







A User Interface for Personalized Web Service Selection in Business Processes

Dionisis Margaris¹ , Dimitris Spiliotopoulos²  ,
Costas Vassilakis² , and Gregory Karagiorgos³

¹ Department of Informatics and Telecommunications,
University of Athens, Athens, Greece
margaris@di.uoa.gr

² Department of Informatics and Telecommunications,
University of the Peloponnese, Tripoli, Greece
{dspiliot, costas}@uop.gr

³ Department of Digital Systems, University of the Peloponnese, Sparti, Greece
greg@us.uop.gr

Abstract. Nowadays, due to the huge volume of information available on the web, the need for personalization is more than necessary. Choosing the right information for each user is as important as the way this information is presented to him or her. Currently, user-triggered recommendation requests for web services are implemented as an automatic recommendation based on parametric computation. This work reports on a specialized user interface for business processes, where writing code entails invocation of business process information. The paper presents the user interface design for Personalized Web Service Selection in Business Process scenario execution and the user evaluation by business process engineers.

Keywords: User interface · Personalization · Web services · Adaptation · Business processes · WS-BPEL

1 Introduction

Nowadays, web services (WSs) are typically offered by multiple providers and hence under different quality of service (QoS) parameters, such as availability, cost, response time and so forth [1]. In this context, it is extremely useful for users to be able to specify the QoS criteria that a WS should fulfill and accordingly the software designed to model business processes should select only WSs that meet the QoS criteria set by users. In this direction, WS-BPEL is the typical language that allows for building high-level business processes through orchestrating individual WSs. Contemporary research allows users not only to select which processes they believe that better support their needs, but also to ask for a WS recommendation, specifying their desirable QoS criteria [2–5].

For example, in a summer holiday planning business process, the user may request a specific luxury hotel service and a specific car rental service and may then ask for a recommendation concerning the airline ticket. The user may also specify that the price

of the flight must be relatively “low” or at least “medium”, while at the same time the invoked WS’s reliability must be “high”. Many research works exist that support the aforementioned operation [6–9], however all of them invoke the WS which they compute that it better covers the user’s need, rather than return a list of the recommended WSs to the user, and let him decide which one he really wants to invoke. As a result, in many cases, the invoked WS is not the one that a user would select on his own and, hence, from a personalization point of view, the WS-BPEL scenario execution adaptation typically fails.

In this work, we present a user interface (UI) for personalized WS selection in WS-BPEL scenarios, which accommodates the QoS criteria set by the users. The main objective is to create a UI that can efficiently present the recommended WSs to WS-BPEL users, so they can decide which WS better fulfills their needs and finally invoke the selected one.

The rest of the paper is structured as follows: Sect. 2 overviews related work, while Sect. 3 presents the necessary foundations for our work, from the area of business processes and WSs, for self-containment purposes. Section 4 discusses the UI design, rationale and the paper experiment, while Sect. 5 presents the results of the user evaluation. Finally, Sect. 6 concludes the paper and outlines the future work.

2 Related Work

Since the information systems in the contemporary WWW are very frequently nowadays implemented by composing WSs offered by individual providers, the adaptation process of WS scenarios is a research field that has attracted significant research efforts over the last years [10–13].

Chen et al. [14] present a QoS-aware WS composition method by multi-objective optimization to assist users make variable decisions. The problem of QoS-aware WS composition is formulated to a multi-objective optimization model where the individual optimization objective is either QoS performance or risk, in order to solve the aforementioned model, an efficient multi-objective evolutionary algorithm is developed. Rodriguez-Mier et al. [15] present a theoretical analysis of graph-based service composition utilizing its dependency with WS discovery. Their work defines a composition framework integrating fine-grained I/O service discovery that enables graph-based composition generation. The composition includes the semantically relevant set of services for input-output requests. Their proposed framework also presents an optimal composition search algorithm for extracting the optimal composition from the graph by minimizing the number and the size of WSs, as well as several graph optimizations for improved system scalability. In order to model WSs with uncertain effects, Wang et al. [16] propose an approach that introduces branch structures into composite solutions to cope with uncertainty in the service composition process. Liu et al. [17] present a WS composition method based on QoS dynamic prediction and global QoS constraints decomposition. Their approach includes 2 phases. The first phase happens before service composition, by decomposing global QoS constraints into local ones, thereby transforming the WS dynamic composition problem to a local optimization one. The second phase happens at runtime, using predicted QoS values to select the optimal WS

