A User Interface for Personalising WS-BPEL Scenarios

Dionisis Margaris[1], Dimitris Spiliotopoulos[2](✉), Dionysios Vasilopoulos[3], and Costas Vassilakis[3]

[1] Department of Digital Systems, University of the Peloponnese, Sparta, Greece
margaris@uop.gr

[2] Department of Management Science and Technology, University of the Peloponnese, Tripoli, Greece
dspiliot@uop.gr

[3] Department of Informatics and Telecommunications, University of the Peloponnese, Tripoli, Greece
{dvasilop,costas}@uop.gr

Abstract. Due to the huge volume of web services available, both locally and in the cloud, the performance of users and systems need significant research attention. Since WS-BPEL is the dominant language for orchestrating individual web services into business processes, by composing WS-BPEL scripts/scenarios, graphical notations facilitating WS-BPEL design can be extremely useful. Current user interfaces allow WS-BPEL designers not only to invoke selected web services, but also to explicitly ask for recommendations. Then, the user interface appends the services achieving the highest score, according to the attributes' importance, set by the designer, to their WS-BPEL scenario. However, since the final selection is produced automatically, rather than relying on the designers' choices, many times, from a personalisation point of view, the adaptation fails. This work reports on the design, development and user evaluation of a user interface that incorporates functionalities that support the designers' selection performance, thereby upgrading the personalisation level of the WS-BPEL scenarios, as well as the success of the adaptation.

Keywords: User interface · Personalisation · Recommender systems · WS-BPEL scenarios · Web services · User evaluation · Usability

1 Introduction

WS-BPEL (Web Services Business Process Execution Language) is the typical language that orchestrates individual web services (WSs) in order to build high level business processes [1, 2]. A WS-BPEL scenario requires multiple invocations to WSs. Those invocations may be either user-specified, i.e., the user selects the services to invoke and the exact parameters, or system-adapted, i.e. a query is made to a recommender engine to retrieve the WSs that realize the desired functionality and select the one best suited

F. F.-H. Nah and K. Siau (Eds.): HCII 2021, LNCS 12783, pp. 399–416, 2021.
https://doi.org/10.1007/978-3-030-77750-0_25

to the user's needs. The desired functionality may be specified in terms of categories of interest [3, 4], such as Hotel, Sea Travel, and Car Rental services.

The recommender engine queries may impose restrictions on quality of service (QoS) attribute values, according to the user's choices, such as cost, availability, reliability, and others [5, 6]. Existing research has reported algorithms that perform adaptation of WS-BPEL scenario execution based on user specifications of limits and importance of QoS parameters, however, these proposals involve automatic computation and execution of the optimal query plan, without allowing the user to intervene and customize the execution according to his preferences [7, 8]. As a result, in many cases, the invoked service is not the one that a user would select and, as a result, from a personalisation point of view, the WS-BPEL scenario execution adaptation yields suboptimal results. Recent HCI research addresses the problem by offering a user interface (UI) that enables the users to both preview the recommended WSs and make the final selection based on their own preferences [9]. Although that is a major step towards the personalisation of the WS-BPEL scenario execution adaptation, experienced WS-BPEL designers/developers that are the end-users of this approach reported the need for functionalities such as user-directed exclusions and automatic assignment of the top candidate WSs in the scenario code, which no WS-BPEL IDE provides.

This work presents three WS-BPEL scenario adaptation functionalities, which assist the WS-BPEL designers to upgrade the personalisation level of their scenarios and are not supported by any other IDE, as well as the design, implementation and user evaluation of the corresponding UI that supports them.

The rest of the paper is structured as follows: Sect. 2 overviews related work, while Sect. 3 presents the necessary foundations for our work, for self-containment purposes. Sect. 4 presents and analyses the proposed functionalities and the overall UI design, while Sect. 5 presents the results of the user evaluation. Finally, Sect. 6 concludes the paper and outlines the future work.

2 Related Work

Over the last years the research field of WS-BPEL scenarios' adaptation process has attracted significant research efforts [10–13]. Moser et al. [14] propose an aspect-oriented approach by intercepting SOAP messages and allowing service exchange during run-time with minimum performance penalty costs in high-availability BPEL environments. Margaris et al. [15] introduce a framework that incorporates runtime quality of BPEL scenarios WS-based adaptation, allowing for tailoring their execution to the needs of each user. Furthermore, their framework supports automatic resolution of system-level exceptions, while both exception resolution and scenario execution adaptation manage to maintain the transactional semantics that may bear invocations to multiple WSs offered by the same provider.

Kareliotis et al. [16] present a framework that includes mechanisms for considering the qualitative parameters of the invoked WSs, so that the WS-BPEL scenarios tailor their execution to each designer's needs or adapt to the WWW dynamic environment, where old WSs may be withdrawn or change their qualitative parameters, or new ones may be deployed. Furthermore, the proposed framework includes mechanisms that are able to handle infrastructure failures in the distributed WWW environment, as well as allows the designers to specify the qualitative parameters that they require and locate and invoke suitable WSs. Charfi et al. [17] present a plug-in architecture for self-adaptive WS composition with well-modularized self-adaptation features in aspect-based plug-ins. Their approach is easily extensible, supports application-specific adaptation WS-BPEL scenarios and finally allows self-adaptation logic to be deployed on running business process instances. Agarwal and Jalote [18] propose an approach for dynamically adapting WS compositions based on non-functional requirements. Their approach selects the suitable WSs at runtime, while the selected WSs need only be semantically equivalent since their system is able to automatically take care of the syntactical differences between the WSs' interfaces. Margaris et al. [19] introduce a framework which extends the qualitative adaptation mechanisms with collaborative filtering techniques, allowing the WS-BPEL designers to further refine the adaptation process by considering WS selections made by other designers in the past. Wu and Doshi [20] incorporate constraint enforcement models and generalized adaptation in order to transform the traditional WS-BPEL process into an adaptive one, producing a WS-BPEL process able to execute on standard BPEL implementations and to respect coordination constraints.

Liu et al. [21] present a middleware-based approach which deploys mobility, tasks and user interactions into WS-BPEL engines. Their approach provides a Domain-Specific Language which includes facilities which support the declarative development of mobile-oriented adaptive and Web-based UIs in WS-BPEL. Hermosillo et al. [22] introduce a framework which combines the strengths of dynamic business process adaptation and complex event processing that is able to maintain the qualitative characteristics of the business processes by dynamically adapting them according to each case. Hielscher et al. [23] present a framework that aims to enable proactive self-adaptation, by exploiting online testing techniques to detect deviations and process them accordingly before they lead to undesired consequences. Furthermore, they present online testing activities required to trigger proactive adaptation, as well as discuss how these testing activities can be implemented by existing adaptation and testing techniques.

