de l'application après son déploiement est réalisée aussi au moyen d'un BRS particulier. Enfin, entre ces deux éléments se situe le processus de déploiement, défini formellement par une opération de composition des deux bigraphes précédents.

1 Introduction

Ces dernières années, les nouveaux logiciels ont gagné en complexité, essentiellement parce qu'ils doivent fonctionner dans des environnements contraints ou changeants. L'ingénierie dirigée par les modèles (IDM ou « MDE ») place le modèle au centre du processus de conception et lui permet de jouer un rôle unificateur vis-à-vis des autres activités du cycle de développement du logiciel. Le souci des architectes actuels est de trouver un cadre unificateur permettant une bonne expression de tous les aspects d'une architecture logicielle. Les langages de description d'architectures (ADL) (Medvidovic et Taylor, 2000) ont été définis afin de définir plus précisément et de mieux comprendre cette architecture, constituée principalement de composants fonctionnels décrits en termes de leurs comportements, interfaces et intercommunications.

Les nombreux ADLs existants dans la littérature possèdent des caractéristiques spécifiques de modélisation, liées à la motivation de chacun d'entre eux, à leur usage et à la sémantique formelle qu'il attache à une description architecturale. Toutefois, avec la révolution informatique qui a donné des logiciels plus complexes et évoluant sans cesse,

plusieurs limites de ces langages ont été relevées, les rendrant inutilisables pour certains aspects du cycle de développement des systèmes logiciels, particulièrement la spécification des mécanismes de *déploiement* et de *reconfiguration*.
En effet, toute application développée doit être déployée pour être exécutée sur un environnement spécifique représenté par un ensemble de composants matériels (processeurs, mémoires et bus de communication) et/ou logiciels (Intergiciel ou Middleware) nécessaires à son déploiement et son lancement. Un défi concernant le déploiement des applications architecturales, est de garantir un déploiement cohérent qui assure d'une part le bon fonctionnement de l'application et d'autre part l'intégrité de l'environnement cible sur lequel cette architecture est déployée.
Les ADLs nécessitent un cadre sémantique formel plus élaboré afin de pouvoir définir les spécificités et les exigences nécessaires à toutes les étapes de déploiement d'une architecture logicielle. Ils doivent disposer d'un certain nombre d'informations relatives à l'application ainsi qu'à son environnement cible pour réussir le processus de déploiement. En particulier, ils doivent être en mesure de (Parrish et al., 2001) :

- Spécifier et vérifier des dépendances entre les composants nécessaires au déploiement,
- Garantir le déploiement correct des composants,
- Garantir une évolution sûre du système après son déploiement.

Notre propos ici est de proposer un cadre formel, à base des bigraphes, permettant la modélisation de l'opération de déploiement d'une application dans un environnement cible. Nous considérons ainsi l'architecture du logiciel au centre des différentes phases (pas uniquement la phase de conception) de son cycle de vie. Nous la retrouvons essentiellement dans la phase de son déploiement comme suit :
En premier lieu, nous décrivons la configuration d'une application comme une instance d'un style architectural donné. Ainsi, nous spécifions uniquement les contraintes les plus importantes, au niveau par exemple de la structure, du comportement, de l'utilisation des ressources, des instances de composants à créer et de la configuration de leurs attributs ainsi que les liaisons de leurs ports. Pour cela, nous adoptons un méta modèle à base des bigraphes, proposé dans (Chang et al., 2007) pour définir formellement un style architectural regroupant une famille d'architectures ayant un ensemble de propriétés communes.
En second lieu, le modèle utilisé pour décrire la plate-forme d'exécution chargée d'accueillir les composants de l'application après son déploiement est aussi un bigraphe particulier. Cette modélisation explicite des plates-formes favorise la prise en compte de leurs caractéristiques (performances, maintenance, etc.)
Enfin, entre ces deux éléments se situe le processus de déploiement, défini formellement par une opération de composition des deux bigraphes précédents.

Les bigraphes ou Systèmes réactifs bigraphiques (BRS) sont des graphes particuliers, introduits par Jensen et Milner (2003, 2004), capables de représenter les deux aspects d'une architecture, les places et les liens, via deux graphes distincts. Les places sont représentées par une arborescence et les liens par un hypergraphe. Cette séparation des aspects permet de travailler sur un aspect indépendamment du deuxième. De plus, les règles de réaction définies dans les bigraphes peuvent être utilisées pour spécifier l'évolution des systèmes.

L'objectif de cet article est double, d'une part, nous offrons une base mathématique rigoureuse à l'architecte chargé de concevoir et de décrire l'architecture logicielle. D'autre part, cette même base mathématique permettra de guider le développeur de l'application dans

l'implémentation du système. A notre connaissance, peu de travaux se sont, à ce jour, intéressés à la modélisation des plates-formes d'exécution. Notre contribution à travers cet article consiste donc à proposer un modèle mathématique approprié pour une formalisation explicite d'une plate-forme d'exécution, favorisant la prise en compte de ses caractéristiques et facilitant son intégration dans le processus de déploiement. La suite de cet article s'organisera comme suit :
La première section dévoilera brièvement les systèmes réactifs bigraphiques et leurs caractéristiques principales. Nous introduirons dans la section suivante l'utilisation des bigraphes dans la littérature pour représenter les architectures logicielles. Notre contribution sera détaillée dans la section 4 suivie par une conclusion.

