# Efficiency Considerations in Policy Based Management in Resource Constrained Devices

Jignesh Kakkad and Nandan Parameswaran

School of Computer Science and Engineering, The University of New South Wales
Sydney, NSW 2052, Australia
{jmka292,paramesh}@cse.unsw.edu.au

**Abstract.** Policies are being widely used in a variety of applications such as healthcare, disaster management and mobile networking. In this paper, we show how policies can be used to manage the resources effectively and in a user friendly way. Further, we advocate that while an agent is required to obey a given policy requirement, there are situations where the agent may consider the possibility of violating the policy (policy deviation) such as in an emergency or during a disaster. Our simulation results show that sometimes policy violations can be beneficial to the community of (application) agents and such violations must be managed *carefully*.

**Keywords:** Policy, Mobile Agent, Resource Management, Policy Violation.

## 1 Introduction

Management involves the process of controlling entities in an organization and it often involves stipulating policies and enforcing them. Policies can be represented in their simplest form by a set of rules which define the behavior of objects involved in a system situated in a given environment and they have been used effectively to achieve flexibility in complex distributed systems [2], [3], [9], [10], [11]. There exist a number of policy languages such as Rei, Ponder2 [4] and XACML in which policies can be written. In this paper, we investigate use of policies for resource management, and propose a measure of policy violation by relating it to the overall performance of the agent community.

The rest of the paper is organized as follows. In Section 2, we discuss policies with an example. Section 3 discusses policy based resource management in a mobile device. In Section 4, we present policy violation. In Section 5, we discuss measuring policy performance. Sections 6 and 7 present related work and conclusion, respectively.

## 2 Policy

Currently, mobile phone users are subjected to policies proposed by various agencies such as telecom operators (Service provider), phone manufacturers, service providers

R. Li, J. Cao, and J. Bourgeois (Eds.): GPC 2012, LNCS 7296, pp. 210–220, 2012.

(through their terms and conditions, for example, from Service provider) and government agencies. An example of polices is as follows:

**Rule 1**
***if*** *(current offer = "pre-paid cap+" && recharge amount == $30)* ***then***
*current balance += current Data;*
*current limit += 400MB;*

**Rule 2**
***if*** *(current offer = "Telstra long life" && recharge amount == $20)* ***then***
***c****urrent expiry += 60 days;*

Apart from mobile usage policies, there are terms and conditions (T&C) imposed by Service providers as well.

## 3 Policy Based Resource Management in a Mobile Device

Fig. 1 shows a simple architecture for policy based resource management consisting of a set of application agents and a set of resources. Access to resources are managed by a resource monitoring agent (MA). The monitoring agent maintains a set of policy rules and the history of the states of the resources. A request from an agent consists of an action *a* to be performed on behalf of the agent. When an application agent puts in a request, the monitoring agent evaluates the request using the states of available resources and policy rules and forwards the request to the resource operator. The resource operator executes the action *a* and the resulting new states of the involved resources are stored for future use. (More resources may be added if the action is modeled with more details of resources.)

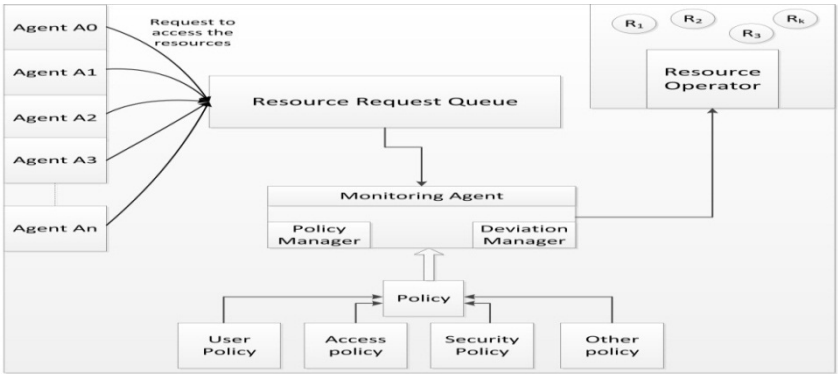



**Fig. 1.** An architecture for policy based resource management in a mobile phone

The monitoring agent has two sub parts: the Policy Manager which is responsible for the overall execution of the requested action; and the Deviation Manager which manages the agent request when the execution of these requests may result in the violation of policy conditions (discussed in the next section).

