future internet

MDPI

Article

From Service Composition to Mashup Editor: A Multiperspective Taxonomy

Abderrahmane Maaradji [1,*], Hakim Hacid [2] and Assia Soukane [1]

[1] ECE Engineering School, 75015 Paris, France
[2] Technology Innovation Institute, Abu Dhabi 9639, United Arab Emirates
* Correspondence: abderrahmane.maaradji@ece.fr

Abstract: Service-oriented computing has become a popular area of research, with a particular focus on service composition. There have been many developments in this field, such as new techniques for data engineering in service description languages, protocols for publication and discovery, the optimization of service selection and scheduling, and the deployment and monitoring of composed services. However, this diversity of approaches and methodologies can make it challenging to navigate between different proposed solutions and identify research gaps. In order to provide a clearer understanding of this body of work, this paper presents a comprehensive framework for the taxonomy of service composition approaches, methodologies, and tools. This framework proposes a structured view of different perspectives, such as formal, semantic, and automatic approaches, with a particular focus on the end-user's perspective and tools such as Mashups.

Keywords: Web service; service-oriented architecture; service composition; Mashup

Citation: Maaradji, A.; Hacid, H.; Soukane, A. From Service Composition to Mashup Editor: A Multiperspective Taxonomy. *Future Internet* **2023**, *15*, 59. https://doi.org/10.3390/fi15020059

Academic Editors: Stanimir Stoyanov and Lyubka Doukovska

Received: 24 November 2022
Revised: 27 January 2023
Accepted: 28 January 2023
Published: 31 January 2023

1. Introduction and Methodology

Service-oriented computing and particularly service composition have been fairly fruitful research topics. Research contributions made in this area vary according to addressed issues and approaches. These contributions may include data engineering techniques for service description languages, protocols for publications and discovery operations, the optimization of services selection and scheduling, and the deployment and monitoring of composed services. This paper aims to present a structured understating of these contributions with the intent to define a comprehensive classification of different Web service composition approaches. Therefore, this paper reviews and analyzes the concepts and existing works related to the composition of Web services from the perspective of the end-user.

The purpose of this research is to propose a taxonomy framework for Web service composition approaches from a user perspective. A taxonomy framework provides a way to organize and categorize existing approaches to Web service composition from the user perspective. This makes it easier to understand the current state of research in this area and identify gaps in the literature. The framework is based on an analysis of existing academic research that classifies Web service composition and is comprehensive, consistent, and easy to understand. The purpose of this research is to provide a comprehensive understanding of the different approaches to composing Web services and how they relate to each other by providing a clear and consistent taxonomy framework. It helps researchers, practitioners, and developers choose the Web service composition approach that best suits their particular needs.

In this regard, we used a variety of methods to select relevant literature for study, including (i) searching and identifying relevant articles and papers containing specific keywords using online databases such as Google Scholar, IEEE Xplore, and ACM Digital Library; (ii) checking the references cited in existing articles and works related to Web service

composition; (iii) identifying works written by prominent researchers; and (iv) searching for articles related to specific conferences, journals, and workshop proceedings. Once relevant literature was identified, we analyzed existing research to understand the current state of research in the field, and identify gaps in the literature, particularly the need for a structured understanding of service composition approaches.

We first acknowledge that there are some interesting existing works on Web service composition classification. They typically propose a classification framework for Web service composition approaches based on an analysis of existing literature and existing approaches. They aim to provide a comprehensive and consistent way to categorize and understand different Web service composition approaches. While some papers classify Web service composition approaches based on the composition techniques used, such as choreography, orchestration, or mediation [1,2], others classify Web service composition approaches based on their maintainability [3], interoperability [4], or specific domain of application, such as multimedia [5]. To the best of our knowledge, the existing classification frameworks that have been proposed in the literature do not consider the end-user perspective as its main classification criteria.

Our main comparison criterion for Web service composition approaches was the end-user perspective. The end-user perspective refers to how easy it is for the end-users to use and understand the Web service composition approach. This includes aspects such as the usability of the user interface, the simplicity of the composition process, and the ease of understanding the results of the composition.

Incorporating the end-user perspective as a comparison criterion can help to ensure that the Web service composition approach is user-friendly and easy to use, which can be beneficial for both the end-users and the developers. For example, an approach that is easy for the end-users to use and understand is more likely to be adopted and used in practice. Additionally, when the end-users can easily understand the results of the composition, they can make better decisions based on the information provided.

It is important to note that the end-user perspective is to be considered, along with other criteria such as software engineering, services performance, and maintainability, as an end-user friendly approach that is not secure or has a poor performance that will not be useful in practice.

We have therefore identified many perspectives of Web service composition. Accordingly, it is possible to address the service composition from the software engineering perspective, which, here, is called *the system's perspective*. It includes semantic, formal, or data engineering techniques and technologies used to address the problem at hand. Another perspective is engendered from the user's point of view. Basically, the user has to define the composition logic in a more or less explicit manner based on the provided tools. In this scope, automatic, manual, or semi-automatic approaches are identified. The rest of the paper details each of these approaches in order to propose an exhaustive taxonomy that would help to better understand research challenges and contributions from different perspectives.

In the remainder of this paper, we first review the concept of Web service composition through the basics of Web services and service-oriented architecture (SOA), and then provide some classifications based on different criteria. Additionally, we present a detailed review of the composition tools of Web services by the end-user, particularly Mashups. In that regard, we analyze the few existing mechanisms of assistance and support to end-users and point out the lack of such features. Finally, we conclude by summarizing the main ideas that emerge from the overall analysis.

