Distribution and Self-Adaptation of a Framework for Dynamic Adaptation of Services

Françoise André, Erwan Daubert, Guillaume Gauvrit
IRISA / INRIA
Campus de Beaulieu, 35042 Rennes cedex, France
{Francoise.Andre, Erwan.Daubert, Guillaume.Gauvrit}@irisa.fr

***Abstract*—The dynamism and scale of the infrastructure of the Internet of Services bring new needs to build autonomous services. These services have to be able to self-adapt to the variation of the environment. Moreover, these adaptations may span across multiple services and thus have to be coordinated, without breaking their autonomy. To this end we describe in this paper the approach we have chosen for SAFDIS, a framework to make coordinated adaptations of services. In this presentation, a particular emphasis is made on the distribution of the framework and how it helps to coordinate distributed adaptation. Benefits from the self-adaptation of the framework itself are also presented.**

Keywords-Dynamic Adaptation; Distributed Services; Distributed Adaptation; Self-Adaptation of Adaptation Framework.

I. INTRODUCTION

The underlying computing infrastructure for the Internet of Services is characterized by its very large scale, heterogeneity and dynamic nature. The system scale is to be measured in terms of number of users, services, computers and geographical wingspan. The heterogeneity comes from its spreading on multiple sites in multiple administrative domains providing very different computers, devices and network connections. Its dynamic nature results from a number of factors such as Internet node volatility (due to computer or network failures, voluntarily connections and disconnections), services evolution (services appearing, disappearing, being modified) and varying demands depending on human being activities.

In a world of services in which more and more personal, business, scientific and industrial activities rely on them, it is essential to guarantee the high availability of services despite failures or changes in the underlying continuously evolving execution environment. Moreover, providing quality of service (QoS) is important considering the number of services related to legal and commercial aspects.

To take into account this dynamism our objective is to design and implement systems that are context aware and able to adapt applications and services at run-time.

The task of making software adaptable is very cumbersome and encompasses different levels:

- At user or business level, processes may need to be reorganized when some services cannot meet their Service Level Agreement (SLA).
- At service composition level, applications may have to change dynamically their configuration in order to take into account new needs from the business level or new constraints from the services and the infrastructure level. At this level, most of the applications are distributed and there is a strong need for *coordinated adaptation*.
- At infrastructure level, state of resources (networks, processors, memory, etc.) have to be taken into account by service execution engines in order to make a clever use of these resources, such as taking into account available resources and energy consumption. At this level there is a strong requirement for *cooperation* with the underlying operating system.

Moreover, the adaptations at these different levels need to be coordinated.

So, our main challenge is to build a generic framework for self-adaptation of services and service based applications. The basic steps of an adaptation framework are Monitoring, Analysis, Planning and Execution, following the MAPE model proposed in [1]. We intend to improve this basic framework by refining each step of the MAPE model, in particular by providing elements that cope with the distribution of the application and the underlying infrastructure. The adaptation system can itself be distributed for the purpose of scalability or to better match the heterogeneity of the environment. Moreover, it can be adaptable, allowing to take into account unforeseen situations.

Our system called SAFDIS for Self-Adaptation For DIstributed Services fully exploits the advantages of the framework concept [2]. It gives a frame, paradigms and rules to develop and implement adaptation mechanisms, as well as the liberty and the flexibility for the developer to specialize its system according to its specific needs. Using this framework, the task of developing concrete adaptation systems for some applications, services or infrastructures will be facilitated as many of the different elements that may be composed in adaptation systems are exposed, their

 ISBN: 978-1-61208-124-3

interfaces clearly defined, the relationships between them coherently specified. Our SAFDIS framework is build as a set of services, providing functionalities useful to build an adaptation system. Not all functionalities are necessarily needed for each instantiation of an adaptation system. For instance we provide a *negotiator* service to negotiate the adaptation decisions when SAFDIS is distributed on several nodes; this service is not useful when SAFDIS is build as a unique centralized adaptation system.

