Personalized Active Service Spaces for End-User Service Composition

Jun Han[1], Yanbo Han[2], Yan Jin[1], Jianwu Wang[2], Jian Yu[2]
[1]*Faculty of ICT, Swinburne University of Technology, Hawthorn, VIC 3122, Australia*
{jhan, yjin}@ict.swin.edu.au
[2]*Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China*
{yhan,wjw,yujian}@software.ict.ac.cn

Abstract

End-user service composition is a promising way to ensure flexible, quick and personalized information provision and utilization, and consequently to better cope with spontaneous business requirements. For end-users to compose services directly, issues like service granularity, service organization and business-level semantics are critical. End-users will certainly be at loss if they have to select from a long list of available Web services expressed in IT jargons. This article introduces the concept of personalized active service spaces *and focuses on the use of* business services, service dependency rules, *and* service personalization rules *to support end-user service composition. It addresses two key issues in end-user composition: how to utilize the user preference and context to restrict the scope of applicable services for selection, and how to capture and utilize dependencies or usage patterns between services in order to provide guidance and enforce temporal/sequential restrictions on service invocations for end-user service compositions.*

1. Introduction

Real-world business scenarios require that virtual organizations of individual applications be set up or re-configured in a just-in-time manner to meet specific business needs. Such examples include dynamic supply chain management, handling of city emergencies, and information mediation of public mass events like the Olympic Games [1, 2]. The traditional approach of "programming by IT professionals" is not suitable for such situations as it takes too long or is not flexible enough. This is better achieved by end-users as they are the most directly involved, but needs greater automated support. For example, Oinn *et al.* illustrate the necessity for biologists to compose Web services [3] and Akkiraju *et al.* argue that business consultants' involvement in service composition can ease the complex work of business process integration [4]. At present, end-user involvement in service composition is not yet given much consideration. Current Web service composition languages such as BPEL4WS and BPML are developed for IT professionals and are still weak in dealing with a spectrum of application scenarios that require Web services be quickly composed and reconfigured by non-IT professionals in order to cope with the spontaneity and volatility of user requirements. To enable end-user-programmable Web service composition, we have to cross a number of hurdles, e.g. how to derive user-understandable, business-level services from the relevant Web services, how to mange the complexity and control the scope of visibility by filtering and grouping suitable business-level services according to individual user requirements, how to enable end-users to assemble business-level services in an appropriate way, and how to ensure the interoperability and QoS constraints of participating business-level services.

In meeting the above challenges, we have proposed an approach to end-user-programmable, business-level service composition, and defined a corresponding composition language VINCA [1, 6]. VINCA includes the key concepts of *business services*, *Web service virtualization* (supporting business services), and *service spaces* (supporting business services composition). A business service is a user-understandable, large-granularity service abstraction with business-level semantics. Web service virtualization is the process of abstracting away the Web service's technical details, describing it with business-level semantics, and then registering it to semantically matchable business services. After this process, Web services are not longer seen and used directly, rather they delegate their capabilities to business services. To facilitate service searching and selection, business services are managed by a service registry (we call it the *service space*). One major function of the service space is to provide a business domain based classification of business services to end-users.

The service composition task is made much easier for end-users within the VINCA framework. To bring VINCA to its full potentials in supporting end-user service composition, two important issues need to be addressed. First, in the process of service composition, the end-user programmer may confront with a large number of business services offered by various service providers. Take the Olympic Games as an example. Services may be from game organizers, meteorology bureaus, travel agencies, public transport, etc. It can become rather difficult and time-consuming for the end-user to find the right business services. To alleviate this problem, service classification, taxonomies, and service semantics can be used to help in service discovery. On the other hand, one can largely reduce the number of services offered to the user for composition by taking into account the user's personal circumstance, e.g. currently focused business, current geographical location, etc. This requires an automatic and somewhat intelligent approach to identify a reasonably sized set of business services that are of interest to the user at each point of time of composition.

Second, it is often the case that some business services can only be used in a certain order, as prescribed or implicitly assumed in domain knowledge. For example, goods is always delivered after a purchase. The end-user may not be aware of such service dependency rules if she is not an expert or have no prior knowledge of that domain. Consequently, the composed application may not be compatible with the business logics implemented in the underlying Web services. Their execution may bring unexpected results or even cause financial or social disadvantages to the user. To avoid this, mechanisms to capture service dependency and check automatically their conformance in service compositions are needed.