for the current abstract service. Margaris et al. [4] perform a horizontal QoS-based adaptation, supporting both the parallel and sequential execution structures within the WS-BPEL scenario, while Margaris et al. [18] present an associated execution framework and employ collaborative filtering (CF) techniques to direct the adaptation. However, this approach uses very limited QoS-based criteria (optionally specifying lower and upper bounds for each QoS attribute), hence running the risk of inferior overall QoS of the formulated solutions when compared to the possibly attainable optimal composition QoS, especially for the cases where CF has to address known issues such as gray sheep and cold start. Margaris et al. [7] present a CF-based algorithm which also takes into account the WSs' QoS parameters, in order to personalize the business processes execution to the users' preferences. Furthermore, they introduce an offline clustering technique for supporting the scalable and efficient execution of the presented algorithm under the presence of large volumes of data in the WS repository. Cardellini et al. [19] present a software platform supporting QoS-driven adaptation of WS-oriented systems, named MOSES, which integrates within a unified framework different adaptation mechanism, achieving a greater flexibility in facing various operating environments and the possibly conflicting QoS requirements of several concurrent users.

Although all the aforementioned works successfully address the WS adaptation problem, they also complete the adaptation process, by invoking the WS that they have found that better satisfies the WS-BPEL designer rather than only recommend one or more WSs to the WS-BPEL designer and let him select the WS he prefers the most, as all the modern general purpose RSs do [20–26].

The complexity of the enterprise solutions that interconnect with several services, locally or in the cloud, affect the performance of systems and users [27–30]. To effectively implement business processes, graphical notation can be useful, especially when used for process driven WS-BPEL design [31]. Haihong et al. [32] propose an asynchronous, process-driven mechanism for easier combination of web services. Jose et al. [33] extend the process driven approach using ontologies to describe the relevant aspects and the user interface to display the type of match to the business process WS. Another approach is the use of static call graphs to contain the semantic knowledge, which is then used for the generation of WS-BPEL test cases that support the sub-processes between WS-BPEL files [34].

The aforementioned approaches to user interface design, try to automate the WS selection matching processes. However, none of those works attempt to simplify the user interaction. The necessity for involvement of the user in more complex tasks, casts a burden to the WS-BPEL designer, resulting in reduced usability of the programming environment. This is similar to complexity-induced cognitive load that is usually present in complex user interfaces [35, 36], multi-source information processing [37, 38] and multimodal interaction [39–42].

In this work, we approach the WS-BPEL design through a user centric view, catering to the preference and selection of WS candidates by the designer. The WS-BPEL designer may have preference and selection criteria that differ per business process and that may apply to ones that are matched to the same parameters.

The motivation is to examine how the WS-BPEL designers make selections from multiple choice and define how the user interface could support the interaction.

3 Prerequisites

The following subsections include a summarize concepts and underpinnings from the domains of WS QoS attributes and WS substitution relationships that are used in our work, for conciseness purposes.

3.1 Web Services QoS Attributes

WS QoS attributes describe measurements of the overall performance of a WS [43, 44]. For conciseness purposes and without loss of generality, in this paper we will consider only the attributes *cost* (*c*), *response time* (*rt*) and *reliability* (*rel*), adopting their definitions from Comerio et al. [45].

In the context of a WS-BPEL scenario execution, a user may want to define constraints (i.e. upper and lower bounds) on the QoS delivered by each WS executed in the context of a specific invocation.

More specifically, for each WS ws_x included in a WS-BPEL scenario, the user may provide the following two vectors:

- $MIN_x(min_{rt,x}, min_{c,x}, min_{rel,x})$ and
- $MAX_x(max_{rt,x}, max_{c,x}, max_{rel,x})$.

where the MAX_x vector includes the upper bounds of the QoS attributes for the WS implementation that will be selected to realize the functionality of s_x (e.g. booking an airticket) and the MIN_x vector includes the respective lower bounds.

Additionally, the user provides a weight vector $W(rt_w, c_w, rel_w)$, indicating the importance of each attribute in the context of the particular invocation of the WS-BPEL scenario, which is multiplied by the value of the corresponding QoS attribute, and these products are then aggregated in order to produce the WS-BPEL scenario's overall score. It has to be mentioned that, in contrary to the upper and lower bound vectors, which are applied to each WS invocation, the weight vector is applied to the whole WS-BPEL scenario (the whole composition).