The following sections present the advantages resulting from the design of our framework. The next section gives an overview of the SAFDIS framework. Section III presents how the distribution is handled in our framework and its advantages. Then, Section IV presents some advantages of having an adaptation framework that is self-adaptable. Finally, Section V presents some related-works and Section VI concludes this paper.

II. SAFDIS: Self-Adaptation For DIstributed Services

Our framework for self-adaptation of distributed services SAFDIS [3] is divided into the four main phases of the *MAPE* model. *Monitoring* is the observation function to detect changes that imply adaptation. When a change is detected, the monitoring phase triggers the *analysis* to analyze it and find an adaptation strategy if it is required. Then this strategy is given to the *planning* phase to compute a schedule of actions that will implement the strategy. The last step is the *execution* of the schedule to reconfigure the system (application, services and the environment).

Our framework is able to work at different levels ranging from a single service, a composition of services for one application, to several applications. Each application can be executed on a set of heterogeneous platforms themselves on a distributed and heterogeneous infrastructure (OS and hardware). Therefore in order to adapt a set of applications, it may be necessary to interact with these platforms and some specific (maybe all) execution nodes which represent only a part of the infrastructure. With SAFDIS, it is possible to monitor the different levels according to the implementation of the available probes and adapt them depending on the adaptation actions (Figure 1).

To cope with the distributed environment, our framework can itself be distributed using multiple autonomous and cooperating instances. An instance has to be deployed on each of the service oriented platforms hosting a least an adaptive service using SAFDIS. Our framework is also fully decentralized, meaning that there are no instances with privileges or special purposes. This design avoids single points of failure and makes the framework scalable.

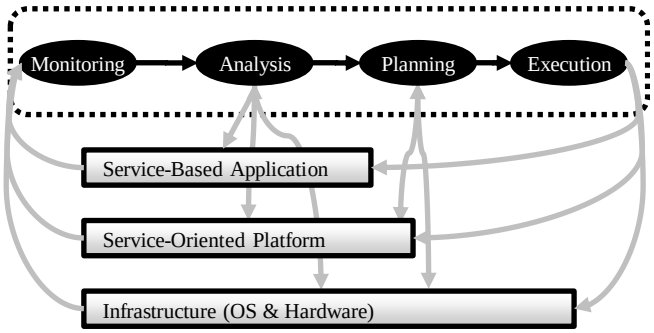

Figure 1. Multi-level Adaptation

In the following subsections, we present each phase of the MAPE model and some of their characteristics in the context of our framework and our current implementation.

A. Monitoring

The monitoring phase is used to provide an informative and dynamic view of the adaptive entity and its environment to the other phases of SAFDIS. Thus, it is the starting point of every adaptation undertaken. It builds one local view per instance of SAFDIS picking relevant information from the service-oriented platform, the adaptive services, the operating system and the hardware. SAFDIS can probe both passively or actively the system to generate events and update the view. The pieces of information that have to be gathered are specified by the other phases of the framework.

B. Analysis

The analysis phase of a MAPE adaptation system has two goals. The first goal is to identify situations needing an adaptation. It listens to updates of the view of the system pictured by the Monitoring phase. Then it analyzes the changes in the system and decides if an adaptation is needed consecutively to this change. The second goal of the analysis phase is to build an adaptation strategy when a need arises. A strategy defines which elements (parameters, functions...) need to be adapted and how.

Within our SAFDIS implementation, the analysis phase takes decisions with multiple temporal scopes. This gives the ability to either react fast or to take proactive decisions for the long term. This implies the ability to analyze the context with a variable depth of reasoning. Our implementation of the SAFDIS analysis phase also distributes and decentralizes its analysis process to spread the computational load and make the analysis process scalable.

C. Planning

The planning phase seeks the set of actions (the plan) needed to adapt the system according to the strategy chosen

 ISBN: 978-1-61208-124-3

by the analysis phase. It also schedules the selected actions to ensure a coherent and efficient execution of the adaptation.

Until now the planning phase has received little attention in the context of adaptation and in many cases the planning algorithms used produce simple total orderings of actions. In these cases, the result is not very efficient since the execution may take more time than necessary as the actions are sequentially executed. Moreover in distributed environments, where actions can be asynchronous, some synchronization actions explicitly have to be added to ensure the predefined sequential order.

