





# A User Interface for Tuning QoS Parameters in Recommendation-Based Business Process Scenario Adaptation

Kiriakos Sgardelis<sup>1</sup>, Dionisis Margaris<sup>1</sup>, Dimitris Spiliotopoulos<sup>2</sup> and Costas Vassilakis<sup>3</sup>

<sup>1</sup>Department of Digital Systems, University of the Peloponnese, Sparta, Greece

<sup>2</sup>Department of Management Science and Technology, University of the Peloponnese, Tripoli, Greece

<sup>3</sup>Department of Informatics and Telecommunications, University of the Peloponnese, Tripoli, Greece

**Keywords:** UI, Adaptation, Personalisation, Recommender Systems, Business Processes, WS-BPEL, Web Services.


**Abstract:** The Web Services Business Process Execution Language (BPEL) is a special-purpose language that orchestrates web services into a high-level business process. A typical BPEL scenario contains invocations to preselected web services, along with their parameters. However, many recent research works support dynamic service selection, based on user-set policies and criteria. Furthermore, users may request a service recommendation, in which case functionally equivalent service offerings by different providers will be considered by the personalization module. Along with the recommendation request, users provide the policy parameters, which include minimum and maximum bounds for the non-functional attributes concerning the service, and the system exploits these bounds to select and use the optimal candidate services. However, in many real-life cases, a person will accept/purchase a product or a service that exceeds the threshold(s) that initially he/she has set, e.g., if the overhead is marginal or the offer is deemed appealing, or no satisfactory service candidates are identified using the initial settings. In this paper, we present and evaluate a specialized User Interface that allows the user to review service candidates marginally exceeding the specified bounds and consider them while making the final service selection.


## 1 INTRODUCTION


The Web Services Business Process Execution Language, mostly known as WS-BPEL or simply BPEL, is a special-purpose programming language suitable for designing and executing business processes that comprise web services invocations (Fu et al., 2004; Pasley, 2005; Ouyang et al., 2006). A typical BPEL scenario supports direct invocations to individual web services, along with their parameters; invocations are organized into sequential and/or parallel execution flows, while additionally control flow structures (e.g., *if*, *switch* and *while*) can be specified (Mukherjee et al., 2008; Deng et al., 2011).


While in the typical BPEL scenario all the invoked web services must be known in advance, many extensions proposed insofar allow the BPEL execution engine to select at runtime the web services,

considering the Quality of Service (QoS) parameters of the service, i.e., non-functional parameters that relate to the service, such as cost, response time, reliability, etc. (Mukherjee et al., 2008; Deng et al., 2011). In particular, when executing a BPEL scenario, the user provides specifications for the Quality of Service (QoS) he wants to receive, and the adaptation/personalization engine uses these specifications to select and invoke services that best match the user's specifications (Karastoyanova and Leymann, 2009; Christos et al., 2009; Margaris et al., 2015a). This is highly desirable, since many services can be supported by multiple providers (e.g., a hotel accommodation in Rome, a flight from Athens to Rome, etc.), where each one may have different QoS attribute values (cost, availability, etc.). Furthermore, to fully support the aforementioned functionality, modern BPEL research works include a web service

 <https://orcid.org/0009-0008-7113-8855>

 <https://orcid.org/0000-0002-7487-374X>

 <https://orcid.org/0000-0003-3646-1362>

 <https://orcid.org/0000-0001-9940-1821>

taxonomy which contains all the (sub-)categories and web service implementations (Chan et al., 2009; Wang et al., 2012; Margaris et al., 2021).

Regarding the web service selection policy, the user is typically able to set (a) the minimum and maximum values for each QoS aspect, as well as (b) the weight/importance of each QoS attribute. This enables the personalization/adaptation engine to (i) exclude the services whose QoS attributes are out of the user-specified bounds and (ii) prioritize qualifying services to either automatically fill or simply recommend the optimal web services for each user. For example, in a simple BPEL scenario, where the attributes considered are (a) cost, (b) reliability and (c) availability, the user sets  $MIN=(3, 6, 8)$ ,  $MAX=(9.5, 9, -)$  and  $W=(0.3, 0.4, 0.3)$ , indicating that the minimum values for the cost, reliability, and availability QoS attributes of the service to be selected are 3/10, 6/10 and 8/10, respectively, while at the same time the maximum values allowed for cost and reliability QoS attributes are 9.5 and 9, respectively (no maximum bound is placed for availability). Deferring, for now, the discussion on process of the attribute weight application to prioritize qualifying services, let us assume that the available services in this scenario are the following three:

- $s1 = (\text{cost: } 3.5, \text{reliability: } 6.2, \text{availability: } 8)$ ;
- $s2 = (\text{cost: } 7, \text{reliability: } 5.8, \text{availability: } 9.5)$ ;
- $s3 = (\text{cost: } 9.5, \text{reliability: } 8.5, \text{availability: } 7.7)$ .