To address the above-mentioned issues we introduce in this paper the concept of *Personalized Active Service Spaces* (PASS). PASS extends the common service space with *service personalization rules* and *service dependency rules*. With service personalization rules, a specific end-user's preference and context are utilized to filter the potentially huge set of available business services in the service space. The aim is to present to the end-user only business services that are of interest or related to her for composition, and reduce the set of services presented at each point of time of the composition to an end-user-manageable scale. With service dependency rules, conformance checking can be done to prevent incorrect service composition, and recommendations for next-steps in the service composition can be made in a proactive way.

This paper is structured as follows. In the next section, a motivating scenario is presented to highlight the problems PASS is intended to address. Section 3 outlines the concepts of business services and service spaces. PASS is then presented in section 4. This includes service personalization rules and service dependency rules. Section 5 revisits the motivating scenario and demonstrates how PASS effectively addresses the issues identified above for end-user service composition. Finally, we discuss the related work in section 6 and conclude the paper in section 7.

2. Motivating Scenario

Let us use a simplified scenario from the FLAME2008 project [1, 4]. FLAME2008 is aimed at facilitating better information supply to the general public during the 2008 Beijing Olympic Games. The targeted users include visitors, reporters, organizers, and local residents. The local government encourages various parties, including government agencies and contracted service providers, to provide a variety of Web services on the Internet for the users. VINCA has been designed within the project to help the different groups of end-users to effectively use and assemble these services to build their own personalized "applications".

Consider that a Sunshine Post journalist John Bull visits Beijing during the Olympic Games as both a sport reporter and tourist. He has diverse personal interests, including Chinese history and culture, Chinese food and a number of specific competition sports. He would like to get a message notification on his mobile phone when he passes by a historic site. He speaks French, English and German, but not Chinese. He would like to read menus in the languages he knows. This wish list can be long. To provide a service composition platform for end-users like John, the available services should be presented in the right granularity and format that are understandable, and well organized for ease of use.

The major issues of concern in this paper include the following: Upon arriving in Beijing as a journalist and later as a tourist, how can John's service space be formed and then evolve dynamically according to his ever-changing personalization information like his current focused business, his preference, and his spatial and temporal coordinates? For example, if John plans to do the journalist's business, he may want services in the space are game-speculating related, e.g. "*Get Game-Schedule*", "*Book Game-Ticket*", "*Deliver Game-Ticket*", etc. Also, if John plans to travel, he may want services in space are travel related, e.g. "*Book Flight*", "*Reserve Hotel*", "*Rent Car*", etc. If John chooses a service for composition, how can the system recommend related services for his convenience and for guiding his composition? For example, if he chooses a "*Reserve Hotel*" service, the most commonly related services such as "*Book Flight*", "*Rent Car*", "*Check Route*", etc. should be prompted as his preceding or next step. Assume that John chooses a service that depends on another service not yet be selected, how does the system warn or even prevent him from doing so? For example, if John chooses the service "*Deliver Game-Ticket*" without choosing "*Book Game-Ticket*" beforehand, he should get a warning at least.

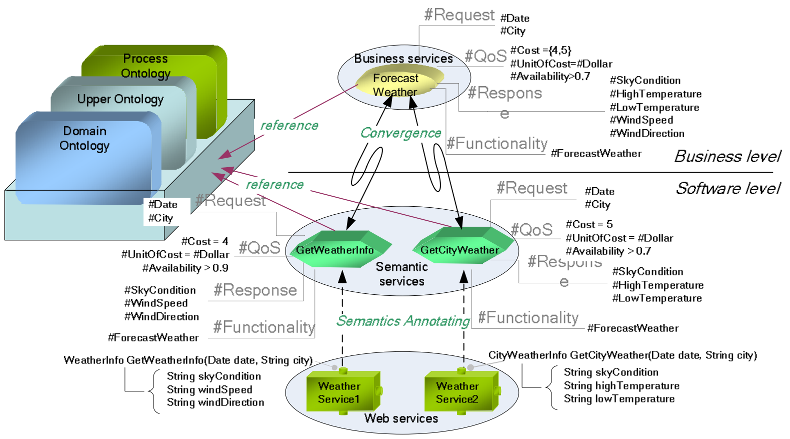

Figure 1. Convergence between business services and web services.