### 3.1 Policy Model

A policy in a multiagent system dictates how a set of resources must be operated by the agents in the world. As the agents perform operations on the resources, the resources change from one state $s_i$ to $s_j$. (We often refer to the aggregate state of the resources as the world state.)

### 3.2 Option Graph

We model the world as a synchronous finite state machine (fsm) where the fsm goes from one state $s_i$ to another $s_j$ at the time when a clock occurs as defined by the transition function $\delta$. Let S be the set of states, $\Gamma$ be the set of actions an agent can perform, and $\Lambda$ be the set of external events. Then, $\delta$: S $X$ ($\Gamma \cup \Lambda$) $\rightarrow$ S.

We let $\eta$ to denote a null action, which we will use to characterize a situation where an agent chooses not to do any action. Thus, $\eta \in \Gamma$, and $\delta(s_i, \eta) = s_i$. That is, a null action does not change the state of the world.

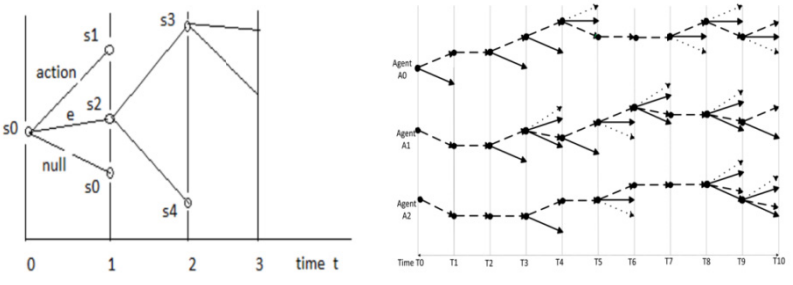



**Fig. 2.** Option graph and Option graph for three agents with policy. Options (dashed and dotted arrows) define the policy.

An option graph is a directed acyclic graph where the nodes of the graph represent the states of the world and edges denote transitions from one state to another which occur when actions are performed (See Fig. 2). Above is the graphical representation of the options available at each instant of time.

In the option graph above, we have shown how the (state of the) world changes as time progresses. Time moves discretely by one unit at a time, starting from t=0. At t=0, the world is at state s0, and there are three options:

a) The first option has an action that the agent may execute intentionally. In this case, the agent is responsible for making the world transit from state $s_i$ to state $s_j$.
b) The second option has an external event that can spontaneously occur which will affect the state of the world making it transit from $s_i$ to $s_j$.

c) The third option is a null action which represents the situation where if the agent chooses to avoid executing any action time t=0, then the world does not change its state at time t=1.

Apart from the states enumerated in the option diagram, there exist other states that are known as error states which we have not shown. The world may enter into the error states when actions from $\Gamma$ are not successfully completed, when agents perform actions that are not in $\Gamma$, or if the agent performs an action from $\Gamma$ when an event from $\Lambda$ has already started occurring. In this paper, we assume that the agent does not choose to perform an action $\beta \in \Gamma$ action if there occurs an event $\lambda \in \Lambda$ that can take the world to a next state valid state $s \in S$. Thus, in our model, an external event gets a higher priority to affect the world than the agent actions.

*Option Graph for multiple agents*

Fig. 2 also shows an option graph for three agents $A_1$, $A_2$, and $A_3$ where each agent $A_i$ has its own option graph. The states in the option graph of $A_1$, for example, are defined by the resources owned exclusively by $A_1$ and the resources it shares with the other agents $A_2$, and $A_3$. Thus, the three agents can all work concurrently as long as they do not operate on the shared resource. Operation on shared resources require specific policies for accessing and using them, but to keep our discussion simple, we will assume that the underlying action execution mechanism permits only one agent to operate on any resource at any time. A consequence of this assumption is that when an agent $A_i$ changes the state of a shared resource, it may affect the *next* states of other agents that are currently operating on non-shared resources. (We will elaborate on this later in the next section.)

