A Fuzzy, Utility-Based Approach for Proactive Policy-Based Management

Christoph Frenzel[1,2], Henning Sanneck[2], and Bernhard Bauer[1]

[1] Department of Computer Science, University of Augsburg, Augsburg, Germany
{frenzel,bauer}@informatik.uni-augsburg.de
[2] Nokia Siemens Networks Research, Munich, Germany
henning.sanneck@nsn.com

Abstract. Policy-based Management with rules is a wide-spread approach for operations automation. However, the continuous pressure for decreasing operational costs and increasing reliability of the systems lead to new challenges. Unfortunately, current Policy-based Management Systems lack the ability to act proactively along operational objectives in an autonomous manner in order to face these challenges. In this paper, we present a Policy-based Management System based on a Fuzzy Logic System that attempts to avoid problematic system states before they occur and that is guided by operator objectives expressed as utilities. Our approach can be seen as an extension of current rule-based Policy-based Management Systems, thus, requiring a reduced implementation effort.

1 Introduction

Policy-based Management (PBM) is the continuous process of configuring the resources of a system such that the overall system performance satisfies the objectives of the operator. This process is controlled by a policy which encodes the technical knowledge and the preferences of the operator, usually as a set of rules like Event-Condition-Action (ECA) rules [3], [16]. This type of rule is triggered by an event which leads to the evaluation of the condition and, depending on the outcome, proposal of an action. The event is raised if the performance of the system is unacceptable which is usually determined through sharp constraints on Key Performance Indicators (KPIs). However, for an event there are usually several possible ways to react, i.e., several actions can be taken. In order to avoid these policy conflicts, human experts need to extend the rule conditions so that the PBM system selects the best of the possible actions in a specific situation based on the operator objectives and their technical experience [7]. As a result, the complexity of the policy is increased because the technical knowledge and operational objectives are mixed up.

In the future, the level of automation of PBM is required to be extended, driven by the increasing need for cost-efficient and reliable operations, as well as

the increasing complexity which prevents the understanding of the whole system by the operator. This leads to new challenges that a Policy-based Management System (PBMS) has to face. For instance, PBMSs are supposed to make more complex decisions automatically. This requires policies which directly express the operational objectives separately from the technical knowledge. This way, a PBMS is enabled to autonomously act towards achieving the objectives [10]. Furthermore, PBMSs must act proactively in order to avoid problems instead of reacting to problems that are already present. Hence, the sharp distinction between acceptable and unacceptable system states is not applicable anymore, and needs to be substituted with a fuzzy system state characterization that allows the system to react to gradually decreasing system performance. In summary, PBMSs need to proactively control the system guided by operational objectives.

In this paper, we present an approach for proactive PBM in order to face future challenges for PBM. Thereby, given the requirements for proactive PBM (Sec. 2), we outline the expected system behavior (Sec. 3), and then present the design of our solution (Sec. 4). The main idea behind the concept is to separate technical knowledge, expressed in fuzzy rules, from operator objectives, encoded in utilities, and utilize a fuzzy logic system in order to compute which actions satisfy the objectives the most. The approach can be seen as an extension of current rule-based PBMSs, thus, requiring a reduced initial implementation effort. The advantages of our system can be shown by comparing it with classical PBMSs (Sec. 5).

2 Problems of Classical PBM

It is assumed that the systems which are managed by a PBMS are composed of several system resources which are running autonomously controlled by some parameters. Driven by external or internal events, the operational state of the system, which is determined by the values of a set of KPIs and the presence of alarms that are produced by the system resources, can change. Thereby, the system can also transition into problematic operational states, e.g., if many users connect to a network at the same time, this can lead to an overload. In order to handle these undesired states, a PBMS, depicted in Fig. 1, continuously monitors the system using probes that identify unacceptable states, i.e., problems, and raise events that indicate the nature of the detected problems. Thereby, the distinction between acceptable and unacceptable states is sharp, usually defined through constraints, e.g., a KPI k representing a ratio could have a constraint requiring that $k > 2\%$.

The decision making component evaluates the policy and proposes actions in reaction to the events, thereby considering the operational context of the system. The latter can be any information about the system, e.g., the current time or Configuration Management (CM) data like the system topology. An action that is proposed can be, e.g., the setting of new configuration parameter values of a system resource or the call of some special system function that determines new parameters and configures a system resource accordingly.

