Document-Based Network and System Management

Utilizing Autonomic Capabilities for Enterprise Management Integration

Edzard Höfig and Peter H. Deussen
Autonomic Systems Engineering Working Group
Fraunhofer Institute for Open Communication Systems
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
{edzard.hoefig peter.deussen}@fokus.fraunhofer.de

ABSTRACT

Document-Based Integrated Management (DBIM) is a novel way of consolidating enterprise network and system management. We present an approach to DBIM that utilizes the autonomic capabilities of a set of distributed Document Processing Units (DPU), which constitute a document processing plane within the enterprise network. Our approach is explained in detail using a scenario from the performance and fault management domain. We also explore the connection of DBIM with current research topics in the autonomic systems field, and the relationships to other popular integrated management approaches, like Policy-Based Management (PBM), Active Networks, Agent-based Network Management, or Web Service (WS) Management.

Categories and Subject Descriptors

C.2.3 [**Computer Systems Organization**]: Network Operations—*Network Management*; C.2.4 [**Computer Systems Organization**]: Distributed Systems; K.6.2 [**Management of Computing and Information Systems**]: Installation Management—*Performance and Usage Management*

General Terms

Reliability

Keywords

Document-based integrated management, autonomic systems, decision making, document processing unit, performance management, fault management

1. INTRODUCTION

With a large number of sophisticated products readily available on the market and common network and system management protocols long standardised, Enterprise Management Integration should be a solved problem. We know that this is not the case — the problem just seems too large to tackle for a simple and clean solution build on existing technology. Enterprise networks may comprise tens of thousands of devices from several vendors, all with slightly varying capabilities, software patch levels, protocol versions. Even if these could be cleanly integrated by following standardized protocols and operating procedures, there are still many large problems to face: Continuous integration whenever new devices are being introduced in the network, as infrastructure (for a detailed description of the history of PBM refer to the article by R.Boutaba and I. Aib[3]). PBM approaches depend on the existence of integration functionality within the network, which is caring for device homogenization and allowing for the reactive and dynamic modi?cation of the overall system's behaviour. More recently, research has been focussing on questions of policy generation and checking, as it is clear, that a large number of simultaneously active policies might not always be consistent or easy to manage. With a growing number of policies introduced in an infrastructure, it is becoming increasingly harder for administrators to conduct management tasks without additional tools supporting them.

Our approach on improving network and system management integration is motivated by the idea of an Autonomic System (AS): Ultimately, an AS is a system that is being able to continue to operate correctly in the face of environmental changes without direct human intervention (a property commonly referred to as homeostasis, since being coined by Ashby in the 1960's [4, chapter 5/3]). Full autonomicity is a highly ambitious goal for system engineers and of course, there are limits to the magnitude of changes that such a system might be exposed to - pulling the plug on a server running the application, e.g. We believe that the creation of a fully autonomic system can only be achieved by employing a self-organising conglomerate of interacting entities which are able to process contextual data and to adapt their own behaviour in a dynamic fashion. We developed and assessed this viewpoint over the course of the last years, mainly in?uenced by work undertaken in the CASCADAS research project.

The remainder of this paper is structured as follows: After detailing the related work in section 2, we give an introduction to DBIM in 3, followed by an example scenario highlighting key issues of our approach in section 4 and ending with conclusion in section 5.

2. RELATED WORK

Autonomic Systems Engineering (ASE) is still a relatively young field, having its roots in the Autonomic Computing [5] and Autonomic Communication [6] research initiatives, and sprouting novel ideas as part of, e.g. the CASCADAS research project [7], which is one of four projects funded within the European SAC program [8]. By now, the ideas that originated in the project have blossomed into concrete concepts, which were evaluated through the construction of an open-source toolkit for the engineering of autonomic systems.
The scope of CASCADAS is focussed on service engineering [9], but it soon became obvious that the discovered concepts can also be applied in a fruitful way to other domains, e.g. network and systems management: For an example take the idea of an Autonomic Communication Element (ACE), as initially defined by CASCADAS [10]. ACEs define basic components for the creation of AS that are able to react on the surrounding context by processing so-called plans, which specify a component's behaviour. They expose an implementation of the discovered concepts by providing *autonomic properties*: The ability to discover one another through group communication primitives, the ability to dynamically form aggregates by employing contracting mechanisms, the provisioning of an explicit representation of internal state that might be transported to other ACEs, or the ability to extend behavioural capabilities by dynamically adding programatic functionality. Would someone engineer a system with similar capabilities, but for other domains than the ones considered by CASCADAS, the resulting architecture would naturally exhibit similarities to the way ACEs are structured, as both architectures would aim at solving the same problems (a more detailed discussion on an architecture for a different domain and using a different approach is to follow in section 3.2.2).
Publications detailing the implementation of autonomic properties (often referred to as self-* properties) for specific distributed systems, frameworks, or middleware technologies are too many to list in this paper. A small (subjective) selection must suffice: Group communication is well researched, an adaptation to autonomic systems (using a declarative approach to group membership) and an overview of the state of the art can be found in one of our recent publications [11], the notion of contracts has been adapted to network management by J. Strassner [12], and a significant contribution is found in the article by K. Calvert and J. Griffioen et al. regarding network management through a collective of interacting entities [13].