All attribute values are encoded in a ‘higher value is better’ scheme, e.g., the reliability of  $s1$  (value: 6.2) is lower than the reliability of  $s3$  (value: 8.5), while the cost of  $s2$  (7) is higher than the cost  $s3$  (9).

The typical recommender engine will never recommend  $s2$  and  $s3$  to the user, since the reliability of  $s2$  and availability of  $s3$  are lower than the user set MIN boundaries. As a result, the service that will be recommended to the user is  $s1$ .

However, in real life many users would observe that  $s2$  is marginally rejected, since its reliability is only 0.2/10 (2%) below the boundary. Additionally, considering the other two QoS attributes,  $s2$  is superior (cost +3.5, availability +1.5), when compared to  $s1$ . As a result, many users might select  $s2$  over  $s1$ , despite the boundary. Furthermore, we can observe that  $s3$  is marginally rejected too, since its availability is only 0.3/10 (3%) below the respective boundary, however, based on the other two QoS attributes,  $s3$  is also superior when compared to  $s1$ .

It is noteworthy that service filtering according to the QoS bounds is of high importance to protect users from information overload (Aljukhadar et al., 2012)

and allow users to effectively select the desired services. Nevertheless, as demonstrated above, the strict application of filtering may eliminate choices that are potentially desirable to the user. To successfully address this issue, the users must be supported by a suitable user interface, which would inform the users about the existence of additional candidates and would allow them to review these candidates, while additionally it would maintain a high level of protection from information overload. This work (a) presents a UI which supports users to efficiently identify and review potentially attractive service candidates whose QoS parameters fall outside the specified bounds, achieving more successful personalizations and (b) evaluates the presented UI in terms of user acceptance.

The rest of the paper is structured as follows: in section 2 the related work is overviewed, while in section 3 the necessary foundations for our work are presented. Section 4 and section 5 present and analyse the proposed functionality and the overall UI design, as well as the results of the user evaluation, and finally the paper conclusion and future work are outlined in section 6.

## 2 RELATED WORK

The adaptation of WS-BPEL scenarios’ execution is a field of major research interest, and numerous publications over the recent 15 years have addressed multiple aspects of this process.

VieDAME is a BPEL extension that enables BPEL process monitoring, based on specific QoS criteria, and an adaptation strategy which, based on various selectors, is able to replace existing partner web services, which can be either semantically or syntactically equivalent in the BPEL process (Moser et al., 2008). The work in (Christos et al., 2009) introduces a framework which provides the BPEL execution engine with functionalities which comprise of QoS attributes restrictions and ranking criteria definitions, dynamic service selection, based on given user policy and exception management techniques for automating handling of exceptions due to system faults. The work in (Tragatschnig and Zdun, 2011) introduces a framework that supports runtime structural modification, both for processes and instances, and provides adaptation support to BPEL execution engines. The work in (Sun et al., 2012) introduces a BPEL fault localization guideline, which is based on the attributes of the BPEL integration-level faults. The work in (Dionisis et al., 2013) introduces a framework which gives the BPEL

designers the ability to define the qualitative requirements for the services invoked in the BPEL scenarios. This framework also supports the system-level exception resolution and service selection affinity. The work in (Margaris et al., 2013) presents an algorithm and the relative framework, which incorporates both QoS specifications and personalization, and more specifically collaborative filtering, techniques into the BPEL execution adaptation process. The work in (Alferez et al., 2014) presents a framework which includes runtime variability models, artifacts and tools, in order to support the service compositions dynamic adaptation. The work in (Margaris et al., 2020a) introduces an integer programming-based recommendation optimization algorithm for the WS-BPEL scenario, which supports user set QoS criteria. This algorithm maintains the optimality of the computed adaptations, while, at the same time, it can efficiently compute the adaptations that satisfy the QoS criteria set by users. The work in (Driss et al., 2022) introduces a Service-Oriented Computing-based approach which supports the discovery, selection and composition of the most suitable services. With this approach, both non-functional and functional requirements are specified by the BPEL designer to satisfy the QoE, QoS and QoBiz parameters and services are chosen.