2 Systèmes Réactifs Bigraphiques

La théorie des systèmes réactifs bigraphiques (BRSs), définie dans (Jensen et Milner, 2004), est basée sur un modèle graphique pour la spécification d'applications distribuées à code mobile. Ce modèle offre les outils nécessaires pour la prise en charge des deux dimensions, interaction et distribution spatiale, de l'application à spécifier en fusionnant deux types de graphes d'où le nom de bigraphes. Le graphe des places spécifie l'hiérarchie des entités physiques ou logicielles, constituant l'application, et capables d'accueillir d'autres entités. Le graphe des liens montre les connectivités possibles entre les différentes entités. Les bigraphes représentent aussi un cadre unificateur pour plusieurs modèles concurrents et mobiles basés sur le calcul des processus, tels que CCS, le π-calcul, les systèmes ubiquitaires, ou les réseaux de Petri.

2.1 Définition Formelle d'un Bigraphe

Sur la base d'un ensemble commun de noeuds, correspondant aux entités physiques ou virtuelles d'une application distribuées, un bigraphe est construit à partir de la fusion de deux structures indépendantes: le graphe des places, ayant la structure d'une forêt, montre la distribution spatiale de l'application. Le graphe des liens est un hypergraphe établissant le schéma de connectivité des différents noeuds. Alors qu'un arc dans le graphe des places montre la relation d'imbrication entre les éléments de l'application, un arc dans le graphe des liens établit une connexion entre les ports des ces éléments. Chaque arbre dans le graphe des places représente une région qui peut contenir des sites, correspondant aux feuilles de l'arbre, où d'autres bigraphes peuvent être insérés ou hébergés.

Définition.1 (Milner, 2004) : Un bigraphe est défini par $G = \left(V, E, ctrl, G^P, G^L\right) : I \to J$, tels que :

- V est un ensemble fini de nœuds qui peuvent être imbriqués.
- E est un ensemble fini d'hyper-arcs.
- $ctrl : V \to \mathbb{K}$ est une transformation qui associe à chaque nœud v_i de V un contrôleur $k \in K$ indiquant le nombre de ports et son comportement dynamique.
- $G^P = (V, E, ctrl, prnt) : m \to n$ est le graphe des places associé à G, où $prnt : m \cup V \to V \cup n$ est une fonction de parenté associant à chaque nœud son parent

hiérarchique. m et n représentent respectivement le nombre de sites et de régions dans le graphe des places.

– $G^L = (V, E, ctrl, link) : X \to Y$ est le graphe des liens de G, où $link : X \cup P \to E \cup Y$ est une transformation montrant le flux de données des noms internes X ou les ports P vers les noms externes Y ou les arcs E.

– $I = \langle m, X \rangle$ et $J = \langle n, Y \rangle$ sont respectivement les interfaces internes et externes du graphe G, avec :

- m est le nombre de sites i.e., emplacements dans le graphe susceptibles d'abriter de nouveaux éléments,
- X est l'ensemble des noms internes,
- n est le nombre de régions i.e., éléments du bigraphe pouvant s'intégrer dans d'autres bigraphes,
- Y est l'ensemble des noms externes.

Une interface $I = \langle 0, \varnothing \rangle$ peut être notée par $I = \varepsilon$.

Exemple.1 : Soit le bigraphe $G = (V, E, ctrl, G^P, G^L) : <0, x> \to <3, \varnothing>$ présenté dans la figure 1 avec un ensemble de nœuds $V = \{v_0, v_1, v_2, v_3, v_4, v_5\}$, un ensemble d'hyper-arcs ou liens $E = \{e_0, e_1, e_2\}$, une transformation $Ctrl = \{(v0:1), (v1:1), (v2:1), (v3:2), (v4:1), (v5:2)\}$ et une fonction de parenté $prnt = \{(v0:v2), (v1:v2), (v2:\varnothing), (v3:\varnothing), (v4:v5), (v5:\varnothing)\}$; Le nombre de sites ici est nul ($m = 0$), c.-à.-d. pas d'emplacement libre dans ce bigraphe. Le bigraphe contient trois régions (n=3) numérotées de 0..2 et representées par rectangles en pointillés.