To sum up, in order for an end-user, e.g. John, to compose services effectively, the service space should be *personalized* and *active*. "*Personalized*" means that the formation of the space should consider John's preference and context. "*Active*" means that the space has the ability to recommend appropriate services to John and also prevent his illogical composition behaviour in a proactive manner.

3. Business Services and the Service Space

To facilitate end-user service composition, we have developed the VINCA framework in [7, 8] and introduced the concept of business services as the entities for service composition. Business services are defined by domain experts to represent an abstract functionality of web services. They can abstract away implementation details of Web services, and thus be described in business terms that are easy to understand by end-users. Their definition is based on domain-specific norms of concepts and functionalities. A business service is formulated based on meaningful combinations of business activities (verbs) and business concepts (nouns) as defined in domain standards, e.g. the OTA travel information standard [9] and HL7 healthcare information standard [10]. We have employed the Process Ontology of the MIT Process Handbook Project [11] to obtain business activities (verbs), and SUMO upper ontology [12] and several domain ontologies [13] to obtain business concepts (nouns). For instance, from the OTA message specifications, we know that "book hotel" is a typical business function. Then we can choose the verb "book" from the Process Ontology and the noun "hotel" from the travel domain ontology to identify a new business service as "Book Hotel".

After the business services are defined by domain experts, the underlying Web services are grouped and associated with business services through a virtualization mechanism. As shown in Figure 1, functional and non-functional information with clear semantics is added to the Web services, forming *semantic services*. Convergent relationships are then established through semantic matching between business services and semantic services, acting as glue between the user-understandable business services and executable Web services [4, 7].

To facilitate the searching and selection of business services, a common repository/registry, called the *service space*, is built to hold business services. There, a business service is identified uniquely by its name. The service space may include metadata for services and facilities like classification system, business owner information, and business rules for more discovery and management capabilities.

4. Personalized Active Service Space

Introducing the concept of business services can largely relieve the burden of end-users in service composition. However, an end-user's attention is not always restricted to a particular domain in a composition exercise. A service space made up of business services from multiple domains may become very large and unmanageable for a human. As such, it can be rather difficult for an end-user to search, identify and compose relevant services to carry out a particular task. Furthermore, for a given business domain, there are usually patterns that govern the logical dependencies between the relevant services, e.g. boundedness and sequential ordering. Service compositions that violate these patterns may be against the accepted practice in the domain and considered as invalid.

To address these problems, in this section, we present a concept called *Personalized Active Service Space* (PASS). It improves upon the common business service space with two distinctive features: service personalization rules and service dependency rules. Service personalization rules are used to obtain a user-

and situation-specific service space from the common service space by taking into account the end-user's preference and context information. Further, service dependency rules are used to capture service dependency patterns. When both dependency rules and personalization rules are predefined by domain experts and applied to the common service space, a *Personalized Active Service Space* is formed. In rest of this section, we will discuss in turn the principles of service dependency rules and service personalization rules. We will also illustrate the formation of PASS using Mr. John Bull as an example.

4.1. Service Dependency Rules

In a business domain, services are often interrelated with each other according to some patterns, reflecting the domain's characteristics. These dependency relationships can be used to help end-users to improve efficiency and ensure correctness in effectively utilizing and composing business services.

To capture service dependency, we make use of the system property specification patterns developed by Dwyer *et al.* in [14]. Similar to system properties, service dependency may restrict the occurrences of individual services and the order (or sequencing) of occurrences between different services. Figure 2 shows the pattern and scope operators we support. Restrictions on service occurrences include absence, existence, bounded existence, etc [14]. For example, a bounded existence dependency "*bs* **exists at most 1 times**" indicates that a business service "*bs*" can occur at most once. The order of occurrences between different services includes precedence, response, and so on [14]. For example, given two business services bs_1 and bs_2, a precedence dependency "bs_1 **precedes** bs_2" states that bs_1 must occur at least once before any occurrence of bs_2. One may think bs_1 enables bs_2. In contrast, a response dependency "bs_1 **leads to** bs_2" states that an occurrence of bs_1 must be followed by an occurrence of bs_2. Essentially, this specifies a cause-effect relationship between bs_1 and bs_2. In general, patterns are coupled with scopes to address complex scenarios [14]. Each scope defines a portion of a service workflow path where the service dependency stated by the pattern must hold. For example, "bs_1 **is absent after** bs_2 **until** bs_3" basically states that an occurrence of bs_2 disables bs_1 until bs_3 occurs. The readers are referred to [15], [16] for more detailed descriptions of patterns and scopes, their finite state automata based semantics as well as means to associate conditions on service input and output parameters.

