PoDMan: Policy Deviation Management

Aishwarya Bakshi
University of New South Wales

Amir Talaei-Khoei
University of Nevada, Reno
University of Technology Sydney
atalaeikhoei@unr.edu

Pradeep Ray
University of Michigan–Shanghai Jiao Tong University Joint Institute,
Shanghai Jiao Tong University, China

Terje Solvoll
Norwegian Centre for Integrated Care and Telemedicine
University Hospital of North Norway

Abstract

Whenever an unexpected or exceptional situation occurs, complying with the existing policies may not be possible. The main objective of this work is to assist individuals and organizations to decide in the process of deviating from policies and performing a non-complying action. The paper proposes utilizing software agents as supportive tools to provide the best non-complying action while deviating from policies. The article also introduces a process in which the decision on the choice of non-complying action can be made. The work is motivated by a real scenario observed in a hospital in Norway and demonstrated through the same settings.

Keywords: Agent; Policy; Deviation

1 Introduction

While policy based systems have largely been successful in controlling and managing cooperative work environments, one of its weaknesses lies in the fact that there is always the risk of exceptions, where following the policy rule might result in non-optimal outcomes and at worst lead to completely devastating consequences. Many studies have proposed the use of adaptive policies to counter the changing environmental circumstances, thereby reducing the risk of exception. As it is humanly not possible to envision all possible exceptions, there is a need to introduce procedures whereby the user is given the decision making support and flexibility to deviate from a policy when absolutely needed. While many decision support systems have been suggested for CSCW systems (Chau 2003, Benbasat 1990,Shum 2011), this paper aims to introduce a method which allows greater flexibility to CSCW systems by assisting in the decision making process of deviating from policies, making sure that the deviations take place in a controlled manner when such exceptions arise. In order to address the problem, the study proposes a methodology called PoDMan (Policy deviation management) which uses software agents to detect policy violations and if required, assist users in the decision making process of deviating from the policy in a controlled manner.

The rest of this paper is organized in the following way: Section 2 illustrates the motivation behind this work by providing a real scenario observed in the wireless communication system in a hospital in Norway. Section 3 presents background and defines different concepts being used in this research. Section 4 proposes the PoDMan methodology that consists of a framework of definitions and semantics as well as a stepwise process for policy deviation deployed in the framework. Section 5 demonstrates PoDMan through the wireless communication scenario introduced in Section 2. Section 6 presents a simulation for evaluating PoDMAn. Section 7 concludes the study and discusses the implications and limitations of the work.

2 Illustrating the motivation: Wireless communication system at a hospital

In this section, we illustrate the motivation behind this study using an example that authors have discussed (Talaei-Khoei 2011) where they were proposing a policy based communication system at St. Olavs Hospital in Trondheim, Norway.

St. Olavs Hospital has implemented a single, converged IP network for all communications among the hospital staff. All medical personnel in the hospital communicate via IP phones and IP enabled devices where they have to log in at the start of the shift and can be reached by other personnel by a press of a button on their device. All communication amongst the personnel is routed via a communications center which continuously monitors the presence of users in hospital and their activity statuses such as busy, available, etc.

The physicians at the hospital often complained that they were frequently distracted by calls from nurses while they were in the middle of a medical procedure which required them to stop what they were doing, remove their gloves, answer the call, wash their hands again and put on a new set of gloves. This reduced the overall efficiency with which they work. In order to prevent such distractions, the hospital laid down policy rules which forbid the nurses from calling physicians if they were unavailable or busy and instead recommended the nurses to get in contact with an alternate physician.

While this works well under normal circumstances, it does not consider the case where the presence of a particular physician is critical to a patient's health e.g. consider the case where only one specialist physician is present at the hospital but he is busy performing a non-critical procedure on a patient. At the same time another patient suddenly requires critical attention from the specialist. When the nurse tries to contact the specialist physician for advice, the system forbids the nurse from doing so as the doctor is busy with another procedure although it might not critical. In such a case, the provision to deviate from the policy and calling the doctor might have saved the life of the patient.

Hence, through this illustration, we have identified the following issues that have motivated us for this study:

- Users of collaborative policy based systems are often unaware of existing policies or changes made to existing policies
- Policies can often be too rigid and might lead to undesirable outcomes under exceptional circumstances
- There is a need for controlled management of non-compliant actions when exceptional situations arise.

However, the aforementioned scenario was relatively simple. The objective, here, was to illustrate the problem that motivates us for PoDMan. The research in software agent has been confirmed, in different cases, that use of agents in order to address the sporadic contacts in distributed communications is beneficial (Ulieru and Worthington, 2006, Ray et al., 2005a and Ray et al., 2005b). Therefore, here, we employ software agents to assist individuals managing the sporadic communication. We also propose a methodology to develop agent systems that assist individuals to deviate from policies.

3 Background

In this section, we provide the background to the study presented in this article. We set definitions and foundation on which the work has been grounded.

3.1 Policy adaptation and policy deviation

Not surprisingly, in cooperative environments, complying with policies is required. However, these policies are built keeping the best possible future in mind. Problems often arise when these policies are too rigid and designed without considering all possible situations, including

exceptional situations and future uncertainties, called unexpected situations. When a cooperative role runs into unexpected or exceptional situation, any behavior that the role may perform may violate the policy constraints, and the behavior may have to be aborted. However, abortion may not be the best strategy (Lewis 2010). Another option is policy adaptation that provides two alternatives; (a) modifying the policy (Dearle 2010) and (b) dealing with the consequences of performing a non-compliant action (Migon 2010). These two alternatives result in two different definitions of policy adaptation.

- Adaptation by changing values of policy parameters depending on performance feedback (Lotlikar 2006).
- Adaptation by relaxation (Lewis 2010) which means monitoring consequences of non-conformance behavior against the desired behavior proposed by the policy to minimum incompliancy; that is policy deviation.

Verhagen (2001) describes policies from two perspectives, namely – policies from the social theory perspective and policies from legal theory perspective. Social theory (Verhagen, 2001) states that a cooperative role complies with policies because it is rational to do so and because that is what others expect from the role. Therefore, policies in social theory can be described as a generalized expectation of behaviors. Policies from this point of view are a set of social rules that are designed by the community of cooperative roles and can be changed on the go. Therefore the first definition of policy adaptation can be applied to social policies. On the other hand, legal theory looks at policy as a sense of duty (Verhagen, 2001) and that a cooperative role complies with a policy because of the authority issuing the policy. Policy in this perspective is a set of constraints that limits behaviors. Since these policies are designed by an authority, they cannot be modified on the go. Therefore, instead of modifying the policies, when exceptional or unexpected situations occur, one has to decide what action must be performed to have minimum incompliancy from the specified legal policy. Therefore, the definition of policy deviation can be applied. The comparison between two definitions of policy adaptation is presented in Table 1.

Definition	Configurations	Outcome	Type of policy
Adaptation by changing values of policy parameters	Performance feedback	Modified policy	Social policies
Policy deviation: Adaptation by relaxation	Comparison of consequences of non-conformance behavior against the desired behavior	Minimum incompliancy	Legal policies

Table 1: Policy adaptation

3.2 Policy adaptation framework: A systematic review on IT-centric policy adaptation in run time

Independent on what type of policy adaptation is concerned, Swanson et.al (Swanson 2009) describe the adaptive policy cycle in three stages; (a) policy set-up, (b) adjustment, and (c) monitoring that involves continuously monitoring and evaluating the performance of the policy for further setup and adjustment.

