Adaptive Policy-based Approach for Static and Dynamic Policy Conflict Detection

Abdelhamid Abdelhadi Mansor[1], Wan M.N. Wan Kadir[2], Toni Anwar[3], Hidayah Elias[4]

[1]Department of Computer Sciences, Faculty of Mathematical Sciences, University of Khartoum, Sudan

[2,3,4] Software Engineering Department, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Malaysia

E-mail: [1]abhamidhn@gmail.com, [2]wnasir@cs.utm.my, [3]tonianwar@utm.my, [4]shahlida@gmail.com

ABSTRACT

Policy-based approach has been mostly acknowledged as a methodology that separates the rules governing the behavior of a system from its functionality. It provides the ability to (re-)configure differentiated services networks so that desired Quality of Service (QoS) goals are achieved, by considering administratively specified rules. Moreover, it promises to reduce maintenance costs of information and communication systems while improving flexibility and runtime adaptability. This paper presents a brief description on the properties of the most prominent approaches, identifying their advantages and disadvantages. Some evaluation criteria have been used to determine our research direction. We strongly believe that the results presented in this paper may provide some foundations to develop our framework. Furthermore, the proposed framework depends on static and dynamic analysis to reduce potential errors and avoid potential conflicts.

Keywords—Policy-based, Policy conflict, Static Analysis, Dynamic Analysis, Adaptive Systems

1. INTRODUCTION

A policy [1] is represented as a means to control when a managed object translated to a new state. The subject of a policy specifies the human or automated managers to which the policies apply. The target of a policy specifies the actions to be performed. Domains are a means of grouping objects and are similar to file system directories [2]. The subject or target of a policy is expressed as a domain of objects and the policy applies to all objects in the domain; so a single policy can be specified for a group of policies. This helps to cater for large-scale systems in that it is not necessary to define separate policies for individual objects in the system, but rather for groups of objects.

Policies can be classified into access-control, obligation, goal-based and meta-policies based on their purpose. Access control policies specify what actions entities can or cannot perform in a system [3]. This type is further classified into Authorization policies, which define what activities a member of the subject domain can perform on the set of objects in the target domain [4]. Delegation policies, which transfer access rights from one entity to another, and Information filtering policies which implement privacy by data obfuscation. For example, the location information of a mobile node can be reported with lesser accuracy to prevent the exact position from being revealed using information filtering policies.

Obligation policies specify what actions entities must or must not perform in a system [3]. Moreover, they are used for fault and configuration and file system management, and so on. Goal-based policies are used to specify the final system state that should be reached from a given state, and meta-policies, which guide the behavior of the management system. Furthermore, they are used to modify policies, resolve conflicts dynamically and change various parameters of the management system.

Human error is one obstacle to accurate access-control policies; the policy authors who assign and maintain these policies are prone to making specification errors that lead to incorrect policies. Access-control policies consist of a set of rules that dictates the conditions under which users will be allowed access to resources. These rules may conflict with each other.

This paper discusses the advantages and disadvantages of the most prominent approaches and presents a brief description on the properties. Some evaluation criteria have been used to determine our research direction.

The rest of this paper is organized as follows. Section 2, classifies conflict between policies. Section 3, discusses the most prominent approaches on policy-based approach. Section 4 introduces explanation on the criteria used in the evaluation. In Section 5, we discuss the outcome of the evaluation result. Section 6 briefly presents the proposed framework. In Section 7, we present our conclusions and plans for the future work.

2. THE CONCEPT OF POLICY CONFLICT

Conflicts may arise in the set of policies and also may arise during the refinement process, between the high-level goals and the implementable policies [5]. For example an obligation policy defines an activity a manager must perform but there is no authorization policy to permit the manager to perform the activity. The system must have to cater for conflicts such as exceptions to normal authorization policies. For instance, in a large distributed system there will be multiple human administrators specifying policies which are stored in distributed policy servers. Conflict detection between management policies can be performed statically for a set of policies in a policy server as part of the policy specification process or at run-time [6], [7].