Conventionally, network and systems management follows a manager-agent paradigm and is conducted using either plain SNMP [14], a Command Line Interface (CLI), RPC paradigms or by relying on distributed objects [15]. Recently — at least in the Operations Support Systems (OSS) management area — XML and WS have gained in popularity, due to ease of integration with existing backend systems and availability of off-the-shelf software. Most prominent are the Web Based Enterprise Management (WBEM) [16] standard by the DMTF, which is implemented in all newer Windows[1] operating systems under the name of "Windows Management Instrumentation" (WMI), and the NETCONF protocol [17, 18], created by the IETF and responsible for introducing the term *document-based* in combination with network management.
Within the network and systems management discipline other approaches have been put forward that try to equip management systems with autonomic capabilities, mainly from the perspective of PBM. A conceptional architecture for lower level regulatory loops in PBM and predecessor of the ideas formulated in this paper has been published by one of us (E. Höfig) [19]. H. Chen, S. Hariri et al. have demonstrated the usage of XML for self-configuration of networked systems with autonomic capabilities [20].

Arguably the currently most prominent approach to autonomic management has been specified by B. Jennings, S. van der Meer et al. through the FOCALE architecture [21], building on previous work (with J. Strassner) that revolves around the notion of a "policy continuum" [22, 23] and the facilitation of ontologies for management integration purposes [24], a topic that has also been researched earlier by J. de Vergara, V. Villagrá et al. [25], as well as N. Samaan and A. Karmouch, who describe the utilization of the Ontology Web Language for Services (OWL-S) for PBM [26]. The idea of a policy continuum has been developed due to the insight that most hard problems are not encountered during the technical implementation of PBM, but stem from the coalescence of PBM with the business processes and operational procedures that are run in an enterprise [27]. Integration issues are also the main reasons why FOCALE is concerned with information models and ontologies, proposing the use of DEN-ng as an information model for network-wide knowledge management [28].

3. DBIM

The main arguments for the usage of documents in integrated management is that they enable a well-known structure for arbitrary data, including mechanisms for solving localization issues using different encoding standards; that a large number of tools is available which allow for validation, transformation, generation, storage, and versioning of the documents; and last but not least, that many people and organizations put tremendous effort in standardizing relevant information models and languages using XML based languages. For an example think about the Security Assertion Markup Language (SAML), which enables single sign-on capabilities; Universal Description, Discovery and Integration (UDDI) as a standard way for discovering available services; Web Service Level Agreements (WSLA) for specifying service constraints between multiple parties; or simply WS-Management as a common way to manage an enterprise's infrastructure. The benefit of employing documents for system and network management is clear to most stakeholders in the market, but likewise it is generally known that a large fraction of today's equipment and nearly all legacy systems do not support management via document-based interfaces.

How is it then possible to transition from today's systems landscapes to ones that are managed using a document-based approach? It is obvious that mediation functionality is needed which would be able to translate management processes described in management documents to infrastructure management protocols understood by the infrastructure elements. We propose to place this functionality between the

[1]Since Microsoft Windows 2000

management systems and the infrastructure itself, as shown in Figure 1. We refer to it as the *document processing plane* of the network.

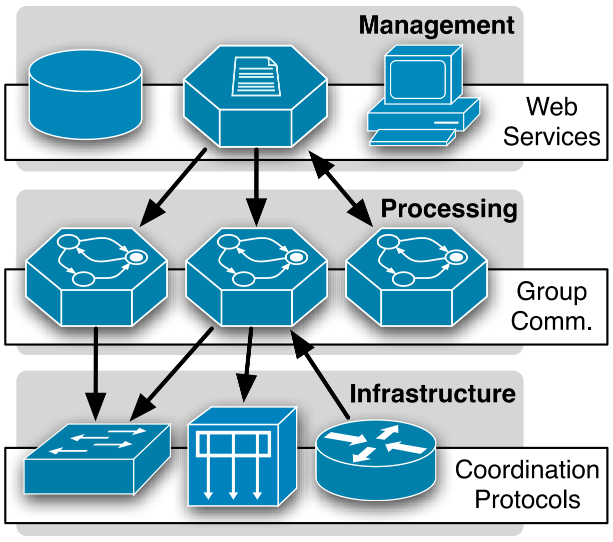


Figure 1: Conceptual DBIM Planes

Each of the three depicted planes is distinctively different from the others and operationally independent, meaning that there are no dependencies that would stop one of the planes from providing it's services should another plane be partially or fully inoperable. Each of the planes employs specific communication primitives (depicted as the white box underlying each of the contained elements) that are internally used between the elements on the plane, and employes a number of other means to communicate with elements external to a plane's scope (exemplarily depicted by the black arrows). Following, each of the planes is described in more detail.

3.1 Infrastructure Plane

The infrastructure plane contains all systems that are the actual management subject. These do not only encompass network elements like switching equipment, routers, firewalls, vpn concentrators, and the like, but also business systems like web portal clusters or directory servers. Internal communication on this plane is done through strictly non-XML coordination protocols, e.g. routing protocols like OSPF or RIP; OSI layer 2 protocols like STP; handover protocols between Wi-Fi base stations; or application server clustering protocols. Communication with a higher plane is carried out using existing management protocols like SNMP, NETCONF, CLI over SSH shell, or telnet. The management traffic mostly follows a pull-paradigm, flowing from the infrastructure towards the processing plane – but might also be inverted (as is the case with SNMP traps or NETCONF events). Our approach does not add or modify anything in this plane.

3.2 Decision Plane

The newly introduced decision plane is at the core of DBIM. It contains a number of distributed DPUs, which might be integrated with networking equipment[2], but may also take the form of standalone units executing on standard server hardware. Each DPU exposes autonomic capabilities and facilitates group communication paradigms to discover and co-ordinate with other DPUs; this enables us to support administrators in the transformation and deployment of documents at the processing units, up to the point of nearly complete automation. The result of a document deployment is a series of actions by the affected DPUs, which are acting in concert to achieve the purpose that was codified in the document.

