Multi-layer coordinated adaptation based on graph refinement for cooperative activities

Ismael Bouassida Rodriguez, Nicolas van Wambeke, Khalil Drira, Christophe Chassot, Mohamed Jmaiel

Multi-layer coordinated adaptation based on graph refinement for cooperative activities

Ismael Bouassida Rodriguez[1], Nicolas Van Wambeke[1,2], Khalil Drira[1], Christophe Chassot[1,2] and Mohamed Jmaiel[3]

[1] LAAS-CNRS, Université de Toulouse; 7, av. du Colonel Roche, F-31077 Toulouse
[2] Université de Toulouse; INSA
[3] Redcad, Enis, Route de la Soukra Sfax, Tunisia
Email: {bouassida; van.wambeke; khalil; chassot}@laas.fr mohamed.jmaiel@enis.rnu.tn

Abstract: Future network environments are likely to be used by cooperative applications. Indeed, the recent advent of peer-to-peer systems where participants collaborate together in an ordered fashion motivates this assumption. In this paper, we present a method that relies on graphs as well as graph grammar productions in order to automatically refine a high level Service interactions representation of a given activity into a deployment topology at the Middleware and the Transport level. At the middleware level, a formal algorithm is presented in order to further optimize the solution. Similarly, at the Transport level, an analytical model to optimize the provisioning in the context of a modular transport protocol implementing collaborative congestion control is presented. The different models and algorithms are implemented in a case study of CMS-like operations for crisis management.

Keywords: dynamic re-configuration, self-adaptation, graph-grammar, context awareness, cooperative activities

1. Introduction

Cooperative group activities using wireless mobile communicating systems constitute an increasingly evolving application domain. It is likely to be one of the most important directions that may enable reliable and efficient human and machine-to-machine cooperation under the current networking systems and software, and may deeply shape their future deployment.

Such activity-support systems have to deal with dynamically evolving activity-level requirements under constantly changing network-level unpredictable constraints. Maintaining reliable connectivity and QoS in such a communication context is difficult. Adaptive service provisioning should help the different provisioning actors for achieving this goal and constitutes a challenge for different research communities.

Ad hoc solutions are likely to be not applicable to solve such a complex problem. Providing a basic framework for automated service and QoS deployment and management may constitute an important contribution towards solving such a problem. Aiming to answering this problem, we propose a formal model-based framework for adaptability management. Our framework has been elaborated in the context of Crisis Management System (CMS) with QoS provisioning at the transport and middleware levels as the final objectives.

This paper is organized as follows. Section 2 describes related work. Section 3 describes the targeted context used for the case study. Section 4 presents the different levels that we consider in our study. Section 5 provides details about the elaborated models, including optimization strategies though an example application. Finally, section 6 provides conclusions and future work.

2. Related works

Adaptation objectives, actions and properties are among the main facets of adaptability. They are studied and classified in this section.

Two different adaptability views may be distinguished: the design time adaptability [8] and the run time adaptability [4, 3]. For the first view, we can find design support tools like Adaptive Application Architecture (AAA). This tool handles the application development cycle and optimizes the resource value by insuring that the infrastructure answering clearly and in a measurable way to activity requirements. For the run time adaptability [10] presents several adaptation techniques among which use proxy services, change model of interaction and reorganize application structure.

Adaptation approaches are also targeting different architectural levels including Service, Middleware and Transport levels. At the first level, the Service-Oriented Architecture (SOA) paradigm is based on dynamically publishing and discovering services. This kind of architectures provides the possibilities to dynamically compose services for adapting applications to contexts. Service descriptions are published, via the registry, by service providers and dynamically discovered by service requesters.

Other frameworks are proposed to provide adaptability for the middleware level. As an example, an adaptive framework for supporting multiple classes of multimedia services with different QoS requirements in wireless cellular networks is proposed in [12].