Conflicts between policies can be classified into four broad categories [8]. Each category may present itself either statically or dynamically. First, internal policy conflict, occurs when there is incompatibility between policies which are assigned to single roles, second, external policy conflict, occurs when combining roles which in isolation of each other present no conflict, but contain policies which in co-existence are in conflict. Third, policy space conflict, occurs when more than one policy space manage the same set of subjects and attempt to enforce various and conflicting policies over them, and fourth role conflict, expected when a user obtains a set of incompatible role assignments.

Policy-based approach uses policies to govern their behavioral choices whilst satisfying the goals of the system, in addition to specify and enforce QoS management in distributed systems. Furthermore, it provides flexibility, adaptability and support to automatically assign network resources [9]. A policy-based management system must provide guarantees when multiple rules need to be enforced concurrently, so that the system behaviour is predictable. However, existing policy-based management systems based on Event Condition Action (ECA) rules do not contain specifications of actions required for reasoning and so do not provide guarantees which can lead to unpredictable system states [10].

3. RELATED WORKS

Many works on policy-based approach discussed policy conflicts using various techniques such as static analysis to reduce potential errors [11], [12], [13], [14], [15], [16] and dynamic analysis to detect and resolve potential conflicts [12], [16], [17], a verification of a policy-conflict process in [12], [16], and a system scalability discussed in [13].

Shiva [12] proposed an extended model of Event-Condition-Action (ECA) called ECA-Post-condition to enable developers and administrators to annotate actions with their effects. The ECA-P model allows deducing that action, which may conflict based on conflicting post-condition; furthermore the framework also uses static and dynamic conflict detection techniques to detect failure in policy execution by using post condition to verify successful completion of policy actions. However, Policy actions may not execute to completion due to various reasons such as changing active space configuration, device and component failure or software errors.

Wu et al [14] introduced dynamic analysis mechanism to ensure consistency among the policies enforce, they used Event Calculus (EC) in the policy analysis to provide a dynamic policy conflict analysis to detect and control dynamic conflicts in trust services for federations. However, their work does not take targets constraints into account, while some of these conflicts are caused by overlapped elements. Davy et al. [11] presented an efficient policy selection process for policy conflict analysis to improve the performance depending on the nature of the relationships between deployed policies. Their process targets pre-deployment identification of potential conflicts between a modified or newly created policy and already deployed policies. They use a tree based data structure to reduce the number of comparisons and therefore reduce runtime complexity in subsequent iterations by maintaining a history of previous policies comparisons. Their conflict analysis algorithm initiates a relationship pattern matrix between candidate and deployed policies, and matches these patterns against a conflict signature. However, this approach is not intelligent and repeats over all deployed policies to ensure that the deployed policy does not cause a potential conflict. Also the algorithm is still limited to detect only conflicts that can be represented as relationships among policies.

In another related work [13] Davy et al. produce a policy conflict analysis approach makes extensive use of information models and ontologies to make it a flexible tool to analyze for conflict in a range of applications. Furthermore, they introduce a novel pre-analysis policy selection to reduce the number of more comprehensive policy analysis operations required. Similar like previous work they use heuristics and historical information from previous comparisons to eliminate group of policies from analysis. Moreover, they separate the definition of a policy conflict from the definition of the conflict analysis algorithm; thereby the approach is extensible and efficient. However, this algorithm needs further improvement; because it eliminates policies instead of refine them. Eliminating some policies does not achieve the system goals and reduce the scalability.

Mohan et al [15] proposed an attribute-based authorization framework that supports changing the rules and policy combination algorithm dynamically based on contextual information. The framework eliminates the need to re-compose the policies when the combination algorithm changes. Moreover, it provides a method to add and remove specialized policies dynamically, in addition to its capability to reduce the set of potential target matches, thus increasing the efficiency of the evaluation mechanism. Furthermore, to resolve the conflicts they use Policy Combination Algorithms (PCA), these algorithms take the authorization decision from each policy as input. However, in a highly dynamic environment these algorithms will lead to reduce the performance.