As far as attribute values are concerned, clearly different attributes may have different measurement units (e.g. milliseconds, Euros and kilograms), hence in order to have meaningful results from combining all the attribute values of a WS-BPEL, we consider that all attributes are normalized in the range [0, 10], through the application of a standard normalization formula [46]:

$$norm(v) = 10 * \frac{v - \min(v)}{\max(v) - \min(v)} \quad (1)$$

where v is the value to be normalized, while $\max(v)$ and $\min(v)$ are the maximum and minimum values of the corresponding attribute in the database extension. Furthermore, we consider that larger attribute values correspond to higher QoS levels: for

attributes where the inverse holds (such as latency and price), the following transformation must be done (altering the normalization formula):

$$norm(v) = 10 * \left(1 - \frac{v - \min(v)}{\max(v) - \min(v)} \right) \tag{2}$$

This approach is typically followed for the handling of QoS attribute values in the context of WS selection and composition [4, 11, 47, 48].

An example of the repository form is shown in Table 1, containing three indicative WSs.

Table 1. Example of repository contents.

Service	Response time	Cost	Reliability
S ₁	6	6	6
S ₂	8	8	4
S ₃	2	2	2

It has to be noted that our framework is also able to handle (a) sequential and parallel (concurrent invocations) structures, (b) service selection affinity and (c) exception resolution in WS-BPEL scenarios. For more details on the aforementioned concepts, the interested reader is referred to [4].

3.2 Substitution Relationship Representation

When a WS-BPEL scenario is adopted to match the QoS designations provided by the user, the execution adaptation process selects WS service implementations to realize functionalities that appear in the context of the WS-BPEL scenario. Therefore, the adaptation software needs to be able to test whether within a particular execution, a WS service implementation *A* can realize some requested functionality *B*. In this work, in order to address this issue, we use the WS matchmaking relationships [49], where any WS that either (i) provides the *same functionality as F* or (ii) provides a *more specific functionality than F*. For more information on matchmaking relationships, the interested reader is referred to [18, 49].

An example of a WS taxonomy is depicted in Fig. 1, concerning an airline ticket booking service. In this figure, the orange shaded rectangles (upper level) represent (sub-)categories, while the white shaded rectangles representing WS implementations. The line between two rectangles denotes that the higher-level node includes the lower-level one in a superclass-subclass relation.

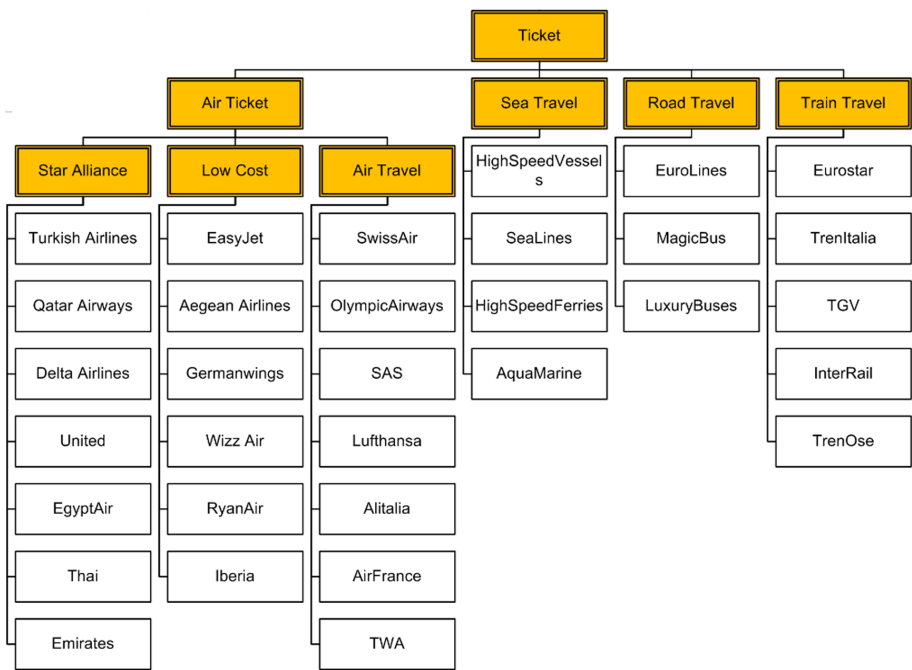


Fig. 1. Hierarchy of WSs (sub-)categories and implementations for the airline ticket booking service.

Each node corresponding to a WS implementation stores information regarding its QoS attributes values in the WS repository. An excerpt of this repository is depicted in Fig. 2.

Category: Ticket

Category: Air Ticket

Category: Star Alliance

Implementation: Turkish Airlines (rt=6, cost=2, rel=9)

Implementation: Qatar Airways (rt=5, cost=8, rel=3)

Implementation: Delta Airlines (rt=2, cost=6, rel=6)

Fig. 2. Excerpt of the WS taxonomy repository.

