Autonomic Re-Configuration of Pervasive Supervision Services

The CASCADAS Approach

Peter Deussen

Fraunhofer Institute for Open Communication Systems
Berlin, Germany

peter.deussen@fokus.fraunhofer.de

Matthias Baumgarten, Maurice Mulvenna

School of Computing & Mathematics
University of Ulster, Belfast, UK

m.baumgarten@ulster.ac.uk
md.mulvenna@ulster.ac.uk

Antonio Manzalini, Corrado Moiso

Telecom Italia
Torino, Italy

antonio.manzalini@telecomitalia.it
corrado.moiso@telecomitalia.it

***Abstract* — This paper describes a supervision system for autonomic distributed systems (e.g. a cloud computing environment). It is designed as a supplementary service and is structured as an ensemble of components that implement an autonomic control loop, which does not require any a priori knowledge on the structure of the supervised system. The architecture devised is highly modular and can be configured towards individual needs. In addition, the supervision system is able to re-configure itself according to the changes of the supervised system.**

Keywords: autonomic computing, pervasive supervision, autonomic communication, fault management, self-reconfiguration

I. Introduction

Networks today are composed of a wide variety of network elements that introduce a high degree of heterogeneity. Telecommunications Management Network (TMN) is a model defined by ITU-T for supervising open systems in a communications network, implementing the FCAPS (Fault, Configuration, Accounting, Performance and Security) management areas. The TMN model can hardly meet the requirements of future trends of Telecommunication, ICT and the Future Internet (e.g. emerging of Cloud Computing): as a matter of fact pervasive diffusion of powerful users' devices and systems heterogeneity are complicating the management and control of the whole network and service infrastructures. As such, there is a need for finding technology and solutions to simplify the management whilst also reducing, at the same time, operational expenses. These are the main directions of Autonomic Computing (AC) [10]. In 2001, IBM's manifesto on AC argued that, due to the increasing complexity of large-scale computing systems, computers and applications need to learn how to manage themselves in accordance with high-level policies specified by human operators. This vision took inspiration from the biological characteristics of the human Autonomic Nervous Systems, where the autonomic system makes decisions on its own, using high-level policies; it will constantly check and optimize its status and automatically adapt itself to changing conditions.

In AC, autonomic managers define a control loop (MAPE-K loop) performing the functions of monitoring, analyzing, planning and executing. Autonomic managers continuously monitor the managed system and handle events that need action to be taken. Nevertheless, in this model all autonomic "intelligence" is contained in the network of autonomic managers. This paper describes a novel approach where the supervision system is structured as an ensemble of communicating autonomic components, whose control loop does not require any a priori knowledge on the structure of the supervised system. The supervision system is also able to re-configure itself according to the changes of the configuration of the system under supervision and the environment; moreover it provides features for a more long-term orientated supervision mechanism that aims at predicting possible evolutions of the system under supervision.

The remainder of this work is structured as follows. Section II describes pervasive supervision as a model for the supervision of distributed systems. Section III describes the supervision proposed approach, its main components and the peculiarities on dynamic re-configuration of the supervision ensemble. Section IV discusses the novelties of the pervasive supervision approach in relation to the current state of the art. Section V introduces an experimental prototype validating the proposed approach. Section VI outlines potential application scenarios before Section VII gives some final remarks and possible future works.

II. Supervision Pervasion

This paper is proposing an innovative approach for the supervision of distributed systems empowered with autonomic capabilities that are structured as several interacting Autonomic Communication Elements (ACEs), an example of AC developed by IST Project CASCADAS [2]. Figure 1 shows the structure of an ACE highlighting individual ACE organs. Here, autonomic behavior is achieved mainly through the Facilitator, which utilizes self-models that describe the business logic each ACE implements as well as how an ACE reacts to internal/external events.

The CASCADAS approach takes the perspective that services are provided by (potentially large) ensembles of relatively simple entities that are realized as ACEs. The functional composition of these entities is done in a self-organized way, using the goal needed/goal achievable (GN/GA) protocol as the basic discovery protocol. Through GN/GA dynamic service composition is facilitated by matching a GN to available GAs, which both semantically describe the type of service or function an ACE desires and offers, respectively. After discovery, ACEs may establish contracts among each other to provide efficient and secure message exchange over multilateral communication channels, which collectively provide a specific service.