Erradi et al. [24] present a policy-based middleware for dynamic self-adaptation of WS compositions to various changes. Their middleware manages to improve reliability by addressing business exceptions and supporting fault management of WS compositions. Mei et al. [25] exploit the XML-based artifacts' structural similarity between test cases and propose a set of test case prioritisation techniques that selects test case pairs without replacement, in an iteratively way, which proved to be cost-effective in exposing faults. Kareliotis et al. [26] introduce a middleware-based framework for system exception resolution that undertakes the tasks of failure interception and discovery and invocation of alternative services, driven by consumer-specified process qualitative policy. Furthermore, the proposed middleware employs XSLT-based transformations [27–29] to solve syntactic differences between the functionally equivalent WS and the originally invoked one.

However, none of the above-mentioned works addressed the personalisation processes from the user interaction perspective by offering useful functionalities to the WS-BPEL designer, while at the same time upgrading the personalisation level of the WS-BPEL scenario adaptation process.

Recently, Margaris et al. [9] presented a UI for WS-BPEL designers, which produced personalised recommendations, based on user generated criteria, enabling the designers to have the final selection choice. In this work a specialized UI for WS-BPEL designers, that allows personalised recommendation and selection of business process functionalities based on user generated criteria is introduced. Compared to previous works, the proposed UI embeds additional functionalities, such as default values and selections for multiple recommendation candidates and user-directed relational bounds based on non-qualitative criteria, aiming to enhance both the personalisation level of the WS-BPEL scenarios, as well as the adaptation success.

3 Prerequisites

For conciseness purposes, the following subsections summarize the major concepts and underpinnings from the areas of WS substitution relationships and WS QoS attributes domains, which are used in our work.

3.1 Web Services QoS Attributes

The overall performance of WSs is typically described using non-functional parameters expressed as WS QoS attributes, such as reliability, availability, cost, etc. [30–33]. In our work, for conciseness purposes and without loss of generality, we consider the attributes of *reliability* (rel), *cost* (c), and *response time* (rt). For information considering their typical definitions, the interested reader is referred to [34–37].

In a WS-BPEL scenario execution, constraints regarding the upper and lower bounds on each of the QoS attributes delivered by each WS in the context of a specific invocation may be specified using two vectors, which will be denoted as *MIN* and *MAX*, correspondingly. Additionally, a weight vector, which will be denoted as *W*, can be used to indicate the importance of each attribute in the context of the particular invocation of the WS-BPEL scenario [38]:

- $MIN_x\left(min_{rt,x}, min_{c,x}, min_{rel,x}\right)$,
- $MAX_x\left(max_{rt,x}, max_{c,x}, max_{rel,x}\right)$ and
- $W(rt_w, c_w, rel_w)$.

The weight vector is used to calculate an overall score for the whole composition, through the application of a weight sum approach. Effectively, after the value each QoS attribute of the whole composition is determined, each QoS attribute value is multiplied by the respective weight, and the partial sums are added to produce the overall score for the particular WS-BPEL scenario adaptation. Notably, while the MIN and MAX vectors, are applied to each WS selection (regulating thus each individual invocation), while the W vector is applied to the whole WS-BPEL scenario (the whole composition).

Furthermore, in this work:

- All attributes are normalized in the range [0, 10], using a standard normalization (min-max) formula.

- We consider that larger attribute values correspond to higher QoS levels and, hence, an inversion transformation is needed for the attributes where smaller attribute values correspond to higher QoS levels (such as latency and price).

The aforementioned approach is typically followed for QoS attribute values handling in the context of WS selection and composition [2, 39, 40].

Table 1 depicts an example of the repository form that contains four 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
S_4	3	10	4

It has to be noted that our framework is also able to handle (a) service selection affinity (i.e. interdependencies between service selections, in the sense that the selection that some functionality Fj is realized through service implementation SIj,k binds the selection of the service to realize some functionality Fm $\neq$ Fj to some specific implementation SIm,n; the affinity concept preserves the transactional semantics of WS-BPEL scenarios [15]), (b) parallel (concurrent invocations) and sequential structures and (c) exception resolution in WS-BPEL scenarios [15, 26].

3.2 Substitution Relationship Representation

In order for a WS implementation A to realize a particular requested functionality B, in the context of a WS-BPEL scenario, the WS matchmaking relationships [36, 41, 42] are used, where the selected WS must either provide:

- the exact same functionality as B, or
- more specific functionality than B.

In order for an adaptation software to compute the matchmaking relationships between an implementation and a functionality, a WS taxonomy must be used. An example taxonomy is depicted in Fig. 1, concerning an airline ticket booking WS, where the white shaded rectangles (lower levels) represent WS implementations, while the orange shaded rectangles (upper level) represent (sub-)categories [43, 44]. The line between two orange shaded rectangles denotes that the higher-level node includes the lower-level one in a superclass-subclass relation.

The WS repository stores the QoS attributes values for each node corresponding to a WS implementation (white shaded rectangles), as depicted in Fig. 2.

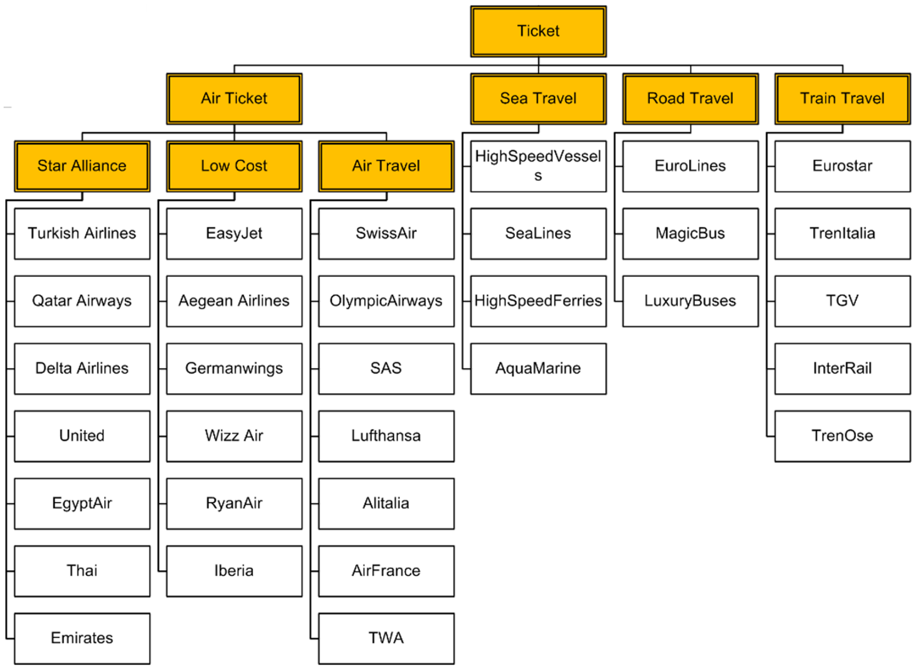
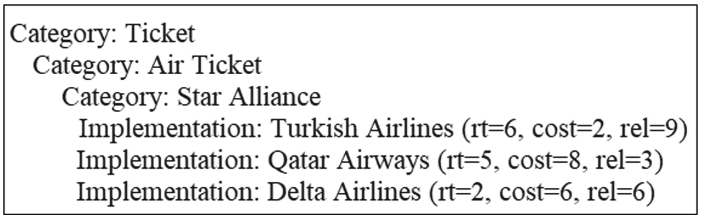


