Noname manuscript No.
(will be inserted by the editor)

A Social Network for supporting End Users in the Composition of Services. Definition and Proof of Concept

Pedro Valderas · Victoria Torres · Vicente Pelechano

Received: date / Accepted: date

Abstract Nowadays, end users are surrounded by plenty of services that are somehow supporting their daily routines and activities. Involving end users into the process of service creation can allow end users to benefit from a cheaper, faster, and better service provisioning. Even though we can already find tools that face this challenge, they consider end users as isolate individuals. In this paper, we investigate how social networks can be used to improve the composition of services by end users. To do so, we propose a graph-based definition of a social structure, and analyse how social connections can be exploited to both facilitate end users to discover services through browsing these connections, and recommend services to end users during the composition activity. As proof of concept, we implement and evaluate the proposed social network in the context of EUCalipTool, a mobile end-user environment for composing services.

Keywords service composition · end-user development · social network

1 Introduction

Technologies and applications evolve to create new eco-systems of heterogeneous and distributed services that are available for people anytime and anywhere. Nowadays, your environment may be plenty of services that support your daily life: services that track your activity through the mobile phone, that allow you to do an efficient use of your home heating and lighting, that allow you to interact with social networks, that provide you with the weather forecast or traffic status in real time, and so on. Although these services can be used individually, it is their composed usage what has the potential to create new value-added services for end users. In addition, in a world where end users play

Pedro Valderas, Victoria Torres, Vicente Pelechano
Universitat Politècnica de València E-mail: {pvalderas,vtorres,pele}@pros.upv.es

a more and more important role in the development of content, it makes sense to think about the possibility of end users creating new services by composing existing ones. By upgrading end users to prosumers (producer+consumer) and involving them in the process of service creation, both service consumers and service providers can benefit from a cheaper, faster, and better service provisioning [1].

Currently, there exist a myriad of end-user environments that face this composition challenge (e.g.[2] [3] [4] [5]). However, only a few of them consider an aspect that end users demand on current software solutions: social support. Nowadays, millions of people use social networks such as Facebook or Twitter to share with others what is happening in their lives. Messages, images, videos or links are continuously spread through the Internet in order to make people feel that they are connected to others. In the same way, social networks allow people to share their relationships: who are their friends and how many they have, who are their relatives, whether or not they have a sentimental relationship, and so on. Even more, they also share relations with things they have or like. We can find applications that allow people to share books, products, car journeys, homes, etc. So, considering this scenario, why not to share also service compositions with other users through social networks?

Indeed, we think that a social structure created specifically to support end users in the composition of services can introduce several benefits. First, it can be a valuable mechanism to facilitate end users to discover services instead of relying on typical internet discovery solutions that cannot scale to the increasing amount of available services (e.g. [6], [7]). For instance, social relationships can be used to allows end users to browse services within a structure they perfectly know (social networks are currently one of the most used mobile apps [8]); and they can be also exploited to make service recommendations based on friends' interests. Second, a social network can help end users to share knowledge with other end users to improve their skills in composing services. Finally, it can also be a collaboration space among end users and developers, allowing the combination of the innovation and creativity of end users with the expertise of developers.

Considering the motivation presented above, the problem that this work tries to improve can be stated by the following two research questions:

- How can we define a social structure that captures the intrinsic characteristics of the composition of services by end users?
- How can this social structure be exploited to support end users in the composition of services?

The main contributions of this work have been developed to answer the research questions presented above:

- We propose a graph-based definition of a social structure that characterizes the activity of composing services by end users.
- We analyze how the underlying connections that are defined in the social structure can be exploited to both (1) help end users to discover services

by browsing social structure's connections; and (2) recommend services to end users during the composition activity.

As proof of concept, we implement the proposed social network in the context of EUCalipTool, a mobile end-user environment for composing services[5]. This implementation is evaluated through an experiment with end users.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 introduces some foundations about the service composition by end users and an intuitive characterization of the proposed social network. Section 4 presents the definition of the social network with graph theory. Section 5 analyses the defined social structure to delimit how it can be used to browse and recommend services. Section 6 presents the implementation of the social version of EUCalipTool as a proof of concept. Section 7 introduces the evaluation of our work. A discussion is presented in Section 8. Section 9 finishes the paper with some conclusions and the analysis of further work.

2 Related Work

Some works face the challenge of supporting end users in the creation of service compositions with solutions that include some social issues. These social end-user composition approaches can be classified into: (1) community oriented approaches, which introduce features to share the knowledge that is produced within a community; and (2) social network oriented approaches, which are those that create social structures to connect users with common interests. Next, we introduce the related works of each type. Note that many other works allow end users to compose services. However, we have only considered those that include some social issue, single end-user oriented approaches have been omitted.

2.1 Community-oriented approaches

As introduced above, approaches in this category introduce some features to share knowledge within a community of users. These features generally consist of mechanisms to annotate, rank or rate compositions as well as recommendation capabilities. However, there is not a social structure underlying the proposed solutions.

One of the most significant solutions in this category is IFTTT [9]. It is based on trigger-action programming to provide a complete software platform that allows end users to create what they call Applets (condition-action rules), which connect apps, devices and services to trigger one or more automation rules. This platform recommends users with Applets that are created by the community in order to be used and rated, allowing also sharing them by sending their URL through different apps installed in the device (e.g. email, whatsapp, facebook, etc.). The main difference with our approach is that Applets are explicitly shared among one user and another sending their URL. There is not social structure that allows end users to follow others in order to browse the Applets they create. In addition, our approach uses the connections created in the social structure to recommend the composition of services done by others according to similar interests.

Dlvr.it [10] is a web tool that allows end users to aggregate data feeds from different websites, sort and filter out those feeds, as well as mash them up into a single feed. It supports end users in the sharing of their feed aggregations through existing social networks such as Facebook or Twitter. In contrast to our approach, this platform is based on social networks of general-purpose to share aggregation of data feeds. Our work proposes a social network specifically designed to support the activity of service creation which allows exploiting the generated connections to improve the discover of services and their recommendation.

Zapier [11] is a web-environment that allows end user to create their own applications (Zaps) by composing conditions and actions (apps) that are already predefined by professional programmers of several vendors. It allows users to define interests on specific apps, and recommend Zaps (compositions) of other users that include these apps. The recommended Zaps can be taken as a basis to create new ones. The main difference with our work is that a structure to socialize the creation of Zaps is not proposed. Thus, users cannot share Zaps among them. This work shares with our proposal the possibility of making recommendations based on interests. However, these recommendations seem to be supported by a typical content-based algorithm, i.e. they recommend the content (in this case, Zaps) that is similar to the content that a user has already consumed (in this case, zaps marked as interesting). Our approach goes a step beyond and recommend services that may be of interest to a user by considering the similarity among this user and others. This similarity is calculated by analyzing the connections that are implicitly created in the proposed social structure.

Node-RED [12] is a visual wiring tool to create action composition in an Internet of Things (IoT) context. It provides a graphical web environment where a user can define nodes that represent actions over IoT devices, and connect these nodes in order to create a sequence. It provides a flow library to share compositions with other users in JSON format. Their efforts to allow users to share composition among them are limited to creating a repository from which users can download compositions created by others. However, there is not a structure that facilitates to follow users and share composition with them. In the same way, there is not support to make recommendations based on the affinity among users.

2.2 Social network approaches

There are a few solutions that specifically focus their efforts on proposing a social network structure in order to improve the composition of services by end users. One of the exceptions is [13] which presents the foundations to support a social-awareness web service composition and focuses on exploiting the social data to make recommendations. However, authors only use the frequency that a user includes a service in their compositions in order to exploit social data. We propose a social structure with a richer set of relationships. Soriano et al. [14] proposed an inception idea for socializing the composition of services through a user-generated catalog of resources founded on the Web 2.0 vision for user co-production and harnessing of collective intelligence. However, a social structure is not proposed. Jiang et al [15] study the idea of social manufacturing and propose a preliminary solution to achieve that production service providers and prosumers collaborate for production. This work analyzes how specific production services can be shared in order to allow prosumers to use them and create new products. However, the creation of new production services is not considered from a social perspective.

Other works use the concept of social network or a similar one in the area of service science, although they are not focused on end-user development. Similar to our idea, Tamburri et al. [16] introduces the main pillars to create a social network of developers and shows the potentials of this idea in practice, by reporting on its application to a real-life industrial scenario. Maamar et al. [17] introduce a social network of web services focusing on three types of relations: recommendation, similarity, and collaboration. Yu & Woodard [18] create a mashups API affiliation network by analyzing the APIs that were used to create each mashup. This study concludes that the web APIs are used in the creation of mashups by following a long-tail distribution. Chen et al. [19] construct a global social service network and provide generic quality criteria for social links which included dependency satisfaction rate, QoS preference, sociability preference, and preferential service connectivity in order to improve the quality of service management. Ren et al. [20] describe a service social network and five kinds of relationships, namely interactive transaction, co-community, physical distance, resource-related, and social similarity relationship. They discuss how these relationships can create a synergy effect, and develop a service selection model that support social collaboration between services.

In the area of Social Internet of Things, we can find several works that focus their efforts on the idea of converging social networks with IoT. For instance, Kranz et al. [21] analyse the implications of a so-called ‘social-technical network’; the concept of Blogject (objects that blog) is presented in [22]; Atzori et al. [23] study the participation of smart objects in current social networks. Guinard et al. [24] propose the use of existing social networks such as Facebook to share data produced by services. Although all these works are not focused on the composition of services, they have done an excellent job in the analysis of social relationships between intelligent things, which has inspired us in the definition of this work. Following the path of these works, Meissa & Benharzallah [25] analyze social relations to determine the main challenges for the composition of services in Social IoT contexts.

In the context of Cyber Physical Social Systems (CPSS), Wang et al. [26] present an approach to dynamically create service composition. They introduce an algorithm based on QoS fluctuation computation and Skyline component computation that considers QoS constraints in order to satisfy users' requirements.

