User-centric service selection, integration and management through daily events

Zhenzhen Zhao, Nassim Laga, Noel Crespi

User-Centric Service Selection, Integration and Management through Daily Events

Zhenzhen Zhao, Nassim Laga and Noel Crespi

***Abstract*—This paper presents an end-to-end framework to manage user-centric services through daily events. In contrast to existing service discovery, selection and composition approaches, the proposed framework addresses the management issue from a new perspective by firstly learning end-user's intent through daily events, while recommending relevant functionalities to the user; and then enabling the user to select the services offering the required functionalities based on their own selection rules. An event hierarchy and a selection model are proposed respectively to retrieve relevant functional requirements and specify the service selection rules in response to the user's non-functional requirements. The context-oriented system framework for functionality discovery, user-centric service selection and intuitive service composition, is also presented in detail. Finally, an event based service provider (EBSP) system is introduced as a proof-of-concept of the proposed approach.**

***Index Terms*—Event, User centric service, Service composition, Service management, Service selection**

I. Introduction

The current web is being increasingly adopted as a user-centered vision. Not only the service providers are provisioning new methods for user-centric services, but also the users are becoming more powerful in the participation of the service management even service creation process. This phenomenon, which is termed user generated service (UGS) [1], has been gradually encouraging end-users for self service composition.

As services are becoming more prevalent and users are becoming more initiative, tools are needed to help users find, filter and integrate services [2]. Various standards and developments in the markets have been proposed in service discovery, selection and composition which leverage users' service creation possibilities. From the end-user perspective, three main challenges are faced during their service creation process:

-- How to discover the relevant and useful functionalities (to be composed) in response to their dynamic context and intension?

-- After discovering the functionality, how to select the best available service from the pool of services which serve the same functionality?

This work is carried out in the frame of the Eureka-ITEA 2 "Do-it-Yourself Smart Experiences" project. It is supported in part by the French Ministry of Industry.

-- Finally, how to integrate the selected services in an intuitive and easy-to-access way?

To solve the first question, current solution in the developments exhibit large service database and often permit access to third party for increased system functionality. The database consists of core building blocks of the integration, where user can perform the semantic search. However, rich service database is not necessarily providing a better solution and quality of experience for the user. Tracking user's intention and preference for automatic service discovery, selection and recommendation become particularly important.

Our aim is to design an intention based approach for the end-user to create and manage their own services in response to their dynamic context. Followed by a survey conducted on user perception of service mashups [3], this paper presents an event-based service integration framework. Our approach targets the concept of event which is the direct way to reflect user intention. We acquire the context information through user generated event, followed by the recommendation of precisely relevant functionalities, and then we allow the user to select the services based on their own selection rules, the selected service are finally integrated in an aggregated manner. Our main contributions are three-fold, by solving the three challenges respectively:

--Firstly, the proposed framework allows the user to play an active role in the context acquisition through event creation. Instead of searching service from the large database by user themselves, our proposed framework provides the contextual service recommendation. The recommendation logic is performed taking into account the overall parameters of the event details, as well as the user profile and user history, to analyze the precise functionalities of interest to the user.

--Secondly, in order to select the best available service from the pool of services with the same functionality, a selection model is proposed to express user desires, non-functional and dynamic requirements. It provides the users with the capability of choosing the selection rules to apply during the process of selecting services at runtime.

--Thirdly, the proposed framework allows the user to do service integration at the presentation layer. Users can visually select services from the recommended pool of services by specify their own selection rule, without having to worry about programming like visual data flow diagrams.

The remainder of the paper is organized as follows. In Section II, related work and existing approaches for service

discovery, selection and composition are briefly discussed. Section III presents an overview of the proposed framework for user-centric service management. Section IV and Section V discuss the details of the proposed event hierarchy and the service selection model, respectively. A usage scenario and prototype are introduced in Section VI. Finally, Section VII concludes the paper.

II. Related Work and Motivation

The ongoing evolution in UGS is evident in the increasing trend of end-users owning their environments to create their services from heterogeneous resources. In the introduction part we address three challenges in the service creation process, which are functionality discovery, user-centric service selection, and intuitive service composition. In the following paragraphs we will discuss the state of the art by going to each of them respectively.