### 3.3 Policy Graph

A policy graph is a sub-graph of an option graph. The option graph in Fig. 2 defines a policy (let us call it) P. The policy defines the permitted options at any given state for an agent according to the policy. The options shown as dashed or dotted arrow at each state signifies the fact that these options are permitted by the policy. The options shown as a black arrow are not permitted by the policy and yet may be possible to execute by an agent. An agent can decide to choose any available option depending on what next state it wants. A policy obeying agent is the one that always chooses an option that is permitted by the policy (dashed line). If an agent chooses an option that is not permitted by the policy, then the agent is said to have violated the policy. (Dotted lines show options permitted by the policy, but not selected by the agent.)

*Obeying policy and shared resources*

When an agent operates on a shared resource, in can in general affect the next states of other agents. A standard solution to this problem is to permit only agent $A_i$ at a time to "lock" the resource and start using it, while other agents wait for their turn. In a policy based model, we abstract out the details of how a shared resource is used,

and merely state that at any state the policy defines the future states that are known to all agents in the community, and when an agent selects an option, it also inherits the consequences of selecting that option. However, when an option not sanctioned by the policy is selected, it may affect the other agents' current state thus affecting their next options.

Fig. 2 shows the traces of three agents moving from time t = 0 towards their future goal states (goals states are not shown in the figure). When an agent executes an action, there are two cases that are considered. Let agent $A_1$ select and execute an action β1 and agent $A_2$ select and execute an action $\beta_2$. (The actions may or may not have been permitted by the policy.)

Case 1: The execution of the action $\beta_1$ by agent $A_1$ will not affect the execution of the action $\beta_2$ by the agent $A_2$. In this scenario, both the agents can execute their selected actions without interfering with each other.

Case 2: The execution of the action $\beta_1$ by agent $A_1$ will affect the execution of the action $\beta_2$ by the agent $A_2$. In this scenario (as we explained earlier), if the agents $A_1$ and $A_2$ both follow the policy (that is, chose the options marked dashed), then the execution of the actions $\beta_1$ and $\beta_2$ is said not have interfered with each other (according to the way the policy is defined). However, if the agents chose to violate the policy then noninterference may not be guaranteed.

### 3.4 Policy Semantics

We can formalize the notion of policy using the policy graph. As an example, consider Rule 1 above. Let us define the world state as a 4-tuple <o,c,d,b>, where o = currentOffer, c = reachargeAmount, and d = currentData, and b = currentBalance. Let the current state be $s_i$ where $s_i$= <"prepaidCap+",0, 0, 0>. Then upon executing action "*pay $30"*, the state $s_i$ changes to $s_{i+1}$ where $s_{i+1}$ = <"prepaidCap+", $30.00, 400MB,400MB>. Thus, in the option diagram, we insert the following transition at $s_i$: $s_i \xrightarrow{pay\ \$30} s_{i+1}$, where $s_i$ and $s_{i+1}$ are defined as above.

*Complex Policies*

Policies can be simple such as the ones shown above, or more complex as shown below:

**Rule**

Let A be the application agent that manages the email account and let the policies it needs to follow are as shown below:

a. Notify the service provider if the Email/SMS Bill email address changes;
b. Contact the service provider if the Email/SMS Bill has not been received; and
c. Keep the email/sms account secure to protect the privacy of your credit information contained in the Email/SMS Bill.

Translating such complex policies poses challenges in the sense that we need to model the underlying domain adequately. The policy rule (a) above can be depicted by a policy (sub) graph with the following characteristics:

Every node of the policy graph has a $\Lambda$ type option in addition to any other options. The event on the $\Lambda$ type option is "address changed". If the previous transition was due to the "address changed" event, then there is a next $\Gamma$ type option with the label "notify (new-address)".

