Toward Goal-Based Autonomic Networking

Åsa Berglund*, Björn Bjurling†, Ramide Dantas‡, Susanne Engberg*, Pablo Giambiagi† and Börje Ohlman*
*Ericsson Research, Sweden. Email: {Asa.Berglund,Susanne.C.Engberg,Borje.Ohlman}@ericsson.com
†Swedish Institute of Computer Science, Sweden. Email: {bgb,pablo}@sics.se
‡Universidade Federal de Pernambuco, Brazil. Email: ramide@gprt.ufpe.br

Abstract*—The ability to quickly deploy and efficiently manage services is critical to the telecommunications industry. Currently, services are designed and managed by different teams with expertise over a wide range of concerns, from high-level business to low-level network aspects. Not only is this approach expensive in terms of time and resources, but it also has problems to scale up to new outsourcing and/or multi-vendor models, where subsystems and teams belong to different organizations. We endorse the idea, upheld among others in the autonomic computing community, that the network and system components involved in the provision of a service must be crafted to facilitate their management. Furthermore, they should help bridge the gap between network and business concerns. In this paper, we sketch an approach based on early work on the hierarchical organization of autonomic entities that possibly belong to different organizations. An autonomic entity governs over other autonomic entities by defining their goals. Thus, it is up to each autonomic entity to decide its line of actions in order to fulfill its goals, and the governing entity needs not know about the internals of its subordinates. We illustrate the approach with a simple but still rich example of a telecom service.

I. Introduction

In the telecommunications world there is an increasing focus on improving the manageability of networks. This is the result of the identification of a number of problems with the way networks are currently managed: long time to market for new services; inability of today's systems to efficiently relate the network with the service and the customer; and a lack of visibility into end-to-end processes [1]. Today, the process of introducing a new end-user service in a telecom network is very time consuming, where most new services take between 6 to 18 months to deploy [1]. With an increasing demand for interoperability between different actors, such as network providers and content providers, and where multi-vendor networks are becoming increasingly common, the telecom industry is facing an even more complex management environment. It will become infeasible for one actor to maintain complete knowledge of all the different parts contributing to the delivery of a service. At the same time it is crucial to be able to efficiently monitor and control the performance of such a service, which further accentuates the growing need for bringing business service management and network management practices closer together.

This paper presents early and initial work toward a solution for these network management challenges. The solution will introduce a hierarchical autonomic network architecture of a new breed of components which we call *Autonomic Entities* (AE). The hierarchy will be induced by Service Level Agreements (SLA) that describe how one AE may govern over other AEs' provision of services. We are aiming at goal-based governance, where each AE should be able both to understand goals and to perform local dynamic goal refinement at execution time. A goal differs from a policy in the sense that it specifies a desired state rather than how that state is to be obtained.

A benefit of our approach to an autonomic network solution will be the provision for a degree of control (governance) of a service without the necessity of complete knowledge of its internal structure. An AE performs goal refinement locally without needing to know about the internal structure of its contracted AEs. Goals and performance reports are communicated between AEs via special governance interfaces. This approach abstracts away low-level details from business and service management, allowing them to manage the network on a business service level. A contribution with this paper is that it gives a detailed outline of our approach and of the initial steps. We believe that our solution will lead to significant reductions both in cost and in time to market for new telecom services, as it aims at alleviating some of the obstacles in cross-organizational service deployment.

Section II presents a scenario, on which the paper is developed. Section III gives the preliminaries for our development. Section IV details our proposed goal-based approach and Section V suggests an architecture for it. Conclusions and future work are given in Section VI.

II. The Virtual Flower Scenario

The scenario shows how existent services are aggregated into a new end-user service. The new service, called "Virtual Flower" or VF, lets a customer send virtual or real flower bouquets to his/her mother on Mother's Day.

A Virtual bouquet consists of an MMS message showing a picture of a flower bouquet chosen by the customer. The sub-services for VF include messaging infrastructure (MMS); a customer order interface; and a coordination service that provides the logic for the VF. The MMS service can be assumed to already have been deployed and running, thus leaving the other services to be built by the service designer.

