POLICY CONTROLLED AUTOMATED MANAGEMENT OF DISTRIBUTED AND EMBEDDED SERVICE SYSTEMS

Stefan Illner, Heiko Krumm and Andre Pohl
Department of Computer Science
University of Dortmund
August-Schmidt-Straße 12
44227 Dortmund, Germany
email: {illner, krumm, pohl}@ls4.cs.uni-dortmund.de

Ingo Lück, Darius Manka and Thomas Sparenberg
Materna Information & Communications
Voßkuhle 37
44141 Dortmund, Germany
email: {ilueck, dmanka, tsparenb}@materna.de

ABSTRACT
The management of distributed service systems is a complex task as changes in the system and the environment may induce reconfiguration tasks to be handled. In this paper, we deal with the automated reconfiguration of service-oriented, embedded systems. Depending on the environment such a system encounters, some of the services may need to be reconfigured depending on certain conditions like temperature or battery state. In our approach, the task of automatic adaptation to a changing environment is divided into two parts: At design time, various configurations are generated for a service and are mapped to specific environmental conditions. For this we adopt the approach of *model-based management* and *GRBAC* to ease the creation of complex management policies. At runtime, a reconfiguration service gets aware of changes in the environment, selects the appropriate configurations for the services it is responsible for, and enforces the new configurations.

KEYWORDS
Model-based Management, Automation, GRBAC, Abstract Modeling

1 Introduction

The work presented in this paper is part of the ITEA SIRENA project [1]. The goal of SIRENA is to develop a universal service infrastructure for realtime, networked and embedded devices, which can be utilized in the industrial, home, telecommunication and automotive domain.

Our work focuses on the automated adaptation and reconfiguration of services in this service infrastructure. As embedded systems are addressed, the time spent for necessary configuration changes on the devices should be low, so no expensive computations can be done to accomplish this. At design time a model of the managed system is generated and configurations are created. At runtime configuration changes confine to selecting and activating a suitable configuration. *Policies* tie together *Profiles* describing environmental conditions and corresponding *Service Configuration Sets*. The enforcement of these policies is done by a separate management process at system runtime.

The design time task, the creation of appropriate environment related service configurations, is supported by a graphical modeling tool which allows the automated derivation of low-level configurations from high-level policies.

In section 2, we present the components that are needed for modeling a service system using our offline modeling tool. The transformation from an abstract, high-level policy to a set of suitable service configurations is also subject of this section. In section 3, we depict how policies are enforced at runtime in reaction to environment changes. An example scenario is outlined in section 4. Section 5 outlines related work regarding automated management and adaptation, and finally section 6 concludes this paper.

2 Abstract Modeling

In the SIRENA project, we assume a service-oriented architecture, where distributed services are running on loosely coupled, embedded computing resources. In this environment there will be services that can be configured in various ways, and the current configuration will affect the services' behavior. The creation of management policies and low-level configurations for such distributed systems is a complex and difficult task and we apply abstract modeling techniques to ease this process.

2.1 Model-based Management

The model-based management approach facilitates an object-oriented model of the managed system to ease the creation and modeling of complex management policies. The graph-based modeling of such a system is supported by a graphical modeling tool. Common policy-based management approaches [2] apply low level policies to describe management demands. Systems created using the model-based approach [3] use much more abstract high-level policies to describe the desired behavior, from which concrete service configurations are automatically created. The model is divided into three horizontal and three vertical layers as depicted in Fig. 1. The elements of the model

are expressed by different types of nodes connected by different types of edges which state dependencies and associations between the model-nodes.

The horizontal segmentation divides the model into three different abstraction layers. The topmost *Roles & Objects*-layer contains a very abstract, business oriented view of the system. On this layer abstract access permissions are modeled using the RBAC approach: *Roles* users are acting in, *Objects* which are accessed, and *Access-Modes* which are used to model a particular access type for a given *Object*. Downwards, the *Subjects & Resources*-layer contains a more extended and less abstract view of the system. This includes individual *Services*, *Service Dependencies*, and *Resources* accessed by *Services*. In addition real people acting in specific *Roles* related to derived *Service Permissions* are depicted here. The *Processes & Hosts*-layer at the bottom deals with the low-level, technical computing and communication parts of the system running the implementing processes for the service modeled at the *Subjects & Resources*-layer. This includes server processes providing the modeled high-level services, communication links between hosts, and security policy enforcement equipment such as firewalls or VPN links, for example.