At the transport level, dynamically configurable protocol architectures provide adaptive stacks based on the protocol module concept. A protocol module is a primitive building block resulting from the decomposition of the protocol's complexity into various successive elementary functions. A protocol is then viewed as the composition of various protocol modules in order to provide a global service. These architectures can be refined into two different categories depending on their internal structure: the event based model (followed by Coyote and Cactus) and the hierarchical model (X-Kernel and APPIA). ETP follows a hybrid approach combining both models. These protocol architectures appear as a good choice for future communication protocol's self-adaptation as they are capable of run-time architectural adaptation, meaning that the modules composing them can change during the communication. The adaptation solutions suggested in the literature distinguish behavioural and architectural aspects. The adaptation is behavioural (or algorithmic) when the behaviour of the adaptive service can be modified, without modifying its structure. Standard protocols such as TCP and specific protocols such as [13] provide behaviour-based adaptation mechanisms. Behavioural adaptation is easy to implement but limits the adaptability properties.

The adaptation is architectural when the service composition can be modified [1] dynamically. In self-adaptive applications components are created and connected, or removed and disconnected during the execution. The architectural changes respond to constraints related to the execution context involving, for example, variations of communication networks and processing resources. They may also respond to requirement evolution in the supported activities involving, for example, mobility of users and cooperation structure modification.

Designing and implementing self-adaptive communicating systems is a complex task. To handle this complexity, several studies showed the need to lay on model-based design approaches associated with automated management techniques.

Static architectures are described by instances of components and interconnection links. The dynamic character of architectures requires additional description rules. Several works have addressed the dynamic architecture description, using different approaches [2]. In order to guarantee the architecture evolving, correctness formal techniques are used. In particular, graphs represent a powerful expressive mean to specify respectively static and dynamic architectures aspects [11]. For such approaches, graph vertices represent the software components, and the edges represent the links between these components. Dynamic architectures are described as graph grammars and architecture transformation is specified and ruled using graph rewriting models.

3. Case study

To expose the targeted problems and concepts and to show the usefulness of the graph-based models, we consider the example of crisis management systems (CMS). We introduce this example and give two different execution steps and some related scenarios.

For CMS-like activities, cooperation is based on information exchanges between mobile participants collaborating to achieve a common mission. A CMS team is composed of participants having different roles: The controller of the mission, several coordinators, and several field investigators. Each group of investigators is supervised by a coordinator (Figure 1). Each participant is represented by his identifier, role and by the devices he uses. Each participant performs different functions. The controller's functions include monitoring and authorizing/managing actions to be done by coordinators and investigators. The controller is the entity which supervises the whole mission. The controller waits for data from his coordinators who synthesize the current situation of the mission. The controller has permanent energy resources and high communication and CPU capabilities. Coordinators that are attached to the controller, have to manage an evolving group of investigators during the mission and to assign tasks to each one of them. The coordinator has also to collect, interpret, summarize and diffuse information from and towards investigators. The

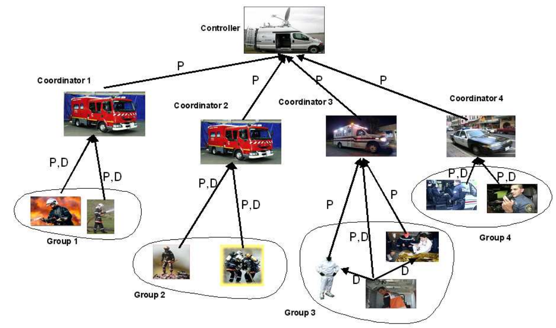

Fig. 1. CMS architecture

coordinator has high software and hardware capabilities. The investigator's functions include exploring the operational field, observing, analysing, and reporting about the situation.

Functions performed by investigators include generating Descriptive data (D) of the exploration field and Produced data (P) feedbacks to the controller. Two kinds of feedbacks are distinguished. Feedbacks D are descriptive data; they are transmitted by means of audio/video and real time text messaging. Feedbacks P are Produced data; they express the analysis of the situation by an investigator. They are transmitted by means of audio and real time text messaging.