A. Functionality Discovery

The first challenge in service creation is to discover the functionality (building blocks) based on the user's requirements. Search engine in large database is always considered discouraging, how to capture user behavior and preferences, or how to let user expose their needs and intentions in an effective way, becomes essential. In the current research, one dynamic approach is through natural language (NL) request [4]. Based on the user request, the NL analyzer is invoked to analyze the functionalities through key words and logical sequence in the sentence, then it starts the service discovery process to retrieve relevant services, which can be seen as a successful case of intent-based service composition.

User intent and behavior can be also acquired through user context. Extensive works have been done on the context aware systems to provide ubiquitous services based on user's dynamic manner, i.e. to recommend services when the user is in certain situation, such as location-aware systems, context-managing frameworks and context-aware service composition architecture [5-7]. However, the user always plays a passive role when unconsciously receiving system recommended services, an approach needs to be discovered to allow the user to expose their needs and intentions. Moreover, these systems normally take one or limited context parameters (for example, location or time) to handle the user's changing manner, an overall context of the user needs to be investigated.

In our proposal, we target the concept of daily event which is the direct way to reflect user intention. Our proposed framework allows the user to play an active role in the context acquisition through event creation. Moreover, the framework takes the event as an overall parameter to analyze user's intention. More details are presented in Section IV.

B. User-centric Service Selection

After the requirement exposure, the next challenge consists in how to select precisely useful service from a pool of similarly appearing services. Significant research work has been done in both functional and non-functional based service selection. The first approaches focused on the goal based discovery and selection using semantic technologies [8-11], but as several services may have exactly the same functional signature, selecting a service only by matching the user goal and the functionality provided by the service is no longer sufficient. Thus, new approaches which consider non-functional parameters such as the Quality of service (QoS) and the user context have emerged [12-18]. The non-functional parameters could be static such as price, or dynamic such as user location or the user presence status. In [14] for example, authors propose the eFlow framework: a service composition framework that supports automatic adaptation according to the composite service parameters. The framework enables the users to express their needs through service nodes and associate to each of them a selection rule that refers to runtime values of existing parameters of the composite service. [15] introduces an approach where context information is published with services, and user context be included in the request to enable the system to match the best service. However, these approaches rely on the rules defined by the selection mechanisms themselves and do not enable the user to specify their own rules during the selection process, which are considered not user-centric.

Indeed, for the same service, the different user may have different selecting preferences, and for the different services, the same user may set different selecting parameters. For example, while certain users may want to select the service that minimizes the price, others may want to select the service that is most suited to their context, while some others may want to select the service according to the language they speak.

We propose in this paper a generic selection model to enable the definition of selection rules that are able to take into account both static and dynamic parameters. Through this model, user is allowed to specify which rule to apply in the selection process of a given functionality. More details are shown in Section V.

C. Intuitive Service Composition

After discovery and selection, the third step in service creation process is service composition. Recently much research is being carried out to introduce new and intuitive platforms for service composition, and to address issues pertaining to this process. [19] introduces Mashmaker, to allow non-expert users to easily create their own mashups based on data and queries produced by other users and by remote sites. Using this tool, users can create mashups by 'browsing' around, without need to type, or plan in advance what they want to do. This particular approach, calls for a much intuitive browsing experience and adequate system recommendation to users. [20] describes Margmash as a tool which allows end-users to add mashup fragments to their favorite web sites. Service composition is done within the browser environment and involves various steps like creating a page for instance, identifying visual clues by selecting markup chunks, and identifying markup fragments. However, these models are still regarded complex, subject to the need of understanding data flows between the services.