Over the recent years research works concerning UIs for BPEL scenario adaptation have been introduced, as well, allowing the BPEL designer to control adaptation tasks, in addition to specifying the services to be invoked and their orchestration. The work in (Liu et al., 2016) presents an approach, namely Mobile User Interactions and Tasks, which is implemented as a standard service that can be included into the BPEL engines. This web service gives the BPEL designers the ability to realize a Web-based UI, using a domain-specific language along with a web programming abstraction. The work in (Yongchareon et al., 2018) presents a framework that produces UI flow models, to support semi-automatic creation of UIs and visualize artifact-centric processes. The presented UI is developed by taking into consideration the relationships among user roles, UIs, and business processes, in an artifact-centric process.

The work in (Margaris et al., 2020b) introduces a UI for personalized service selection in BPEL scenarios that considers the user set QoS parameters. Its main target is to efficiently display the appropriate service to the BPEL user, according to his profile, each time the user asks for a service recommendation. Afterwards, the UI provides the user the ability to select the exact service to be invoked, according to his needs. The work in (Diaz et al., 2021) takes BPMN

models as input, and with the use of stereotypes and Class Diagrams, it develops mapping rules to produce GUIs. Furthermore, in the cases where more than one possible option is available, it recommends the alternative, which optimizes the user experience of the end user. The work in (Margaris et al., 2021) presents a UI for BPEL designers that allows web service personalized recommendation and selection in BPEL scenarios, according to user set criteria. More specifically the presented work gives the BPEL user the ability (a) to preselect the service achieving the highest score, based on the user set criteria, for each recommendation asked, (b) to specify non-qualitative criteria restrictions and (c) to select the number of the candidate (equivalent) services, which they will be depicted for each recommendation asked.

However, none of the abovementioned works addresses the issue of allowing the users to efficiently consider potentially attractive service choices whose QoS attribute values do not fall within the specified bounds and supporting users in this task through a suitable user interface.

The presented UI offers the aforementioned functionality, notifying the user about the presence of potentially attractive service candidates, allowing the user to review these candidates and finally select the most prominent option.

### 3 PREREQUISITES

The following subsections summarize concepts from the areas of (a) service QoS attributes and (b) representation service functionality hierarchies, which are used in this paper, for self-containment purposes. We will also briefly introduce the BPEL scenario adaptation framework, which will be used to exemplify the concepts introduced in this paper.

#### 3.1 Web Services QoS Attributes

In business processes, the non-functional aspects of a service are typically quantified and represented using QoS attributes, such as cost, response time, reliability, availability, etc. (Maximilien and Singh, 2004; Canfora et al., 2005; Raj and Sasipraba, 2010). Without loss of generality, in this paper we consider the attributes of cost (c), reliability (r), and availability (av). The extension of the models and algorithms presented in this paper to accommodate additional QoS attributes is straightforward. Typically, in a BPEL scenario adaptation scheme, the boundaries concerning the maximum and minimum allowed values for the QoS attributes used are

provided as two vectors, denoted as MAX and MIN, respectively. A third vector W has to be provided also, in order to specify the importance/weight of each QoS attribute in the adaptation, effectively driving the prioritization of services according to the values of their QoS attributes. Hence the vector triplet provided along with a BPEL scenario execution request will be the following:

- MAX:  $(min_c, min_r, min_{av})$ ;
- MIN:  $(min_c, min_r, min_{av})$ ;
- W:  $(w_c, w_r, w_{av})$ .

The MAX and MIN boundaries are normalized in the range [0, 10], using a normalization equation, and follow the rule that higher attribute values correspond to higher QoS levels, to avoid confusion (Christos et al., 2009; Dionisis et al., 2013; Margaris et al., 2015b). Furthermore, these boundaries are applied to each service within the BPEL scenario individually, while the W is applied to the whole composition.

Finally, in order to select the optimal service, for each invocation, the BPEL execution engine applies a simple weighted sum approach, considering the QoS attributes values of the overall composition (which are computed based on QoS attributes values of the constituent individual services and the parallel/sequential execution flow structures (Margaris et al., 2015a)), and the weight (vector W) set by the user.