The controller's function includes supervising the entire mission, i.e. deciding actions to be performed from the analysis of the observation feedbacks D transmitted by the coordinators. Initially, all investigator groups are in the "exploration step" where investigators provide continuous feedbacks D to the coordinator; they also provide periodical feedbacks P. The coordinator sends continuous feedbacks P to the controller.

When an investigator finds a critical situation, its group architecture has to be reconfigured in order to move to an execution step called "action step" where: the investigator that discovers the critical situation keeps sending both feedbacks D and P to the coordinator but also provides feedbacks P to the other investigators of its group. Other investigators report feedbacks P to coordinator on the basis of feedbacks D transmitted by the critical investigator. The coordinator continue sending feedbacks P to the controller.

In this senario, feedbacks D are more important than feedbacks P. When the critical situation is resolved, the investigation group comes back to the exploration step.

4. Model-based approach for adaptability management

Managing self-adaptability for optimizing QoS in CMS-like activities is complex as requirements and constraints evolve constantly requiring adaptation at multiple levels of the stack. This raises a coordination problem which may lead to sub optimal solutions. Adapting architectural adaptation may potentially be handled at different levels. For example, an energy constraint may be handled by modifying the servers' deployment at one level or by acting on the pull/push mode another level. Considering that servers consume more CPU as they serve many clients, we can suppose that they are less energy-efficient than clients. Moreover, considering that puller clients are more active than pushed clients, we can deduce that they need more CPU time and consume more energy. Both servers and puller clients may be placed on wired machines to save energy of mobile machines. However, actions at both these communication levels are not necessarily mandatory for a given energy loss rate and bandwidth constraints may lead to consider acting only on puller clients. Managing architectural adaptations requires defining and modeling abstractions levels dedicated to specific parts of the whole adaptation. This allows designers and developers to respectively master the design of adaptation rules. For a given configuration $A_{n,1}$ at level n, multiple configurations $(A_{n-1,1}, ..., A_{n-1,p})$ may be implemented at level $n-1$).

Adapting the architecture to constraint and requirement changes at level $n-1$ by switching among these multiple configurations allows maintaining unchanged the n-level configuration. Moreover, when adaptation requires changes at level n, this may need no changes at level $n-1$ if initial and new configurations of level n (e.g. changes from $A_{n,1}$ to $A_{n,2}$) have common implementations (e.g. $A_{n-1,p}$) at level $n-1$.

We consider three main abstraction levels for adaptability management which allow describing process-to-process, component-to-component and service-to-service architectural properties. From a communication point of view they represent respectively, the Transport layer, the Middleware layer and the upper users-oriented service layers. In the following, we will refer at these three levels as: the Transport-level adaptation (T-Adapt), the Middleware-level adaptation (M-Adapt) level and the Service-level adaptation (S-Adapt).

The S-Adapt level constitutes the highest level of the communication. It describes the services and their associated requirements and constraints provided by entities exchanging high level information. S-Adapt entities can for example represent the different roles the human participants may have within the considered activity. For CMS-like activities, depending on its role in the mission (e.g. controller, coordinators, investigators...) each participant has to perform a set of given functions (e.g. observe, report...).

The M-Adapt level is viewed as a component-to-component communication level aiming at supporting a given S-Adapt architecture, considering resource-related constraints. Three roles are distinguished: “event producers” (EP), “event consumers” (EC) and “channel manager” (CM).

The T-Adapt level constitutes the lowest level that we consider. It handles the process-to-process communications, i.e. the Transport level connections supporting the component-to-component communications of the M-Adapt level.

We consider distributed component-based applications deployed on mobile communication nodes. The communication has to be maintained adapted to the context change factors. These factors are given according to the application and the node properties. Being aware of these factors, that we call context, provides adaptability. The application and the node properties are: the mobile nodes move in a limited perimeter and each node has limited resources in term of energy and memory. The transport connections evolve in an open environment in which they will have to enforce “friendliness” and “fairness” [9] with regards to other non-cooperating connections. We drive the evolutions between levels by considering the context factors.

