Policy-based self-management of industrial service systems

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

Ingo Lück, Darius Manka, and Franz-Josef Stewing
Materna GmbH
Voßkuhle 37
44141 Dortmund, GERMANY
{ingo.lueck, darius.manka ,franz-josef.stewing} @materna.de

***Abstract*—Service-orientation supports the construction of flexible and comprehensive industrial applications. The growing scale and complexity of the applications, however, demand for enhanced self-management functions providing efficient self-adaptation and repair mechanisms. We propose the approach of policy-controlled self-management which has been developed and successfully tested in the context of Web Service based control applications. We use hierarchically structured management policies where high-level policies serve as abstract definitions of management objectives and low-level policies represent concrete rules for resource monitoring und correcting interventions. The definition, analysis, refinement and deployment of the policies are supported by an interactive graphical modeling tool.**

Index Terms—model-based management, fault tolerant systems, web services

I. INTRODUCTION

Currently service-orientation is expanding into the field of industrial automation in order to support flexible, dynamically adaptable systems which – particularly when employing open and standardized communication mechanisms like Web Services – facilitate the mutual integration of factory automation and enterprise information systems [Jam05, Jam05a]. Contemporaneously, the growing scale and heterogeneity of networked IT-systems lead to an increasing complexity of the tasks of technical system administration and management which have to provide for the continuous, accessible and user transparent operation. Therefore there is a strong demand for self-managing systems which e.g. led to IBM's initiative of Autonomic Computing [Gan03], the efforts of which focus on the integration of self-management techniques for the automated configuration, fault correction, optimization and protection of comprehensive networked systems.

The term of technical management comprises all the technical tasks which are necessary for the proper operation of IT-systems in addition to software development and distribution, i.e. particularly component deployment, initial setting-up, hard- and software configuration, logging, auditing, monitoring, short- and long term adaptation, alert handling, fault detection, diagnosis, proactive maintenance, repair and reconfiguration. According to the ISO/OSI framework, technical management concerns the five functional areas of fault, configuration, accounting/administration, performance and security management (cf. e.g. [Heg99]).

As automation in general, management automation also requires definitions of control objectives and control algorithms. Both purposes can suitably be served by the approach of policy-based management [Slo94] which is well-known in the field of communication network management. The so-called management policies describe the relevant preferences, objectives and rules which govern the execution of management operations. For automation, particularly policy hierarchies can be employed which support different levels of abstraction [Mof93]. So, high-level policies can act as abstract definitions of management objectives and low-level policies can represent concrete rules for resource monitoring und correcting interventions.

Our approach of model-based management is an enhancement of policy-based management and policy hierarchies. Moreover it relies on a tool and a hierarchical system model. Tool operation and system modeling are performed at design-time. The tool supports the system modeling, the high-level policy acquisition and their analysis. Due to the information of the system model, it automatically refines the policies and derives corresponding correct, consistent and automatically enforceable low-level policies [Lue02] which reflect all possible system conditions and define the configuration of an efficient and lightweight automated runtime management system.

In the last two years we participated in the SIRENA project [Jam05b] in the course of which a comprehensive Service Infrastructure for Real time Embedded Networked Applications has been developed and successfully demonstrated in industrial, automotive and home environments. Moreover we studied the application of Web service based systems in industrial environments by means of a distributed conveying system control application [Bri06]. In each case our work concentrated on the provision of automated technical management and was based on an extension and specialization of the model-based management approach. Several aspects of that work have already been published.

In particular, [Ill04] reports on the automated adaptation of security service configurations to changing environment conditions. [Ilp05] expands the scope to general technical management and presents the overall context of modeling, tool and distributed management systems. [Ilk05] enters into the tool-based hierarchical system and policy modeling which is exemplified by means of an automotive scenario.

In this paper, we focus on the principles of those policy elements which define objectives and rules of self-management functions for industrial service systems. We present an adequate policy hierarchy, outline the principles of suitable system architectures, and discuss the employment of the policies for the automated control of the run-time management functions.

In the sequel a short introduction to policy-based management is given. It is followed by an outline of the model-based management approach. Since, besides of proper functionality, reliability is a central requirement of industrial systems, principles of fault tolerant computing are of relevance and outlined in the next section. The following three sections describe the developed policy definition and representation in the three hierarchy layers of the approach which provide an abstract logical view, a service view, and a device view. Concluding remarks present a summary of the contributions and moreover address application experiences.