3.3 WS-BPEL Scenario and Dataset Example

To exemplify the proposed approach, we will use the case of adapting the invocation of a simple WS-BPEL scenario (for simplicity only one sequential construct is used). This scenario describes a business process of booking a summer holiday vacation package (modeled as a WS-BPEL scenario, termed as *SHVP*), which includes the functionalities of (i) *asking a recommendation* for an airline ticket, (ii) booking a specific luxury hotel room (explicitly defined by the user) and (iii) renting a car (also explicitly defined by the user). Figure 3 depicts the pseudocode of the aforementioned WS-BPEL scenario.

```
SHVP
WEIGHTS(respTime=0.2, cost=0.3, reliability=0.5)
SEQ
  (name=bookAirTravel, REC, Ticket / Air Ticket / Low Cost , min=(-,3,5), max=(-,9,-) )
  (name=bookHotel, INV, "Hilton")
  (name=bookCarRental, INV, "Rent a Car")
END_SEQ
```

Fig. 3. Pseudocode of business process execution request example

In Fig. 3, the *WEIGHTS* entity corresponds to the weight vector that indicates the importance of each attribute in the context of the particular adaptation (as defined in Subject. 3.1). Then, each construct follows, either indicating an invocation to a specific WS (term *INV*) or asking for a WS recommendation (term *REC*). In the latter case, the user must enter the full path to the WS (sub-)category in the repository (see Fig. 1), where the WS to be recommended must belong (in our example, the user is interested in booking a *Low Cost Air Ticket*), along with the lower and upper bounds for the QoS attributes follow (the ‘-’ symbol is used when a user does not want to set an upper/lower bound for a specific attribute).

4 Experiment

The user interface design was evaluated by 9 WS-BPEL designers, each with more than 3 years of experience in BPEL design using various commercial or in-house software development platforms. The mean age of the participants was 34 years old, 7 male and 2 female. The participants took part in the study using a custom BPEL design IDE designed to provide the facilitators the choice to adjust the number of items to display as well as alternative visualization elements for standard BPEL.

The participants were asked to create simple BPEL code (consisting only of WS invocations) using the pseudocode supported by the user interface. They were presented with visual elements for WS candidates and parameter fine-tuning options. Each participant was asked to design two randomly assigned business processes, out of the five in total that were available as scenarios, that included at least two explicitly user-defined functionalities.

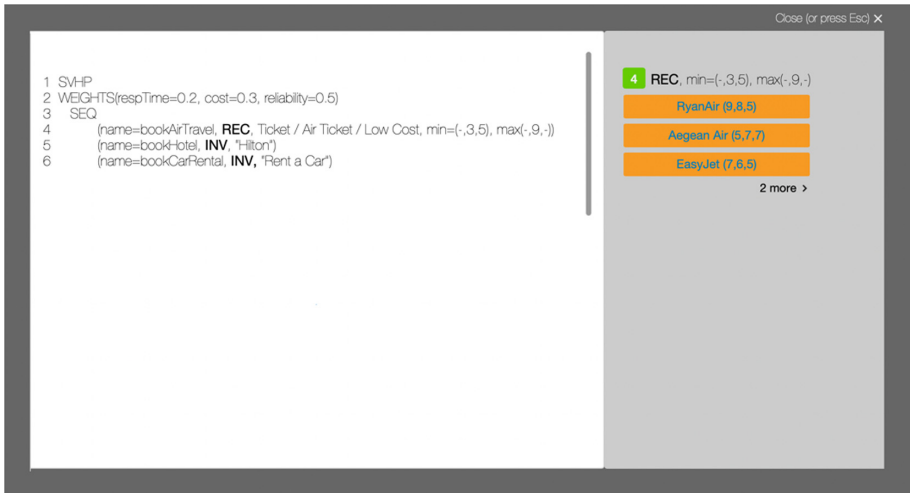


Fig. 4. BPEL design environment that supports recommendation of multiple functionality options and user selection.

The participants were asked to complete the BPEL design and report on their user experience. Before the study, they had the opportunity to engage in BPEL design using the standard interface and parameterization options albeit without the recommendation options.

Figure 4 depicts the user that typed a *REC* command and the parameters of his choice. The interface, once the line feed was pressed, triggered the recommendation algorithm and calculated the functionalities that satisfied the user criteria. The functionalities are shown on the right panel. The right panel is showing the line number, the *REC* information and the list of functionalities, ordered (descending) based on their total calculated score. Therefore, the first airline has a total score of 6.7, the second has a total score of 6.6 and so on.