### 3.2 Service Functionality Hierarchies

When the BPEL scenario contains an invocation to some service X providing functionality F, we can invoke a service Y, either if Y has the exact same functionality as X, or if Y provides a more specific functionality than X, analogously to a superclass-subclass relation (Margaris et al., 2020a). This information can be represented using a taxonomy: In this taxonomy, non-leaf nodes represent functionalities, with the most generic ones being placed towards the root and more specific ones towards the leaves. Specific service implementations are placed as leaf nodes, which may accommodate the QoS values of the services, unifying thus the service QoS database and the functionality relationship taxonomy under a single repository. An example taxonomy accommodating functionalities and service implementations is illustrated in Figure 1.

### 3.3 WS-BPEL Scenario Adaptation Framework

The WS-BPEL scenario adaptation framework considered in this paper (Margaris et al., 2015a;

Margaris et al., 2020) allows for the designation of (a) specific services that the user wants to explicitly invoke and (b) services for which a recommendation is requested. This is accomplished using the keywords *INV* and *REC*, respectively, in the BPEL scenario, as depicted in Figure 2.

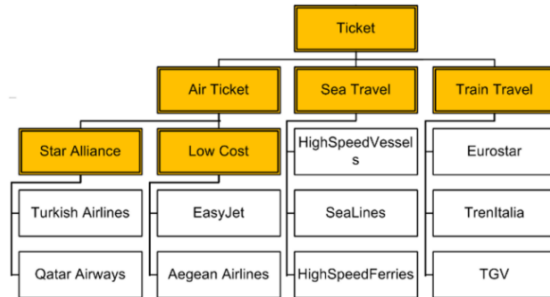


Figure 1: Hierarchy of services implementations (white background) and (sub-)categories (orange background) concerning a travel service.

```

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
    
```

Figure 2: Pseudocode concerning a business trip (air travel, accommodation, and car rental) BPEL execution request.

## 4 UI DESIGN

The UI presented in this paper gives the user the ability to enter the BPEL process specification using a simplified syntax, like the one depicted in Figure 2. This syntax allows for (a) the specification of the services to be invoked, (b) designation of whether services are executed sequentially or in parallel, (c) the use INV and REC notations to distinguish between functionalities where specific service implementations have been pre-chosen by the user and functionalities for which a recommendation is requested, respectively, and (d) the provision adaptation-related information, in the form of the MIN/MAX vectors (per service to be recommended) and QoS parameter weights (globally, at scenario level). For each functionality designated with the INV keyword, the service explicitly listed by the user is invoked, while for each functionality designated with the REC keyword, the services which satisfy the bounds set by the user are retrieved presented, in descending order their score.

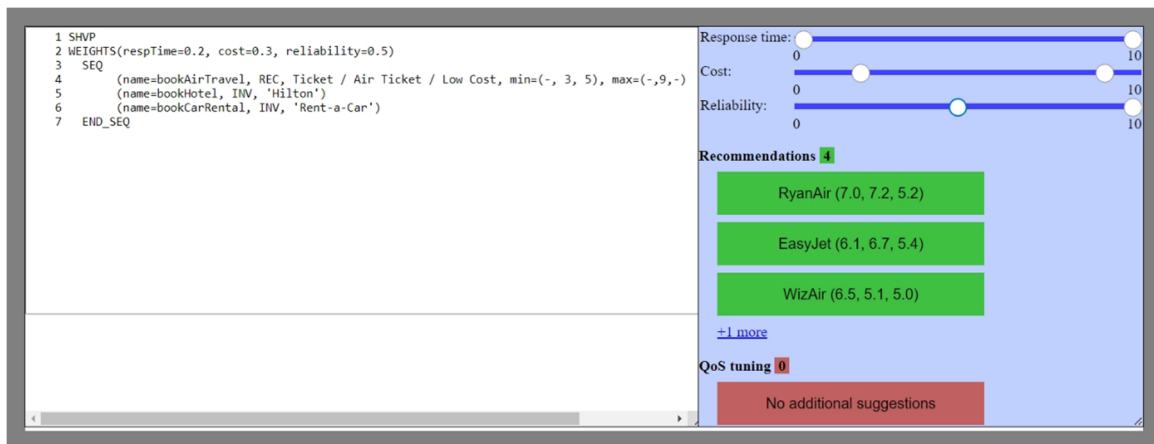


Figure 3: The UI applied on the business trip adaptation scenario.