Fig. 1. Hierarchy of WSs (sub-)categories and implementations for the airline ticket booking service.



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

Our WS-BPEL scenario example contains one sequential structure, concerning the process of booking a summer holiday vacation package that includes the following three functionalities:

1. *asking for a recommendation* considering an airline ticket,
2. booking a *specific* (direct invocation to a WS by the user) luxury hotel room, and
3. renting a car from a *specific* (direct invocation to a WS by the user) car rental WS.

The aforementioned scenario's pseudocode is depicted in Fig. 3.

```
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 the summer holiday vacation package business process execution request example.

Each functionality, in Fig. 3, either:

- asks for a WS recommendation (term *REC*): in this case, the user enters the full path to the WS (sub-)category in the repository, as well as the upper and lower QoS attributes' bounds (or enters the "-" symbol, denoting that no specific binding is requested), or
- indicates an invocation to a specific WS (term *INV*).

4 Interface Design and Functionalities

The proposed UI was designed and implemented in order to cover the following three functionalities:

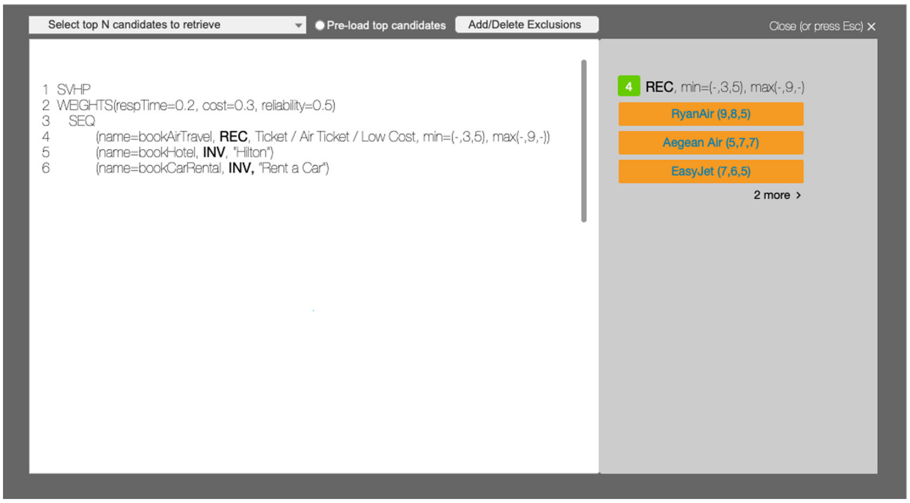
- Preselection of the top candidate service, to allow automatic preloading for all *REC* recommendations, reducing the user effort required to perform manual selection.
- User-directed relational bounds based on non-qualitative criteria, such as "exclude *CountryOrigin* airline" or "exclude *airline_name* = *RyanAir*", thereby satisfying the personalisation user criteria.
- N-result visualization based on user personalisation parameters, such as combination of score/bound metrics and user-specified preference criteria.

In the following subsections the above functionalities will be analysed.

4.1 Preselection of the Top Candidate Service

The UI allows the user to enter the BPEL scenario specification using pseudocode, utilising the *REC* and *INV* commands as illustrated in Fig. 3. For every *REC* placeholder, the services that satisfy the criteria are retrieved and presented to the user as a list ranked by the highest criteria values (Fig. 4). The right-hand side shows the retrieved services for line 4 of the user code that used *REC* for the WS recommendation for specific criteria. It shows the line it refers to, the criteria values for easy user lookup and the retrieved services ordered by highest-to-lowest criteria satisfaction.

The user may adjust the criteria at any time. When so, the list is repopulated with the items and ranking that satisfies the new criteria.



```
SVHP
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")
```

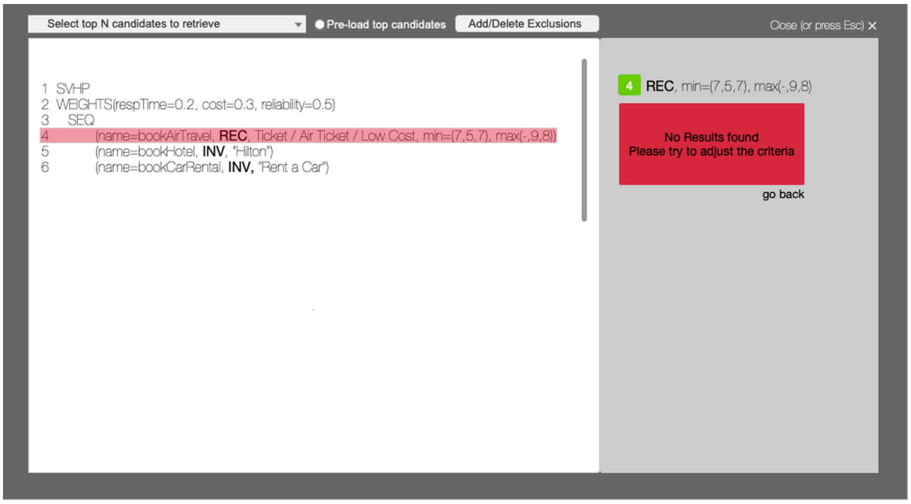
4 REC, min=(-,3,5), max(-,9,-)

RyanAir (9,8,5)

Aegean Air (5,7,7)

EasyJet (7,6,5)

Fig. 4. The UI showing the user code and the WS recommendation request list if retrieved services.