Although not focused on service composition, social networks have been also considered in the area of end-user development. Reuter et al. [27] propose a EUD environment to analyze social network big data. This environment includes a social media API and a quality assessment service as well as a web application targeted at end users. Massa & Sapano [28] introduce FaceMashup, an end -user development environment supporting the manipulation of the Facebook graph and allowing end users to analyze their social data.

3 Socializing the Composition of Services

According to Boyd & Ellison [29], a social network allows individuals to construct a public or semi-public profile within a bounded system and articulates a list of connections that can be viewed and traversed. Considering this general description, we introduce some foundations on service composition by end users that helped us to intuitively characterize the social network that is proposed in this work.

Characterization 1. Hung et al. [30] define a web service as an autonomous unit of application logic that provides either some business functionality or information to other applications through an Internet connection. In Service-Oriented Computing (SOC), developers use services as fundamental elements in their application-development processes. Service composition accelerates rapid application development, service reuse, and complex service consummation [31]. In this way, developers can solve complex business problems by combining and ordering available basic services to best suit their problem requirement.

Thus, the social network should consider services of two types: Basic Services, which are executable logic units implemented by programming activities; and Composed Services, which are created through the composition of other services. In this sense, a service profile of the social network should differentiate between these two types of services. In addition, there should exist a connection among services that indicates that one Composed Service includes another service in its definition.

Characterization 2. There exist several solutions to help developers in the composition of services [32]. However, the explosion of the number of web services and APIs exposed through the Web has accentuated the need for allowing end users to create their service compositions [1]. We can find a myriad of environments focused on allowing end users to compose services by their own (e.g. [9] [11] [10]).

Thus, the social network should consider two types of users (End User and Developer). End users should have the possibility of creating Composed Services while Developers should be authors of both Basic and Composed

Services. In this sense, the social network should include a connection between a user and a service that indicates that this user is the author of the service.

Regarding the user profiles, the social network should differentiate between End User and Developer. In addition, inspired by existing social networks, we introduce the possibility of creating a connection between two users in order to define the interest of a user in the composition activity of another one.

Characterization 3. In the context of end-user development [33] it is well known that end users have many difficulties to create solutions from scratch. It is a good practice to provide predefined elements that can be taken as a basis to define new ones [34]. The composition of services is not an exception. Environments that support the composition of service by end users usually provide them with predefined compositions to facilitate the creation of new ones (e.g. [35] [9] [11] [14]).

In this sense, a Composed Service should be taken as a basis to create a new Composed Service. Thus, the social network should include a connection that indicates that one Composed Service has been created from another one.

Characterization 4. Currently, the execution of services does not depends only on the computer where they are deployed but also on the context they are being executed (Dey, 2001). For instance, the rising of the Internet of Thing (IoT) paradigm has introduced new services that depend on mobile devices or wearables (e.g. smart phones and watches, glasses, devices incorporated into clothing, and so on) and they must go with the user in order to provide their service wherever the user is [36]. Other services, however, are highly coupled with the physical environment where they are executed. For instance, smart buildings [37] provide services to control lighting, temperature or the doors of specific locations.

Thus, a service can be characterized, among other contextual data, by the devices required for its execution, and the location they are coupled to (if any). In this sense, a service profile should include this data.

Characterization 5. Independently of how end users create a composition of services, from scratch or taking as a basis a predefined one, they need to find, select and include the services they need. According to the analysis done in the previous section, some approaches (e.g. [9] [11] [13]) introduce mechanisms based on recommendations to help end users to discover the service they need. In the context of recommendation systems, one of the most used techniques is based on tags or keywords [38].

Thus, the social network should provide mechanisms to allow a tag-based recommendation of services in order to help users to find those that fit their interest. Thus, a service profile should include a set of semantic keywords that describe its execution logic and can be used to make recommendations. In the same way, a user profile should include a set of tags that describe service domains in which a user is interested in.

Characterization 6. Finally, according to Boyd & Ellison [29] user profiles generally include data that can be used to characterize and "know" the user. In addition to the data introduced above (user type and service domain tags), we consider that a name, one photo and a textual description is enough data to create a user profile for our purpose. In the same way, based on some existing service profiles [39][40][41][42], in addition to the data introduced above (service type, device, location, and semantic keywords), we think a service profile should include a name, a textual description, inputs required to execute a service, and outputs obtained after the execution.

4 Social Network Definition

In this section, we present a semi-formal description of the profiles and connections that define the social network that we propose to support end users in the composition of services. Social networks are usually represented as graphs (Wellman, 1988). We used a directed, typed, constrained and attributed graph [43][44]. In particular, we define the proposed social network with the following graph-based specification:

- There exists a set of vertices V and a set of directed edges E. The functions *source* and *target* indicate the source and target vertex of an edge.
- There exists a set of types for vertices $T_V = \{User, Service\}$ and a function $type_V$ which indicates the type associated to a vertex.
- There exists a set of user subtypes $T_U = \{EndUser, Developer\}$ and a function $subtype_U$ which indicates the subtype associated with a vertex whose type is User.
- There exists a set of service subtypes $T_S = \{Basic, Composed\}$ and a function $subtype_S$ which indicates the subtype associated with a vertex whose type is Service.
- There exists a set of types for edges $T_E = \{follower, author, consumer, includes, definedFrom\}$; and a function $type_E$ which indicates the type associated to an edge.
- Some constraints are defined over types of edges. For instance, a $follower$ edge represents the interest of a user A on the service composition activity done by a user B. Thus, they can only be defined between two users.

  $\forall\, e\ \in E \,|\, type_E(e) = follower \rightarrow type_V(source(e)) = User$
  $\&\, type_v(target(e)) = User$

  The rest of constrains have been omitted in order to not overload the paper. They are defined analogously and can be found in [45].
- A Composed Service is defined as a set of includes edges.

  $\forall s \in V \mid type_V(s) = Service \;\&\; subtype_S(s) = Composed \rightarrow$
  $s = \{e_1, e_2, ..e_N\} \;\&\; \forall e \in s \rightarrow e \in E \;\&\; type_E(e) = includes$
- To represent the order in which services are included in a Composed Service we complement each Composed Service the relation $<$ that defines the order among elements according to the edge creation time. Thus, $e1 < e2$ specifies that $e1$ was created before $e2$. The function definition returns the ordered set associated with each Composed Service. Thus, the definition of a Composed Service cs_i is defined by as an ordered set:

$definition(cs_i) = \{e_1 < e_2 < e_3 < ... < e_N\}$

Functions $first$ and $last$ indicate the first and last *includes* edges of the ordered set associated with a Composition Service. Functions *previous* and *next* indicate the previous and next includes edges of another one included into the ordered set associated with a Composed Service. Note that this formalization defines a Composed Service as a sequence of Basic Services. If we need to consider more complex compositions such as those done with, for instance, BPMN, a Composed Service would also include other elements such as conditions, loops or parallel executions. This could be done including additional types of vertex and edges that represent these elements and the relationships among them. However, this is out of the scope of this paper.

- There is a set of data vertices V_D and a set of vertex-attribute edges E_{VA} together with the $source_{VA}$ and $target_{VA}$ functions that indicate the source and target vertex of each vertex-attribute edges.
- We include a set of types for vertex-attribute edges $T_{VAE} = \{name, location, \quad interest, description, input, ouput, semantics, device\}$ and a function $type_{VAE}$ which indicates the type associated to a vertex-attribute edge.

  $\forall\, eae \in E_{VA} \,|\, type_{VAE}(eae) = input \rightarrow type_V(source(eae)) = Service\,\&\, target(eae) \in V_D$

  The rest of constraints have been omitted to not overload the paper. They can be found in [45].

In order to better understand the proposed social structure Figure 1 shows its definition in a UML Class Diagram. As we can see, there are Users and Services with their properties. There are two types of users: Developers and End Users. There are two types of Services: Basic and Composed.

All users can be followers of other users and consumers of any type of service. Developers can be authors of any type of service. End users can be authors of Composed Services. A Composed Service includes services of any type, and there is a Previous association class that, given the services included in a Composed Service, indicates the service that is previous to another. A Composed Service can be defined from another Composed Service.

5 Exploiting social network's connections to browse and discover services

The main goal of the above-introduced social network is supporting end users in the composition of services. One of the most interesting features that it provides is the possibility of exploiting the set of connections that are created to improve the problem of service discovery. The proposed social structure can be used to allow end users to discover services by browsing profiles, but also to recommend end users with services during the composition process.

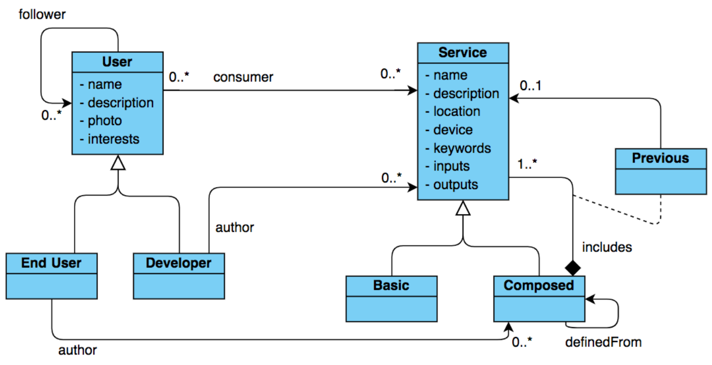


Fig. 1 Class diagram of the proposed social network

5.1 Browsing services through the social structure

Efficiently supporting end users in exploring services created by other users is a key requirement of the proposed social network. Exploring services is the process of navigating through available services and acquiring important knowledge of them [1] and it heavily relies on how the characteristics of services are represented, organized, and rendered.

With the proposed social network, we provide end users with a tool that allows them to find services by browsing a set of users and services profiles. Social networks are currently one of the most used mobile apps [8] and the use of a social structure is familiar to end users, which may facilitate the task of finding services in comparison to current service repositories.

