2012 IEEE 14th International Conference on Commerce and Enterprise Computing

Context-Based Reputation Management for Service Composition and Reconfiguration

Jinhwan Lee, Jing Zhang, Zhengqiu Huang, and Kwei-Jay Lin
Dept. of Electrical Engineering and Computer Science
University of California, Irvine
Irvine, CA 92697 USA
{jinhwanl, jingz, zhengqiuh, klin}@uci.edu

Abstract—**Service-Oriented Architecture (SOA) provides a framework for service composition and reconfiguration. In addition to making composition decisions by considering functional attributes, technology for composition by meeting non-functional requirements such as QoS is also needed. In this paper, we present an approach to compose business process by considering context-based service reputation. Since some services may perform well only at specific context, we may use context similarity to filter out those services that are not fit given a user's context. We present the design of a Context Manager which works as part of an SOA middleware to compose and reconfigure services. Simulation results show the effectiveness of our design.**

I. INTRODUCTION

Service Oriented Architecture (SOA) is a powerful technology to integrate interoperable web services in a business process. In general, service integration should try to maximize the performance of a business process such as to have the minimum latency, the shortest response time, the optimal reliability, an acceptable level of security, etc. Although technology exists to optimize the performance in terms of the above QoS attributes, it still may not meet the needs of all users, since users may have unique service requirements at some specific context. For example, many web-based map services provide general geographic and street information based on a user's location. On the other hand, each map service can have a different latency due to the network bandwidth between the user and the map's real-time data center. Some services may perform perfectly only at some specific context (e.g. at certain location, at certain time) but badly outside of it. Therefore, it is useful to consider user context when selecting services to improve a user's service experience.

This paper proposes a methodology to enhance the service composition and reconfiguration mechanism by using context-based reputation management [5] in SOA middleware. In our study, we assume user feedback on a service's quality can be collected. Moreover, due to context dependency, some services may have a better reputation than others in some specific context (such as location and time) for some group of users. Therefore, we can derive a service's working context (referred as *service context* in this paper) and use that to match with a user context to decide if the service should be a candidate for service composition in SOA.

In our implementation, context aware computing is delegated to a service middleware component called *context manager* so that a user's local or mobile device can be relieved from the complex context management yet can utilize the context and reputation knowledge effectively. The context manager collects and reasons from the collected context information, and may trigger a service reconfiguration in case of unacceptable service degradation due to context changes.

Our innovation in this research is as follows.

1) Reputation management driven by context: Most existing reputation framework reports service reputation by a single value (e.g. 0 to 5 stars). In our work, service reputation is reported differently based on the similarity of user context and service context. Our context manager generates service context and derives the service reputation in a user context using accumulated user feedbacks.

2) Efficient context-based service filtering: For service integration, services that are not performing well in a user's context should be left out before service selection. We propose a way to integrate context into service composition and reconfiguration to provide context-based service experience, which can deliver a siginficant improvement on service performance.

The rest of the paper is organized as follows. Section 2 shows some related work to our study. Section 3 motivates our work with a context-dependent business process scenario. We give an overview on service reputation and service selection by utilizing service reputation in Section 4. In Section 5, context framework, management and architecture are presented. Finally, we present a performance study in Section 6.

II. RELATED WORK

A. Context-Aware Computing

Much research has been done to adapt context into applications [3] or web services [4], although most worked on application-specific context-aware computing. A. Schmidt, et al. propose context mapping method using PDA or handheld device [1]. The device is installed with a recognition application which analyzes raw data from device sensors and obtains high-level context information with the predefined scenario. B. Schilit, et al. [2] study dynamic customization of mobile application for the changing application preferences during program execution. As a user

978-0-7695-4857-9 2012
DOI 10.1109/CEC.2012.18

moves around in physical area, mobile computing environment becomes dynamic with selection of proximate devices. T. Gu, et al. propose a Service Oriented context-Aware Middleware (SOCAM) [10] for building rapid prototyping context-aware services.

B. Context Matching

MUSIC project characterizes the situation of an entity, as context, in order to increase usability and effectiveness on their operations by talking environmental context into account [9]. They propose two graphs (service description and current context description) comparison methodology as comparing two nodes individually and measuring globally the whole graph. Zhong et al. propose an approach for semantic search by matching conceptual graphs that describe the documents' content [6], and obtained the similarity of two graphs by measuring the distance of two nodes positions. Tsang and Stevenson provide a graphical formalism incorporating both the semantic distance of the components nodes and distributional differences between the context profiles by utilizing a minimum cost flow (MCF) problem by transporting the flow from one context to the other [12].