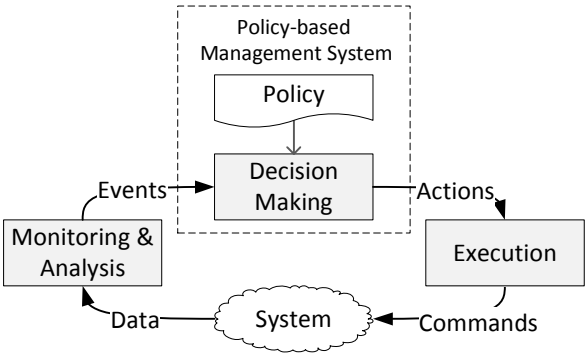

Fig. 1. The PBM process

Usually, the PBMS will propose several actions at the same time because, on the one hand, there can be several rules triggered by one event and, on the other hand, several events can be raised in parallel. A common challenge is that the actions might be in conflict [13], [3], i.e., they are somehow mutually exclusive. Policy designers try to detect these conflicts using sophisticated methods and resolve them by adapting the rule. Thereby, they often add further conditions to the rules which determine in which situations one action is preferred over another one based on the operational objectives. As a consequence, the policy mixes up both the technical knowledge about which action is a reasonable reaction for an event in a specific situation, and the operator objectives which describe the operator's preferences for the actions in specific situations. However, since it is not possible to detect and resolve all policy conflicts at design-time [13], there is still the need for an on-line conflict resolution. Unfortunately, current systems do not allow to encode complex operational objectives for this.

Figure 2 depicts an example from mobile networks management where a PBMS can distinguish between four operational states given two KPIs, the Dropped Call Rate (DCR) and the Call Setup Success Rate (CSSR). The DCR indicates unacceptable reliability problems if the DCR is high, whereas the CSSR indicates unacceptable availability problems if the CSSR is low. The two lines show the thresholds that are defined for the KPIs separating the acceptable from the unacceptable states. Specifically, Zone 1 is acceptable with respect to both KPIs, Zone 2 is unacceptable regarding DCR but acceptable regarding CSSR, Zone 3 is acceptable regarding DCR but unacceptable regarding CSSR, and Zone 4 is unacceptable for both KPIs. In a classical PBMS, a probe would raise an event as soon as an unacceptable state regarding a KPI is reached. Consequently, in Zone 4 two events indicating the DCR problem and CSSR problem would be triggered. Hence, the PBMS is required to perform some conflict resolution if the respective actions are in conflict.

The sharp thresholds in classical PBM do not allow to proactively react to possibly upcoming events. One solution to this is to lower the thresholds and, so, reduce the number of acceptable system states. In Fig. 2, this would mean to shrink Zone 1 to the dashed rectangle. However, then the system cannot distinguish between an event indicating a system state in Zone 1, i.e., a proactive

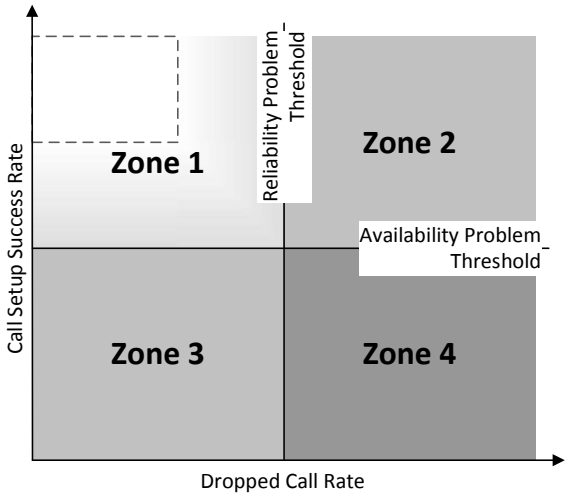

Fig. 2. Operational states for the KPIs DCR and CSSR

but not severe problem, and an event indicating a system state in Zone 2, i.e., a severe problem. As a result, the PBMS might select non-optimal actions.

If the level of automation in PBM needs to increase in the future, a classical PBMS as outlined in this section has two shortcomings:

- The sharp differentiation between acceptable and unacceptable operational states solely allows a reactive behavior, i.e., the problem must occur before the system can take countermeasures. However, it is desirable that the system avoids problematic state by proactively performing some action.
- The mingling of technical knowledge and operational objectives makes the maintenance of the policy costly. Both have different life cycles, i.e., the technical knowledge usually changes less frequently than the operational goals, thus, with every single change, the whole policy needs to be revised.

3 A Concept for Proactive PBM

In order to solve the issues with classical PBMSs, we model a utility-based rule system as a specification for a fuzzy logic system.

A utility-based rule system allows the separation of technical knowledge and operator objectives by expressing the former as rules, e.g., ECA rules, and the latter as utilities [7]. Upon a raised event, the rules are evaluated and applicable actions proposed, i.e., actions that treat the event from a purely technical point of view. Since these actions might be in conflict, a conflict resolution utilizes the utilities in order to calculate the *value* of each action based on the events an action handles, the *utility* of the events' treatment, and the severity of the events. Finally, an action is selected for execution based on its value, i.e., usually the action with the highest value. This can be seen as a kind of rational behavior [15].