[2]The authors are currently working on an implementation harnessing a Cisco 2800 Series Integrated Services Router

3.2.1 *Document Content*

A major crux of our approach lies in the question which kind of data to store in the documents, and how to interpret this content at the DPUs – using XML documents gives us an elegant way of processing arbitrary data content, but will not help in deciding on the management actions to take. Fortunately, this is a common problem and many organizations have already standardized enterprise management information models in XML, which we can use. This is a huge advantage of the DBIM approach: The DPU can deal with whatever information model is already in use: May it be based on the XML version of the Common Information Model (xmlCIM), on DEN-ng, or on SNMP MIBs that are automatically converted to XML [29, 30].
For data-intensive management purposes it might be sufficient to merely encode information according to the data model, e.g. capturing a certain version state for use in configuration management, but for most applications an operational specification is also needed. Following CASCADAS research, we postulate that documents should be **active**, meaning that they encode a behavior that should be executed by one or more DPUs (from now on we refer to these kind of documents as *plans*).
We chose state machines (SM) to encode such behavior, because the formalism is intuitive, simple, and well understood. Transitions between states in a SM are triggered by events, checked against a guard condition and result in an action. This is exactly the same as the Event-Condition-Action (ECA) paradigm used in PBM, with the notable difference that a transition is only a very small part of a SM, whereas a single ECA statement is central to the overall policy. Using SM we are able to specify dependencies between policies by describing sequences or alternatives on how policies are applied, giving human personnel a better understanding of the complete management process and allowing for a automated, exhaustive check of interdependencies between the codified management directives.

3.2.2 *DPU Architecture*

Plans are transformed and deployed to the DPUs by means of a proprietary management protocol. Figure 2 shows a conceptual diagram of the architecture of a DPU with a single deployed plan depicted in the center of the picture.

Besides the plan, the group communication mechanism is depicted at the right side of the diagram, as is the lifecycle and plan management functionality at the top. Shown at the bottom is a set of Functional Capabilities (FC) – these black-box components encapsulate programmatic functionality that can be bound to a DPU. For example a FC component could implement all necessary means to communicate

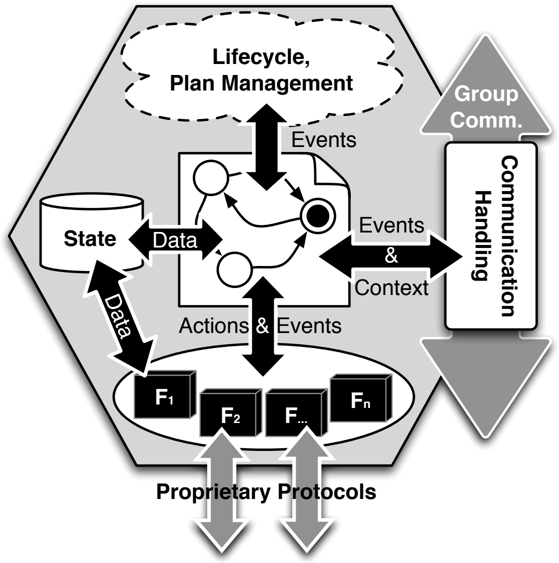


Figure 2: Internal Architecture of a DPU

via Java Management Extensions (JMX), allowing a plan, that is processed by a DPU with this FC, to manage a J2EE application server.

Transitions between states in the plan are triggered by events – these may be external messages (transported via group communication) or internal ones (e.g. initiated by the lifecycle management, or raised by a FC). Transitions, as well as states, may trigger actions which are understood as calls to arbitrary functionalities implemented by a FC. The final building block of the architecture is an explicit notion of state represented in the form of in-memory XML data. The DPU state contains session and management information and is accessible by both the plan, and the FC.

3.3 Management Plane

Communication on the management plane is defined as being carried out over WS. This might not always be the case nowadays, but is likely to become more and more common in the future. It also gives us ideal conditions for demonstrating the integration of high-level business management with low-level device management as both utilize XML and its associated technologies.
The management plane encompasses all conventional management systems, like alarm databases, directory servers, traffic flow monitors, and so forth. In addition to these, a central Plan Management (PM) entity is introduced. The PM allows for the creation of new, and the re-use of existing plan documents, it triggers deployment of plans and enables the management of plans that are processed by the DPUs.

Plans could be created by experts using visual editors, they could be inferred by recording the operations an administrator carries out when performing a management task, or they could even be generated by expert systems facilitating Artificial Intelligence (AI) methods. At the current stage of our research however, we do not prescribe how plans are created, but suppose that they are available through a document management system in the management network.

One of the major ideas behind the employment of plan documents for describing management operations is the potential re-use of previously captured expert knowledge. Once a plan is created for solving routine tasks, it can be added to a so-called Knowledge Base (KB) and be re-used, even by less-knowledgable people, than the original administrator that created the plan.

4. EXAMPLE SCENARIO

To demonstrate the feasibility of our ideas, this section contains a detailed scenario description from the realm of performance and fault management in an enterprise network: the utilized plans include formalized knowledge necessary for enacting troubleshooting procedures on the network. We choose to represent state machines using the State Charts formalism by D. Harel [31], which offers notions for abstraction and parallelism. For sake of clarity we do not show a XML representation of the plan documents, but a equivalent graphical representation, which is more intuitively understood by humans.

4.1 Network Description

The Scenario is based on a hypothetical network, belonging to the ACE Ltd., an enterprise in the manufacturing trade with a couple of hundreds of employees. There are two human actors: the *Support Assistant (SA)* – an employee in first level IT support and the *User (U)* – an accountant.

4.1.1 Network Topology

As depicted in figure 3, the ACE Ltd. uses two, logically separated, networks: a management network (thin black lines) and a productive network (thick gray lines). In the places where these are on the same physical medium, a logical separation[3] has been put in place. The complete network consists of five subnets, connected by four routers and some switches.