III. MOTIVATING SCENARIO

The quality of many real-time services is context-dependent. Figure 1 shows a use case for the personal medical care scenario that is now a growing business area. For example, Vesag [14] provides a wrist watch which, by collecting patient biometric information such as heart rate, blood pressure and sending them to remote users, can be used to keep track of patient status thousands of miles away. Lifecomm [15] serves similar services with a button device which connects to specialists for emergency situations.

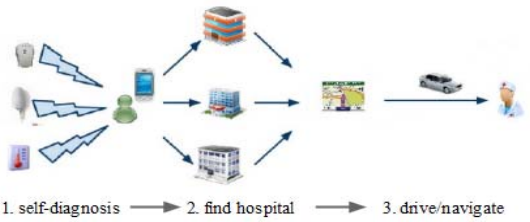

Figure 1. Medical Care Scenario

In this scenario, services are connected as process node in a business process. The scenario starts with a patient (user) being monitored by sensors in a user's house. The first service is to analyze and diagnose patient's condition from sensor data. In this service, sensors also provide user context. Once it detects some abnormal status on the patient, the second service tries to find the hospital based on the patient's symptom. In this step, time and location are used as context. It is important to find the closest one during its operating hours. The next service is to provide the navigation system to guide the patient to the hospital. Based on patient's location, a localized navigation system should be provided. In addition, since some system requires high network bandwidth for real-time map, device network capability will be useful for service selection.

IV. REPUTATION DRIVEN SERIVCE COMPOSITION

A. Collection of User's Feedback

User feedbacks are very important to a system management since service reputation is built from feedbacks and can be used to select well-performing services. When a service is executed, some users may be satisfied with the outcome but others may not. Similar to VRESCo system [16], we use a structured feedback instead of unstructured one. While VRESCo uses the numerical rating between 1 and 5, we adopt the binary feedback, either as positive(1) or negative (0).

Since most users may not proactively provide their feedback, feedbacks will be implicitly derived from users when they complete the service. For example, if a service is executed with no user interruption or even with prolonged connection, a positive feedback is assumed. Otherwise, if a user drops the service in the middle or a service simply fails to meet its goal, a negative feedback is entered.

B. Selection of Qualified Candidate Services

Our project presents an approach to select better services by service reputation given a user's context. For the service composition and selection decisions, only reputable services that have similar context become the candidates for the service process. By adopting context-aware service filtering and reputation-driven service selection, users have a less chance to be given a service that has a bad reputation.

For this purpose, context manager is designed to handle context-related computing and management. When a user requests for a business process, context manager retrieves candidate services from service database and inspects context for each service. Once context manager acquires context similarity values of all candidate services, it lists the services from the highest context similarity value to the lowest.

V. CONTEXTED REPUTATION MODEL IN LLAMA

In this project, we design a context tree structure which is used not only to store the context data, but also to efficiently compare and to effectively maintain context data.

A. Context Tree with Reputation

A context tree consists of three main subtrees: user, device, and environment. User context contains a user's personal profiles such as gender, age, etc. Device context includes the information of the user interface device, such as OS, screen size, etc. The last context, environment includes data describing external situations. Time and location data are good examples as dynamic environment context, time always changes and location data is updated if the user is moving. In the tree, context node consists of four tuples: attribute ID, current context value, reputation value and reputation count. Reputation value represents the accumulated feedbacks on context value for context attribute and count is the number of feedbacks collected. Figure 2 shows a simple context tree.

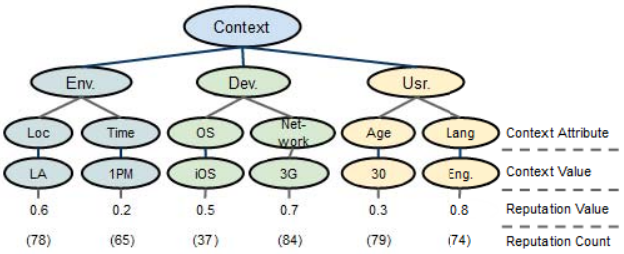

Figure 2. Context Tree

B. Context Reputation Data Management

After a user receives the execution of a service, his feedback on the service performance as well as his context information will be collected by the context manager. Assume that a service is well performed in location A, but not in location B. Users who use the service are more likely to produce a positive feedback from location A, but a negative feedback from location B. The collection of user's feedbacks are used to create reputation which is maintained in the context reputation tree with the following operations:

1. Adding a new reputation: when a feedback for a new context value is received by context manager, the reputation based on positive(1.0) or negative(0.0) feedback is added as new context node.
2. Updating existing reputation: when a feedback for an existing context value has arrived, the existing reputation value is updated by [5].