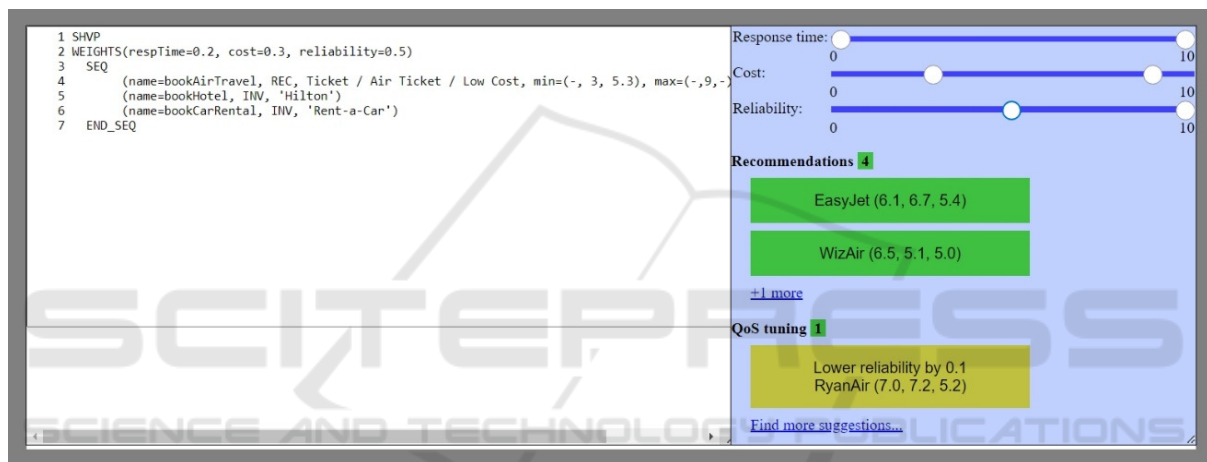


Figure 4: Displaying QoS tuning options.

In Figure 3, we can see an instance of the proposed UI, which depicts the functionality for the adaptation of the business trip scenario introduced in the previous section. At the upper part of the right pane, the services retrieved for the recommendation request (keyword “REC”) in line 4 are listed. These services meet the criteria (bounds) set by the user, and are listed in a highest-to-lowest score order. This score is computed based on their QoS values and the weights set by the user at the beginning of the BPEL scenario (line 2).

Beneath the list of the qualifying services, the UI informs the user regarding additional adaptation possibilities, through the tuning of the QoS parameters. In this illustration, we consider that no services have been rejected due to the application of the user-specified QoS bounds, hence the UI informs the user that no QoS tuning options are applicable.

When the BPEL user makes his selection for the service to be invoked, this is highlighted.

In the case that there exist QoS tuning opportunities, i.e. some services have been found not to fulfill the bounds criteria set by the user, but are deemed to be potentially attractive for the user, the ‘QoS tuning’ area lists these options, effectively informing the user that small modifications to the QoS bounds may result to suggestions that he might be interested in selecting.

In the example shown in Figure 4, we increased the minimum reliability boundary from 5 to 5.3 and, thus, the RyanAir service is rejected. However, Ryanair is only rejected by 0.1/10 (equals to 1%) and its weighted average is calculated at  $7.0 \cdot 0.2 + 7.2 \cdot 0.3 + 5.2 \cdot 0.5 = 6.16/10$ , while the weighted average of the EasyJet service, which is ranked first, is calculated at  $6.1 \cdot 0.2 + 6.7 \cdot 0.3 + 5.5 \cdot 0.5 = 5.98/10$ . As a result, the rejected service has been found to be superior to the optimal one and, hence, the RyanAir service will be included in the ‘QoS tuning’ area, as illustrated in Figure 4.

In this work, we have set the criteria for including a service that has been rejected due to violation of the QoS bounds as follows:

- a) Each QoS attribute value of the service that has been rejected deviates by at most 0.5 from the user-set bounds. For example, if the user sets  $\text{min}_{\text{cost}}=4$  and  $\text{max}_{\text{cost}}=8$ , the “QoS tuning” area will include services having a cost in the range [3.5-8.5] (and, obviously, not appearing in the “Recommendations” area).
- b) The overall score of the service is either superior to the score of the services in the recommendation area, or inferior by a margin of 0.5. The inclusion of services with inferior scores allows the user to slightly modify “on-the-fly” the initially set weights, without needing to change the weight specifications in the left pane. For instance, if the Volotea service with QoS parameters (3, 9.3, 5.2) exists within the service implementation database, with an overall score of 5.98, this might be appealing for the user due to its significantly lower cost, and therefore he might choose to choose it, disregarding the fact that it scores lower than both RyanAir and EasyJet, considering the initially set bounds.