Fig. 5. BPEL design environment depicting recommendation and user selection for a WS-BPEL scenario.

The user decided that the most fulfilling functionality was the second one in the ordered list since it has a higher reliability score. Based on his personal preference for the higher reliability score, the user selected the second choice and the code was auto filled with that information, changing the *REC* to *INV* with the selected value (Fig. 5). The former *REC* is shown in green, allowing the user to click to re-select, if required. The right panel now shows the user current choice. The re-selection can be triggered from either the code or the right panel.

5 Evaluation

The participants were asked to design two business processes and report their rationale to the facilitators using the think-aloud method. It was the method of choice based on previous experience with professional developers and IDE environments. From the objective metrics, 83% of the times the users selected a functionality, their choice was not the first presented functionality (WS). That WS would have been automatically invoked in traditional approaches, requiring additional human effort and time to repair (and in most cases the result is irreversible). This shows that the proposed approach fills a gap in the core functionality of WS-BPEL design user interfaces and builds toward better user experience.

After the end of the session, the participants were asked to fill in a usability questionnaire and then invited to discuss suggestions for improvement. Figure 6 shows that the users reported very high acceptance and relatively high satisfaction. Regarding the latter, the follow up discussion was focused on the next steps of the user interface development and additional functionalities.

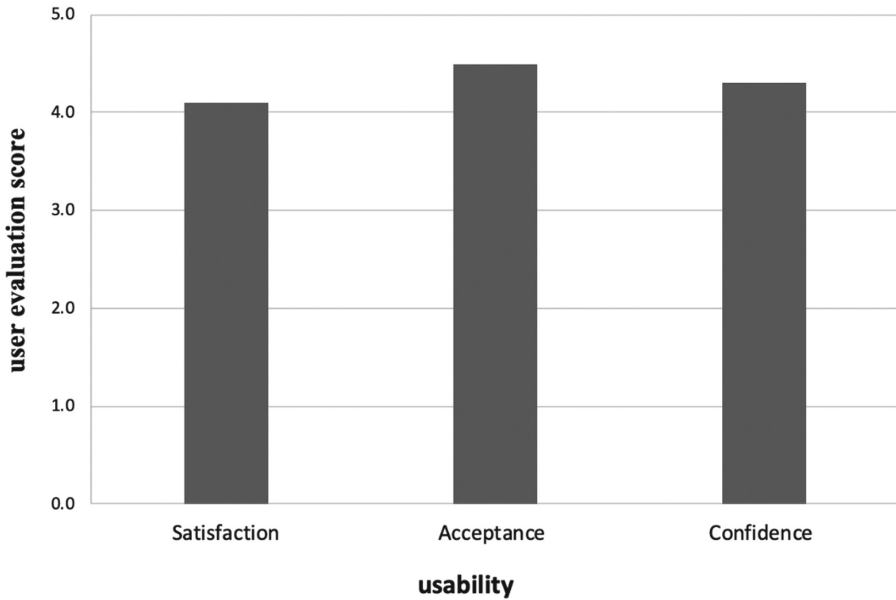


Fig. 6. BPEL design environment usability evaluation overview.

Most of the users reported that they would use this IDE in its current state for their WS-BPEL design. All users mentioned that they would definitely use the user interface if additional functionality for BPEL overview is available. The overview was defined as a visual summary of their choices and the capability to edit and fine tune specific choices for overall targets, such as maximum cost.

6 Conclusion and Future Work

This paper presented a specialized UI for WS-BPEL designers that allows personalized recommendation and selection of business process functionalities based on user generated criteria. The proposed design allows WS-BPEL designers to ask for recommendations and select functionalities out of ordered lists, based on the total or particular scores, per user choice. Allowing the users to have the final selection choice, mitigates the issues caused by automatic system selection that leads to adaptation failure, as shown in the experimental study. The user empowerment is further strengthened from the capability to edit back the selections at will, via the UI.

The UI and the recommendation-selection process were evaluated by WS-BPEL designers. The majority of the participants made selections that were not the highest scoring (based on the total score) in the ordered list. This reinforces the identified requirements and needs that user personalization is reflected in the user-entered parameters. Additionally, it clearly shows that scores are perceived by the users differently and so does their importance in the user selection.

The future work has been identified during the user evaluation. The users suggested that the same recommendation-selection capability be expanded on the business process level, providing an overview of the selections and the functionality to edit individual selections to achieve global scores, such as maximum reliability or minimized costs.