5. Graph-based refinement framework

We introduce our general framework of models. We use models based on graph grammar [5] to handle the architectural adaptation refinement management.

Graph grammars constitute a powerful and very expressive formalism for style description. Moreover, theoretical work on this field provides formal means to specify and check constraints on these architectures [14, 7]. We use productions of type $(L; K; R; C)$ where $(L; K; R)$ corresponds to the structure of a Double PushOut (DPO) production [6] and where C is a set of connection instructions. The instructions belonging to C are of the edNCE type [14]. They are specified by a system (n, δ, d, d') where n corresponds to a vertex belonging to the daughter graph R, δ is a vertex label, and d and d' are elements of the set in, out. For example, a production defined by the system $(L; K; R; (n, \delta, d, d'))$ is applicable to a graph G if it contains an occurrence of the mother graph L. The application of this production involves transforming G by deleting the subgraph $(Del = L \setminus K)$ and adding the subgraph $(Add = R \setminus K)$ while the subgraph K remains unchanged. All dangling edges will be removed. The execution of the connection instruction implies the introduction of an edge between the vertex n belonging to the daughter graph R and all vertices n' that are p-neighbours[1] of and d-neighbours[2]. This edge is introduced following the direction indicated by d' and labelled by q.

5.1 Refining from S-Adapt to M-Adapt

This section presents the steps and formalisms used to automatically refine a given configuration at the S-Adapt level to an optimal configuration at the M-Adapt level.

5.1.1 Graph grammar productions

In the following, we provide an example of graph grammar for our case study to implement architecture refinement. The proposed grammar generalize the use case by considering a variable number of investigators.

Following the commonly used conventions, we consider that vertices represent communicating entities (e.g. services, components) and edges correspond to their related interdependencies (e.g. communication links, composition dependencies).

For our study, we consider an architecture instance that includes a coordinator (coord) that manage three investigators (Inv). The graph edges are labeled by the exchanged data types and the priority of each type (D_H/P_L). Each participant have two attributes: the identifier and the deploying machine.

Architectural refinement deals with characterization of all the architectures that may be generated at an abstraction level n (Service) to implement a given architecture of the abstraction level $n-1$ (Middleware).

In this subsection we give the graph grammar addressing the refinement of any architecture of the S-Adapt level in all possible architectures of the M-Adapt level, during the exploration step. Since this graph grammar transforms S-Adapt architecture into M-Adapt architectures, its non-terminal nodes are S-Adapt entities while terminals nodes are M-Adapt enti-

[1] p-neighbours of a vertex n are all vertices n' such that there exists an edge labeled by p which connects n and n'.

[2] In-neighbours if $d = in$ and out-neighbours otherwise.