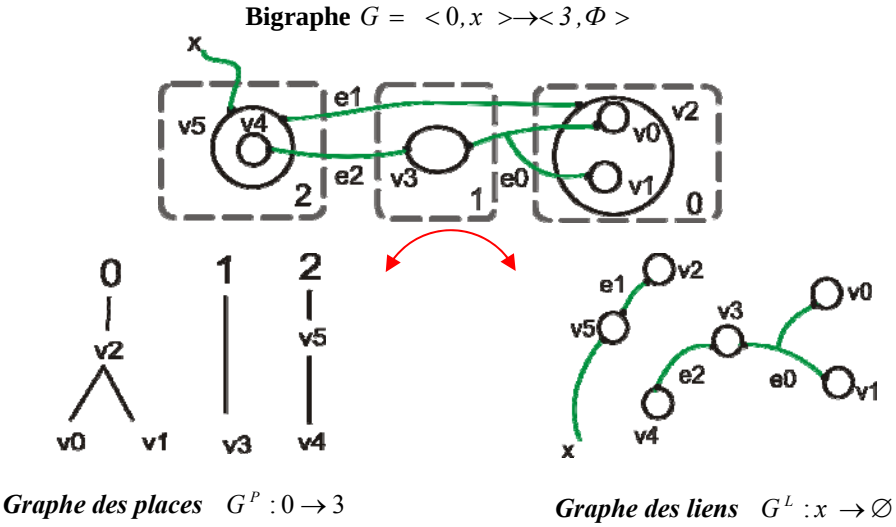

FIG. 1 – *Un exemple de bigraphe et ses deux structures.*

2.2 Transformation des Bigraphes

La dynamique d'un bigraphe est spécifiée par des règles de transformation, appelées règles de réaction. Deux types de transformation sont possibles sur un bigraphe. La transformation sur les places exprime l'opération de mobilité dans le système considéré. Elle représente l'arrivée ou le départ d'une entité. De son côté, la transformation sur les liens exprime la connexion ou déconnexion d'un nœud du bigraphe à travers l'une de ses interfaces internes.

Définition.2 Une règle de réaction a la forme $(R : m \to J, R' : m' \to J, \eta)$ et stipule que le bigraphe *R*, appelé bigraphe *Redex*, peut être transformé en *R'*, appelé bigraphe *Reactum,* $\eta : m' \to m$ est une application sur le nombre de sites. Les bigraphes R et R' sont sans noms internes (X=∅, X'=∅). Et les noms externes sont identiques ($Y = Y'$).

Exemple.2 : La règle de réaction de la figure 2 exprime le fait qu'un agent, se trouvant dans la même région qu'une chambre, puisse y entrer. Avec R le bigraphe Redex et R' le bigraphe reactum : $(R : 1 \to \langle 1, \{x, y\}\rangle$ *et* $R' : 1 \to \langle 1, \{x, y\}\rangle, \ \eta = \{0 \to 0\})$.

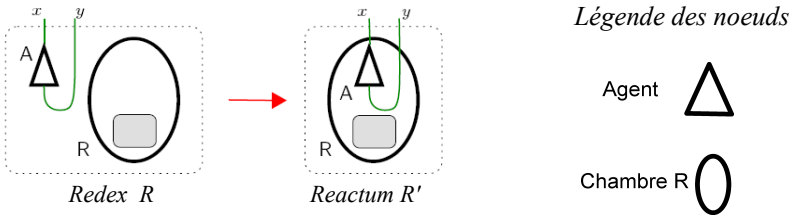

FIG. 2- *Transformation sur les places (Chang et al. 2007).*

Définition.3 Etant donnée une signature *K*, un système bigraphique (BRS) noté généralement *BIG(K,* R), est défini par la catégorie BIG(K) de bigraphes, équipée d'un ensemble de règles de réactions *R.*

Informellement, la sémantique des systèmes bigraphiques est définie à base des catégories mathématiques. Chaque BRS est une catégorie ayant comme objets l'ensemble de ses bigraphes et comme morphismes l'ensemble de règles de transformations des bigraphes.

2.3 Composition de Bigraphes

Comme dans tout type de graphes, la composition de bigraphes est une opération permettant d'obtenir un nouveau bigraphe en combinant deux ou plusieurs autres bigraphes. La composition $G = H \circ F$ de deux bigraphes *F* et *H* est une opération d'hébergement du bigraphe *F* dans le bigraphe hôte ou contextuel *H*. Pour que cette insertion soit correcte, il est nécessaire que chaque région de *F* trouve un site libre dans *H*, i.e., il y'a suffisamment de sites libres dans *H* pour pouvoir héberger *F*. La connexion des deux bigraphes se ferra à travers un appariement des interfaces externes de F avec les interfaces internes de *H*.

Exemple.3 : Soit un bigraphe *F* constitué de 2 régions (0, 1), ayant *x* et *y* comme noms externes. Et un graphe contextuel *H* avec 2 sites, et *x* et *y* comme noms internes. La composition des bigraphes *F* et *H* est un bigraphe *G* (voir figure 3) sans sites ni noms externes.

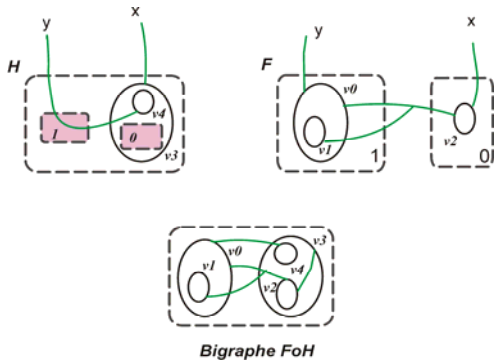

FIG. 3 – *Composition de deux bigraphes*.