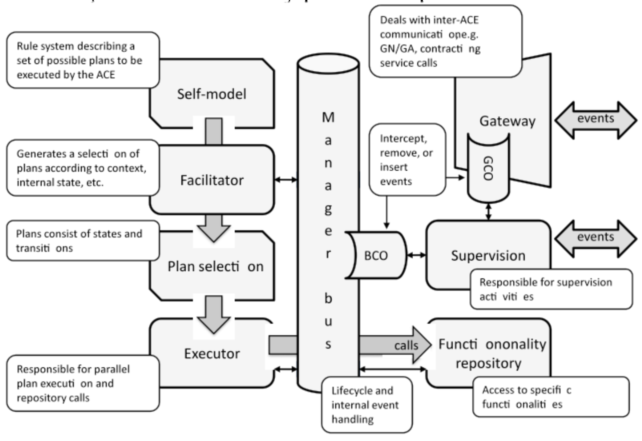

Rule system describing a set of possible plans to be executed by the ACE
Self-model
Generates a selection of plans according to context, internal state, etc.
Facilitator
Plans consist of states and transitions
Plan selection
Executor
Responsible for parallel plan execution and repository calls
Manager bus
Deals with inter-ACE communication e.g. GN/GA, contracting service calls
Gateway
GCO
events
Intercept, remove, or insert events
BCO
Supervision
events
Responsible for supervision activities
calls
Functionality repository
Access to specific functionalities
Lifecycle and internal event handling

Figure 1 Autonomic Communication Element (ACE)

Conforming to this architecture, supervision capabilities are also realized as ACEs and offered to the system in form of supplementary services. A supervisor itself is an ensemble of other ACEs, which are dynamically re-configured through the flexible discovery of required services (ACEs) and, subsequently, self-organized via the establishment of flexible contracts, which define the relation among individual components. Thus, basic supervision functions can be provided as default services to allow for e.g. event filtering, data correlation, and the processing of events that are produced by the supervised ACEs, and for autonomously elaborating corrective or optimization actions.

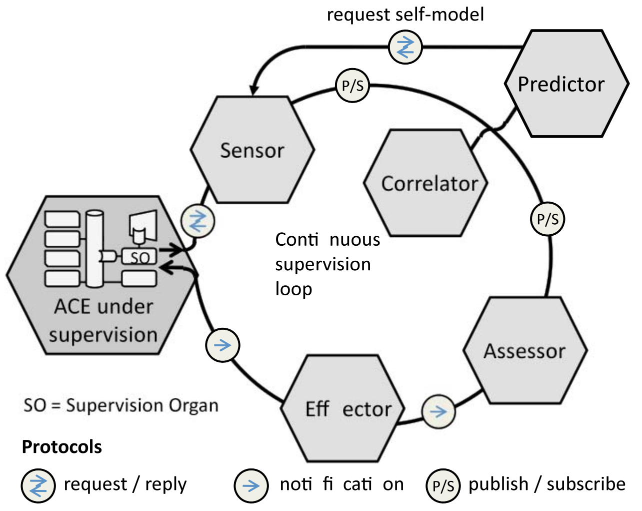


Figure 2 Organization of supervision components

Figure 2 depicts the architecture of a supervisor, where the use of the interaction protocols is indicated through individual arrows. For the sake of simplicity, only one instance of each component is shown, while in practice there will be always a number of supervised ACEs, sensors, effectors, etc. In addition to the basic supervision functions, supervisors may include other components in order to perform e.g. predictions, contingency planning, etc.

This service centric perspective allows formulating and implementing supervision infrastructures which go beyond the supervision of singular ACEs towards a more flexible and dynamic set of autonomic control (MAPE-like) loops which are able to adjust their own structure and function to the structure of the supervised system, thus forming enhanced service configurations which are able to secure themselves against faults, performance problems, etc. To emphasize the close relationship and organizational similarity between the supervised system and supervisor, those infrastructures were named *supervision pervasions*.

Therefore, the supervised ACEs and the supervisor ACEs work synergistically thus realizing a pervasive supervision where:

- The supervisors' autonomic behavior co-operatse with and complements the autonomic behavior of supervised ACEs.
- The structure of the supervisor is interwoven with the one of the supervised system and as such is also aligned with its changes.
- Supervision can be performed based on internal and external stimuli as well as in accordance to service-specific management policies.