For the real flower service, the mother receives the virtual bouquet and is also asked for delivery instructions for the real bouquet. A third party service provided by a flower vendor (call it "Florist") must be contracted for the delivery of the real flower bouquet. The coordination service must consider the collaboration between the MMS service and Florist to allow the correct delivery of bouquets.

A. Realizing the Scenario

An initial non-trivial task for creating a new service is to identify the required sub-services. For the VF service, it is also critical to meet a hard deadline: Mother's Day. It must be ensured, via service level agreements, that the VF sub-services are able to provide the expected functionalities on time and according to requested performance guarantees. The agreements relate the VF's elements in a scalable hierarchy. Service operation raises a new set of challenges. Failure of components to meet agreements, or resource shortages (e.g., Florist running out of bouquets) may require services to be changed "on the fly". Such a failure will need to be reported to the higher level service, in order to enable it to take appropriate action to fulfill it's own agreements.

The scenario illustrates that sub-services may be delivered by several different organizations. We argue in Section IV that current policy-based autonomic networking solutions cannot satisfactorily cope with this issue.

III. Preliminaries

A. Services

In the literature, the concept of *service* has a broad range of connotations, ranging from *end user service*s, like for example Internet connectivity, to lower level *system service*s, like charging or terminal location services.

The web service community contributes with various languages for the description of services. For example, WSDL [2] is used for describing inputs and outputs of services that are invoked over the Web. While we will adopt such service description facilitators, this notion of service is too narrow in scope for our purposes, as it concentrates only on Internet-based technologies. Our conception of *service* also draws from advances in the economic and business sciences. Kotler [3] defines a service as being any act or performance that one party offers another party. Grönroos [4] adds to this definition by saying that a service is provided as a solution to a specific consumer request. An important delineation of the concept for this paper, is that a service is *delivered* by a *responsible* agent. In connection with network or Internet services, it is essential to be able to identify and give structure to the systems that deliver services.

The area of Service Oriented Architectures (SOA) contributes to our notion of service specifically in terms of automation or semi-automation of the creation of new services through combinations of already existing ones. We focus on the ability to enable easy interoperability and combination of services, potentially provided by different organizations.

Requirements on the delivery and performance of a service are usually stated in a Service Level Agreement (SLA). An SLA is a contract between two parties (a *service provider* and a *consumer*) whose terms may include, among other items, a clear definition of the service and performance parameters. What distinguishes an SLA is that the terms of an SLA are equipped with *metrics*, which facilitate the description of the service levels in the contract. The literature provides several languages for SLAs. For example, IBM's language for *Web Service Level Agreements (WSLA)* [5] is used for regulating web service provisioning and monitoring, which works similarly to how we described SLAs above.

B. Autonomic Networking

The complexity of current networks has driven major players in the IT industry to look for solutions that reduce the burden of managing such large, heterogeneous systems. Companies like IBM, HP, and Microsoft are researching technologies that should enable self-* capabilities, like for example self-configuration, self-optimization, self-healing and self-protection. Such capabilities should ultimately result in systems that can manage themselves, requiring human intervention only for higher level business decisions.

IBM's Autonomic Computing (AC) initiative [6], first introduced in 2001, has published concepts for autonomic, or self-managing, systems. A key concept in the AC architecture is the Autonomic Element, which consists of one or more Managed Resources and an Autonomic Manager that controls them. The Autonomic Manager accesses the Managed Resource, which can be any type of hardware or software resource, through a touchpoint, sometimes called a manageability endpoint, which is a consistent, standard manageability interface for accessing and controlling a Managed Resource [7].

C. Policy-Based Management

Policy-based Management (PBM) is an approach used to simplify the management of networks and systems by establishing policies to deal with situations that are likely to occur **[8]**. PBM separates the rules governing a system's functionality from its implementation, allowing dynamic change in behavior without recoding. A common definition of a policy is that of "if *event* and *conditions* then *actions*", which prescribes that if certain conditions are present under the occurrence of a specific event, then specific actions must be taken in a policy-controlled environment. A somewhat broader definition, taken from the business domain, is that a policy is something that guides actions toward those that are most likely to achieve a desired outcome.