3 Architecture Logicielle et BRS

La spécification des styles architecturaux, et par conséquent les architectures, nécessite un formalisme de description permettant, en plus de la description de l'architecture, de s'assurer de sa cohérence pour analyser et vérifier ses propriétés.
Le graphe offre un modèle formel et visuel pour définir une architecture logicielle. Ses nœuds représentent les composants de l'architecture et les arcs définissent alors les connecteurs entre les différents composants. En plus, les opérations de reconfiguration structurelle d'une architecture peuvent être vues comme des opérations de transformations de graphes. D'autre part, l'introduction de la notion de typage dans les graphes permet non seulement de travailler sur une architecture logicielle donnée, mais sur tout un style architectural, et ce en définissant le vocabulaire des éléments conceptuels d'une architecture et l'ensemble de règles indiquant comment ils peuvent être connectés.

En effet, plusieurs approches dans la littérature adoptent le concept des graphes et leurs transformations pour modéliser les styles architecturaux et leurs instances. Dans (Métayer, 1998), les auteurs proposent de lier les styles architecturaux aux grammaires de graphes pour pouvoir considérer par la suite les reconfigurations d'architectures comme étant des transformations de graphes. Cette même idée a été exploitée dans (Bruni et al., 2007) et étendue par une approche hiérarchique plus générique (ADR) facilitant la représentation des règles de reconfiguration plus complexes ayant comme paramètres des architectures typées. Autrement dit, les styles architecturaux sont spécifiés via des algèbres ayant des modèles d'interprétation à base des graphes. Les règles de construction d'une architecture (ou productions) sont vues comme étant des opérations de composition d'architectures élémentaires, selon leurs types, donnant des résultats bien définis. Ainsi, les reconfigurations d'architectures avec préservation de styles peuvent être exprimées par des règles de la réécriture de termes algébriques formés autour des informations sur ces architectures.

L'originalité de ces travaux réside dans le fait qu'ils utilisent un seul modèle unificateur, réécriture des graphes, pour représenter les architectures logicielles, leur comportement ainsi que leurs reconfigurations. D'autres travaux se référant aux mêmes objectifs sont ceux de (Chang et al., 2007) qui partent du constat de la nécessité de définir des types de composants, de connecteurs et de ports/rôles pour former un style, sans négliger la spécification des sous structures récurrentes exigées, ainsi que leurs propriétés et contraintes. Ils proposent le modèle BRS pour modéliser un style architectural d'un système en tenant compte de toutes ses spécificités. Les types associés aux interfaces expriment le typage des ports/rôles et les attributs des contrôleurs dénotent les types des composants et des connecteurs. Dans leur contexte de travail, une architecture est définie donc par un bigraphe et la famille des bigraphes déterminés par un BRS forme son style. En outre, ils utilisent les règles de réaction du BRS pour restreindre les reconfigurations de ces architectures et s'assurer qu'elles préservent leur style. La différence et l'intérêt de cette approche de modélisation d'un style d'architectures par rapport aux autres approches existantes, est la capacité du formalisme sous-jacent (BRS) à représenter naturellement des systèmes complexes réels, en particuliers les systèmes mobiles et « context-aware ». En effet, leur modèle sémantique est basé sur la théorie des catégories. Cette base mathématique lui prodigue un moyen visuel assez simple permettant de fournir des informations sur les deux aspects statiques et dynamiques d'un système.

Les travaux de (Chang et al., 2007) constituent notre source d'inspiration pour proposer un processus générique de déploiement d'architecture en exploitant le concept de composition des bigraphes. Nous reprenons alors les détails des points que nous jugeons importants pour notre approche de modélisation:

1. Les principaux éléments d'une instance d'architecture sont les composants, les connecteurs, les ports, les rôles et les configurations. Une correspondance est alors définie entre tous ces éléments et ceux d'un bigraphe. Chaque nœud dans un bigraphe représente un composant ou un connecteur. La structure arborescente du graphe des places permet de conserver la structure de la hiérarchie des composants ou connecteurs composites. De plus, la notion de contrôleur d'un nœud (composant ou connecteur) sert à définir les interfaces d'un composant (ports) ou celles d'un connecteur (rôles) ainsi que les comportements associés. Chaque arc décrit une connexion entre les ports et les rôles.

2. Les architectures peuvent être regroupées dans un style architectural qui spécifie uniquement les contraintes les plus importantes, au niveau par exemple de la structure, du comportement, de l'utilisation des ressources, des composants et des connecteurs dans un système. Du moment qu'un bigraphe peut représenter une architecture, alors tous les bigraphes déduits d'un BRS forment un style. Des conditions supplémentaires peuvent enrichir ce modèle pour décrire les contraintes d'un style donné. Il peut ainsi aider à la définition de la sémantique d'un système et son analyse.

3. De plus, les instances d'architectures d'un style ont souvent besoin d'être reconfigurées durant leurs exécutions. Ces changements possibles conformément à un style sont définis par les règles de transformation (réaction) d'un BRS.

4 Un Modèle de déploiement à base des Bigraphes

Dans cet article, nous proposons un cadre formel, à base des bigraphes, permettant la modélisation de l'opération de déploiement d'une application dans un environnement cible. Nous considérons ainsi l'architecture du logiciel au centre des différentes phases de son cycle de vie. R. Milner et ses collaborateurs se sont appuyés sur la notion de mobilité pour motiver cette séparation entre les places et les liens. Nous exploitons cet aspect pour décrire les aspects structurels et comportementaux d'une opération d'installation d'une application dans un environnement cible. La notion de place peut être traduite par la disponibilité des ressources requises par l'application, et la notion de liens peut se traduire par les dépendances qu'il faut satisfaire pour réussir le déploiement.

