A Reputation-Based Approach to Self-Adaptive Service Selection

Jan Sudeikat[1], Wolfgang Renz[1], Ante Vilenica[2], and Winfried Lamersdorf[2]

1 **Multimedia Systems Laboratory**
Hamburg University of Applied Sciences
Berliner Tor 7, 20099 Hamburg, Germany
[jan.sudeikat;wolfgang.renz]@haw-hamburg.de

2 **Distributed Systems and Information Systems**
Computer Science Department, University of Hamburg
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
[vilenica;lamersd]@informatik.uni-hamburg.de

Abstract

Service-orientation provides concepts and tools for flexible composition and management of large-scale distributed software applications. The automated run-time management of such loosely coupled software systems, however, poses still major challenges and is therefore an active research area, including the use of novel computing paradigms. In this context, the dynamic and adaptive selection of best possible service providers is an important task, which can be addressed by an appropriate middleware layer that allows considering different service quality aspects when managing the adaptive execution of distributed service workflows dynamically. In such an approach, service consumers are enabled to delegate the adaptive selection of service providers at run-time to the execution infrastructure. The selection criteria used are based on the *cost* of a service provision and the continuous, dynamic evaluation of *reputations* of providers, i.e. maintained track records of meeting the respective service commitments. This paper discusses the design and operating principle of such an automatic service selection middleware extension. Its ability to balance different quality criteria for service selection, such as service *cost* vs. the *reliability* of provision, is empirically evaluated based on a multi-agent platform approach.

1998 ACM Subject Classification I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence — Multiagent systems; D.2.11 [Software]: Software Architectures — Service-oriented architecture (SOA)

Keywords and phrases Service, Workflow, Multagent System, Self-Adaptivity

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.14

1 Introduction

Service-orientation is a successful paradigm for the construction of flexible, loosely coupled distributed software systems. This loose coupling can be exploited to enable flexible compositions, orchestrations, and choreographies of such services and, thus, forms a technical foundation for the construction of *self-adaptive* systems (e.g. see [10], p. 20). For such systems, their respective adaptivity allows them to operate in dynamic execution contexts without explicit human or explicitly pre-programmed intervention. Especially when the availability of service providers and the demand for service invocations are (dynamically) changing, the management of service executions, in particular their respective selection and composition, has to be *adapted* according to changes in the environment (see Section 4).

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 14–25

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The adaptive operation in dynamic execution contexts is also of economic interest when service providers and consumers interact in open markets. In such cases, enterprises may schedule the achievement of individual tasks, i.e. the execution of workflows and the selection of services, based on economical as well as on technical criteria. An example for such an application may be provided by a car manufacturer that requires well-specified parts for the completion of a specific manufacturing process. Here, the automated selection of suppliers — based on given criteria — could provide means for controlling lean production processes.

Enabling flexible executions of workflows in these markets of service providers requires the adaptive selection of providers. Self-managing and self-organizing design techniques are attractive for controlling service selections while taking into account different quality aspects (see Section 4). Self-adaptive self-managing approaches introduce software components that control adjustments of subordinate system elements (see e.g. [10]). Self-organizing approaches realize system-level coordination in decentralized system architectures (see e.g. [11]), where the coaction of system elements gives rise to adaptive system level features.

In this paper, we propose a flexible service selection mechanism and related middleware extension for the management of service-based applications that can be used to equip service-oriented software infrastructures with adaptive service selection strategies. This framework provides a software layer that can be supplemented to service-oriented execution infrastructures. It provides an architectonic bluprint for integrating both self-adaptive and self-organizing dynamics in service-oriented execution infrastructures. The management of individual processes, i.e. workflows of service invocations, is handled by *software agents*. These agents do not manage the complete application but only those partitions that are influential for the completion of a specific workflow. Therefore, this architecture allows for the *self-organization* of concurrent, parallel workflows that are executed in an open (and dynamically changing) market of service providers. In general, self-adaptivity and self-organization are typically understood as opposing concepts for the creation of adaptive software systems (e.g. see e.g. [10], p. 5). The relevance of both concepts for the development of complex distributed systems and the respective need for comprehensive development concepts has been identified before (e.g. see [2]). In this context, the architecture proposed here provides an environment which allows to examine the interplay of managing entities.

We present both (1) the architecture of this middleware extension and (2) a design approach for adaptive selection strategies. First, the corresponding software architecture allows for integrating different quality criteria to be considered during the run-time selection of services. A flexible, agent-based design allows for the integration of different selection strategies. It provides a generic blueprint for integrating both self-adaptive and self-organizing dynamics in service execution middlewares. Secondly, we discuss and exemplify the principled design of distributed adaptation strategies. The conception of adaptive control algorithms, based on integrating feedback loops in software systems, is an active research topic [2] and we discuss the use of a visual modeling techniques. More specifically, a *reputation-based* approach is proposed here in which past experiences can be used, in addition to the service cost, for estimating the reliability of potential service providers. Balancing influences of several of such aspects for dynamic service selections allows even more specific tuning of the economic benefits. First experiences of such an approach in a case study are reported as well.