Thus, considering the social structure presented above, a given user can:

1. Browse its $follower$ connections. The user can access the list of its followed users and the data attributes that describe each of them (i.e. their user profiles). To describe this, we propose de function Followed Users (FU):

   $FU(u_i) = \{u_j \in V \,|\, type_V(u_j) = User \,\&\, \exists\, e \in E \,|\, type_E(e_j) = follower \,\&\, source(e) = u_i \,\&\, target(e) = u_j\}$

2. Browse the *author* connections of each followed user. The user can access the services created by each followed user and the set of data attributes that describe each of them (i.e. their service profiles). To describe this, we propose the function Followed Created Services (FCS):

   $FCS(u_i, u_j) = \{s \in V \,|\, type_V(s) = Service \,\&\, \exists\, e \in E \,|\, type_E(e) = author \,\&\, source(e) = u_j \,\&\, target(e) = s \,\&\, u_j \in FU(u_i)\}$

3. Browse the $location, semantics, device, includes$, and $definedFrom$ connections of each service created by a followed user. Thus, given a specific service profile S, a user can browse:
   (a) The services that are associated with the same location as S.
   (b) The services that share some semantic keyword with S.

(c) The services that depend on a same device as S.
(d) The services that include S in their definition.
(e) The services that have been defined taking S as a basis.

A formal definition of these browsing options can be found in [45]. As representative example, we present the function Same Parent Services (SPS), that define the browsing option 3e:

$SPS(u_i, s_i) = \{s_j \in V \,|\, typev(s_j) = Service \,\&\, \exists\, e_j \in E \,|\, type_E(e_j) = definedFrom \,\&\, source(e_j) = s_j \,\&\, target(e_j) = s_i \,\&\, \exists\, e_k \in E \,|\, type_E(e_k) = author \,\&\, target(e_k) = s_j \,\&\, source(e_k) \in FU(u_i)\}$

Section 6.2 shows the implementation of these browsing options and their practical usage in the context of EUCalipTool.

5.2 Recommending services in composition activities

As commented above, end-user environments usually provide the possibility of composing services in two ways: (1) from scratch and (2) from a predefined composition. We focus on analyzing how social network's connections can be used to recommend services to end users in these two situations:

1. End users are creating a Composed Service from scratch. In this case, we want to provide end users with a list of services that can fit their needs before starting the composition of a new service.
2. End users want to compose a service from an existing Composed Service as a basis. In this case, we want to recommend the Composed Services that better fit end users' interests.

In addition to these two situations, service recommendations can also be provided at any time during the composition process, independently of the way end users create a composition:

3. End users have a Composed Service partially defined. In this case, recommendations are focused on providing a list of the most suitable services to be included after the last service included in the Composed Service.

In order to make recommendations in these three situations we propose the following functions:

Composed Services (CS). This function returns the services composed by a user u_i.

$$CS(u_i) = \{s \in V | type_V(s) = Service \,\&\, \exists\, e \in E \,| type_e(e) = author \,\&\, source(e) = u_i \,\&\, target(e) = s \,\&\, = Composed\}$$

Inclusion Number (IN). This function returns the number of times that a user u_i has included a service s_i when composing another service.

$IN(u_i, s_i) =| \{e_i \in E \,|\, type_E(e_i) = inclusion \,\&\, target(e_i) = s_i \,\&\, \exists\, e_j \in E \,|$

$$type_E(e_j) = author\&target(e_j) = source(e_i)\ \&\ source(e_j) = u_i)\} \mid$$

Global Inclusion Number (GIN). This function returns the total amount of services included by a user u_i.

$$GIN(u_i) = \sum_{k=1}^{N} IN(u_i, s_k)$$

Service Interest Level (SIL). This function returns the level of interest that a user u_i has in a service s_i. This level is calculated from the frequency that a service s_i is included in the compositions done by a user u_i. In case GIN(u_i) returns zero, the fraction is not calculated to avoid problems of division by zero, and this function returns 0.

$$SIL(u_i, s_i) = \frac{IN(u_i, s_i)}{GIN(u_i)}\ if\ GIN(u_i) > 0,\ otherwise\ 0$$

Composed Service Interest Level (CSIL). This function returns the level of interest that a user u_i has in a Composed Service s_i in terms of the interest that the services included in the Composed Service has for the user.

$$CSIL(u_i, s_i) = \sum_{k=1}^{N} SIL(u_i, s_k)\, \forall\, s_k \mid \exists\, e \in E\, \&\, type_E = includes\, \&$$
$$source(e) = s_i\, \&\, target(e) = s_k$$

Service User Similarity (SUS). This function returns the level of similarity that a user u_i has with a user u_j in relation to the interest in service s_i.

$$SUS(u_i, u_j, s_i) = \frac{SIL(u_i, s_i) * (1 - 0^{SIL(u_j, s_i)})}{SIL(u_j, s_i) + 0^{SIL(u_j, s_i)}} + \frac{SIL(u_j, s_i) * (1 - 0^{SIL(u_i, s_i)})}{SIL(u_i, s_i) + 0^{SIL(u_i, s_i)}}$$

The SUS function should return the similarity level between two users from the division of the values that return their SIL functions. Instead of that, note that this function returns the result of adding two similar fractions in which the values of the SIL functions are divided but alternating the users between the numerator and the denominator. We do that to make the function symmetric, i.e. the function returns the same value independently of the order in which users are passed as arguments. Note also that to avoid problems of division by zero in case the SIL function returns this value, the numerator of each fraction is multiplied by $1 - 0^{SIL(u_N, s_i)}$, and the denominator is added with $0^{SIL(u_N, s_i)}$.

Global User Similarity (GUS). This function returns the level of similarity that a user a u_i has with a user u_j globally, i.e. considering their interest of both in every service.

$$GUS(ui, uj) = \sum_{k=1}^{N} SUS(u_i, u_j, s_k)$$

Next, we introduce three algorithms that use these function to make recommendations in the three proposed situations. Note that the main goal of these algorithms is to provide contextualized recommendations, i.e. recommendations that fit the end-user situation in a specific moment. To do so, these algorithms try to predict the most probable actions that they can do in the context of the three different situations presented above. For instance, when end users are creating a Composed Service from scratch we want to predict the most probable services that they may want to add at a first place.

In the area of prediction algorithms, the strategy that is usually followed to predict actions is analyzing a sequence of past actions done by a user in order to detect a pattern that matches with the current situation. The proposed algorithms try to do something similar, but instead of having a narrow view of a user's past actions, they have a broader vision when considering what the full list of followed users did before in each situation. For instance, if end users want to add a first service, we analyze what other users did in the same situation, and recommend the services they added first. Other solutions such as recommending the most used service in each situation could be used. However, we think that this solution does not consider properly the context of each situation. In the same sense, other dimensions such as service consumption (usage) are not considered by algorithms since they focus on the situations created when end users are composing a new service. The usage should be considered when recommending end users services to use.

End users are creating a Composed Service from scratch. This algorithm recommends end users the first services used by their followed users in the Composed Services. This last recommendation is defined in Algorithm 1: given a specific user u_i, we obtain all the Composed Services created by its followed users. For each of this Composed Services, we access the service included in the first place. Note that a composition is defined as a sequence of *includes* edges (see Section 4), so we access the target of the first edge to obtain the first service. Next, we calculate for it a Recommendation Level (RL). RL is calculated from the product between the interest of the followed user in the service (Service Interest Level) and the Global User Similarity between both users. Then, the list of services ordered by this level is returned.

Algorithm 1. Recommendation of first services

```
input $u_i \in V \,\&\, type_V(u_i) = User$
  for each $u_k \in FU(u_i)$
    for each $cs_k \in CS(u_k)$
        $s_k = target(first(cs_k))$
        $RL_k = SIL(u_k, s_k) * GUS(u_i, u_k)$
        $Add(s_k, RL_k)$ in RecList
    end for
  end for
  Sort RecList in descending order of $RL_k$
output RecList = $\{(s_k, RL_k)\}$
```

End users want to compose a service by taking an existing Composed Service as basis. This algorithm recommends end users the Composed

Services built by followed users with similar interests. This recommendation is defined in Algorithm 2: given a specific user u_i, we obtain all the Composed Services created by its followed users. For each of these Composed Services, a Recommendation Level (RL) is calculated from the product between the Composed Service Interest Level of the followed user and the Global User Similarity between both users. Then, the list of Composed Services ordered by this level is returned.

Algorithm 2. Recommendation of Composed Services

```
input $u_i \in V \,\&\, type_V(u_i) = User$
  for each $u_k \in FU(u_i)$
    for each $cs_k \in CS(u_k)$
      $RL_k = CSIL(u_k, cs_k) * GUS(u_i, u_k)$
      $Add(cs_k, RL_k)$ in RecList
    end for
  end for
  Sort RecList in descending order of $RL_k$
output RecList = $\{(cs_k, RL_k)\}$
```

End users have a Composed Service partially defined. This algorithm recommends end users the services included in the Composed Services created by their followed users. This recommendation is defined in Algorithm 3. Given a specific user u_i and a service s_i that is the last service included in the composition that u_i is currently creating: first, we access all the users followed by u_i; Next, we obtain all the includes edges that target s_i and whose source is a Composed Service created by each followed user; for each edge, we obtain the service associated to the next includes edge within the service definition they belong; then, a Recommendation Level (RL) is calculated from the product between the interest of the followed user in the service (Service Interest Level) and the Global User Similarity between both users. Finally, the list of Composed Services ordered by this level is returned. Note that this algorithm is applied after a first service is added, each time end users add a new service, independently of the position of the added service within the composition. Thus, when end users want to add the second service, the last added service (s_i) refers to the first service; when users want to add the third service, the last added service (s_i) refers to the second service; and so on.

Algorithm 3. Recommendation of next services

```
input $u_i \in$ V \& type_V($u_i$)=User
  for each $u_k \in FU(u_i)$
    for each $e_k \in \{e_h \in E \mid type_E(e_h) = includes \,\&\, target(e_h) = s_i \,\&$
    $\exists\, e_j \in E \mid type_E(e_j) = author \,\&\, target(e_j) = source(e_h)$ \& $source(e_j) = u_k\}$
      $s_{next}$=target(next($e_k$))
      if ($s_{next}$ != null) then
        $RL_{next} = SIL(u_k, s_{next}) * GUS(u_i, u_k)$
        $Add(s_{next}, RL_{next})$ in RecList
      end if
    end for
  end for
  Sort RecList in descending order of $RL_{next}$
output RecList = $\{(s_{next}, RL_{next})\}$
```

6 Social EUCAlipTool. A proof of concept

In this section, we present an implementation of the social network presented above. To achieve this, the first thing we need is an end-user environment to compose services and extend it to create such a social structure. For this purpose, we used EUCalipTool, an end-user mobile tool for composing services [5].