The main objective of the policy set-up phase is to determine the policy goals, constraints and performance factors in order to understand the policy vulnerabilities and in turn develop triggers, signposts, mitigating actions and contingency plans. Using these outcomes from the policy setup phase, policy makers depending on the policy being social or legal choose to adjust the policy based on the findings of the previous step or adjust the behavior. Once the policies are put in practice, their performance need to be continuously monitored and reviews must be carried out to learn from the outcomes of these policies to further improve them. Depending on the outcomes of these reviews, another cycle of policy adaptation starting with policy setup will be conducted.

Having taken the Swanson's model into account, authors aimed to autotomize the adaptation cycle. To that end, they conducted a systematic review papers proposing IT-centric CSCW methods that can be used in each of the Swanson's model. The total number of paper reviewed was 1418. The result of the review was a framework for policy adaptation which provides detailed steps that needs to be taken for automated policy adaptation in run-time. For each step the methods that can be used are introduced. For more detailed information, please refer to (Bakshi et.al 2013).

It was found, in the systematic review, that IT-centric CSCW methods proposed to address the adjustment step in Swanson's policy adaptation model, have been heavily focused on Lotikar's (Lotlikar 2001) perspective i.e. adaptation by changing values of policy parameters. While the literature has largely ignored that the Lotikar's perspective can only be applied in social policies, adaptation for legal policies requires further investigation. Although Lewis et al (2010) have presented an idea to solve this problem for policy deviations from a legal policies; the work lacks a sufficient level of details and implementation.

Having said that, this paper focuses on a definitive method on how IT-centric CSCW systems assist individuals to deviate from legal policies in an organization and perform non-conformance behavior while an unexpected or exceptional situation concurs.

4 Policy Deviation Management (PoDMan)

In this section, we propose policy deviation management (PoDMan), which aims to assist policy based collaborative systems in detecting potential policy violations in future and in return present alternative actions to the users to achieve the desired outcome whenever possible. In order to so, we shall first define a framework, including semantics that set a foundation of the different definitions under which PoDMan is grounded.

4.1 PoDMan Framework

4.1.1 Use of software agents to detect and assist for policy deviations

The use of software agents has proved useful in many CSCW application areas such as emergency management (Yuan and Detlor, 2005), risk management (Ulieru and Worthington, 2005), mobile health monitoring (Ray et al., 2008), disaster management (Ray and Chattopadhyay, 2009), and wireless communication in hospitals (Talaei-Khoei 2011) .CSCW tends to agree that the following characteristics of software agents make them useful in assisting individuals (Woolridge and Wooldridge, 2001): Autonomy, an agent can operate without the intervention of individuals; Social Ability, an agent is able to interact with other agents and individuals; Reactivity, computing power of agents makes it possible for them to react on the changes of the environment in a timely fashion (however, this is heavily dependent on the computing power); Pro-Activity, an agent tries to achieve the purpose for which it was initiated, which can be highly useful to assist individuals employing agents for particular purpose. Therefore, PoDMan has chosen software agents to present individual roles.

The software agent acts as the mediator between an individual or organizational user and the system in which PoDMan has been implemented. The agent communicates all requests on behalf of the user and delivers the responses from the application back to the user. It is also the responsibility of the agent to determine whether the requested user actions are compliant with the policies which govern the system. Our system is composed of multiple such agents, one for every role. If required, these agents can communicate with other agents in the collaborative system in order to gather relevant information.

4.1.2 From decision trees to branching time models (BTM)

Research in software agents has been interested in the natural semantics for the possible scenarios in a given situation as a mental attitude of agents describing internal features that must be interpreted subjectively from the agents' point of view. For more about mental attitudes of agents, see (Rao and Georgeff, 1991; Boella and van der Torre, 2003). The classical approach is possible-world (or in short, world) in which a situation can be considered possible

in addition to true or false (Rao and Georgeff, 1991). This can be modeled using a tree-like structure in which each path of the tree represents one possible scenario and each node depicts one possible world. This single past and multiple futures structure is called branching time model (BTM), used in PoDMan; see Figure 3.

As we discussed above, in the branching-time model of worlds, there may be more than one choice available to execute i.e. multiple future. Here, we begin with the classical decision tree and show how we can generate a BTM utilized for our problem. Informally, a decision tree consists of decisions and chances. Decisions represent points where the agent has to choose one alternative from a number of choices available. Chances represent points where probability plays a dominant role and reflects alternatives over which the agent has no control. Decision trees also include probability functions that map chances to real-value probabilities and a pay-off function that maps decisions to a real number. We transform decision trees to an equivalent model (Rao and Georgeff, 1995) that represent decisions with optional actions and chances with events. This transformation provides an alternative basis for cases in we are not able to form pay-off or probability functions.

We begin by considering a decision tree, in which every possible path (scenario), including those with zero pay-off, is represented. We start from the root node of the tree and traverse each arc. For each unique arc emanating from a chance node, I create a new decision tree that is identical to a tuple made of (1) the chance node, called event and (2) the decision node located before the chance node. The difference between the generated tree and the original decision tree is that (I) the chance node is removed and (II) the arc coming to the chance node is connected to the arc emanating from the chance node. This process is carried out recursively until there is no chance node left. This will result in a set of trees with no chance node, while each corresponds to a different possible state of the environment; that is, from the traditional possible worlds perspective. Indeed, each of these trees represents a possible world triggered by an event; see Figure 3.

Any changes to the environment or the system are known as events. These events can trigger the agents to choose an action, modeled by arcs in BTMs (see Figure 3) and transform from one situation to another by performing a specified action, although it is not mandatory that they do so. While one agent can react to a certain event in the environment, another agent can be completely apathetic towards it. Therefore the agents treat each event differently depending on the cognitive state they are in. In PoDMan, this cognition is presented by a set of BTMs programmed into the agent. If the agent has a BTM trigged by an event, then the agent would react agents the event according to the BTM; See Figure 3.

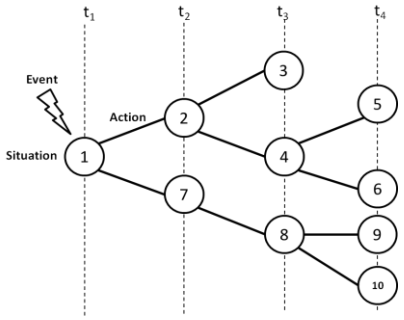


Figure 1: An example of branching time model for an agent.