Management Subnet Provides the IT administration systems. It contains, besides connections to all routers, three databases for storing management-relevant data, a system for tracking support issues, a flow collector for capturing IP flow information, and a dedicated machine running the PM and a DPU.

Intranet Subnet A network containing the backend systems (application, file, and directory server), as well as a DPU. Incoming and outgoing IP traffic is monitored on a suitable port of the distribution router.

Division B Subnet A fully switched subnet utilizing an access router. The machine used by SA is found in this subnet. There are dedicated connections to the other subnets for improved network resilience.

Division A Subnet Is a slightly larger, but technically similar, subnet as the one for Division B. Accounting is found here and the access router is equipped with a DPU.

Internet Access Subnet Is directly connected to the internet. Protocols IP data flow information and contains an internet subscriber edge router with a DPU.

[3]For example Virtual LAN (VLAN)

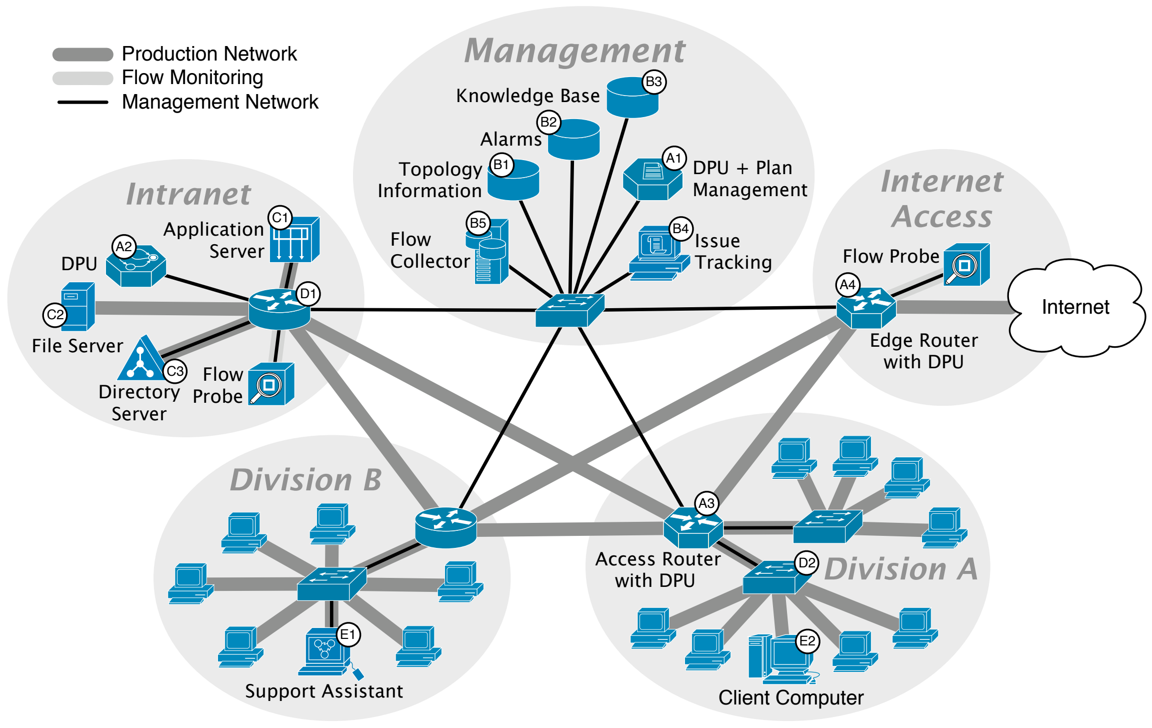


Figure 3: Example Network Topology

4.1.2 *Description of Individual Machines*

Elements that are relevant for the scenario are depicted by a designator and explained in the following text.

Machines with DPUs.

There are four systems that possess a DPU, each of them is shortly described and the general purposes of the configured FCs are mentioned.

A1 PM System
Contains a DPU and cares for document management: Coordinates plans, transforms them to conform to the network topology, and deploys them on the infrastructure.
FC: Statistics, Classification, Alarm Correlation, Messaging, Discovery, Query IPFIX Flow Information, Data Aggregation, Web Service (WS) Access, SQL Database Access, LDAP Directory Access, Plan Management

A2 Intranet DPU
A DPU which monitors performance of the intranet servers.
FC: JMX Access, Statistics, Classification, Messaging, Discovery, WS Access

A3 Access Router for Division A
A router with a DPU. May monitor performance of other network elements (NE).
FC: Statistics, Classification, Messaging, Discovery, NETCONF Management Access, SNMP Management Access

A4 Internet Subscriber Edge Router
A router with a DPU, which is able to monitor the performance of other NE.
FC: Statistics, Classification, Messaging, Discovery, NETCONF Management Access, SNMP Management Access

Support Systems.

Are systems that can be found within the ACE Ltd. infrastructure and which are being queried by the PM or DPUs.

B1 Database with Network Topology Information
Network topology information is being recorded. How this is done (e.g. manually or via discovery mechanisms) is irrelevant for the scenario. The topology information may be queried from a database using an SQL interface.

B2 Alarm Database
There is a database that collects network alarms. How this happens is again irrelevant for the scenario. The alarm database can be queried using a SQL interface.

B3 Troubleshooting Knowledge Base
Contains all plans, as well as the information related to troubleshooting tickets (TT). Queried over SQL.

B4 Issue Tracking
Collects and manages TT. The system communications with the KB and SA. It is possible to change TT states using a WS API.

B5 Data Flow Collector
Captures data flows in the productive network using IPFIX. Provides data flow information through a WS API. Is able to provide historical information up to a month in the past. This System was introduced by

ACE Ltd. to gain a better understanding over the data flows within their internal networks.