6.1 Service Authoring Tool

EUCalipTool proposed a mobile authoring environment which allowed end users to compose new services. The services that end users could compose were those registered into the platform by developers. To do so, developers were provided with a service registration form that allowed them to describe a service [5]. This form has been reused to provide developers with a user interface to create Basic Service profiles.

EUCalipTool also proposed an editor to allow end users to create new services by composing those services that had been previously registered by developers. In this work, we have extended this editor in order to allow users from the social network to create services by composing both Basic Services (created by developers) and Composed Services (created by developers or end users). Figure 2 shows some snapshots of the first steps provided by the mobile authoring tool to create a new Composed Service. Such service can be created in two ways (see Figure 2A): from scratch or from an existing service. To create a Composed Service from scratch users must introduce a name and a description by using the proper form. To create a Composed Service from an existing one (Figure 2B), users must select a service from the catalogue and customize it according to their needs. The catalogue of the non-social version of EUCalipTool provided end users with a list of: (1) predefined examples and (2) previous services composed by the own user. In this work, we go a step further providing a third option: (3) Composed Services created by followed users that can be of interest to a given user. These Composed Services are provided by the Recommendation Algorithm 2 (recommendation of compositions of interest) introduced in Section 5.2.

We have also extended this tool in order to maintain the graph that represents the social network's structure and automatically create the edges (connections) that are inferred from the activity of composing services. Thus, note that the creation of a Composed Service through the mobile authoring tool results in the implicit definition in the social graph of an *author* edge between the current user and the newly created service. In addition, if the user selects an existing service to be used as a basis, a $definedFrom$ edge is implicitly created between both services.

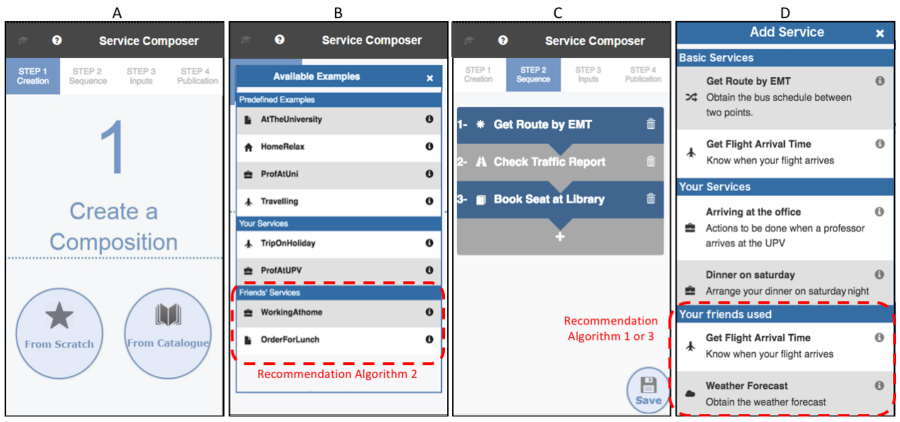


Fig. 2 Creation of a service composition with Recommendations

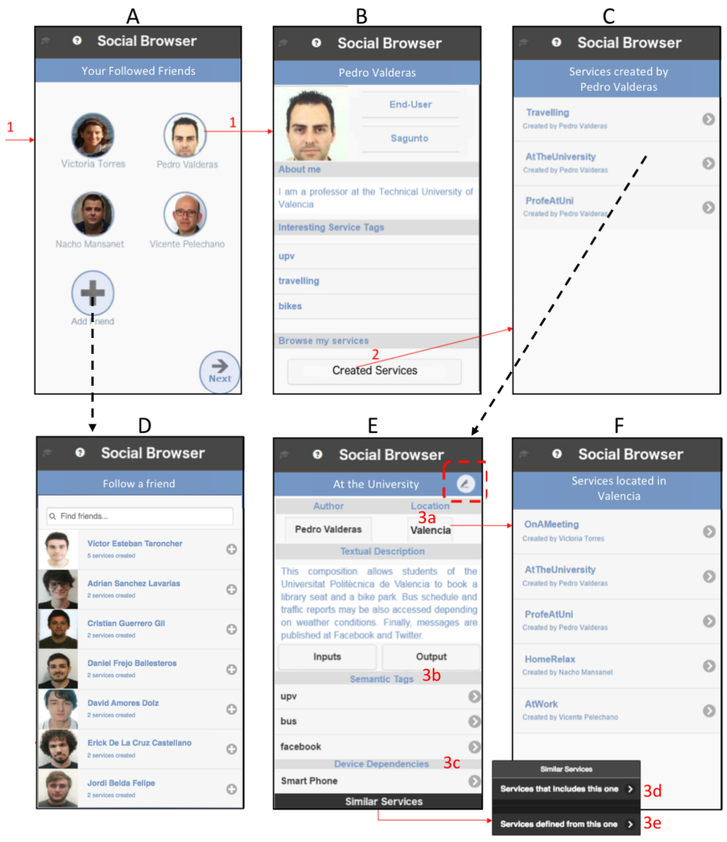
A Composed Service is defined by using the metaphor of "adding an element" to a container. The Composed Service is the main container and users must include the services they want to compose to create the new service. The workflow metaphor is used to represent the sequence of the services included in a Composed Service. Graphically, it is represented by using the List layout (see Figure 2C). The order in which services are displayed (from top to bottom) represents the order in which services must be executed. Each service included in a Composed Service is connected graphically to the next one by a small inverted triangle. This aspect is inspired by the jigsaw metaphor, which defines pieces inserted into others to reinforce the notion of connection or combination of elements.

Users just need to click the '+' button to access the list of available services (see Figure 2D). This list includes: (1) the Basic Services created by developers, (2) the services composed by the own end user; and (3) the services used by followed users in the composition of services. The third option corresponds to the services that are recommended through the implementation of the Recommendation Algorithm 1 (recommendations of a first service for an empty composition) or 3 (recommendation of a next service for a partial composition) introduced in Section 5.2.

Note that each time a service is added to a Composed Service an includes edge is implicitly created in the social graph between both services. In the same way, the ordered set that is associated to the Composed Service is dynamically updated.

6.2 Service Browsing

EUCalipTool has been extended with new user interfaces to browse services according to the possibilities presented in Section 5.1. If we were users of the proposed social network we would have the possibility of accessing the screen in Figure 3A, which shows a list of followed users (browsing option 1, see Section 5.1). If we select a user from this list, we can access its profile (see Figure 3B), which includes the information proposed in Section 4. From a user profile, we can browse the services the user has created (browsing option 2), which is shown in Figure 3C. In the same way, if we click on the plus button from the screen in Figure 3A we access the screen in Figure 3D that allows us to follow new users.



I am a professor at the Technical University of Valencia

This composition allows students of the Universitat Politècnica de Valencia to book a library seat and a bike park. Bus schedule and traffic reports may be also accessed depending on weather conditions. Finally, messages are published at Facebook and Twitter.

Fig. 3 Browsing followed users and their associated services

If we select a service from the list in Figure 3C we access its profile (see Figure 3E). This profile includes all the information proposed in Section 3.2. At the upper side of the screen, we can find a rounded icon next to the service's name. This button is indicating that it is a Composed Service. If we click it we access the authoring tool in order to create a new service taking as a basis the definition of this selected service (see Figure 2B). The rest of the information allows us to browse services from the different options proposed in Section 5.1.

The location ((browsing option 3.a), the semantic tags (browsing option 3.b) and the device dependencies (browsing option 3.c) allow us to access the list of services that are related to this one according to these data. As a representative example, Figure 3F shows the list of services that have a dependency on the same location as the service in Figure 3E. In contrast to the list shown in Figure 3C, this one includes services of several followed users. Finally, note that a button labelled as “Similar Services” is located at the bottom side of the screen Figure 3E in order to access a menu that allows browsing other similar services. In particular, this menu presents two options to access the services that include this one (browsing option 3.d) and the services defined from this one (browsing option 3.e)

6.3 Service Execution

Although the execution of services is out of the scope of this paper, in this subsection we briefly introduce how a service is executed within the EUCalipTool platform. A detailed description of this issue can be found at [46]. Note, however, that we faced how an end user can execute their services. The execution of services created by others users of the social network requires a detailed analysis of all the security issues that may arise as well as how to solve the problem of executing services that depend on a device or a location that only have sense in the context of a specific user. This issue will be considered as further work.

Figure 4 shows some snapshots of the screens that end users interact with when executing services. Figure 4A shows the list of services that are available for execution. Figure 4B shows an intermediate screen that informs about the execution process. Figure 4C shows a screen that requests end users some data required at runtime. Finally, Figure 4D shows the results of the execution, indicating the services that have been executed and the outcome provided by each of them.

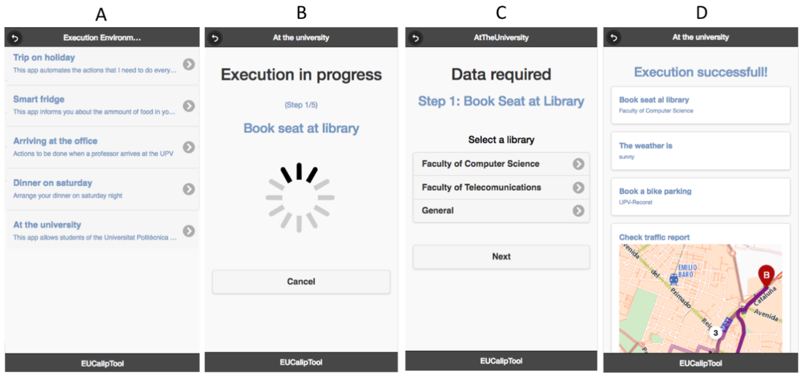


Fig. 4 Execution of a Service

7 Evaluation

In this section, we present an experiment done with some students of our University, which were proposed to use the social network for one month.