The area of policy refinement is starting to receive an increasing interest from the research community, and has been identified as a key area to enable policy-based management [9]. The aim of refinement is to enable network administrators to specify only high level (business level) policies and goals, and let the system (semi-)automatically refine these into enforceable low level policies and configurations.

Current refinement methods start with a formal representation of the system and the goals to be fulfilled. A high-level goal is transformed into operations and policies that are supported by the underlying system, which, when executed, will achieve the high-level goal. The process can be viewed as an off-line process, where complete knowledge about the system, and the high-level goals, are fed to an algorithm that derives the enforceable policies for the network devices. These configurations are then pushed to the network devices,

and if everything is performed correctly this should result in the achievement of the given high-level goals. The goal-based approach presented by Bandara *et. al*. [10] uses goal elaboration, which is a manual or at most a semi-automated procedure, based on the KAOS method [11], combined with abductive reasoning to infer the mechanisms by which the given system can achieve a particular goal. A similar approach is presented by Rubio *et. al*. [9], using model checking to obtain the sequence of actions needed for accomplishing a goal. Lehtihet *et. al*. [12] present an approach for autonomic management based on goals, where the definition of a common data model enables the network administrator to define the set of elements, and the respective low-level goal specifications, that are required to achieve a particular high-level goal.

IV. A Goal-Based Approach

The Virtual Flower scenario illustrates both the need for modeling multi-organizational networks and for provisioning for the creation of aggregated services. These two requirements may clearly be in conflict. The systems that implement those services may belong to, and be controlled by, different organizations, which do not necessarily agree on disclosing the internal structure of the services. Without knowledge of the internal structure of the services, it may be impossible to write policies, and exert control, for how the services should contribute to the new service. On this point, policy-based management may thus not be applicable. We propose instead a solution that enables the control of subordinate services through the means of *goals*. Goals are explored in this section.

A further requirement, also evident from the Virtual Flower scenario, is that the gap between service and network management needs to be reduced. In analogy with policy refinement techniques, we introduce a new approach to *goal refinement* for abstracting away low-level details from business and service level management. What is new with our approach is that the refinement process takes place in a distributed fashion online and dynamically inside the autonomic entities, which build up network components, as described in Section V.

A. Goals and Goal Refinement

The definitions of *goals* and *policies* related to businesses and organizations can be quite different from definitions related to networks and systems. Since we want to bridge the gap between these domains there is a need to have a terminology that can be agreed and applied to both areas.

A *goal* is, following Kavakli and Loucopoulus [13], "...a desired condition potentially attained at the end of an action (or a process)". We shall say that what distinguishes a goal from a policy is that a goal is something that a stakeholder hopes to achieve in the future, independent of plans, procedures or other means of attaining the goal, whereas a policy is an exact specification of actions that are likely to achieve a desired outcome.

In general, a consumer of a particular service has some expectations on the delivery of the service. The consumer has normally neither the wish nor the authority to specify the exact implementation details of that service, but needs to specify goals for the service which are relevant and understandable to the consumer. In our approach, the consumer's goals are refined and communicated to the underlying sub-services.

Current refinement methodologies, presented in subsection III-C, require that the actual refinement process is performed "off-line", outside of the system where the policies are to be implemented. These methods require a separate knowledge base for storing a formal representation of the system. High-level goals or policies are refined using this representation, and the resulting low-level policies are pushed onto the corresponding network resources. Bandara et.al. [10] state that *"in our approach, it is expected that the user would provide a representation of the system description, in terms of the properties and behavior of the components, together with a definition of the goals that the system must satisfy."*

We do not make the assumption that there is a user who is able to provide a complete representation of the system for the purpose of goal refinement. Instead, goal refinement is performed locally and dynamically inside the entities that actually provide the services on the basis of their current states obtained from self-monitoring.

B. Service Level Agreements

SLAs are the means by which a service provider and a consumer, possibly belonging to different organizations, agree on the conditions under which the consumer can define goals for the service. What these conditions really look like depends on the SLA definition language. As an example, an SLA may list goals that must be satisfied by the service at all times, goals that the consumer may decide when to request, and even goals that the provider can refuse depending on resource availability. The agreement defines obligations on the service provider to report on the status of goals, indicating whether they have been, can or cannot be fulfilled. In case a goal cannot be fulfilled, the agreement may determine new goals to be imposed on the service. Goals are usually expressed in terms of metrics on service performance. These metrics are also defined by the SLA.