Backend Server.
The server machines that run the business critical applications of ACE Ltd.

C1 Application Server
This J2EE server executes applications, for example the accounting system. The Virtual Machines (VM) can be managed remotely using an JMX interface. Two network connections lead to the machine: One from the production network and the other one from the management network.

C2 File Server
Stores the files for all of the enterprise employees.

C3 Directory Server
The machine provides directory data (user identities, inventory data) using an LDAP interface. It is also connected in a redundant fashion, similar to the application server.

Legacy NE.
Are conventional devices that do not expose autonomic capabilities. The scenario follows an evolutionary approach, therefore it is possible to introduce AC Devices one after another in the network. It is not necessary to equip all NE with autonomic capabilities for the scenario to work. To show this, the scenario integrates two legacy devices.

D1 Intranet Distribution Router
The device is of an older date, without autonomic capabilities. Its configuration and performance values can be queried using the NETCONF protocol.

D2 Ethernet Switch of Division A
A managed switch which might be configured or monitored using SNMP.

Frontend Systems.
Are machines that are used by the scenario actors.

E1 SA Console
The machine that SA uses to connect to the issue tracking (B4) and PM system (A1).

E2 Desktop Computer of U
The machine that U uses for everyday tasks, e.g. for connection with the accounting application on the application server (C1).

4.2 Scenario Description

This section describes a typical troubleshooting scenario in a step-by-step manner, including a detailed description of the plans that are being used to teach the network about troubleshooting activities. Tangible processes are set in sans-serif font.

4.2.1 Anamnesis

The Phone rings at IT support. SA answers and talks to U. He complains about the network performance: "My access to the accounts is sometimes really slow". SA creates a TT and starts to record the incident: Who reported the issue, which systems are affected, which machine was used for accessing the accounts, how often does the problem appear, what exactly means "really slow"?
The problem is not known to SA and she starts looking up similar issues in the KB. She discovers a number of other TT with similar issues described, pointing to several different root causes.

4.2.2 Root Cause Hypotheses Establishment

After reading some of the TT descriptions, the SA identifies four reasons that might be potential causes for the observed behavior:

1. The application server that executes the accounting software is running under full load. For example, this could be due to a slow processor or a bad configuration of the Java VM garbage collection parameters.

2. The network route between the desktop computer of U and the application server is operating at full capacity. This could be due to, e.g. transmission of abnormally high data volumes or because of badly dimensioned bandwidth during capacity planning of the network.

3. The hardware of a NE en route between the desktop computer of U and the application server is damaged. For example, a cable could have a loose connection or an interface card could have switched of due to high temperature.

4. The desktop computer of U is operating at its limits, e.g. due to an overeager virus scanner that hugs the processor, or because the machine is generally of an older date and not capable to keep up with the requirements.

U explains that "the problem only happens from time to time and I'm not a computer-savvy guy, anyway". SA decides, that all four possibilities are relevant and that they should be monitored to find a solution to the problem.

Normally, the SA would now check all of the four hypotheses — for the given scenario and using nowadays tools this would be a very tough problem for a first-level IT support employee. On the one hand, the SA would need to have access to the necessary toolset (plus the expertise on how to use them) for checking each of the four possibilities, on the other hand it would be inevitable (due to the sporadic nature of the glitch) that U tells the SA exactly when the problem is happening. Arranging such a feedback is not trivial: Helpdesk staff is often not allowed to give out their internal extension number, or they are on the line with another client and unavailable to take the call.
The most likely reaction of the SA would be to only falsify hypothesis 3 through a manual search in the alarm database and then to escalate the ticket to second-line support.

Now — let us assume that the network is equipped with a PM and several DPUs, facilitating our novel technology

("Autonomic Capabilities for Enterprise Management Integration"). In this case the SA would have access to a collection of troubleshooting plan documents, which would either be directly found in the KB or attached to an already solved TT. The SA would be able to re-use knowledge by copying these plans (from now on referenced as Plan A, B, C, D – matching to hypothesis 1, 2, 3, 4) to the new TT and coordinate the four plans by adding a new plan (E).

4.2.3 Plan documents

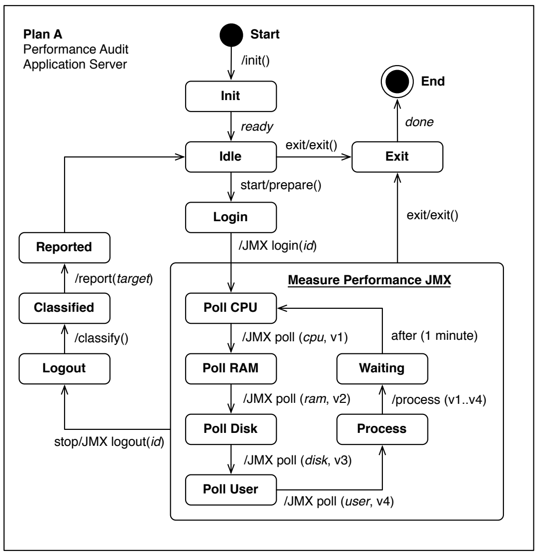
Following is a description of the five plan documents A) – F), that would now be found in the TT. We are detailing which requirements a plan has, what a plan offers to other plans, and what kind of functional logic it implements. Plan logic is represented graphically in the figures **Plan A – E**.

Plan A) Performance Audit of an Application Server.