C. Critical Context Attribute Clustering

Although context reputation tree contains reputation data for all context attributes, only a certain subset of contexts are sensitive to affect the service performance. The context attribute which affects the performance is called *critical* context attribute. For example, if a service has a much better (short) latency in certain location (CA) than other location (NY), location is considered a critical context. On the other hands, if a service's performance is the same regardless of when it is used, time is not a critical context attribute.

In order to figure out whether context attribute is critical or not, K-mean clustering algorithm [13] is used. Since we only want to cluster values as critical or not, we set K to 2. The clustering algorithm is as follows.

1. Place the first two values on centroid 1 (C1) group and centroid 2 (C2) group, respectively
2. Assign each value to the closest centroid group
3. Calculate centroid values as group average
4. Repeat steps 2 and 3 until no value moves to the other group.

Using reputation data management, a positive feedback increases the reputation value on the context attribute and the negative one decreases it. As a result, reported reputation on critical context attributes usually shows a context-dependent distribution. However, for users with different values on non-critical context attributes may report feedbacks without any context-dependent pattern. In this way, context manager may decide whether context attribute is critical or not. Therefore, for the critical context attribute, the difference of the centroid values is quite different while the difference for non-critical context attribute is relatively trivial.

D. Context Comparison Algorithm

Given a context tree, we can compute the weight of the whole tree. A parent context weight is the sum of all children's weight. Some context can have a higher weight for more important context attributes. If all children have the same weight of 1, the total weight of context is the total number of context attributes. When a service context tree is compared with a user context tree, the context comparison engine goes from the root through each context attribute on a service context and tries to find which is matched and which is not against a user context. When the comparison is completed, the root shows how many context values match with the user's context.

Figure 3 shows an example of context tree comparison. Service context tree has 9 concrete contexts with which the service can perform in the best condition. Assuming that all concrete contexts are equally weighted, when it is compared with user's context, the context similarity value is 7/9 = 0.78 because there are 7 matches except C-1-1 and C-2-2. Note that C-1-4 is not in service context. Therefore, it doesn't count to calculate the context similarity.

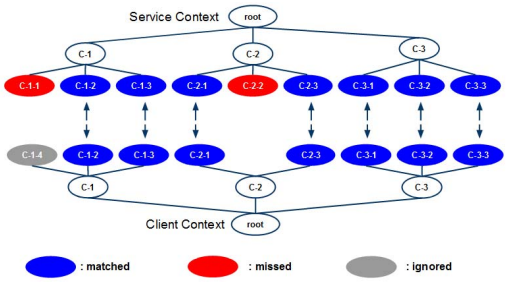

Figure 3. Simple Context Tree Comparison

E. Context Manager Architecture

The Llama [7] middleware extends an enterprise service bus (ESB) to provide powerful management capabilities for SOA. Q-Broker [11] and Adaptation Manager [8] have been developed to compose BP's and reconfigure failed services, respectively. We have implemented a context manager in Llama to enable context awareness on service composition and reconfiguration. The components in the context manager are designed to interact with and support other components in Llama. Figure 4 shows the context manager architecture. The workflow of a user's request is described along with context manager components.

1. CPU (Context Process Unit): CPU consists of three components; context collector, context reasoning engine and context comparison engine. Context collector plays a role of interface between user and context manager and data are reasoned in order to compare with other context by context reasoning engine. And context comparison engine computes how similar or different two context data are.
2. Context Reputation Manager: once users complete to execute a service, they provide feedbacks with

context data, which are stored in DB. When a service is requested for context comparison, this component provides the context reputation data to infer which context value is meaningful for a certain service.

3. Context Handler: in reputation DB, since reputations are stored for context attribute values, they need to be refined as critical and non-critical and rebuilt as service's reputation tree. Moreover, it is also cached for when it is required again.
4. Service Filterer: service filterer selects only qualified service described in III-B to provide user's context friendly services and reputation generator computes service reputation. However, the service context tree contains reputation values for different context. Reputation generator derives an overall reputation for each service given the user's context. With the tree, it takes reputations of matched values and normalizes them with the number of feedbacks. For example, in Figure 5, if the user's current location is LA and time is Evening, the normalized value is $(0.7x10 + 0.6x14) / (10 + 14) = 0.64$
5. Composition/Reconfiguration Trigger: composition trigger and reconfiguration trigger play similar roles as injecting the selected services with their QoS parameters including service reputation to Q-Broker at build-time and Adaptation Manager at run-time.
6. Context Monitoring Agent: user's context keep changing, which can cause the experience of service. This agent collects user's context and predicts any possibility of service failure or service performance degradation, before actually it happens. By anticipating the failure or performance degradation, the agent decides for the reconfiguration, prior to actually it gets the problem, so that overall service process performance can be improved.