An agent that obeys the above policy should perform the following behavior: If the previous transition was due to the $\Lambda$ type event "address changed", then the agent chooses the $\Gamma$ type option action "notify (new-address)". The policy semantics for the rule (b) can similarly be given as follows: Each node in the policy graph will have at least one $\Gamma$ type option included: "contact service provider". Let the current node be $s_i$. If none of the last k transitions were triggered by the occurrence of the $\Lambda$ type event "arrival of Email/SMS bill", then the agent chooses the $\Gamma$ type option "contact service provider", and executes the action.

The policy graph for (c) requires that as soon as the agent receives an email/sms account, then the agent will choose the option of performing an operation on the confidential account details (such an encrypting them, etc.), and all the future states will contain the encrypted form of the confidential account details. Further, each future state may have a $\Gamma$ or $\Lambda$ type action that may for example decrypt the account details thus threatening the confidential nature of the account details.

## 4 Policy Violation

Policy violation occurs when agents choose options not sanctioned by policies. It may be permitted in a world that is inhabited by a single agent. However, in a world where multiple agents coexist, policy violations result in unpredictable future states of the world. In certain situations, however, it may become necessary for an agent to violate the policy in order to maximize its chances of achieving its goals. In such cases, policy violations (that is, deviation from policy based behavior) must be performed carefully and managed. Deviation Manager (DM) in Fig. 1 above monitors such violations and decides whether at any time policy violation may be permitted.

*Dependent agents and Degree of violation*

In a generalized scenario, each agent depends on other agents to achieve its goals, the degree of dependency varying according to agents and situations. While there are no simple ways to predict the consequences of a violation, an estimate of the consequences is still necessary to manage the deviations which we discuss in the following section.

## 5 Measuring Policy Performance

To measure the performance of a policy, we consider a community of agents and consider its option graph. The edges in the option graph have a numerical weight

which indicates the score an agent gets if it chooses that option in its next move. Thus, the total score up to the current moment t is the sum of all the weights $w_i$ of the options that were chosen by the agent so far; that is,

$$score\ s = \sum_{i=0}^{t} w_i \ .$$

In the simulation below, we consider three agents, and the state consisting of three resources: $<r_1,r_2,r_3>$. A policy is implemented by choosing a DAG that is a sub-graph of the option graph restricting the number of options at any state in the option graph to not more than some maximum value (5 in our simulation). The maximum number of steps along the time axis is limited to $t_{max}$ where $t_{max} = 50$.

Table 1 shows our initial measurement with the points scored by each agent for each policy P1, $P_2$ and $P_3$. Typically, an agent will continue to choose the next highest score option until it reaches the state where the number of options is 0; that is, from this state the agent cannot proceed further as no more future states exist. As the agent chooses an option and executes the corresponding action, resources are consumed in the action execution, and thus the resources to go to another state.

**Table 1.** Points scored by each agent

| **Policy** | **Agent's score** | |
|---|---|---|
| | All resources are shared | No resources are shared |
| $P_1$ | 519 | 1151 |
| $P_2$ | 779 | 1443 |
| $P_3$ | 280 | 479 |

When no resources are shared, agents do not worry about the policies, and they choose any option that gives them the highest score. However, when resources are shared, agents need to follow the policy if they do not want to affect each other's behaviors. Thus, we see that in Table 1, the scores for individual agents are highest when they ignore other agents, but they are lower when the agents are required to follow the policies (in this example). While following a policy, each agent can only choose the option with the highest score from the options permitted by the policy. Incidentally, among the three policies we used, policy P2 appears to be a better policy for this small community since it helps the community achieve the highest score.

Fig. 3 shows the agent community behavior that is, the average score of each agent over the period of time when each agent follows the policy P1.