Note that we assume that a party enters an SLA with good and honest intentions. In this sense, our systems and system components are *autonomic* rather than *autonomous.* An autonomous component possesses *freewill* [14], and in an autonomous system, no component can force another to change its behavior. An *autonomic system* is a system with the ability to monitor, control, and adjust its behavior, however subject to constraints set by an authority. An autonomic system may not refuse to enter an SLA for a specific service if it has the ability to deliver the service, in contrast with autonomous systems that may choose whether to do so.

V. Architecture

Building upon the ideas of autonomic computing, we propose an architecture consisting of a hierarchy of *autonomic entities*. Each autonomic entity is responsible for providing a service. As a service may be constructed upon other simpler

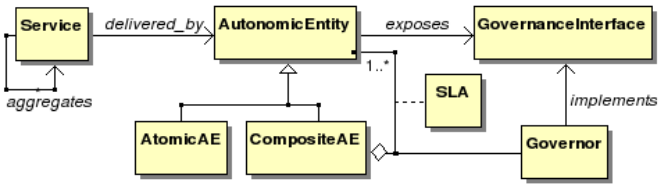

Figure 1. Relation between services and autonomic entities.

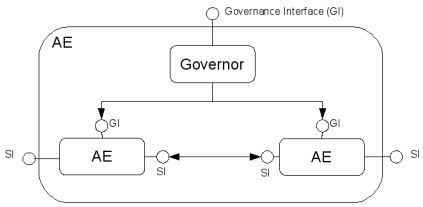

Figure 2. Internal Architecture of an Autonomic Entity.

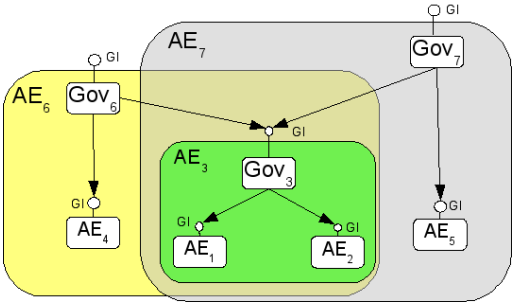

Figure 3. An AE can participate in several AEs.

services, the hierarchy of autonomic entities is determined by the relations among the services they provide (see Fig. 1).

An AE is not restricted to only a single network element, it could consist of several network elements, parts of a network element, a process, or even a software module.

When building a new autonomic entity, say AE_1, which requires a service that is implemented by another autonomic entity, say AE_2, a service level agreement ($SLA_{1,2}$) must be established. The agreement gives AE_1 limited rights to govern over AE_2. We call this relation between AE_1 and AE_2, a *governance* relation, and say that AE_1 *governs over* AE_2 *according to* $SLA_{1,2}$. We also say that AE_2 *is* AE_1*'s subordinate*. The graph of the governance relation is directed and acyclic, meaning that no autonomic entity is its own subordinate, and that an autonomic entity may be subordinate to more than one other autonomic entity. An autonomic entity that does not have any subordinates is called atomic. All other autonomic entities are called composite (see Fig. 1).

Viewed from the outside, an autonomic entity exposes two kinds of interface: a Service Interface (SI), allowing access to the service it provides; and a Governance Interface (GI), through which external entities can set goals that apply to the autonomic entity as a whole and receive reports on the entity's performance and status. This authoritative role, called *governing entity*, can be played either by another autonomic entity or by a Business Service Manager, which can be a person with business expertise and limited technical knowledge that governs the network on a business service level. The emphasis is thus on the ability of the AE to accept and apply goals set up by its governing entities, always within the borders of pre-established SLAs. Note that a goal does not tell exactly *how* an AE should behave, it just defines the states that the AE should eventually reach (or simply avoid). In order to achieve its goals, an AE dynamically adapts its behavior based on environmental changes and its own internal state.