The vertical diversion groups the model into three different parts: *Productive System*, *Control* and *Policy*. The *Productive System* section of the model contains the abstract *Objects*, *Services* and *Hosts*. The *Policy* column contains the security policies that should apply to the *Productive System* section. The *Control* column contains *Objects* that are required to enforce a given policy. The *Control* section is left empty on the topmost *Roles & Objects*-layer because on this level the security enforcing control services are not visible.

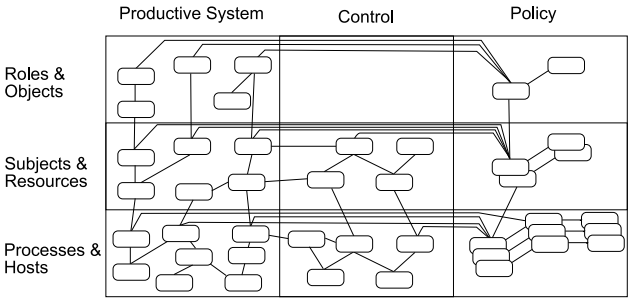

Figure 1. Model Overview

Based on this comprehensive model, our tool is able to automatically create low-level policies from the modeled high-level policies. The *Control* section has to be filled with appropriate enforcement services. The model elements are developed and integrated into so called *metamodels* which comprise system domain specific modeling objects.

2.2 RBAC & GRBAC

This section introduces the concepts of *Role Based Access Control* and *Generalized Role Based Access Control*. We apply these approaches in their original notion for the abstract modeling of access permissions. In addition we use the GRBAC model to facilitate the modeling of environment-aware management policies as these are required for our automatic reconfiguration approach.

The *RBAC* [4] approach uses roles instead of a per user access control policy to model access permissions. These roles reflect the structure of an organization and carry specific access rights. Users or applications acting on behalf of a user are referred to as subjects in the RBAC terminology. Subjects are assigned a set of authorized roles, i.e. each subject can only act in a role that is defined in this set. In addition the RBAC model defines so called objects and operations. Objects are the entities protected by an RBAC system, e.g. services or files. Operations describe sequences of accesses of subjects acting in specific roles to RBAC objects. A formal definition of the RBAC model follows:

Subject S	a user representation in the system
Role R	a categorization primitive for subjects
Object O	a system resource
Operation T	a sequence of one or more accesses to one or more objects
AR(S)	the authorized role set for subject S
AT(R)	the authorized operation set for Role R
exec(S, T)	true iff subject S is authorized to execute operation T; true iff $\exists\, role\ R : R \in AR(S), T \in AT(R)$

Based on the active set of subject roles, the operation and the associated object a system based on the RBAC model decides whether or not the access should be granted.

The *Generalized Role Based Access Control* model [5] extends the RBAC model outlined above by adding two more types of roles: *object-* and *environment-roles*. Object-roles are intended to model the type or internal state of an RBAC object. This can be a role that describes if an object is able to handle some kind of specific data. When considering data as an object itself this type of role may describe the type of information provided. Environment-roles are used to model the environment of an application. This can be the time of day or the weight of the subject trying to access a specific object. Due to the addition of two more roles, the decision process to determine whether or not an operation should be allowed is more complicated. This process is as follows:

1. It is checked, if there exists an object-role R_O of object O, and
2. an environment-role R_E which is active at the moment, and
3. an operation T which is allowed for a subject-role R_S to access an object O in the role R_O while the role R_E of the environment is active.

2.3 Policy Modeling

Our model adopts the model-based management approach and is extended by the use of GRBAC's environment representation. On the highest layer we adopted the elements *Object*, *Access-Mode*, *Subject-Role* and *Permission* to model access control considerations. Moreover we include so called *Profiles* to describe the environment and abstract *Requirements* to model security and management demands regarding *Objects* and *Access-Modes*.

The notion of *Objects* describes abstract entities which are accessed using a specific *Access-Mode*. Such an *Object* could be Internet for example. The *Access-Mode* outlines the type of access to an *Object* more precisely. *Subject-Roles* are used to express that a specific group of users is allowed to execute a modeled *Access-Mode*. A *Permission* element is used to connect all these model-elements to model a specific part of an access policy.