$$\begin{array}{l} bs_1 \text{ is absent} \\ bs_1 \text{ exists } [\begin{bmatrix}\text{at most}\\ \text{at least}\end{bmatrix} n \text{ times}] \\ bs_1 \text{ precedes } bs_2 \\ bs_1 \text{ leads to } bs_2 \end{array} \times \begin{array}{l} \text{globally} \\ \text{before } bs_3 \\ \text{after } bs_3 \\ \text{after } bs_3 \text{ until } bs_4 \\ \text{between } bs_3 \text{ and } bs_4 \end{array}$$

Figure 2. Dependency patterns

Utilizing patterns for specifying service dependencies has a major advantage over other approaches based on formal specification languages, such as process algebra, Petri nets, description logics, etc. While being formal, this approach is also intuitive and easy to use by domain experts who are usually unfamiliar with formal methods. On the other hand, compared to industrial process description languages such as UML activity diagrams, BPEL4WS, BPML ebXML BPSS, this approach is amenable to automated formal analysis.

Service dependency rules can be defined by domain experts concerning specific services in their domain. For example, an expert or service provider in the tourism domain may state that "*Book Day-Tour*" **precedes** "*Check Route*" **globally**. This rule will be applicable to all end-user compositions involving the two services. In general, not all service dependency rules need to be defined up-front. They can be added incrementally as they are identified. All the specified rules will conjunctively determine the constraints on the composition. Services that are not related directly or conjunctively by the specified dependency rules can appear in any order in a service composition.

Note that dependency rules are stated relative to a particular domain as it is defined at a given time. When the domain evolves, eg, by adding new activities, certain rules may become invalid and need revision. To accommodate this, we allow a dependency rule to be *prescriptive* or *suggestive*. A prescriptive rule dictates that a service composition must obey. An example may be *"Book Game-Ticket"* **leads to** *"Deliver Game-Ticket"* **globally**. A suggestive rule states a recommended practice but the end-user may choose to ignore the raised violation warnings. An example may be *"Book Day-Tour"* **precedes** "*Check Route*" **globally**.

4.2 Service Personalization Rules

Service personalization rules are intended to filter out services that are not currently of interest to an end-user. A user's interest to a service is derived from her personalization information such as *preference* and *context*. Preference reflects the user's interests and habits. For example, John likes Chinese history and culture and is accustomed to read and speak in French. Context is any information that is relevant to the

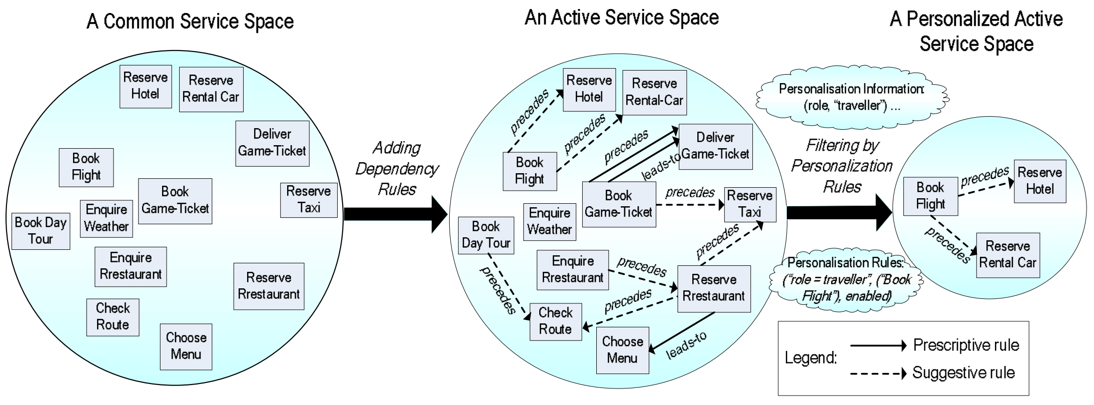

Figure 3. From common service space to PASS

interactions between a user and an environment [17], for example, the current geographical location of John's. Generally, personalization information can be expressed with a *key-value* mapping where *key* is a unique attribute name of the user. For example, the tuple *(role, "tourist")* means that John's current role is a tourist. Note that the *value* can also be a set, for example, the tuple *(role, ("tourist", "game spectator"))* means that John is acting as a dual role: He is both a traveller and game spectator.