2. Related Concepts

2.1. Web Services

Currently, the concept of service is so pervasive that *Science of Services* is now established as a standalone paradigm [6]. This domain combines the understanding of organizations (enterprises or institutions) and humans with business and technological

sciences. In this regard, a service is defined as *"Any act or performance that one party can offer to another that is essentially intangible"* [7], in contrast with the physical industry (manufacturing and agriculture). More widely, Zeitham et al. [8] state that *"Services are deeds, Processes, and Performance"*. As a synthesis of existing definitions, we propose the following definition:

Definition 1. *"A service is an intangible provision, composable, expressed in a perceptible manner, which, in a predetermined operating condition, is a source of value for the consumer and the supplier (service provider)".*

This concept is much more prevalent in the IT world, where people speak of Web services. Web services are platform-independent software, available in distributed environments such as the Internet. They are mostly used in enterprise contexts for application integration and streamlining B2B software, where they enable developing applications by assembling existing Web services that translate the service-oriented architecture (SOA) philosophy [9]. Indeed, Web services are the most significant achievement of the SOA, in which applications are self-descriptive and low-coupled modules. They are defined by a set of standards that allow us to describe software interfaces and access functions on a network using XML messages [10]. Web-service-related underlying standards and technologies (such as WSDL and UDDI) are exposed in Section 4.1.2. Basically, the World Wide Web Consortium (W3C) (http://www.w3.org/TR/ws-gloss/, accessed on 7 March 2020) has defined a Web service as the following:

Definition 2. *"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). The Web service can be invoked through what is commonly referred to as API (application programming interface) using SOAP messages (typically conveyed using HTTP with an XML serialization) or other Web-related standards such as REST" (cf. Section 4.1.2).*

The next section introduces the SOA concept that relies on Web services as the basic building blocks in a structured architecture.

2.2. Service-Oriented Computing

The reference architecture SOA is conceptually derived from the service-oriented computing (SOC) paradigm [11]. The SOC paradigm advocates the use of the concept of a service (not just Web services) as a building block in any information system project. This paradigm found its dedication in integrating enterprise applications due to business needs (merge, acquisition, consolidation, outsourcing, etc.), which definitely replicates the concept of business services within the IT world. Going beyond the technological and compatibility constraints, this paradigm aims to surpass the silo-based information system model towards a systemic (holistic) model where inter-system transactions happen seamlessly. Through this paradigm, one (organization) can offer, find, use, and compose services according to its own needs and business requirements. Figure 1 shows an example of enterprise "A" outsourcing two services from enterprise "B" thanks to the SOC paradigm. The SOC concept allows for supporting several research issues, including the composition of services.

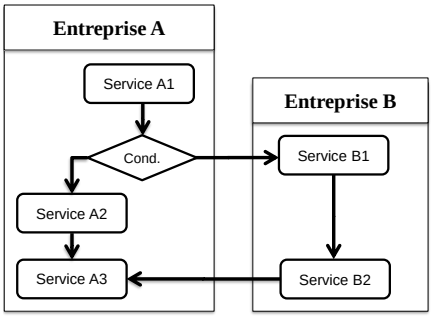

Figure 1. Illustration of SOC paradigm through outsourcing example.

2.3. Service-Oriented Architecture

SOA is one of the most successful examples of the service-oriented computing paradigm. It offers a straightforward model for developing and deploying applications using Web standards. This architecture utilizes Web services as fundamental building blocks. It establishes a modular software architecture in an open information system, where each business function is represented by a basic service. This architecture enables the representation of an organization's business processes as services and the integration of these services into workflows that enable more complex business processes to be executed. Through a layered view, Figure 2 illustrates how business processes are represented through the system based on basic services in an SOA.

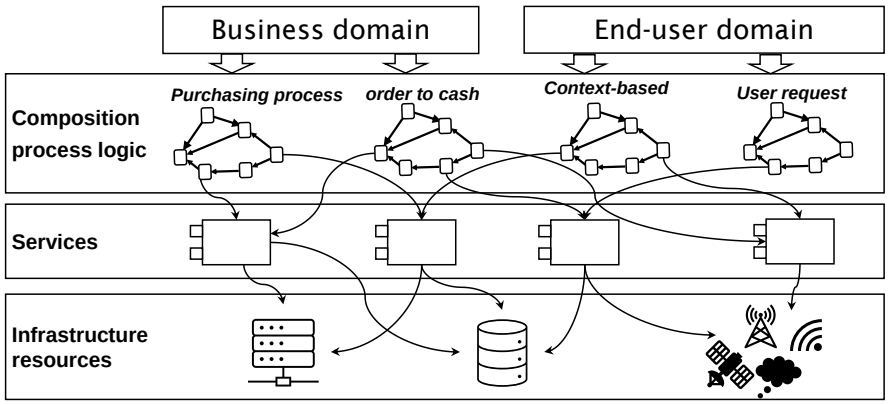

Figure 2. A layered view of SOA concept.

The SOA reference architecture was first introduced in [12], and then was adopted and integrated by many standardization bodies such as OASIS, OMG, The Open Group (http://www.opengroup.org/onlinepubs/7699909399/toc.pdf, accessed on 10 July 2020). It relies on three main actors as illustrated in Figure 3. As mentioned before, the basic concept is a Web service that represents a function encapsulated in a component. This component can then be invoked using a query with one or more input parameters and providing one or more outputs. In order to promote reusability and interoperability, each service should ideally be autonomous and not dependent on other services. Service-oriented architecture's main actors are:

- *The service provider* (or a third party mandated by them) is in charge of the service creation, deployment, description, and then publication through the repository (registry) publication interface.
- *The service repository* hosts the description of services that have been published by the service provider and offers the possibility for clients to seek a specific service among those available and to access service descriptions.
- *The client* (service consumer) should be able to look for services described in the service repository and select those of interest to them. Based on a service description, a client should be able to invoke this specific service hosted by the service provider.