The user is also offered the option to further relax the criteria regarding the area around the bounds within which services are searched, to be included appearing in the “QoS tuning” section.

Each time the “Find more suggestions...” link is clicked, the margin is increased by 0.5. For example, while for the setting  $\text{mincost}=4$  and  $\text{maxcost}=8$ , the “QoS tuning” area would initially include services that have a cost in the [3.5-8.5] range, after the “Find more suggestions...” link is clicked once, the “QoS tuning” area would include services that have a cost in the [3.0-9.0] range. Finally, a recommendation process may produce no results (no service can be found either within the user boundaries to be listed in the “Recommendations” area or fulfilling the criteria to be listed in the “QoS tuning” area), the user is informed and may employ the “Find more suggestions...” link to extend the search range, as described above.

## 5 EXPERIMENTAL EVALUATION

The usability of the UI was assessed via a user study. For the user evaluation, 14 WS-BPEL designers (5 female and 9 male, with a mean age of 41 years old;

all 14 are BSc holders in Computer Science or Engineering) were selected to participate. Each participant had more than 6 months of experience (either academic or commercial) in BPEL design. 12 out of the 14 participants have already used UIs for BPEL scenario adaptation in the past. However, those did not include the extended functionality of tuning QoS parameters.

Initially, the BPEL designers were briefed about the interface, the notation used for the specification of BPEL scenarios and were familiarized with the functionality and the visual elements of the user interface. The duration of the briefing was between 16 and 24 minutes. We asked the participants to use the pseudocode that the presented UI supports (see Figure 2) and develop simple BPEL code that included at least one recommendation. We also gave the participants access to the taxonomy that was already stored. After UI experimentation, the participants were asked to report on their experience. The participants reported their feedback using an online questionnaire that was administered at the end of the experiment. The usability metrics included the user acceptance, confidence, and user satisfaction.

Each participant evaluated each criterion in a Likert scale from 1 to 7 (Albaum, 1997; Allen and Seaman, 2007). Figure 5 depicts the average values of the responses given by the users for these criteria.

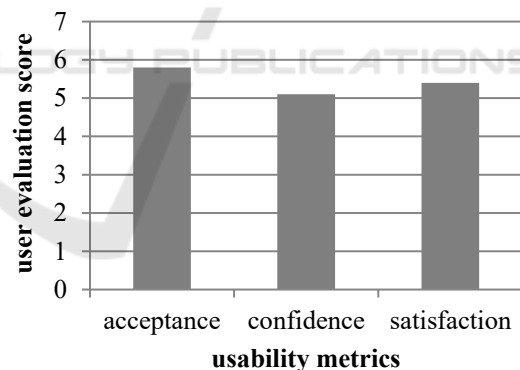


Figure 5: User evaluation results.

According to the evaluation results, shown in Figure 5, we can conclude that the BPEL users were overall satisfied with the UI. Interestingly, the lowest evaluation score (4.0) was found to be given by the two BPEL users that had not used another UI for BPEL scenario adaptation in the past. This is attributed to the fact that these users had limited experience on how the BPEL scenario adaptation is applied and required more time to fully comprehend the adaptation concepts and process, resulting in above average number of trial-and-error attempts.

After the end of the evaluation, an open discussion followed, where most of the participants stated that they would be happy to use this UI both in its current state and by incorporating it into another compatible UI, like the ones in (Margaris et al., 2020b; Margaris et al., 2021).

Regarding further improvement, two suggestions were made by the users. The first suggestion was about the threshold value used to determine whether a service is considered to be of potential interest to the user, which, for the experiments, and was set to  $\pm 0.5$  for every QoS attribute participating in the process. The participants suggested that the user should be allowed to tune this threshold. This will be part of our future work. The second issue concerns the threshold tuning level, where the participants suggest the UI to give the BPEL designer the ability to configure the threshold both globally (to be applied in all REC requests) and per REC request. This suggestion will be also considered for our future work. An initial assessment was that these suggestions could result in a more flexible interface and allow for finer-granularity tuning. A concern will be that of the complexity of the interface that could also suffer, as a result. Therefore, the interplay between these dimensions will have to be assessed, together with the ability of these extensions to contribute to the formulation of solutions of considerably higher overall quality.