Khakpour et al. [16] presented an analysis using Rebeca [18] which is an actor-based language for modeling concurrent asynchronous systems which allows to model the system as a set of reactive objects called rebecs, interacting by message passing. In order to introduce this, a new classification of conflicts may occur during governing policies. They also proposed Linear Temporal Language LTL [19], which expresses each type of conflicts and enables to automate detection of conflicts patterns to classify conflict types, thereby to automate a significant portion of policy analysis process. Moreover, they introduced a number of correctness properties of the adaptation process in the context of their models. Then, they used static analysis of adaptation policies in addition to model checking technique to verify those properties. Whenever an event which requires adaptation occurs, relevant managers are informed. However, the adaptation cannot be done immediately and when the system reaches a safe state, the manager switches to the new configuration. While their system includes many different managers each manager uses a set of policies to govern system sensors and actors. There may be more than an event, which require adaptation, occur simultaneously, and this will reduce the system scalability.

Ma et al. [20] proposed conflict detection and resolution in workflow management systems (WFMSs) approaches to help workflow designers in constructing a flexible, consistent workflow authorization schema. In this work a new type of constraint, context constraint, is proposed since context constraints can meet the complicated requirements of security policies in WFMSs. Moreover, they define an effective set of rules to detect and resolution of static and dynamic conflict for authorization policies in WFMSs. Furthermore, they classify conflicts into two broad categories i.e. (i) policy-policy conflicts which occur when two or more authorization policies are considered incompatible, and (ii) policy constraint conflicts which occur when the performance of two or more authorization policies will lead to situations that are prohibited by other constraints (e.g., separation-of-duty constraints) in the system. However, their works do not put into account conflicts in authorization policy itself, in addition to policies are considered to assign by different administrators.

Table 1 shows a simplified view of the comparison between the presented works, the comparison based on the criteria defined in the following section.

4. EVALUATION CRITERIA

4.1. Dynamic Conflicts Analysis

Dynamic analysis makes use of meta-information at runtime to detect and control potential conflicts among different policies which cannot be detected during the compilation time [17].

4.2.Static Conflicts Analysis

Static analysis is used by the policy compiler to detect specification errors and to reduce run-time conflicts which occurs among rules; whose event and condition parts can be statically matched; it may not be able to evaluate policy constraints, as conflicts may depend on the run-time state of the system [12].

4.3.Policy Decoupling

Decoupling of policies refers to decompose long policies into several policy segments or a small functional policy unit which describes a complete behavior. Each segment contains an object set that describes an action's target, a subject set to identify the action's executor, an action set which represents a temporary binding between subjects and objects, and an additional related information set [17].

4.4.Policy Classification

Classification of conflicts is needed during development time and completely depends on the type of actions .According to the informal definition of conflicts, the classification of various conflicts may exist among interacting governing policies. Rebeca language [16] is used to introduce a new classification of conflicts, in addition to providing temporal specification patterns to discover such conflicts.

4.5.Policies Combination

A combination of Policies refers to combined several segments or units of policies before enforcing them together. Moreover, it is very important when there are multiple policy authors defining policies for a given system [11]. The combined actions of the policies will result in the system reaching different final states depending on the order of execution of these actions.

4.6.Conflict Avoidance

Avoidance is a method that deals with conflict which attempts to avoid directly confronting the issue at hand [21]. Such methods can include changing the subject, putting off a discussion until later, or simply not bringing up the subject of contention. However, conflict avoidance method is time consuming and costly, thus it is better to use as a temporary measure.

4.7.Correctness of Adaptation

Correctness refers to the verification, that a software system meets a user’s needs, also to ensure that we are building the product right and the software should conform to its specification. The verification system checks if the condition of an action is true after the action completes execution [18]. Moreover, the verification system determines the failed propositions and forwards them to the exception generation system along with the enforcement context of the failed rule. The software inspections which concern with analysis of the static system representation to discover problems, is defined as dynamic verification, and a software testing that concerns with exercising and observing product behaviour defined as a static verification [12].