The following section introduces the agent-based selection-framework and illustrates the operating principle of the reputation-based selection strategy. Then the paper presents and discusses first simulation results (see Section 3) to evaluate the adaptive management. Finally, related work on the self-management of service-based applications is outlined before the paper concludes and gives prospects for future work.

2 An Adaptive Service Selection Middleware

The delegation of service selections to an additional middlware layer allows to separate the adaptiveness of service invocations from the realizations of service providers and consumers. In the middleware layer proposed here, *agent technology* is used as a connector between the service-oriented application elements, which constitute the application infrastructure, and the use of coordination means, e.g. utility-based service selections. In such an approach, software agents interact with service-based workflow executions and provide services themselves. Since the adaptations are delegated to supportive agents, adaptive system level aspects can be supplemented to the service execution middleware. Using the example of reputation-based service selections (see Section 2.2), the run-time invocations of services are observed by the agent system in order to reason from the past experiences to the reliability of service providers. The coordination of the involved agents is built on top of a coordination infrastructure for the creation of *self-organizing multi-agent systems* (MAS) [14]. This infrastructure provides a generic reference model for the integration of decentralized coordination processes in MAS.

Here, a generic model of the selection problem from [8] is adopted in which *Service Agents* (SAs) are providers of services and *Service Market Agents* (SMAs) are responsible to manage the execution of workflow instances. These workflows are distinguished between *Instantiated Workflow* (IW) and *Partial Instantiated Workflow*. In an instantiated workflow, service providers are associated to service invocations and when one or more services are not associated the workflow is partially instantiated.

2.1 Architectural Concept: A Supportive Middleware Layer

The selection middleware automates the allocation of service providers. This supportive system allows to automate and encapsulate the management of service-based software systems. The system is built in different layers (see Figure 1) which separate the business logic of the system elements from the accomplishment of adaptive features. The topmost *Application Layer* contains the functional elements of the managed application. Here these are providable service realizations and the service consumption that is controlled by a *Process Execution Engine*. The underlying *Connector* layer contains software agents that interact with these elements. The *Jadex Processes*[1] framework is used to realize these interactions. A process engine controls the execution of processes in the *Business Process Modelling Notation* (BPMN). The identification of service providers is delegated to SMAs. The provision of services is managed by SAs. As new SMA and SA (types) can be added at runtime the architecture enables the dynamic provision of new service providers/consumers. The selection mechanism (see Section 2.2) is embedded in an underlying *Coordination Layer*. This layer follows the architectonic model from [14] and separates the participation of agents in decentralized coordination processes from the agent models. This separation facilitates the reconfigurability and changeability of processes.

The constituent elements are the *Endpoints* and *Media*. Endpoints contain and control the activities that are conceptually related to the coordination of agents. These are computational elements that are able to observe and influence the execution of software agents [17]. The media are infrastructure elements that connect endpoints. Media provide interaction mechanisms for the coordination of agents that are based on communication infrastructures [17] and/or shared agent spaces [21]. Using a generic publish/subscribe interface, these

[1] http://jadex-processes.informatik.uni-hamburg.de/

infrastructures are integrated as a distributed event-based system.