References

1. O'Sullivan, J., Edmond, D., ter Hofstede, A.: What's in a service? *Distrib. Parallel Databases* **12**, 117–133 (2002). <https://doi.org/10.1023/A:1016547000822>
2. Margaris, D., Vassilakis, C., Georgiadis, P.: An integrated framework for adapting WS-BPEL scenario execution using QoS and collaborative filtering techniques. *Sci. Comput. Program.* **98**, 707–734 (2015). <https://doi.org/10.1016/j.scico.2014.10.007>
3. Furusawa, Yu., Sugiki, Y., Hishiyama, R.: A web service recommendation system based on users' reputations. In: Kinny, D., Hsu, J.Y., Governatori, G., Ghose, A.K. (eds.) *PRIMA 2011. LNCS (LNAI)*, vol. 7047, pp. 508–519. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25044-6_41
4. Dionisis, M., Costas, V., Panagiotis, G.: An integrated framework for QoS-based adaptation and exception resolution in WS-BPEL scenarios. In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC 2013*, p. 1900. ACM Press, New York (2013). <https://doi.org/10.1145/2480362.2480714>
5. Chen, X., Zheng, Z., Yu, Q., Lyu, M.R.: Web service recommendation via exploiting location and QoS information. *IEEE Trans. Parallel Distrib. Syst.* **25**, 1913–1924 (2014). <https://doi.org/10.1109/TPDS.2013.308>
6. Mukherjee, D., Jalote, P., Gowri Nanda, M.: Determining QoS of WS-BPEL compositions. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) *ICSOC 2008. LNCS*, vol. 5364, pp. 378–393. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89652-4_29
7. Margaris, D., Georgiadis, P., Vassilakis, C.: A collaborative filtering algorithm with clustering for personalized web service selection in business processes. In: *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, pp. 169–180 (2015). <https://doi.org/10.1109/RCIS.2015.7128877>
8. Dionisis, M., Costas, V., Panagiotis, G.: A hybrid framework for WS-BPEL scenario execution adaptation, using monitoring and feedback data. In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing – SAC 2015*, pp. 1672–1679. ACM Press, New York (2015). <https://doi.org/10.1145/2695664.2695687>
9. Margaris, D., Vassilakis, C., Georgiadis, P.: Improving QoS delivered by WS-BPEL scenario adaptation through service execution parallelization. In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pp. 1590–1596. Association for Computing Machinery, New York (2016). <https://doi.org/10.1145/2851613.2851805>
10. Gupta, R., Kamal, R., Suman, U.: A QoS-supported approach using fault detection and tolerance for achieving reliability in dynamic orchestration of web services. *Int. J. Inf. Technol.* **10**(1), 71–81 (2017). <https://doi.org/10.1007/s41870-017-0066-z>
11. Halfaoui, A., Hadjila, F., Didi, F.: QoS-aware web services selection based on fuzzy dominance. In: Amine, A., Bellatreche, L., Elberichi, Z., Neuhold, E.J., Wrembel, R. (eds.) *CIAA 2015. IAICT*, vol. 456, pp. 291–300. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19578-0_24