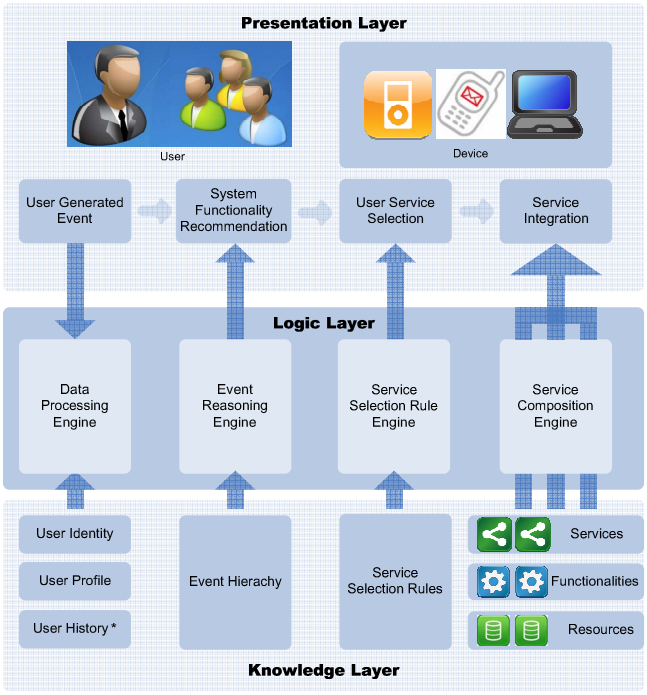

Fig. 1. System framework.

We propose in this paper a strong emphasis on "functionality integration" rather than "data integration. We further manage the intuitive issue by presenting a high level of abstraction to end-users through service aggregation at the presentation layer.

III. Proposed System Framework

This section presents a brief overview of the proposed system framework and its functionalities. The primary goal of the framework is to solve the three limitations in the service creation process as described in section II. The proposed framework consists of three layers: *Knowledge Layer*, *Logic Layer* and *Presentation Layer*. Firstly, in the *presentation layer*, the user exposes his intent by entering the details in a daily event like meeting, travelling etc. Then the user input data with user identity and profile are passed to the *logic layer*, where the data is processed, and reasoning is performed. During this process, resources and rules are looked up from the *knowledge layer* for the system recommendation and user service selection. Finally, the service integration is done in the presentation layer. Fig. 1 presents the schematic description of the main components of the framework.

In the *knowledge layer*, there are four local databases in the system: user database (user id, user profile, and user history), event hierarchy database, service selection database (static and dynamic selection rules) and resource database (content, data or application functionality).

The *logic layer* consists of four engines: data processing engine, event reasoning engine, service selection rule engine and service composition engine. The *logic layer* extracts resources from the *knowledge layer* according to two rules: 1) system recommendation based on the event hierarchy, user profile and user's selection history; 2) user service selection based on the selection rules. The subsequent two sections explain the idea of the event hierarchy and describe the steps of the service selection process, respectively.

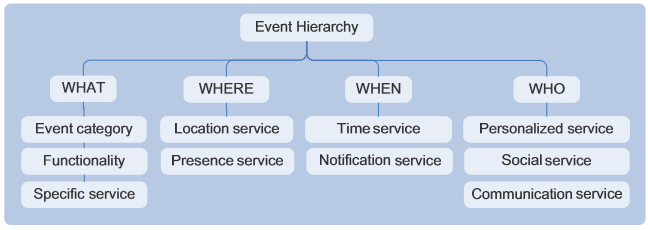

Fig. 2. Functional descriptions of event.

In the *presentation layer*, after the user generating the events by entering the details, the relevant functionalities are pooled and sent back to the *presentation layer* as service recommendation to the user. Of the recommended functionalities, the user selects the services of interest with their own selection rules. The selection of services sends a trigger to the *logic layer* to record the selection history, and update it to the *knowledge layer*. The finalization of service selection will allow the end-user to create a composite application through service integration. The integration is done in a visually intuitive manner, aggregating and linking selected services from heterogeneous resources. The actual application will be a workspace created by the users themselves to suit their individual need, without skills being required for programming.

IV. Proposed Event Hierarchy

This section describes the methodology of the functionality recommendation based on the event hierarchy.

Since our system aims at bringing the concept of event to explore the user intention for service integration and management, the first challenge is how to define event in an efficient way to retrieve and organize relevant services, i.e. the functional description of the event. In current event-based system, the related event elements are nothing less than event theme, occurrence place, occurrence time and involved people, which can be expressed as *what*, *where*, *when* and *who*. In our approach, we follow the same definition of event elements. As shown in Fig. 2, each event element comprises a hierarchy of related information organized in a dependent manner. Each element (attribute) of the event is related to the user's goal, which is further associated with the functional description of the event to retrieve relevant services. *What* defines the user's main objective, which is associated with the event category; *Where* is associated with location and presence service; *When* is functionally related to the time based service and notification service; and finally, *Who* defines whether the event is a personal or a social event, which is associated with personalized service, communication and social service. Note that the event attributes are regarded as the first layer of the event hierarchy.