4.8.Check the System Scalability

Scalability is important for an adaptive software to prevent a difficult software evolution [22]. It refers to the capability of a system to increase total throughput under an increased load when resources (typically hardware) are added. Moreover, it indicates its ability to either handle growing amounts of work in a graceful manner or to be enlarged [23].

5. OUTCOME OF THE COMPARATIVE EVALUATION

The simplified results in Table 1 shows that all selected approaches covered dynamic analysis to detect conflicts at runtime, while conflicts detected dynamically, during runtime, are resolved by resolution policies.

However, detecting conflicts statically are resolved by the user before the enforcement of policies. Some approaches used static analysis to reduce the potential errors. It is very important to use static analysis, because after a policy is compiled, detected conflicts are resolved by the user before generating the policy object file. Detecting conflicts among rules are done by matching these rules events and conditioning statically; to determine matching it is required to compare event symbols and types of the rules parameters. Furthermore, dynamic analysis during runtime is required since all rule conflicts cannot be detected during the static analysis done at the compilation stage. After that, rules must be combined to use dynamic conflict technique to detect potential conflicts during run time.

From the shown results in the table, it is clear that current works leave some gaps between the used techniques and conflict specification. To cover these gaps, the relation between different criteria such as: combination and decoupling of policies, classification of rules, scalability, correctness and conflict avoidance must be taken into account, since the classification of rules occurs after decoupling long policies into segments, to match them statically. However, the combination of these segments into policies is required before the final evaluation.

Obviously there is a limitation in developing policy-based management models that do not provide ensuing support to detect and resolve conflicts. While a considerable attempt at static conflict detection has been presented in [5], the very complex and crucial issue of dynamic conflict detection in policy-based management has gone largely unresolved. Moreover, current research has revealed that there is still a large class of policy conflict which simply cannot be determined statically.

Static and dynamic conflicts are considered as two classes of conflict which need to be understood and independently managed [8]. Furthermore, the distinction between these two classed is important; as detecting and resolving of conflict can be computationally intensive, time consuming and hence, costly and is most preferably done at compile-time. However, a dynamic conflict is quite unpredictable, in that it may, or may not; proceed to a state of a realized conflict. This class of conflict must be detected at run-time.

Our research direction is to develop an adaptive architectural framework to avoid potential errors and policy conflicts. The main effort to develop the framework completely depends on checking system scalability in order to improve the system adaptability.

TABLE 1: A comparative of Policy-Based Approaches

Criteria	Approaches						
	ECA-P	Dynamic Policy Conflict Analysis	Policy Conflict Analysis for Autonomic Network	An Attribute-based Authorization	Automatic Policy Conflict Analysis	PobSAM	Conflict detection and resolution in WFMSs
Dynamic Conflicts	√	√	√	√	√	√	√
Static Conflicts	√	√				√	√
Policy Decoupling		√					
Policy Classification			√			√	√
Policies Combination		√	√	√	√		
Conflict Avoidance				√			
Check the System Scalability			√				
Correctness of Adaptation	√					√	

6. THE PROPOSED FRAMEWORK

The proposed framework [24] provides support for both, behavioral and structural changes which cause the issue of policies to govern the system. In administrative terms, the framework considers five main components of the framework are shown in Fig. 1. below which view a high level of the framework. Each component of the framework has a specific responsibility;