III. Architecture and Components

A. *Basic Control Loops.*

As introduced in the previous section, the pervasive structure of a supervisor enhances the system under supervision by "completing" it through ACEs that implement basic supervision capabilities. Supervision is based on the establishment of an interface for observation and control mechanism between the system under supervision and the supervisor. ACEs are built upon an event-driven architecture where effectively all processes are controlled by events that are propagated through the internal communication bus (for intra-ACE communication between organs), and the gateway (for cross-ACE communication, i.e. GN/GA based discovery and contract based message exchange). Observing and controlling the bus and the gateway provides sufficient information to understand and to influence all ongoing processes within an ACE. Thus, effective observation and control can be performed by interception, removal, and insertion of events that are sent over the above mentioned communication channels. This functionality is provided by so-called supervision checker (SC) objects (a gateway checker object – GCO, and a bus checker object – BCO), which

- provide basic filtering functionalities to distinguish between events that are of interest for supervision, and those which are not,
- can be used to query specific information about the ACEs under supervision such as its current internal state or the plans which are currently executed,
- provide control functionalities to steer the internal processes of ACEs (see below), and
- establish a communication channel to sensors and effectors.

SC objects can be deployed at run-time by a supervisor (the deployment process is handled by the supervision organ of an ACE). This mechanism therefore provides a very flexible and generic way to set-up task specific interfaces for monitoring and actuation.

Sensors, correlators, and assessors form the analytical part of the supervision service and can be summarized as follows:

Sensors link the supervision system with the ACE under supervision by deploying SC objects to the ACEs under supervision and by establishing a dedicated communication channel for monitoring. Their goal is to translate events delivered by the SC objects into the internal message format used by the supervision infrastructure, and to distribute them to other components of the supervision infrastructure, in particular to correlators and predictors. A description of the communication mechanisms internal to supervision pervasions is given in Section III.C

Correlators are responsible to aggregate monitored data from distributed sources and to correlate them with other information in order to extract meaningful indicators of the current condition of the system under supervision.

Assessors make assumptions on the current (or future) system health based on the output of correlators, and invoke associated effectors if necessary.

Effectors are responsible to distribute contingency actions to the SC objects of the various ACEs under supervision, where they are used to steer the execution of the ACE under supervision.

The reactive part can be further extended by additional components such as planners. A detailed discussion of such functionality is however outside the scope of this paper; the interested reader is referred to [3].

B. Long-term supervision

Special considerations have to be made if long-term supervision and adaptation is desired. Applications of such supervision require a temporal aspect to be taken into account that can otherwise be discarded. In relation to ACEs, relevant concepts to be analyzed include the detection of drift behavior as well as the modeling, monitoring and prediction of events, states or situation an ACE can step into or reach in the future. Thus, the general objective of long-term supervision components can be summarized as to observe and analyze numerical as well as symbolic concepts over time in order to predict future properties, behavior and situations. This would ultimately allow counteracting any form of behavior that could potentially lead to critical or undesired states before they are actually reached / occur.

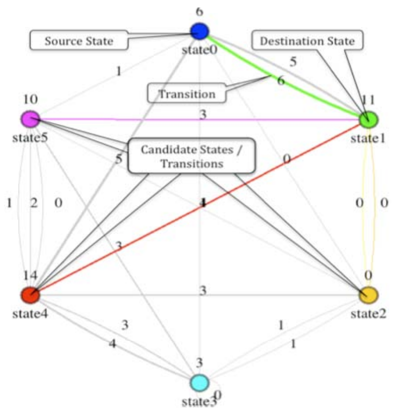
For that, three types of services have been devised that are each capable of performing a long-term supervision task, which can be requested by a supervisor in the same fashion as any of the standardized supervision services:

Drift Analysers (DA) allow facilitating flexible long-term supervision by analyzing and forecasting numerical concepts that reflect the boundaries a system should operate in or, alternatively, an ideal state of operation that reflects the most optimum performance. Such numerical properties can refer to business goals or to operational parameters of a supervised ACE and its environment. The rationale of DA's is that underlying numerical concepts are likely to change constantly over time and that there is a strong desire to keep them within certain boundaries that reflect the correct or optimum behavior of the system. Thus, detecting if a parameter slowly drifts towards its operational boundaries or away from its ideal state of operation would allow corresponding countermeasures to be applied before any more serious "violation" occurs.

