On the Design of User-centric Supporting Service Composition Environments

Eduardo Gonçalves da Silva, Luís Ferreira Pires, Marten van Sinderen
Centre for Telematics and Information Technology
University of Twente, The Netherlands
P.O. Box 217, 7500 AE Enschede
Email: {e.m.g.silva, l.ferreirapires, m.j.vansinderen}@cs.utwente.nl

***Abstract*—In a user-centric service creation process, users should drive the service creation, in which services can be composed out of existing services. However, the creation is expected to take place also at runtime and possibly be performed by non-technical users. These users require support in the composition process, since they cannot deal with all the technical details of service composition. Furthermore, users have different characteristics and properties. In this paper we propose a classification of users in terms of their application domain and technical knowledge in order to identify different types of users and their requirements in the service composition process. These requirements can be used to design and build suitable supporting composition environments. Although our classification of users may seem trivial at first sight, it is quite essential for identifying users requirements and derive the appropriate supporting environment. Nowadays most of the approaches are technology-driven rather than user-oriented. We argue that only by taking the user into account a truly user-centric service creation and delivery can be achieved.**

***Keywords*-User Characterisation; Service Composition, User-centric services.**

I. Introduction

With the emergence of service-oriented systems [1] new approaches to deliver more personalised services to end-users are being devised. The delivery of personalised services to end-users may be achieved by composing existing services according to the concrete needs of the end-user. Most developments implicitly assume that the composition has to be performed by the user himself, supported by a composition environment, which is supposed to shield the user from the complexity of the composition process [2] [3] [4] [5]. However, different users may have different characteristics and requirements. Some users have knowledge of the application domain in which they are seeking for a service, while others may not have this knowledge. Some users have technical skills, and are able to understand and manipulate advanced interfaces that mediate the user-composition system interaction process, others do not have such skills. The user situation and preferences are other examples of constraints. These are characteristics to be considered when designing and configuring environments to effectively support users in the service composition process. In this paper we present a characterisation of users of user-centric service creation systems according to the users' knowledge and technical skills. Based on this characterisation we identify the required properties to be considered when designing a supporting service composition environment suitable for each type of user, so that user-centricity is achieved. The supporting composition environment has to behave and offer interfaces in accordance with the type of user being supported. This conclusion seems trivial, but is being ignored in most of the current approaches to service composition.

This paper is organised as follows: Section II discusses some user service composition approaches from the literature; Section III describes in detail the challenges of user-centric service composition, presenting a general architecture and characteristics of these systems; Section IV characterises the users of user-centric systems; Section V presents property profiles that the service composition environments have to support for each type of user; and finally Section VI presents our conclusions and some directions for future work.

II. User Composition Approaches

This section presents and classifies some relevant user service composition approaches from the literature. In order to evaluate them we define a classification schema based on a simplified life-cycle containing the typical phases of a user-centric service composition process.

A. User Service Composition Life-cycle

We consider that a simplified life-cycle for user service composition has the following phases [6]:

1) *Service Request*: in which the desired service requirements are described;
2) *Service Discovery*: in which services that match the service request requirements are discovered;
3) *Service Composition*: in which services are composed to fulfil the user service request requirements;
4) *Execution*: in which the created service is prepared for execution and executed. This phase comprises service deployment, generation of a client interface, and service execution to deliver the requested functionality.

Our life-cycle focuses on the creation process, and does not consider testing nor runtime monitoring and adaptation of service compositions.

B. Classification Schema

Table I shows the classification schema we defined, based on the life-cycle presented in Section II-A, to

evaluate existing user service composition approaches. This classification schema also considers the roles of the users in the process, which may be to compose services or execute the resulting service composition or both. We explain these user roles in Section IV-A.

	User		System
Serv. Request			
Serv. Discovery			
Serv. Composition			
Serv. Execution			
	Manual	Semi-Automatic	Automatic
User	Composer		
	End-user		

Table I
CLASSIFICATION SCHEMA

In our classification schema, for each life-cycle phase we consider three levels of automation: *Manual*, *Semi-automatic* or *Automatic*. *Manual* means that the user takes full control over the activities performed in the life-cycle phase, i.e., there is no automation of the activities of this phase. *Semi-automatic* means that the system interacts with the user to perform the activities associated with a life-cycle phase. *Automatic* means that the system handles the activities of the phase completely, without interacting with the user.