II. Policy-based Technical Management

In the last two decades promising basic approaches for automated management centering around management policy definitions have particularly been developed in the context of technical network management [Ver02, Slo94]. Policy-based systems separate the policy from the implementation of a system and therefore permit the policy to be modified without changing the systems underlying implementation. Different types of policies (e.g., user policies, provider policies, resource policies, security policies, accounting policies) reflect the relevant objectives of operation and serve as prescriptions of automated system control. Meanwhile a series of policy management standards is emerging (e.g. [Box04, Cim03, Wes01]) and we also anticipate policy extensions for the Web Services Distributed Management (WSDM) standardization [Oas05].

The architecture of the policy-based management system is affected by the presence of policy enforcement units, each near to a corresponding managed resource [Wie94]. The defined policies are distributed and deployed to the units in form of dedicated descriptions. Policies to be supplied to and interpreted by resource-near enforcement units essentially express management actions in a low-level, resource-oriented and efficiently executable form, mostly following the "if *condition* then *action*" metaphor (e.g. [Lob99]). Advantages of more abstract and system-integral policy descriptions, however, have been recognized early. Particularly the approach of policy hierarchies proposes the employment of hierarchical abstraction layers supporting the stepwise refinement of policies [Mof93, Wie94]. Due to the increasing details and resource dependence of low-level policies, however, the refinements cannot be computed mechanically without additional means. Further important design problems of management policies concern the global consistency of those policies which are composed from different parts. The parts can reflect certain but not necessarily orthogonal management aspects or are oriented at certain resources. Therefore, especially in complex and heterogeneous networked systems, global policies tend to contain incompatible or contradictious elements (see e.g. [Kem05]).

III. Model-based Management

The approach of model-based management (MBM) considers that each automated management system has to be tailored to its given managed system, since it has to consider its special purpose and operational requirements. In consequence the cost reduction of management automation is paid by high costs needed for the design and implementation of the automated management system. MBM therefore shifts the focus (and the major efforts) of management system development from the management application design to the identification of the managed system and the management objectives.

The identification is performed by modeling. By means of a three-layered model, the managed system is represented with respect to architecture and configuration, services provided, and management objectives. High-level management policies express the abstract management objectives. After modeling of the managed system and the high-level management policies, the model is extended by models of the components of the management system. The low-level management policies which enforce the high level policies in correspondingly governing the functions of management system are derived from the model as well as

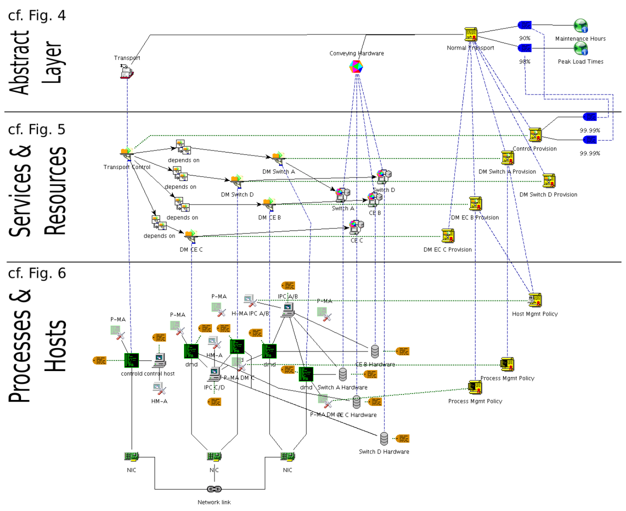

Fig. 1. Example of a layered model

the adequate management system configuration.

The modeling is supported by an interactive graphical tool and by meta-model class libraries. A model has the form of an object instance diagram (as known from UML) and is handled by its graphical diagram representation. The extension of models with the management system components and the derivation of management policy refinements moreover are supported by graph rewriting rule libraries. Furthermore backend functions of the tool achieve the generation of configuration data files.

The three-layered modeling (cf. Fig. 1) of managed system, management system, and management policies particularly supports the design of adequate and well-understood management policies since one can inspect, discuss, and check the policies on three different levels of abstraction. The automated derivation of the middle layer policies and of the lowest layer ones supports policy refinements which guarantee that the abstract high-level policies are correctly and completely enforced by the concrete management system.