A fuzzy logic system is a rule system that can perform logical inference with fuzzy sets, i.e., it can fuzzify input values into fuzzy sets, apply fuzzy rules

on these sets, and defuzzify the resulting sets into sharp output values [12]. The general idea is to replace boolean predicates used in classical rule systems with continuous memberships in the domain $[0, 1]$ representing the degree to which a predicate is true. Fuzzy logic systems provide a scientific and well-known foundation for decision making in uncertain or inaccurate domains. There are several implementations of fuzzy logic system available, e.g., jFuzzyLogic [5], and also a standard syntax, called Fuzzy Control Language [9], that is used in this paper.

The basic idea of combining a utility-based rule system and a fuzzy logic system is to

- fuzzify the outputs of the monitoring probes into fuzzy event levels, i.e., each system state has a fuzzy membership to the set of unacceptable states regarding each event in the interval $[0, 1]$,
- model the technical knowledge as fuzzy rules and weight them with the utilities, i.e., the preferences of the operator,
- utilize a fuzzy logic system to compute the value of each action, i.e., the degree to which an action satisfies the operator's preferences [7], and
- defuzzify the result such that conflicts between actions are resolved by selecting the actions according to their value.

In order to outline the expected system behavior, consider Fig. 2 again. The sharp constraints distinguishing acceptable and unacceptable states are substituted by fuzzy event levels, i.e., fuzzy memberships to the unacceptable zone indicated by the events. Therefore, the Zone 1 is divided into two sub-zones:

- A zone indicating a perfect system state, i.e., a state where the system is perfectly working and no problem is supposed to arise soon. In this zone, depicted as the dashed rectangle, all fuzzy event levels are 0.
- A jeopardy zone [1], i.e., states where the system performance is still acceptable but there is the danger of running into a faulty state. Therefore, the system should react in order to avoid the unacceptable state. In this zone, indicated by the gradient, at least one fuzzy event level is greater than 0.

Given the fuzzy event levels, the system selects the best action with respect to the level of the events and the utility the action has to the operator, i.e., how important the treatment of the problem is according to the operator goals.

4 Design of the Fuzzy PBMS with Utilities

The reasoning process of the fuzzy PBMS with utilities, depicted in Fig. 3, is performed in three steps which are spread over the monitoring and analysis as well as the decision making phase of the PBM process depicted in Fig. 1:

Fuzzification is performed by the monitoring probes in the monitoring and analysis phase. Thereby, sharp thresholds for the decision to raise an event are replaced by fuzzy membership functions defined by the system operator in the event specification. As a result, the probes are raising fuzzy events, i.e., events containing a fuzzy event level representing the membership.

Fuzzy inference is performed as first step of the decision making phase. Based on fuzzy events and fuzzy context information, the fuzzy rules are evaluated. The rules, which are provided by the system administrator, define possible actions in reaction to some event in a specific operational context. As a result, this process provides the value of all actions based on the system objectives which are provided as utilities by the operator.

Defuzzification is the second step of the decision making phase. The proposed actions are analyzed for conflicts that are defined in the conflict specification given by the administrator. The detected conflicts are resolved by selecting actions for execution based on the value.

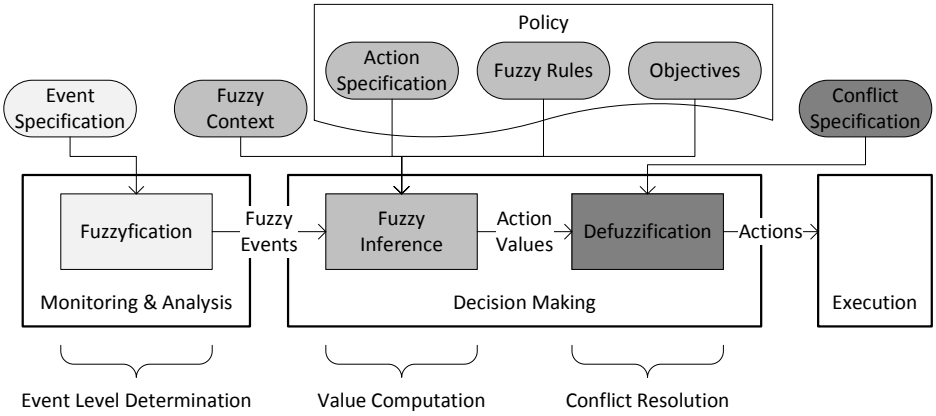

Fig. 3. The reasoning process of the fuzzy PBMS with utilities

4.1 Fuzzification

The fuzzification extends the monitoring probes with the ability to annotate the events with an event level in order to turn them into fuzzy events. This level has the semantics of the degree to which the operator wants to handle the event in order to avoid a possible negative system state in the future. The event levels are computed by membership functions, e.g., sigmoid or linear functions, which determine the membership degree of a system state to the fuzzy set raised for an event. The membership functions form the event specification provided by the operator. They are determined through an analysis of the system behavior, e.g., whether the system state changes rapidly, and the trade-off between the benefits of reacting proactively and the efforts of executing potentially more actions.