The planning topic is a well known subject in AI research works and many algorithms already exist in that field to produce efficient schedules. With our SAFDIS framework, the planning phase is able to reuse these algorithms. The resulting plan of actions can have actions that can be executed in parallel.

D. Execution

Once the planning phase has computed the action plan corresponding to the strategy, the execution phase is called to perform the adaptation actions on the service, the application or the environment. These actions are application, platform, OS or hardware specific. That's why, with SAFDIS, we have introduced two kind of actions. The first kind called *concrete actions* corresponds to the action implementations which are specific to the adapted element. The second kind called *abstract actions* constitutes an abstraction of the concrete actions. This allows the planning phase to work with abstract actions without taking into account their specific implementations and doing so to build generic action plans.

III. DISTRIBUTION

The SAFDIS framework is meant to be distributed in the same way the applications it adapts are distributed. When deployed, it is composed of multiple autonomous instances, each one in charge of the adaptation of the services deployed on its platform. However, these autonomous instances cooperate in order to coordinate the adaptations involving distributed elements.

Moreover it is also fully decentralized: there are no instances with privileges or special purposes, therefore there is no single point of failure. When an instance fails, for example from a hardware failure or power issue, the other instances can continue to operate normally, even though the adaptations related to the failed instance will fail.

The absence of an instance with a central role avoids the bottleneck problems that could arise from this role. Also, there is no need for a server dedicated to the management of the adaptation of the services.

For example, let's consider the adaptive services S_a, S_b and S_c respectively executed in the service-oriented platforms P_a, P_b and P_c on the execution nodes N_a, N_b and N_c. The service S_a uses the services of S_b and S_c. The three services are using SAFDIS in order to be adaptive, thus there is an instance of SAFDIS on each service-oriented platform: I_a, I_b and I_c. If I_a and I_b are involved in an adaptation on S_a and S_b whereas at the same time the node N_c sees its CPU and memory load decrease, I_c can without having to ask the other SAFDIS instances make the adaptation decision consisting in allocating more memory to S_c thus improving the quality of this service. Both adaptations, the one concerning S_a and S_b and the one concerning S_c, can be executed in parallel.

When deployed, SAFDIS is a set of distributed instances. Each instance is a set of services which fulfill the four main functions of SAFDIS: monitoring, analysis, planning and execution. The cooperation between the instances is done at the level of the services. For example, analysis services cooperate among themselves but none of them interact with the monitoring and planning services that are outside of its SAFDIS instance. This respects the separation of the adaptation process into four phases.

As there is one instance of SAFDIS on each service oriented platform, each monitoring service is in charge of monitoring its execution node, the platform itself, the services deployed on its platform and the SAFDIS instance it is part of. The other services of SAFDIS always send their requests of information to the local monitoring service. This monitoring service then retrieves the information from another monitoring service if it doesn't have it. The connection between the various monitoring services are made on demand, using a service registry.

Instead of trying to picture a global view of every elements contributing to the application, which would consume communication resources and not be scalable, SAFDIS pictures multiple local views. This allows to spread the computation load of the analysis on the execution nodes related to the adaptations. But this means that the instances of the analysis component have to take decisions based on partial knowledge of the system. This knowledge alone is not always enough to decide of adaptation strategies. Thus, the analysis instances use negotiation mechanisms in order for them to cooperate in the decision process.

The analysis services cooperate by negotiating strategies. A strategy is initiated by an analysis instance and then negotiated with the other analysis instances that are (or may be) impacted in the adaptation. Those instances can enhance the proposed strategy. They may in turn involve other analysis instances into the negotiation process.

In the previous example involving three adaptive services and SAFDIS instances, if I_a chooses a strategy and negotiates it with I_b and I_c, the last two instances analyze in parallel the portion of the strategy requiring their involvement.

 ISBN: 978-1-61208-124-3

I_a takes the final decision to apply the negotiated strategy or to abandon it.