Among those attributes the event category is the most

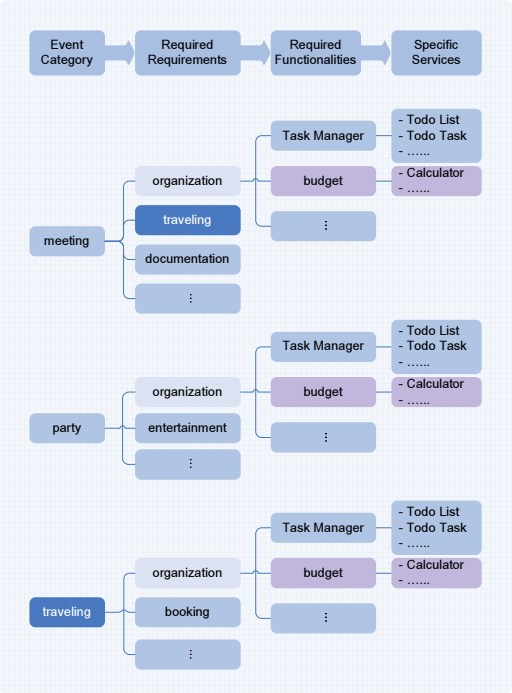

Fig. 3. Example of event category hierarchy.

important factors in choosing related services, which is defined further firstly by the related functional requirements, and then the specific services. The contribution of this event category hierarchy is two-fold. On the one hand, it defines the useful functionality inside each event activity/type for purposes of filtering out less useful or useless services, i.e. increase the accuracy of retrieved services; on the other hand, it provides the relationships among different events, thereby enabling reusability of the functionality for different events.

One example of the event category hierarchy is shown in Fig. 3. In this scenario, the event category is shown as meeting, party and travelling, among which the travelling can be reused as a functional sub-requirement for the meeting. In each of these event activities, at least one reusable requirements "organization" has been included, with the identical list of relevant functionalities and specific services associated. The definition of event category hierarchy not only makes the searching process more accurate (which is particularly important when today the services are growing at an ever faster pace), but also paves the way to understand the user behavior through the user selection in the reusable functionality in different event activities.

The detailed recommendation process is reasoned by two rules: on the one hand, it is based on the current activity of the user which provides an explicit definition of the context for the system. The system performs a contextual, hierarchy based search, for recommending functionalities to the user. The service recommendation feature is useful in providing precise functionalities of interest to the user, in the absence of which the user would have to search for each service manually from the often large service databases, and often yielding less useful

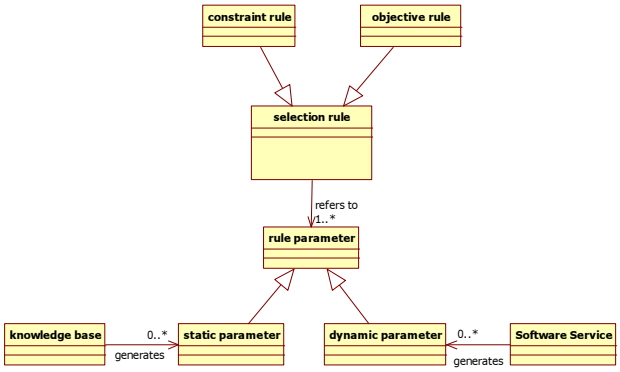

Fig. 4 Service selection rule model.

results. On the other hand, the recommendation process being based on the user profile and service selection history realizes an implicit means of context acquisition by the system, and makes the service recommendation process an automatic one, by keeping track of user preferences in the system. It is particularly important to keep track of user's activities in the system, as it is a means to know about user's preferences, which can be different to another user in the system. This feature of creating session management to keep track of user behavior, and recommend functionalities based on user's preferences and current activity brings in the system context-orientation and personalization abilities, which are considered much important to improve user experience in information systems.

V. Proposed Service Selection Model

In this section, we discuss the user selection process of the specific services from the similar or the same functionality. In order to select the best available service at runtime, we need mechanisms to assess services and to decide which one to select. As we previously detailed, the assessment of these services differ from one user to another. As a consequence, we need a user-centric approach in the selection process: an approach where the users specify themselves the rules to apply in the selection mechanism. This requires a generic model, where both static and dynamic parameters could be considered. Fig. 4 describes the model which focuses on the selection rule class.