```
SVHP
WEIGHTS(respTime=0.2, cost=0.3, reliability=0.5)
  SEQ
      (name=bookAirTravel, REC, Ticket / Air Ticket / Low Cost, min=(7,5,7), max(-,9,8))
      (name=bookHotel, INV, "Hilton")
      (name=bookCarRental, INV, "Rent a Car")
```

4 REC, min=(7,5,7), max(-,9,8)

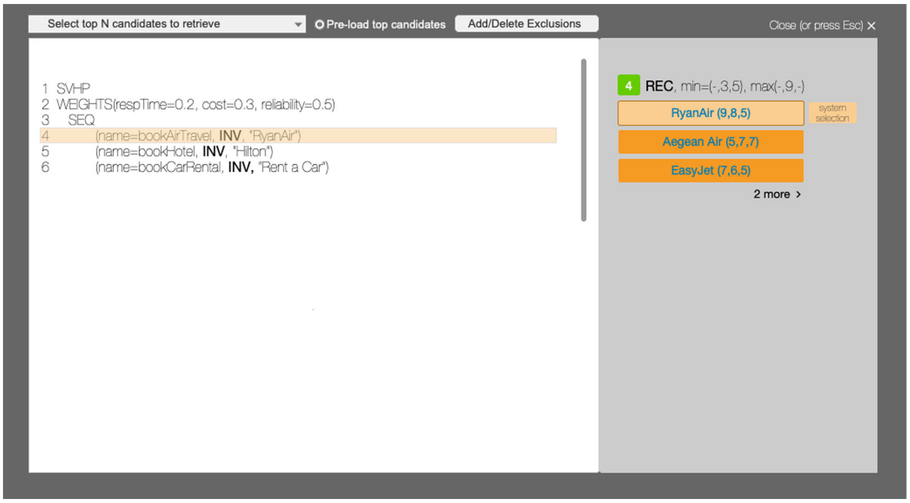
No Results found
Please try to adjust the criteria

Fig. 5. The UI showing the user adjusted criteria and the case where no services that satisfy the criteria can be retrieved.

When no services can be retrieved, the REC line is highlighted in red allowing the user to go back to their previous criteria values or further adjust the criteria that were too strict. The realisation of this functionality is depicted in Fig. 5.

Many BPEL scenarios are characterized by several requests for WSs' recommendations (e.g., over 20). In this case, the BPEL designer has to devote considerable amount of time in order to manually select each specific WSs, so that the BPEL scenario can be executed. To tackle this issue, the proposed system may automatically make selections for the WS implementations that will be used for each *REC* placeholder, selecting for each placeholder the WS implementation that has the highest score, considering the QoS attribute weight vector and the overall composition. Automatic choices are clearly highlighted on the user interface, and the user has the ability to modify the selections.

The user may request the system to pre-load the top candidate services for all *REC* lines using the "Pre-load top candidates" button. When this action is performed, the *REC* code is replaced by specifications of the concrete service invocations of the top candidate preselected by the system, i.e. the top service from the retrieved list. The system preselection is highlighted, both for the lines of code and their selections, as depicted in Fig. 6 (elements coloured in light orange).



```
1  SVHP
2  WEIGHTS(respTime=0.2, cost=0.3, reliability=0.5)
3    SEQ
4         (name=bookAirTravel, INV, "RyanAir")
5         (name=bookHotel, INV, "Hilton")
6         (name=bookCarRental, INV, "Rent a Car")
```

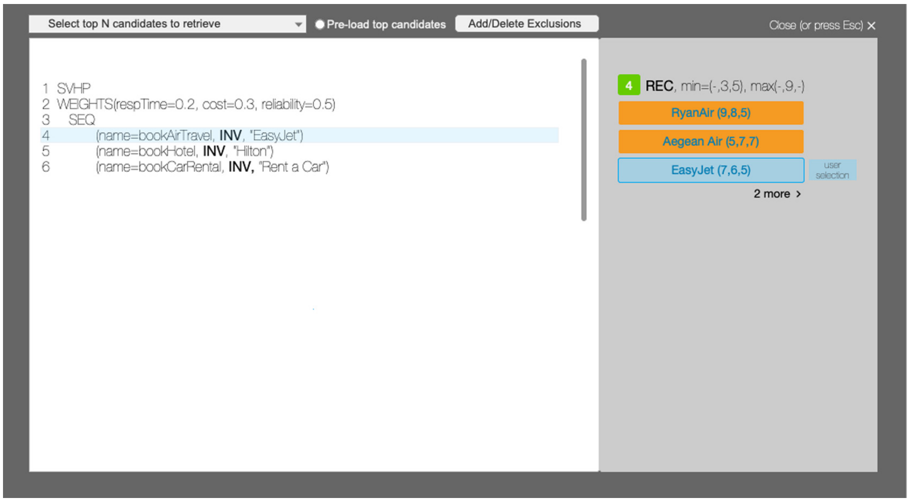
Fig. 6. The system preselection of the top candidate service is pre-loaded and inserted into the user code.

The user may also make a manual selection after the most prominent selections are automatically filled in by the system. In that case, the user's selection replaces the system automatic selection in the code and the highlighting properties are changed to make the fact that a manual selection is applied clear on the UI. The user selection is also shown in blue on the list of the retrieved services (Fig. 7). That is also the case for every user manual selection thereafter.

This functionality allows the user to directly use the system's automatic selections or apply manual selection as many times as necessary, highlighting the current selection and the system preselection. Currently, light orange and blue colours are used for highlighting automatic and manual selections, respectively, making the different cases clearly discernible at a visual level within the UI.

4.2 User-Specified Exclusions

A functionality that would be very useful for WS-BPEL designers, would be the ability to exclude specific or full categories of WSs (e.g., for personal reasons), from being considered in the recommendation process, despite the values in their qualitative attributes,