## 6 CONCLUSION AND FUTURE WORK

In this paper, we presented a specialised UI that facilitated the WS-BPEL scenario adaptation process, by allowing the users to request recommendations for web service implementations, designating QoS criteria that the recommended implementations must satisfy. The UI provides a dedicated supporting mechanism that enables the users to explore additional service implementations that may not satisfy the initially set criteria but are deemed to be of potential interest to the user. The rationale behind the proposed functionality was that in many real-life cases, a person will accept/purchase a product or a service that exceeds the threshold(s) that were initially set by them, e.g., if the overhead is marginal or the offer is deemed appealing.

We validated the proposed UI by conducting a user study, in which WS-BPEL designers, both with academic and commercial BPEL background, used the UI to create and adapt WS-BPEL scenarios and

subsequently rated different aspects of the UI and recorded their suggestions. This evaluation results showed adequately high level of acceptance, confidence, and satisfaction. Furthermore, the majority of the BPEL designers mentioned that they would happily use this UI, both in its current state and by incorporating it into another compatible IDE.

Our future work will primarily focus on extensions and improvements suggested by the BPEL designers who participated in our experiments. These involve the inclusion of a functionality that allows the BPEL designer to (a) tune the threshold value used to determine whether a service is considered to be of potential interest to the user and (b) configure the threshold both globally (to be applied in all REC requests) and per service recommendation request. Furthermore, we plan to provide the BPEL users the ability to write their BPEL code using graphical tools (e.g. implement suitable UI controls through which users specify sequential and parallel execution branches of the BPEL scenario, loop and conditional execution constructs, and so forth). Finally, we plan to extend the BPEL recommender process, including collaborative filtering techniques between users who share identical or similar functionalities (Zhao et al., 2020, Wu et al., 2022).

## REFERENCES

- Albaum, G. (1997). The Likert scale revisited. *Market Research Society. Journal.*, 39(2), 1-21. SAGE Journals.
- Alf3rez, G. H., Pelechano, V., Mazo, R., Salinesi, C., Diaz, D. (2014). Dynamic adaptation of service compositions with variability models. *Journal of Systems and Software*, 91, 24-47. ELSEVIER.
- Aljukhadar, M., Senecal, S., Daoust, C.-E. (2012). Using Recommendation Agents to Cope with Information Overload. *International Journal of Electronic Commerce*, 17(2), 41-70. Informa UK Limited.
- Allen, I. E., Seaman, C. A. (2007). Likert scales and data analyses. *Quality progress*, 40(7), 64-65.
- Canfora, G., Di Penta, M., Esposito, R., Villani, M. L. (2005). An approach for QoS-aware service composition based on genetic algorithms. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation* (pp. 1069-1075). ACM.
- Chan, K. M., Bishop, J., Steyn, J., Baresi, L., Guinea, S. (2009). A fault taxonomy for web service composition. In *Service-Oriented Computing-ICSOC 2007 Workshops: ICSOC 2007, International Workshops, Vienna, Austria, September 17, 2007, Revised Selected Papers 5* (pp. 363-375). Springer Berlin Heidelberg.
- Christos, K., Vassilakis, C., Rouvas, E., Georgiadis, P. (2009). QoS-driven adaptation of BPEL scenario