In particular, we asked 177 students to participate in the experiment. The experiment consisted of: (1) completing a pre-test questionnaire to collect some data on students' knowledge and skills, (2) using the proposed social network for a month and (3) completing a post-test questionnaire to know the opinion of the participants about the social network. Next, we introduce the details about the experiment.

Participants. From the 177 students that participated in the experiment, 94 were studying the forth (and last) year from the Degree of Computer Science. The other 83 students were studying the third (out of four) year in the Degree of Audiovisual Communication. 124 were male while the rest were female. With some exception, most of the participants were between 20 and 25 years old.

Apart from this demographic data, we used the pre-test questionnaire in order to know their technological habits. As expected due to the Degree they were studying, all of them had a high experience using smartphones and computers or laptops to browse the web, read e-mails, or use word processors or spreadsheets. In the same way, 96% of them used some social network daily. 48.7% of the participants also indicated that they play games assiduously. All of them had some experience in programming. As we can see, participants had a high technological background and skills. This was not a problem to perform the experiment since we wanted to evaluate the usefulness of the proposed social network. The usability of the composition environment provided by EUCalipTool to end users with little technological background was evaluated in a previous work [46].

One of the most interesting data we wanted to know about participants was their experience on composing services or automating some set of actions. We though the evaluation of participants with this experience would be really interesting to estimate the acceptation of the proposed social network. Therefore, we included some questions in the pre-test questionnaire that asked if they had ever used a tool to automate tasks, including mobile apps like IFTTT with their applets, or home assistants like Alexa or Google Home with their routines. Around 31% of the participants (55 out 177) answered affirmatively.

Design. To present the experiment to students, we used the online teaching platform of our university as well as face to face classroom lessons.

The experiment was done during a month and it was conducted as follow:

1. We asked students to complete the pre-test questionnaire to collect the data about participants presented above. This was a mandatory task that all of them must to do.
2. We arranged a classroom session to talk to students about the composition of services and present EUCalipTool. We prepared a case study to train

students in the use of EUCalipTool. We also sent them a video tutorial of the tool in order to be further revised if they needed it.
Afterwards, we asked them to use the tool freely at any time they wanted for a month. This was not a mandatory task. Not using it was an accepted option. We just proposed them to identify any situation in their daily life in which a set of tasks could be automated, and describe them with EUCalipTool. We reminded them that they had two options to do that, creating a Composed Service from scratch or reusing an existing one created by some followed friend. When composing services, we logged the selections done by students in order to know if the recommendations done by EUCalipTool were chosen. If students needed some service that was not available in the platform they could ask us for registering it. Once a week, we sent them an email remanding that they had EUCalipTool available to create service composition when they needed. We also introduced some remainder about EUCalipTool during the classroom lessons of the month that last the experiment.

3. Finally, those participants who used EUCalipTool to create some Composed Service were requested to complete a post-test questionnaire after the given month went by. We prepared a questionnaire with questions that focused on the experience of using the social network to compose services. It includes two type of questions: some that were used with a seven-part Likert scale from 1 (lowest score) to 7 (highest score) points to evaluate them; and others that allow participants to introduce a free answer that justify some of the previous evaluations. In particular, we asked participants about: (1) using a similar environment in a real scenario, (2) using services composed by others when creating theirs, (3) share their services with others; (4) reasons for sending following requests; and (5) the usefulness of the recommendation algorithms. Furthermore, we also included some questions to apply the Microsoft Product Reaction Cards [47] in order to evaluate the end-user satisfaction level. This method consists of providing participants with a list of words and asking them to choose the words that they would use to describe a product. The list includes positive words like 'Useful' and 'Engaging', but also negative words, such as 'Frustrating' and 'Ineffective'.

Students were provided with an execution simulator that allowed participants to check the execution of services. This simulator is based on the user interface shown in Section 6.3. This simulator can be also used to execute the services of the followed users providing the possibility of testing these services before reusing them.

Results. We analyzed the results of our experiment from two perspectives. On the one hand, we studied the participation obtained during the month that the experiment lasted. On the other hand, we analyzed the feedback provided by students through the post-test questionnaire.

Regarding the participation, close to 57.6% of the students (102 out of 177) created, at least, one Composed Service. From these students (see left side in

Figure 5), 44.1% (45 out of 102) created only one Composed Service; around 30% (31 students) created two; 18.6% (19 students) created three; and finally, only a 6.8% (7 students) created four or more Composed Services.

In absolute values, 193 Composed Services where created. Considering the total amount of potential users (i.e. all the 177 students), we obtained, in average, a little more of 1 Composed Service per user. In order to know if these results can be considered good or bad from a participation point of view, we compared them with a well-established platform of task automation. In particular, we considered the data presented in [48] about the IFTTT platform, which had, in 2018, 14 million registered consumers (although it was not said how many were active) and 75 million of applets since launched in 2010. In average, this means that each user created around 5 applets during 8 years.

It is true that the comparison between a tool validated in an academic environment like ours and a platform used by millions of real users can be a little difficult due to the multiple contextual variables that need to be considered. For example, IFTTT users were not asked to use the tool (they had the option of using another) in contrast to our students who were explicitly requested to use only EUCalipTool. However, we think that, on average values, it is a good point of reference to evaluate the participation of students in the experiment. Thus, considering that students were not force to create compositions, and that those created responded to a real situation identified by them, we consider that 1 Composed Service per user in a month reflects a good participation level if we compare this value with the 5 applets per user created with IFTTT during 8 years.

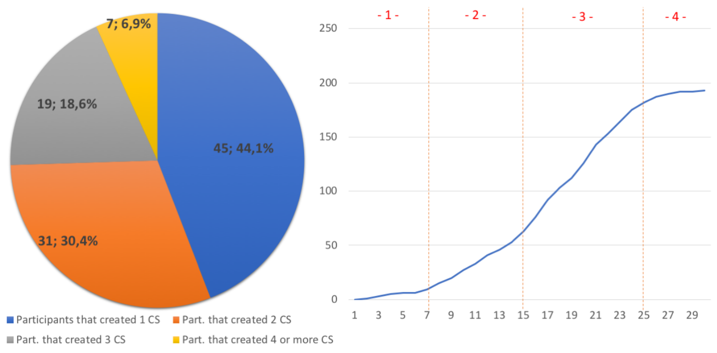


Fig. 5 Participants and the number of services they created (left) and Amount of created services along the whole month (right)

An additional interesting data that we studied was the participation of those students with some experience in the automation of tasks. As commented above, 31% of the students (55 out of 177) indicated to have ever used the capabilities of tools such as IFTTT, Alexa o Google Home to compose an automation of tasks. From these students, close to 85% (46 out of 55) created two or more compositions, which illustrates a considerable interest in the use of EUCalipTool by those who were previously interested in automating some type of behaviour. This same data can be read from another perspective: 53.5% of those participants who created two or more Composed Services (38 out of 71) had some experience in this activity, and 46.5% of them (33 out of 71) used a tool for composing behaviour for the first time. Considering all the students that created some Composed Service (102), the platform provided by EUCalipTool was engaging enough to get 33 students who had never created such a composition before to create two or more Composed Services after using the platform.

From the 193 Composed Services that students created, 41% of them were created from scratch and the other 59% were created by taking an existing service as a basis. This result reinforce the idea of using a social network as a valuable mechanism to create a repository of predefined elements for end users as it is recommended in [34]. The usefulness of taking a Composed Service as a basis to create a new one can be also appreciated in the right side of Figure 5. This figure shows how the total amount of Composed Services evolved along the month that the experiment lasted. We have identified four main periods, which are depicted by dashed red lines and explained next. We can see how the possibility of taking an existing service as a basis encouraged end users to create new ones.

1. Period 1 corresponds to the first week after we presented the experiment to students. As we can see, there was little activity in the creation of Composed Service by students. In average, 1.28 Composed Services per day were created.
2. Period 2 corresponds to the second week. After some reminders about the possibility of using EUCalipTool, students started to use the platform more intensively. In average, 6.28 Composed Service per day were created. An interesting data is that 92% of the Composed Services that were created during the first and second weeks of the experiment were created from scratch. In this sense, we concluded that students identified a scenario were some automation of tasks can be defined, searched for one that can support this scenario, didn't find any, and created a new one from scratch.
3. Period 3 corresponds to the next 10 days of the experiment. During these days the amount of created Composed Services increases significantly. In average, 11.72 Composed Service per day were created. In this case, however, 86% of them were created by taking an existing one as a basis. We concluded that students followed identifying scenarios were some automation of tasks can be defined. However, after the first two weeks of the experiment, there was a more extensive catalogue of existing Composed Services that can be used by students to create a new one that supports their need.
4. Period 4 corresponds to the last 5 days of the experiment. In this case, the amount of created Composed Services per day decreased. On average, 2.2 Composed Service per day were created. We concluded that students were

not able to identify more scenarios that can be supported by a Composed Service.

As far as the situations for which Composed Services were created, we detected that 35% of them were generally based on a smart home; 25% of them were based on smart cities; another 25% were based on the integration of mobile devices with social networks; and finally, a 15% supported some sport activity.

Regarding the recommendations done by EUCalipTool, we suffered a well-known problem in recommendation systems called the cold start problem during periods 1 and 2. This problem appears when a system tries to make recommendations and its knowledge base is empty. In our case, the problem appeared because students had created too few $follower$ connections for the recommendation algorithms to provide valuable recommendations. This problem may be improved by initially considering the actions done by all the users, not only the followed ones. However, an exhaustive analysis of how solving this problem is needed. Note also that this evaluation focused on studying the usefulness of the proposed algorithms. An individual analysis of the functions that are used by these algorithms (presented in Section 5.2) is out of the scope of this paper. In [45], we studied the correctness and completeness of these functions through a set of JUnit tests. Also, we did an experiment to evaluate the performance of their implementation and we obtained an acceptable execution time to achieve a good user experience when using EUCalipTool.