C. Composition Approaches

Many approaches to support users in the service composition process have been reported recently. In this overview we discuss four of these approaches. We selected approaches that target users without advanced programming skills, for which we thing support is lacking. We have selected these particular approaches because they are the most cited and relevant, or have interesting properties for our discussion.

X. Liu et al. [2] propose an approach that mines existing services to collect information (semantic tags) associated these services (Approach I). The semantic tags are created associated to the services by the users. Based on these semantic tags, they create a *cloud tag*, which users can use to select service *types*. Once users select a type, they are prompted with all the services of that type, and they can select one of them. When a service is selected, further services are presented to the users, which have inputs that can be composed with the selected service outputs. The user performs the service composition with a mashup like interface, by incrementally selecting services and adding them to the composition.

J. Han et al. [3] propose the notion of *Business Service* to shield the users from the complexity and technicalities associated to services (Approach II). In this way users only have to know what the service offers, abstracting from all other technical details. To achieve this abstraction, this approach uses semantic descriptions. The composition process uses *service dependency rules* and *personalisation rules*. The first rules define how services relate to each other, allowing in this way the supporting system to give suggestions to the users based on the service that is selection. Personalisation rules allow user roles to be defined. Whenever these roles are triggered, predefined services are made available for the users to access. Both types of rules are defined by domain experts. The authors propose the creation of *active spaces*, which deliver services to the users, according to their roles and the services they select.

A. Ro et al. [4] propose an approach based on the concept of *Story Board* (Approach III). This approach applies a mashup-like interface, and aims at providing a high-level representation of service compositions. A service is represented as a *stone*, a composition as a *story* and the playground to create a service composition as *story board*. The user has *stones*, and has to select and place them iteratively in the *storyboard*. The *stones* available for composition are constrained by the previously added stone (service) to the composition.

In [5] we propose an approach that supports on demand automatic service composition creation (Approach IV). This approach uses ontologies to semantically describe existing services, and to discover and compose them. The users specify declaratively which *goals* they want the service (composition) to deliver, the expected outputs and inputs that they can or want to provide. Based on the specification of the *desired* service, services are automatically discovered and composed, and then one of the matching services is delivered to the user. In this process we have assumed the user knows all the requirements for the desired service beforehand.

Approaches I, II and III follow a mashup-like style, where the user deals with the composition process in a graphical manner. This is a characteristic of many current approaches for user service composition. Graphical approaches are intuitive and appropriate for some users, but even though we consider that they may be too complex for some types of users, specially users with limited technical knowledge or with devices with small screens. Approaches similar to approach IV may shield the user from service discovery and composition details, however, they tend to require a lot of information from the user at the service request phase. This can be a drawback, especially when the users do not have a clear idea of the service they want, and require interaction(s) with the supporting system to increase their knowledge and be able to take decisions. Table II classifies these four approaches, according to the classification schema introduced in Section II-B.

	User		System
Serv. Request	II,IV	I,II,$III*$	
Serv. Discovery	III	I,II,$III*$	IV
Serv. Composition	III	I,II	IV
Serv. Execution		I,II,III,IV	
	Manual	Semi-Automatic	Automatic
User	Composer	I,II,III,IV	
	End-user	$I*$,II,III,IV	

(∗) Deduced from available information, may be incorrect.

Table II
CLASSIFICATION OF EXISTING APPROACHES

III. User-centric Service Composition

This section proposes a general architecture for user-centric service composition systems. This architecture contains the basic components to characterise and reason about the users of user-centric service composition systems.

User-centric service composition concerns with the process of creating new services by composing existing ones based on a specific set of end-user requirements. Contrary to traditional development approaches, in which a service developer creates a new service (composition) to make some functionality available to be used by different types of end-users, in an user-centric environment, the service composition is created for a specific end-user.