The thresholds used by classical monitoring probes define two classes of system states: the acceptable state and the unacceptable state. In fuzzy monitoring probes, the membership functions define three classes of system states as shown in Fig. 4: the acceptable state, the unacceptable state, and the jeopardy state in between. In the concrete example, the PBMS should not react if the DCR is below 1%, because the system works perfectly fine. This is indicated by an event level of 0 and, so, the event would not be raised at all. Between a DCR of 1% and 2% the event level linearly increases from 0 to 1. Hence, in this jeopardy

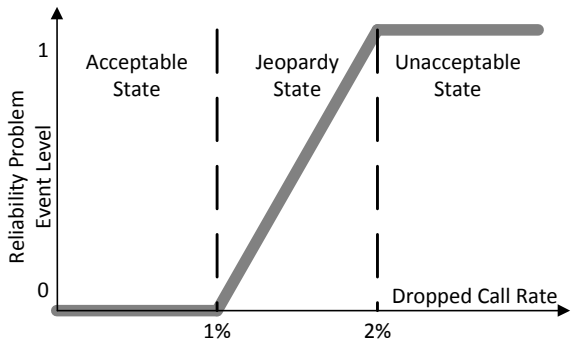

Fig. 4. The three classes of system states determined by fuzzy monitoring probes

state the handling of the problem becomes more and more urgent. Finally, at a DCR of 2% and above, the system is in an unacceptable state. Hence, the event level is 1 in this range. Note that a classical threshold would be a DCR of 2%.

4.2 Fuzzy Inference

The fuzzy inference is the core of the system and determines the value of the actions as shown in Fig. 3. Besides the fuzzy events, it has four inputs:

- A fuzzy context that provides contextual information about the system.
- An action specification defining the actions the system can take.
- A set of operator objectives including their utilities.
- A set of fuzzy rules that define the possible reactions to a fuzzy event in a specific context from a technical point of view.

Fuzzy Context. The operational context of the system contains information about, e.g., the system topology, system configuration, current system status, and current time and date. In principle, the context needs to provide all the information that is necessary for evaluating the conditions of the fuzzy rules.

The context is fuzzified by using membership functions just like the events. For instance, instead of representing the time with a concrete hour and minute, it is possible to define fuzzy sets like daytime and nighttime. Note that the membership functions can also define crisp sets.

Objectives. Utility theory [15] provides a framework to represent and reason with complex objectives by using a single measure called utility. In other words, the utility represents the degree to which an operational state satisfies the objectives, so, allowing to compare different states in order to make decisions. The elicitation of utilities is a non-trivial process and an active field of research [6].

In the presented approach, each fuzzy rule has an assigned objective and each of these objectives is mapped to a utility, i.e., a real value. The semantics of a rule is that if the action of the rule is executed in response to the event under the operational context determined by the rule, then it fulfills the assigned objective

and produces the respective utility. Technically, the objectives and utilities are a set of variables and their respective real values in the fuzzy logic system.

Although this approach allows a fine-granular rule–objective mapping, it is not recommended to assign different objectives to every rule due to the costly utility elicitation. An example for a reasonable objective assignment is the definition along the events that can be raised. That means that there is an objective for every event and the utility represents the importance of the event resolution.

Action Specification. The action specification defines the values of the output variable of the fuzzy logic system, i.e., the representation of the possible actions. Specifically, it defines the output variable action as well as a crisp singleton set, i.e., a single value, for each action on that variable. Although singletons are not common in fuzzy logic systems, it is a reasonable modeling approach since the single values represent discrete actions on the continuous variable action. For instance, consider the two actions Mobility Robustness Optimization (MRO) and Mobility Load Balancing (MLB) that are treating a reliability problem and an availability problem respectively. For each, a singleton set is created on the variable action, e.g., MRO is linked to a singleton set with the value 1 and MLB is linked to a singleton set with value 2. Note that, depending on the defuzzification method, the values of the actions can represent preferences (cf. Sec. 4.3).

Fuzzy Rules. The fuzzy rules express the expected behavior of the PBMS from a technical point of view. Hence, these rules should be designed based on the technical knowledge of the rule designers without considering the operational goals. Specifically, these rules can contain policy conflicts. We consider Mamdani-style [12] fuzzy rules with the following general structure:

IF event IS raised AND condition **THEN** action IS singleton **WITH** objective

The form of the rules is aligned with the ECA rule pattern. It has three parts:

IF part or antecedent consists of two components which are connected by a conjunction. First, the evaluation of a single event level, i.e., the membership of event to the set raised. Second, a condition part which can be any fuzzy logic formula over the operational context.