The fundamental difference between an autonomic entity, as understood in this paper, and an autonomic element as described in subsection III-B , resides precisely in that the former presents a governance interface, whereas the latter presents a classic management interface. The governance interface is different from a classic management interface in the way that other entities only specify what needs to be achieved, and not how that is to be accomplished.

A. Composite Autonomic Entities

A composite autonomic entity governs over one or more subordinate AEs. The Governor is the component responsible for interpreting goals defined for the whole composite AE and refining them into goals for the subordinate AEs. The goal refinement process is performed dynamically during service operation, internally within the autonomic entity. This is different from previous refinement approaches as described in subsection III-C. To the outside, the Governor implements the AE's Governance Interface: it publishes the AE's capabilities, which include both the description of the service that the AE can offer and the governance capabilities that can be used by other entities to govern the AE. It also accepts updated goal definitions and reports on the AE's performance.

Functionally, the Governor is responsible for planning how to fulfill the given goals, monitoring the goals and making sure that they are being fulfilled during service operation, for example by changing the goals for the subordinate AEs. If the AE still fails to achieve one of its goals, thus incurring an SLA violation, it is the duty of the Governor to inform the governing entities so that they can take appropriate action.

As an abstract example, Figure 3 shows four atomic and three composite autonomic entities. Notice that AE_3 is subordinate to both AE_6 and AE_7. The Governor of AE_3, Gov_3, needs to discern between the goals defined by the governors of AE_6 and AE_7, and make sure that AE_1 and AE_2's goals are updated accordingly.

On a low network level, an example of an Autonomic Entity could be a number of routers, which together provide the service of transporting packets through the network. No single router can by itself make any promises about delivering such a network-wide service. However, grouped with a Governor in an Autonomic Entity they are able to present a more complex capability than the sum of the individual routers. In practice, the Governor could be distributively implemented across the routers.

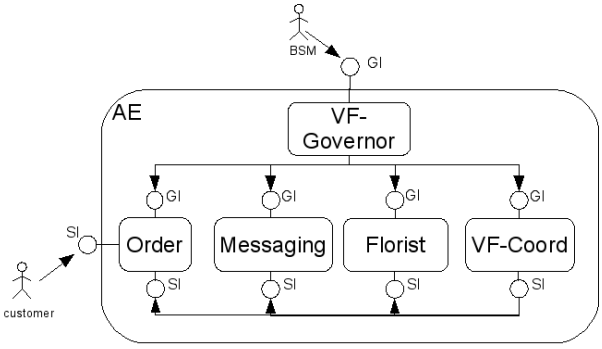

Figure 4. Autonomic Entities in the Virtual Flower scenario.

B. Architecture of the VF scenario

Let's consider a simplified version of the Virtual Flower scenario. Figure 4 shows the autonomic entity that implements the Virtual Flower service. A Business Service Manager interacts with the autonomic entity through its Governance Interface. Internally, this autonomic entity is composed of a Governor and four other autonomic entities. Order is the AE responsible for handling customer orders for virtual and real flowers; Messaging provides the MMS service; Florist delivers real bouquets to the customer's mother; and VF-Coord takes care of aggregating the aforementioned services so that MMS messages and real bouquets are timely delivered in accordance with the customer's wishes. Notice how part of the Service Interface (SI) implemented by the subordinate AEs are exposed by the overall autonomic entity (e.g. to the customer), while other parts of the SI are used internally by the VF-Coord.

To illustrate the approach, consider the following brief example: Before launching the service, the Business Service Manager (BSM) defines goals for the whole VF service which are refined by the Governor into goals for the subordinate AEs. As an example, the BSM defines a business goal saying that VF should be a able to accommodate 1000 customers. A refinement of this goal could include the (sub-)goal, for Order, to allow 1000 customer requests for flowers only in those cities where the service is to be launched. Similarly, a (sub-)goal for Florist could be to have in stock 1000 flowers for delivery during Mother's Day (MD). A couple of days before MD, Order tells VF-Governor of a sudden surge in the demand for a particular kind of flower in city C, which prompts VF-Governor to update Florist's goals accordingly. Whether such an update is possible is regulated by the SLA between the VF AE and Florist. If, on MD, an unexpected problem prevents Florist from distributing all flowers of kind T to city C, then VF-Governor tells Order to stop that offering (a new goal) and notifies the BSM about the SLA violation.