```
SVHP
WEIGHTS(respTime=0.2, cost=0.3, reliability=0.5)
  SEQ
      (name=bookAirTravel, INV, "EasyJet")
      (name=bookHotel, INV, "Hilton")
      (name=bookCarRental, INV, "Rent a Car")
```

Fig. 7. The manual user selection is inserted into the code, overriding the system pre-loaded code.

based on non-qualitative criteria, such as "exclude *CountryOrigin* airline" or "exclude *airline_name* = *RyanAir*", thereby satisfying the personalisation user criteria.

The aforementioned functionality is available through the user interface and can be accessed using the "Add/Delete Exclusions" UI control, which allows the user to gain access to the WS taxonomy. Through this taxonomy, the user may exclude and re-include services and categories, as well as reset the actions. Figure 8 shows the user exclusion management page.

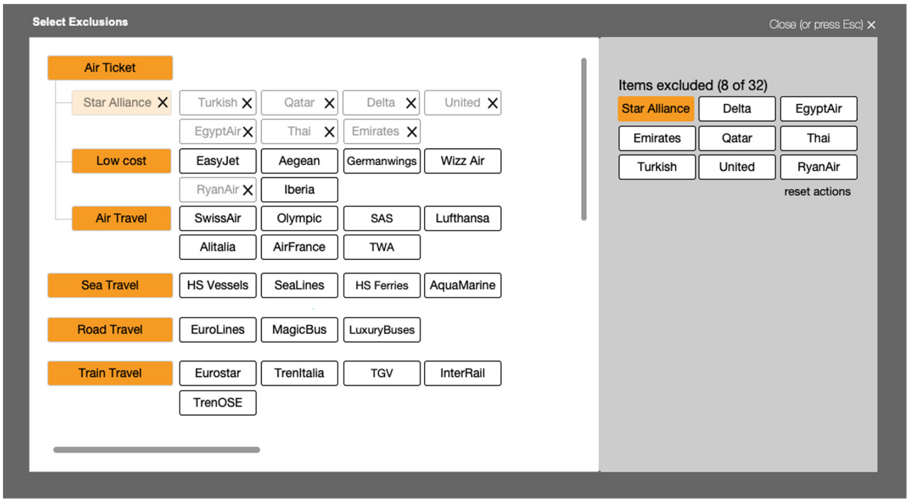
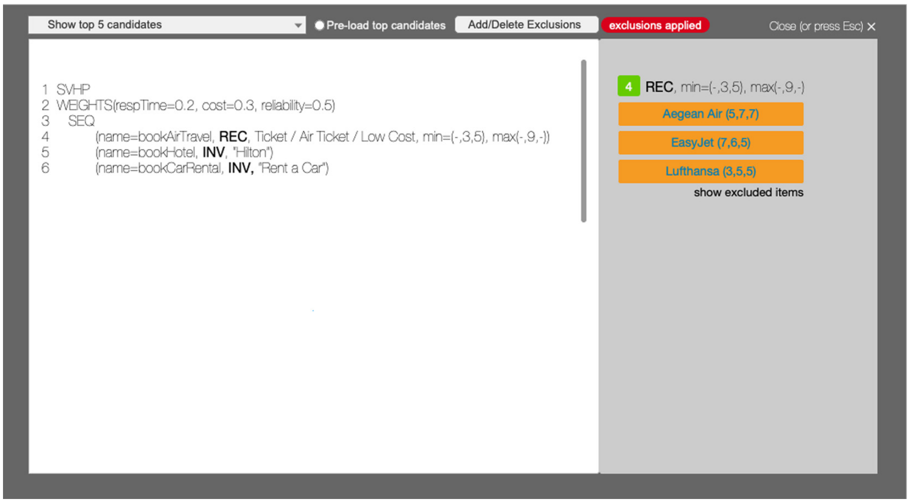


Fig. 8. User-directed exclusions functionality.

The user-specified exclusions lead to the retrieval of filtered results that encompasses only the non-excluded services, as shown in Fig. 9. In this example, the retrieved results exclude all the services under the *Category: "Star Alliance"*, as well as the *airline_name* = "*RyanAir*". The "pre-load top candidate" function will populate the code using the filtered list items. However, the user may choose to view the excluded items and manually select services that were initially excluded based on the user exclusion actions.



```
SVHP
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")
```

4 REC, min=(-,3,5), max(-,9,-)

Aegean Air (5,7,7)
EasyJet (7,6,5)
Lufthansa (3,5,5)

show excluded items

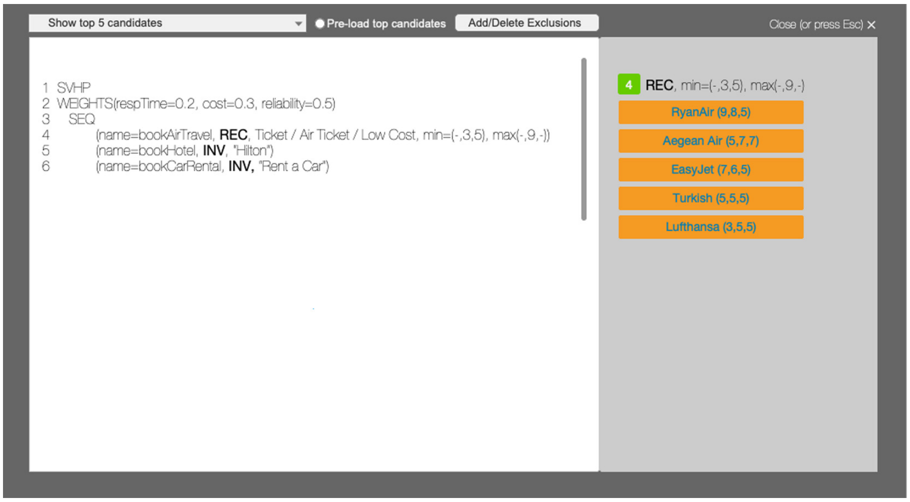
Fig. 9. The WS recommendation results filtered by the user-set exclusions.

4.3 Number of Retrieved Candidates

A very useful feature that WS-BPEL designers have requested is to be able to choose the top-N candidate WSs, based on their feature values and designer parameters (weights and limits), which should appear when requesting for a recommendation, by the same token that some users who perform Google search, stay at the first 4–5 results, while others do a more thorough work, reaching up to the 4th–5th page [45].

This functionality is illustrated in Fig. 10. The optimal value for the top-N parameter, will be experimentally determined in the next section.

The next section presents the evaluation of the UI by WS-BPEL designers, after determining the optimal value for the top-N parameter, concerning the number of candidate WSs that should appear in the right-hand-side list when requesting for a recommendation.