To make such a model environment-aware, we introduce *Profiles* which are logical expressions over the activation state of environment roles. Each profile expresses a specific environmental state. The *Profiles* are graphically modeled by connecting logical operator nodes and environment roles using directed edges to form the desired logical formula (cf. Fig. 2).

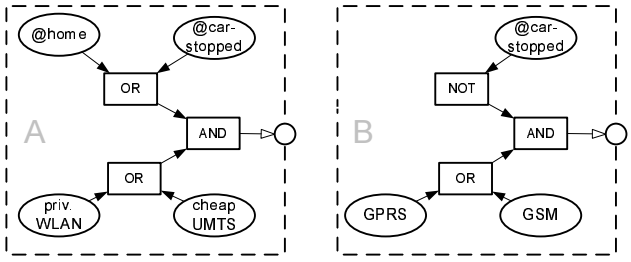

Figure 2. Profile Definition Model

The figure depicts two different *Profiles*, *A* and *B*. A is active when the *@home* or the *@car-stopped* and one of *private WLAN* or *cheap UMTS* roles are active. B defines an environment where the *car is not stopped* and only small bandwidth connection types like *GPRS* or *GSM* are available. All role elements can only have outgoing edges pointing at logical operators or to the border of the profile. The latter is allowed if only a single role is used to model a *Profile*. The logical operators can be nested and combined to be able to express arbitrary logical formulas. The top-level logical operator, the *AND* in the example, has an edge pointing at the *Profile's* border to state that the value of this operator defines the boolean property of the given *Profile*.

Such a *Profile* is connected to the *Permission* element to model an environmental dependency of this policy part. To complete the high-level model description, management and security requirements have to be modeled. For this purpose, abstract *Requirement* elements are introduced, which allow to design demands on management and security issues (cf. Fig. 3).

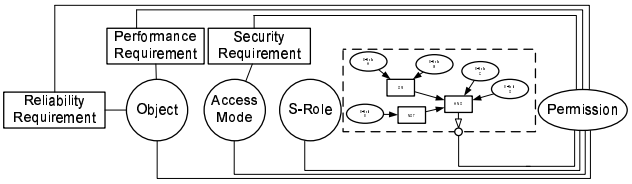

Figure 3. High Level Model

The *Requirements* used in our model are centered around the FCAPS framework [6], standardized by ISO. Its management strategies are categorized into five major functional areas: Fault, Configuration, Accounting, Performance, and Security Management. Fault management encompasses the activities of detection, isolation, recording, and correction of abnormal system behavior. Configuration management activities include the configuration, maintenance, and updating of managed components. Accounting management relates to administrative functions and user management. It comprises tasks such as defining costs for services and charging users for resource usage. Performance management includes mechanisms to recognize current or impending performance issues and activities, enabling analysis and maintenance of acceptable performance. Security management deals with the management of security services for purposes like access and traffic logging, authentication and authorization, certificate and trust management, privacy and confidentiality enforcement.

To give some examples, general abstract requirements could be *robustness* and *self-healing* with respect to failure recovery, implying the provision of fault tolerant services (e.g. automated failover). In the field of configuration management, *self-configuration* as well as *self-adaptation* are in strong demand, both comprising automated deployment, discovery and dynamic reconfiguration of services at runtime. Furthermore, in order to track user access to resources and to determine usage for which the users may be billed, aspects such as *cost control* and *chargeability* are necessary. Additionally, *performance control* requirements are mandatory. Critical *QoS* parameters (typically obliged in so called service level agreements) like availability, accuracy, and reliability must be considered. Common security requirements may include *confidential communication*, *communication integrity* and *non-repudiation* to ensure the responsibility of users for their actions. Requirements can be handled and designed using different levels of abstraction. On the highest level, they can be expressed in terms of general and relatively informal parameters, such as *video-streaming-performance* or *secure communication*. By means of the refinement process, concrete requirements can be derived from the abstract ones on the *Subjects & Resources*-layer.

A policy definition including the access control policy and the management and security requirements does not have to be connected to a profile. In this case the given

policy is considered to be the *Default Policy* which in fact means, that this policy should always be active when no environment-aware policy can be applied. Another consideration is, that the default policy prescribes ranges for configuration parameters in which the service is allowed to operate. This may anticipate unmeant or deliberate misconfiguration of services and ensures service availability.