Considering the Composed Services created in periods 3 and 4 (see Figure 6, solid bars), around 68% of those that were created by taking another as a basis were selected from the options recommended by EUCalipTool (Algorithm 2). The first services recommended by EUCalipTool (Algorithm 1) were selected almost 55% of times, and the recommendations for the next services (Algorithm 3) were selected 73% of times. These results are reinforced by the answer obtained in the post-test questionnaire, which was completed by the 102 students that created one or more Composed Service. Around 70% found Algorithms 2 and 3 useful (they cored these questions between 5 and 7). Algorithm 1 was found useful (cored between 5 and 7) by a 61% of the participants. If we consider only those participants that had some experience on using tools for automating behaviour (55 students, see Figure 6, dashed bars) the results were the following: algorithm 1 was found useful by 69% of them, algorithm 2 by 76%, and algorithm 3 by 73%.

These results indicate that the proposed algorithms help end users in the composition of services. However, they must be improved to obtain greater acceptance from users. A way these recommendations can be improved is by asking end users to describe the Composed Service they want to create in natural language, before starting to create it. We can match this description with the semantic keywords associated to the service recommended by algorithms in order to highlight those that better fit the Composed Service end users want to create. To do so, natural language processing techniques should be integrated with EUCalipTool.

Following with the post-test questionnaire (see the left side of Figure 6), around 80% of them stated (by coring this question between 5 and 7) that they would use a tool like EUCalipTool if it was available in a market such as Google Play or Apple Store. Regarding the questions about using the services created by others, around 74% of the participants would feel comfortable both sharing their services with others as and reusing services created by their followed users. Around a 69% of the participants indicated that the social network helped them to discover services (coring this question between 5 and 7). If we focus only on those participants that had some experience on using tools for automating behaviour (55 students, see right side of Figure 6), the results were the following: around 98% of them scored between 5 and 7 the question that stated that they would use a tool like EUCalipTool if it was available in a market. A 80% of the participants would feel comfortable sharing their services with others, and close to a 88% of the participants would feel comfortable reusing services created by their followed users. Finally, around 82% of the participants scored between 5 and 7 the question that indicates that the social network helped them to discover services.

These results allow us to consider that, from a global perspective, the capabilities implemented in EUCalipTool from the proposed social network helped participants in the activity of creating services.

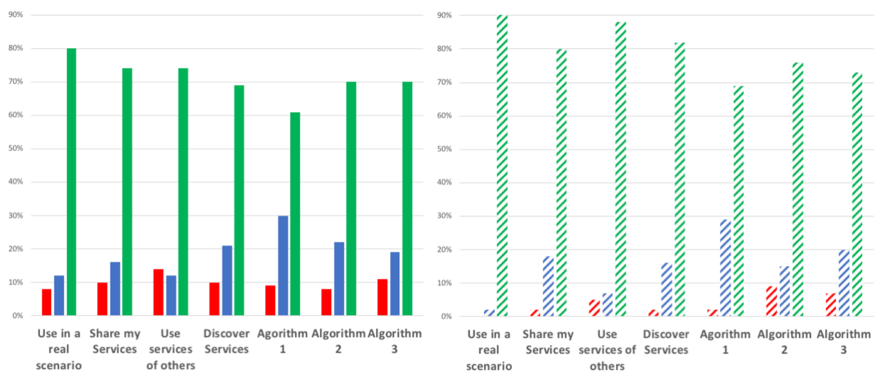


Fig. 6 Results of the post-test questionnaire: all participants (left, solid bars) and only those with experience in automating behaviour (right, dashed bars)

An interesting issue commented by some participants was the possibility of sharing only some Composed Service to do only specific actions with them. Currently, an end user can access all the Composed Services created by their followed users to both take it as a basis to create a new one and include it in the definition of another service. The proposed idea consists in allowing end users to indicate if a Composed Service is shared or not (it can be defined for private use), and if shared, which actions can do their followers with it. We found this idea really interesting and we are currently working on an extension of the social network to include this possibility.

Participants also detected a problem when they reused a Composed Service that was published by a followed user and this service depended on a specific location or device. The actions that depend on these locations or devices cannot be directly reused and needed to be adapted. To improve this problem we are working on defining abstract services (e.g. order a pizza) that can be instantiated when they are included by a specific end user (e.g. selecting the Italian restaurant with home service that is close to the current end-user location).

Regarding the connections among end users, a total of 218 following connections were created by the 102 students that create some composition. According to Figure 7(see left side), a 9.8% of these students (10 out of 102) created one following connection; around a 32.3% (33 students) created two; around a 42.1% (19 students) created three; and finally, around a 17.7% (16 students) created four or more following connections. In average value, we obtained around 2 following connections per user, which represent a 1.9% of the active users (i.e. 102). If we compare these results with a well-established social platform like Twitter, we can consider them quite good. According to [49], Twitter had in 2016 around 95 millions of active users (those who had tweeted at least once in the last 6 months) and the average user had 707 followers, which represent 0.0007% of the active users. This percentage is quite lower than the one obtained in our experiment.

The right side of Figure 7 shows the reasons that participants indicated to send a following request to another user. Two main answers were given: (1) the user was a friend of mine (almost 48% of the times that a student sent a following request), and (2) the user had services I was interested in (40.6% of the times). Note that, initially, EUCAlipTool only allowed a user to know the services other user had created if it was a followed user. However, during the experiment, some students asked us to know the services that each user had created before to decide whether or not to send it a following request. Thus, we updated the EUCalipTool platform in order to show a preliminary list of the service that each user has created. In this sense, the use of a social platform seems to be an interesting mechanism to create trustworthiness and encourage end users to reuse services created by others to define their ones.

As far as the end-user satisfaction level with the whole platform, which was evaluated through the use of Microsoft Reaction Cards, Figure 8 shows the words that got 10 or more occurrences. As we can see, “easy to use”, ”friendly” and “intuitive” were the participant’s most selected keywords. These keywords show that subjects were pleased with the functions provided by the social platform and the way of using them.

Conclusions. We can state that sharing services and browsing those published by others through a social structure is natural to most end users since they are accustomed to using social networks. In addition, recommendations done by analysing the social connections help end users in the activity of composing services. Thus, we can consider that the proposed social network is usefulness for end user to compose services. Of course, this is not a closed research work and some issues identified in the evaluation experiment need to

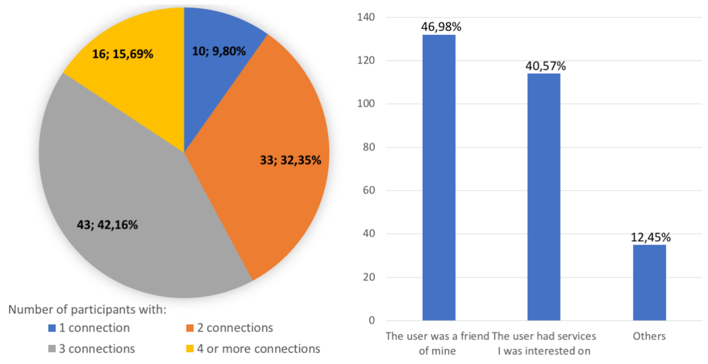


Fig. 7 Participants and the number of connections they created (left) and reasons to create a connection (right)

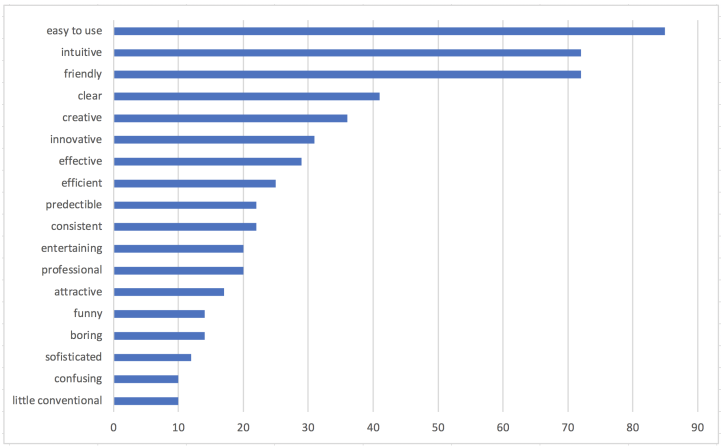


Fig. 8 Satisfaction evaluation with Microsoft React Cards

be considered as further work. However, we feel reinforced in the idea that a social structure that support end users in the composition of services can be a valuable mechanism to help them to become into prosumers of services.

Finally, as a representative example of the social network constructed in the experiment, Figure 9 shows a partial view of its social graph. It includes some fictitious Developers created by us (*Amazon*, *Trip Advisor*, *Twitter*, *Tune in*, and *AEMT*, whose representative nodes are depicted as grey circles labelled with a D). These developers provided several Basic Services (depicted by black squares labelled with a B) that students could compose to create the composed ones (depicted by black squares labelled with a C). The graph also includes two End users (depicted as gray circles labelled with a EU) that represent two students named *Luis* and *Miguel*. *Luis* created a Composed Service named *waking up*. *Miguel* followed *Luis* and created his *waking up* service by taking as a base the one created by *Luis*. In addition, *Miguel* created a Composed Service to support *dinner at home*.

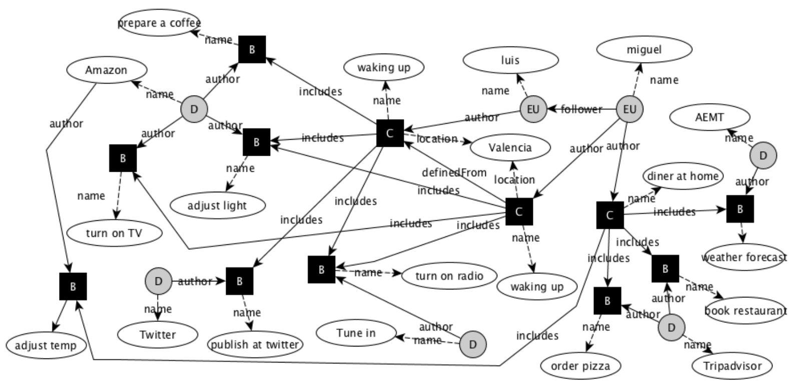


Fig. 9 Partial view of the social graph created in the experiment

Threats of validity. This experiment was performed in an academic environment. We cannot assure the same results if the experiment is done in a real environment. The main reasons to do this statement are:

1. Students may have felt comfortable using and reusing services created by others because they all belong to the same group (students of the same Degree). Although unconsciously, a confidence relationship may exist among them. We cannot ensure the same behaviour when users have the possibility of reusing a service created by a totally strange user.
2. Our main goal was to have a set of users that can use our social network in a scenario close to a real one. However, note that a real scenario implies having users with a real need of creating a Composed Service. We cannot discard that some of them may have created services just for satisfying professor's desires.