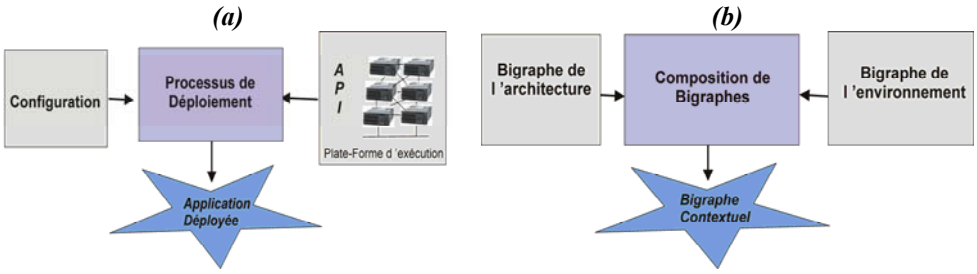

FIG. 4 – *Processus de déploiement et bigraphes.*

Notre approche de modélisation (Figure 4.b) est composée essentiellement de trois phases. Dans un premier temps, nous décrivons la configuration d'une application comme une instance d'un style architectural donné. Ceci nous permet de spécifier uniquement les contraintes les plus importantes de l'architecture, au niveau par exemple de la structure, du comportement, de l'utilisation des ressources, des instances de composants à créer et la configuration de leurs attributs ainsi que les liaisons de leurs ports.
Pour cela, nous adoptons un méta modèle à base des bigraphes, proposé dans (Chang et al., 2007) pour définir formellement un style architectural regroupant une famille d'architectures ayant un ensemble de propriétés communes. Ce modèle est spécifié formellement par la définition suivante :

Définition.4 : Un style architectural S est représenté par un BRS, une catégorie de bigraphes $\boldsymbol{Cat_{AL}}$ composée d'un ensemble de bigraphes et d'un ensemble de règles de réactions $\boldsymbol{R_{AL}}$ sur ces bigraphes. Un bigraphe $\boldsymbol{F}$ de $\boldsymbol{Cat_{AL}}$, représentant une instance d'architecture de ce style est défini par le tuple $\boldsymbol{F} = \left(\boldsymbol{V_F}, \boldsymbol{E_F}, \boldsymbol{Ctrl_F}, \boldsymbol{G_F^P}, \boldsymbol{G_F^L}\right) : \boldsymbol{I_F} \rightarrow \boldsymbol{J_F}$ où :

- $\boldsymbol{V_F}$: est l'ensemble des composants et connecteurs de l'architecture,
- $\boldsymbol{E_F}$: est l'ensemble des hyper-arcs représentants les différentes interactions entre ports et rôles des éléments de $\boldsymbol{V_F}$,
- $\boldsymbol{Ctrl_F}$: est le contrôleur associé à l'ensemble $\boldsymbol{V_F}$,
- $\boldsymbol{G_F^P}$ et $\boldsymbol{G_F^L}$ représentent respectivement le graphe de *Places* et le graphe de *Liens* du bigraphe $\boldsymbol{F}$.

- $I_F = \langle m_F, X_F \rangle = \langle O, \varnothing \rangle$, l'instance de cette architecture est sans sites ni noms internes. Cela signifie que l'application architecturale correspondante est complète et n'a pas besoin d'être enrichie par d'autres composants.
- $J_F = \langle n_F, Y_F \rangle$, n_F désigne la portée de la distribution spatiale de l'architecture logicielle considérée, et Y_F désigne l'ensemble des interfaces externes d'interaction avec l'environnement de déploiement. Ces interfaces expriment également les différentes dépendances que l'environnement doit satisfaire pour réussir un déploiement.
- R_{AL} : constitue l'ensemble des règles de réactions possibles sur ce bigraphe et qui représentent en termes d'architecture logicielle, un ensemble d'opérations de reconfiguration possible de cette instance d'architecture.

En second lieu, un autre BRS noté Cat_{ENV} est utilisé pour décrire la plate-forme d'exécution chargée d'accueillir les composants logiciels de l'application lors de son déploiement. R_{ENV} représente l'ensemble des règles de réaction sur les bigraphes de Cat_{ENV}. Cette modélisation explicite des plates-formes favorise la prise en compte de leurs caractéristiques et contraintes. Soit H un bigraphe de Cat_{ENV} représentant l'environnement cible, $H = (V_H, E_H, Ctrl_H, G_H^p, G_H^L) : I_H \rightarrow J_H$ tels que :

- $V_H, E_H, Ctrl_H, G_H^p, G_H^L$: les principaux éléments du bigraphe
- $I_H = \langle m_H, X_H \rangle$, avec $m_H \neq 0$ et $X_H \neq \varnothing$; Cette double condition exprime que l'environnement doit être capable d'offrir des sites (m_H) pour héberger des applications logicielles et des interfaces internes (X_H) servant à l'interaction. Les interfaces internes peuvent alors désigner dans ce cas les dépendances de déploiement que peut satisfaire l'environnement vis-à-vis une application logicielle donnée.
- $J_H = \langle n_H, Y_H \rangle$, avec éventuellement $n_H \geq 0$ et $Y_H \neq \varnothing$ en cas d'un environnement distribué dans l'espace.
- R_{ENV} : définit l'ensemble des opérations de reconfiguration possible de l'environnement.