12. Comes, D., Baraki, H., Reichle, R., Zapf, M., Geihs, K.: Heuristic approaches for QoS-based service selection. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) *ICSOC 2010*. LNCS, vol. 6470, pp. 441–455. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17358-5_30
13. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for Web services composition. *IEEE Trans. Softw. Eng.* **30**, 311–327 (2004). <https://doi.org/10.1109/TSE.2004.11>
14. Chen, F., Dou, R., Li, M., Wu, H.: A flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing. *Comput. Ind. Eng.* **99**, 423–431 (2016). <https://doi.org/10.1016/j.cie.2015.12.018>
15. Rodriguez-Mier, P., Pedrinaci, C., Lama, M., Mucientes, M.: An integrated semantic web service discovery and composition framework. *IEEE Trans. Serv. Comput.* **9**, 537–550 (2016). <https://doi.org/10.1109/TSC.2015.2402679>
16. Wang, P., Ding, Z., Jiang, C., Zhou, M., Zheng, Y.: Automatic web service composition based on uncertainty execution effects. *IEEE Trans. Serv. Comput.* **9**, 551–565 (2016). <https://doi.org/10.1109/TSC.2015.2412943>
17. Liu, Z.Z., Jia, Z.P., Xue, X., An, J.Y.: Reliable Web service composition based on QoS dynamic prediction. *Soft. Comput.* **19**(5), 1409–1425 (2014). <https://doi.org/10.1007/s00500-014-1351-4>
18. Margaritis, D., Georgiadis, P., Vassilakis, C.: Adapting WS-BPEL scenario execution using collaborative filtering techniques. In: *Proceedings - International Conference on Research Challenges in Information Science*, pp. 174–184 (2013). <https://doi.org/10.1109/RCIS.2013.6577691>
19. Cardellini, V., Casalicchio, E., Grassi, V., Iannucci, S., Lo Presti, F., Mirandola, R.: MOSES: a platform for experimenting with QoS-driven self-adaptation policies for service oriented systems. In: de Lemos, R., Garlan, D., Ghezzi, C., Giese, H. (eds.) *Software Engineering for Self-Adaptive Systems III. Assurances*. LNCS, vol. 9640, pp. 409–433. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-74183-3_14
20. Aivazoglou, M., et al.: A fine-grained social network recommender system. *Soc. Netw. Anal. Min.* **10**(1), 1–18 (2019). <https://doi.org/10.1007/s13278-019-0621-7>
21. Margaritis, D., Vassilakis, C., Georgiadis, P.: Query personalization using social network information and collaborative filtering techniques. *Futur. Gener. Comput. Syst.* **78**, 440–450 (2018). <https://doi.org/10.1016/j.future.2017.03.015>
22. Margaritis, D., Vassilakis, C., Spiliotopoulos, D.: What makes a review a reliable rating in recommender systems? *Inf. Process. Manag.* **57**, 102304 (2020). <https://doi.org/10.1016/j.ipm.2020.102304>
23. Margaritis, D., Vassilakis, C.: Exploiting Internet of Things information to enhance venues' recommendation accuracy. *SOCIA* **11**(4), 393–409 (2017). <https://doi.org/10.1007/s11761-017-0216-y>
24. Margaritis, D., Vassilakis, C., Georgiadis, P.: Recommendation information diffusion in social networks considering user influence and semantics. *Soc. Netw. Anal. Min.* **6**(1), 1–22 (2016). <https://doi.org/10.1007/s13278-016-0416-z>
25. Margaritis, D., Vassilakis, C., Spiliotopoulos, D.: Handling uncertainty in social media textual information for improving venue recommendation formulation quality in social networks. *Soc. Netw. Anal. Min.* **9**(1), 1–19 (2019). <https://doi.org/10.1007/s13278-019-0610-x>
26. Margaritis, D., Kobusinska, A., Spiliotopoulos, D., Vassilakis, C.: An adaptive social network-aware collaborative filtering algorithm for improved rating prediction accuracy. *IEEE Access* **8**, 68301–68310 (2020). <https://doi.org/10.1109/ACCESS.2020.2981567>