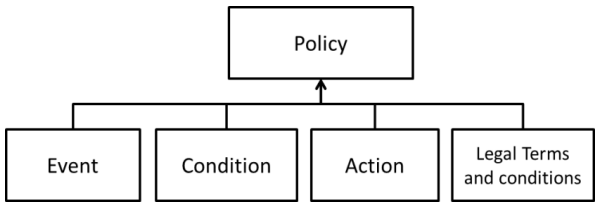


Figure 2: PoDMan Policy structure

As an example, at t1 the possible paths that the agent can follow are: (a) 1-2-3, (b) 1-2-4-5, (c) 1-2-4-6, (d) 1-7-8-9 and (e) 1-7-8-10. In this example, when an event occurs at t1, the agent can transform situation 1 to either situation 2 or 7 by performing the specified action. On completion of the action, the agent marks the previous situation as '*done*' and the new situation is marked as current. This process continues till the agent reaches the final situation in the path, signifying the end of a possible scenario.

4.1.3 Policy structure

The policy structure for PoDMan is borrowed from Directory Enabled Networks – next generation (DEN-ng) (Strassner et al., 2009), which provides a meta-model for the implementing business policies (Strassner et al., 2008). According to the DEN-ng model, policies are defined as a composition of policy rules where each policy rule is a proposition with event-condition-action (ECA) semantics. These rules are triggered by the events described earlier. Based on the event and condition specified, an action may or may not be allowed. Along with ECA, PoDMan also requires the inclusion of another attribute to the policy specifications; that is called Legal Terms and Conditions. The legal conditions serve as guidelines for users in the decision making process of policy deviation. An example of such a guideline would be, "*The following information is strictly confidential and sharing of this information with any unauthorized party may result in prosecution*". Figure 4 illustrates the basic policy structure used in PoDMan. In following, we may say policy in short while we refer to policy rule.

Sloman (1994) defines modality of actions by introducing four types of policy rules: permitting, forbidding, requiring and deterring. Table 2 shows examples for each type of policy rules.

Type	Example	Code	Presentation of an action govern by these rules in BTM
Permitting	When an employee leaves a company, if he or she has been working in the customer service unit, it is permitted to remove his or her file from the archive.	F	**F**
Forbidding	When an employee leaves a company, if he or she has ever worked in the accounting unit, it is forbidden to remove his or her file from the archive.	D	**D**
Requiring	When an employee leaves a company, if he or she has ever worked in the accounting unit, it is required that his or her file be kept in the archive.	P	**P**
Deterring	When an employee leaves a company, if he or she has been working in customer service unit, it is deterred to keep his or her file in the archive.	R	**R**

Table 2: Examples of policy modality types

Twidle et al. (2009) and Moffett & Sloman (1991) state that amongst these, only forbidding and requiring modalities are usually in force. While both deterring and permitting policies recommend certain behaviors to the user, they do not enforce a user to abide by the rules i.e. the users can choose whether they want to follow the policy or not. We call these types of policies Weak Polices. On the other hand, forbidding and requiring policies do not offer any options to the users and they are forced to follow the rules irrespective of the situation. We call these types of policies Strong Policies.

While strong policies are useful for operational control under known and expected conditions, problems due to the inflexibility of such policies might arise when unexpected or exceptional situations come to the fore and a policy prevents the user from performing a critical action. In such cases, some of the policies might need to be demoted to a weak modality, e.g. forbidden to Deterring or Requiring to Permitting to temporarily allow the user to choose a non-

compliant action under the circumstances. Of course there are some policies which need to remain strong irrespective of the situation e.g. for security and auditing reasons, under no circumstances can a user perform an action without signing into the system. For these types of policies, the decision to deviate from these policies does not lie with the user but with a higher authority.

4.1.4 BTM governed by policy rules

Actions in the BTM are governed by policy rules and even though the same action might appear at different situations in the BTM, the modalities can be different or stay the same depending on the situation the action emerges from; See Table 2. In the example of wireless communication in the hospital, the action of a calling a doctor by a nurse might be prohibited if the doctor is in the operating theatre, but the same action of calling the doctor might be allowed if the doctor is performing clinic duties.

The path which does not contain any action governed by a strong policy modality is called free path. Other than free paths, whenever an agent comes across a strong policy, it utilizes the presence of multiple similar actions across the BTM to determine whether the strong policies are conditional or absolute. This is done by checking if at least one of the instances of the action in the BTM is governed by a weak policy, i.e. a permitting or deterring policy, and if such an action exists at any point in the BTM, it is considered to be a conditional-strong policy. If no similar actions are found or none of the other similar actions are governed by weak policies, the action is treated as an absolute-strong policy by the agent. Therefore, in our illustrative example, since the nurse is permitted to call the doctor in at least one situation, i.e. when the doctor is performing clinic duties, the forbidding action of calling the doctor is considered to be a conditional-strong policy by the agent. This implies that if the nurse considers the circumstances critical enough, and understands the implications of the decision, she is allowed to deviate from the policy and call the doctor even if he is in the operating theatre. It must be ensured that all the actions leading to the alternate action must be followed.

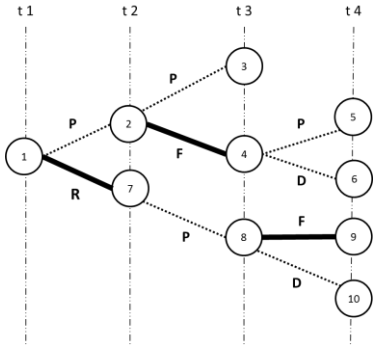


Figure 3: An example of branching time model governed by policy rules

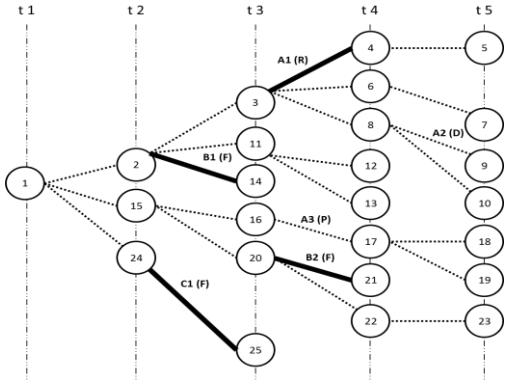


Figure 4: An example to describe the PoDMan process: BTM governed by policy rules

In this BTM presented in Figure 3, we noted that there are 5 complete paths to choose from at t1 [1-2-3], [1-2-4-5], [1-2-4-6], [1-7-8-9] and [1-7-8-10]. But after the policy rules are placed, we see that there are paths which contain actions forbidding the agent from completing the path and reaching the final situation: [2-4] and [8-9]. We also note that there is only one path which does not contain any action governed by a strong policy modality [1-2-3].

We can also observe that in this BTM there are three situations from which policy deviations might be required Situations 1, 2 and 8. This is because at least one of the actions emerging from these situations is governed by a strong policy.

4.1.5 Conditions in which policy deviation is required.

Policy deviation comes into the picture, only when a user wants to perform a forbidding action or not perform a requiring action, in other words only when a strong policy is encountered. When a user requests the agent to deviate from a strong policy, the agent calculates if the policy is an absolute-strong policy or a conditional-strong policy. It does so by trying to locate identical actions in the BTM that are governed by a weak policy modality. If it finds at least one such action, it considers the policy to be of the conditional-strong type and allows the user to deviate from the policy once the legal terms and conditions have been signed. The agent also makes sure that the path and all the actions leading up to the alternate action are also performed by the user before proceeding. If more than one such action is found, the agent ranks these alternate paths for the user to choose from. A detailed explanation of calculating alternative paths is discussed in Section 4.2.1. If no satisfactory alternative path can be found, the agent considers the policy to be an absolute strong policy and PoDMan puts forward exception management procedures to temporarily permit the agent to perform the forbidden action. Both techniques will be discussed in detail in Section 4.2.