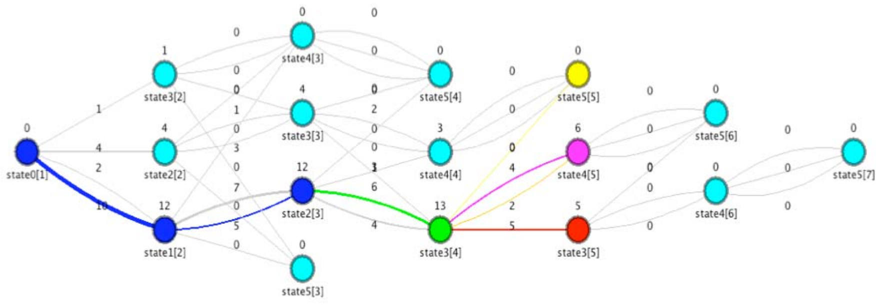
Event Predictors (EP) predict the time window in which a certain event is most likely to occur next. Based on the monitoring of past events it computes static as well as dynamic time statements around which a given type of event may reoccur. The static service provides the mean distance between events as its prediction, whereas the dynamic service is based on the time of calculation/request, thus taking into account the time elapsed since that last event has been registered. This service is of particular interest for periodic services and it would allow for both, the validation of correct behavior (e.g. an event should occur periodically) or, alternatively, for the detection of fraudulent behavior (e.g. if an event occurs outside of its predicted time window).

State Predictors (SP) are aiming at observing and predicting the execution logic of ACEs as represented by their self-models. In particular, (a) they allow monitoring the execution of ACEs, (b) build an execution model based on these observations and (c) based on an observed state change predict potential next states / transitions. SP's operate based on the observation of past and/or mass behavior as inspired by e.g. [8] For instance an SP could monitor the execution of all ACEs (services) of a certain type and would, over time, construct a model that reflects how this particular type of service operates. If a new instance of this type of service is requested then the associated SP could provide recommendations of how the service should perform or behave, which would be based on the successful execution of past instances of the same service type. This would allow preventing illegal or dangerous behavior of an ACE and would also allow for the optimization of service execution in the long term. Based on the concept of ACE's and the associated self-model two distinct types of SP's have been designed. Firstly, a meshed SP (MSP) that only takes into account a single state change thus discarding all preceding activity and, secondly, the directed SP (DSP) that takes into account the entire execution path from a defined start to a defined end state. Thus the former can be used to validate stateless operational behavior as defined by individual states and transitions of a given selfmodel whereas the latter is more appropriate to analyze business behavior as reflected by service compositions where the path of execution is relevant.

Figure 3 (a) and (b) show an example execution model for the MSP and the DSP respectively. As can be seen the full path of execution is maintained in (b) whereas (a) only takes into account a single state change as described by the triple source state – destination state – transition traversed. Based on this and the properties of each state/transition, which reflect how often they have been visited or traversed in the past by the same service type, the likelihood of



(a)



(b)

Figure 3 State Predictor Variants – (a) MSP, (b) DSP

states/transitions to occur next can be computed. Thus an SP indicates how a service is most likely to continue based on its past behavior or based on the behavior of other instances of the same service. Such information can be directly used to e.g. initialize subsequent behavior, provide system guidance, detect system violations etc.

Considering the modular architecture of the supervision architecture both of the state predictor models could be combined into a single more powerful supervision service capable of supervising a complex service at various levels of granularity. Furthermore, within the overall supervision control flow, the outputs of SP components, jointly with the correlators' outputs, can be used by assessors, to make assumptions on the current (or future) system health, and invoke e.g. a Planner component if necessary.

C. *Automatic Configuration of Pervasion*

The configuration of a supervision pervasion is done in a number of steps:

Contracting: As supervision is a supplementary service to be used by ACE ensembles that provide service(s) to a user (or another ACE ensemble). The first step consists in contracting all the components (sensors, effectors, correlators, etc.) to be involved in the supervision pervasion. This is done via a special *controller* ACE, which commits a supervision contract with the system to be supervised. Then the controller discovers the remaining ACEs, and sets up another contract for communication within the supervision pervasion. Finally, it obtains relevant configuration information (in particular the identities and addresses of the ACEs to be supervised) which are necessary to establish an SC based monitoring and control channel as discussed next.