8 Beyond a matter of sharing services among end users. Discussion

In this work, we focused on improving the area of end-user development with a social platform to help end users to create their services and share them with others.

Segal [34] recommends that end users can access a library of predefined components to use them as starting point. However, the same author demonstrated in [50] that these repositories of components are very difficult to maintain by developers in real contexts. The proposed social platform allows end users to access the services created by developers and other end users, and start to compose new services by using these "predefined examples" as a basis. In this sense, the social network becomes a continuously increasing repository of components that is maintained and improved by both end users and developers. Also, the proposed social network allows end users to find services by browsing a set user and services profiles, which are notions that are familiar for them. Social networks are currently one of the most used apps [8] and, as we validated in the previous section, the use of a social structure is familiar to end users, making the task of finding services easier than current service repositories.

However, the idea of a social network in which end users and developers live together can be generalized to other domains rather than service creation. Even more, it can allow broader collaboration between end users and developers than only a shared creation of software artefacts. Regarding this last issue, it is interesting to remind that End-User Development (EUD) focuses on allowing users without programming skills become into producers of software artefacts. From a Software Engineering perspective, EUD mainly focuses on the implementation stage. Other approaches advocate for the participation of end users in other phases of the development process. We can consider, among others, areas such as End-User Software Engineering [51], Participatory Design [52], Meta-Design [53], and so on.

In this sense, we think that a social platform for end users and developers can be a valuable mechanism to allow them to collaborate not only in the implementation of software artefacts but also in other software development activities that focus on improving the quality of the developed artefacts. For instance, a user can publish an initial product description to allow other users of the social network to participate in the elicitation and management of requirements; or a software product can be published to be tested and debugged by others. This type of social platforms can support software development by and for the crowd. Note that current crowd computing research is investigating the ability to aggregate and employ human time and talent to develop specific tasks through digital media. We think that social platforms can be a valuable tool to do so in the context of software development. In any case, it requires an exhaustive analysis in order to properly define the social network that is needed for each purpose, determining user profiles, artefacts that can be shared, and the actions that each user profile can done with these artefacts.

Next, we present a general description of the main concepts we think that are required to define a social network for developers and end users target at some software activity. Note that this is only an initial effort of generalizing our work. Further analysis and research are needed, but they are out of the scope of this paper. Figure 10 shows the concepts that we propose to characterize a social network target at software development activities. Each User of the social network is either a Professional, which is a user with skills in some development phase, or an End user, which lacks from these skills although have knowledge of the domain. Users are linked to each other through a Connection, which may be characterized as symmetric or asymmetric if we consider current trends in social networks. With a Symmetric connection, if a user A is connected to a user B it means that user B is also connected to user

A (e.g. friendship relationship supported by Facebook). With an Asymmetric connection, the fact that a user A is connected to a user B doesn't mean that the user B is also connected to the user A (e.g. the *follower* connection used in this work).

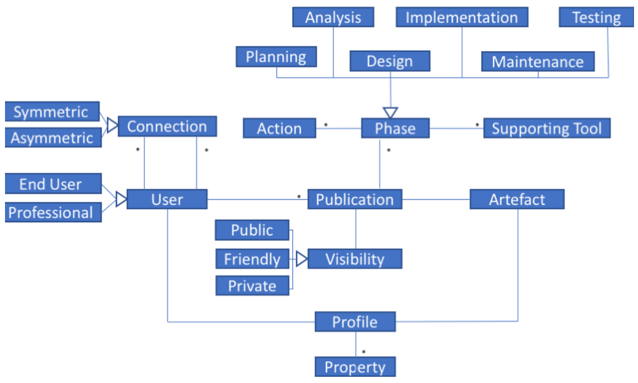


Fig. 10 Preliminary generalization effort

Users can create Publications in order to share some Artefact with other users. We will need to precisely specify the artefact or artefacts that can be shared. Each user and artefact must have a profile made up of a set of properties which should be defined according to the software domain and development activities that must be supported by the network. These properties will be a valuable mechanism to browse artifacts.

Each publication can have a Visibility in order to indicate which users can access it (and then, the published artefact). Considering current social networks and some conclusions obtained from the experiment presented above, we can though of three types of visibility: Public, which indicates every user in the network can access the published artefact; Friendly, which indicates that the published artefact can only be accessed by connected users; and Private, which indicates that the artefact can only be accessed by the author.

Each publication is associated with one or more phases of the development process which should be associated with the list of Actions that can be done with the published artifacts in each phase. For instance, if we define a social network that supports the phases of implementation and testing of a software product, we should define the actions that users can do with artefacts published in these phases. Finally, note that the artefacts published in each phase may be defined with tools specifically created for this purpose. Thus, the proposed social platform should be integrated with these tools in order to provide the proper working flow.

Considering the social network proposed in this paper, the artefact that is shared among users of the social network is a Service. Each Service is published in the context of the implementation phase and has friendly visibility. Users are connected through asymmetric connections. The tool that supports the implementation of services is the EUCAlipTool authoring environment which is integrated with the new developed social platform. The actions that users can do with the published artefacts are two: (1) taking it as a basis to create a new one and (2) including it in the creation of another. If our social network would have supported the testing of services, we would have to consider which artefacts may be published for the testing phase (e.g. services to test and testing reports), the actions that end users may do with them (e.g. execute a service, add a report, validate a report, etc.), and the integration of the social network with some execution and testing tool.

In order to have another representative example, let's consider, for instance, the domain of Web Augmentation [54]. In this area, end users usually add, alter, or remove features of a web application interface by creating scripts that run on their web browser. A social network focused on the creation of these scripts would help end users to access scripts that may solve their needs, and which would have been created by professional scripters or other end users like them. This social network may be characterized, for instance, as follows: the artefact to be shared is a Web Augmentation Script, which is published for the implementation and testing phases. The actions that each user can do with a published artefact are two: (1) the installation of the script in order to be tested, and (2) the adaptation of a script in order to create a new one. The social network should be tightly integrated with some web browser and authoring tool in order to allow the deployment and edition of scripts. If additionally, we also want to consider the analysis phase, as approaches like CrowdMock [55] do, the artefact to be published can be, for instance, user histories that describe the new requirements to be supported by a Web Augmentation Script. The actions that other users can do with these histories are: (1) to adapt or extend them in order to create new histories, (2) to be associated to some existing script that solves it, or (3) to be supported with the creation and publication of a new script.

9 Conclusions and further work

In this work, we have presented a social network to support end users in the composition of services. We want to encourage end users to become into producers of services and contribute to improving the research of end-user service composition. We have provided end users with a solution not only to build service compositions in an intuitive way but also to share the acquired knowledge among them.

We have characterized the proposed social network and have defined it in a semi-formal way by using graph theory. We have also analysed how social connections can be exploited to (1) facilitate end users to discover services through browsing these connections, and (2) recommend services to end users during the composition activity.

As proof of concept, we have extended EUCalipTool with social support. This tool supports end users in the composition of services by using mobile devices. We have extended it with social capabilities. This social version of EUCalipTool has been evaluated with students of our university for a whole month. The results of this experiment reinforced the idea that a social structure can be useful for end users in the activity of creating services on their own.

However, there are still some challenges that need to be faced as further work. For instance, instead of using a simulator to test services, it should be interesting to allow end users to directly execute the services created by others. This also needs to face the security aspect of the social network by providing the possibility of defining grants over the shared Composed Services or integrating security frameworks with social support such as Anahita or Elgg to properly manage personal data. Furthermore, we are studying the possibility of including natural language descriptions done by end users in order to improve the recommendation process.

Another interesting issue that we want to face is to include the usage dimension in the recommendations done by our social network. The main idea is to recommend end users with services to be consumed depending on the consumption done by their followed users in the same context (i.e. same time, location, device, and so on). Furthermore, we are investigating how to apply a previous research work focused on considerate computing [56] studied how to achieve a considerate interaction with users (i.e. disturbing them as less as possible) when executing services. We plan to apply this work to the presented architecture to create social notifications in a considerate way. The main idea is to propose a conceptual framework that allows us to characterize both: (1) the notifications that a user can receive from a social network like the proposed in this work, and (2) the communication resources that must be used to deliver each notification to end users depending on their current situation.

Finally, note that, although implemented in the context of EUCalipTool, the definition of the social structure can be applied to other end-user environments for composing services to provide them with social support. In the same way, we have presented a preliminary generalization effort to describe the main concepts that characterizes a social network target at any software development activity and any domain.