THEN part or consequent proposes a single action by setting the value of the output variable action to a singleton set representing the proposed action.

WITH part weights the rule with the utility of the assigned objective.

Note that this general rule structure also allows to model complex technical knowledge as well as operator objectives (cf. [7]). For instance, by extending the WITH part to be the product of the objective's utility and an *effectiveness*, one can represent a confidence in the action, i.e., the likeliness that the action will treat the event correctly and generate the utility. Furthermore, imagine that actions have fixed costs: this can be modeled by creating rules which have an empty antecedent, i.e., they are always triggered, and a cost instead of an objective which is some negative number.

System operators working with PBMS are used to ECA rules and so, they might be unfamiliar with fuzzy rules. Furthermore, the structure of the fuzzy rules is inconvenient since it is driven by the technical capabilities of a fuzzy logic system. However, a simple transformation is able to translate ECA rules into fuzzy rules. Thereby, it is necessary to group the ECA rules into policy groups [16] which model the objectives the rules are fulfilling.

Inference. A fuzzy logic system evaluates each fuzzy rule and calculates the output membership of the proposed action, i.e., the *expected utility*. Thereby, the fuzzy logic semantics is aligned with the ECA paradigm, i.e., AND, OR, and NOT are interpreted as usual as minimum, maximum, and complement [12]. The expected utility of a rule r, which proposes an action, is calculated as

$$U_{\text{exp}}(r) = W_{\text{act}}(r)U(r) \tag{1}$$

with, $W_{\text{act}}(r)$ being the rule activation, i.e., the combined fuzzy membership of the antecedent, and $U(r)$ being the utility of the objective of the rule.

After that, the expected utilities of the fuzzy rules must be combined in order to compute the value of each action. This combination is defined by the *accumulation method* in a fuzzy logic system. A common approach is to sum up the utilities that an action produces, however, this requires mutual preferential independence [15] between the objectives, i.e., the utility of an objective for an action is independent of the expected utilities of the other objectives for that action. This assumption is implied by the structure of the objectives and fuzzy rules, specifically the direct rule–objective assignment. However, the expected utilities cannot be simply summed up since several rules can propose the same action with the same objective. Hence, the accumulation needs to distinguish between rules assigned to the same objective and rules assigned to different objectives as outlined by the following example.

Consider the fuzzy rules r_{1a} and r_{1b} that are both assigned to objective o_1, and rule r_2 which is assigned to objective o_2. All three rules are triggered by the event e and propose the action a. In this setting, the expected utilities for o_1 from r_{1a} and r_{1b} should count once since, even though the action has been proposed twice, it can satisfy o_1 only once. However, the expected utilities for o_1 and o_2 are summed up since the action satisfies both objectives in parallel.

Formally, the value V of a action a is defined as:

$$V(a) = \sum_{\rho \in R(a)/\sim} \max(\{U_{\text{exp}}(r) | r \in \rho\}) \tag{2}$$

whereby $R(a)/\sim$ is the partition of the set of all rules proposing a with respect to equal objectives. Hence, the value of an action is determined by, first, calculating the maximum of the expected utilities for each set of rules that have the same objective and, second, summing up these maxima.

Technically, the accumulation can be implemented by grouping the fuzzy rules assigned to the same objective into one Fuzzy Control Language rule group.

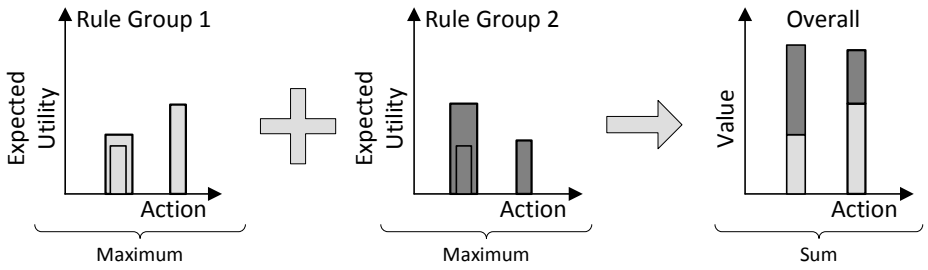

Fig. 5. The accumulation of the expected utilities to the value

Within a rule group, the expected utilities of the rules are accumulated by selecting the highest one, whereas the combined expected utilities of different rule groups are summed up as shown in Fig. 5. This functionality is not standardized and, so, this usually requires an adaptation of the fuzzy logic system. Note that this adaptation can be avoided if it can be ensured that no two rules that propose the same action for the same objective are ever triggered together.