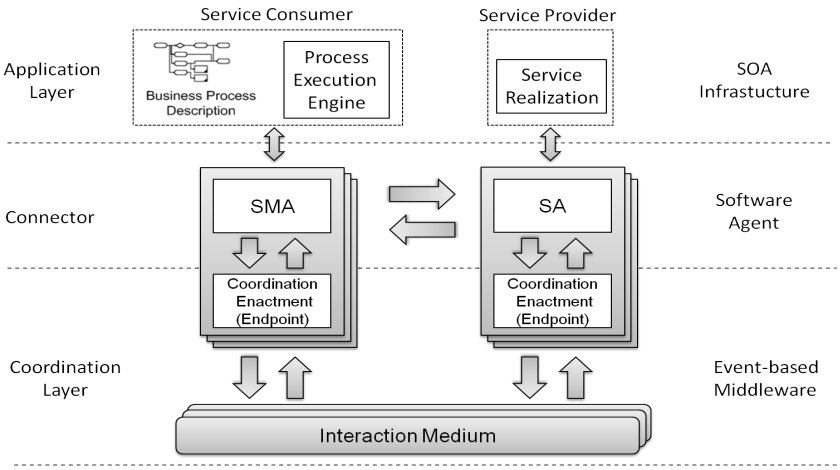

Figure 1 Service composition/selection architecture

Using this architecture, a reputation-based selection mechanism is realized (see the following section). Endpoints control the enactment of the coordination. In this case they control the selection and the update of the measured reputations of service providers. Two media connect the endpoints. One medium handles the *selection* of an appropriate service provider. Providers initially register at this medium and SMAs can inquire a service provider, decided by a given utility function (see Section 2.2.1). A second medium handles the reputations of service providers. This medium maintains the quality indicators of providers that are increased/decreased by the successful/unsuccessful completions of service. The configuration of the adaptive selections is be separated from the business logic, which is given in BPMN-based workflow descriptions. Each service provider and workflow is managed by an agent, e.g. to ensure that the commitments of individual providers do not overlap.

2.2 Adaptation Mechanism: The Dynamic Perspective

Within the Coordination layer (see Figure 1) adaptive, collaborative processes can be integrated into distributed systems. Here, the integration is exemplified by a process that allows to manage service invocations. We assume a setting where multiple providers are able to achieve specific tasks. The quality attributes, e.g. cost, processing time, reliability, etc., of the providers differ and thus the service consumers have to make economic selections. The consideration of different providers is not handled on the application level but is delegated to a supportive middleware.

For the description and configuration of decentralized processes, a dynamical modelling level has been developed that supplements agent-oriented design techniques with descriptions of dynamic aspects within MAS [18]. It makes use of *System Dynamics* [13] modelling concepts. Particularity the *Causal Loop Diagram* (CLD), a formalism for the illustration of system variables and their interdependencies, is extended for the description of the dynamics and coaction within MAS. Using this modelling stance, adaptive system behaviours can be pictured: Nodes in the graphs represent system variables, e.g. the number of agents that play a specific role and connection between these nodes denote influences and interdependencies. Influences describe additive or subtractive relations, e.g. when agents place or remove items

in/from a stock. Interdependencies describe causal relations, e.g. an increase of service requests in a SOA-based application consequently leads to an increase of service invocations.

2.2.1 Conception of the Dynamic System Behaviour

A pragmatic approach to the conception of a decentralized coordination processes is based on the comparison of the system behavior *as is* with the *intended*, i.e. coordinated and adaptive, behavior of the application [16]. The problematic, unintended behaviour is modeled first. Based on this *problematic dynamic*, a corresponding *solution dynamic*, i.e. an additional process fragment, is modelled, which improves the system behaviour.

The problem dynamic of the service allocation scenario is illustrated in Figure 2. A set of service providers (SAs) have committed to provide services in the given time-frame(s). Committed agents exhibit the role *Allocated Service* as they are locally aware of their obligations. One agent instance can commit to the participation in several IW, if the intervals of each service provision do not overlap. Before invocating allocated service instances, service providers may decide to change their commitment. In Figure 2, two possible reasons are denoted. First, agents may decide by themselves (*autonomous reallocation*) that a simultaneous service commitment is more profitable. Secondly, internal failures in the provider component (*execution error*) enforce that providers indicate their inability to satisfy commitments. The impact of these influences is characterized by an interaction rate (*leave IW*). Increases in this rate reduce the overall number of allocations and increases the number of agents that are *falsely committed*, i.e. agents that fail to achieve their commitments. When agents are in this state, the associated IWs are not executable and the number of *executable* IWs is reduced. The system exhibits a balancing feedback loop that limits the number of allocations (α), due to the perturbations, which influence the rate of agents that want to cancel their commitment (leave IW). The objective of the self-adaptive management of workflows is to counteract this feedback, i.e. to re-establish stable system configurations where all IWs are executable.