A personalization rule is expressed in a 3-tuple: (Condition, ServiceSet, Status), where *Condition* is a logical expression, *ServiceSet* is a set of business services, and *Status* can be either "enabled" or "disabled" to indicate the applicability of the rule. For example, rule ("role = traveller", ("*Book Flight*", "*Reserve Hotel*", and "*Rent Car"), "enabled")* will take effect when the end-user claims to be a traveller by modifying her personalization information explicitly. This results in the inclusion of the services in *ServiceSet* to her PASS.

Domain experts are normally assumed to be responsible for defining personalization rules concerning specific services in their domain. The end-user may also disable/enable/add/delete/modify the default personalization rules according to her real needs.

4.3 Constructing Personalized Active Service Spaces

To illustrate how the PASS is constructed from the common set of business services for a particular end-user, let us return to the case of Mr. John Bull attending Olympic Games.

Common Service Space. The common service space of the FLAME2008 platform contains all kinds of business services provided for end-users. These services are from a large variety of business domains and will serve a wide range of end-users from athletes, game spectators to tourists. Figure 3 shows a much simplified example of the service space (the left-most circle). The business services shown are from four domains: Travel, Ticketing, Tourism and Food & Beverage. Generally, the difficulty of finding a specific service from a very large set can be significant and quickly grow with the number of constituent business services.

Active Service Space. The common service space becomes an active service space after dependency rules between the business services are added. The specification of dependency rules can be conducted by domain experts, meta-users, or end-users. In the example of Figure 3, the circle in the middle is an active service space with added dependencies such as "**precedes**" and "**leads to**".

Personalized Active Service Space. The addition of dependency rules does not reduce the number of business services to cater for a specific user's needs for composition. The service space can however shrink to a user-manageable size by applying personalization rules. For example, suppose there is a personalization rule:

("role = traveller", ("Book Flight", "Reserve Hotel", "Rent Car"), "enabled")

If John explicitly sets his role as a traveller, then the above rule will put only three services into the PASS (as shown in Figure 3) but leave the other services aside.

5. Utilizing PASS for Service Composition

In this section, we demonstrate how we make use of the added rules in PASS to ease the end-user's effort in service composition. Basically, we use service personalization rules to help the end-user concentrate on business services relevant to their personal circumstances, alleviating the difficulty of selecting services from a much larger set. On the other hand, we use service dependency rules to provide guidance and conformance checking from the business logic's perspective, and accordingly ensure that the composed application does not violate the norms in each involved domain.

5.1 Application of Service Personalization Rules

As soon as an end-user logs on the end-user service composition environment (e.g. the VINCA studio) and her personalization information is available, the pre-defined service personalization rules will take into effect. Personalization information will be retrieved and used to evaluate the *Condition* property of each active rule. If the *Condition* property of an active rule is evaluated to true, the corresponding services in *BusinessSet* will be chosen to form this end-user's personalized active service space. Furthermore, services that are related by dependency rules to already selected services are also included. The process of filtering the common service space and creating a PASS is transparent to the end-user. Whenever an end-user logs in, her PASS is ready for use. Later, the personalization information may change implicitly (e.g. the geographical location changes automatically) or explicitly (e.g. user changes her preference manually). The end-user may disable/enable/add/delete/modify the default personalization rules pre-defined by domain experts. In any such event, her PASS will adjust itself accordingly.

For example, suppose the common service space currently has three services, *"Book Flight"*, *"Book Game-Ticket"*, and *"Introduce Historic-Site"*. We also have three pre-defined personalization rules:

1. *("role = traveller", ("Book Flight", "Reserve Hotel", "Rent Car"), "enabled")*
2. *("role = Game spectator", ("Book Game-Ticket", "Deliver Game-Ticket", "Get Game-Schedule"), "enabled")*
3. *("location = historic site", "Introduce Historic-Site", "enabled")*

When the end-user logs in and explicitly sets her role as traveller, then the first rule will take into effect. Since there are only three services available now, there will be only *"Book Flight"* in her personalized service space. Later if the end-user's location is within a historic site, the third rule will take into effect automatically and the *"Introduce Historic-Site"* service will be added to the end-user's service space.