IV. Fault Tolerant Service Systems

Though using the most reliable hardware and software products for an IT system, various minor or major faults may occur. Depending on the importance of the affected subsystems fast and solid fault management has to take place to ensure the proper overall operation of the complete system. The design and operation of fault tolerant systems is a very complex task. The manual location and repair of faulty soft- or hardware components may be to slow and time consuming and not applicable in commercial and industrial IT environments. Moreover each minute that the system is not properly operational reduces the overall availability of the system and thus may violate negotiated service level agreements. For this reason the installation of fault tolerance mechanisms is mandatory. Fault tolerant systems rely on some basic mechanisms that are able to deal with different types of fault situations.

Fault detection recognizes and identifies the occurring faults and is based on the instrumentation of the system with sensors and test functions. The following *fault repair* is mainly based on *redundancy*. There are several types of redundancy that address hard- and software fault handling. The notion of redundancy makes a distinction between *structural*, *functional, informational* and *time* redundancy. Moreover there is a distinction concerning the activation of redundancy: *static redundancy* states that the redundant components are operational all the time whilst *dynamic redundancy* defines that redundant components only are activated in case of an error (cf.. [Ech90]).

In case of a fault a redundant system is able to react in different ways. Using *error passivation* the system is able to remedy erroneous subsystems by reconfiguration of the erroneous or surrounding system parts, by elimination of faulty and insertion of replacement components or by evacuation of parts to not affected areas of the system. To gain a proper system state, *error recovery* mechanisms like forward- and backward error recovery can be used. To be able to use *backward error recovery* the fault tolerant system has to collect a history of proper system states to which the system may be reset in the case of a fault. The *forward error recovery* does not require any history, as it just sets the system to a newly computed proper system state. Finally, *error compensation* can be used to ensure a correct result of a given request by using *fault masking* or *error correction* mechanisms. Fault masking can be achieved by using multiple implementations for the processing of the same requests. After termination of request processing, a majority decision finally determines the result to be considered correct. Error correction uses informational redundancy to compute a correct value from the erroneous value plus additional error correction information accompanied with this value.

Additionally to decrease the MTTR (meantime to repair) the concepts of *Recovery Oriented Computing* [Roc02] may be applied.

To be able to create fault tolerant service systems the employed services have to implement a defined interface that enables the runtime monitoring, controlling and resetting of the service. In some cases transaction-oriented interservice communication schemes are of interest in order to support efficient backward recovery mechanisms. Since the system state of industrial applications, however, is often connected with physical state components of the technical system, in many cases generic backward recovery mechanisms are not appropriate and adequate service-specific forward recovery operations have to be introduced during service design.

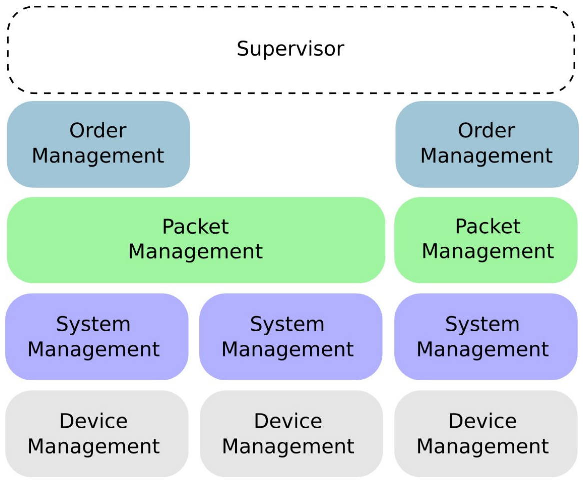

Fig. 2. Service layers

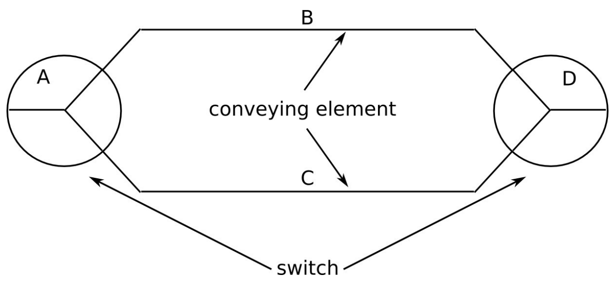

Fig. 3. Example conveying system

V. Abstract Policies for Industrial Service-Oriented Systems

The policy- and model-based management of reliable industrial systems shall be exemplified by means of a service oriented industrial control system that was developed by a group of students in Dortmund [Bri06]. The system controls a conveying system and can flexibly adapt to changing conveying system states. Basically it applies Web Service and UPnP technology in order to achieve modularity, flexible interactions and automatic discovery.