4.2 PoDMan process: steps towards policy deviation

Having introduced the PoDMan framework, we shall discuss the process in which policy deviations can be managed. We illustrate the process in a hypothetical example, in which we can describe and clarify the PoDMan steps towards policy deviation; see Figure 6.

Let us assume that in the BTM shown in Figure 6, action A1, A2 and A3 are identical and actions B1 and B2 are also identical. Table 3 maps the situations and modality strength of actions. Using our description of conditional and absolute strong policies, we can see that Action A belongs to the conditional strong policy group as it has at least one instance where it is governed by weak policies (A2 and A3). On the other hand, both the instances of Action B (B1 and B2) are governed by strong policies, and therefore belong to the absolute-strong policy group.

Action		From situation	To Situation	Modality strength	Modality type	Modality sub-type
A	A1	3	4	Strong	Requiring	Conditional
	A2	8	9	Weak	Deterring	
	A3	16	17	Weak	Permitting	
B	B1	2	14	Strong	Forbidding	Absolute
	B2	20	21	Strong	Forbidding	
C	C1	24	25	Strong	Forbidding	Absolute

Table 3: An example to describe the PoDMan process: situation mapping

In the beginning of the scenario depicted in Figure 6, at t1, the user agent performs a depth-first traversal of the entire BTM, allowing it to record every possible complete path ahead; in this case 12 of them. Figure 7 illustrates the 7 paths that contain only the above mentioned actions.

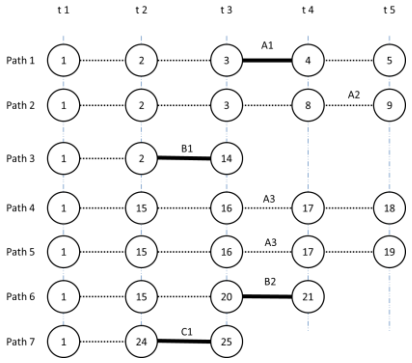


Figure 7: An example to describe the PoDMan process: possible paths from t1 contacting instances of action A, B, or C

Let us consider the following possible scenarios from Figure 7:

1. The user follows a path containing no strong policies
2. The user follows a path containing conditional-strong policies
3. The user follows a path containing absolute-strong policies

These present different possible settings that may happen. In following, we explain the PoDMan process in each setting.

I. Free path: a path containing no strong policies: From Figure 7, we can see that Path 2 is a free path containing no strong policies and therefore can be chosen for this scenario:

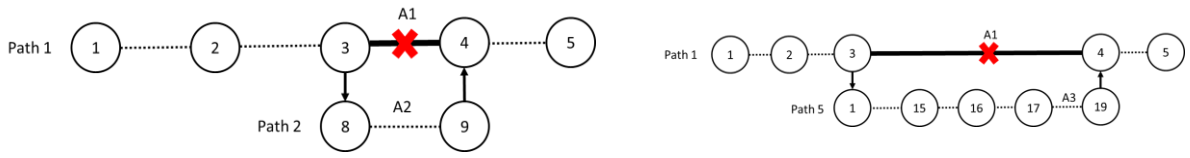
- At t1 the agent knows that amongst all the paths ahead, there is at least one free path i.e. a path that does not encounter any strong policies [Path 2]. When the user requests the agent to transform Situation 1 to 2, the agent proceeds to do so regardless of whether the user actually follows the policy of the action or not as the policy modality for the action is weak. It also marks Situation 1 as 'done'.
- At t2, the agent recalculates the possible future paths from Situation 2 and again finds that there is at least one path with no forbidden policies and thereby allows the user to transforms from Situation 2 to 3.The process is repeated till the last action is performed, transforming Situation 8 to 9 and the task is completed.

This example illustrates a scenario where no deviations are required as all the policies encountered are weak and the user can freely choose whether or not they want to comply with the policy rules in order to complete the task i.e. no deviation requests are required.

II. Path contains a conditional strong policy: A conditional strong policy is one which can be deviated from if required at the discretion of the users, as long as they accept the legal terms and conditions of doing so. In this scenario, we consider a path containing only conditional strong policies, which in our case is Path 1.

- In this path, the agent does not warn the user of any possible policy violation till t3 as it calculates the existence of at least one free path till that point, i.e. Path 2.
- At t3 if the user wants to deviate from the action that transforms Situation 3 to 4, the agent detects a strong policy violation and warns the user. If the user still wants to persist with the decision, the agent first calculates the existence of alternate paths if available. From figure 4-5, we observe that paths 2, 4 and 5 contain identical actions (A2 and A3) which are governed by weak policies and therefore determines the strong policy governing A1 to be a conditional strong policy. A conditional strong policy signifies that under exceptional circumstances, the agent can deviate from the policy as long as an identical action with weak policy modality exists in the BTM and the user understands and accepts the legal implications of performing the non-compliant action. The agent then builds alternative paths by replacing A1 with all the actions from the path containing A2 or A3, starting with

the last common situation and ending at the situation where A2 or A3 is performed. It must however be noted that the agent ignores an alternate path if it contains even one forbidden policy. Figure 8.a and Figure 8.b illustrate the alternative paths calculated by the agent.



a. Choosing alternative actions from path 2

b. Choosing alternative actions from path 5

Figure 8: An example to describe the PoDMan process: Alternative paths to deviate from conditional strong policies

Once the alternative paths are calculated, the agent ranks the choices for the user to choose from. The process of ranking alternate paths is discussed in Section 4.2.2.

Once the user chooses the preferred alternative, the agent proceeds to fetch the legal conditions associated with the policy and presents it to the user to sign and accept. The reasons for violation are logged for audit after which the agent is permitted to perform the requested action. Figure 9 shows the PoDMan methodology of handing conditional-strong policy deviations.

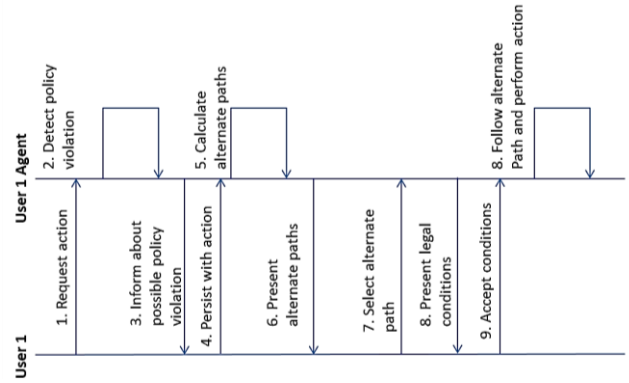


Figure 9: An example to describe the PoDMan process: sequence what need to be done by the agent in conditional-strong policy deviation

III. Path containing absolute-strong policies: Finally, in this scenario, we consider a path containing absolute-strong policies. This means that even under exceptional circumstances, the user does not have the sole decision making power of deviating from the policy without permission from an authority. In our system, we consider an authority to be any role that is permitted to perform the required action, i.e. a role which has, in its BTM, the permissions to perform an identical action. The reasoning behind this is that a role who is allowed to perform the action under normal circumstances has the ability to decide whether the action is required or not under other circumstances.