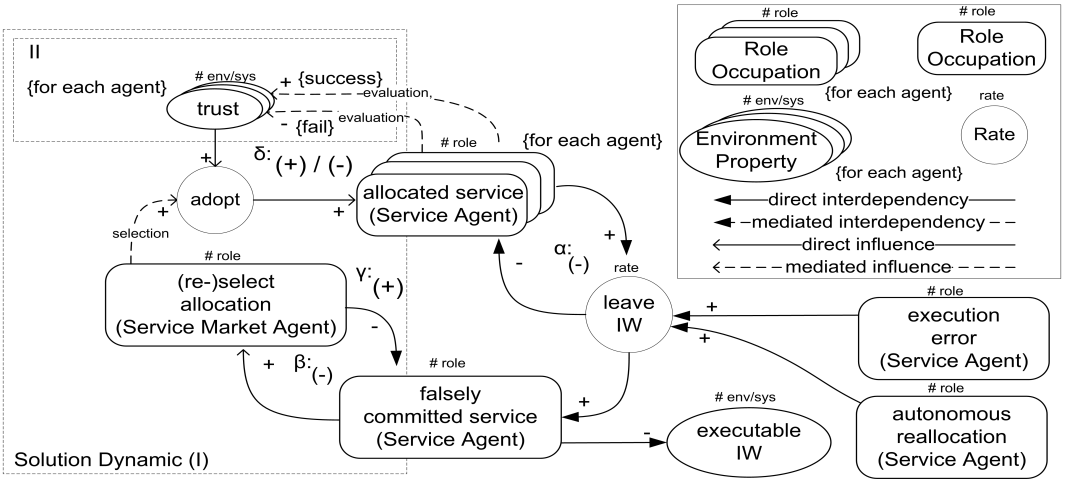

Figure 2 Macroscopic model: Dynamic service selections

The corresponding *solution dynamic* (see Figure 2, I) proposes the supplementation of the application with an additional agent role and related agent-interdependencies to establish additional feedback loops. Service Agents are enabled to communicate their intention to cancel a commitment to the corresponding SMA. This agent then initiates a *selection* of a

service provider to replace the missing commitments. When a replacement has been found the false commitment is revoked. The revoking manifests a second balancing feedback loop (β) that limits the number of agents that are falsely associated, i.e. that are deficient for the provision of the allocated service. As the selections also reinforce the number of allocations, a third, reinforcing feedback is manifested (γ) that reinforces allocations. This reinforcement alleviates the effects of the problematic feedback (α). This reinforcing feedback itself is controlled by the availability of falsely committed agents (β).

The establishment of the counterbalancing feedback γ is a requirement. The exhibited system dynamics is expected to show this feedback and the model neglects how this feedback is established. The approach followed here makes use of a utility function that takes into consideration not only the cost of the service invocation but also the *Reputation* of providers, i.e. the confidence that service providers can meet their commitments.[2] These reputations, called *trust*, are adjusted at run-time (see Figure 2, II). A set of agents (allocated service) participates in the workflow and for each agent such a value is maintained (*evaluation*). The corresponding values are increased when the service execution *succeeds* and are decreased when service providers *fail* to meet their obligations. The current trust value is considered in the service selection and thus influences the overall rate (*adopt*) of provider selection.

An alternative, decentralized design approach would be to equip Service Agents with the ability to negotiate their replacement with other agents. The causal structure of the application dynamics would not be affected by this adjustment, but the role of the negotiation initiator (*(re-)select allocation*) would be played by the Service Agent, not the Service Market Agent. An example for such a completely decentralized approach, embedded within a Coordination Layer, is given in [19], where failing machines in a production line are responsible to find replacements for their designated activities. In the following, the former approach is examined in system simulations.

2.2.2 The Selection Dynamics

The dynamic model in Figure 2 describes the macroscopic dynamics of systems that are equipped with the supportive middleware layer. This modelling level is particularly suited to describe self-organizing dynamics (e.g. see [18, 16]), and is here used to denote the operation of sets of self-adaptive managers (SMAs). For the explanation of concrete simulation experiments (see the following section) a detailed model of the dynamics is needed that does not only refer to macroscopic system variables, e.g. the overall numbers of agents that show specific behaviour, but that describes the concrete influences in system variables. This description level gives a more specific illustration of the relations of system variables and is obtained by refining macroscopic models. A detailed discussion of the different modelling levels for self-organizing systems can be found in [15].

The corresponding model (see Figure 3) describes how the computation of services' utilities influences the overall profit generated by the process executions. The variable *Profit (Process)* describes the economic benefit that results from the execution of workflows. The

[2] Utility functions offer the possibility to specify preferences among (potentially conflicting) objectives and can be mathematically expressed as

$$U(x) = \sum_{i=1}^{N} x_i \cdot w_i$$

where w_i expresses the weight of an objective x_i [5].

major (additive) contributions are the profits, which are generated by the constituent service invocations (*Profit (Service)*). The selection of the corresponding service providers is based on the computation of the utility (*Available Utility*). This value estimates the most appropriate service provider and thus the provider with the maximal utility value is selected. Several factors influence this estimation and in Figure 3 two are exemplary denoted. These are the current *Reputation* and *Cost* values of services. In addition, two disturbance variables, which are externally set to fixed values in the system simulations, influence the generation of profit. First, the failing of a service (*Service failure*) is associated with a penalty cost that subtracts from the overall profit (β). Failures also minimize the trust values and therefore have an indirect influence on the service selection. A second disturbance is the blackout of services (*Black out probability*). When services are temporarily not available, e.g. due to internal errors or unreliable network connections, this affects the current service selection and also minimizes the current reputation value. The successful provisioning of services is recorded by the corresponding medium and increases the reputation values that are associated to specific providers. Therefore, the variables are steadily increased as denoted by positive (+), i.e. reinforcing, feedback loop [13]. On the other hand, the external disturbances, i.e. failures and blackouts, reduce the individual reputations.