A. General Architecture

Figure 1 presents our architecture for a user-centric service composition system. This architecture has three main parts: the *application domain*, the *user* and the *service composition environment*.

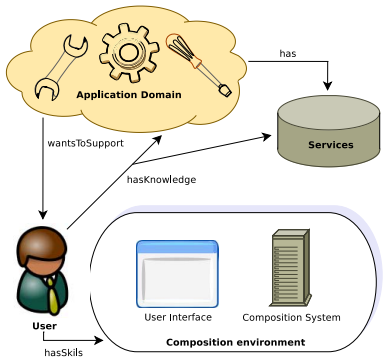

Figure 1. System Architecture

The application domain is part of the users' *real* world, with which the user deals in their daily life, and in which the users can use application services. Examples of application domains are *Telecom* and *Health* domains. In a user-centric service composition system, service providers create services in a given application domain to deliver some functionality to users. A *Service* is used to fulfil some task, or deliver some functionality, in a given application domain (e.g., *send SMS message* in the Telecom domain).

The *composition environment* component is necessary to access services and, if necessary, compose them. This component mediates the interaction between the user and the existing services in a given application domain. In an user-centric service composition process, the composition environment has to shield the user from technical issues that are associated to the composition process. This follows the assumption that not all possible users are professional service developers, i.e., some may have limited technical skills on the service composition process. We divide the composition environment in two main components: the *user interface* and the *composition system*. The user interface allows the user to request, select, compose and (possibly) execute services. The composition system deals with all the technical details associated to the process of discovering and composing services. These two components have to be adaptable to the users being supported.

B. Limitations of Current Approaches

We consider that the majority of users in a user-centric service composition environments are not professional developers. However, this assumption is not enough to define a unique simplified composition environment capable of supporting all possible types of users. In Section II-C we observed that the approaches we considered mainly concentrate on offering intuitive interfaces that allow the users to create new services by using simplified representations of available services and their composition. These approaches facilitate the support to non-professional users in the service composition process. However, we argue that these approaches are not suitable for all possible types of users of user-centric service composition systems. Figure 1 indicates that the user is required to have knowledge on the application domain and its services, and also skills on the composition environment. For example, in the composition approaches discussed in Section II-C, users without enough application domain knowledge or skills on the composition environment would not be able to compose services.

IV. User Characterisation

To overcome the drawbacks of existing approaches, we propose bellow a characterisation of user types in user-centric service composition systems. The objective of this characterisation is to identify the requirements of composition environments, so that they can be made to support *all* types of users on the creation of new services as users require them. We argue that intuitive environments enable more users to create new services. However, we consider that only by taking the specific user of the system into account real user-centric support will be achieved. Different types of users have different knowledge and skills, require different types of interactions with the supporting systems, etc.

A. User Roles

We consider that a user can have two roles: *Composer*, whenever the user creates a service composition or *End-user*, whenever the user executes a service composition. Figure 2 depicts the possible roles of a user in an user-centric service composition system.

In a user-centric process a service is created based on the requirements of a specific *End-user*. However, the *End-user* may not be the one creating the service. Given this, user-centricity paradigm is twofold in this situation. Service is *centric* to its final *End-user*, but the supporting composition environment has also to be *centric* to the user being supported on the creation of a service, i.e., the *Composer*. In this work we specially focus on the *Composer* role. We claim that the *suitability* of the composition environment can only be achieved if

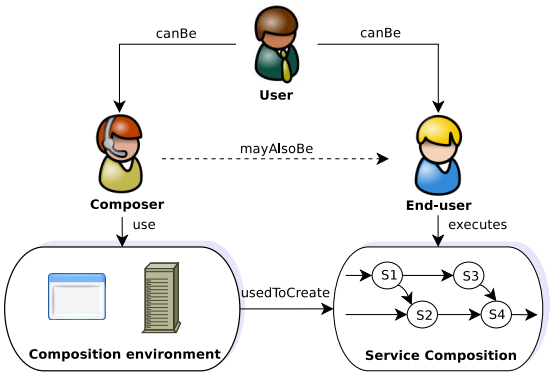

Figure 2. User Roles

the users are characterised and modelled, and then this model is used to adapt the supporting system accordingly. We characterise users in terms of their knowledge and technical skills.