- ***Policy Refinement***
  This component depends on the application and is carried out by the administrator developer. Consequently this component is carried out during the design and implementation of the system. This will reduce the potential errors, but during execution of policy, this activity which is intended to check each type of conflict and uses the priority of execution rules, elimination rules or change them according to the request, is a part of this activity role [25].
- ***Dynamic Conflict Resolver***
  Combination and evaluation of policies is considered as a part of this component. Since each policy gives a single decision, the policy combination algorithms (PCAs) combine these decisions into a single policy decision. PCAs use to resolve conflicts during runtime. These algorithms take the authorization decision from each policy as an input and apply some standard logics to come up with a single decision. There is a need to include algorithms such as these as PCAs in authorization languages to provide more functionality and flexibility in defining policies.
- ***Policy Verification***
  Verification depends on decoupling of the adaptation logic from its functional logic (its business logic). Thus, an adaptation layer can be verified independently from the actor layer provided. Moreover, policy verification verifies the action and purpose specified by the user; in PobMC we assume that what is stated by the user is correct.
- ***Context Monitor***
  The monitoring of the operating environment helps to detect structural and behavioral changes. For instance, sensors and actors state malfunction of devices or new devices in addition to the number of working sensors and the state of non-working. Collected information about managers and their states, which are stored in the variable states, helps managers to govern system changes and coordinate their tasks. The Context Monitor allows users to register and log in and query the system for resources using various APIs and receive requests. Moreover, to check if the detected event is allowed or denied based on the setup time information that it has received

from the policy analyzer. If allowed, it checks if there is an obligation mandated by the relevant rule, then the Context Monitor informs the Self-Coordinator (which is the obligation enforcement component in PobMC), the Self-Coordinator marks the resource item in the corresponding file, based on the resource type. Subsequently, the Self-coordinator informs the request Context Monitor on the ‘Allow’ or ‘Deny’ ruling, as applicable. The Context monitor then displays the permitted results to the user. In addition to the mentioned functions, Context monitor observes the execution of obligation over the runtime periods, since some obligations could be defined to take effect much later in time than the time of resource access.

- ***Self-Coordinator Component***

  This component is the core of the system which coordinates all the activities during runtime. Each policy is checked first by this process before triggering execution of processing. Moreover, any task taken by each process must be checked in this process in order to take the right decision.

  The self-coordinator determines the triggered rules, and uses the *ActionCondition* checker to test the action and rule condition expressions. If a condition evaluates to be true the rule is added to the policy live list.

  Once the static conflicts have been detected and resolved, the *policy compiler* compiles the policies including the resources constraints, then generates a policy object file. The *policy loader* loads the generated object file into the Self-coordinator component before it is evaluated to detect and resolve potential dynamic conflicts.

  A library of actions stored in *Action Library* can be invoked from the action part of the policy rule. When an event occurs in a situation where condition is true, then the action is a call to a method in a library of actions where each action is annotated with a post-condition by the programmer. These post-conditions of the actions are used for a conflict detection. Event Receiver is responsible for subscribing and receiving events since they occurred and have been detected by the Context Monitor. Then *Event Receiver* verifies the types of the parameters in the events and notifies the self-coordinator of the event occurrence along with the parameters.