There are two types of rules:

--Constraint rules: aims to remove services that do not fulfill a list of constraints. The result of the rule evaluation is true or false value. Constraint rules enable the database to specify conditions that the selected service must satisfy. For instance, if we consider an SMS sending functionality, a constraint rule could be formulated as follows: select services whose home network location is the same as the location of the recipient.

--Objective rules: aims to rank a service from the perspective of a given objective. The result of the rule is a quantitative value that enables the classification of the different services. Objective rules might be for instance the optimization of the price of the selected service. It can also be a linear objective function that refers to several parameters such as

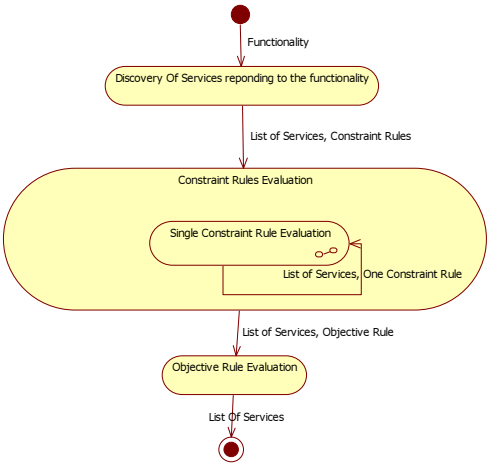

Fig. 5. Service selection algorithm.

price, bandwidth, or reputation.

Both constraint rules and objective rules indirectly refer to static parameters and/or dynamic parameters. Static parameters are those whose value is known before runtime; service price and user preferences are typical examples of such parameters. Dynamic parameters are instead those whose value is known only at runtime. These parameters are usually results of the execution of other services such as presence and location. Thus, in order to cover all possible parameters in our rule model, we assume that each parameter (static or dynamic) is either known by the system database (e.g. service price, negotiated QoS parameters…etc), or require the execution of a service (e.g. presence status, user location…etc). The former are generated through the invocation of the knowledge base component, and the latter are generated through the invocation of the corresponding service in the resource database. Therefore, in the rule language, it is important to be able to refer to the knowledge base component as well as to any existing service present in the database.

In order to select the best available service that responds to the functionality, satisfies constraint rules and optimizes the objective rule selected by the user, we introduce the Interpreter component. As we illustrate in Fig. 5, the first action carried out by the interpreter component is the discovery of all available services that perform the received functionality. Thereafter, the discovered services are filtered according to a set of constraint rules. Each constraint rule may refer to the static parameters, the dynamic parameters, and the inputs provided by the user. The static parameters are referenced through the knowledge base component (e.g. services prices, and QoS parameters); the dynamic parameters are referenced through the corresponding service (e.g. invocation of localization for a user location parameter); and the inputs are referenced through the corresponding tag. Once all constraint rules are applied and a set of services is selected, the interpreter evaluates the objective rule if present. Such as constraint rules, the objective rule may refer to static parameters, dynamic parameters, and the inputs values provided by the user. At the end, a set of services is

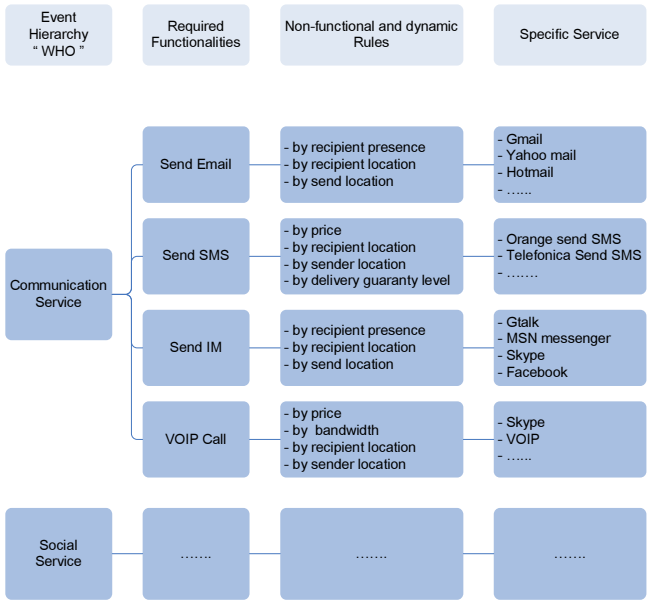

Fig. 6. Example of the service selection process.

selected: services that satisfy the constraint rules and optimize the objective rule. This list of selected services is sent back to the user. One example of service selection process is shown in Fig. 6.