SC Deployment: SC objects (GCO and BCO) are deployed by sensors. To this end, a temporary contract is established between sensor and an ACE at which an SC object has to be deployed. The SC object itself is sent as part of a specific message, which is handled by the supervision organ of the ACE to be supervised. After deployment, the SC objects establish a connection to the sensor and effector.

Subscription: A publish/subscribe based interaction mechanism is used as a general communication paradigm within the supervision pervasion. Each component of the pervasion publishes a set of so-called *topics*, i.e. categories of information that this component is able to provide. Other components can *subscribe* to these topics in order to be notified if new data is published. For instance, correlators as well as state predictors subscribe to information *published* by sensors (where the specific selection of topics obviously depends on the supervised system and the supervision tasks to be performed). Hence, the publish/subscribe protocol provides a data-flow driven group communication schema, where groups are defined by topics.

Re-configuration: Changes in the architectural structure of the supervised system can be detected in several ways. The most generic approach is to use the BCO to intercept events steering the reconfiguration (contract cancellation, discovery, new contract establishments, etc.) on the internal communication bus, and to forward this information via sensors to a specialized correlator. In some cases it is however easier to simply notify the supervisor ACEs about an ongoing reconfiguration.

The supervision pervasion reacts to the reconfiguration of the ACE ensemble under supervision by performing reconfiguration operation on itself. In particular it removes SC objects from ACEs which are not longer part of the supervised system, and deploys new once in newly introduced ACEs. Moreover, it adapts its internal structure to reflect the new architecture of the supervised ensemble using the mechanisms (contracting, subscription) as described above.

Termination: Supervision activities are terminated (or suspended in the case of long-term supervision) when the ACE ensemble under supervision decides to break the supervision contract, which is usually the case when the service contract grouping this ensemble is terminated. The controller ACE notifies all components of the supervision pervasion, and breaks the contract between them. As for the long-term supervision components, a contract can be re-instantiated to the same statefull supervision object.

IV. ADVANCES BEYOND THE STATE OF THE ART

ACE based systems provide services by means of interactions of a probably large distributed set of ACEs with a dynamically adapted interaction structure and task diversification [6][13]. Hence, the basic assumption underlying to traditional supervision approaches (see for instance [11][5][9][1][7]) that the system under supervision maintains a static architectural structure (i.e. does not perform run-time architecture adaptations) is not longer valid for ACE based services.

This notion of a service providing system makes a novel approach for the formulation and deployment of autonomic control loops necessary, which do not require any a priori knowledge on the structure of the supervised system. In order to address this need, the pervasive supervision approach includes a novel scheme to set-up those control loops that are based on the interaction of various ACEs forming a supervision ensemble. Evidently, the structure of the supervision pervasion adapts itself dynamically to the changes of the actual structure of the supervised system.

Another novel achievement is the use of a common technological basis (namely the ACE software component) both for the system under supervision and the supervision system, which promotes self-similarity among components. This has a number of advantages: Firstly, the introduction of additional technologies does always increase the complexity of a system, hence a reduction of operational efforts by using a supervision system technologically different from the supervision system is at least questionable. On the other hand, for the supervision system described in this paper, a number of basic functions necessary for supervision already had been provided by the ACE component platform itself: A service discovery and contracting mechanism based on the GN/GA protocol, supporting dynamic adaptation as described above, the separation between process logic (provided by ACE self-models) and function implementation (provided by ACE functional repositories), as well as the built-in monitoring and control mechanisms the ACE supervision organ offers. Note that generic supervision tasks (such as liveliness validation as described in the case study in the Section V.B) can be applied to the components of a supervision pervasion as well. Thus, self-supervision can be performed through the proposed approach.

The temporal supervision of quantitative as well as symbolic based parameters and behavior is provided as a set of long term supervision components: a more complex supervision ensemble can be enhanced through a flexible configuration / orchestration of these components with the one offering basic supervision features. These components have been specialized to work with the ACE model and its declarative execution logic (i.e., based on self-models). For instance, StatePredictors have been specifically designed to address individual features of the ACE self-model / plan philosophy to model detailed ACE behavior over time and subsequently provide detailed predictions of potential future behavior.