Path 3, 6 and 7 all contain at least one action with an absolute-strong policy and thus match this scenario. We will analyze Path 3 and Path 7 for this illustration.

- If the user decides to follow Path 3, as described in the previous scenarios, the agent allows the user's requests and transforms up till situation 2 without any warning as it detects at least one free path till that point.
- At t2 if the user chooses to follow Path 3 and requests to perform the forbidden action B1 connecting Situation 2 to 3, the agent detects a violation and immediately warns the user. The agent then performs a check to verify if the forbidden policies are absolute or

conditional. Since the other instance of action, B2, is also governed by a forbidden policy, the agent realizes that the policy is an absolute-strong policy.

- If the user wants to persist with the action, the agent broadcasts a message requesting any agent who's BTM contains an identical action that is governed by a weak policy for permission to deviate. Other agents in the system on receiving this message check to see if their policies permit them to perform the requested action and if it does, notifies its users of the request. Any user belonging to the role with necessary privileges can respond to the request. This process is illustrated in Figure 11. Once the request is accepted by a user with the higher privileges and the legal conditions are co-signed, another message is broadcast to notify the agents that the last request can be dropped. Only once both roles co-sign the terms and conditions is the agent allowed to deviate from the policy.
- If the user decides to follow Path 7, the moment the agent receives a request to transform situation 1 to 24, it detects that there are no free paths ahead and the path ahead contains a forbidding policy. Thus it premeditates the policy violation and warns the user ahead of time. If the user still chooses to go ahead, the process described earlier is followed.

Figure 11 illustrates the complete process for absolute-strong policy deviation.

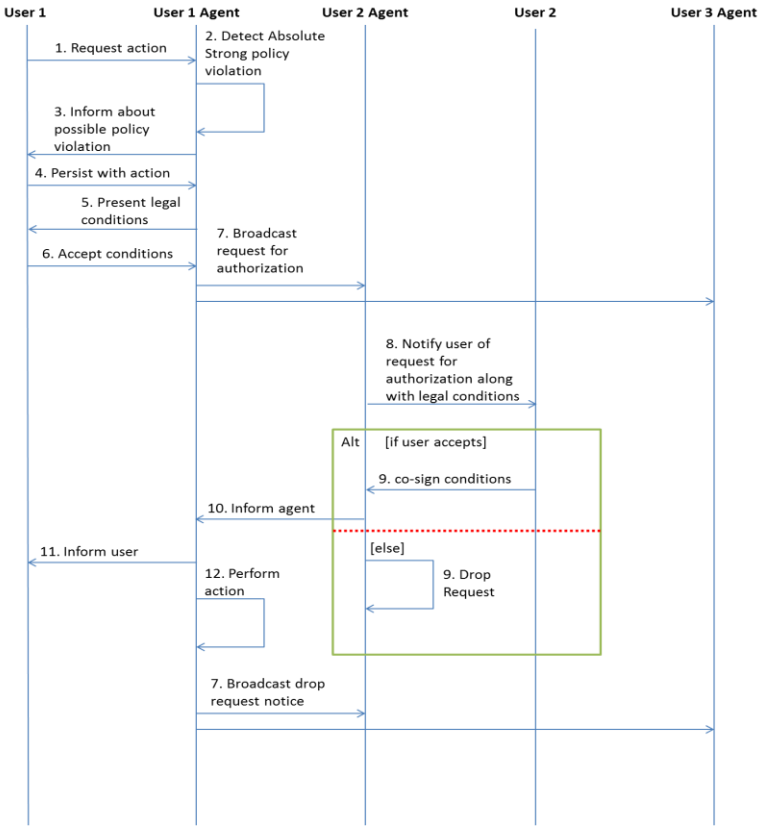


Figure 11: An example to describe the PoDMan process: what need to be done by the agent in absolute-strong forbidden policy deviation

4.2.1 Calculating the alternative paths

As discussed in the previous section, whenever an agent receives a request to deviate from a conditional strong policy, its first response should be to notify the user of the imminent violation followed by calculating and presenting the user with alternate options to continue its path till the goal. It is illustrated below how the agent calculates the alternate routes depending on the location of the alternate actions.

Consider the paths in Figure 12 to be the complete paths that contain available alternatives for the agent to choose from while deviating from a policy rule and the sub-paths marked in grey to be the actual steps that the agent must replace the original action with during the deviation process. These paths are used as examples to highlight how the agent ranks the alternative actions to assist the user in choosing a path.

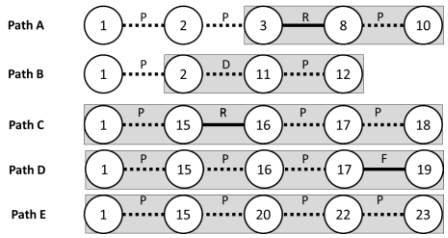


Figure 12: An example to describe the PoDMan process: alternate paths available to the agent (Shaded grey)

4.2.2 Ranking the alternative paths

Once the alternative paths are calculated by the agent, ranking these paths can assist the user in choosing the most convenient option. In our system, the agent ranks the alternative paths in two separate ways – by path length and path flexibility; see Table 4.

Path	Total number of situations	Total Forbidden Policies	Total Requiring Policies	Total deterring policies	Total permitting policies
A	3	0	1	0	1
B	3	0	0	1	2
C	5	0	1	0	3
D	5	1	0	0	3
E	5	0	0	0	4

Table 4: An example to describe the PoDMan process: total situations and policy distribution for each alternate path

I. Ranking by path length: Ranking by path length involves sorting the paths based on the number of situations encountered in each of the paths. This provides the quickest path to reach the required objective but is not necessarily the most flexible. If two paths are of the same length, they are ranked by their path flexibility. While it can be argued that since weak policies are optional and can be ignored from the total count, it must be noted that policies are in place to improve organizational processes. Policy rules should be ignored only if deemed absolutely necessary. Hence it is included in the total count. From Table 4, we can therefore rank the paths as follows

[Shorter] Path B < Path A < Path E < Path C [Longer]

II. Ranking by path flexibility: Ranking by path flexibility involves calculating the number of policies of each type encountered by each alternate path. Path flexibility can be said to be the amount of power the user or agent has to perform the task without being constrained by the policies. The path flexibility is ranked in the order of

- The least number requiring policies in the path
- The least number of deterring policies in the path
- The most number of permitted policies in the path

Paths ranked by flexibility provide the easiest route to the required objective but are not necessarily the shortest. If two paths are equally easy, they are sorted by path length. Based on Table 4, we can therefore rank the ease of path as follows.

[Flexible] Path E < Path B < Path A < Path C [Rigid]

5 Illustrating PoDMan: Wireless communication system at a hospital

Keeping the scenario explained in Section 2 in mind, let us consider a case where due to some unforeseen circumstances, only one neurologist is present at the hospital. As she prepares for a surgery, all of a sudden one of the other patients starts showing major neurological symptoms. At this point the nurse tries to call the attending neurologist to ask for advice but the policy forbids the call from going through, which leads to the patient suffering major nerve damage due to the lack of appropriate medical attention. A step-by-step illustration is presented of how this situation could be handled by PoDMan. The branching time model of the nurse agent is shown in Figure 13.