27. Sturm, R., Pollard, C., Craig, J.: Application programming interfaces and connected systems. In: *Application Performance Management (APM) in the Digital Enterprise*, pp. 137–150. Elsevier (2017). <https://doi.org/10.1016/B978-0-12-804018-8.00011-5>
28. Risse, T., et al.: The ARCOMEM architecture for social- and semantic-driven web archiving. *Futur. Internet*. **6**, 688–716 (2014). <https://doi.org/10.3390/fi6040688>
29. Demidova, E., et al.: Analysing and enriching focused semantic web archives for parliament applications. *Futur. Internet*. **6**, 433–456 (2014). <https://doi.org/10.3390/fi6030433>
30. Bernaschina, C., Falzone, E., Fraternali, P., Gonzalez, S.L.H.: The virtual developer. *ACM Trans. Softw. Eng. Methodol.* **28**, 1–38 (2019). <https://doi.org/10.1145/3340545>
31. Stiehl, V., Danei, M., Elliott, J., Heiler, M., Kerwien, T.: Effectively and efficiently implementing complex business processes: a case study. In: Lübke, D., Pautasso, C. (eds.) *Software Engineering for Self-Adaptive Systems III. Assurances*. LNCS, vol. 9, pp. 33–57. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17666-2_3
32. Haihong, E., Lin, Y., Song, M., Xu, X., Zhang, C.: A visual web service composition system based on process tree. In: *2019 5th International Conference on Information Management (ICIM)*, pp. 274–278. IEEE (2019). <https://doi.org/10.1109/INFOMAN.2019.8714694>
33. Jose, H.S.A.S., Cappelli, C., Santoro, F.M., Azevedo, L.G.: Implementation of aspect-oriented business process models with web services. *Bus. Inf. Syst. Eng.* **17**(1), 1–24 (2020). <https://doi.org/10.1007/s12599-020-00643-2>
34. Bousanoh, W., Suwannasart, T.: Test case generation for WS-BPEL from a static call graph. *J. Phys. Conf. Ser.* **1195**, 12004 (2019). <https://doi.org/10.1088/1742-6596/1195/1/012004>
35. Pino, A., Kouroupetoglou, G., Kacorri, H., Sarantidou, A., Spiliotopoulos, D.: An open source/freeware assistive technology software inventory. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A. (eds.) *ICCHP 2010*. LNCS, vol. 6179, pp. 178–185. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14097-6_29
36. Androutsopoulos, I., Spiliotopoulos, D., Stamatakis, K., Dimitromanolaki, A., Karkaletsis, V., Spyropoulos, C.D.: Symbolic authoring for multilingual natural language generation. In: Vlahavas, I.P., Spyropoulos, C.D. (eds.) *SETN 2002*. LNCS (LNAI), vol. 2308, pp. 131–142. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46014-4_13
37. Schefbeck, G., Spiliotopoulos, D., Risse, T.: The Recent challenge in web archiving: archiving the social web. In: *Proceedings of the International Council on Archives Congress*, pp. 1–5 (2012)
38. Antonakaki, D., Spiliotopoulos, D., Samaras, C. V., Ioannidis, S., Fragopoulou, P.: Investigating the complete corpus of referendum and elections tweets. In: *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2016*, pp. 100–105 (2016). <https://doi.org/10.1109/ASONAM.2016.7752220>
39. Kouroupetoglou, G., Spiliotopoulos, D.: Usability methodologies for real-life voice user interfaces. *Int. J. Inf. Technol. Web Eng.* **4**, 78–94 (2009). <https://doi.org/10.4018/jitwe.2009100105>
40. Xydias, G., Spiliotopoulos, D., Kouroupetoglou, G.: Modeling emphatic events from non-speech aware documents in speech based user interfaces. *Proc. Hum. Comput. Interact.* **2**, 806–810 (2003)
41. Spiliotopoulos, Dimitris., Xydias, Gerasimos, Kouroupetoglou, Georgios: Diction based prosody modeling in table-to-speech synthesis. In: Matoušek, Václav, Mautner, Pavel, Pavelka, Tomáš (eds.) *TSD 2005*. LNCS (LNAI), vol. 3658, pp. 294–301. Springer, Heidelberg (2005). https://doi.org/10.1007/11551874_38

42. Spiliotopoulos, D., Stavropoulou, P., Kouroupetroglou, G.: Acoustic rendering of data tables using earcons and prosody for document accessibility. In: Stephanidis, C. (ed.) UAHCI 2009. LNCS, vol. 5616, pp. 587–596. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02713-0_62
43. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for QoS-aware service composition based on genetic algorithms. In: Proceedings of the 2005 Conference on Genetic and evolutionary computation - GECCO 2005, p. 1069. ACM Press, New York (2005). <https://doi.org/10.1145/1068009.1068189>
44. Hammas, O., Ben Yahia, S., Ben Ahmed, S.: Adaptive web service composition insuring global QoS optimization. In: 2015 International Symposium on Networks, Computers and Communications (ISNCC), pp. 1–6. IEEE (2015). <https://doi.org/10.1109/ISNCC.2015.7238593>
45. Comerio, M., De Paoli, F., Grega, S., Maurino, A., Batini, C.: WSMoD. *Int. J. Web Serv. Res.* **4**, 33–60 (2007). <https://doi.org/10.4018/jwsr.2007040102>
46. Daqing He, Wu, D.: Toward a robust data fusion for document retrieval. In: 2008 International Conference on Natural Language Processing and Knowledge Engineering, pp. 1–8. IEEE (2008). <https://doi.org/10.1109/NLPKE.2008.4906754>
47. Liu, Y., Ngu, A.H., Zeng, L.Z.: QoS computation and policing in dynamic web service selection. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters - WWW Alternate 2004, p. 66. ACM Press, New York (2004). <https://doi.org/10.1145/1013367.1013379>
48. Margaris, D., Georgiadis, P., Vassilakis, C.: On replacement service selection in WS-BPEL scenario adaptation. In: Proceedings - 2015 IEEE 8th International Conference on Service-Oriented Computing and Applications, SOCA 2015, pp. 10–17 (2015). <https://doi.org/10.1109/SOCA.2015.11>
49. Bellur, U., Kulkarni, R.: Improved matchmaking algorithm for semantic web services based on bipartite graph matching. In: IEEE International Conference on Web Services (ICWS 2007), pp. 86–93. IEEE (2007). <https://doi.org/10.1109/ICWS.2007.105>