Each analysis service uses three sub-services: a decision maker service in charge of the reasoning and decision making process and a pair of services to handle the negotiation: the negotiation manager and the negotiator services. Each negotiator handles one to one negotiations while the negotiation manager divides the negotiations involving more than two peers into multiple negotiations involving two peers and coordinates them. This design was chosen to enforce a separation of concerns and to ease potential upgrades when SAFDIS is deployed. In our implementation of the framework, the negotiation protocol used by the negotiation manager and negotiator services is the Iterated Contract Net Protocol [4]. However, this is transparent for the decision maker service, so other negotiation protocols could be used.

Once a strategy is negotiated, it is sent to the planning service that is in charge to make a plan of actions to implement it. As said in II, due to the planning algorithms used in SAFDIS, some actions can be scheduled to be executed parallel.

So, in the last phase of the adaptation process which is the effective execution of the adaptation, the distributed aspect of the adaptation process is again emphasized as the actions that can be executed in parallel are executed in parallel, which in a distributed context improves the execution time of the adaptation.

Overall, the distribution of the analysis process and the distribution and the parallelism of the execution phase allows our framework to spread the computation load of the adaptation process and to gain time in this process.

IV. Self-Adaptation of the adaptation system

As our SAFDIS framework is developed as a service-oriented application it can itself be adapted as any other application, using its own mechanisms. In this section we present this self-adaptation property and detail its use for the planning phase.

The current implementation of each MAPE phase of SAFDIS can be dynamically replaced by a new implementation. At deployment a first implementation for each phase is chosen by the expert. If the expert think that there is no chance that the context will necessitate his choice to be called into question, SAFDIS will remain the same a long time. But the expert can predict that his initial choice may not be the best one in every circumstances or if some changes appear in the future. In that case he can add policies to the SAFDIS framework he initially used to adapt the application, in order to adapt the framework itself.

Thanks to the use of a service based adaptation framework design, the need to stop the application and its execution environment when changing the adaptation system is avoided. The adaptation system itself needs not to be completely stopped as a phase may be changed without affecting the others. The expert only has to have foreseen other implementations for the phase subject to potential changes and the policies to decide the change. Of course the new implementation should respect the services specifications, especially be conform with the interfaces defined in our framework to ensure the communication between the MAPE phases. At run-time, the new implementation will be looked for in the services repository, started and then the previous one will be stopped. Interconnections between phases are automatically done through the interfaces, without the help of the expert.

Portions of a phase are also self-adaptable without having to replace the complete phase. This is the case for instance of the negotiation part of the analysis phase.

To illustrate this self-adaptation property, we detail below the way it has been conceived and developed in our current prototype for the Planning phase.

As adaptation is performed at run-time, the time needed to actually perform the adaptation have to be minimized. Therefore, planning is an important phase of the MAPE model. It chooses the actions necessary to properly apply the adaptation strategy, and schedules the actions to ensure the consistency of the adaptation execution.

A simple planning algorithm as used by most adaptation systems uses a static total ordering between all possible actions and leads to a sequential schedule.

For example, if we consider the three possible actions *stop service, update service* and *start service* and an order that imposes that whatever the number of services to change all the *stops* must be done before the *updates* and all *updates* before the *starts*, this adaptation method will maximize the time during which all the services are unavailable, and also consume more time than needed in case some actions may have been processed in parallel.

Moreover, if the adaptation takes place on a distributed and asynchronous environment, explicit synchronization operations should be added to enforce the respect of the schedule between the different parts of the actions that have to be executed on different platforms.

Research works on planning methods such as Artificial Intelligence planning, Motion planning or Control theory, have produced algorithms [5], [6] that overcome these limitations, but without applying them in the context of dynamic adaptation. In SAFDIS, we propose an architecture for the planning phase (called F4Plan for "Framework For Planning adaptation" [7]) that offers the possibility to use, according to the needs, one of these algorithms. Moreover this architecture includes a set of language translators that allow to translate the possible output languages from the analyze phase (languages used to describe the current configuration

 ISBN: 978-1-61208-124-3

of the application and the target configuration) into the different input languages used by the planning algorithms.