2.4 Policy Transformation

The major high-level model elements of the model-based management are *Objects*, *Access-Modes*, *Subject-Roles*, and *Permissions*. In the *Subjects & Resources*-layer objects are refined to so called *Abstract-Resources* which are a representation for a less abstract view of the resource the high-level *Object* corresponds to. *Access-Modes* are refined to *Services* and *Service-Dependencies* providing a certain *Access-Mode*. These are associated with the corresponding *Abstract-Resource* elements. The *Subject-Roles* are related to real *User* objects and so called *Subject-Types* which are used to restrict the role types available to specific users in different situations. The high-level *Permission* elements are refined to *Service-Permissions* and relate the *User*, *Subject-Type*, *Service* and *Abstract-Resource* elements. On the *Processes & Hosts*-layer the *Abstract-Resource* elements are concretized to real resources, e.g. HTML files or pictures. The *Service* nodes are mapped to *Processes* providing the type of service, e.g. an HTTP-service would be mapped to an implementing Apache-Webserver process. *Users* and *Subject-Types* are merged to specific *User-Credentials*, e.g. the user Dave gets one credential for the Intranet access when he is at his office and another one when he's at home. At last the *Service-Permissions* are refined to so called *Protocol-Permissions* which relate the model elements on this layer with each other to express the definite access permissions. This low-level model is used to automatically create service configurations for certain services.

The *Profiles* introduced in section 2.3 are not refined. The high-level expression describing the environmental state is kept the same over the three abstraction layers and is a fundamental part of the configuration for the management infrastructure services. The abstract *Requirements* are refined by relating them with more tangible objects on the *Subjects & Resources*-layer. For instance, general performance demands like *Video-Streaming-Performance* can be easily refined into more detailed network and resource requirements such as *bandwidth consumption*, *response time*, *latency*, *packet loss rate* and *CPU utilization*. These nodes are connected to the high-level *Profile* and the service objects modeled on this layer. On the bottom layer these requirements are directly associated with the process, host and network infrastructure elements such as routers or simple network interfaces.

The complete policy for a modeled service system is expressed by a so called *Policy Set*. Such a set contains an arbitrary number of environment-aware policies and is automatically created from the information available on the *Processes & Hosts*-layer. The structure of a *Policy* can be seen in Fig. 4. A single policy consists of a profile and a configuration set, which is a set of individual service configurations. Each configuration applies to exactly one service, and contains the settings which control the whole service, or a certain aspect of it.

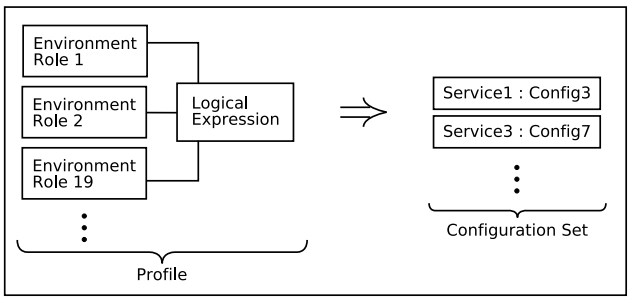

Figure 4. Structure of a policy

As long as the settings contained in the given configuration set do not conflict, it is also possible to include more than one configuration for a certain service in a configuration set. This is especially helpful if the configuration settings can be divided into disjoint sections, so that each section can be configured individually. A configuration set will only consist of configurations for those services or service sections which control the behavior that needs to be reconfigured.

For the purpose of policy enforcement, a bunch of infrastructure services is provided. They comprise a range of functionalities employed for controlling, management and administration of devices and services and for the enforcement of the policy-based configuration decisions.

Our approach primarily concentrates on aspects of fault, configuration and security management. In particular we focus on solving the problem of how to automatically configure the services and devices with respect to changing environments. Figure 5 illustrates the services used to enforce the specified management policies and their interaction.

3 Policy Enforcement

In the fault management area, the sub-services *Poll Monitor* and *Trap Monitor* are responsible for polling of the devices, querying managed objects to obtain their states, catching error events and monitoring the environment for changes. The captured events can be dispatched to the *Fault Management* service, where further actions, such as aggregation, filtering, notification, etc., can be performed.