V. EXPERIMENTAL VALIDATION

A. *Supervision prototype*

A prototype of the pervasive supervision has been implemented: it is integrated with the Cascadas ACE Toolkit [2] and it is available as open source. Within the prototype, each of the supervision components is a separate ACE. In particular, the supervision toolkit includes a set of generic ACEs, one for each component thus providing basic and long-term supervision features (i.e., sensor, effector, correlator, assessor, drift analyser, planner, state predictor, event predictor). These components must be specialized in order to implement the logic to supervise specific systems requirements. Moreover, the toolkit includes communication and interaction protocols (request/reply, notification, publish/subscribe) to set-up and (re-)configure a supervision pervasion, and its components.

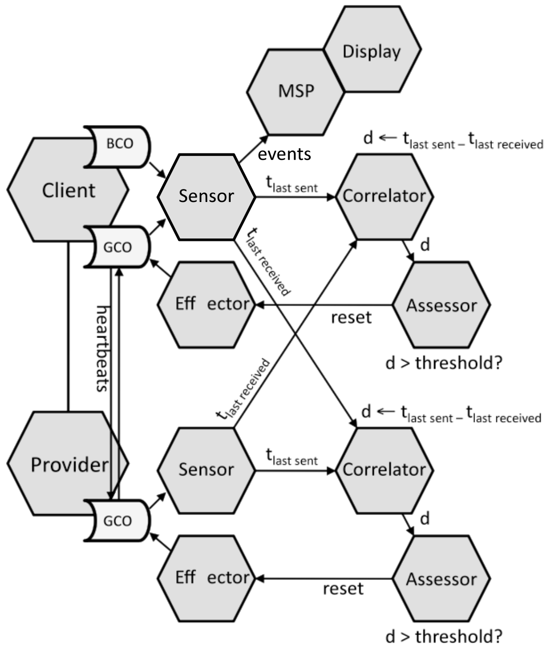


Figure 4 Dynamic Reconfiguration Scenario

B. *Service scenarios*

The described supervision approach has been applied to supervise a video service implemented as a set of distributed ACEs. Pervasive supervision has been introduced in order to handle failures of ACEs implementing the video client and one of several available video providers (see Figure 4).

Subject of the supervision activities is the liveliness of the contract between the client ACE and the provider ACE. Supervision is done by issuing an exchange of heartbeat signals between these two ACEs, hence, if the contract is malfunctioning in one or both directions, this fault can be detected by comparing the time stamps of sending and receiving a heartbeat signal. Heartbeats are handled by GCOs injected into the supervised ACEs, hence the liveliness validation mechanism is transparent to self-models of supervised ACEs where dynamic reconfiguration of the supervision pervasion takes place if the provider changes. In this case, the structure depicted in Figure 4 is automatically adapted to work with the new service contracts.

For long-term supervision, the MSP computes the probability of subsequent states based on observed state changes within the execution logic of the system under supervision (in our case, the video player ACE). Hence, in the above scenario predictions are related the probability that a certain channel will be selected or the probability that a fault will occur. The former is of particular interest, as it would allow a system to determine the channel that has been selected most in the past, which in turn could be selected if the currently selected channel becomes unavailable or if the selection procedure develops a fault.

VI. Application Scenarios

A possible application scenario for the pervasive supervision is the management of computing clouds, i.e., infrastructures consisting of reliable services delivered through next-generation data centers built to compute and store virtualization technologies and data respectively. Autonomic supervision can support load balancing, dynamic configuration, fault detection and recovery as well as SLAs fulfillment. Moreover, the autonomic supervision could also be extended to the management of end-users devices, which dynamically use the services executed by the cloud. For example, an important task is periodically informing supervisors about execution check points to allow dynamic re-configuration of the execution power. Furthermore, supervision strategies should also face the problem of handling the proper number of data duplications for the requested persistency or for the performance needs in data access.