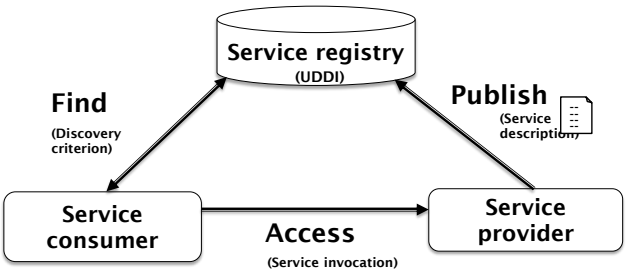

Figure 3. SOA basic architecture.

Then, SOA defines a set of operations and roles as follows (those defined operations and roles are based on standards that are described in Section 4.1.2):

- The description of the service consists of enumerating the input parameters of the service as well as the output parameters (type of data). The primary format for describing services is WSDL (Web Services Description Language) standardized by W3C.
- The service publication is to publish in a registry (or repository) services available to clients (service consumers).
- Service discovery includes the ability to search for a service among those that have been published. The primary standard used is UDDI (Universal Description Discovery and Integration), standardized by OASIS.
- The invocation consists of the customer query (connection) and interaction with the service. The main protocols used for the invocation of services are SOAP (Simple Object Access Protocol) and REST (Representational State Transfer), presented in Section 4.1.2.

This section presented the basic concepts of service systems, such as the notion of service and the SOA reference architecture. These concepts are at the base of different standards and research issues. The following section describes and analyzes related existing research from the service composition point of view, which is the main research topic of our work.

3. Service Composition

Creating value-added services by reusing existing ones, which is also known as service composition, has been a key facet for service delivery both for IT and Telecom worlds [13]. Numerous services need to be produced quickly because of the growing demand of customers. Developing new services from scratch to meet the growing demand would require a large number of programmers, consume a significant amount of resources, and result in a prolonged time-to-market.

In this context, using the services composition approach can offer a good opportunity to fix those issues. The purpose of service composition is the reuse of existing services to create new ones. This optimizes the development cycle and deployment of innovative services. Another important goal of services composition is to provide the ability to customize services according to end-user's preferences. This approach provides end-users with personalized and user-centric services. With the generalization of the Internet, we are witnessing, in recent years, the evolution of the service composition paradigm, initially dedicated to a restricted audience of IT specialists for business application integration (software architects, developers, etc.), toward a broader audience of Web users.

We can already see the two different dimensions that the composition of services includes: organization (EAI) and end-user dimensions. From the EAI perspective, the existing conducted research aims to mainly overcome the technological constraints by defining a range of standards and protocols for information engineering under the banner of SOA principles. This dimension dominates the majority of research works on service composition. The second dimension represents the challenges in integrating the end-user in the process of composition. The end-user dimension of service composition is expanding

and becoming increasingly prominent. This development is driven by end-users' desire for customization, particularly in the Internet context that encourages sharing, creating, and commenting on content. A new concept, known as Mashups, has emerged on the Internet, and enables end-users to create and share their own services by combining existing services [14].

Next, we review the research challenges that rise in the service composition topic through the main themes of SOA and the two dimensions mentioned before. Figure 4 (based on [15]) represents an extended architecture of SOA in an information system. It schematically illustrates the different levels of SOA within which research contributions have been made. Proposed technologies (standards and protocols) and formalisms for each challenge are detailed in Section 4.

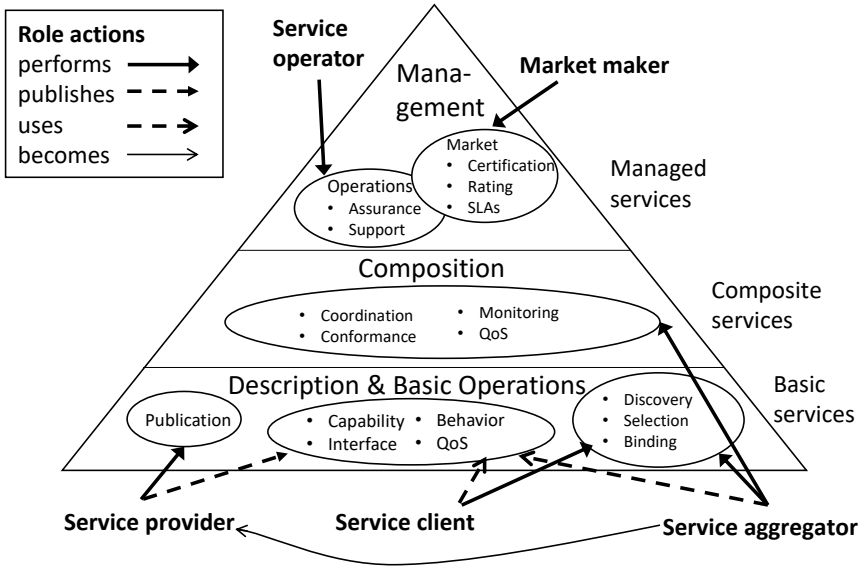

Figure 4. The extended service-oriented architecture emphasizing main research topics (adapted from [11] with permission).