The hierarchical architecture of the distributed control system (see Fig. 2) has four layers:

- On the lowest layer, a series of device managers (DMs) control the different components of the conveying elements. Each DM offers services for the basic control of a corresponding technical device (e.g. switch motor on, set speed frequency). Additionally an event interface is offered to subscribe to low level hardware events (e.g. the activation of light-barrier sensors, temperature warning of a transport motor).
- On the second layer, for each conveying element one system manager (SM) is instantiated which controls the operation of the conveying element. SMs use the services of the DMs and provide services which correspond to the function of a conveying element as a whole (e.g. set switch direction).
- On the third and fourth layer the *package managers (PMs)* and the *order managers (OMs)* are located. A PM is responsible for the correct routing of packets through the system and is only created for routing decision points, e.g. switches. The OMs are only instantiated at in- or outports where orders can be injected or removed from the conveying system.
- On the uppermost layer a set of *redundant supervisors* are instantiated that are responsible for the initialization, monitoring and control of the whole system. The supervisors represent the client applications of the OM and PM services. At startup one main supervisor is elected.

Fig. 3 shows a small section of the conveying system consisting of two parallel conveying elements each with a two way track switch at its ends. We assume that this subsystem is required to be 98% available all time during normal operating hours. During service intervals its availability can be reduced to 90%. In our approach these requirements are represented by the application-near high-level management policies.

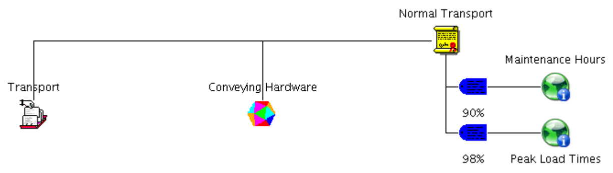

Fig. 4. Topmost model layer

The high-level management policies used in the topmost model layer solely show abstract management objectives and represent corresponding requirements:

- *Functional requirements* are represented by tuples of the form: <accessmode, object>. Such a tuple denotes that an abstract function access mode of the abstract object has to be implemented by the modeled system. The connection between the object and the access mode is established via a functional obligation element. The conception follows the well-known approach of role-based access control (RBAC) [San96].
- *Non-functional requirements* are represented similarly to service level agreements (SLAs) by additional numerical attributes to the functional obligations.

Each requirement can be extended to reflect *modalities* applying conceptions of the generalized role-base access control model (GRBAC) [Moy01], which additionally to RBAC's subject roles introduces environment and system roles representing states of the managed system's environment and of its components (e.g. roles "when power is low", "under high load").

Fig. 4 depicts the high level model of our application scenario. The system's conveying hardware is abstracted to a single object on this layer. The access mode *Transport* describes the abstract operation of this use case. Both elements are linked with the *functional obligation* element *Normal Transport* that defines the functional requirement for the system. Additionally the functional obligation is attributed by two different *availability requirements*, each defining abstract availability requirements during different periods of operation.

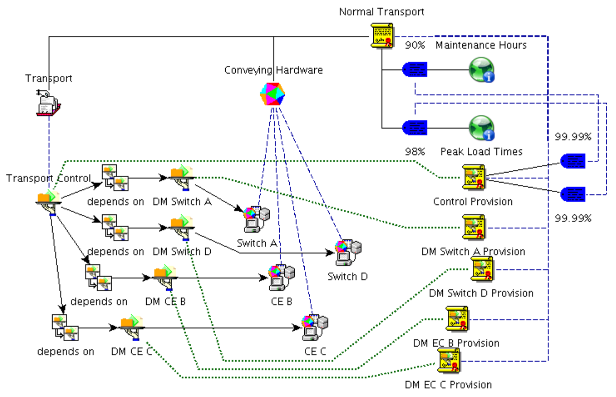

Fig. 5. Services & resources

VI. Service Policies

The middle model layer represents the managed system and management policies in a more implementation near, but still abstract service-oriented view. Here, each tuple of the topmost level is refined by a set of implementing service associations.