$GG_{S\rightarrow M,exp} = (AX, NT, T, P)$ with: $T = \{CM(cm, D, m), EC(ec, D, m), EP(ep, D, m)\}$, and $NT = \{Coord(c, m1), Inv(A, m2)\}$, and $P = \{p_1, p_2\}$
$p_1 = (L = \{Coord(co, m1), Inv(i1, m2), Inv(i2, m3), Inv(i1, m2) \xrightarrow{<D_H,P_L>} Coord(co, m1),$ $Inv(i2, m3) \xrightarrow{<D_H,P_L>} Coord(co, m1)\}; K = \{Inv(i1, m2), Inv(i2, m3)\};$ $R = \{EC(ec1, D, m1), EC(ec2, P, m1), CM(cm1, D, m2), CM(cm2, P, m3),$ $CM(cm1, D, m1) \xrightarrow{<Pull,1\|Push,0>} EC(ec1, D, m1), CM(cm2, P, m2) \xrightarrow{<Pull,1\|Push,0>} EC(ec2, P, m1),$ $Inv(i1, m2) \xrightarrow{<Pull,1\|Push,0>} CM(cm1, D, m2), Inv(i1, m2) \xrightarrow{<Pull,1\|Push,0>} CM(cm2, P, m3)$ $Inv(i2, m3) \xrightarrow{<Pull,1\|Push,0>} CM(cm1, D, m2), Inv(i2, m3) \xrightarrow{<Pull,1\|Push,0>} CM(cm2, P, m3)\}\ C = \{ic1, ic2\})$
$p_2 = (L = \{CM(cm1, D, m1), CM(cm2, P, m2), Inv(i, m3); Inv(i, m3) \xrightarrow{<Pull,1\|Push,0>} CM(cm1, D, m1),$ $Inv(i, m3) \xrightarrow{<Pull,1\|Push,0>} CM(cm2, P, m2)\}\ K = \{CM(cm1, D, m1), CM(cm2, P, m2)\};$ $R = \{EP(ep1, D, m3), EP(ep2, P, m3), EP(ep1, D, m3) \xrightarrow{<Pull,1\|Push,0>} CM(cm1, D, m1),$ $EP(ep2, P, m3) \xrightarrow{<Pull,1\|Push,0>} CM(cm2, P, m2)\}; C = \{\ \})$ with: $ic1 = (CM(cm1, D, m2), < D_H, P_L > /Pull, 1 \parallel Push, 0 >, Inv, in/in)$ $ic2 = (CM(cm2, P, m3), < D_H, P_L > /Pull, 1 \parallel Push, 0 >, Inv, in/in)$

Table 1. Refinement graph grammar $GG_{S\rightarrow M,exp}$

ties. $GG_{S\rightarrow M,exp}$ (Table 1) allows implementing this refinement in the exploration step.

Production $p1$ allows the refinement of the pattern consisting of the coordinator, and the two investigators that host the channel managers $CM1$ and $CM2$. Connection instructions $ic1$ and $ic2$ allow considering the pull/push options. Production $p2$ allows refining for other investigators. Figure 2 gives the refinement generated by $GG_{S\rightarrow M,exp}$ and depicts the case of three nodes represented with their communications and their priorities. The application of the sequence $p1; p2; p2; p2$ generates a configuration containing only terminal nodes (i.e. nodes belonging to the M-Adapt level).

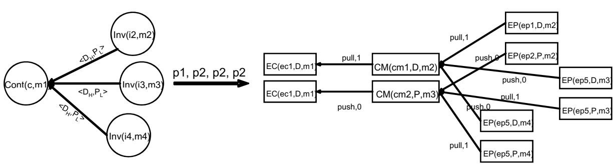

Fig. 2. Using $GG_{S\rightarrow M,exp}$ to achieve the refinement from S-Adapt to M-Adapt

5.1.2 Optimization at the Middleware level

The refinement graph grammar $GG_{S\rightarrow M,exp}$, for a given configuration $A_{n,i}$ at the service level, produce the set of configurations $(A_{n-1,1}, ..., A_{n-1,p})$ that can implemente at the middelware level. We provide an algorithm (Table 2) that allows to select the optimal implementation of $A_{n,i}$.

Let C denotes the context Let $A_{n,i}$ Let $S = \{A_{n-1,1}, ..., A_{n-1,p}\}$ the result of the refinement graph grammar $GG_{S\rightarrow M,exp}$ for $A_{n,i}$ Select $S_1 = \{A_{n-1,k} \in S\}$ such that: $Cost(A_{n-1,k}, C) \leq Cost(X, C), \forall X \in S\}$ if $card(S_1) \neq 1$ Select any configuration from S_1.

Table 2. Selection alogorithm

We define the context C as the percentage of the remain energy of each node (L_E) and the percentage of the remain memory of each node (L_M). We define also the fonction $Cost()$ (Table 2).We proceed by step. First, for each node (i) for a given deployment configuration $A_{n-1,p}$, we calculate an evaluation $V_i = \frac{\alpha L_E + \beta L_M}{\alpha + \beta}$. Where the values α and β are weights that allow an importance degree to be associated with each factor. For instance, if we know that for a specific node the memory saturation level is the most important factor, we set β to a value higher than α. Second, we calculate $Cost(A_{n-1,p}, C) = \frac{1}{N} \sum_1^N V_i$,where the value N is the node number of the deployment configuration $A_{n-1,p}$. In the end of this refinmement, we optain the optimal $A_{n-1,p}$ (Service level) that implements $A_{n,i}$ (Middelware level).