Acknowledgement

This work has been developed with the financial support of the Spanish State Research Agency under the project TIN2017-84094-R and co-financed with ERDF

References

1. J. Yu, Q. Z. Sheng, J. Han, Y. Wu, and C. Liu, “A semantically enhanced service repository for user-centric service discovery and management,” *Data & Knowledge En-*

gineering, vol. 72, pp. 202–218, 2012.
2. F. Daniel, F. Casati, B. Benatallah, and M.-C. Shan, "Hosted universal composition: Models, languages and infrastructure in mashart," in *International Conference on Conceptual Modeling*, pp. 428–443, Springer, 2009.
3. J. Danado and F. Paternò, "Puzzle: A mobile application development environment using a jigsaw metaphor," *Journal of Visual Languages & Computing*, vol. 25, no. 4, pp. 297–315, 2014.
4. S. Aghaee and C. Pautasso, "End-user development of mashups with naturalmash," *Journal of Visual Languages & Computing*, vol. 25, no. 4, pp. 414–432, 2014.
5. P. Valderas, V. Torres, I. Mansanet, and V. Pelechano, "A mobile-based solution for supporting end-users in the composition of services," *Multimedia Tools and Applications*, vol. 76, no. 15, pp. 16315–16345, 2017.
6. E. Al-Masri and Q. H. Mahmoud, "Wsce: A crawler engine for large-scale discovery of web services," in *IEEE International Conference on Web Services (ICWS 2007)*, pp. 1104–1111, IEEE, 2007.
7. A. Santanche, S. Nath, J. Liu, B. Priyantha, and F. Zhao, "Senseweb: Browsing the physical world in real time," *Demo Abstract, ACM/IEEE IPSN06, Nashville, TN*, pp. 1–2, 2006.
8. J. Nielsen, "Tops of 2015: Digital. media and entertainment." `http://www.nielsen.com/us/en/insights/news/2015/tops-of-2015-digital.html`, 2015. [Last time accessed: January 2019].
9. IFTTT, "If this then that." `https://ifttt.com/`, 2015. [Last time accessed: January 2019].
10. Dlvr.it, "Social media auto posting & scheduling tool." `https://dlvrit.com/`, 2018. [Last time accessed: January 2020].
11. Zapier, "Connect your apps and automate workflows." `https://zapier.com/`, 2018. [Last time accessed: January 2019].
12. Node-RED, "Flow-based programming for the internet of things." `https://nodered.org/`, 2017. [Last time accessed: January 2019].
13. A. Maaradji, H. Hacid, J. Daigremont, and N. Crespi, "Towards a social network based approach for services composition," in *2010 IEEE International Conference on Communications*, pp. 1–5, IEEE, 2010.
14. J. Soriano, D. Lizcano, J. J. Hierro, M. Reyes, C. Schroth, and T. Janner, "Enhancing user-service interaction through a global user-centric approach to soa," in *Fourth International Conference on Networking and Services (icns 2008)*, pp. 194–203, IEEE, 2008.
15. P. Jiang, K. Ding, and J. Leng, "Towards a cyber-physical-social-connected and service-oriented manufacturing paradigm: Social manufacturing," *Manufacturing Letters*, vol. 7, pp. 15–21, 2016.
16. D. A. Tamburri, P. Lago, and H. v. Vliet, "Service networks for development communities," in *Proceedings of the 2013 International Conference on Software Engineering*, pp. 1253–1256, IEEE Press, 2013.
17. Z. Maamar, L. K. Wives, Y. Badr, and S. Elnaffar, "Even web services can socialize: A new service-oriented social networking model," in *2009 International Conference on Intelligent Networking and Collaborative Systems*, pp. 24–30, IEEE, 2009.
18. S. Yu and C. J. Woodard, "Innovation in the programmable web: Characterizing the mashup ecosystem," in *International Conference on Service-Oriented Computing*, pp. 136–147, Springer, 2008.
19. W. Chen, I. Paik, and P. C. Hung, "Constructing a global social service network for better quality of web service discovery," *IEEE transactions on services computing*, vol. 8, no. 2, pp. 284–298, 2013.
20. M. Ren, L. Ren, and H. Jain, "Manufacturing service composition model based on synergy effect: A social network analysis approach," *Applied Soft Computing*, vol. 70, pp. 288–300, 2018.
21. M. Kranz, L. Roalter, and F. Michahelles, "Things that twitter: social networks and the internet of things," in *What can the Internet of Things do for the Citizen (CIoT) Workshop at The Eighth International Conference on Pervasive Computing (Pervasive 2010)*, pp. 1–10, 2010.

22. J. Bleecker, “A manifesto for networked objects—cohabiting with pigeons, arphids and aibos in the internet of things,” in *Proc. of the 13th International Conference on Human–Computer Interaction with Mobile Devices and Services, MobileHCI*, pp. 1–17, 2006.
23. L. Atzori, A. Iera, and G. Morabito, “Siot: Giving a social structure to the internet of things,” *IEEE communications letters*, vol. 15, no. 11, pp. 1193–1195, 2011.
24. D. Guinard, M. Fischer, and V. Trifa, “Sharing using social networks in a composable web of things.,” in *PerCom Workshops*, pp. 702–707, 2010.
25. M. Meissa, S. BENHARZALLAH, and L. KAHLOUL, “Service composition based on the social relations in the internet of things,” in *The 18th International Arab Conference on Information Technology (ACIT'2017)*, 2017.
26. S. Wang, A. Zhou, M. Yang, L. Sun, C.-H. Hsu, *et al.*, “Service composition in cyber-physical-social systems,” *IEEE Transactions on Emerging Topics in Computing*, 2017.
27. C. Reuter, M.-A. Kaufhold, and T. Ludwig, “End-user development and social big data–towards tailorable situation assessment with social media,” in *New Perspectives in End-User Development*, pp. 307–332, Springer, 2017.
28. D. Massa and L. Spano, “Facemashup: An end-user development tool for social network data,” *Future Internet*, vol. 8, no. 2, p. 10, 2016.
29. D. M. Boyd and N. B. Ellison, “Social network sites: Definition, history, and scholarship,” *Journal of computer-mediated Communication*, vol. 13, no. 1, pp. 210–230, 2007.
30. P. C. Hung, H. Li, and J.-J. Jeng, “Ws-negotiation: an overview of research issues,” in *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, pp. 10–pp, IEEE, 2004.
31. Z. Ding, L. Xiao, and J. Hu, “Performance analysis of service composition using ordinary differential equations,” in *2008 12th IEEE International Workshop on Future Trends of Distributed Computing Systems*, pp. 30–36, IEEE, 2008.
32. N. Milanovic and M. Malek, “Current solutions for web service composition,” *IEEE Internet Computing*, vol. 8, no. 6, pp. 51–59, 2004.
33. H. Lieberman, F. Paternò, M. Klann, and V. Wulf, “End-user development: An emerging paradigm,” in *End user development*, pp. 1–8, Springer, 2006.
34. J. Segal, “Two principles of end-user software engineering research,” *ACM SIGSOFT software engineering notes*, vol. 30, no. 4, pp. 1–5, 2005.
35. Workflow.is, “Workflow. spend less taps, get more done.” `https://workflow.is/`, 2018. [Last time accessed: January 2019].
36. D. Steinbock, *The mobile revolution: The making of mobile services worldwide*. Kogan Page Publishers, 2005.
37. D. Snoonian, “Smart buildings,” *IEEE spectrum*, vol. 40, no. 8, pp. 18–23, 2003.
38. A. K. Milicevic, A. Nanopoulos, and M. Ivanovic, “Social tagging in recommender systems: a survey of the state-of-the-art and possible extensions,” *Artificial Intelligence Review*, vol. 33, no. 3, pp. 187–209, 2010.
39. V. Ermagan and I. H. Krüger, “A uml2 profile for service modeling,” in *International Conference on Model Driven Engineering Languages and Systems*, pp. 360–374, Springer, 2007.
40. R. Amir and A. Zeid, “A uml profile for service oriented architectures,” in *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, pp. 192–193, ACM, 2004.
41. M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, “Semantic matching of web services capabilities,” in *The Semantic Web - ISWC 2002*, (Berlin, Heidelberg), pp. 333–347, Springer Berlin Heidelberg, 2002.
42. M. Klusch and K. Sycara, “Brokering and matchmaking for coordination of agent societies: A survey,” in *Coordination of Internet Agents*, pp. 197–224, Springer, 2001.
43. H. Ehrig and B. Mahr, *Fundamentals of algebraic specification 1: Equations and initial semantics*, vol. 6. Springer Science & Business Media, 2012.
44. J. de Lara, R. Bardohl, H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer, “Attributed graph transformation with node type inheritance,” *Theoretical Computer Science*, vol. 376, no. 3, pp. 139–163, 2007.

45. P. Valderas, V. Torres, and V. Pelechano, "A graph-based definition of a social network for the composition of services by end-users. technical report pros-tr-2019-01," tech. rep., Universitat Politècnica de València, 2019. [Last time accessed: October 2019].
46. P. Valderas, V. Torres, and V. Pelechano, "Towards the composition of services by end-users," *Business & Information Systems Engineering*, pp. 1–17, 2019.
47. J. Benedek and T. Miner, "Measuring desirability: New methods for evaluating desirability in a usability lab setting," *Proceedings of Usability Professionals Association*, vol. 2003, no. 8-12, p. 57, 2002.
48. C. Smith, "Interesting ifttt statistics and facts." `https://expandedramblings.com/index.php/ifttt-statistics-and-facts/`, 2018. [Last time accessed: October 2019].
49. M. Ryan, "The average twitter user now has 707 followers." `https://kickfactory.com/blog/average-twitter-followers-updated-2016/`, 2016. [Last time accessed: January 2020].
50. J. Segal, "The nature of evidence in empirical software engineering," in *Eleventh annual international workshop on software technology and engineering practice*, pp. 40–47, IEEE, 2003.
51. M. Burnett, C. Cook, and G. Rothermel, "End-user software engineering," *Communications of the ACM*, vol. 47, no. 9, pp. 53–58, 2004.
52. D. Schuler and A. Namioka, *Participatory design: Principles and practices.* CRC Press, 1993.
53. G. Fischer, E. Giaccardi, Y. Ye, A. G. Sutcliffe, and N. Mehandjiev, "Meta-design: a manifesto for end-user development," *Communications of the ACM*, vol. 47, no. 9, pp. 33–37, 2004.
54. N. O. Bouvin, "Unifying strategies for web augmentation," in *Proceedings of the tenth ACM Conference on Hypertext and hypermedia: returning to our diverse roots: returning to our diverse roots*, pp. 91–100, Citeseer, 1999.
55. D. Firmenich, S. Firmenich, J. M. Rivero, L. Antonelli, and G. Rossi, "Crowdmock: an approach for defining and evolving web augmentation requirements," *Requirements Engineering*, vol. 23, no. 1, pp. 33–61, 2018.
56. M. Gil, E. Serral, P. Valderas, and V. Pelechano, "Designing for user attention: A method for supporting unobtrusive routine tasks," *Science of Computer Programming*, vol. 78, no. 10, pp. 1987–2008, 2013.