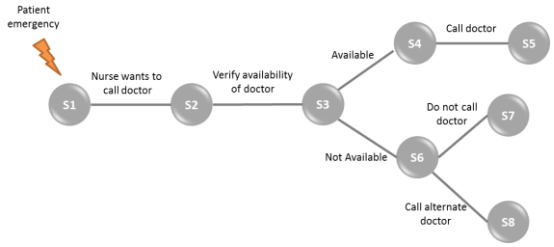


Figure 5: Illustrating PoDMan in wireless communication systems at a hospital: nurse agent's BTM

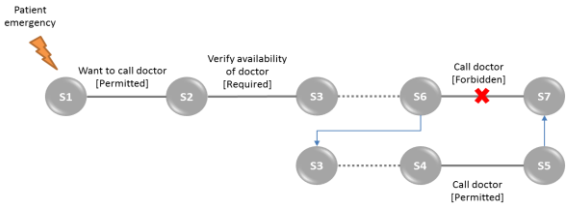


Figure 6: Illustrating PoDMan in wireless communication systems at a hospital: forbidden actions being replaced by alternative path

To begin, we have created transformation table for this scenario; see Table 5.

Transformation	Action	Modality
S1-S2	Nurse wants to call neurologist [doctor]	Permitted
S2-S3	Verify availability of neurologist	Required
S3-S4	Neurologist is available	-
S4-S5	Call doctor	Permitted
S3-S6	Neurologist not available	-
S6-S7	Call doctor	Forbidden
S6-S8	Call alternate doctor	Permitted

Table 5: Illustrating PoDMan in wireless communication systems at a hospital: transformation table

Initially, when the nurse decides to call the neurologist, the agent transforms S1 to S2. This is followed by a requiring policy to find out if the neurologist is available or not. On completing that action, the agent transforms S2 to S3. At S3, the agent is informed that the neurologist is unavailable and transforms to S6, from which the nurse is forbidden to call the doctor (S6-S7). At this point, the agent warns the nurse that calling the neurologist is against the policy rules and the nurse should call an alternate doctor. Considering the situation to be an emergency where policy deviation is required, the nurse requests the agent to persist with the action of calling the neurologist even though she is unavailable. On receiving the request for deviation, the agent first tries to determine whether the policy is a conditional strong policy or an absolute strong policy. It does this by backtracking one situation at a time and checking each emerging branch from that situation for a path containing the same action but governed by a weak policy.

Thus from S6, the agent backtracks to S3 and observes that the path S3-S4-S5 contains the action [call doctor] but governed by a permitting policy (weak). This implies that the forbidding policy at S6 is a conditional strong policy and therefore the decision to deviate from the policy lies with the nurse. The agent then proceeds to fetch the legal conditions of deviating from the policy and once the conditions are accepted by the nurse, replaces the action (S6-S7) with actions (S3-S4, S4-S5), thereby permitting the nurse to contact the doctor. This is illustrated in Figure 14.

Furthermore, the request for deviation is logged by the agent. This log can be used by the management in future to modify the agent's branching time model in a way that the exception is handled.

In case of absolute strong policy deviations, the request would be broadcast by the agent to all other agents with the necessary deviation rights. Simulation

In this section, we explain a simulation that was aimed to evaluate the effectiveness of the PoDMan. It is important to first establish definition of effectiveness. In the context of policy deviation, we have defined effectiveness as the likelihood of policy deviation being allowed under exceptional or unexpected circumstances. In Table 2, the various modalities of policy rules were introduced. It was determined that strong policies could be further classified into conditional-strong policies and absolute-strong policies. Policy deviations from conditional strong policies can be performed by a user based on his or her own judgment and therefore the likelihood of the policy deviation being allowed lies completely with the user i.e. if the user chooses to deviate from a conditional policy, it will be permitted by the system each time. Similarly, without PoDMan, the deviation will not be permitted by the system each time. Thus for conditional policies, the effectiveness of PoDMan is accordingly confirmed.

Policy deviations from absolute strong policies require the permissions from other users who have the necessary authorizations to perform the action under normal circumstances. Therefore the probability of being allowed to deviate from the policy depends on other users in the system. Thus a simulation is performed to determine the probability of a deviation from absolute-strong policies being permitted in a cooperative environment under different parameters.

5.1 Input and output parameters

When a user requests deviation from an absolute-strong policy, in order to calculate the probability of absolute-strong policy deviation being allowed, first the parameters that affect this value need to be determined. These are the input parameters for the simulation; see Table 6.

- The number of agents assisting individuals in the system who have the right to allow deviation.
- The probability of permission for policy deviation.

Since the decision to permit someone to deviate from an absolute-policy is eventually an individual decision in PoDMan, it can be stated that the probability of a user permitting deviation is mutually exclusive amongst all the users with the rights to do so. Thus to calculate the total probability of the permission to deviate being granted, the inclusion-exclusion principle (Szpankowski 2001) was used, which states

$$\left|A_1 \cup A_2 \cup \ldots \cup A_p\right| = \sum_{1\le i\le p}|A_i| - \sum_{1\le i_1\le i_2\le p}\left|A_{i_1} \cap A_{i_2}\right| + \sum_{1\le i_1\le i_2\le i_3\le p}\left|A_{i_1} \cap A_{i_2} \cap A_{i_3}\right| - \cdots + (-1)^{p-1}\left|A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_p}\right|$$

Here A_i is the probability of each user with the necessary permissions allowing another user to deviate from a policy while $\left|A_1 \cup A_2 \cup \ldots \cup A_p\right|$ is the total probability of the permission to deviate being granted.

In this simulation, the number of users was set with the rights to allow deviations; that is n, and randomly assign each user agent with the probability of them granting the permission to deviate. Furthermore, for each case of the simulation, a limit was set for the maximum probability (0% to 100%) of a user agent granting permission to deviate at intervals of 25%. This probability was generated randomly for each user agent and ranged between 0 and the maximum specified probability. The maximum probability parameter reflects the overall perception of policy deviation being the correct decision amongst the privileged users. This was the output parameter that we measured in this simulation; see Table 6.

Input Parameters	Output Parameter
- Number of users with deviation rights - Maximum probability of user permitting policy deviation*	- Total probability of deviation being permitted.

* Individual probability for each user is randomly generated between 0 and max.

Table 6: Inputs and output parameters for the simulation

5.2 Results

The simulation was run 100 times for each case and calculates the total probability of policy deviation being allowed each time. Finally, the average probability of deviation being permitted under each case was calculated.

. The results of the simulation are shown in Figure 15. A maximum probability of 0% represents an environment without PoDMan where no deviation is allowed. In this case, it can be seen that the overall probability stays at 0% regardless of the number of users in the environment as deviations are not possible under any circumstances. However, once the maximum probability of allowing deviation to 25% is increased, an increase is noted in the overall probability of deviation being permitted. Moreover, it can clearly be seen that the total probability of deviation increases with the increase of maximum probability for each user in the environment. In fact, unless the maximum probability all users in the system is 0, the total probability of deviation being permitted is always greater than 0%. This not only shows the effectiveness of PoDMan compared to a system without PoDMan, it also illustrates that the overall perception amongst all users affects the chances of deviation being permitted. Similarly, an increase is noted in the overall probability of deviation being allowed, i.e. an increase in effectiveness of PoDMan, when the total number of users in the system is increased.