5.2 Refining from M-Adapt to T-Adapt

Given the M-Adapt graph obtained by the sucessive $p1; p2; p2; p2$ productions presented figure 2, it appears that the different connections have different priorities assigned to them, directly derived from the type of media that they transport. In a process similar to what has been described in the previous section to refine an element from S-Adapt into M-Adapt, a graph grammar ($GG_{M\rightarrow T,exp}$) generating a T-Adapt graph (see figure 3) can be written. However, it is not presented here due to space limitations.

The refinement graph grammar $GG_{M\rightarrow T,exp}$, for a given configuration $A_{n-1,i}$ at the Middleware level, produces a set of configurations $(A_{n-2,1}, ..., A_{n-2,p})$ at the transport level. A first step in transport provisioning consists in selecting the adequate $(A_{n-2,p})$ composition to be instantiated given the media that is being transport as well as the context in which the connection takes place. This step is presented in [15]. The approach relies on analytical models that define an optimization problem which's output is the most suitable composition for the modular protocol.

Given the collaborative nature of the case study, the analytical model that leads to the protocol compositions is extended in order to make it compulsory for them to include a congestion control that is "collaborative". A collaborative congestion control is a modified version of the standard algorithm in order to allow 2 or more connections to collaborate meaning that one of them will reduce its sending rate in order for the other to be able to acheive greater throughput. The drop and raise being simultaneous, the good properties of *fairness* and *friendliness* [9] are maintained towards non-collaborating connections.

In what follows, we describe a method that takes the data available on the M-Adapt graph as input in order to compute the parameters to be provided to a collaborative congestion control module instantiated in the dynamic transport protocol instance supporting the connection.

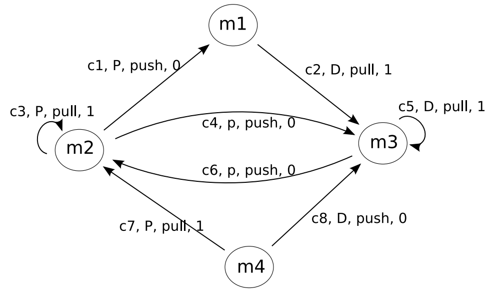

Fig. 3. Transport Connections Graph

Figure 3 presents the transport connections, it is derived from figure 2. Nodes represent the different machines. Arrows denote connections, they are labeled by a tuple representing the connection ID,the type of transported data, the service paradigm as well as the connection's priority.

Definition: Let $c_1..c_n$ the connections transporting the collaboration data. A connection c_i is defined as a pair: (S_i, D_i) where S designates the ID of the source node for c_i and D designates the ID of the destination node of c_i.

Definition: Let $\{c_i = (S_i, D_i), \quad c_j = (S_j, D_j)\}$ two connections, c_i and c_j are said to be *dependant* iif:
$(S_i = S_j \vee S_i = D_j) \wedge (D_i = S_j \vee D_i = D_j)$.

Definition: Let D the dependency matrix for the activity such that:

$$D_{i,j} = \begin{cases} 1 & \text{if } c_i \text{ and } c_j \text{ are dependant} \\ 0 & \text{otherwise} \end{cases}$$

Given the above definitions, the ratio of the total available bandwidth between two nodes to be consumed by each connection without taking priorities into account is given (in an activity composed by n connections) by: $F_i = \frac{D_{i,j}}{\sum_{j=1}^{n} D_{i,j}}$

Let P_i the priority associated to connection i. In order to take this priority into account, the F_i expression is modified as follows: $F_i = \frac{(P_i+1)}{\sum_{j=1}^{n} D_{i,j}(P_j+1)}$

For the current scenario, the graph presented on figure 3 leads to the following dependency matrix:

$$\mathbf{D} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The priorities vector is: $\quad \mathbf{P} = (0, 1, 1, 0, 1, 0, 1, 0)$

Applying the previously presented heuristics, we obtain $\mathbf{F} = \left(1, 1, 1, \frac{1}{2}, 1, \frac{1}{2}, 1, 1\right)$ the parameters vector to be fed to the collaborative congestion control.