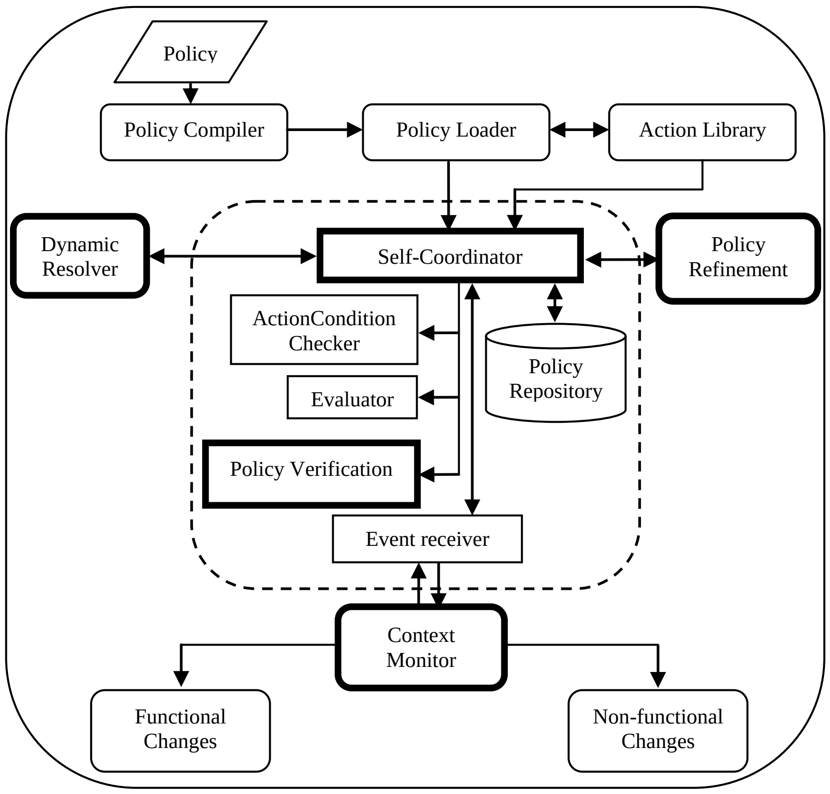

Figure 1. High level view of the adaptive framework

7. CONCLUSION AND FUTURE WORK

A comparison based on important techniques and criteria used to address the weaknesses of policy-based approaches is presented in this paper. The comparison outlined the main aspects of the current research in a policy conflict, which requires a further investigation to address these aspects. One of the main efforts is to identify the opportunities for improvement based on current approaches. The improvement will be implemented by the proposed framework.

We showed that existing policy-based systems do not reason about concurrent rule enforcements and define no enforcement ordering. Furthermore, they do not verify an action execution and assume that a rule enforcement was successful. In addition to all these drawbacks most of previous works do not thoroughly investigate the effects of different policies. Based on these facts, policy-based systems are still suffering from many weaknesses such as, the scalability which needs to be checked when policies are assigned by different administrators, a previous information is needed to avoid potential conflicts, and there is a need for effective tools to verify the adaptability of policy-based systems before during and after adaptation.

We have proposed an adaptive framework based on Event-Condition-Action (ECA) rules for policy-based management distributed system. The proposed framework uses appropriate mechanisms to detect potential conflicts by decoupling and classifying policies, in order to classify each type of conflicts, in addition to the ability to resolve conflicts mechanism during runtime.

Our future works will concentrate on discussing static and dynamic analysis approaches that make extensive use of information models and ontologies to make it flexible and scalable enough to be used as a tool to analyze for a conflict in a range of applications. We will use heuristics and historical information from previous comparisons to aid in the elimination of groups of policies from analysis.

8. ACKNOWLEDGEMENT

The authors would like to express their deepest gratitude to Universiti Teknologi Malaysia (UTM) for their financial support under Research University Grant Scheme (Vot number Q.J130000.7128.01H13).