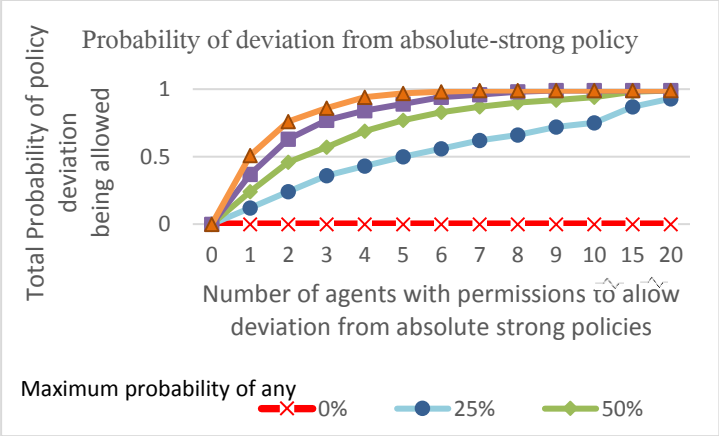


Figure 15: Simulation results: probability of absolute policy deviation being allowed

These results were obtained when the simulation was programmed in C++ and run on a 2.1 GHZ quad core CPU with 4 gigabytes of memory and running Ubuntu 13.04.

5.3 Discussion

The simulation was set out to test the effectiveness of PoDMan. In order to make sure that that PoDMan can be successful under different conditions, a simulation was designed to calculate the probability of a policy deviation while user agents are being permitted based on randomly assigned probabilities for their permissions. The simulation was run while the number of user agents was varying. These settings were deployed to perform an experiment in which different conditions could be tested. Under all conditions, it was observed that the probability of policy deviation being permitted is always greater when PoDMan is in use compared to when it is not. While this is of course expected, it is interesting to note that the overall perception of whether deviation should be permitted or not affects the total probability of deviation being permitted. E.g. if the maximum probability of an user permitting another user to deviate is 25%, the total probability of deviation being permitted remains much lower than when this maximum probability is higher. In other words, this shows that the total probability of deviation depends on the overall perception of the users and therefore the final decision of whether or not deviation is eventually permitted tallies with what the expected decision should be.

While this simulation might seem simplistic, it covers a whole range of circumstances that might occur in a collaborative organization and gives a generic picture of the effectiveness of PoDMan when used by multiple users.

6 Summary and Outlook

The main objective of this work was to assist individuals and organizations when exceptional or unexpected situations enforce the decision of policy deviation. The paper proposed an information technology methodology, including a framework and a process, to help users performing non-complying action; that is policy deviation management (PoDMan). The study was initially motivated by a real scenario observed in St Olvas hospital in Norway. While policies fail to manage a situation in which unexpected or exceptional conditions occurs, one can change the policy or perform a non-complying action. In real-time response, the first solution can only cover social policies that are changeable, however for legal policies changing would require authorities' permission and organizational processes and would not be possible in real-time. Therefore, the idea of policy deviation, or relaxing policies to perform non-complying action, policy deviation, was introduced by (LEWIS 2010) for legal policies. However, the idea lacks of technical details and a definitive method. A systematic review conducted to identify methods that can assist in implementation of policy deviation. However, it was found that the literature has ignored such a method to help individuals or organizations to decide about policy deviations and the non-complying actions.

The paper has introduced different types of deviation depending on the application of the policies in the scenario. The study has also presented a detailed method of how software agents can be used to detect the different types of policy deviations and how to compute and rank alternative actions when a policy deviation is detected.

6.1 Implications for research

The decision of choosing whether to deviate from a policy under exceptional circumstances is not a simple one, especially when users are unaware of all the policies that are in place and the legal implications behind them. Thus it is important to develop a method that makes this decision making process simpler and more controlled. The existing research on policy deviation focuses on policy adaption where policy parameters are modified to match changing environmental conditions or exceptions. Swanson provides a framework for policy adaption which involves setting up, adjusting and maintaining policies. The major drawback with this is that all the exceptions need to be considered before-hand or much after the event has already happened during the maintenance phase. PoDMan on the other hand presents a methodology with allows policy deviations and exceptions to be managed at run time and involves multiple parties in the decision making process when required. Along with this it opens up avenues for

researchers of decision support systems and CSCW to design and refine policy based systems in order to make them more flexible than they are currently.

6.2 Implications for practice

As more and more organizations move towards policy based collaborative systems, it becomes important to make sure that the right users have the right privileges to access information and perform tasks. While policies are designed keeping exceptions in mind, it is impossible for policy makers to consider every single exception. Thus there is a definite need for a more flexible system that allows policy deviations in a controlled manner when absolutely necessary. PoDMan provides a methodology to create such a system, and both the designers and users of policy based collaborative platforms can use PoDMan to add a higher degree to flexibility to their systems as PoDMan will not only allow deviations to take place under exceptional situations, it will also allow managers to monitor the exceptions and incorporate them into existing policies.

6.3 Limitations

The present work involves some limitations that we would like to point out.

6.3.1 Determining the magnitude of deviation

While PoDMan provides a method of managing and handling policy deviations by determining whether the policy being violated is a conditional or absolute strong policy, it does not inherently know the actual impact the policy violation has on the environment. Researchers in the future can refine the PoDMan by incorporating the magnitude of deviation or the impact of deviation in the process e.g. though policy ranking.

6.3.2 Managing policy updates

PoDMan provides managers and policy designers with the information containing all the deviation requests in the system using which they can modify existing policies to incorporate previously unforeseen exceptions. Currently PoDMan simply facilitates a manual process for policy maintenance where the information can be used to update the existing policies and branching time models. There is potential future work in developing methods to automatically update policies and BTMs when a previously unforeseen scenario is considered by the management.

6.3.3 Automatic generation of branching time models from decision trees

Currently the process of generating BTMs from decision trees is a manual process and can be tedious to create. It might be interesting to design intelligent systems which can automatically generate BTMs based on the existing policy and procedure documents for each role and update these BTMs automatically whenever policy or procedures are changed in the environment. The instruction to implement this automation has been explained in Section 6.

6.3.4 Complex decision process while complex actions and events are involved.

While BTM is an effective way of representing role behaviors and actions, the events and actions in this representation are primitive. i.e. the events are accepted directly by the agents and the actions are performed directly by the agents as well. This is obviously an idealization. In real settings the events and the actions are usually very complex and made up different events and actions. In some cases this representation might be too simplistic and more complex behaviors might need to be represented to accurately describe the roles. Future research can explore methods of representing complex events and actions through BTMs or develop a method to reduce complex actions into multiple primitive actions.

Having done that, the decision process would be possible to be deployed in real settings before transforming the complex events and actions to primitive ones.

6.3.5 Real life implementations

While the current evaluation demonstrates how PoDMan can be applied in real life scenarios to prove its applicability in real life, an actual organizational implementation should provide a greater insight to the strengths and weaknesses of this technique.