Fig. 5 shows the service policies layer. The *Transport* access mode is refined to a service that offers the required functionality. The abstract object subsuming the conveying hardware of the system is refined to abstract resources; one abstract resource for each real hardware element. As the control service cannot access the hardware directly it depends on device-services that provide functionality to control the devices. This dependency and the required services are depicted on the left side of the model. On the right side of the model one can see the automatically derived *service provision* obligations and their accompanied availability requirements. These elements are computed from the functional obligations, their associated availability requirements and the implementing services that provide the modeled access mode (additional availability requirements are hidden in the figure due to readability reasons). On this layer the availability requirements are still abstract.

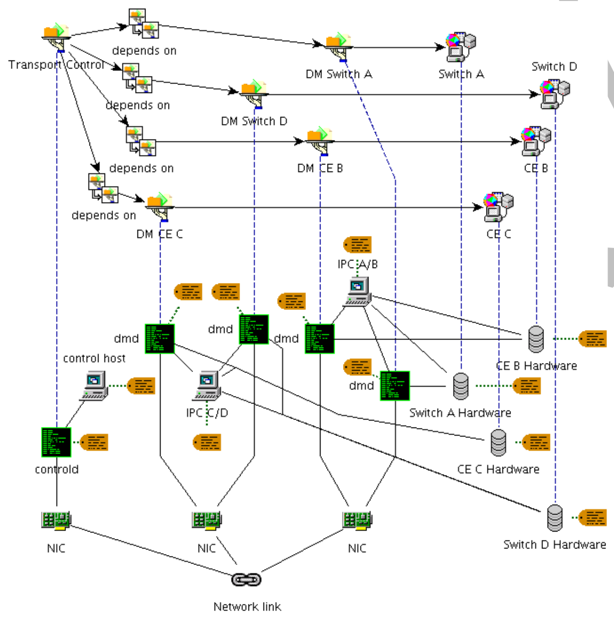

Fig. 6. Services, processes & hosts

VII. Device Policies and Policy Enforcement

The lowest model layer finally represents the concrete implementation architecture of the managed system. Client-subjects and services of the middle layer are refined by sets of implementing client- and server processes residing on networked devices. Accordingly, the managed system appears as a diagram of processes, physical resources, devices and network links. Moreover the lowest model layer is extended by the components of the concrete management system (e.g. watchdog processes, monitoring and control elements). To gain information about the availability of system components like hosts, processes or resources these elements have to be augmented by *availability assumptions.* These assumptions are given in the terms of the MTTF (meantime to failure), MTBF (meantime between failures) and MTTR (meantime to repair) of the associated component. The management policies of the lowest layer have the character of low-level management policies and directly correspond to the control parameters of the management system components (e.g. restart condition thresholds, monitoring periods).

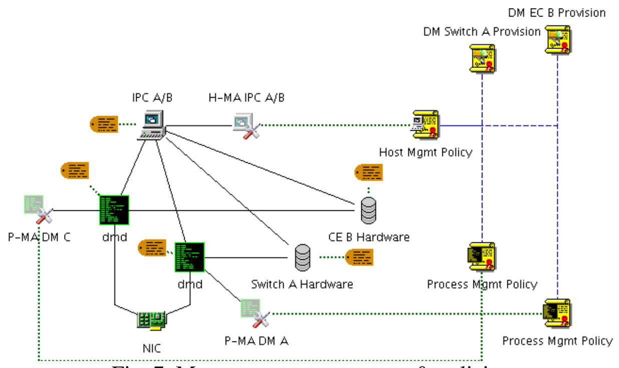

Fig. 7. Management components & policies

Fig. 6 depicts the low level model of the conveying system. The *Transport control* service is implemented by a process running on a dedicated host. Both process and host are associated to specific availability assumptions. The device-services are organized similarly despite the fact that one industrial pc (IPC) hosts two processes and is also connected to the conveying hardware to be controlled. The latter has also to be extended by the definition of a correlated availability assumption for the complete device (including all single parts like motors, sensors, etc).