Requires Access to management interface of accounting system, identification of the administrator, ability to receive Start and Stop messages, access to an interface for delivery of result data, access to an interface for the transmission of failures, ability to use JMX, ability to calculate statistical data, ability to classify result data

Provides Performance data of the application server

Description Measures performance values (CPU utilization / RAM utilization / hard disk utilization / number of users) by employing cyclic polling using an JMX interface. Calculates statistical values (minimum / maximum / average) using the measured values and classifies the result (red / yellow / green).



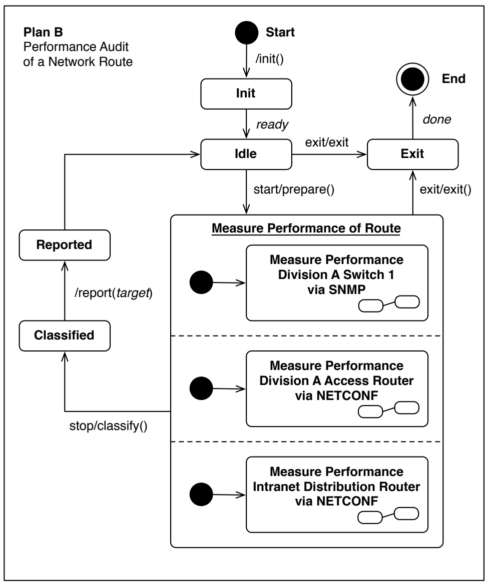
Plan B) Performance Audit on a Network Route.

This plan comprises three sub plans in separate diagrams, which are processed in parallel. The sub plans are simple, and quite similar to each other; their specification is left as a task to the reader.

Requires Knowledge about the route configuration, access to all NE on the route, identification of the administrator, ability to receive Start and Stop messages, access to an interface for delivery of result data, access to an interface for the transmission of failures, ability to use NETCONF, ability to use SNMP, ability to calculate statistical data, ability to classify result data

Provides Performance data of all NE on a route

Description Measures performance values (CPU utilization / lost packets / TCP retransmissions / packet delay) of all NE on a route using cyclic polling over NETCONF and SNMP. Calculates statistics and classifies the results to give an overall view of the route's state.



Plan C) Alarm Collection on a Network Route.

Requires Knowledge about the route configuration, Knowledge about the time-span of the activity, access to the alarm database, access to the interface for delivery of an alarm list, ability to correlate and filter alarms

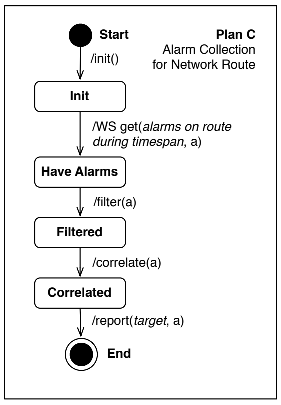
Provides List of all occurred alarms in given period

Description Queries the alarm database for certain types of alarms (high packet loss, link loss, line card fail) that were send by NE on the route and in the given period of time. Prepares results by correlating and filtering the alarms.

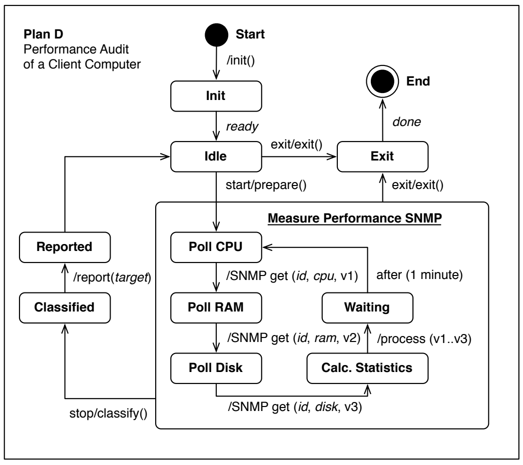
Plan D) Performance Audit of a Client Computer.

Requires Access to management interface of U's desktop computer, identification of the administrator, ability to receive Start and Stop messages, access to an interface for delivery of result data, access to an interface for the transmission of failures, ability to use SNMP, ability to calculate statistical data, ability to classify result data

Provides Performance data for the desktop computer of U



Description Measures performance (CPU utilization per process / RAM utilization / hard disk utilization) of a PC by cyclic polling using SNMP. Calculates statistical values and classifies the results.



Plan E) Coordination.

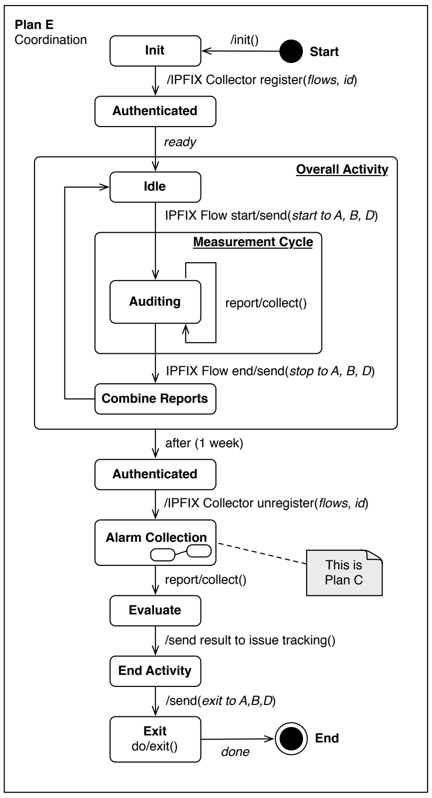
Requires Knowledge about the time-span of the activity, knowledge about the route configuration, identification of the administrator, access to the flow collector, access to the issue tracking system, ability to collect results

Provides Collected performance data, list of relevant alarms, ability to send Start and Stop messages, interface for receiving individual result data, interface for receiving failure information, interface for receiving alarm list

Description Triggers performance measurements whenever U uses the accounting system by analyzing IPFIX flow information. Collects results and any errors over the period of a week. Ends the activity after expiration of the time period and informs SA by activating the TT.