Firstly, service composition involves methods, mechanisms, and tools that allow for the expression of needs, whether at the enterprise level for business specification or at the end-user level. In this regard, numerous formalisms and tools have been proposed. Some of these tools are based on formal models. Business Process Execution Language (BPEL) [16] is by far the reference in the field, but remains unsuitable for our intended end-user (non-developers). Other tools are listed in Section 4.1.2 within their corresponding category.

After the expression of needs, the process of composition consists of selecting the most suitable services and then scheduling (arranging) them in the most appropriate schema in order to fit the logic of the expressed need. Once the composition schema is defined, the resulting composed service needs to be deployed. The deployment could take the form of a choreography or orchestration of services. After the deployment operation, tools and control measures are implemented to monitor the various performance indicators of the deployed service. This operation is called monitoring. The overall composition process is illustrated in Figure 5. In the following, we detail each step from the research point of view.

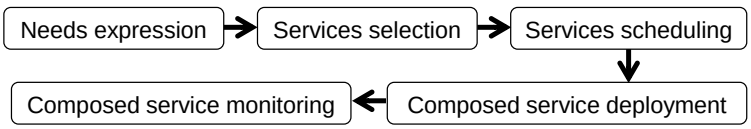

Figure 5. Illustration of the composition process steps.

3.1. Service Composition's Research Landscape

This subsection covers the service description, publication, and discovery; the composition description and optimization, including service interoperability; and finally the composed service deployment and monitoring.

3.1.1. The Description of Services

It is clear that the service description plays an important role in the composition process. A well-described service increases the relevance of its selection as well as the consistency (correctness) of the resulting composition pattern. Indeed, a service is represented by its description, which corresponds to the functional and non-functional priorities.

The functional properties, as their name suggests, refer to the functionality delivered by the service. It includes descriptions of the input/output parameters and logic function (business) that the service performs. For example, a service whose logic function is sending an SMS has as input of two strings: the number of the recipient and the message body. The description of non-functional properties is an important aspect of the process of composition. Indeed, this part of the description indicates, for instance, the availability of the service, response time, or even its business model (for example, the rates per hour). For instance, for the SMS service, non-functional properties could be the business model (prices, promotions, ...) and quality of service (the maximum delay of delivering a message, ...) [17].

The functional and nonfunctional descriptions generate non-insignificant complexity in the composition process. Thus, several protocols have been proposed where the functional aspect is predominant compared to the non-functional aspect. For example, WSDL (the current reference) is used to express the operations provided by the service. Web Ontology Language for Service (OWL-S) and Web Service Modeling Ontology (WSMO) add a layer of semantic description based on either domain-specific or general ontologies to assist service discovery (see Section 4.1.3).

In addition, all proposed protocols and languages (described later) were designed from the perspective of information systems and are intended for experienced users. The semantic description, especially tagging techniques, contributes not only to a better interpretation by machines through reasoning but to bridging the gap between service description technologies and end-users as well.

3.1.2. The Publication and Discovery of Services

Service publication and discovery are two important operations for the composition process and particularly for the selection of the most relevant service. The publishing operation essentially raises issues of data and information engineering summarized in database technologies and access means to populate those databases with services description. Service discovery, based on those database technologies, has to provide not only access means but should also select the service descriptions that best fit the selection criteria (the request). This actually constitutes an optimization problem. From the perspective of SOA specifications, UDDI technology with its variants is the reference. Semantic technologies are also an alternative for optimizing the discovery and selection services in the composition process.

3.1.3. The Efficiency of the Composition Process

The heart of the service composition is the selection and scheduling of services to match the description of the service that we would like to compose. This description should provide the hints needed to form the composite service schema. Several approaches and technologies are possible. For this purpose, many standards have been proposed to explicitly define the description of a composition pattern, namely BPEL4WS, BPML, and WSCI. Less explicit tools, based either on textual or graphical interfaces, have been proposed to allow for the definition of the composed service logic. For the automatic approach, the logic of composition is formed based on information taken from the user context.

3.1.4. Interoperability, Execution, and Monitoring of Composite Services

Interoperability between services is also a key issue in service composition. Factually, a composite service is represented by a composition pattern that reflects the logic of this service. This logic includes the information flow between services and settings. Two schemes of interoperability are defined in the state of the art of choreography and orchestration [18] (described below), where services communicate with each other through standardized languages. The defined composite service has to be defined according to the interoperability schema. Based on this schema, monitoring tools are designed to gather information about the composite service running state and issues that could occur due to the unpredictability of external partner services or unexpected behavior of composite services [19].

4. Taxonomy for Services Composition

In this section, we provide a detailed taxonomy of different existing approaches for service composition from both system and user perspectives. On the one hand, service composition is organized from the system perspective based on the following identified approaches: formal, structured, or semantic. On the other hand, service composition is categorized from the user perspective based on the end-user's involvement (i.e., manual, automatic, or semi-automatic approaches).

4.1. System Perspective

In contrast to the user's perspective, the system perspective provides details about the techniques and mechanisms used to achieve service composition in terms of service publication/discovery, scheduling, and deployment. Concerning this perspective, we identified three non-exclusive approaches: the *Formal approach*, which provides the tools and formalisms that allow, for example, the formal validation or verification of a number of predefined properties; the *Structural approach*, which looks to establish data structures and access methods in formal operational protocols and languages that are often used by other approaches; the *Semantic approach*, which brings semantics to improve and optimize the composition's operations mentioned above; and, finally, the domain-specific service composition, such as a *Grid Service* and *Geographical Information System*.