B. Users Knowledge

Users may have different characteristics according to their knowledge of the composition process. We group users according to their application domains knowledge (*domain knowledge* and *services knowledge*), where the user seeks a service, and their *technical knowledge*, or skills, on the environment that supports the service composition process.

1) Domain Knowledge: A user may have different levels of familiarity concerning the *domain knowledge* (concepts and tasks) that may exist in the domain. This type of knowledge is normally associated to how familiar the user is with the application domain. The user can *gather* this knowledge by experience, learning, advertisement, etc.

Domain knowledge is essential for users to define the requirements the service being created need to deliver. For example, if a user wants to buy a mobile phone online, he knows that he should look for an object/concept telephone, which may be bought in a given online store, which he knows that has probably the best prices, etc. Based on this domain knowledge, the user accesses a given online store and specifies his requirements in the provided interface, to check whether the store can deliver the product he wants. If the store cannot deliver this product, it most probably gives some feedback, for example, an alternative products. This feedback can also be interpreted as domain knowledge that the user can make use of, for example, to search for an alternative way to fulfil his goal.

From this simple example we can observe that in the process of service composition often the users start with limited knowledge of the application domain, and require some interactions with components of the domain, possibly executing some services, in order to increase their knowledge to the point of being able to take decisions. Lack of domain knowledge is therefore a constraint to the service composition process. The service composition environment not only has to support the composition process itself, but also support the user to get further information (or knowledge) about the domain, if necessary. This aspect has been ignored in most of the approaches found in the literature.

2) Service Knowledge: The *domain knowledge* is general and orthogonal to all different activities that the user can perform in a domain, not only the activities related with the process of discovery and creation of services. However, the services offered in an application domain require specific domain knowledge, concerning with the delivery of some functionality or product in the application domain.

In the context of user service composition, *service knowledge* concerns the users awareness and understanding of the existing services in the application domain. This knowledge is acquired by interacting with service providers, by learning, by advertisement, etc. With service knowledge the user is able to determine which services he wants to have.

3) Technical Knowledge: Service composition is being used in different applications nowadays, mainly by professional users or developers, which are capable of handling complex tooling and understand the composition process. For example, many companies nowadays use Web services technology, in which WSDL (Web Service Description Language) [7] is used to describe the services available in the company, and BPEL (Business Process Execution Language) [8] is used to compose and coordinate services. However, not all types of users, specially non-professional users, are expected to know these technical artifacts.

For example, if a user wants to find a mobile phone and then buy it, this task may be supported by two services (find and buy services). The supporting environment has to allow the user to find suitable services and then help him in the composition process, by possibly automating some tasks in the process or by suggesting the user directions in the composition process. The supporting tooling may depend on the type of application domain. For example, in the health application domain, more particularly in assisted living scenarios, one can imagine scenarios where a caregiver specifies procedures (or sequence of activities) to supervise a particular patient remotely, e.g., (i) measure blood pressure and (ii) send a message to the patient's doctor, if the blood pressure is above a threshold. Therefore, the caregiver has knowledge about the domain and may also have technical knowledge on the composition environment. In case he does not have this knowledge he may learn it, for example, by training. However, in case a user wants to buy a mobile phone, the user may not know the services available in the domain to realise this objective. To overcome this, the composition environment has to provide suggestions and support to guide the user towards the creation of the service (composition) he wants.

Figure 3 presents an overview on the different aspects associated to the user technical knowledge.

The user is required to understand the composition environment interface in order to use it. To use the environment, i.e., understand what are the different parts

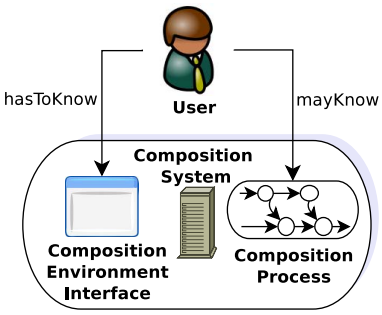

Figure 3. User Technical Knowledge

of the interface, how to interact with it, etc., there should be mechanisms to explain the users the composition environment, possibly "on the fly".