4.3 Defuzzification

All actions that have a value greater than 0 are seen as proposed for execution. However, not all can be executed in parallel because there might be conflicts between them which are modeled in the conflict specification shown in Fig. 3. Hence, the defuzzification selects the best actions that can be executed based on their values. The actual output of the defuzzification are real numbers which represent values of the variable action. They can be translated into the actions by using the action specification for the fuzzy inference.

In simple cases, all actions are in conflict, i.e., only one action can be selected at a time. Then, the best action is the one with the highest value since it satisfies the most severe and important objectives. It can be selected from the fuzzy variable action using a standard defuzzify method which selects the singleton set with the highest membership degree. Thereby, the operator can prioritize actions in the action specification if there are ties. For instance, if ties are broken by selecting the action with the smallest value of the variable action, the operator should assign smaller variable values to the actions that are more preferred.

More sophisticated defuzzification methods can also determine a set of best actions if the conflicts are more complex. Suppose that the conflict specification is a set of pairs of actions that cannot be executed together. So, the defuzzification needs to find a combination of actions the maximizes the overall value produced by the selected actions. This assignment problem can be formulated as a constraint optimization problem which can be solved with a constraint optimizer.

5 Case Study

In the following case study, we show the application of the fuzzy PBMS with utilities in mobile networks management and evaluate its performance using a demonstrator system based on jFuzzyLogic [5].

5.1 Scenario

The scenario is a 3rd Generation Partnership Project (3GPP) Long Term Evolution (LTE) network which is managed as a Self-Organizing Network (SON), i.e., the network has self-configuration, self-optimization, and self-healing features [8]. A SON is characterized by autonomous SON functions which continuously monitor the network and trigger the execution of algorithms in order to resolve detected problems. Thereby, the SON functions can influence each other, e.g., via common configuration parameters (output of SON functions) or commonly influenced KPIs (as input to SON functions), which leads to conflicts. These conflicts are detected and resolved by the SON coordination function.

The PBMS for the case study is a simplified SON coordination. The SON functions can be seen as monitoring probes which trigger the execution of some resolution action. Since the monitoring of the functions is quite radio specific, the details of the fuzzification are not presented here. Instead, we concentrate on the decision making component, i.e., the SON coordination, and consider the fuzzy events as given. We assume that, in discrete time intervals, all raised fuzzy events are passed to the SON coordination which determines suitable actions to resolve the problems, analyzes their values with respect to the operator objectives, and triggers the best action. Thereby, all actions are assumed to be mutually in conflict in order to keep the case study simple.

5.2 System Model

Figure 6 depicts the technical knowledge that is encoded in the fuzzy rules for the fuzzy logic system. There are five fuzzy events that can be raised and six actions that can be triggered. The arrows between the events and actions outline the rules of the system, i.e., for each event e and action a pair which is connected by an arrow, there is a rule which proposes a if e is present. However, a rule can also contain a context condition which is not depicted. For instance, the arrow between reliability problem and Remote Electrical Tilt (RET) optimization represents the following fuzzy rule:

IF reliability_problem IS raised AND ret_available IS true
THEN action IS ret_optimization **WITH** objective_dcr

Note that the rule evaluates the context whether the RET feature is available. The events in SON encode specific network problems and, so, we defined the objectives with respect to the fuzzy events, i.e., there is an objective for each event. The utilities of these objectives are normalized, i.e., each utility is in the range $[0, 1]$ and the sum of the utilities is 1.

The fuzzy PBMS with utilities is evaluated against two simpler PBMSs:

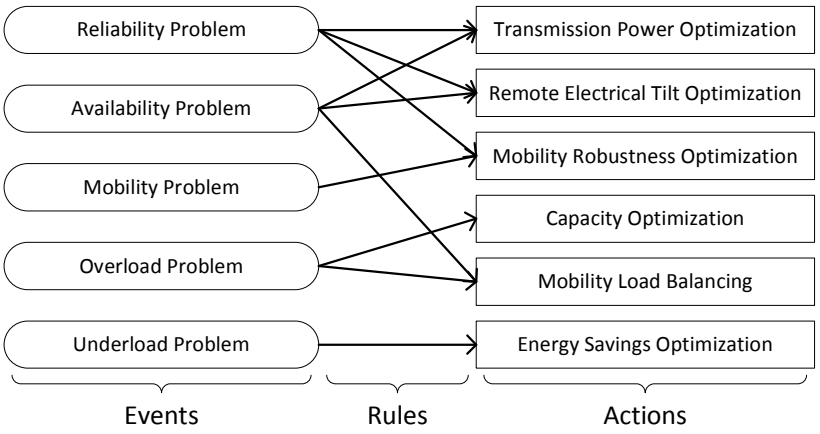

Fig. 6. Events, actions, and fuzzy rules in the case study

Classical PBMS considers neither fuzzy event levels nor operator objectives. Hence, the event fuzzification is actually a sharp threshold which is set to the border between jeopardy state and good state. Furthermore, the utility of all objectives is equal.