VI. Performance Study

The main goal of our performance study is to show that using context-based reputation for service composition and reconfiguration brings significant improvement on the performance of business processes through the observation on how often the context manager triggers service reconfiguration for different simulation parameters, and how different context models affect the overall performance of business process execution such as service failure ratio.

In the simulation, the sequential business process has 4 process nodes and each node has 8 candidate services. Context manager selects up to 4 candidates for composition based on the threshold value. Since the context manager doesn't know the right threshold value with which candidate services are filtered out, 5 different values are used from 0.7 to 0.9 with 0.05 interval. A higher threshold value to trigger reconfiguration causes a higher reconfiguration ratio because the context manager is more sensitive to user's context changes. Moreover, in order to show how different context similarity models affect the overall performance of business process execution, various context sensitivity value(μ) is applied from 0.5 to 0.7. The higher value(μ) is, the less sensitive to the context change a service is. However, the reconfiguration ratio doesn't guarantee a user's satisfaction. If context manager triggers too many reconfigurations, it may cause higher False Positive (FP) rate, which introduces additional overload. On the other hand, low reconfiguration ratio may reduce FP rate but increase False Negative (FN) rate. Thus, context manager fails to predict the service performance degradation and misses reconfiguration chance.

Figure 6 shows the FP and FN rate with different context models. Lower threshold value increases FN rate while it decreases FP rate. However, for each model, they meet around 0.11% which can be the point where the context manager can decide the threshold value for reconfiguration. For example, with μ value 0.6, context manager decides 0.8 as the threshold value. Figure 7 compares the result on the number of reconfigurations between reputation-driven vs no-reputation model. The first three legends are for reputation-driven reconfiguration for μ value and threshold value pairs which are optimized in Figure 6. As figure shows, reputation-driven models have higher numbers on the 1st reconfigurations. This is because reputation-driven model reconfigures when user's context changes before the service failure. However, after the 1st reconfiguration, the number is significantly reduced for the reputation model, because context manager has effectively coped with context changes, and reduces the possibility of service failure. Figure 8 shows study with service failure ratio. It shows two phases of results; before and after the 1st reconfiguration, driven by the composition process and service reconfiguration, respectively. For both cases, the reputation driven model brings much less failures than no-reputation model. Moreover, the non-reputation model has a higher failure ratio on the second phase than the first. However, for the reputation-driven model, the failure ratio is significantly reduced on the second phase, because selecting user's context friendly services brings less possibility of service failure.

VII. Conclusions

In this paper, we present an approach to enhance service composition and reconfiguration by using context-based reputation for services. Some locally-developed services may perform very well at specific context and should be adopted whenever a user is in a favorable context. We have designed a context-based reputation generation mechanism and used a context similarity measure to filter out those services that are not a good fit given a user's context. We present the design of a Context Manager to manage context and reputation in the SOA middleware that is used to compose and reconfigure business processes with pre-selected services.