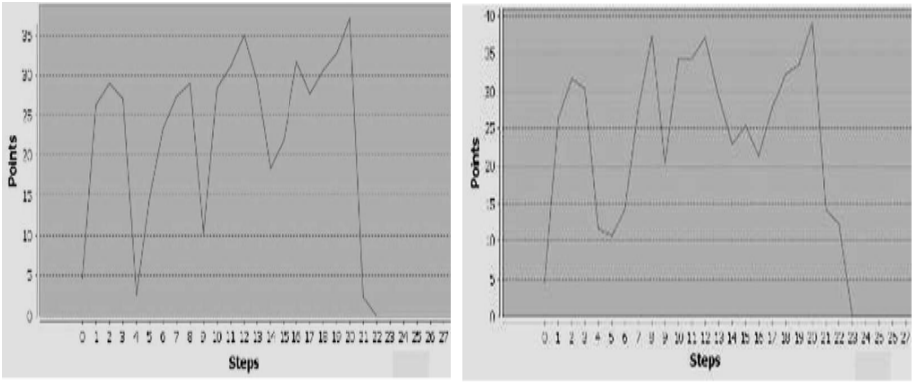



**Fig. 3.** (a) and (b): Community behavior when all agents follow policy(a) and violate policy(b)

Fig. 3 also shows the community behavior when agents violate the policy. From above figure, we see that when the agents violate the policy, the overall community score is different from the score where the agents do not violate the policy. Fig. 4 shows the difference in the behavior when agents follow and agents violate the policy.

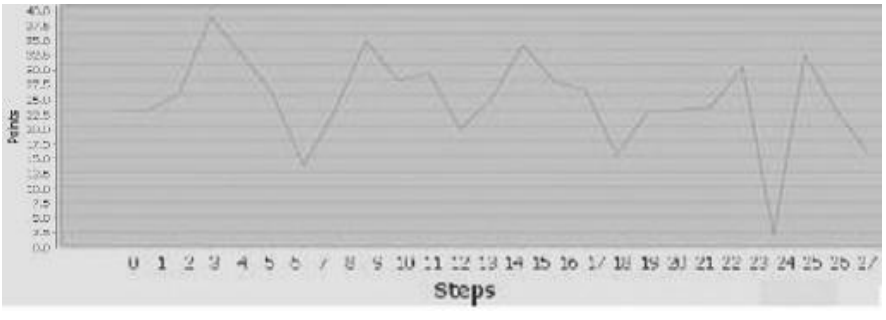
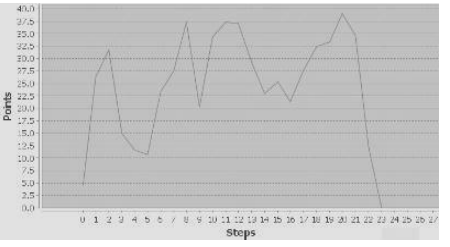



**Fig. 4.** Effect of Policy violation on agent community

The difference between these two graphs plays a vital role in deciding whether to permit a policy violation or not. Based on this, we propose a management policy for the monitoring agent (Fig. 1) since all requests from the application agents to use any available resource go through the monitoring agent. When a request from any agent to violate the policy comes to the MA, the MA computes the consequence (score difference) of the policy violation at the current state from Fig. 4. If the score difference is more than the specified threshold value, then MA rejects the request for policy violation, else the MA approves the violation.

The threshold value is an important parameter to decide in policy violations. This parameter is configurable and can be set by either the mobile phone user or service provider in their policy. If a value of the threshold is high, the MA is said to be

lenient. On the other hand, if the value of the threshold is too low, the MA is said to be strict. Following is an example of a policy violation management rule:




**Rule:** ***if*** *(current request = "Policy violation")* ***then***
***if*** *(get_actual_effect(current step) > Threshold)* ***then***
*do ("reject a request for policy violation");* ***else*** *do ( "permit the request for policy violation");*

**Fig. 5.** Community behavior when threshold value is 15

Fig. 5 shows the behavior of the community when we consider the management policy rule above to manage policy violation and Table 2 shows the performance of the agent community for a different threshold value. From the table below, we observe that policy violation is not always a "bad thing" after all. Sometimes, the community seems to perform better when agents are permitted to violate policies. This indicates that the current policy is not a "good" policy.

**Table 2.** Points gained for different threshold values

| Threshold value | Agent community points |
|---|---|
| 0 | 529 |
| 2 | 535 |
| 5 | 541 |
| 10 | 561 |
| 15 | 583 |

## 6 Related Work

Policies can be expressed more formally in languages such as Ponder2 [4] and Rei. Twidle et al [7] have proposed a new approach to monitor the behavior of a dynamic system. The normal event-condition-action (ECA) rules are not able to monitor the behavior of the system. These rules direct a system to behave according to a given situation.