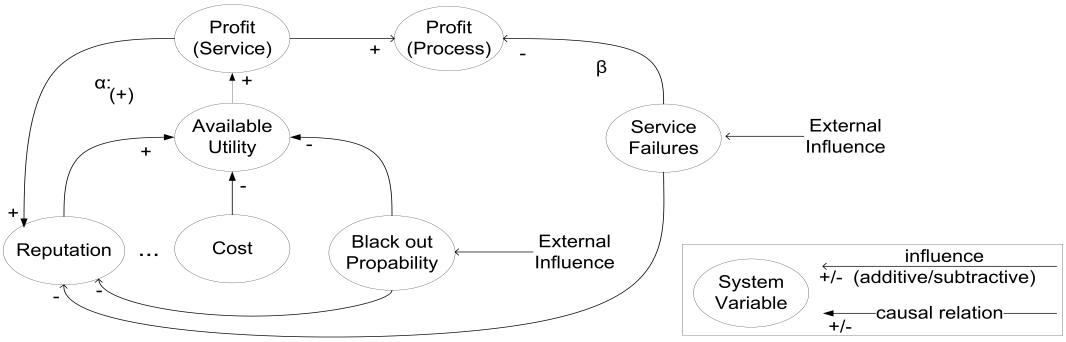

Figure 3 The dynamics of process costs, as shown by the implemented simulation model (see the following section)

3 System Evaluation

3.1 Operating Principle

That the system is self-adaptive, i.e. that services are adaptively selected, is validated by system simulations. A sample simulation run, with a fixed reputation value[3] is shown in Figure 4. The measurements were obtained from a single SMA that repeatedly selects one service. The SMA has three options: an *expensive*, a *moderate* and a *low-cost* service provider. These agents can perform the same service type but differ in their cost and the probability of a service failure, i.e. a blackout, which is detected by the infrastructure applying timeouts. The three service providers were set up with following cost characteristics: expensive (1200), moderate (1000), low-cost (800) and with an initial value of 30 for reputation. For the ease of evaluation the SMAs service selection function, i.e. the utilitfy function, only considered the costs and reputation of respective SAs. Quality of service parameters where not considered

[3] set to: 0.4

but could easily be integrated by extending the arguments of the service selection function with additional parameters.

The history of service invocations shows that the invocations (I) and the recorded reputation values (II) mutually influence each other. These influences validate a set of the relations in the dynamic application model (see Figure 3). In phases where one service type is invoked repeatedly and performs as expected, such as between the time steps 40 to 55,[4] the corresponding reputation value is amplified. This is expressed as the reinforcing feedback (α) in Figure 3. The systems quality also decides the likelihood that service invocations will fail using following function to compute the time between two blackouts: *t = -log(1-x)*ServiceCharacteristic*. Thereby, the three service providers had following *ServiceCharacteristics*: expensive (0.1), moderate (0.5) and low-cost (0.999). Moreover, service failures are drastic events that lead to sudden decrease in the reputations (negative contribution to the Reputation node in Figure 3). Upon these failures the provider with the highest utility is selected.

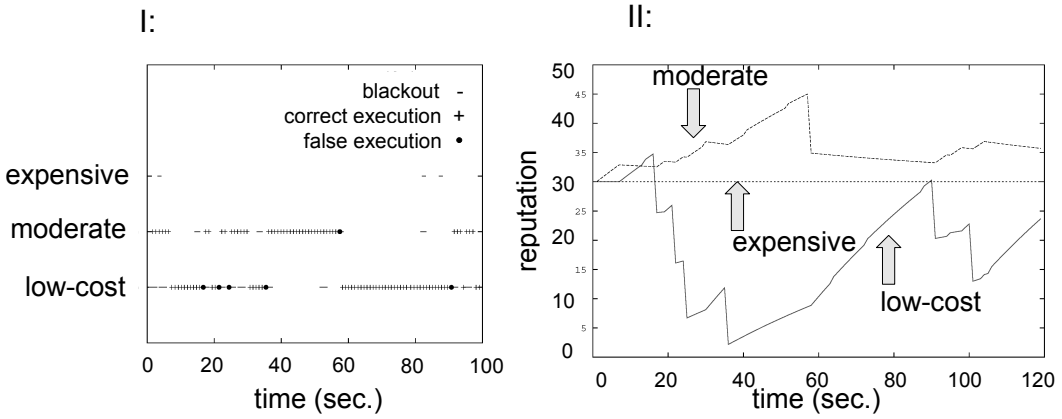

Figure 4 Sample simulation run: Service invocations (I) and the corresponding reputations (II)

3.2 The Impact of Considering Reputations

The adjustment strategy proposed here makes use of past experiences. The weight of the gained reputation is a static parameter that is set initially. In Figure 5 (I), it is shown how this parameter affects the fractions of agents that participate in workflows. Simulations are carried out for two minutes and the SMAs repeatedly select one service provider. For low values (x axis) the historic reputation that services gain have limited impact on the selection. Instead the service cost is the dominant factor and low-cost service providers are preferred. When higher weight values are used, more reliable, i.e. more expensive, service providers are used.