5.2 Application of Service Dependency Rules

When the end-user adds and links a service to other services in a composition, the application of relevant dependency rules can help the end-user in two ways: composition guidance and conformance validation. Both of them rely on constructing a finite state automaton (FSA) for each dependency rule and advancing the automaton as the user adds a new step to the composition. Due to space limitations, the readers are referred to [[15], [16]] for detailed descriptions about FSA construction. To illustrate, example FSAs for precedence and causality rules are shown in Figure 4, where circles are states, among which final states are identified by double circles.

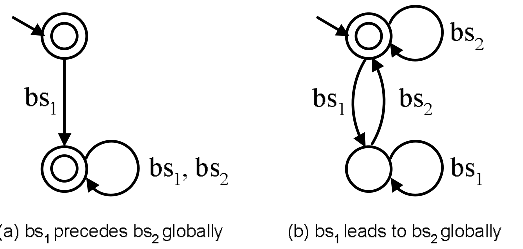
bs_1
bs_1, bs_2
(a) bs_1 precedes bs_2 globally
bs_2
bs_1
bs_2
bs_1
(b) bs_1 leads to bs_2 globally

Figure 4. Example FSAs for dependency rules

Composition guidance. Relative to the service in question, composition guidance refers to the ability of suggesting services for the next steps of the composition. For example, suppose bs_1 is the service in question (meaning that bs_1 has been included in the composition), then the application of a rule like "bs_1 **precedes** bs_2" or "bs_1 **leads-to** bs_2" will list bs_2 as a potential next step, indicating that bs_2 may happen next or in the future. Generally, a service *bs* is recommended when all the following conditions hold:

1. *bs* is in the user's current PASS;
2. *bs* is enabled at the current state of each of the dependency rule automata concerning *bs* and a service added in the past, and the current state is not the initial state;

Conformance validation. To determine if a composition conforms to a dependency rule, we transform the composition into a FSA and use conventional algorithms to check the simulation relationship between it and the FSA representing the dependency rule. More specifically, at the composition time, when a service is added to a composition or the termination point is encountered, we obtain each path from the starting point of the composition to the service or termination point and update the states of all relevant dependency rule automata. If a path is not acceptable by an automaton (ignoring all irrelevant services), we will issue either an error message or a warning, depending on whether the rule is prescriptive or suggestive. In particular, violations to prescriptive rules are considered as composition errors and the end-user is required to make a correction before the composed application can be executed. For instance, suppose *"Book Game-Ticket"* **leads to** *"Deliver Game-Ticket"* **globally** is a prescriptive causality rule. Then if the user has chosen *"Book Game-Ticket"*, she must choose to include *"Deliver Game-Ticket"* between the booking service and any termination point of the composed application. In contrast, assume *“Book Flight”* **precedes** *“Book Hotel”* **globally** is a suggestive precedence rule, then the user may choose to ignore this rule and book a hotel before booking a flight.

The use and on-demand application of service dependency rules can provide proactive support for end-user service composition in terms of guidance and feedback, and can therefore lead to consistent and efficient service composition.

5.3 Motivating Scenario Revisited

According to his itinerary, John first acts as a traveller, and then he wants to act as both a game spectator and tourist. Suppose three personalization rules are associated with John’s PASS:

1. *(“role = traveller”, (“Book Flight”, “Reserve Hotel”, “Rent Car”), “enabled”)*
2. *(“role = game spectator”, (“Book Game-Ticket”, “Deliver Game-Ticket”, “Get Game-Schedule”), “enabled”)*
3. *(“role = tourist and location = historic site”, “Introduce Historic-Site”, “enabled”)*

Then in Figure 5, we show the dynamic change of PASS and also the corresponding recommended service list.

6. Related Work

In this section, we discuss four areas of related work: service registry, business rules, service dependency, and service personalization.