This compulsory technical knowledge of the composition environment is different from the knowledge on the details of the composition process. *Composition process* knowledge is related with the technicalities of the service composition process, namely the notion of service, service interface, etc. Furthermore, this type of knowledge also includes the details normally associated with the activities of the service composition life-cycle (service discovery, selection, composition). The user may not need to know these details, if the supporting composition system is capable of mediating these activities on behalf of the user. However, in some situations the user is aware of these details, in which cases the composition environment can offer a more advanced and detailed interface for the composition process. Users with these characteristics will be able to control more details and possibly tailor further the composition process to their needs.

C. Types of Users

Table III shows our initial classification of user types, based on the user knowledge characteristics presented above.

	User Knowledge		
Type of User	Domain	Services	Technical
Layman	No	No	No
Domain Expert	Yes	Yes	No
Technical Expert	No	No	Yes
Advanced	Yes	Yes	Yes

Table III
TYPES OF USERS

A *Layman* is a user who does not know in detail the application domain, neither has knowledge on the tooling supporting the composition process. This is probably the most common user type in the user-centric service delivery paradigm. A *Domain Expert* is a user who knows the application domain but may not have technical knowledge on the environment supporting the composition process. A *Technical Expert* is a user who has knowledge on the service composition environment, but may not know the application domain in depth. An *Advanced* user is a user who has technical knowledge on the supporting composition environment, and furthermore knows the application domain.

We expect that other user types may be identified possibly in between the types we identified so far. We are investigate how to support these users. However, these user types already give us an indication of the requirements a supporting composition environment has to fulfil.

V. USER-CENTRIC COMPOSITION ENVIRONMENT DESIGN

Based on the type of users and their knowledge, as presented in Section IV-C, we can identify properties that the supporting service composition environments have to offer to achieve user-centric service creation as composition of existing services. To specify these properties we define profiles for the supporting composition environments, in terms of the user service composition life-cycle presented in Section II-A. Table IV presents the profiles we have defined so far. We argue that alternative profiles can be created for these types of users. For example, in Table IV we present more that one profile configuration for each user type.

	User		System
Serv. Request	$D*$, $A*$	L, D, T, A	
Serv. Discovery		$L*$, $D*$, $T*$, $A*$	L, D, T, A
Serv. Composition		T, A	L, D
Serv. Execution		$L*$, $D*$, T, A	L, $D*$
	Manual	Semi-Automatic	Automatic
User	Composer	L, D, T, A	
	End-User	L, $D*$, $T*$, $A*$	

$\{L, D, T, A\}$: preferred profile configurations.
$\{L*, D*, T*, A*\}$: possible, but not preferable, profile configurations.

Table IV
USER PROFILES

In Table IV, *Layman* is represented as L, *Domain Expert* as D, *Technical Expert* as T and *Advanced* user as A. From these users we can derive the required characteristics of the supporting environments.

For *Layman* users, we assume that the service request supporting functionality has to help them concerning domain knowledge, by complementing or suggesting possibilities. Furthermore the supporting environment also has to offer an intuitive interface, since *Layman* users do not have advanced technical skills. The supporting environment has to automate as much as possible the discovery, composition and execution phases due to the limited knowledge of these users. The composition environment needs to support an iterative specification of users' requests, since many times *Layman* users have a set of intentions but not clear set of requirements to drive the service creation process. These users need to interact and improve their knowledge on the application domain, so that they can specify a set of requirements that are sufficient to achieve the creation of a service that satisfies their intentions.

Domain Expert users have knowledge on the application domain, so the specification of service requirements may

not require many interactions with the supporting environment. However, since these users do not have advanced technical knowledge, the supporting interfaces have to be intuitive and guide the users through the service request specification. The discovery, composition and execution have to be as automated as possible, since this type of users may have difficulties to deal with the technicalities of the composition process.