The performance of the workflow executions is indicated by two measurements that are shown in Figure 5 (II). Simulations are carried out for a fixed duration and higher weights for the reputation values lead to a slight increase of the total number of services that can be completed in this time. This effect results from the increased utilization of more reliable, and thus expensive, service providers. Due to their reliability these fail more seldom and re-invocations of services are avoided.

[4] measured in seconds, see Figure 3

The weight of the reputation significantly affects the overall profit that is generated. For small wights (< approx. 0.4), the selection is mainly influenced by the cost of invocations and therefore the fraction of selected services that fail are comparatively high. Failures cost penalties and enforce re-selections of providers. Thus the overall profit that can be generated is limited. For high weights (> approx. 0.8), the selection is biased toward expensive services. Due to their cost, it is not of economic interest to use these providers extensively and the overall profit is reduced. A weight about approx. 0.6 maximizes the profit as the use of expensive/low-cost service providers is balanced and the majority of invocations refer to moderately priced providers. In the simulations, the cost of expensive providers and the penalties of failures are set to values that only allow for a small net earnings area that is reached by using mid-level reputation weights.

Qualitatively, the simulation results indicate that the selection process conforms to the systemic model of the selection dynamics that are given in Figure 3. Service failures, an external influence, affect the generated profit (β). Therefore, unreliable providers have to be avoided. Since failures reduce the reputation, SMAs are repelled from service providers that have shown to be unreliable. This explains that the use of reputation mechanisms can increase the systems performance. On the other hand, the steady increase of reputations (α) leads to a problematic use of expensive providers for high-level reputation weights. Their cost limits the total profit. Thus the balancing of the former influence (external factors) and the latter feedback is required. This balancing depends on the system parametrization and for the simulation setting studied here, moderate weights of the reputation in the utility function are appropriate.

In conclusion it can be stated that the simulation results prove the applicability of the proposed framework to enable self-adaptive service compositions by using reputation as a criterion (among others like costs) for selecting service partners. At the same time, it has to be stated that using this framework for automated service selection requires simulation effort in order to find appropriate settings for an environment that exhibits high dynamics. The mathematical analysis of the dynamics for inferring appropriate parametrizations is left for future work. Additionally, there is the need to provide more implementations of wide-spread negotiation protocols in order to increase the utilization of this framework as well as to study the impact of using reputation with these protocols. Nevertheless, this evaluation shows that using reputation, i.e. upon to a certain factor, to measure the quality of service partners increases the profit of service executions in contrast to settings that take only service costs into consideration.

4 Related Work: Adaptive Service Selections

The fact that agent-orientation can be used as a valuable enrichment for service-oriented architectures has been argued by several authors, e.g. in [12]. Here, an enhancement is realized by a layered architectural model. Software agents represent the consumers and providers of services and form the logical link between the application business logic and the adaptivity-related coordination, i.e. the selection.

Consequently, the automation of the orchestration and composition of services has been studied. Approaches for the adaptive management have to prepare for (possibly) large scale of service-oriented systems and for the handling of dynamic execution contexts. Due to these challenges, the use of novel computing paradigms, which focus on bringing about intended application level dynamics, in this application domain have been proposed. Examples are the use of self-adaptive architectures and self-organizing problem solving strategies. Self-

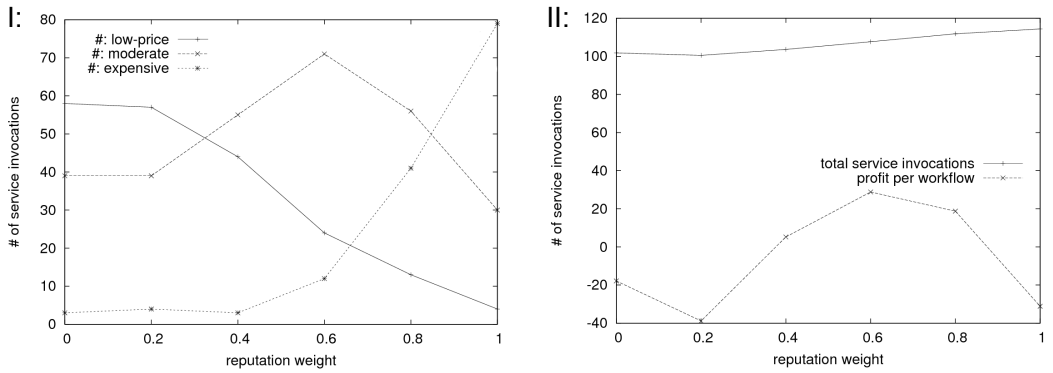

Figure 5 Averaged cumulative invocations of service providers plotted over the fixed reputation value (I), averaged, total sum of service invocations (II) and averaged profit generated plotted over fixed reputation values (II).