L'opération de construction de graphe de places va permettre d'identifier les « sites » que peut contenir le bigraphe de l'environnement de déploiement H. Ces sites représentent des emplacements libres dans l'environnement et susceptibles d'accueillir des régions en provenance de l'application à déployer, on parle alors d'hébergement. Ces sites peuvent être : un espace mémoire disponible, un processeur libre, une bibliothèque prête à être exploitée etc. la même opération conduit à l'identification des régions dans F qui sont des zones relogeables de l'application. L'opération de construction de graphe de liens va permettre d'identifier les interfaces internes que peut contenir le bigraphe de l'environnement de déploiement H. Ces interfaces représentent des dépendances que l'environnement doit satisfaire et qui doivent correspondre aux interfaces externes de l'application.

Enfin, la dernière phase est consacrée à la description formelle du processus de déploiement, en utilisant l'opération de composition des bigraphes appliquée dans notre contexte aux bigraphes représentant respectivement l'architecture logicielle d'une application et l'environnement matériel qui va la supporter (FIG.5(b)). La fusion des deux bigraphes demandeur et fournisseur, est modélisée par une opération de composition. Afin d'assurer la réussite de l'opération de composition; en d'autres termes le déploiement de

l'application dans son environnement d'exécution; la vérification de certaines conditions est nécessaire: La correspondance entre l'interface d'entrée de l'environnement $\boldsymbol{I_H = \langle m_H, X_H \rangle}$ et l'interface externe de l'application $\boldsymbol{J_F = \langle n_F, Y_F \rangle}$, c-à-d., la correspondance des noms externes de l'application avec les noms internes de l'environnement et le nombre de régions de l'application avec le nombre de sites de l'environnement. Il faut pouvoir s'assurer qu'il y'a suffisamment de sites libres dans $\boldsymbol{H}$ pour pouvoir héberger $\boldsymbol{F}$. Formellement, la formalisation de cette opération est réalisée comme suit :

- Soit un bigraphe $\boldsymbol{H}$ modélisant la plate-forme de déploiement *E*,
- Soit un bigraphe $\boldsymbol{F}$ modélisant l'architecture à déployer,
- L'opération d'installation de l'instance F dans l'environnement E est spécifiée par un bigraphe $\boldsymbol{G = F \circ H}$ résultat de la composition des bigraphes $\boldsymbol{F}$ et $\boldsymbol{H}$. Formellement, cette installation est définie comme suit :

Définition5 : Etant donné les deux bigraphes $\boldsymbol{F}$ et $\boldsymbol{H}$ associés respectivement à une instance d'architecture et un environnement de déploiement, le bigraphe $\boldsymbol{G}$ obtenu par composition des bigraphes est défini par $\boldsymbol{G = (V_G, E_G, Ctrl_G, G_G^P, G_G^L) : I_G \rightarrow J_G}$ tels que :

- $\boldsymbol{V_G = V_F \cup V_H}$,
- $\boldsymbol{E_G = E_F \cup E_H}$,
- $\boldsymbol{Ctrl_G = Ctrl_F \cup Ctrl_H}$: ensemble des contrôleurs associés aux noeuds de G.
- $\boldsymbol{I_G : \langle m_G, X_G \rangle}$ *et* $\boldsymbol{J_G : \langle n_G, Y_G \rangle}$, constituent ses interfaces avec : $\boldsymbol{m_G = m_H - n_F}$, $\boldsymbol{X_G = X_H - Y_F}$, $\boldsymbol{n_G = n_H}$, et $\boldsymbol{Y_G = Y_H}$, sachant que $\boldsymbol{m_G}$ représente le nombre des sites encore disponibles de G et qui sont ceux de H moins ceux occupés par les régions de F, les régions de G sont ceux de H.

Exemple.4 : Considérons le bigraphe $\boldsymbol{F = (V_F, E_F, Ctrl_F, G_G^P, G_G^P) : \varepsilon \rightarrow \langle 3, \{x, y\} \rangle}$, représentant l'application architecturale F à déployer, (la figure 5), les nœuds $\boldsymbol{V_F}$ = {v1,v3, v4,v5} représentant les composants et les connecteurs, les hyper-arcs E_F ={e1, e2} exprimant les relations de dépendances entre eux . $Ctrl_F = \{(v_1{:}2), (v_3{:}2), (v_4{:}2), (v_5{:}1)\}$, et $\varepsilon = \langle 0, \varnothing \rangle$ indique que $\boldsymbol{F}$ ne possède pas de sites ni de noms internes par contre $\langle 3, \{x, y\} \rangle$ représente ses interfaces externes {x, y} qui sont ses exigences (à satisfaire par l'environnement de déploiement). En plus, le bigraphe $\boldsymbol{F}$ contient trois régions devant trouver leurs sites correspondants au niveau du bigraphe $\boldsymbol{H}$ spécifiant l'environnement afin que l'installation soit faite correctement.