4.2.4 Preparation of the Observation Activity

SA uses the PM system to bind, in consultation with U, open plan parameters to concrete values. With some parameters, she keeps the default values (e.g. time period of the activity – one week), while other values come from information



in the TT (e.g. identification of the desktop computer of U), are disclosed by the U (e.g. identification of the accounting system), or stem from her own experience (e.g. identification of the administrator).

After filling in the missing parameters, the system tries to automatically resolve any dependencies between the plans by an analysis of the required and the provided information of all plans. If this is not possible or not ambiguous, the SA is asked to correct the dependencies manually. The result is an aggregated document, combining all five plans. SA explains the start of the measurement activity to U and promises to resume contact in a week's time. The phone call between U and SA ends.

4.2.5 Plan Deployment

To convey to the network how to run the automatic measurement activities, it is necessary to transform the aggregate plan document for deployment on the individual DPUs. To prepare for this, the PM needs to first collect the existing autonomic capabilities of the available DPUs, for example by employing a decentralized resource discovery mechanism. Based on utilization and retrieved capability information, the system then partitions the plan in a suitable manner and assigns plan fragments to individual devices or alternatively responds with an error if a partitioning of the plan is not possible.

The following mapping of plans to DPUs is the result of this transformation step.

PM (A1) Responsible for coordination plan E and collection of alarms using plan C, chosen due to the required capabilities.

Intranet DPU (A2) Will measure performance of the application server (plan A), because this device is the only one able to access application server via JMX.

Division A Access Router (A3) Is assigned plan D (Performance Audit of a Client Computer), because of required capabilities and the proximity to U's computer.

Internet Edge Router (A4) Will process plan B (Performance Audit on a Network Route) due to available capabilities and a low utilization of the device.

By triggering the plan deployment, the SA ends handling the TT. Subsequently, the AC management system separates the aggregated document in four individual documents that are deployed at the DPUs.

As soon as a plan is received by a DPU, it is adapted to the local environment by a transformation: Local symbols are resolved (e.g. the management interface of the accounting system is resolved to an IP address and a port number) and the specified messaging channels are bound using a contracting mechanism (e.g. a logical connection between A4 and A1 will be established for the delivery of performance results). Finally the plans are completely deployed, and they are started to fulfill the assigned tasks.

4.2.6 *Observation*

Every time that U access the accounting system, a processing of the plans A, B, D, and E is triggered (using an IPFIX Flow Start message to E and subsequent Start message to A, B, and D – an end of the data flow leads to a stopping of the plans in almost the same manner). After one week, the measured performance values for each cycle are combined with the alarms in that period and stored in the TT. The TT is made active and the plans are automatically removed from the infrastructure by the PM.

4.2.7 *Diagnosis*

The SA is prompted to evaluate the results by the finding the active TT in a ticket queue that is assigned to her or to her support team.

Following is a summarized report of the results of the measurement activity.

Measurement Plan A Result classified as "green": The server has not been over-utilized.

Measurement Plan B Result classified as "red": A certain link in the subnet of division A was two times utilized with full capacity (Mo. 14:01h - 14:23h and Fr. 14:05h - 14:18h). This has been determined by a unusual high number of TCP re-transmits and packet losses.

Alarms Plan C No relevant alarms have been detected in the time-span.

Measurement Plan D Result classified as "yellow": There were utilization peaks, but nothing that would not be acceptable.

SA concludes to conduct a specific audit on the over-utilized link. U is informed about the intermediary results. After further analysis of the link in question (using more plans to do traffic analysis, etc.), it turns out that all computers in the subnet of division A start to run a backup script at 14:00h each day – this traffic chokes the link completely. The administrator of division A modifies the backup scripts and the problem vanishes. U is informed, the TT closed and added to the KB. The troubleshooting plans are now available to other troubleshooters as knowledge for re-use or modification.

5. CONCLUSION

The current state of systems and network management is characterized by a struggle to integrate not only the most heterogeneous devices and software tools, but also to align integrated management with the overall business processes in an enterprise. By using technologies like WS and XML, which are widely adopted in the Service-Oriented Architecture (SOA) and business process field, a realistic opportunity is created that allows to bridge this gap. The struggle for management integration will always continue, but it might be eased with the proper technologies and tools. The utilization of autonomic capabilities together with a document-based approach promises to at least take some weight from the shoulders of the people that run networks by enabling a better re-use of management knowledge. Our research indicates that it is possible to capture the human expertise that is necessary to operate network management tools by employing active documents (plans) which are interpreted by distributed execution units within the network. Adopting such an approach would lead to a situation were administrative personnel would not need to exactly know how to work a certain tool, but only where to look for a suitable plan.

Document-based management integration is an evolutionary approach, as it does not force an organization to adopt a certain information model. Quite the contrary is the case: It would be possible for an enterprise to employ several different information models at the same time, and to integrate them using adequate plans, which are easily adjusted as the underlying operational procedures stay the same. The approach is well-suited for slowly transitioning from one technology to another: It is always possible to only have part of the systems managed by DBIM, but to use conventional processes in other parts. The same flexibility would hold true for hardware costs and maintenance efforts: As DPUs are self-organized entities, they could be added to the network in locations where they would be most needed; and because they are self-similar units, maintenance is easily done in the same fashion for all of them.

Finally, some might argue that DBIM is the same as Active Networks or similar to Agent-Based Programming. We do not believe that this is true. Active Networks and Agent-Based Programming both aim at creating programmable network nodes: an approach that conflicts with enterprise security needs, as well as with the limited hardware resources available at the NE. DBIM does not touch the NEs, but introduces an additional processing plane in the network that interprets well-formed operational specifications. An operator has complete control over the activities in its network and can decide itself which functionalities to provide at the DPU level.