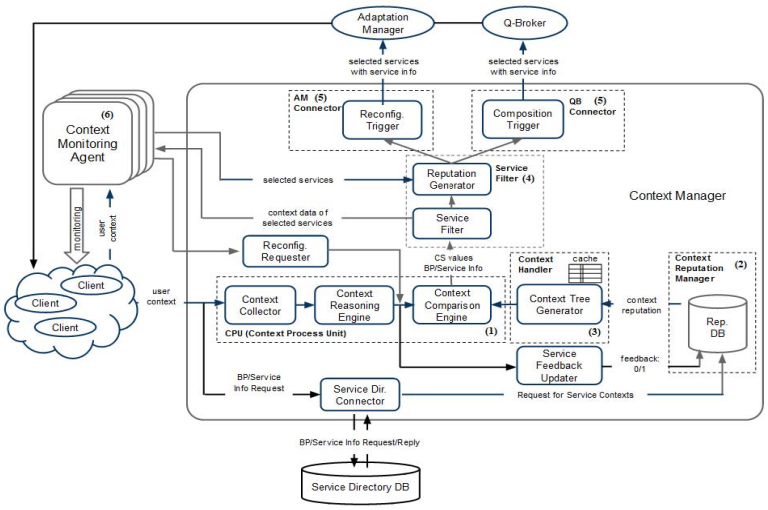

Figure 4. Context Manager Architecture

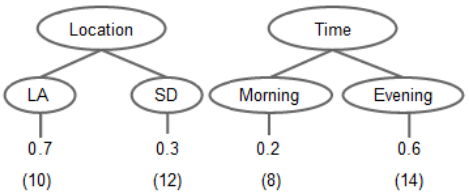

Figure 5. Service Reputation

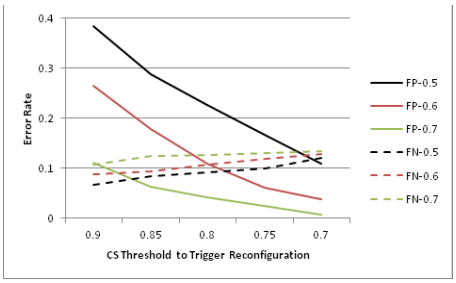

Figure 6. FP and FN Rate

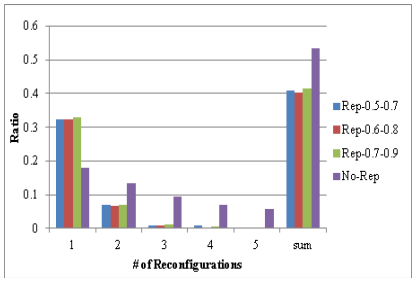

Figure 7. Number of Reconfiguration

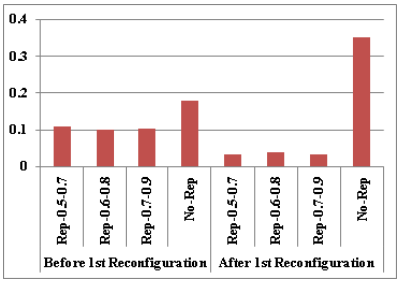

Figure 8. Service Failure Ratio

REFERENCES

[1] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, and W. V. d. Velde. Advanced interaction in context, In Proc. of the 1st International Symposium on Handheld and Ubiquitous Computing. Heidelberg, Germany: Springer-Verlag, 1999.

[2] B. N. Schilit, N. Adams, and R. Want. Context-aware computing applications, In Proc. of the 1st International Workshop on Mobile Computing Systems and Applications, Los Alamitos, CA: IEEE, 1994, pp. 85- 90.

[3] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, Computer Science, 2000.

[4] H. L. Truong and S. Dustdar. A survey on context-aware web service systems. International Journal of Web Information Systems, 5(1):5–31, 2009.

[5] J. Lee, and K.J. Lin, Context-Aware Distributed Reputation Management System In Proc. of IEEE International Conference on e-Business Engineering (ICEBE'08), Xian, China, 2008, pp. 61-68.

[6] J. Zhong, H. Zhu, J. Li, and Y. Yu, Conceptual Graph Matching for Semantic Search, International Conference on Computational Science (ICCS), Borovets, July 2002.

[7] K. J. Lin, M. Panahi, Y. Zhang, S. H. Chang, and J. Zhang. Building accountability middleware to support dependable SOA, IEEE Internet Computing, 13:16–25, March 2009.

[8] M. Panahi, W. Nie, and K.J. Lin, RT-Llama: Providing Middleware Support for Real-Time SOA. International Journal of Systems and Service-Oriented Engineering, Vol. 1, Issue 1, 2010, pp. 62-78.

[9] M. Kirsch-Pinheiro, Y. Vanrompay, Y. Berbers, Context-Aware Service Selection Using Graph Matching. 2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop, Ireland, 2008

[10] T. Gu, H. K. Pung, and D. Q. Zhang. A Service-Oriented Middleware for Building Context-Aware Services. Journal of Network and Computer Applications (JNCA), Vol. 28, Issue 1, 2005, pp. 1-18.

[11] T. Yu, Y. Zhang, and K.-J. Lin, Efficient algorithms for web services selection with end-to-end QoS constraints. ACM Transactions on the Web, May 2007.

[12] V. Tsang and S. Stevenson. A graph-theoretic framework for semantic distance. Computational Linguistics, 2010.

[13] T. Katungo, D.M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A.Y. Wu. An Efficient k-Mmeans Clustering Algorithm: Analysis and Implementation. In IEEE Transactions on Pattern Analysis and Machine Intelligence, volume 24, pages 881–892, 2002.

[14] http://www.vesag.com/

[15] http://www.lifecomm.com/index.html

[16] P. Leitner, A. Michlmayr, F. Rosenberg. and S. Dustdar, Selecting Web Services based on past user experiences. In: Proc. of IEEE Asia-Pacific Services Computing Conference, 2009.