Service Registry: PASS enhances the common service space with service dependency rules and service personalization rules. The common service space can be implemented in a full-fledged registry like UDDI and semantic registry GRIMOIRES [18]. The rules can be encoded as metadata to the services.

Business Rules: PASS features two kinds of business rules: service dependency rules and service personalization rules. A business rule is a statement that defines or constrains some aspect of the business [19]. [20] classifies business rules into four categories:: constraint rules, action enabler rules, computation rules and inference rules. Under this classification, service personalization rules are action enablers, their action is to put valid services into PASS and remove invalid services from PASS. On the other hand, service dependency rules are constraint rules.

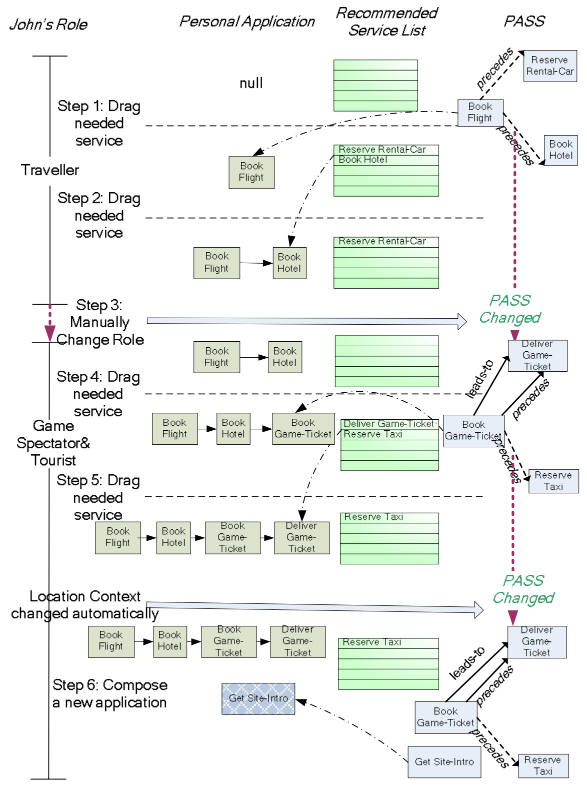

Figure 5. Automatic adjustment of PASS and guidance for John’s service composition

Service Dependency: Service dependency may be described using formal specification languages such as finite state automata, temporal logics, process algebra, Petri nets, etc. We chose to use Dwyer *et al.*’s pattern system [14] due to its intuitiveness, which makes it easier for end-users to accept and use. We define the patterns’ semantics using FSA [15,16]. Building upon FSA enables us to easily extend the pattern system with additional patterns and scopes.

Service Personalization: Personalization has been a very hot topic in the World-Wide Web and Web service research communities [21]. [22] proposed a set of event-condition-action (ECA) style personalization rules to resolve the conflicts arising in Web service federation. [23] proposed a framework for automatic and dynamic composition of personalized Web services and uses the end-user’s personalization information in service discovery. [24] proposed an approach that can personalize Web services composition and provision with user context. [25] describes a context-aware mobile tourist application that can recommend services to users according to their interests and current context. However, to our best knowledge, there is no other approach that enhances a service registry with personalization rules.

7. Conclusion

For end-users to compose services directly and effectively, not only is a user-understandable business-level service abstraction needed, but also is the provision of guidance and means to ensure correct composition essential. In this paper, we have introduced the concept of personalized active service spaces. PASS enhances the common business-level service space with service dependency rules and personalization rules. It offers effective assistance to the end-user in building personalized service-based applications through composition guidance and conformance checking. Currently, we are building a prototype implementation of PASS and example scenarios. Our future work includes design a comprehensive methodology for the management and use of PASS.

8. References

[1] Y. Han, H. Geng, H. Li et al, "VINCA - A Visual and Personalized Business-level Composition Language for Chaining Web-based Services," *Proc.1st Int'l Conf. on Service-Oriented Computing (ICSOC03)*, LNCS 2910, Springer-Verlag, 2003, pp. 165-177.

[2] D. S. Levi and P. Kaminsky, *Designing and Managing the Supply Chain*, McGraw-Hill, 2003.

[3] T. Oinn, MJ Addis, J. Ferris et al., "Delivering Web Service Coordination Capability to Users", *Proc.14th Int'l Conf. on World Wide Web*, ACM Press, 2004, pp. 438-439.