De l'autre côté *H* est défini par $H = (V_H, E_H, Ctrl_H, G_H^P, G_H^L) : \langle 3, \{x, y\} \rangle \rightarrow \langle 2, \varnothing \rangle$ où V_H = {v0, v2} et E_H = {e0} , et $Ctrl_H = \{(v_0 : 1), (v_2 : 0)\}$. En ce qui concerne $\langle 3, \{x, y\} \rangle$ cela indique que *H* possède trois sites et deux interfaces internes {x,y}. C'est à travers ces interfaces que *H* interagira avec *F*. Les trois sites serviront à héberger les trois régions de *F*. Le bigraphe *H* possède comme interfaces externes $\varepsilon = \langle 2, \varnothing \rangle$, spécifiant que l'environnement est lui-même, constitué de deux régions mais n'a pas de d'interfaces externes. Les préconditions de l'opération de composition dans ce cas s'expriment ainsi :

- les régions de *F* doivent trouver leurs sites correspondants, dans l'ordre, i.e., la région 0 de F sera hébergée par le site 0 de *H* et ainsi de suite.
- Les noms externes de *F* doivent eux aussi, trouvés leurs correspondants au niveau de *H*. L'interface externe *x* de *F* sera connectée à l'interface interne *x* de *H.*

La capacité du formalisme utilisé (BRS) et sa sémantique catégorielle sous jacente contribuent à la modélisation du processus de déploiement d'une architecture *F* ayant un style donné, sur une plateforme *H* appartenant à une classe d'environnements cibles présentant des caractéristiques communes.

Afin de rendre les BRS capables de décrire un style architectural, (Chang et al. 2007) ont introduit la notion de type pour les composants, connecteurs, ports/rôles et les noms ainsi qu'une liste de propriétés et de contraintes. Ces propriétés que l'on peut définir sur les composants et connecteurs (en format XML) permettront dans notre contexte de spécifier des meta-informations associées à chaque composant logiciel à déployer, tels que l'espace mémoire requis, la version ainsi que le fichier script associé. De l'autre côté, et pour la plateforme d'exécution, ces mêmes informations définiront les caractéristiques de la machine cible en termes de disponibilité en espace mémoire, caractéristiques du processeur, bus de communication et bibliothèques, etc.

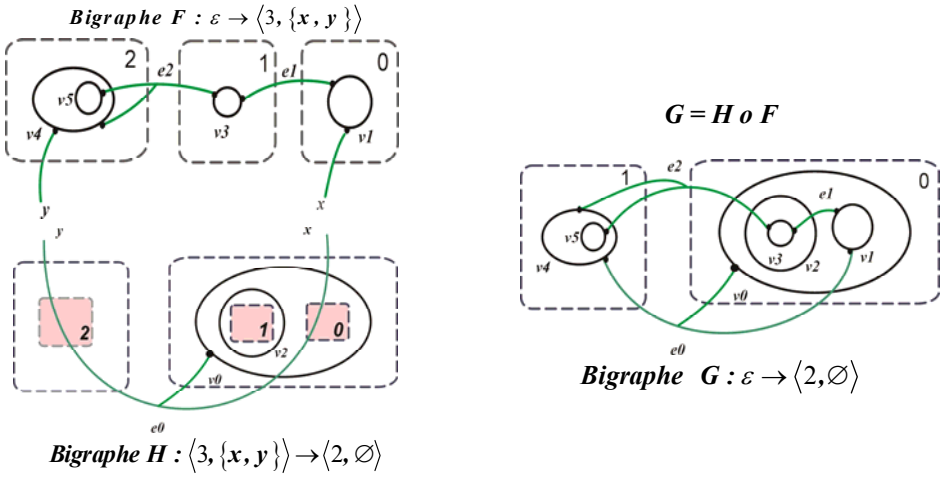

FIG. 5 –Déploiement via les bigraphes.

Le processus de déploiement modélisé par une composition de bigraphes, peut être évalué au préalable par la vérification et la validation des différentes dépendances et contraintes exprimées au niveau du bigraphe de l'architecture et celui de la plateforme d'exécution. Dans le contexte de généralisation de l'utilisation des BRS, (Grohmann et Miculan, 2007) ont introduit une polarisation à une interface (un signe (+,-) qui indiquera le sens du flux de communication). Ce signe permet ainsi de renforcer le contrôle sur la façon de connecter un port. Un port positif ne peut être connecté qu'à un autre port négatif.

De plus, la reconfiguration d'une architecture dans ce cas, n'est autre qu'une transformation de bigraphe. L'application d'une règle de réaction peut introduire une transformation dans l'architecture du point de vue structurel (transformation de places) ou comportemental (transformation de liens). Un bigraphe transformé est une application

reconfigurée. Un bigraphe représentant une instance architecturale d'un style donnée, reste toujours conforme au style de départ quelque soit les transformations obtenues par l'application des règles de réaction prédéfinies (Chang et al. 2007). De cette manière le concept de bigraphe ne nous permet pas uniquement d'évaluer en amont, le processus de déploiement caractérisé par l'action de l'installation mais également d'anticiper les effets de la reconfiguration. Et puisque les règles de réactions sont définies région par région (Milner, 2008). Le bigraphe obtenu par composition, peut être transformé par l'application des mêmes règles définies pour les bigraphes initiaux en respectant un certain ordre d'application. La reconfiguration concernera probablement l'application et l'environnement à la fois.