VI. User Scenario and Prototype

In this section we introduce our prototype - event based service provider (EBSP) system. Firstly, to proof the concept of intention based functionality discovery, we provide an interface that allows the end-user to enter event details and get recommended functionalities (Fig. 7-8). Secondly, the system enables the user to personalize the selection criteria of a given functional need, select the best service, and execute it. (Fig. 9) The aggregation of selected services is presented as a collection of ready-to-use gadgets on the calendar, which is an intuitive service composition approach (Fig. 10).

Fig. 7. User input in the event details.

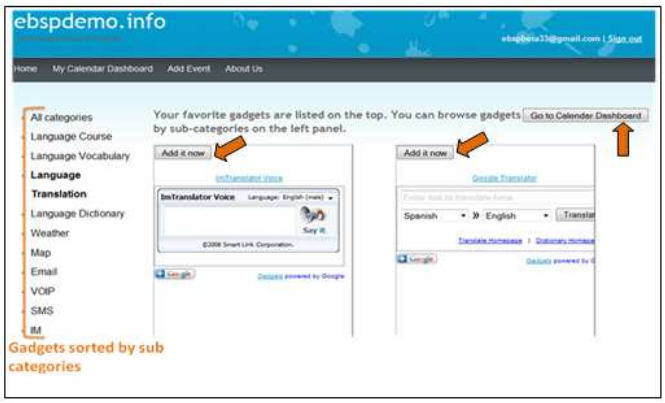

Fig. 8. Service recommendatio. System automatically recommends functionalities, based on the event entered details, user profile and user selection history, which are shown as event categories in the left panel. The default view gives all services in all for the chosen event. User can browse from different categories to find precise services of interest.

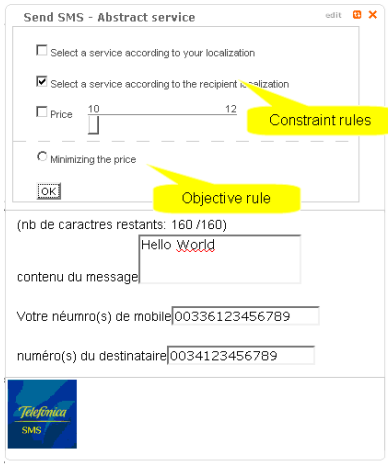

Fig. 9. User-centric service selection. For each functionality that the system recommend, the user needs to select the rules to apply to get the specific service. After selection of services, the user clicks on "Go to Calendar Dashboard".

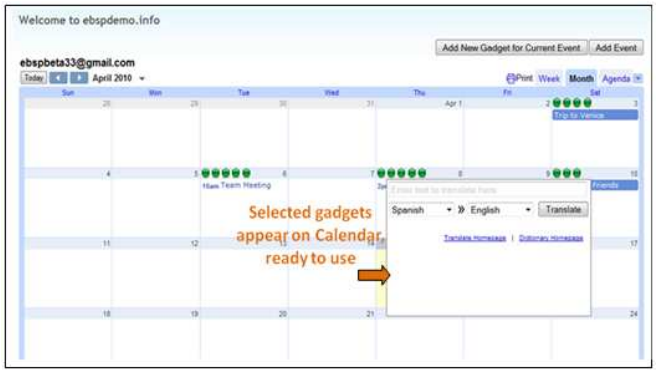

Fig. 10. Service integration on the calendar. User-selected services are mashed up on the calendar (shown as circular green icons on top of the specified event), and can be accessed and used directly from the calendar.

VII. Conclusion and Future Work

In this paper, we have presented an end-to-end framework to manage the user-centric services though daily events. The contributions for the service creation process including event-based functionality discovery, user-centric service selection, and intuitive service integration have been illustrated respectively. We have further implemented an event based service provider (EBSP) system to prove our concept. Concerning the limitations of our work, at present the service selection rules haven't been integrated into the current prototype. One direction for future work will be semantic approach for the functionality recommendation.

Acknowledgment

Our sincere thanks to Ms. Sirsha Bhattarai for her user research studies on end user perspective of service creation, and Mr. Hui Wang, Mr. Honguang Zhang for their efforts on system construction.

References

[1] Z. Zhao, N. Laga, N. Crespi, "A Survey of User Generated Service," in *in Proc of IC-NIDC 2009*, Beijing, China. pp. 241--246, Nov. 2009.