4.1.1. Formal Approach

Formal models can be used for the automatic or manual modeling of composed services. The formal description techniques allow for the use of methods and tools to make the development cycle of services more reliable, faster, and cheaper. Formalisms for specifying these services are based on precise and mathematically based syntaxes and semantics. Developing models will apply methods and tools in three major phases of the development lifecycle of the service: (i) the verification, (ii) the automatic or semi-automatic code generation, and (iii) the generation of the test benches.

The objective of the verification phase is to improve the reliability of the process of developing an implementation by ensuring that the formal model on which the implementation is based is valid with respect to a given set of properties. These properties are represented in the form of logical properties or sub-sets of an automaton [20,21].

The formal test phase is a set of executions of specific test sequences on the implementation. Test sequences are obtained from the formal model by trying to cover all aspects of the service compound. Tests can be generated automatically or semi-automatically based on criteria, goals, or assumptions. There are many stages and types of tests in the development process of a service: the conformance, the interoperability, the unit test, and the integration tests. The majority of these procedures are standardized or described in some reference software development lifecycle management [22].

Depending on the degree of modeling, it is possible to generate code for all or part of the application model. The more precise the semantics modeling language, the more complete the code generation. For instance, in the UML formalism, semantics are weak

or nonexistent. At best, they will generate the interfaces from the model. Many modeling languages have been standardized and are based on various concepts such as automata, the states/transitions systems, temporal logic, interaction, etc. [23]. Some of them are briefly described in the following.

The Specification and Description Language (SDL)

SDL [24] is a specification language standard defined by the International Telecommunication Union (ITU-T), which aims to describe communication protocols. Even if the SDL is a modeling language that was initially used for communication protocols description, it is more generally used for modeling real-time applications. This is due to the syntax of the language, which describes a service using the following:

- Description in the form of state machines;
- Exchange of information via asynchronous messages;
- Use of timers.

The ECharts Formalism

The ECharts is a state machine-based programming language that was developed by ATT Laboratories and was first introduced in 1999. It is still actively supported today due to its textual and graphical syntax. The goal of ECharts is to provide an easy way to describe modular, verifiable, maintainable, and reusable services and composite services. ECharts is based on a state machine formalism, which allows for the modeling of transitions with priorities (event-driven systems). This feature provides flexibility in reusing models from a given machine, as new transitions can be added to certain states with a higher priority than the existing transitions, thus modifying the behavior of the original machine [25].

In summary, this approach offers formal methods to verify or test software components (services) automatically, which are necessary for the composition of services. However, these formalisms are too specific and require high technical and mathematical skills, and therefore cannot be directly used by end-users, but can intervene at underlying layers to ensure some required properties (such as security properties).

4.1.2. Structural Approach (Software Engineering Approach)

This approach is more about providing formalisms and tools to describe service interfaces (inputs/outputs) and behavior in order to compose services and create new ones. For instance, WSDL provides XML-based syntax for describing services and BPEL provides a framework for orchestrating services. By contrast, formal methods (Automata, Petri nets) provide tools to improve the reliability of the process of developing an implementation by ensuring the conformity of the formal model on which the implementation is based to a given set of properties. We present hereafter a number of standards in the area of Web services that allow for implementing the SOA concepts [26]. This includes WSDL, SOAP [27], HTTP, XML, and UDDI.

Web Services Description Language (WSDL)

WSDL [28] is an XML-based language that is used to describe the Web service. In other words, it describes: what a Web service can do, where it is, how to access it, and in which format. The WSDL provides features for service naming and operations naming (input parameters and responses organized in the form of messages). It also contains detailed information about the used communication protocol (often HTTP), information on the technique of data encoding, and the network address in the form of a URL. It does not contain semantic information about exposed operations, and there is no notion of order in the invocation process. The client can use SOAP to actually call one of the operations listed in the WSDL file.

The Universal Description Discovery and Integration (UDDI)

The service directory (also called repository or registry) is the place where the services are registered. The SOA concepts can be instantiated by using the standard UDDI (https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec, accessed on 26 August 2020) as a service directory. The UDDI is structured into three pages (components): white (information by name), yellow (information by category), and green (service provided by WSDL). It is designed to be interrogated by SOAP messages and to provide access to service description documents (WSDL).

Simple Object Access Protocol (SOAP)

SOAP [29] is a messaging protocol for exchanging structured data between applications over a variety of network protocols. The structured data are encapsulated in XML-based messages according to the SOAP exchange standard. SOAP is itself represented in XML with a header part and a part that corresponds to the application payload (called *body* or simply *payload*). The SOAP header part is optional and generally used to transfer data authentication or session management. These are aspects that are supported by the underlying protocol. The body part is in charge of encoding the names of operations and their parameters and returned results. The SOAP is typically deployed over HTTP but can also operate over SMTP or JMS. The SOAP is also defined by an envelope that allows for describing the specification of the namespace. One of the main advantages of SOAP is that it has built-in error handling, making it more robust and less error-prone. SOAP also provides a way to ensure the security of the message being exchanged through features such as WS-Security. Additionally, SOAP is language-independent, meaning that it can be used in any programming language, platform-independent, and can be used on any operating system. However, SOAP is more complex and verbose than REST, which can make it less efficient and more difficult to implement. SOAP also requires more bandwidth and resources than REST and uses XML as its message format, which is less lightweight than JSON.

Representational State Transfer (REST)

REST [30] is simple and easy to use, making it easier to implement and less error-prone. REST uses a smaller message format (JSON), which makes it more lightweight and efficient. REST is less resource-intensive and requires less bandwidth, and it is typically faster than SOAP. However, REST does not have built-in error handling, so it relies on error codes to indicate problems. REST does not provide the same level of security as SOAP, so it may not be suitable for sensitive information. REST is not always language- or platform-independent and may be limited to certain programming languages or operating systems.