6. REFERENCES

[1] A. Clemm, *Network Management Fundamentals*, Cisco Press, ISBN 1-58720-137-2, 2006.

[2] J. Strassner, *Policy-Based Network Management*, Morgan Kaufman Publishers, ISBN 1-55860-859-1, 2003.

[3] R. Boutaba, I. Aib, *Policy-Based Management: A Historical Perspective*, Journal of Network and Systems Management, Volume 15, Number 4, pp. 447-480, 2007.

[4] W. Ross Ashby, *Design for a Brain*, Second Edition, Chapman & Hall Ltd., London, UK, 1966.

[5] IBM Corporation, *An Architectural Blueprint for Autonomic Computing*, White Paper, Third Ed., 2005, `http://ibm.com/autonomic/pdfs/ACwpFinal.pdf`

[6] Fraunhofer FOKUS, *Autonomic Communication*, White Paper, Second Edition, 2004, `http://www.autonomic-communication.org/publications/doc/WP_v02.pdf`

[7] CASCADAS Research Project, *Bringing Autonomic Services To Life*, White Paper, 2008, `http://www.cascadas-project.org/D8.4.pdf`

[8] F. Sestinim et al., *Situated and Autonomic Communication an EC FET European Initiative*, ACM SIGCOMM Computer Communication Review, Volume 36, Number 2, 2006.

[9] A. Manzalini and F. Zambonelli, *Towards Autonomic and Situation-Aware Communication Services: the CASCADAS Vision*, In Proc. of IEEE Workshop on Distributed Intelligent Systems, 2006.

[10] E. Höfig, B. Wüst et al., *On Concepts for Autonomic Communication Elements*, In Proc. of 1st IEEE International Workshop on Modelling Autonomic Communication Environments, 2006.

[11] E. Höfig, *Autonomic Reliable Multicast: Application-Level Group Communication Using Self-Organization Principles*, Proc. 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems, 2007.

[12] J. Strassner, *Using Lifecycles and Contracts to Build Better Telecommunications Systems*, Lecture Notes in Computer Science, Volume 3262, pp. 483-497, 2004.

[13] K. Calvert and J. Griffioen et al., *Scalable Network Management Using Lightweight Programmable Network Services*, Journal of Network and Systems Management, Volume 14, Number 1, pp. 15-47, 2006.

[14] *Simple Network Management Protocol*, `http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol`

[15] G. Pavlou, *On the Evolution of Management Approaches, Frameworks and Protocols: A Historical Perspective*, Journal of Network and Systems Management, Volume 15, Number 4, pp. 425-445, 2007.

[16] Distributed Management Task Force, *Web Based Enterprise Management*, `http://www.dmtf.org/standards/wbem`

[17] *Network Configuration Protocol and Transport Bindings*, RFC 4741 – RFC 4744.

[18] M.-J. Choi, H.-M. Choi et al., *XML-Based Configuration Management for IP Network Devices*, IEEE Communications Magazine, Volume 42, Number 7, pp. 84- 9, 2004.

[19] E. Höfig, H. Coşkun, *Using Pattern Bound Policies to Construct Regulatory Mechanisms for Autonomic Systems*, In Proc. 10th International Conference on Quality Engineering in Software Technology, pp. 373-393, 2007.

[20] H. Chen, S. Hariri et al., *An Innovative Self-Configuration Approach for Networked Systems and Applications*, Proc. IEEE International Conference on Computer Systems and Applications, pp. 537-544, 2006.

[21] B. Jennings, S. van der Meer et al., *Towards Autonomic Management of Communications Networks*, IEEE Communications Magazine, Volume 45, Number 10, pp. 112-121, 2007.

[22] S. van der Meer and A. Davy et al., *Autonomic Networking: Prototype Implementation of the Policy Continuum*, In Proc. 1st Workshop on Broadband Convergence Networks, 2006.

[23] S. Davy, B. Jennings et al., *The Policy Continuum – A Formal Model*, In Proc. 2nd IEEE International Workshop on Modelling Autonomic Communications Environments, 2007.

[24] J. Strassner, D. O'Sullivan et al., *Ontologies in the Engineering of Management and Autonomic Systems: A Reality Check*, Journal of Network and Systems Management, Volume 15, Number 1, pp. 5-11, 2007.

[25] J. de Vergara, V. Villagrá et al., *Semantic Management: Application of Ontologies for the Integration of Management Information Models*, Proc. of the 8th IFIP/IEEE International Symposium Integrated Network Management, 2003.

[26] N. Samaan, A. Karmouch, *A Novel Approach Towards Autonomic Management in Context-Aware Communication Systems*, Lecture Notes in Computer Science, Volume 3744, pp. 374-383, 2005.

[27] J. Strassner, *DEN-ng: Achieving Business-Driven Network Management*, Proc. Network Operations and Management Symposium, 2002.

[28] J. Strassner, *Knowledge Management Issues for Autonomic Systems*, In Proc. Database and Expert Systems Applications, 2005.

[29] K. Youn, C. Hong, *An XML-Based Dynamic Network Management System Using Web Technology*, In Proc. 22nd International Conference on Distributed Computing Systems Workshops, 2002.

[30] J.-H. Yoon, H.-T. Ju et al., *Development of SNMP-XML Translator and Gateway for XML-Based Integrated Network Management*, International Journal of Network Management, Volume 13, Number 4, pp. 259-276, 2003.

[31] D. Harel and M. Politi, *Modeling Reactive Systems with Statecharts: The STATEMATE Approach*, McGraw-Hill, 1998.