VI. Conclusions and Future Work

Ideally, business service developers and managers should only deal with high-level service descriptions and affect services by defining high-level goals for the systems that provide them. The business service manager should define what the service must achieve, not the concrete measures that the service implementation should take to achieve it. In this paper we have sketched a system architecture that exploits this distinction not only at the highest, business level, but also all the way down to network components. Central to our proposal is the notion of an autonomic entity which takes part in other autonomic entities in ways regulated by SLAs.

In order to materialize the proposed architecture we need to address the languages to be used for describing SLAs, AE capabilities and goals. Then we shall explore the process of goal refinement, which is of utmost interest for the implementation of the Governor component of a composite autonomic entity. Most approaches to goal refinement in the literature assume a rather static situation, where higher level goals are refined to lower level goals using possibly some kind of service description. For the Governor, the approach needs to be more dynamic and also take into account the current state of subordinate AEs as well as their SLAs.

Acknowledgments: The present work was funded by the GOPS and PBMAN3 projects. GOPS is sponsored by Vinnova.

References

[1] Coleman Parkes Research, "Transforming Operations Support Systems: Trends, Issues and Priorities of Communications Services Providers," 2007. [Online]. Available: http://www.amdocs.com/NR/rdonlyres/C18BDD2C-B2C1-49D9-8238-6D878822B081/0/OSS_Transformation_April_23.pdf

[2] W3C, "Web Services Description Language (WSDL) 1.1," 2001. [Online]. Available: http://www.w3.org/TR/wsdl

[3] P. Kotler, *Marketing Management, Analysis, Planning, Implementation, and Control*, 6th ed. Prentice Hall, Englewood Cliffs, NJ, 1988.

[4] C. Grönroos, *Service Management and Marketing: A Customer Relationship Management Approach*, 2nd ed. John Wiley & Sons, 2000.

[5] IBM Research, "Web Service Level Agreements," 2003. [Online]. Available: www.research.ibm.com/wsla/

[6] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer, IEEE*, vol. 36, no. 1, pp. 41–50, 2003.

[7] Enterprise Management Associates, "Practical Autonomic Computing: Roadmap to Self Managing Technology," Jan 2006. [Online]. Available: http://www-03.ibm.com/autonomic/pdfs/AC_PracticalRoadmapWhitepaper_051906.pdf

[8] D. Verma, "Simplifying Network Administration using Policy-Based Management," *Network, IEEE*, vol. 16, no. 2, pp. 20–26, Mar/Apr 2002.

[9] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, and G. Pavlou, "A Functional Solution for Goal-Oriented Policy Refinement," in *Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*. IEEE, June 2006, pp. 133–144.

[10] A. K. Bandara, E. C. Lupu, J. Moffett, and A. Russo, "A Goal-based Approach to Policy Refinement," in 5^{th} *IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY'04)*, 2004, pp. 229–239.

[11] R. Darimont and A. van Lamsweerde, "Formal Refinement Patterns for Goal-Driven Requirements Elaboration," in *Proc. of the 4th ACM SIGSOFT Symposium on Foundations of Software Engineering*, vol. 21, no. 6. ACM Press, Nov 1996, pp. 179–190.

[12] E. Lehtihet, H. Derbel, N. Agoulmine, Y. Ghamri-Doudane, and S. van der Meer, "Initial Approach Toward Self-configuration and Self-Optimization in IP Networks," in *Proc. of the 8th Int. Conf. on Management of MultiMedia Networks and Services (MMNS 2005)*, ser. LNCS, vol. 3754. Springer, 2005, pp. 371–382.

[13] E. Kavakli and P. Loucopoulos, *Information Modelling Methods and Methodologies*, 2005, ch. Goal Modelling in Requirements Engineering: Analysis and Critique of Current Methods, pp. 102–124.

[14] M. Burgess and S. Fagernes, "Promise Theory – a Model of Autonomous Objects for Pervasive Computing and Swarms," in *Int. Conference on Networking and Services*. IEEE Computer Society, 2006, p. 118.