Business Process Execution Language (BPEL)

The BPEL has been introduced by the OASIS standardization group as the successor of XLANG and WSFL. The BPEL is an XML representation used as an instantiation of a service-oriented architecture (SOA) concept. Specifically, in the SOA, the enterprise applications are managed from a common platform to enhance the dialog between applications and their integration. BPEL organizes the dialog between the different applications of the SOA architecture by invoking basic services according to a predefined schema Figure 6. Specifically, BPEL4WS 1.1 (Business Process Execution Language for Web Services) [31] and its successor WS-BPEL 2.0 (Web Services for Business Process Execution Language) [32] are the BPEL standards that allow the user to describe their business processes in the form of Web services, and to specify how they are interconnected in order to accomplish particular tasks.

BPEL is a complete and open standard with many supporting engines. BPEL was quickly accepted by the industry and is now the dominant technology in the field of Web service composition. It takes the form of an XML file readable in the engines of business process management. It drives the execution of business processes (workflow). The BPEL

file therefore concerns matters such as processing data, sending messages, or calling a function. There are two types of BPEL processes:

- An abstract, which specifies the exchange of messages between the various parties without specifying the internal behavior of these parties;
- An executable process, which specifies the execution order of activities. Each activity represents a given process (a Web service) involved in the main composition script.

```
namespace pns = "http://example.com/loan-approval/";
namespace lns = "http://example.com/loan-approval/wsdl/";

@type "http://schemas.xmlsoap.org/wsdl/"
import lServicePT = lns::"loanServicePT.wsdl";

@suppressJoinFailure
process pns::loanApprovalProcess {
    partnerLink customer = (lns::loanPartnerLT, loanService, null),
    approver = (lns::loanApprovalLT, null, approver),
    assessor = (lns::riskAssessmentLT, null, assessor);
    try {
        parallel {
            @portType "lns::loanServicePT" @createInstance
            request = receive(customer, request);
            signal(receive-to-assess, [$request.amount < 10000]);
            signal(receive-to-approval, [$request.amount >= 10000]);
        } and {
            join(receive-to-assess);
            @portType "lns::riskAssessmentPT"
            risk = invoke(assessor, check, request);
            signal(assess-to-setMessage, [$risk.level = 'low']);
            signal(assess-to-approval, [$risk.level != 'low']);
        } and {
            join(assess-to-setMessage);
            approval.accept = "yes";
            signal(setMessage-to-reply);
        } and {
            join(receive-to-approval, assess-to-approval);
            @portType "lns::loanApprovalPT"
            approval = invoke(approver, approve, request);
            signal(approval-to-reply);
        } and {
            join(approval-to-reply, setMessage-to-reply);
            @portType "lns::loanServicePT"
            reply(customer, request, approval);
        }
    }
    @faultMessageType "lns::errorMessage"
    catch(lns::loanProcessFault) { |error|
        @portType "lns::loanServicePT" @fault "unableToHandleRequest"
        reply(customer, request, error);
    }
}
```

Figure 6. A simple BPEL script.

Services Composition Using BPEL

The ability to integrate or compose existing services into new services is the most important functionality provided by SOAs. The service composition must be created taking into account the maintenance of services that rely on other services. The SOA offers a homogeneous environment for the composition in such a way that its components are described in the same protocol and communicate with the same standards for exchanging messages. The composition of services is achieved through a framework that consists of three parts:

- Models of composition and language: The composition of services means the creation of a workflow that defines the order in which the services are invoked, how the data are transmitted, and how the logic is implemented. A composition model provides a language in which the composite service workflow has to be written.
- A development environment: This development environment consists of an editor for the language of composition, such as a programming language integrated development environment (IDE).

- A runtime environment: A composition of services is executed by creating instances of the composition script and deploying them in an execution environment (application servers).

There are two distinct ways to conceive a composition of services, i.e., the choreography and orchestration:

- *Choreography:* The choreography of services describes the collaboration between services to accomplish a given goal. The control logic for a choreography is distributed. Each service knows what to perform and which service to contact. Choreography languages allow for a description of protocols that the participants have to follow. In [33], two main choreography approaches were defined: (1) the global model, which describes a protocol from a global view of the messages exchanged by all parties, and (2) the interaction model, in which each service describes its temporal and logical dependencies among the exchanged messages, which is similar to defining a kind of interface. WS-CDL (WS Choreography Description Language) adopts the global model, whereas WSCI and the abstract BPEL process are based on the interaction model.
- *Orchestration:* The orchestration of services allows for a definition of the sequence of services according to a predefined schema and runs based on predefined "orchestration scripts". These scripts are often represented by business processes or workflows inside or outside an organization (enterprise). They describe the interactions between applications by identifying the messages and by connecting the logic and invocation sequences. Orchestration describes the way in which Web services can interact together using messages, including the business logic and execution order. These could include different services from different organizations, and the result could be a model of a long-term transactional and multi-stage process.
  An important difference between orchestration and choreography is that the orchestration is centralized, i.e., the process is under control from the business perspective. However, the choreography provides a comprehensive and collaborative coordination. It describes the role of each participant involved in the application.

In summary, we have examined various methodologies for composing services based on different techniques, protocols, and standards. Although these protocols and standards are essential for service composition, they require specialized technical knowledge that is not always accessible to end users. A more pragmatic approach is to develop simpler tools that encompass these protocols and standards, such as tools for publishing and discovering services or tools for creating BPEL scripts for service composition (Figure 7 (https://www.oracle.com/technical-resources/articles/matjaz-bpel.html, accessed on 22 January 2020)). These tools lower the technical skill level required, but still may be too complex for end-users.