The major part of the adaptation scheme is in fact realized by the configuration management services. In particular, the *Automated Adaptation and Dynamic Reconfiguration* service (AADR) is the most fundamental service in our service architecture. The following procedure clarifies

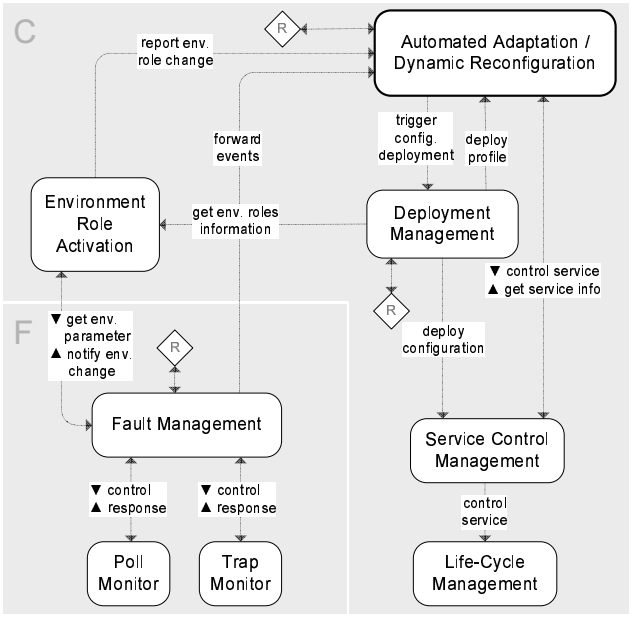

Figure 5. Management Services

the adaptation process: The *Environment Role Activation* service gets notified of environment changes by the fault management and maps these properties to appropriate environment roles. If such a change event results in an environment role activation state change, the AADR service is informed of the affected role and its new activation state. At that point, the AADR has to determine the set of currently active profiles.

In case that several profiles may be active at the same time, this is detected at design time by our tool which performs this check amongst others to ensure that the generated model does not contain logical errors. The user can then decide to make the corresponding profiles disjoint by altering the logical expressions that describe the profiles, or by introducing a precedence level for every profile to determine the order in which the profiles are checked for activation. The configuration set in that policy whose profile is found to be activated at first will then be the one to use.

After determining the current active profiles, the configuration deployment is triggered by the AADR and carried out by the *Deployment Management* service. There are two options for the operation of this service. On the one hand, it can be a real service, that manages the configurations for the services it is responsible for. When one of these services is to be reconfigured, the necessary configuration settings are deployed using a standardized configuration submission action. The advantage of this approach is that the device does not need to store all configurations itself, which is good for small devices with limited or no nonvolatile memory. It has to be considered, though, that it takes time to transmit the configuration settings over a network, so this is usually not an option in cases where the real-time constraints are of great importance. On the other hand, the appropriate configuration sets can be completely deployed to devices with sufficient memory. In this case, only a short message must be sent over the network by the *Service Control Management* service, which tells the device to activate a specific configuration set. The *Life-Cycle Management* service can be used to control the service runtime status including starting or stopping the service, deploying software updates, etc.

4 Example Scenario

The scenario described herein spans the home and automotive domain. The home domain contains a SIRENA service infrastructure including common security and management services, multimedia devices like a video recorder (VCR), and a home gateway which enables the access to home devices via Internet. This home gateway provides an EPG service (Electronic Programme Guide) which offers the television programmes, pictures from the movies and movie trailers. Network techniques available in home domain are LAN, WLAN and Bluetooth. The provided services can be used by applications which are able to communicate with SIRENA services. Such applications may run on PCs or PDAs. Moreover they could provide a web-based interface which may be accessible from arbitrary web browsers. This example uses a PDA as remote control to the VCR at home.

The automotive domain is represented by a car which includes a service platform offering services like Internet access, car sensors, GPS or navigation services. Moreover the common security and management services are available. All car located services are made available through a WLAN or Bluetooth network inside the car. A network connection to the outside world is established via GSM, GPRS or WLAN.

So far our scenario deals with two devices which are exposed to changing environmental conditions: The mobile PDA carried by a user, and the car, respectively the installed service platform. At home the connection between the control application and the services is established directly via LAN, WLAN or Bluetooth. But what if the PDA is used outside the home domain, e.g. in the car, in an Internet café or at an airport? There we have a situation that differs completely from the home use case. The EPG/VCR services cannot be connected directly, so reconfiguration of network components has to take place. In particular, the security requirements are also different, as the connection to the VCR is no more direct but established via unknown routes using public Internet backbones. Therefore the PDA located reconfiguration process includes the management of connection properties, as the Internet access service of the car platform has to be determined to be suitable for a connection to the VCR. Moreover, the security modules on the PDA have to be instructed to encrypt and digitally sign the request created by the PDA control application.