organization allows for decentralized management strategies that are particularly suited for large-scale application infrastructures. Examples include the transfer of nature inspired phenomena e.g. the chemical reactions [3] and the differentiation of individuals as found in biological systems [9].

Management of self-organization in service-based applications is discussed in [7] and a corresponding reference architecture is proposed. A *management overlay* is constructed by associating each element in the application with a managing software element. These elements follow the Observer/Controller (O/C) architecture that is developed by the Organic Computing research initiative (e.g. see [20]). The proposed architecture conforms to the framework proposed here as each element is associated with a computational element that control the local adaptations. However, O/C elements are comparatively complex entities that contain among others data analysis, prediction, and simulation components in order to estimate the system states that the system is approaching. The model presented here is comprehensive as it does not only contain the decision making elements but also abstractions of their interactions. Endpoints are comparatively lightweight as they only contain the logic that is required to impose the participation in the collaborative process, which steers the system operation. Simplification of the controlling system elements is enabled by the explicit modelling of the process (see Section 2.2) [16].

Furthermore, there is existing work that targets either the provision of infrastructures for the dynamic selection of services or the development of new negotiation/bidding strategies in open marked-based scenarios. On the one hand, Borrisov et al. [1] propose a framework that automates the generation of bids for the allocation of computing services in grid-based systems. Similar to the approach presented in this work, [1] incorporate a technique, i.e. machine learning, to deal with aspects related to reputation. At the same time the approach focuses on the domain of grid-systems and gives therefore only partial solutions to general problems related to adaptive service selections, e.g. the approach does not deal with blackout of services. On the other hand, Lewis et al. [6] focus on developing a new negotiation strategy that can cope with the challenges of dynamic, decentralized and service-based systems. This strategy is encapsulated within so called "evolutionary markets agents" that act on behalf of service providers and service consumers. Therefore, it would be of interest to extend the framework used within this work with this new bidding strategy of Lewis et al. in order to evaluate its advantages and disadvantages with respect to existing negotiation strategies.

5 Conclusions

This paper proposes a generic management architecture for the adaptive management of service-oriented applications based on multi-agent middleware. The corresponding framework uses agent technology to enable the self-adaptive operation of services. It provides an additional middleware layer that can extend infrastructures in order to provide for decentralized service selection management. The design of this framework is not biased towards specific management strategies, but is, on the contrary, highly configurable. This paper approaches the service selection problem via a *reputation-based strategy*, where past experience with service provision is used as a(n additional) selection criterion for service providers. The corresponding middleware layer is based on a programming approach for *self-organizing multiagent systems* [14]. Accordingly, the work reported here does not only concern the service infrastructures but also exemplifies how the decentralized, agent-based management can be used to supplement conventional software infrastructures. In such an approach, the corresponding MAS serves as a connector between application-level software artefacts and the participation of such artefacts in managing distributed processes with adaptive system properties.

Future work shall address the elaboration of management strategies and the validation of the management framework in more realistic scenarios. This includes studying non-functional aspects of the approach, e.g. scalability, and the mathematical analysis of the system dynamics. The framework presented here and the underlying programming model first demonstrate in principle the ability to supplement decentralized management processes into software systems. This however, requires additional policies, guidelines as well as heuristics that enable development teams to conceive the appropriate management approach for specific application scenarios in order to decide between, e.g. centralized, managing software entities, as addressed by autonomic computing systems (e.g. see [4]), and the use of decentralized coordination processes [19].

Acknowledgements The authors would like to thank *Deutsche Forschungsgemeinschaft* (DFG) for supporting this work through a joint research project on "Self-organization based on decentralized co-ordination in distributed systems" (SodekoVS) and, in addition, Claudia Di Napoli and Maurizio Giordano from C.N.R, Naples, Italy, for related discussions on service selection problems.

References

1 Nikolay Borissov, Dirk Neumann, and Christof Weinhardt. Automated bidding in computational markets: an application in market-based allocation of computing services. *Autonomous Agents and Multi-Agent Systems*, 21:115–142, 2010.