The low level model introduced so far does not contain any additional policy or management elements. These elements are automatically determined from the policies defined on the middle layer and the availability assumptions defined on the lower layer. If the refinement process detects that the required availability is already provided by the system components, namely processes, hosts and resources, it would not add any additional management components or policy descriptions. Nevertheless if the system does not satisfy the requirements inherently the low-level system model is augmented by management components and low-level policy definitions. In the case that the required availability cannot be assured due to missing hardware redundancy or awkward service dependencies the refinement functions create a warning.

Fig. 7 exemplifies the extension of the model for IPC A/B that is responsible for the control of track switch A and conveying element B. To increase the overall availability level the device-service processes are observed by *process management agents* that are able to monitor and control (e.g. start, stop, configure) the process. Through the created management policies the agents have information about the normal operational behavior of the service processes and thus can restart the process if necessary. Additionally the IPC is monitored and controlled by a *host management agent* that monitors critical host parameters and acts on behalf of the related host management policy. These policies, either *host-* or *process management policy* are automatically created from the information provided in the low-level system model and the defined high-level policies on the upper two layers of the model. The derived low-level policy descriptions which are deployed on the host- and process management agents contain the definition for the behavior of the management agent and the observed process or host. Typical low-level policies of our example are:

- The response time of process requests is monitored. If the response time exceeds the threshold of 2 seconds, the process has to be restarted. This specific time span can be computed from the MTTR and MTBF values of the process and the service reliability requirements.
- The IPC onboard controller has to monitor the frequency of transmission errors of the data connection to the conveying element. If that frequency exceeds the threshold of 50, an IPC reset has to be issued.

Moreover, the low level policies reflect convenient forward recovery and compensation mechanisms for faults and failures of technical resources. For instance, a failure of conveying element B has to result in a disintegration of B and a corresponding dynamic adaptation of the packet routing mechanisms.

Consequently, the policy refinement process is not limited to the insertion of dedicated management components but furthermore includes the creation of additional (functional) redundancy by process replication, creation of process groups and the introduction of application-specific forward error recovery mechanisms.

VIII. Concluding remarks

The approach of policy-controlled self-management and its model-based and tool-assisted implementation have been outlined with emphasis on the hierarchical policy representation of high-level reliability requirements and corresponding low-level control policies for fault tolerance mechanisms. Most parts of the development have been performed in the course of the SIRENA [Sir04] project and during a comprehensive student project [Bir06] which both have been completed in March 2006. Currently we test our efficient Micro Java based implementation of the Web Services for Devices stack and plan the application of Micro Java devices as hosts for self-management components (particularly policy enforcement components). Current research investigates the relationships and dependencies between application service patterns, fault tolerance mechanisms and policy schemes in order to develop comprehensive policy definition and refinement procedures.

IX. Acknowledgment

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

References

[Box04] D. Box, F Curbera, M. Hondo et al., “Web Services Policy Framework (WS-Policy). Version 1.1” , http://www-128.ibm.com/developerworks/webservices/library/specification/ws-polfram/, Sept. 2004.

[Bri06] B. Brill et al., “Management of cooperating Web Services.” Report on the 1-year / 12-student project PG475 (in German), http://www.roastedkit.org, Fachbereich Informatik, Universität Dortmund, 2006.

[Cim03] DMTF, CIM Policy Model CIM Version 2.7 – White Paper, DMTF Inc., http://www.dmtf.org/standards/published_documents#whitepapers, Jun. 2003.

[Ech90] Klaus Echtle, *Fehlertoleranzverfahren*, Springer, ISBN-3-54052-680-3, 1990.

[Gan03] A. G. Ganek, T. A. Corbi, “The dawning of the autonomic computing era. “ *IBM Systems Journal*, Vol. 42, No. 1, 2003.

[Heg99] H.-G. Hegering, S. Abeck, and B. Neumair, “Integrated Management of Networked Systems”, *Morgan Kaufman*, 1999.

[Ilk05] S. Illner, H. Krumm, A. Pohl, I. Lück, D. Manka, and T. Sparenberg, “Policy Controlled Automated Management of Distributed and Embedded Service Systems.” *In Proc. IASTED Int. Conf. on Parallel and Distributed Computing and Networks (PDCN 2005)*, Innsbruck, pp. 710-715, 2005.