A second application scenario is the supervision of distributed service provisioning platforms, where different actors can develop, provide, connect and interact, in a secure and reliable way, for selling, buying, negotiating, exchanging and trading any content, information, services and service components. In such a context where components are dynamically negotiated and aggregated, in addition to equip service components with self-management capabilities, pervasive supervision could be used by an actor creating a service by aggregating a cluster of components to enforce service-specific management policies, or by a provider of service components to supervise the instances of a service

VII. Conclusions and Final Remarks

One of the most serious technological challenges of future Telecommunication, ICT and Internet will be interconnection and management of a huge amount of heterogeneous systems and small devices tied together in networks of networks. Autonomic Computing already argued that, due to the increasing complexity of large-scale computing systems, computers and applications need to learn how to manage themselves in accordance to high-level policies as specified by human operators. Nevertheless current autonomic solutions don’t exploit the real biological metaphor in distributed systems. This paper presented a novel autonomic supervision system, which is structured as an ensemble of components that implement an autonomic control loop, which does not require any a priori knowledge on the structure of the supervised system. The architecture devised is highly modular and can be configured towards individual needs. In addition, the supervision system is able to re-configure itself according to the changes of the supervised system. The solution was experimentally validated by the development of a prototype, which has been made available as open source.

A possible evolution of the prototype would include the definition of the management policies through a specific language. The long-term supervision components could be enhanced to facilitate the dynamic orchestration into more advanced hierarchical supervision pervasions.

The pervasive supervision approach is mainly orientated to the supervision of clusters of ACEs implementing specific services in accordance to service-specific management policies. Considerations on how to extend the basic supervision pervasions that are described in this paper to a global supervision backbone have been made in [4].

Acknowledgment

Authors would like to acknowledge European Commission for funding the IP CASCADAS (IST-027807) (FET Proactive Initiative, IST-2004-2.3.4 Situated and Autonomic Communications).

References

[1] Baresi L., Ghezzi C., Guinea S.; “Towards Self-healing Compositions of Services”, Proc. of 1st Conf. on PRIciples of Software Engineering (PRISE’04), pp. 11 – 20, 2004.

[2] CASCADAS Project, ACE Toolkit open source, http://sourceforge.net/projects/acetoolkit/.

[3] Deussen P. H., "Model Based Reactive Planning and Prediction for Autonomic Systems", INSERTech '07: Proceedings of the 2007 Workshop on INnovative SERvice Technologies: 1-10, 2007.

[4] Deussen P. H., "Supervision of Autonommic Systems --- Tutorial", Proc. Budapest Tutorial and Workshop on Autonomic Communications and Component-ware, 2008. Published on CD.

[5] Deussen P. H., Valetto G., Din G., Kivimaki T., Heikkinen S., Rocha A.; Continuous On-Line Validation for OptimizedService Management, in EURESCOM Summit 2002, Hiedelberg, Germany, October 21–24, 2002.

[6] Devescovi D., Di Nitto E., Dubois D., Mirandola R.; Self-organization algorithms for autonomic systems in the SelfLet approach. In *Proceedings of the 1st international Conference on Autonomic Computing and Communication Systems*. Autonomics, vol. 302, Institute for Computer Sciences Social-Informatics and Telecommunications Engineering), pp. 1 – 10, 2007.

[7] Garlan D., Cheng S., Huang A., Schmerl B., Steenkiste P.; “Rainbow: Architecture-based Self-adaptation with Reusable Infrastructure”, IEEE Computer, 37(10):46-54, Oct. 2004.

[8] Han J., Pei J.; “Mining Frequent Patterns by Pattern-Growth: Methodology and Implications”; ACM SIGKDD Explorations Newsletter, Volume 2, Issue 2, , pp 14 – 20, 2000.

[9] Kaiser G., Parekh J., Gross P., Valetto G.; “Retrofitting Autonomic Capabilities onto Legacy Systems”; Journal of Cluster Computing, Vol. 9, No. 2, 2006, pp. 141–159.

[10] Kephart J., Chess D.; “The Vision of Autonomic Computing”, IEEE Computer, Vol. 36, No. 1, 2003, pp. 41–52.

[11] Knight J. C., Sullivan K. J., Elder M. C., Wang C.; “Survivability Architectures: Issues and Approaches”; In Proceedings of the DARPA Information Survivability Conference and Exposition, IEEE Computer Society Press, pp. 157 – 171, 2000.

[12] Manzalini A., Zambonelli F., Baresi L., Di Ferdinando A.; “The CASCADAS Framework for Autonomic Communications”; “Autonomic Communication”; A. Vasilakos, M. Parashar, S. Karnouskos, W. Pedrycz (Eds.), Springer book, 2009.

[13] Shackleton M., Saffre F., Tateson R., Bonsma E., Roadknight C.: “Autonomic Computing for Pervasive ICT - A Whole-System Perspective”; *BT Technology Journal* 22, 3, pp. 191-199, 2004.