The car's Internet access service also needs reconfiguration as available and preferred Internet access con-

nection types may differ. If the car is near the home and the own, home located WLAN is available, the connection type should be switched to use this existing Internet access. En route public WLANs such as WLAN hot spot services around gas stations may be available. In other cases GSM or GPRS uplinks should be used to establish a connection to the Internet.

Different connection types induce different costs and quality, e.g. available bandwidth and latency, or packet-loss rate. Therefore the type of information transferred across these leased connections has to be adjusted. If there is a cheap and fast connection, the PDA may connect to the EPG service to list movies, view preview pictures and get some movie trailers. If the PDA is connected via GSM or GPRS, every minute of established connection or every byte leads to additional costs. Moreover, the available bandwidth is much more limited compared to WLANs. In such environments a configuration is useful where only a small amount of data will be transferred, e.g. only a list of movies with short textual introductions, neglecting pictures or trailers.

The applied configuration changes are transparent to the user. All changes, the establishment of a connection to the cheapest available network, the disconnection if a more suitable network is available, encryption, signature, are done automatically – no user interaction is required.

5 Related Work

Chisel [7] proposes a context-aware, policy-driven framework for adaptation of services. In this approach, the services will be adapted to use different behaviors, driven by a human-readable declarative adaptation policy script. Furthermore, the chisel framework allows to make mobile-aware dynamic changes to the behavior of various middleware-services, and it provides the addition of new unanticipated behaviors at run-time, without changing the middleware or the application using it.

Similarly, Lymberopoulos et al. [8] investigate dynamic adaptation of policies in response to changes within the managed environment. Here, policy adaptation includes both dynamically changing policy parameters and reconfiguring the policy objects. The proposed framework is primarily used to provide dynamic management of services in Differentiated Services (DiffServ) networks.

6 Conclusion and Future Work

In this work, we have shown how services in distributed and embedded systems can be managed automatically. This goal is achieved by a two-phase-process, which separates modeling, analysis and configuration generation at design-time and configuration activation at runtime using special management services. The benefit of this approach is that the complex, time-consuming task of modeling and configuration generation is done only once, before the system is set up. At runtime, only the detection of a suitable profile and activation of the corresponding configurations takes place, which is a task that can be accomplished also by embedded devices, and under soft real-time constraints.

Our future work consists of further research regarding dissolving of configuration conflicts during design time and the development of the modeling tool and required metamodels for specific application domains. The management services described in section 3 will be integrated into a comprising testing environment and will be used in demonstrators provided by the other SIRENA partners.

7 Acknowledgments

The work described herein was funded by the German Federal Ministry of Education and Research (BMBF) within the ITEA-SIRENA project (01ISC09G).

References

[1] SIRENA (Service Infrastructure for Real-time Embedded Networked Applications), `http://www.sirena-itea.org`, 2004

[2] M. Sloman, Policy Driven Management for Distributed Systems, *Journal of Network and Systems Management*, 2(4), 1994.

[3] Ingo Lück, Sebastian Vögel and Heiko Krumm, Model-based configuration of VPNs, *8th IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Florence, Italy, 2002, 589–602.

[4] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein and Charles E. Youman, Role-Based Access Control Models, *IEEE Computer*, 29(2), 1996, 38–47.

[5] Matthew J. Moyer and Mustaque Ahamad, Generalized Role-Based Access Control, *Proc. 21st Int. Conf. on Distributed Computing Systems, Mesa, USA, 2001, 391–398*.

[6] H.-G. Hegering, S. Abeck and B. Neumair, Integrated Management of Networked Systems - Concepts, Architectures and their Operational Application, *Morgan Kaufmann Publishers*, ISBN 1-55860-571-1, 651 p., Januar, 1999.

[7] J. Keeney and V. Cahill, Chisel: A Policy-Driven, Context-Aware, Dynamic Adaptation Framework, *Fourth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2003)*, Lake Como, Italy, 2003.

[8] L. Lymberopoulos, E. Lupu and M. Sloman, An Adaptive Policy-Based Framework for Network Services Management, *Journal of Network and Systems Management*, 11(27), 2003.