```
SVHP
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")
```

4 REC, min=(-,3,5), max(-,9,-)

- RyanAir (9,8,5)
- Aegean Air (5,7,7)
- EasyJet (7,6,5)
- Turkish (5,5,5)
- Lufthansa (3,5,5)

Fig. 10. The number of service recommendations may vary by design.

5 Experimental Evaluation

In this section, we report on our experiments aiming to:

1. determine the optimal value for parameter *N*, corresponding to the number of candidate WSs that should become available to the user when requesting for a recommendation, and
2. evaluate the usability of the proposed UI, in terms of user satisfaction regarding the offered functionalities [46–50].

5.1 Determining the Number of the Candidates Displayed Per Recommendation

The first experiment is aimed at determining the number of alternative WSs, *N,* that should be initially shown to the user on the UI when a WS-BPEL designer requests for a recommendation, so that, on the one hand designers may have enough options to be displayed to them, while on the other hand this list of WSs is appropriately presented in a manageable, low-cognitive-load fashion. Therefore, in this experiment we vary the number of the candidate WSs considered, seeking the optimal point between "enough alternatives" and "too many alternatives" for a WS-BPEL designer. In this experiment 8 WS-BPEL designers (6 male and 2 female) were used, each with more than 2 years of experience in BPEL design using various commercial or in-house software development platforms [51–53].

In Fig. 11, we can observe that the optimal number of alternative WSs, the designers opted, is 4, marginally outperforming the case of 5 alternatives (9.4/10 versus 9.3/10) and hence, in the subsequent experiments we set parameter *N* to 4. It has to be mentioned that the results shown in Fig. 11 have been found to be fairly independent of the service category within which each designer requested for a recommendation.

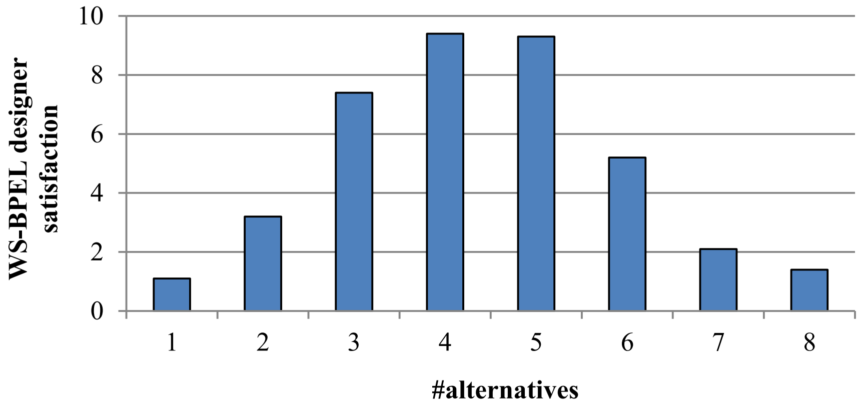


Fig. 11. User feedback on the preference for the number of top N WS recommendations.

5.2 Evaluation

After having determined the optimal number of candidate WSs, *N,* per recommendation to the user, the next experiment is aimed at quantifying the participant satisfaction from the proposed UI, including the three functionalities analysed in the previous section.

In the UI evaluation process, 10 WS-BPEL designers (3 female and 7 male) took part, each with more than 2 years of experience in BPEL design.

The 10 designers were asked to create simple BPEL code (consisting only of WS invocations) using the pseudocode supported by the proposed UI (see Fig. 3) which included at least two recommendation cases. After completing the BPEL design, the participants were asked to report on their user experience, by filling a usability questionnaire, through which satisfaction from the tool, acceptance (i.e. willingness to use the tool) and confidence in tool usage (i.e. self-assessed level of proficiency with the usage of the tool). For each dimension, a score was given in a Likert scale from 0 to10. Figure 12 depicts the user-reported acceptance, confidence and satisfaction. The user evaluation results show that the BPEL designers appreciated the UI features (access to recommendation computation engine, assignment of defaults, customization, exclusion of services or service branches in the taxonomy), commenting that these features provide a good mixture of automation and tailorability.

The main outcome of the follow up discussion was that the vast majority of the designers mentioned they would happily use this IDE in its current state. Two main points for further improvement were reported. Regarding the weights, it was suggested that the *W* values can be adjusted globally as well as individually for specific *RECs.* A similar suggestion was reported for the exclusion functionality, that is to allow for global and individual adjustment. Both suggestions will be considered in the context of our future work.

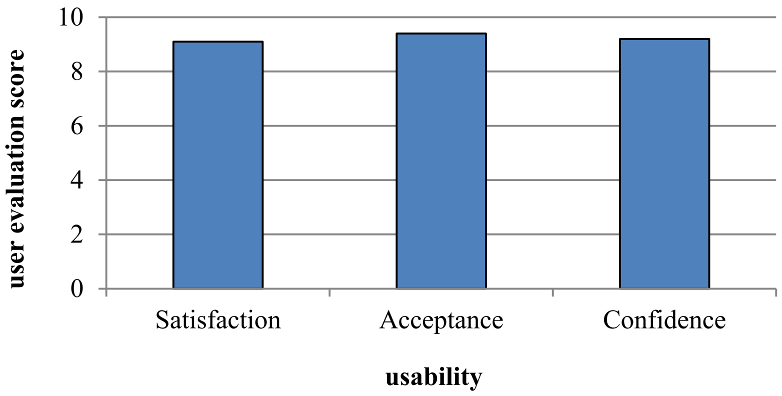


Fig. 12. UI usability evaluation results.

6 Conclusion and Future Work

In this work, a specialised UI for WS-BPEL designers, which allows personalised recommendation and selection of business process functionalities based on user generated criteria, was introduced. Compared to previous work, the proposed UI incorporates three extra functionalities: (a) preselection of the WS with the highest score for all *REC* requests of the BPEL scenario, (b) user-specified restrictions based on non-qualitative criteria, thereby satisfying the personalisation user criteria, and (c) tuning of the number of retrieved candidate service implementations shown to the user, to achieve a balance between (a) increased tailorability through a high number of choices and (b) tackling information overload. Experienced WS-BPEL designers participated in an experiment through which the UI and the recommendation-selection process were evaluated. The evaluation process showed very high levels acceptance, confidence and satisfaction, while the majority of them mentioned that they would happily use this IDE in its current state.

Our future work will focus on considering social media data for search enrichment and personalised ranking of the results by recommending information retrieved from the users' social networks [54–59], as well as collaborative filtering techniques between users sharing similar/identical functionalities [60–63].

References

1. Moser, O., Rosenberg, F., Dustdar, S.: VieDAME - flexible and robust BPEL processes through monitoring and adaptation. In: Companion of the 13th International Conference on Software Engineering - ICSE Companion 2008, p. 917. ACM Press, New York (2008). https://doi.org/10.1145/1370175.1370186
2. Margaris, D., Vassilakis, C., Georgiadis, P.: 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

3. Zhang, H., Gao, Y., Chen, H., Li, Y.: TravelHub: a semantics-based mobile recommender for composite services. In: Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 476–482. IEEE (2012). https://doi.org/10.1109/CSCWD.2012.6221861