For *Technical Expert* users, the service request may have to be semi-automated, to deliver information about the domain and available services to these users, since they may not have much knowledge of the domain. However, since they have technical skills, the interfaces for service request, discovery and composition may be more advanced.

For *Advanced* users the service request may even be manual, since these users have knowledge of the application domain and have technical skills. However we consider that a semi-automated approach is probably more suitable, since the number of services and elements of a domain the users have to handle may be very large, and even with the necessary skills the users may struggle to specify the services they wants. The process of discovery, composition and execution shall also be as much as possible automated. However, this user may be able to deal with more advanced interfaces compared with the other users.

We consider that this classification may not be unique and probably a "fuzzy" classification may be more appropriate, since users have different properties and as they get familiar with application domains and the supporting environment they may change another user type. A mechanism should be in place to identify and map a user onto the proper user type. We are currently investigating these mechanisms, but in this work we limit ourselves to the characterisation of user types and the required properties of the composition environment to support the identified user types. Even with this simple classification we can observe that the approaches discussed in Section II-C fulfil the requirements of some types of users, but not all of them. New approaches are necessary that can adapt to the user being supported, on demand, so that truly user-centric service creation and delivery can be achieved.

VI. Conclusions and Future Work

This paper introduces and discusses the challenges of designing a supporting environment for users in user-centric service creation systems. We started by analysing the properties of user-centric systems, identifying their general architecture and stakeholders. Based on this we observed that users of these systems may have different characteristics, namely that they may have different degrees of knowledge on the different parts of the system (domain, services and technical knowledge). Therefore, we concluded that the supporting service composition environment has to be capable of complementing the users knowledge, providing them the necessary knowledge when required. Furthermore, the supporting environment has to offer appropriate interfaces, so that the target users can make use of the environment. Based on this, we identified and described four types of users: *Layman*, *Domain Expert*, *Technical Expert* and *Advanced*. For each type we presented possible profiles with properties that a composition environment has to take into account to provide a suitable support to the user.

As future work we will develop techniques to implement composition environments by defining a general and flexible architecture that can be organised according to the types of users being supported. Alternatively, a dedicated architecture for each of the user types can be defined. However, the basic capabilities required in the composition process will be similar for the different types of users. This means that many components of the supporting environments can be reused, i.e., an adaptable architecture that can be configured to fit the different types of users will allow a better support to the users. To achieve this goal users have to be further modelled, not only in terms of knowledge, but also in terms of their situation (or context), their devices, the types of user-system interactions, etc. This user model will allow the supporting system to adapt itself to the type of user, as required. We argue that only by considering such a user model a truly user-centric service creation and delivery can be achieved.

References

[1] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005.

[2] X. Liu, G. Huang, and H. Mei, "A user-oriented approach to automated service composition," in *Web Services, 2008. ICWS '08. IEEE International Conference on*, Sept. 2008, pp. 773–776.

[3] J. Han, Y. Han, Y. Jin, J. Wang, and J. Yu, "Personalized active service spaces for end-user service composition," in *Services Computing, 2006. SCC '06. IEEE International Conference on*, Sept. 2006, pp. 198–205.

[4] A. Ro, L. S.-Y. Xia, H.-Y. Paik, and C. H. Chon, "Bill organiser portal: A case study on end-user composition," in *WISE '08: Proceedings of the 2008 international workshops on Web Information Systems Engineering*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 152–161.

[5] Eduardo Silva, Luís Ferreira Pires, and Marten van Sinderen, "Supporting Dynamic Service Composition at Runtime based on End-user Requirements," in *1st International Workshop on User-generated Services, International Conference on Service Oriented Computing*, Nov. 2009.

[6] E. Silva, J. M. López, L. Ferreira Pires, and M. J. van Sinderen, "Defining and prototyping a life-cycle for dynamic service composition," in *International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing*, Portugal, July 2008, pp. 79–90.

[7] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana, "Web services description language (wsdl) version 2.0," http://www.w3.org/TR/wsdl20/, June 2007.

[8] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana, "Business process execution language for web services, version 1.1," 2003.