As it can be observed, connection's which don't "collaborate" have a fraction of 1. This instructs the collaborative congestion control to use the standard algorithm. On the other hand, c_4 and c_6 have a fraction of $\frac{1}{2}$ indicating that they share resources with another collaborating connection of the same priority.

6. Conclusion

In this paper, we presented a framework for the context-aware dynamic provisioning and automated management of group cooperative activities. All the levels ranging from Service to Transport have been introduced. The graph formalism used to model each of the levels has been illustrated in the context of CMS-like activities. In this context, the graph grammar as well as a the productions required for automated top-down refinement of the models have been explicitly presented. Finally, algorithms and models to further refine the decision at the Middleware and the Transport level have been introduced to take advantage of context information captured in a CMS-like case study.

Future works include the implementation of the models and algorithms presented in this paper and the introduction of a coordination approach to avoid conflict between the adaptation at different levels. Moreover, the integration of the optimization algorithms as well as the graph transformation modules with a monitoring and measurements system would provide all the needed inputs for the deployment of the solution in controlled network environments.

References

[1] In *IEEE Std 1471-2000, IEEE Recommended practice for architectural description of software-intensive systems*, pages i–23, 2000.

[2] R. Allen and D. Garlan. A formal basis for architectural connection. *ACM Transactions on Software Engineering and Methodology*, 6(3):213–249, 1997.

[3] K. Bade, E. W. D. Luca, A. Nürnberger, and S. Stober. Carsa - an architecture for the development of context adaptive retrieval systems. In K. van Rijsbergen, A. Nürnberger, J. M. Jose, and M. Detyniecki, editors, *Adaptive Multimedia Retrieval: User, Context, and Feedback*. Springer-Verlag, 2006.

[4] F. Chang and V. Karamcheti. Automatic configuration and runtime adaptation of distributed applications. In *HPDC*, pages 11–20, 2000.

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

[6] 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.

[7] H. Ehrig and H.-J. Kreowski. *Graph Grammars and Their Application to Computer Science: 4th International Workshop, Bremen, Germany, March 5-9, 1990 Proceedings*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1991.

[8] H. Fahmy and R. Holt. Using graph rewriting to specify software architectural transformations. In 15^{th} *IEEE international Conference on Automated Software Engineering, ISBN 0-7695-0710-7*, pages 187–196, Grenoble, France, September 2000.

[9] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *SIGCOMM*, pages 43–56, 2000.

[10] A. Friday, N. Davies, G. Blair, and K. Cheverst. Developing adaptive applications: The most experience. *Integrated Computer-Aided Engineering*, 6(2):143– 157, 2000.

[11] D. L. Metayer. Describing software architecture styles using graph grammars. *IEEE Transactions On Software Engineering*, 24(7):521–533, July 1998.

[12] N. Nasser and H. Hassanein. Adaptive bandwidth framework for provisioning connection-level qos for next-generation wireless cellular networks. *Canadian Journal of Electrical and Computer Engineering*, 29(1):101–108, 2004.

[13] Özgür B. Akan and I. F. Akyildiz. Atl: an adaptive transport layer suite for next-generation wireless internet. *IEEE Journal on Selected Areas in Communications*, 22(5):802–817, 2004.

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

[15] N. Van Wambeke, F. Armando, C. Chassota, and E. Expósito. A model-based approach for self-adaptive transport protocols. *doi:10.1016/j.comcom.2008.02.026, Computer Communications*, 2008.