- execution. In 2009 IEEE International Conference on Web Services (pp. 271-278). IEEE.
- Deng, C., Yang, H., Liao, H., Sun, M., Qiu, Z. (2011, August). Analysis of ws-bpel processes in prism. In 2011 Fifth International Conference on Theoretical Aspects of Software Engineering (pp. 199-202). IEEE.
- Diaz, E., Panach, J. I., Rueda, S., Vanderdonck, J. (2021) An Empirical Study of Rules for Mapping BPMN Models to Graphical User Interfaces. *Multimedia Tools and Applications*, 80, 9813–9848. Springer Science+Business Media, LLC.
- Dionisis, M., Costas, V., Panagiotis, G. (2013). 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* (pp. 1900-1906). ACM.
- Driss, M., Ben Atitallah, S., Albalawi, A., Boulila, W. (2022). Req-WSComposer: a novel platform for requirements-driven composition of semantic web services. *Journal of Ambient Intelligence and Humanized Computing*, 13, 849–865. Springer-Verlag.
- Fu, X., Bultan, T., Su, J. (2004). Analysis of interacting BPEL web services. In *Proceedings of the 13th international conference on World Wide Web* (pp. 621-630). ACM.
- Karastoyanova, D., & Leymann, F. (2009). BPEL'n'Aspects: Adapting service orchestration logic. In 2009 IEEE International Conference on Web Services (pp. 222-229). IEEE.
- Liu, X., Xu, M., Teng, T., Huang, G., Mei, H. (2016). MUIT: a domain-specific language and its middleware for adaptive mobile web-based user interfaces in WS-BPEL. *IEEE Transactions on Services Computing*, 12(6), 955-969. IEEE.
- Margaris, D., Georgiadis, P., Vassilakis, C. (2013). Adapting WS-BPEL scenario execution using collaborative filtering techniques. In *IEEE 7th International Conference on Research Challenges in Information Science (RCIS)* (pp. 1-11). IEEE.
- Margaris, D., Vassilakis, C., Georgiadis, P. (2015a). An integrated framework for adapting WS-BPEL scenario execution using QoS and collaborative filtering techniques. *Science of Computer Programming*, 98, 707-734. ELSEVIER.
- Margaris, D., Georgiadis, P., Vassilakis, C. (2015b). 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). IEEE.
- Margaris, D., Spiliotopoulos, D., Kardiasmenos, A., Pantazopoulos, D. (2020a). An integer programming-based algorithm for optimising the WS-BPEL scenario execution adaptation process. *International Journal of Web Engineering and Technology*, 15(3), 307-332. Inderscience Enterprises Ltd.
- Margaris, D., Spiliotopoulos, D., Vassilakis, C., Karagiorgos, G. (2020b). A user interface for personalized web service selection in business processes. In *HCI International 2020—Late Breaking Papers: Interaction, Knowledge and Social Media: 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings 22* (pp. 560-573). Springer International Publishing.
- Margaris, D., Spiliotopoulos, D., Vasilopoulos, D., Vassilakis, C. (2021). A user interface for personalising WS-BPEL scenarios. In *International Conference on Human-Computer Interaction* (pp. 399-416). Cham: Springer International Publishing.
- Maximilien, E.M., Singh, M.P. (2004). A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, 8(5), 84-93.
- Moser, O., Rosenberg, F., Dustdar, S. (2008). Non-intrusive monitoring and service adaptation for WS-BPEL. In the 17th international conference on World Wide Web (pp. 815-824). ACM.
- Mukherjee, D., Jalote, P., Gowri Nanda, M. (2008). Determining QoS of WS-BPEL compositions. In *Service-Oriented Computing—ICSOC 2008: 6th International Conference, Sydney, Australia, December 1-5, 2008. Proceedings 6* (pp. 378-393). Springer Berlin Heidelberg.
- Ouyang, C., Dumas, M., Ter Hofstede, A. H., Van der Aalst, W. M. (2006). From BPMN process models to BPEL web services. In 2006 IEEE International Conference on Web Services (ICWS'06) (pp. 285-292). IEEE.
- Pasley, J. (2005). How BPEL and SOA are changing web services development. *IEEE Internet computing*, 9(3), 60-67. IEEE.
- Raj, R.J.R., Sasipraba, T. (2010). Web service selection based on QoS Constraints. In *Trendz in information sciences & computing (tisc2010)* (pp. 156-162). IEEE.
- Sun, C. A., Zhai, Y., Shang, Y., Zhang, Z. (2012). Toward effectively locating integration-level faults in BPEL programs. In 2012 12th International Conference on Quality Software (pp. 17-20). IEEE.
- Tragatschnig, S., Zdun, U. (2011). Runtime process adaptation for bpel process execution engines. In *2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops* (pp. 155-163). IEEE.
- Yongchareon, S., Liu, C., Zhao, X., Yu, J., Ngamakeur, K., Xu, J. (2018). Deriving user interface flow models for artifact-centric business processes. *Computers in Industry*, 96, 66-85. ELSEVIER.
- Wang, Q., Ying, S., Wen, J., Lv, G. (2012). Policy-based exception handling for BPEL processes. In *2012 IEEE International Conference on Information Science and Technology* (pp. 326-331). IEEE.
- Wu, L., He, X., Wang, X., Zhang, K., Wang, M. (2022). A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 35(5), 4425-4445. IEEE.
- Zhao, S., Zhang, Q., Peng, Z., & Lu, X. (2020). Personalized manufacturing service composition recommendation: combining combinatorial optimization and collaborative filtering. *Journal of Combinatorial Optimization*, 40, 733-756. Springer Science+Business Media, LLC.