9.REFERENCES

1. Strassner, C.J. Policy-based Network Management, Solutions for the Next Generation. Elsevier, Morgan Kaufmann Publishers. ISBN: 1-55860-859-1, (2004).
2. Sloman, M. and Twidle, K. Domains. (1994). A Framework for Structuring anagement Policy. In Network and Distributed Systems Management, Addison Wesley, (1994), pp. 433–453.
3. Sloman, M. Policy driven management for distributed systems. Journal of Network and Systems Management, (1994),pp.333–360.
4. Nicodemos, C. Damianou. A Policy Framework for Management of Distributed Systems. PhD dissertation, Imperial College, London, (2002).
5. Lupu, E., Sloman M. Conflicts in Policy-based Distributed Systems Management, IEEE Transactions on Software Engineering – Special Issue on Inconsistency Management ,(1999).
6. Sibley, E., Wexelblat R.L., Michael J.B., Tanner M.C., and D.C. Littman. The Role of Policy in Requirements Definition. In IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, Los Alamitos, California, (1993), pp. 277-280.
7. Michael, J., Sibley E., and Littman D. Integration of Formal and Heuristic Reasoning as a Basis for Testing and Debugging Computer Security Policy. In Proceedings of the New Security Paradigms Workshop, IEEE Computer Society Press, Los Alamitos, California, (1993), pp. 69-75.
8. Dunlop, N., Jadwiga Indulska, Kerry Raymond, Dynamic ConJlict Detection in Policy-Based Management Svstems , IEEE Enterprise Distributed Object Computing Conference (EDOC'2002), Lausanne, Sept (2002).
9. Wang, G., Alice C., Haiqin W., Yichi P., Casey F., Stephen U. A Policy-Based Approach for QoS Specification and Enforcement in Distributed Service-Oriented Architecture, Proceedings of the IEEE International Conference on Services Computing (SCC'05), (2005).
10. Shiva, C.-S. Policy-based Pervasive Systems management Using Specification-Enhanced Rules, Dissertation for the degree of Doctor of Philosophy in Computer Science, University of Illinois at Urbana-Champaign, (2006).
11. Davy, S. Jennings, B. Strassner, J. Efficient policy conflict analysis for autonomic network management. 5th IEEE Workshop on Engineering of Autonomic and Autonomous Systems, EASe, March 31- April 4, 2008, Belfast, Ireland, Inst. of Elec. and Elec. Eng. Computer Society, (2008).
12. Shiva, C.-S. Anand Ranganathan and Roy Campbell, An ECA-P Policy-based Framework for Managing Ubiquitous Computing Environments, Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'05), 0-7695-2375-7, IEEE, (2005).
13. Davy, S. Jennings, B. Strassner, J. On harnessing information models and ontologies for policy conflict analysis. 2009 IFIP/IEEE International Symposium on Integrated Network Management, IM, June (1-5) 2009, New York, NY, United states, IEEE Computer Society, (2009).
14. Wu, Z. Liu, Y. Wang, L. Dynamic policy conflict analysis in operational intensive trust services for cross-domain federations. 1st International Conference on Intensive Applications and Services, INTENSIVE, April (20-25) 2009, Valencia, Spain, Inst. of Elec. and Elec. Eng. Computer Society, (2009).
15. Mohan Apurva, Douglas M. Blough, An Attribute-based Authorization Policy Framework with Dynamic Conflict Resolution, Gaithersburg, MD, ACM ISBN, 978-1-60558-895-7/10/04, (2010).
16. Khakpour, N. Khosravi, R. Sirjani, M. Jalili, S. Formal analysis of policy-based self-adaptive systems. 25th Annual ACM Symposium on Applied Computing, SAC 2010, March(22- 26) 2010, Sierre, Switzerland, Association for Computing Machinery (2010).
17. Wu, Z. and Y. Liu. Automatic policy conflict analysis for cross-domain collaborations using semantic temporal logic. 2009 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom, November (11-14) 2009, Washington, DC, United states, IEEE Computer Society (2009).
18. Sirjani, M., A. Movaghar, A. Shali, and F. S. de Boer. Modeling and veri_cation of reactive systems using Rebeca. Fundamamenta Informaticae, (2004), pp.385-410.
19. Manna, Z. and Pnueli A. The Temporal Logic of Reactive and Concurrent Systems: Specification". Springer-Verlag, (1992).
20. Ma, C. Lu, G. Qiu, J. Conflict detection and resolution for authorization policies in workflow systems. Journal of Zhejiang University-Science, (2009), pp.1082-1092.
21. Bacal, Is Conflict Prevention The Same As Conflict Avoidance, www.work911.com/conflict/carticles/conav.htm
22. A. Mansor and W. M. N. Wan-Kadir, A Comparative Evaluation of State-of-the-Art Approaches in the Design of an Adaptive Software System. Kuala Lampur ASME Press, (2011).

23. Bondi André B. Characteristics of scalability and their impact on performance, Proceedings of the 2nd international workshop on Software and performance, Ottawa, Ontario, Canada, ISBN 1-58113-195-X, (2000), pp. 195 – 203.
24. Mansor, A. W. M. N. Wan-Kadir, Elias, H. Policy-based Approach for Dynamic Architectural Adaptation: A Case Study on Location-Based System, MySEC011, (12-14) December (2011).
25. Mansor, A. Wan M.N. Wan Kadir, Toni Anwar & Elias, H. Policy-based Approach to Detect and Resolve Policy Conflict for Static and Dynamic Architecture Journal of Theoretical and Applied Information Technology, 31st March (2012).