The self adaptation of the planning phase consists in choosing the most suitable planning algorithm according to policies defined by the expert of adaptation. These policies are based on some non-functional constraints defined by the system such as the system overload or the duration which may be acceptable in order to apply the strategy but they also take into account the strategies sent by the analysis phase. For example, if the strategy comes from a reasoning engine that is used to do local adaptations to solve local problems, such as the one we use to make reactive decisions based on event-conditions-actions rules, it is not necessary to use a planning algorithm that searches for a parallel schedule. Indeed, there will probably be relatively few actions to schedule and all of them should be executed on the local node. In that case the simplest planning algorithm is convenient, being able to plan the strategy as quickly as possible, thus minimizing the time spent in the planning phase.

At the opposite, if the strategy comes from a reasoning engine based on utility functions such as the we use to make proactive decisions to do wide adaptations impacting the distributed system, it is interesting to use a planning algorithm able to plan a strategy as efficiently as possible. This planning algorithm should take into account several constraints for example the potential asynchronism between actions and the amount of resources that will be used during the execution phase.

So, the modularity and the service based design of our SAFDIS framework allows a great flexibility in the conception of an adaptive system. We do not neglect of course the task of the adaptation expert who has to conceive the adaptation policies.

V. Related Works

Today research works on autonomic computing aim mainly to build autonomic components but very few works consider building autonomic services or autonomic service-based applications. Among these works most of them as [8], [9] integrate the adaptation process into the components or services. Each element constitutes an autonomous element of the system and it doesn't interact with other elements to coordinate more complex adaptations. So, these solutions are not appropriate to manage wider adaptation spanning over multiple services constituting one or more applications. Meanwhile in [8], the authors add some predefined high-level adaptation components to be able to adapt a set of elements constituting an application. But this possibility is restricted to some specific cases for example to resource management or application deployment.

Other works as in [10], [11] separate the behavior of the components or services from the adaptation process. In [10] the generic framework called Dynaco needs to be specialized and is specific for each application, so several instances of the Dynaco framework are needed to adapt multiple applications.

Among these solutions, very few manage distributed systems and are themselves distributed. Based on Dynaco, [12] proposes some coordination patterns to cope with the distribution and decentralization of the adaptation system. However, to our knowledge these solutions are not able to manage heterogeneous applications.

VI. Conclusion

Nowadays, software developments should consider the issue of their adaptation when confronted with the dynamism of execution environments. However current solutions for adaptation are most often ad hoc and in consequence are not satisfying as long term solutions.

With our framework, which targets service-based applications, we propose to externalize the adaptation process into a distinct and distributed application. This new application is able to interact with various heterogeneous applications, services and execution platforms to adapt them. Moreover, as a distinct application it is able to adapt itself. In this paper, we have described some characteristics and advantages of our SAFDIS framework to ease the design of adaptation systems for service-based distributed applications. Some relevant parts of our implementation have also been presented.

Our framework provides a set of interfaces useful to build an adaptation system including some optional functionalities, such as a negotiator service which is used to negotiate the adaptation decisions when SAFDIS is distributed on several nodes. It is the role of an expert designer who knows his application and the execution environment to specialize our framework and to choose whether to use those optional functionalities. Moreover, our implementation is built as a Service-Based Application in order to take advantage of the service-oriented approach. For example, the dynamic binding between services eases the replacement of services when updating some part of the adaptation system. We also integrate some self-adaptation capabilities in our adaptation system and use them to select the planning algorithm.

The SAFDIS framework has been experimented to adapt test applications such as video streaming and multi-support video conferences applications. The planning phase has been used for the adaptation of an home-automation application [7], showing significant improvement compared to the initial version of the adaptation system. We are currently working on the design of the adaptation system for a large and very dynamic firemen assistance application.