Other Related Technologies

Some of the recent technologies in software engineering, such as microservices and containers, can be used in Web service composition in different ways. They provide different ways to create, compose, and manage Web services by allowing developers to create more flexible and scalable solutions. These technologies can improve the maintainability of the solutions and can facilitate the deployment and scaling of the solutions. In particular, microservices are a software architectural style that structures an application as a collection of small, loosely coupled services. Web service composition can be implemented using microservices by breaking down a complex application into smaller, independent services that can be composed together to form the desired functionality. On the other hand, containers are a way to package and deploy software applications. Web service composition can be implemented using containers by packaging each service in a container and then composing the services together by connecting the containers. This allows for an easier deployment, scaling, and management of the services.

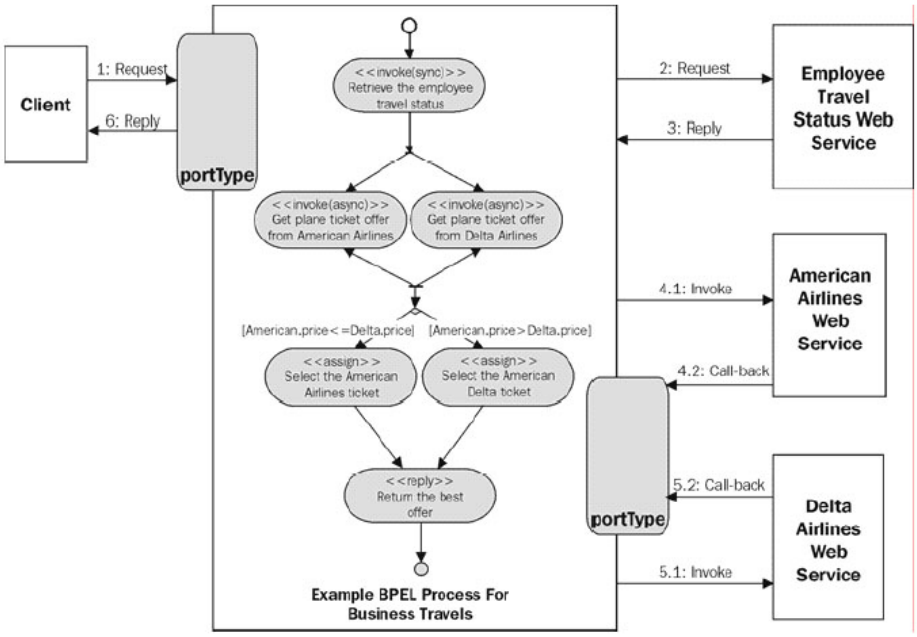

Figure 7. Example BPEL process for travel arrangements.

4.1.3. Semantic Approach

Semantic Web technologies are meant to enable greater access to services on the Web. Users and software should be able to discover, invoke, compose, and monitor Web resources offering particular services and having particular properties, and should be able to do so with a high degree of automation. A number of standards and technologies were introduced in the world of semantic Web services. Hereafter, we describe two main ones.

Web Ontology Language for Service (OWL-S)

(formerly DAML-S) [34] is a services ontology that provides a solution to these functionalities. The overall structure of the OWL-S ontology is composed of three main parts: (i) a service profile describes what the service requires from users and what it gives them; (ii) a service model specifies how the service works; and (iii) a service grounding gives information on how to use the service [35]. The process model is a service model subclass that describes a service in terms of inputs, outputs, preconditions, postconditions, and, if necessary, its own subprocesses. In the process model, we can describe composite processes and their dependencies and interactions. OWL-S also defines three models of processes: atomic, which have no sub-processes; simple, which are not directly invokable and are used as an abstraction element for either atomic or composite processes; and composite, which consists of sub-processes.

With respect to BPEL tools, OWL-S efforts are still focused on research issues and few implementations are currently available. However, we can cite the OWL-S Editor [36] developed by SRI International as a protege (http://protege.stanford.edu/, accessed on 10 June 2020) plugin, and provides a graphical environment for editing an OWL-S service resource, the control flow graph of a process, and "runs" (tests) a defined process. The OWL-S IDE project (http://projects.semwebcentral.org/projects/owl-s-ide/, accessed on 7 October 2010), formerly known as CODE, is also concerned with the development of OWL-S services. The OWL-S IDE is a plug-in for Eclipse, and attempts to integrate the semantic markup with the programming environment. Developers can write their Java code in Eclipse and run a Java2OWLS tool to generate an OWL-S "skeleton" directly from the Java sources. The idea of integrating SWSs more closely with the programming environment used to develop the service implementations is a powerful feature. However, it will often be more useful to generate the semantic markup before the Java (or other) code, as the semantic

descriptions can be seen as a higher level of abstraction of the programming modules. The OWL-S IDE does not provide any graphical visualization of services or processes.

Another OWL-S editor is provided by the University of Malta [37]. It is a stand-alone program providing a WSDL import as well as a graphical editor and visualization for control flow and data flow. Not being integrated with an ontology editor, it shares some of the drawbacks of the OWL-S IDE, without gaining the advantage of programming language integration.

Web Service Modeling Ontology (WSMO)

The Web Services Modeling Ontology (WSMO) [38] shares with OWL-S the vision that ontologies are essential to supporting mechanisms such as the automatic discovery, interoperation, and composition of Web services. Similarly to OWL-S, WSMO is an ontology for describing various aspects related to semantic Web services. Moreover, the WSMO effort defines an expressive Web-oriented language, WSML [39], which provides a uniform syntax for sub-dialects ranging from description logic to first-order logic. Like OWL-S, WSMO Web services specifications are based on the service capability, which consists of inputs, outputs, preconditions, and results. Unlike OWL-S, WSMO does not provide a notation for building the composite processes in terms of control flow and data flow. Instead, it focuses on the specification of internal and external choreography and orchestration using an approach based on abstract state machines (with guarded transitions).