In addition, although this thesis provides a generic methodology for policy deviation under exception using information systems, it would be interesting to study how it can be used to design complete systems in different domains such as banking and health care. It would also be of interest to note and compare how these systems would perform under actual emergencies such as natural disasters or terrorist attacks.

Acknowledgements

The work was partly supported by the ARC Centre for Excellence in Population and Ageing Research (CEPAR), UNSW and the Smart Services CRC. The work was carried out at the Asia Pacific ubiquitous Healthcare Research Centre (APuHC), UNSW.

References

Bakshi, A., Talaei-Khoei, A., & Ray, P. (2013). Adaptive policy framework: A systematic review. *Journal of Network and Computer Applications*, 36(4), 1261-1271.

Benbasat, I., & Nault, B. R. (1990). An evaluation of empirical research in managerial support systems. *Decision Support Systems*, 6(3), 203-226.

Boella, G., & van der Torre, L. (2003, July). Attributing mental attitudes to normative systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems* (pp. 942-943). ACM.

Chau, M., Zeng, D., Chen, H., Huang, M., & Hendriawan, D. (2003). Design and evaluation of a multi-agent collaborative Web mining system. *Decision Support Systems*, 35(1), 167-183.

Czap, H. (2002). Policy-agents for negotiation within virtual organizations. International *Journal of Information Technology & Decision Making*, 1(03), 385-400.

Ellis, C. (1991). Groupware: Overview and perspectives. In *Verteilte Künstliche Intelligenz und kooperatives Arbeiten* (pp. 18-29). Springer, Berlin, Heidelberg.

Fitzpatrick, G., & Ellingsen, G. (2013). A review of 25 years of CSCW research in healthcare: contributions, challenges and future agendas. *Computer Supported Cooperative Work* (CSCW), 22(4-6), 609-665.

Lewis, L., Parameswaran, N., & Ray, P. (2010, July). Fusion and policy relaxation in patient healthcare. In *2010 13th Conference on Information Fusion* (FUSION), (pp. 1-5). IEEE.

Lotlikar, R. M., & Mohania, M. (2006, September). Adaptive policies in information lifecycle management. In *International Conference on Database and Expert Systems Applications* (pp. 612-621). Springer Berlin Heidelberg.

Migon, L. B., Borges, M. R., & Campos, M. L. M. (2010, September). A method for identification and representation of business process deviations. In *International Conference on Collaboration and Technology* (pp. 49-64). Springer Berlin Heidelberg.

Moffett, J. D., & Sloman, M. S. (1991, October). The representation of policies as system objects. In *ACM SIGOIS Bulletin* (Vol. 12, No. 2-3, pp. 171-184). ACM.

Ouyang, S. (2006, May). Integrate Policy based Management and Process based Management--A New Approach for Workflow Management System. In *10th International Conference on Computer Supported Cooperative Work in Design*, 2006. CSCWD'06. (pp. 1-6). IEEE.

Prakash, A., Shim, H. S., & Lee, J. H. (1999). Data management issues and trade-offs in CSCW systems. *IEEE Transactions on Knowledge and Data Engineering*, 11(1), 213-227.

Rao, A. S., & Georgeff, M. P. (1991). Modeling Rational Agents within a BDI-Architecture. *KR*, 91, 473-484.

Rao, A. S., & Georgeff, M. P. (1995, June). BDI agents: From theory to practice. In *ICMAS* (Vol. 95, pp. 312-319).

Ray, P., Shahrestani, S. A., & Daneshgar, F. (2005). The role of fuzzy awareness modelling in cooperative management. *Information Systems Frontiers*, 7(3), 299-316.

Ray, P., Parameswaran, N., Chan, V., & Yu, W. (2008). Awareness modelling in collaborative mobile e-health. *Journal of telemedicine and telecare*, 14(7), 381-385.

Ray, P., & Chattopadhyay, S. (2009). Fuzzy awareness model for disaster situations. *Intelligent Decision Technologies*, 3(1), 75-82.

Shum, S. B., Cannavacciuolo, L., De Liddo, A., Iandoli, L., & Quinto, I. (2013). Using social network analysis to support collective decision-making process. In *Engineering Effective Decision Support Technologies: New Models and Applications* (pp. 87-103). IGI Global.

Sloman, M. (1994). Policy driven management for distributed systems. Journal of network and Systems Management, 2(4), 333-360.

Strassner, J., De Souza, J. N., Van der Meer, S., Davy, S., Barrett, K., Raymer, D., & Samudrala, S. (2009). The design of a new policy model to support ontology-driven reasoning for autonomic networking. *Journal of Network and Systems Management*, 17(1-2), 5-32.

Strassner, J., Liu, Y., Jiang, M., Zhang, J., van der Meer, S., Foghlú, M. Ó., ... & Donnelly, W. (2008, April). Modelling context for autonomic networking. In *IEEE Network Operations and Management Symposium Workshops*, 2008. (pp. 299-308). IEEE.

Strassner, J., Liu, Y., & Zhang, J. (2008, July). A Context-Aware Policy Model to Support Autonomic Networking. In *32nd Annual IEEE International Computer Software and Applications*, COMPSAC'08. (pp. 1097-1102). IEEE.

Swanson, D. & Bhadwal, S. (2009). *Creating adaptive policies: A guide for policymaking in an uncertain world*, Sage Publications Pvt. Ltd.

Swanson, D., Venema, H. D., Rust, C., & Medlock, J. (2000). *Understanding adaptive policy mechanisms through farm-level studies of adaptation to weather events in Alberta, Canada*. International Institute for Sustainable Development.

Szpankowski, W. (2001). *Inclusion-Exclusion Principle. Average Case Analysis of Algorithms on Sequences*, 49-72.

Talaei-Khoei, A., Solvoll, T., Ray, P., & Parameshwaran, N. (2011). Policy-based Awareness Management (PAM): Case study of a wireless communication system at a hospital. *Journal of Systems and Software*, 84(10), 1791-1805.

Twidle, K., Dulay, N., Lupu, E., & Sloman, M. (2009, April). Ponder2: A policy system for autonomous pervasive environments. In *Fifth International Conference on Autonomic and Autonomous Systems*, 2009. ICAS'09. (pp. 330-335). IEEE.

Ulieru, M., & Worthington, P. (2005, October). Holonic risk management framework. In *IEEE International Conference on Systems, Man and Cybernetics*, (Vol. 1, pp. 209-214). IEEE.

Ulieru, M., & Worthington, P. (2006). Autonomic risk management for critical infrastructure protection. *Integrated Computer-Aided Engineering*, 13(1), 63-80.

Verhagen, H. (2001, December). Norms and artificial agents. In Sixth Meeting of the Special Interest Group on Agent-Based Social Simulation, ESPRIT Network of Excellence on Agent-Based Computing.

Woolridge, M., & Wooldridge, M. J. (2001). *Introduction to Multiagent Systems*. New York, NY, USA: JohnWiley & Sons.

Yuan, Y., & Detlor, B. (2005). Intelligent mobile crisis response systems. *Communications of the ACM*, 48(2), 95-98.