Fuzzy PBMS uses fuzzy events in order to take the severity of the events into consideration. However, the utilities of all objectives are equal.

Fuzzy PBMS with utilities considers both fuzzy events and utilities.

Operators adopting the fuzzy PBMS with utilities will likely start with the classical approach and introduce the fuzzy PBMS as an intermediate step. After gaining some experience with the system, the operator will adapt the utilities to meet the true objectives and, so, switch to the fuzzy PBMS with utilities.

5.3 Evaluation

The evaluation is performed by comparing the performance of the three PBMSs regarding the average value of the actions they select. Several sets of operator objectives, i.e., utilities, and problem situations, i.e., event levels and contexts, are created randomly. Thereby, the probabilities for an event level of 0 or 1 is 0.25 each, whereas, the event levels in $]0, 1[$ are uniformly probable. Finally, each combination of operator objectives and problem situation is fed into the three PBMSs and the values of the proposed actions with respect to the operator objectives and the problem situation are recorded.

Figure 7 depicts the average value of the selected actions for 100 different objective sets, each evaluated with 1000 different problem situations. As expected, the classical PBMS has the lowest average value since it utilizes the least information, i.e., it neglects the fuzzy event levels and utilities. As a result, this system always picks the action which resolves most of the raised events. The fuzzy PBMS performs better since it utilizes the event level information. Hence, it neglects events with low levels, i.e., that are less important, and concentrates on the events with high levels. In summary, this system picks the action with the

highest sum of the levels of the resolved events. However, the best performance is shown by the fuzzy PBMS with utilities because it knows the objectives of the operators and, so, concentrates on the events with high levels that are also valuable to the operator. Actually, the fuzzy PBMS with utilities always selects the optimal action in this scenario where the actions are mutually conflicting.

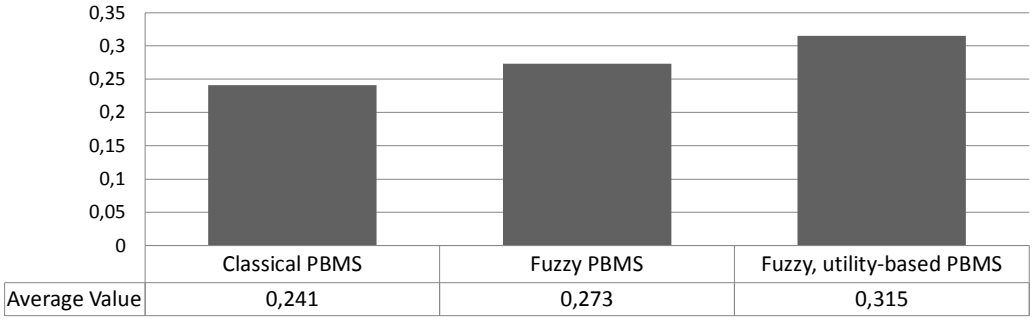

Fig. 7. Average value produced by the three PBMSs

Although the ranking of the PBMSs is not surprising, it is interesting that the fuzzy PBMS performs 13% better than the classical system and the fuzzy PBMS with utilities performs 15% better than the fuzzy system. This significant performance increase between these two evolutionary steps indicates that a gradual adoption of the fuzzy PBMS with utilities is a reasonable approach.

6 Related Work

Using a fuzzy logic system for systems management is not a new idea. For instance, Kousaridas and Nguengang [11] utilize two fuzzy logic systems in sequence: the first estimates the network state given low-level technical measurements and the second proposes actions to be performed based on the fuzzy network state. The fuzzy membership degree is used to compute a confidence in the decision. However, the approach neither separates the operator objectives from the technical knowledge nor considers values.

Fuzzy logic systems have also been used for multi-criteria decision making [14], [2], [17], i.e., the evaluation of preferences for specific decision options with respect to some objectives. Thereby, the system is given a fuzzy satisfaction value for each action regarding each objective. Based on these values, the system selects the best action using the Max-Min selector, i.e., it determines the minimal satisfaction value regarding an objective for each action and selects the action with the highest minimal value. Boutalis and Schmidt [4] present a nice application for a fuzzy discrete event system in the domain of mobile robots. However, the pessimistic multi-criteria decision making approach is not applicable for PBM: usually no action is satisfying all objectives at once and, thus, the minimal satisfaction value for all actions would be 0. Hence, we adopt a more decision theoretic approach [15] which can be seen as a Sum-Max selector. Nevertheless,

by changing the accumulation methods (cf. Sec. 4.2) appropriately, one can also use the pessimistic Max-Min selector with our approach.