4. Chen, X., Liu, X., Huang, Z., Sun, H.: RegionKNN: a scalable hybrid collaborative filtering algorithm for personalized web service recommendation. In: 2010 IEEE International Conference on Web Services, pp. 9–16. IEEE (2010). https://doi.org/10.1109/ICWS.2010.27
5. 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
6. 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
7. Rosenberg, F., Enzi, C., Michlmayr, A., Platzer, C., Dustdar, S.: Integrating quality of service aspects in top-down business process development using WS-CDL and WS-BPEL. In: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), p. 15. IEEE (2007). https://doi.org/10.1109/EDOC.2007.23
8. Sakka Rouis, T., Bhiri, M.T., Kmimech, M., Sliman, L.: A generic approach for the verification of static and dynamic behavioral properties of SCDL/WS-BPEL service-component architectures. In: Park, J.H., Shen, H., Sung, Y., Tian, H. (eds.) PDCAT 2018. CCIS, vol. 931, pp. 381–389. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-5907-1_41
9. Margaris, D., Spiliotopoulos, D., Vassilakis, C., Karagiorgos, G.: A user interface for personalized web service selection in business processes. In: Stephanidis, C., et al. (eds.) HCII 2020. LNCS, vol. 12427, pp. 560–573. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60152-2_41
10. Maâlej, A.J., Krichen, M., Jmaïel, M.: WSCLim: a tool for model-based testing of WS-BPEL compositions under load conditions. In: Gabmeyer, S., Johnsen, E.B. (eds.) TAP 2017. LNCS, vol. 10375, pp. 139–151. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61467-0_9
11. Wang, X., Feng, Z., Huang, K., Tan, W.: An automatic self-adaptation framework for service-based process based on exception handling. Concurr. Comput. Pract. Exp. **29**, e3984 (2017). https://doi.org/10.1002/cpe.3984
12. Margaris, D., Vassilakis, C.: Improving collaborative filtering's rating prediction quality in dense datasets, by pruning old ratings. In: 2017 IEEE Symposium on Computers and Communication, pp. 1168–1174 (2017). https://doi.org/10.1109/ISCC.2017.8024683
13. Ding, Z., Zhou, Z.: RaceTest: harmful data race detection based on testing technology in WS-BPEL. SOCA **13**(2), 141–154 (2019). https://doi.org/10.1007/s11761-019-00261-1
14. Moser, O., Rosenberg, F., Dustdar, S.: Non-intrusive monitoring and service adaptation for WS-BPEL. In: Proceeding of the 17th international conference on World Wide Web - WWW 2008, p. 815. ACM Press, New York (2008). https://doi.org/10.1145/1367497.1367607
15. Margaris, D., Vassilakis, C., Georgiadis, P.: 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
16. Kareliotis, C., Vassilakis, C., Rouvas, E., Georgiadis, P.: QoS-driven adaptation of BPEL scenario execution. In: 2009 IEEE International Conference on Web Services, pp. 271–278. IEEE (2009). https://doi.org/10.1109/ICWS.2009.80
17. Charfi, A., Dinkelaker, T., Mezini, M.: A plug-in architecture for self-adaptive web service compositions. In: 2009 IEEE International Conference on Web Services, pp. 35–42. IEEE (2009). https://doi.org/10.1109/ICWS.2009.125

18. Agarwal, V., Jalote, P.: From specification to adaptation: an integrated QoS-driven approach for dynamic adaptation of web service compositions. In: 2010 IEEE International Conference on Web Services, pp. 275–282. IEEE (2010). https://doi.org/10.1109/ICWS.2010.39
19. Margaris, 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
20. Wu, Y., Doshi, P.: Making BPEL flexible &#150; Adapting in the context of coordination constraints using WS-BPEL. In: 2008 IEEE International Conference on Services Computing, pp. 423–430. IEEE (2008). https://doi.org/10.1109/SCC.2008.71
21. Liu, X., Xu, M., Teng, T., Huang, G., Mei, H.: MUIT: a domain-specific language and its middleware for adaptive mobile web-based user interfaces in WS-BPEL. IEEE Trans. Serv. Comput. **12**, 955–969 (2019). https://doi.org/10.1109/TSC.2016.2633535
22. Hermosillo, G., Seinturier, L., Duchien, L.: Using complex event processing for dynamic business process adaptation. In: 2010 IEEE International Conference on Services Computing, pp. 466–473. IEEE (2010). https://doi.org/10.1109/SCC.2010.48
23. Hielscher, J., Kazhamiakin, R., Metzger, A., Pistore, M.: A Framework for proactive self-adaptation of service-based applications based on online testing. In: Mähönen, P., Pohl, K., Priol, T. (eds.) ServiceWave 2008. LNCS, vol. 5377, pp. 122–133. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89897-9_11
24. Erradi, A., Maheshwari, P., Tosic, V.: Policy-driven middleware for self-adaptation of web services compositions. In: van Steen, M., Henning, M. (eds.) Middleware 2006. LNCS, vol. 4290, pp. 62–80. Springer, Heidelberg (2006). https://doi.org/10.1007/11925071_4
25. Mei, L., Cai, Y., Jia, C., Jiang, B., Chan, W.K.: Prioritizing structurally complex test pairs for validating WS-BPEL evolutions. In: 2013 IEEE 20th International Conference on Web Services, pp. 147–154. IEEE (2013). https://doi.org/10.1109/ICWS.2013.29
26. Kareliotis, C., Vassilakis, C., Rouvas, E., Georgiadis, P.: IQoS-aware exception resolution for BPEL processes: a middleware-based framework and performance evaluation. Int. J. Web Grid Serv. **5**, 284 (2009). https://doi.org/10.1504/IJWGS.2009.028346
27. Gu, G.P., Petriu, D.C.: XSLT transformation from UML models to LQN performance models. In: Proceedings of the Third International Workshop on Software and Performance - WOSP 2002, p. 227. ACM Press, New York (2002). https://doi.org/10.1145/584369.584402
28. Janssen, W., Korlyukov, A., Van den Bussche, J.: On the tree-transformation power of XSLT. Acta Inform. **43**, 371–393 (2007). https://doi.org/10.1007/s00236-006-0026-8
29. Spiliotopoulos, D., Xydas, G., Kouroupetroglou, G.: Diction based prosody modeling in table-to-speech synthesis. In: Matoušek, V., Mautner, P., Pavelka, T. (eds.) TSD 2005. LNCS (LNAI), vol. 3658, pp. 294–301. Springer, Heidelberg (2005). https://doi.org/10.1007/11551874_38