 ISBN: 978-1-61208-124-3

In order to improve our implementation, we plan to study the use of already defined planning algorithms which are able to distribute the planning process ([13], [14], [15]) and to integrate them. This will help distribute the computation load in the same way it helps the analysis process. We also plan to work on conflicts that may appear in simultaneous adaptation processes. This kind of conflicts may appear because since a distributed and decentralized adaptation system is used, many adaptation processes may be launched and these processes may have to adapt the same element. In that case one of those adaptations may fail or may be inefficient. A third point we plan to study is the use of a knowledge base to share data between adaptation phases and to build a history about the system. This history may be used to improve the quality of the analysis phase by providing feedback on previous adaptations and to ease the resolution of conflicts by providing some information about the state of the running adaptations.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

REFERENCES

[1] S. R. White, J. E. Hanson, I. Whalley, D. M. Chess, and J. O. Kephart, "An architectural approach to autonomic computing," *Autonomic Computing, International Conference on*, pp. 2–9, 2004.

[2] D. Riehle and T. Gross, "Role model based framework design and integration," in *In OOPSLA '98: Proceedings of the 1998 Conference on Object-Oriented Programming Systems, Languages, and Applications*. ACM Press, 1998, pp. 117–133.

[3] G. Gauvrit, E. Daubert, and F. André, "SAFDIS: A Framework to Bring Self-Adaptability to Service-Based Distributed Applications," in *36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, Lille France, 09 2010, pp. 211–218.

[4] "Fipa interaction protocol specifications," 2010. [Online]. Available: http://www.fipa.org/repository/ips.php3

[5] A. L. Blum and M. L. Furst, "Fast planning through planning graph analysis," *Artificial Intelligence*, vol. 90, pp. 1636–1642, 1995.

[6] Y. Chen, C. wei Hsu, and B. W. Wah, "Sgplan: Subgoal partitioning and resolution in planning," in *In Edelkamp*, 2004, pp. 30–32.

[7] F. André, Daubert Erwan, Nain Grégory, M. Brice, and B. Olivier, "F4Plan: An Approach to build Efficient Adaptation Plans," in *7th International ICST Conference on Mobile and Ubiquitous Systems (Mobiquitous)*, Sydney, Australia, December 2010.

[8] D. M. Chess, A. Segal, and I. Whalley, "Unity: Experiences with a prototype autonomic computing system," in *ICAC '04: Proceedings of the First International Conference on Autonomic Computing*. IEEE Computer Society, 2004, pp. 140–147.

[9] J. Sudeikat, L. Braubach, A. Pokahr, W. Renz, and W. Lamersdorf, "Systematically engineering self–organizing systems: The sodekovs approach," in *Proceedings des Workshops über Selbstorganisierende, adaptive, kontextsensitive verteilte Systeme (KIVS 2009)*. Electronic Communications of the EASST, 3 2009, p. 12.

[10] J. Buisson, F. André, and J. Pazat, "Supporting adaptable applications in grid resource management systems," in *Proceedings of 8th IEEE/ACM International Conference on Grid Computing*, Austin, USA, 2007, pp. 58–65.

[11] P. C. David and T. Ledoux, "An Aspect-Oriented approach for developing Self-Adaptive fractal components," in *Software Composition*, ser. LNCS, vol. 4089, 2006, pp. 82–97.

[12] M. Zouari, M.-T. Segarra, and F. André, "A framework for distributed management of dynamic self-adaptation in heterogeneous environments," in *The 10th IEEE International Conference on Computer and Information Technology*, Bradford Royaume-Uni, 06 2010, pp. 265–272. [Online]. Available: http://hal.inria.fr/inria-00471892/en/

[13] F. Gechter, V. Chevrier, and F. Charpillet, "A reactive agent-based problem-solving model: Application to localization and tracking," *ACM Trans. Auton. Adapt. Syst.*, vol. 1, no. 2, pp. 189–222, 2006.

[14] M. P. Georgeff, "Distributed artificial intelligence," A. H. Bond and L. Gasser, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988, ch. Communication and interaction in multi-agent planning, pp. 200–204.

[15] P. Buzing, A. T. Mors, J. Valk, and C. Witteveen, "Coordinating self-interested planning agents," *Autonomous Agents and Multi-Agent Systems*, vol. 12, no. 2, pp. 199–218, 2006.

 ISBN: 978-1-61208-124-3