In [7], we have presented a PBM approach that considers uncertainty in the inputs and operational goals. The Rational Policy System extends an ECA rule system with probabilistic events and a dynamic action conflict resolution that is based on the value of the actions. Hence, the approach makes a clear distinction between technical knowledge, represented in ECA rules, and operator objectives, encoded in utilities. However, there are some differences between the Rational Policy System and the presented approach. First, the reasoning of the Rational Policy System is hard coded in program code and, so, less flexible to adapt than the fuzzy logic problem formulation of our approach, e.g., if an operator decides to use the Max-Min selector. Second, in contrast to the Rational Policy System which solely allows for probabilistic events, the fuzzy logic approach allows both fuzzy events and fuzzy context information. Third, the semantics of probabilistic events used in the Rational Policy System is different from fuzzy events.

Another approach for PBM with utilities and uncertainty is presented by Bartolini et al. [1]. In order to keep the efforts for system modeling low, they introduce a jeopardy system state for each KPI, i.e., a special acceptable state in which the system soon runs into an unacceptable state with some probability. The system selects the best action based on the system state, a specification of the probabilistic action effects, and the operator objectives. However, modeling the system behavior as probabilistic action effects instead of rules is a complex mental process for policy designers who are used to rule-based PBMSs.

7 Conclusion

In this paper, we presented a fuzzy PBMS with utilities which enables proactive PBM. It determines the best action in response to fuzzy event with respect to operator objectives. Thereby, technical knowledge, expressed in fuzzy rules, and operator objectives, encoded in the utilities, are separated to facilitate different life-cycles of both information models. The focus has been on providing an extension of classical PBM which requires little modeling effort.

In the future, it seems very promising to add a learning component to the system which can estimate the effectiveness of the rules by observing the system behavior. Furthermore, the system needs to avoid the recurring execution of actions if they fail to work. This can be done by keeping a history of the executed actions and avoid them.

References

1. Bartolini, C., Sallé, M., Trastour, D.: IT service management driven by business objectives: an application to incident management. In: Proc. 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006), pp. 45–55. IEEE, Vancouver (2006)
2. Bellman, R.E., Zadeh, L.A.: Decision-Making in a Fuzzy Environment. Tech. Rep. NASA CR-1594, National Aeronautics and Space Administration (1970)

3. Boutaba, R., Aib, I.: Policy-based Management: A Historical Perspective. Journal of Network and Systems Management 15(4), 447–480 (2007)
4. Boutalis, Y., Schmidt, K.: Multi-objective decision making using fuzzy discrete event systems: A mobile robot example. In: Proc. 18th Mediterranean Conference on Control and Automation (MED 2010), pp. 575–580. IEEE, Marrakech (2010)
5. Cingolani, P., Alcala-Fdez, J.: jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation. In: Proc. IEEE International Conference on Fuzzy Systems (Fuzz 2012), pp. 1–8. IEEE, Brisbane (2012)
6. Domshlak, C., Hüllermeier, E., Kaci, S., Prade, H.: Preferences in AI: An overview. Artificial Intelligence 175(7-8), 1037–1052 (2011)
7. Frenzel, C., Sanneck, H., Bauer, B.: Rational Policy System for Network Management. In: Proc. International Symposium on Integrated Network Management (IM 2013). IEEE, Ghent (2013)
8. Hämäläinen, S., Sanneck, H., Sartori, C.: LTE Self-organizing Networks (SON): Network Management Automation for Operational Efficiency. Wiley, Chichester (2011)
9. International Electronical Commission (IEC): IEC 1131 - Programmable Controllers: Part 7 - Fuzzy Control Programming (January 1997), `http://www.fuzzytech.com/binaries/ieccd1.pdf`
10. Kephart, J., Walsh, W.: An artificial intelligence perspective on autonomic computing policies. In: Proc. IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004), pp. 3–12. IEEE, Yorktown Heights (2004)
11. Kousaridas, A., Nguengang, G.: Deliverable D2.3: Final Report on Self-Management Artefacts. Tech. rep., Self-Management of Cognitive Future InterNET Elements, Self-NET (2010)
12. Kruse, R., Gebhardt, J., Klawonn, F.: Foundations of Fuzzy Systems. Wiley, New York (1994)
13. Lupu, E., Sloman, M.: Conflicts in policy-based distributed systems management. IEEE Transactions on Software Engineering 25(6), 852–869 (1999)
14. O'Hagan, M.: A Fuzzy Decision Maker, `http://www.fuzzysys.com/fdmtheor.pdf`
15. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice Hall, Upper Saddle River (2003)
16. Strassner, J.: Policy-Based Network Management: Solutions for the Next Generation. Morgan Kaufmann, Amsterdam (2004)
17. Zimmermann, H.J.: Fuzzy Programming and Linear Programming with Several Objective Functions. Fuzzy Sets and Systems 1(1), 45–55 (1978)