Al Sum et al [6] have proposed a framework for dynamic policy based management of the resources in a mobile device. Their proposed framework manages sensitive resources such as network resources which can cost money or system resource such as mobile phone battery which can affect phone performance. A policy based solution in an educational application has been proposed by D Goel et al in [1]. They propose a solution to access various resources such as class rooms using a set of policy rules. Information such as current location, time and role of the user

are used to grant access to use any class room. People use online social networking websites such as Facebook, Twitter, etc. to keep in touch with their family and friends. A number of solutions have been proposed to manage the information using policies [2] [3].

In pervasive computing scenarios, it is important for the administrator to monitor policy deviation. In [7] example of health care system, the authors have mentioned about deviation manager which keeps monitoring the request against policy. It grants the access to treat any patient based on policy rules and role of the user who is requesting the access. Ahmed AlSum et al [6] suggest a dynamic policy approach for monitoring all resources in a mobile device. Samir Al-Khayatt et al [5] propose a solution for detecting policy violation. As per their suggested approach, they use an automated tool to monitor the internet usage by all employees in the office which detects violations whenever they occur.

## 7 Conclusion

We have in this paper given option graph based semantics for policies, and sketched out a scheme for measuring the performance of policies in a multi agent framework where the application programs on a mobile device are viewed as resource hungry autonomous agents. Using this formulation, we have shown how two policies can be compared in terms of their performance metrics. We also have discussed the significance of policy violations, and shown how the consequences of violations can be quantified. Based on this violation metric, we argued that it is not always bad to violate policies, and we proposed how policy management violation rules can be written where a resource monitoring agent MA can decide at any time whether to allow an application program to violate a policy.

## References

[1] Goel, D., Kher, E., Joag, S., Mujumdar, V., Griss, M., Dey, A.K.: Context-Aware Authentication Framework. In: First International ICST Conference on Mobile Computing, Applications, and Services. Springer Pub. Co., San Diego (2010)

[2] Shehab, M., Cheek, G., Touati, H., Squicciarini, A.C., Pau-Chen, C.: User Centric Policy Management in Online Social Networks. In: 2010 IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY), pp. 9–13 (2010)

[3] Kodeswaran, P., Viegas, E.: A Policy Based Infrastructure for Social Data Access with Privacy Guarantees. In: 2010 IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY), pp. 14–17 (2010)

[4] Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The Ponder Policy Specification Language. In: Sloman, M., Lobo, J., Lupu, E.C. (eds.) POLICY 2001. LNCS, vol. 1995, pp. 18–38. Springer, Heidelberg (2001)

[5] Al-Khayatt, S., Neale, R.: Automated detection of Internet usage policy violation. In: ACS/IEEE International Conference on Computer Systems and Applications, pp. 507–510 (2001)

[6] AlSum, A., Abdel-Hamid, A., Abdel-Aziem, M.: Application-specific dynamic policy rules (ASDPR) for J2ME. In: IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2009, pp. 512–516 (2009)
[7] Twidle, K., Marinovic, S., Dulay, N.: Teleo-Reactive Policies in Ponder2. In: 2010 IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY), pp. 57–60 (2010)
[8] Finnis, J., Saigal, N., Iamnitchi, A., Ligatti, J.: A location-based policy-specification language for mobile devices. Pervasive and Mobile Computing (2010)
[9] Talaei-Khoei, A., Bleistein, S., Ray, P., Parameswaran, N.: P-CARD: Policy-Based Contextual Awareness Realization for Disasters. In: 2010 43rd Hawaii International Conference on System Sciences (HICSS), pp. 1–10 (2011)
[10] Wang, F., Turner, K.J.: Towards personalised home care systems. In: Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments, pp. 1–7. ACM, Athens (2008)
[11] Talaei-Khoei, A., Bleistein, S., Ray, P., Parameswaran, N.: P-CARD: Policy-Based Contextual Awareness Realization for Disasters. In: 2010 43rd Hawaii International Conference on System Sciences (HICSS), p. 110 (2010)