[2] E. Sirin, J. Hendler, and B. Parsia, "Semi-automatic composition of web services using semantic descriptions," in *Web Services: Modeling, Architecture and Infrastructure" workshop in conjunction with ICEIS2003*, 2003.

[3] S. Bhattarai, Z. Zhao and N. Crespi, "Consumer Mashups: End-User Perspectives and Acceptance Model," in Proc of iiwas 2010, Paris, France, 2010

[4] F. Lécué, E. Silva, L.F. Pires, "A Framework for Dynamic Web Services Composition," in *WEWS 07*, Halle (Saale), Germany, 2007.

[5] L. Baresi, D. Bianchini, V. D. Antonellis, M. G. Fugini, B. Pernici, P. Plebani, "Context-aware Composition of e-Service. In Technologies for E-Services," *Third International Workshop, TES 2003*, Berlin, German, Sep, 2003.

[6] A. Sanchez, B. Carro, S. Wesner, "Telco services for end customers: European Perspective," in *Communications Magazine*, IEEE, vol.46, no.2, pp.14-18, Feb 2008.

[7] A. K. Dey, D. Salber, G. D. Abowd, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *Human-Computer Interaction*, 16, pp, 1-67, 2001.

[8] N. Blum, S. Dutkowski, T. Magedanz, "InSeRt - An Intent-based Service Request API for Service Exposure in Next Generation Networks," in *Proc of 32nd Annual IEEE Software Engineering Workshop*, Porto Sani Resort, Kassandra, Greece, Oct 2008.

[9] C. Rolland, N. Kraiem et R. Kaabi "On ISOA : Intentional Service Architecture," in *CAISE'07*,Trondheim, Norway, pp158-172, Springer, 2007

[10] F. Piessens, B. Jacobs, E. Truyen, W. Joosen, "Support for Metadata-driven Selection of Run-time Services in .NET is Promising but Immature. *Journal of Object Technology, Special issue: .NET: The Programmers Perspective*, 2003.

[11] Google. Intents and Intent Filters. Available at http://developer.android.com/guide/topics/intents/intents-filters.html, accessed on August 12th, 2010.

[12] Ben Hassine, A., Matsubara, S., Ishida, T. A constraint-based approach to horizontal web service composition. In ISWC, pp. 130-143, (2006).

[13] Santhanam, G. R., Basu, S., and Honavar, V. On Utilizing Qualitative Preferences in Web Service Composition: A CP-net Based Approach. In Proceedings of the 2008 IEEE Congress on Services, Services - Part I, 2008, vol., no., pp.538-544, 6-11 July 2008.

[14] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, M. Shan, "Adaptive and Dynamic Service Composition in eFlow". in *Proc of the 12th interna-tional Conference on Advanced information Systems Engineering* Lecture Notes In Computer Sci-ence, vol. 1789. Springer-Verlag, London, 13-31. Jun. 2000

[15] G. Spanoudakis, K. Mahbub, A. Zisman, " A Platform for Context Aware Runtime Web Service Discovery" in *ICWS 2007*, vol., no., pp.233-240, 9-13 July 2007

[16] M. A. Cibrán, B. Verheecke, W. Vanderperren, D. Suvée and V. Jonckers, "Aspect-oriented Programming for Dynamic Web Service Selection, Integration and Management," *World Wide Web 10*, pp 211--242, Sep. 2007

[17] Y. Liu, A. H. Ngu, L. Z. Zeng, "QoS computation and policing in dynamic web service selection," in *Proc of the 13th international World Wide Web Conference on Alternate Track Papers &Amp*; NY, pp66—73, May. 2004.

[18] Q. Ding, X. Li and X. Zhou, "Reputation Based Service Selection in Grid Environment," in *Proc of the 2008 international Conference on Computer Science and Software Engineering* - Volume 03, Washington, DC, pp58--61. Dec. 2008.

[19] R. Ennals, and M. Garofalakis, "Mashmaker: Mashups for the Masses," in *Proc of the 2007 ACM SIGMOD International Conference on Management of Data*, China, pp. 1116--118, 2007.

[20] O. Diaz, S. Perez, and I. Paz, "Providing Personalized Mashups within the Context of Existing Web Applications," in *WISE*, pp. 493--502, 2007.