2 Yuriy Brun, Giovanna Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Müller, Mauro Pezzè, and Mary Shaw. *Software Engineering for Self-Adaptive Systems*, chapter Engineering Self-Adaptive Systems through Feedback Loops, pages 48–70. Springer-Verlag, Berlin, Heidelberg, 2009.

3 Claudia Di Napoli, Maurizio Giordano, Zsolt Németh, and Nicola Tonellotto. Using chemical reactions to model service composition. In *SOAR '10: Proceeding of the second international workshop on Self-organizing architectures*, pages 43–50. ACM, 2010.

4 Rajarshi Das Jeffrey, O. Kephart, Charles Lefurgy, Gerald Tesauro, David W. Levine, and Hoi Chan. Autonomic multi–agent management of power and performance in data centers.

In *Proc. of the 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008) – Industry and Applications Track*, pages 107–114, 2008.

5 Henry M. Levin and Patrick C. McEwan. *Cost-Effectiveness Analysis: Methods and Applications*. Sage Publications, 2. edition, 2000.

6 Peter R. Lewis, Paul Marrow, and Xin Yao. Resource allocation in decentralised computational systems: an evolutionary market-based approach. *Autonomous Agents and Multi-Agent Systems*, 21(2):143–171, 2010.

7 Lei Liu, Stefan Thanheiser, and Hartmut Schmeck. A reference architecture for self-organizing service-oriented computing. In *ARCS*, volume 4934 of *LNCS*, pages 205–219. Springer, 2008.

8 Claudia Di Napoli. *Software Agents to Enable Service Composition through Negotiation*, chapter 12, pages 275–296. Studies in Computational Intelligence. Springer, 2009.

9 F. Saffre, J. Halloy, M. Shackleton, and J. L. Deneubourg. Self-organized service orchestration through collective differentiation. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(6):1237–1246, Dec. 2006.

10 Mazeiar Salehie and Ladan Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):1–42, 2009.

11 G. D. M. Serugendo, M. P. Gleizes, and A. Karageorgos. Self-organisation and emergence in MAS: An overview. In *Informatica*, volume 30, pages 45–54, 2006.

12 Munindar P. Singh and Michael N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley & Sons Ltd, 2005.

13 John D. Sterman. *Business Dynamics - Systems Thinking and Modeling for a Complex World*. McGraw–Hill, 2000.

14 Jan Sudeikat, Lars Braubach, Alexander Pokahr, Wolfgang Renz, and Winfried Lamersdorf. Systematically engineering self-organizing systems: The SodekoVS approach. *Electronic Communications of the EASST*, 17, 2009.

15 Jan Sudeikat and Wolfgang Renz. *Applications of Complex Adaptive Systems*, chapter Building Complex Adaptive Systems: On Engineering Self–Organizing Multi–Agent Systems, pages 229–256. IGI Global, 2008.

16 Jan Sudeikat and Wolfgang Renz. On the modeling, refinement and integration of decentralized agent coordination – a case study on dissemination processes in networks. In *Self-Organizing Architectures*, volume 6090/2010 of *LNCS*, pages 251–274. Springer, 2010.

17 Jan Sudeikat and Wolfgang Renz. Separating agent-logic and inter-agent coordination by activated modules: The decomas architecture. *Electronic Proceedings in Theoretical Computer Science*, 27:17–31, 2010. (Proceedings of the Workshop DCDP 2010).

18 Jan Sudeikat and Wolfgang Renz. Qualitative modeling of mas dynamics - using systemic modeling to examine the intended and unintended consequences of agent coaction. In Michael Luck and Jorge J. Gomez-Sanz, editors, *Agent-Oriented Software Engineering X*. Springer, 2011. (to be published).

19 Jan Sudeikat, Jan-Philipp Steghöfer, Hella Seebach, Wolfgang Renz, Thomas Preisler, Peter Salchow, and Wolfgang Reif. Design and simulation of a wave-like self-organization strategy for resource-flow systems. In *Proceedings of The Multi-Agent Logics, Languages, and Organisations Federated Workshops (MALLOW 2010)*, volume 627, 2010.

20 Stefan Thanheiser, Lei Liu, and Hartmut Schmeck. Towards collaborative coping with it complexity by combining service–oriented architectures and organic computing. *System and Information Science Notes*, 2(1):82–87, 2007.

21 Ante Vilenica, Jan Sudeikat, Winfried Lamersdorf, Wolfgang Renz, Lars Braubach, and Alexander Pokahr. Coordination in multi-agent systems: A declarative approach using coordination spaces. In *Proc. of EMCSR 2010 - Int. Work. From Agent Theory to Agent Implementation (AT2AI-7)*, pages 441–446. Austrian Society for Cybernetic Studies, 2010.