[4] R. Akkiraju, K. Verma et al, "Executing Abstract Web Process Flows", *Proc.14th Int'l Conf. on Automated Planning and Scheduling*, 2004.

[5] B. Holtkamp, R. Gartmann, and Y. Han, "FLAME2008-Personalized Web Services for the Olympic Games 2008 in Beijing," *Proc. eChallenges 2003*, Bologna, Italy, 2003.

[6] J. Yu, J. Wang, Y. Han et al., "Developing End-User Programmable Service-Oriented Applications with VINCA," *Proc. 2nd Ljungby Workshop on Information Logistics*, Ljungby, Sweden, 2004.

[7] J. Yu, J. Fang, Y. Han et al, "An Approach to Abstracting and Transforming Web Services for End-user-doable Construction of Service-Oriented Applications", *Proc. 2nd Int'l Conf. on Grid Service Engineering and Management (GSEM'05)*, Erfurt, Germany 2005.

[8] J. Wang, J. Yu, and Y. Han, "A Service Modelling Approach with Business-Level Reusability and Extensibility", *Proc. IEEE Int'l Workshop on Service-Oriented System Engineering (SOSE2005)*, Beijing, China 2005.

[9] Open Travel Alliance (OTA), http://www.opentravel.org/.

[10] Health Level 7 (HL7), http://www.hl7.org.

[11] T. Malone, K. Crowston, and G. Herman, eds., *Organizing Business Knowledge: The MIT Process Handbook*, MIT Press, 2003.

[12] A. Pease and I. Niles, "IEEE Standard Upper Ontology: A Progress Report," *Knowledge Engineering Review, Special Issue on Ontologies and Agents*, vol.17, 2002, pp. 65-70.

[13] Teknowledge Corporation, *Transportation Domain Ontology and Financial Domain Ontology*, http://ontology.teknowledge.com

[14] M. Dwyer, G. Avrunin and J. Corbett, "Property Specification Patterns for Finite-state Verification", *Workshop on Formal Methods in Software Practice*, 1998.

[15] Y. Jin and J. Han, "Consistency and Interoperability Checking for Component Interaction Rules", *Proc.12th Asia-Pacific Software Engineering Conf. (APSEC'05)*, pages 595-602, December 2005, IEEE Computer Society Press.

[16] Z. Li, J. Han, and Y. Jin, "Pattern-Based Specification and Validation of Web Services Interaction Properties", *Proc. 3rd Int'l Conf. on Service Oriented Computing (ICSOC'05)*, pages 73-86, December 2005, Springer.

[17] A. K. Dey, G. D. Abowd, and D. Salber, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", *Human-Computer Interaction Journal, Special Issue on Context-Aware Computing*, 16(1), 2001.

[18] W. Fang, S. C. Wong, V. Tan, S. Miles, and L. Moreau, "Performance Analysis of a Semantics Enabled Service Registry" *Proc. 4th UK e-Science All Hands Meeting (AHM)*, Nottingham, 2005.

[19] The Business Rules Group, "Defining Business Rules, What are they really?", http://www.businessrulesgroup.org, July 2000.

[20] B. von Halle, *Business Rules Applied: Building Better Systems using the Business Rules Approach*, Wiley, 2001.

[21] M. Bonett, "Personalization of Web Services: Opportunities and Challenges" http://www.ariadne.ac.uk/issue28/personalization/.

[22] V. R. Aragão and A. A. A. Fernandes, "Conflict Resolution in Web Service Federations", *Proc. Int'l Conf. on Web Services -Europe 2003*, LNCS 2853.

[23] W. Zahreddine and Q. H. Mahmoud, "A Framework for Automatic and Dynamic Composition of Personalized Web Services", *Proc. 19th Int'l Conf. on Advanced Information Networking and Applications (AINA'05)*, Taipei, Taiwan 2005.

[24] Z. Maamar, G. AlKhatib, and S. K. Most΄efaoui, "Context-based Personalization of Web Services Composition and Provisioning", *Proc. 30th EUROMICRO Conf. (EUROMICRO'04)*, Rennes, France 2004.

[25] M. van Setten, S. Pokraev, and J. Koolwaaij, "Context-Aware Recommendations in the Mobile Tourist Application COMPASS", *Proc.3rd Int'l Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2004)*, Eindhoven, Netherlands 2004.