[Ill04] S. Illner, A. Pohl, H. Krumm, “Security Service Adaptation for Embedded Service Systems in Changing Environments.” *In Proc. 2nd IEEE Int. Conf. on Industrial Informatics (INDIN04)*, Berlin, Germany, IEEE Computer Society Press, 2004, pp. 457-462.

[Ilp05] S. Illner, A. Pohl, H. Krumm, I. Lück, D. Manka, Th. Sparenberg, “Automated Runtime Management of Embedded Service Systems Based on Design-Time Modeling and Model Transformation.” *In Proc. 3rd IEEE Int. Conf. on Industrial Informatics (INDIN05)*, Perth, Australia, IEEE Computer Society Press, Catalogue Number: 05EX1057C, Paper PD-001854, 2005.

[Jam05] François Jammes and Harm Smit, “Service-Oriented Paradigms in Industrial Automation.” *In IEEE Transactions on Industrial Informatics*, Vol. 1, No. 1, pp. 62-70, 2005.

[Jam05a] Jammes, F., Smit, H., Lastra, J.L.M., Delamer, I.M, “Orchestration of Service-oriented Manufacturing Processes.” *In Lo Bello, L & Sauter, T. (eds.). Proc. IEEE Int. Conf. on Emerging Technologies in Factory Automation (ETFA2005)*, Cernobio, Italy, pp. 617-624, 2005.

[Jam05b] F. Jammes, H. Smit, “Service-Oriented Architectures for Devices – the SIRENA View.” *In Proc. 3rd IEEE Int. Conf. on Industrial Informatics (INDIN 2005)*, Perth, Australia, IEEE Computer Society Press, Number: 05EX1057C, Paper PD-001865, 2005.

[Kem05] Bernhard Kempter and Vitalian A. Danciu, “Generic Policy Conflict Handling Using a priori Models.” *In J. Schönwälder and J. Serrat (Eds.): DSOM 2005*, Springer Verlag, LNCS 3775, pp. 84–96, 2005.

[Lob99] Lobo, Jorge, Bhatia, Randeep, Naqvi, Shamim, “A Policy Description Language.” *In Proc. 16th Nat. Conf. on Artificial Intelligence (AAAI-99)*, pages 291– 298, MIT Press, 1999.

[Lue02] I. Lück, S. Vögel, H. Krumm, "Model-Based Configuration of VPNs." *In 8th IEEE/IFIP Network Operations and Management Symposium (NOMS2002)*, pages 589-602, Florence, April 2002. IEEE Computer Society Press.

[Mof93] Moffet, Jonathan, Sloman, Morris, "Policy Hierarchies for Distributed Systems Management." *IEEE Journal of Selected Areas in Communications*, 11,9, 1993.

[Moy01] Matthew J. Moyer and Mustaque Ahamad, "Generalized Role-Based Access Control." *Proc. 21st Int. Conf. on Distributed Computing Systems*, Mesa, USA, pp. 391–398, 2001.

[Oas05] Oasis, "An Introduction to WSDM." Committee Draft 1, Sep. 2005, http://www.oasis-open.org/committees/download.php/14351/cd-wsdm-introduction_v3.doc

[Roc02] Patterson et al, "Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies." *UC Berkeley Computer Science Technical Report UCB//CSD-02-1175*, March 15, 2002

[San96] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, "Role-based access control models." *IEEE Computer* 29 (2), pp. 38-47, 1996.

[Sir04] SIRENA (Service Infrastructure for Real-time Embedded Networked Applications), http://www.sirena-itea.org, 2004

[Slo94] Sloman, Morris, "Policy Driven Management for Distributed Systems." *Journal of Network and Systems Management*, 2(4), p. 333-360, 1994.

[Ver02] Verma, D., "Simplifying Network Administration using Policy based Management." *IEEE Network*, March 2002.

[Wes01] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, S. Waldbusser, "Terminology for Policy-Based Management." *Request for Comments 3198*, Internet Engineering Task Force, November 2001.

[Wie94] Wies, Rene, "Policies in Network and Systems Management – Formal Definition and Architecture." *Journal of Network and Systems Management*, Plenum Publishing Corp., 2(1), pp. 63-83, 1994.