5 Conclusion

Toute application développée doit être déployée pour être exécutée sur un environnement spécifique. Cet environnement est souvent représenté par un ensemble de composants matériels (processeurs, mémoires et bus de communication) et logiciels (Intergiciel) nécessaires pour le déploiement et le lancement de cette application logicielle. Ce déploiement doit être cohérent afin d'assurer d'une part, le bon fonctionnement de l'application et d'autre part, l'intégrité d'un tel environnement. Par ailleurs, les graphes offrent un modèle formel et visuel pour définir une architecture logicielle où les opérations de reconfiguration structurelle peuvent être vues comme des opérations de transformations de graphes.

Dans cet article nous avons proposé un modèle mathématique, à base de bigraphes, permettant la modélisation de l'opération de déploiement d'une application dans un environnement cible. Nous avons d'abord adopté un méta modèle à base des BRS (systèmes réactifs bigraphiques) pour définir formellement un style architectural regroupant une famille d'architectures ayant un ensemble de propriétés communes. Nous avons alors défini le bigraphe $\boldsymbol{F}$ et l'ensemble de règle de réactions $\boldsymbol{R_F}$, pour représenter une instance architecturale de ce style. Nous avons ensuite utilisé un BRS particulier, pour décrire formellement la plate-forme d'exécution chargée d'accueillir les composants de l'application après son déploiement. Le bigraphe $\boldsymbol{H}$ et l'ensemble de règles de réactions $\boldsymbol{R_H}$ définies sur ce bigraphe formalisent la plate-forme d'exécution et favorisent ainsi la prise en compte de ses propres caractéristiques pour faciliter son intégration dans le processus de déploiement. Enfin, nous avons défini formellement le processus de déploiement par une opération de composition $\boldsymbol{G = H \circ F}$ des deux bigraphes $\boldsymbol{F}$ et $\boldsymbol{H}$.

Cette approche de modélisation proposée nécessite des implémentations et des études de cas concrètes afin de la tester et la vérifier. Nous pensons que la mise en pratique du modèle mathématique bigraphe sera facilitée énormément par l'utilisation de « BPL », un langage dédié aux bigraphes (bigraphical programming language) (Birkedal et al., 2006).

Références

Bruni, R. A. Lluch Lafuente, U. Montanari, et E. Tuosto (2007). "Style-Based Architectural Reconfigurations ", Bulletin of the European Association for Theoretical Computer Science, EATCS. 94:161-180.

Birkedal, L., M., Bundgaard, T. C., Damgaard, and ect. (2006). “Bigraphical programming languages for pervasive computing”, In Proc. of Pervasive International Workshop on Combining Theory and Systems Building in Pervasive Computing, Dublin, Ireland, , 653-658.

Chang, Z., X. Mao, et Z. Qi (2007). ” An Approach based on Bigraphical Reactive Systems to Check Architectural Instance Conforming to its Style “, First Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering(TASE'07). 57-66.

Grohmann D. et M. Miculan (2007) . “Directed Bigraphs”, Electronic Notes in Theoretical Computer Science (ENTCS), 173:121-137, ISSN: 1571-0661

Jensen O.H. et R. Milner (2003). “Bigraphs and mobiles processes “. Technical Report 570, University of Cambridge. ISSN: 1476-2986

Jensen O.H. et R. Milner (2004). “Bigraphs and mobiles processes (revised)“. Technical Report 580, University of Cambridge. ISSN: 1476-2986

Medvidovic, N. et R. N. Taylor (2000). “A Classification and Comparison Framework for Software Architecture Description Languages”, IEEE Transactions on Software Engineering, 26(1):70-93.

Métayer, D. L. (1998). “Describing software architecture styles using graph grammars. IEEE Trans. Software Eng., 24(7):521–533,

Milner, R. (2008). "Bigraphs: a space for interaction" *disponible sur le site* : http://www.cl.cam.ac.uk.

Milner, R. (2004). ” Bigraphs whose names have multiple locality”, Technical Report, no 603, Cambridge University.

Parrish, A., B. Dixon, et D. Cordes (2001). “A Conceptual Foundation for Component-Based Software Deployment,” Journal of Systems and Software, 57(3): 193-200.

Summary

The deployment represents an important phase in the lifecycle of software, often built in an ad-hoc way. The concerns of architects are now almost common, and revolve around the definition of a generic deployment process to assemble and properly distribute software applications regardless of their implementation technology. This paper proposes a formal framework, based on bigraphs, allowing the modeling of the deployment operation of application in a target environment. On the one hand, a Meta model-based bigraphical reactive systems (BRS) is used to formally define an architectural style involving a family of architectures with a set of common properties. On the other hand, the formal description of the platform carrying charge to host components of the application after deployment is also performed using a particular BRS. Finally, between these two elements is the deployment process, formally defined by an operation of composition of the two previous bigraphs.