30. Kalepu, S., Krishnaswamy, S., Loke, S.W.: Verity: a QoS metric for selecting web services and providers. In:2003 Proceedings of Fourth International Conference on Web Information Systems Engineering Workshops, pp. 131–139. IEEE. https://doi.org/10.1109/WISEW.2003.1286795
31. 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
32. Alrifai, M., Skoutas, D., Risse, T.: Selecting skyline services for QoS-based web service composition. In: Proceedings of the 19th International Conference on World Wide Web - WWW 2010, p. 11. ACM Press, New York (2010). https://doi.org/10.1145/1772690.1772693

33. Kobusinska, A., Boron, M., Kerebinska, A., Margaris, D.: Exploiting recommender service to enhance efficiency of replication. In: 2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA), pp. 64–72. IEEE (2019). https://doi.org/10.1109/SOCA.2019.00017
34. 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
35. Tran, V.X., Tsuji, H.: QoS based ranking for web services: fuzzy approaches. In: 2008 4th International Conference on Next Generation Web Services Practices, pp. 77–82. IEEE (2008). https://doi.org/10.1109/NWeSP.2008.41
36. 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
37. Varitimiadis, S., Kotis, K., Spiliotopoulos, D., Vassilakis, C., Margaris, D.: "Talking" triples to museum chatbots. In: Rauterberg, M. (ed.) HCII 2020. LNCS, vol. 12215, pp. 281–299. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50267-6_22
38. 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
39. Yu, T., Zhang, Y., Lin, K.-J.: Efficient algorithms for Web services selection with end-to-end QoS constraints. ACM Trans. Web. **1**, 6 (2007). https://doi.org/10.1145/1232722.1232728
40. 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
41. 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
42. Pal, K.: A semantic web service architecture for supply chain management. Proc. Comput. Sci. **109**, 999–1004 (2017). https://doi.org/10.1016/j.procs.2017.05.442
43. Qu, C., Calheiros, R.N., Buyya, R.: Auto-scaling web applications in clouds. ACM Comput. Surv. **51**, 1–33 (2018). https://doi.org/10.1145/3148149
44. Mezni, H., Fayala, M.: Time-aware service recommendation: taxonomy, review, and challenges. Softw. Pract. Exp. (2018). https://doi.org/10.1002/spe.2605
45. Beel, J., Gipp, B.: Google Scholar's ranking algorithm: the impact of citation counts (an empirical study). In: Third International Conference on Research Challenges in Information Science. IEEE, Fez (2009). https://doi.org/10.1109/RCIS.2009.5089308
46. Kouroupetroglou, 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
47. Spiliotopoulos, D., Stavropoulou, P., Kouroupetroglou, G.: Spoken dialogue interfaces: integrating usability. In: Holzinger, A., Miesenberger, K. (eds.) USAB 2009. LNCS, vol. 5889, pp. 484–499. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10308-7_36
48. Spiliotopoulos, D., Tzoannos, E., Stavropoulou, P., Kouroupetroglou, G., Pino, A.: Designing user interfaces for social media driven digital preservation and information retrieval. In: Miesenberger, K., Karshmer, A., Penaz, P., Zagler, W. (eds.) ICCHP 2012. LNCS, vol. 7382, pp. 581–584. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31522-0_87
49. Korableva, O., Durand, T., Kalimullina, O., Stepanova, I.: Studying user satisfaction with the MOOC platform interfaces using the example of coursera and open education platforms. In: Proceedings of the 2019 International Conference on Big Data and Education – ICBDE 2019, pp. 26–30. ACM Press, New York (2019). https://doi.org/10.1145/3322134.3322139

50. Spiliotopoulos, D., Xydas, G., Kouroupetroglou, G., Argyropoulos, V., Ikospentaki, K.: Auditory universal accessibility of data tables using naturally derived prosody specification. Univers. Access Inf. Soc. **9**, 169–183 (2010). https://doi.org/10.1007/s10209-009-0165-0
51. Adadi, N., Berrada, M., Chenouni, D., Halim, M.: AWSCPM: a framework for automation of web services composition processes. In: 2019 7th Mediterranean Congress of Telecommunications (CMT), pp. 1–4. IEEE (2019). https://doi.org/10.1109/CMT.2019.8931389
52. Anvari, M., Takht, M.D., Sefid-Dashti, B.: Thrift service composition. In: Proceedings of the International Conference on Smart Cities and Internet of Things - SCIOT 2018, pp. 1–5. ACM Press, New York (2018). https://doi.org/10.1145/3269961.3269973.
53. 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
54. Margaris, 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
55. Margaris, 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
56. 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
57. Risse, T., et al.: The ARCOMEM architecture for social- and semantic-driven web archiving. Future Internet **6**, 688–716 (2014). https://doi.org/10.3390/fi6040688
58. Petasis, G., Spiliotopoulos, D., Tsirakis, N., Tsantilas, P.: Sentiment analysis for reputation management: mining the Greek web. In: Likas, A., Blekas, K., Kalles, D. (eds.) SETN 2014. LNCS (LNAI), vol. 8445, pp. 327–340. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07064-3_26
59. Campana, M.G., Delmastro, F.: Recommender systems for online and mobile social networks: a survey. Online Soc. Netw. Media **3–4**, 75–97 (2017). https://doi.org/10.1016/j.osnem.2017.10.005
60. Margaris, D., Spiliotopoulos, D., Vassilakis, C.: Social relations versus near neighbours: reliable recommenders in limited information social network collaborative filtering for online advertising. In: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2019), pp. 1160–1167. ACM, Vancouver (2019). https://doi.org/10.1145/3341161.3345620
61. Margaris, 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
62. Nilashi, M., Ibrahim, O., Bagherifard, K.: A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. Expert Syst. Appl. **92**, 507–520 (2018). https://doi.org/10.1016/j.eswa.2017.09.058
63. Raghuwanshi, S.K., Pateriya, R.K.: Collaborative filtering techniques in recommendation systems. In: Shukla, R.K., Agrawal, J., Sharma, S., Singh Tomer, G. (eds.) Data, Engineering and Applications, pp. 11–21. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-6347-4_2