The service basis of WSMO is defined in the same way as the one of OWL-S. This task is achieved by a mediator, which is a key concept in WSMO. In WSMO's approach, mediators perform tasks such as translation between ontologies, or between the messages produced by one Web service and those expected by another. WSMO includes a taxonomy of possible mediators that helps to classify the different tasks that mediators are supposed to solve. The definition of mediators in WSMO calls attention to some important translation tasks associated with Web services. Not surprisingly, these same translation tasks are needed in support of interactions with OWL-S-described Web services. Some OWL-S-based systems [40] also make use of mediator components. However, rather than requiring the existence of a distinguished type of entity in the Web services infrastructure, OWL-S takes the view that mediators are services, and, as such, these mediation services can use the mechanisms provided by OWL-S for discovery, invocation, and composition. Other distinguishing characteristics include WSMO's emphasis on the production of a reference implementation of an execution environment, WSMX, and the specification of mediators (i.e., mapping programs that solve the interoperation problems between Web services).

WSMO instances can be created with WSMO Studio [41], which is a real, complete, and open-source semantic Web service and semantic business process modeling environment. It provides support for WSMO editing with an integrated WSML reasoner, WSML text editor and validator, choreography designer, SAWSDL editor for adding semantic annotations to WSDL documents, execution engine, and many other features. Moreover, it also provides semantic business process modeling according to the business process modeling ontology, a semantically extended version of BPEL, called BPEL4SWS [42].

In conclusion, the semantic approach adds an extra layer on top of the structural approach (Section 4.1.2) by integrating the semantic properties within the operations of description/discovery, and the composition of services [43–45]. With these properties, it is possible to link services together semantically. For instance, it is possible to propose a schema of the composition from a natural language request (see the natural composer in Section Natural Language Composer). This is a major step forward from the end-user perspective. While this approach is valid for very simple patterns of composition, it is unfortunately not advanced enough to allow for expressing the logic of the composition for complex cases. Another hybrid approach was proposed in [46,47], combining both semantic capabilities of service description and non-functional aspects of a service (in this case, QoS).

4.1.4. Horizontal vs. Vertical Compositions

Several recent research efforts have dealt with the Web service composition problem by trying to divide it into two or more sub-problems, introducing vertical/horizontal service compositions and abstract/concrete services concepts. In [48], authors argue that automatically composing Web services involves two main processes of composition, i.e., vertical and horizontal compositions. Vertical composition aims at finding the "best" combination of abstract Web services, namely the abstract workflow, to achieve the main objective, while satisfying all restrictions interdependently. Abstract services refer to each of the sub-tasks (abstract functionality) that, when joined together, represent the main objective of the composite services. Each abstract service can be executed by many equivalent Web services called concrete services. Consequently, the horizontal composition goal is to find the "best" concrete Web service among a set of functionally equivalent services available on the Web. These functionally equivalent services represent a Web service community (a concept introduced in [49–51]). The choice of a concrete Web service is made based on functional attributes such as inputs' types and/or non-functional attributes, such as QoS constraints [52–54]. In this case, the actual binding of the composite service can be performed dynamically (at the execution time) [55]. For instance, authors in [56,57] proposed an evolutionary approach for QoS-aware composite Web services, whereas, in [58], an optimization approach is proposed. These QoS indicators need to be monitored, such as the method proposed in [59]. Non-functional attributes include SLA and pricing consideration. For instance, the authors in [60] proposed a service selection approach from a cloud service market. Other approaches, such as a multi-agent service framework [61], are considered as part of the structural approach too.

The main advantage of distinguishing between these two processes of composition is to simplify the Web service composition problem to reduce the computational complexity. It provides an easier way to consider user intervention, so the user is able to modify/adapt the abstract workflow where necessary [62].

4.1.5. Domain-Specific Approaches

Grid Service for Web Service Composition

A grid service is a type of Web service that allows for the sharing of resources, such as computational power or data storage, across a network of computers. Grid services are relevant to Web service composition because they provide a way to aggregate the capabilities of multiple Web services into a single, cohesive system. By utilizing grid services, developers can create complex, distributed systems that are able to leverage the resources of multiple machines to perform tasks that would be infeasible for a single machine to accomplish alone. Additionally, grid services can be used to manage and orchestrate the interactions between multiple Web services in a composite application, making it easier to create and maintain large-scale, distributed systems.

In [63], the authors provide an introduction to the concept of grid services and how they can be used to create distributed systems that can leverage the resources of multiple machines. The authors also discuss how grid services can be used to manage and orchestrate the interactions between different Web services in a composite application.

For instance, the authors in [64] introduced BPEL4WS (Business Process Execution Language for Web Services) for defining process flows between grid services based on the Open Grid Services Infrastructure (OGSI) standard.

Geographic Information System (GIS) for Web Service Composition

A geographic information system, or GIS, is a domain-specific framework for storing, managing, analyzing, and displaying spatial data. GIS systems can be used to create and display maps, perform spatial analysis, and manage and query large spatial databases. The importance of GIS in Web service configuration is that the user can integrate the GIS functionality into Web-based applications using Web services. Web service composition allows developers to combine the functionality of multiple Web services to create more

complex applications [65]. By integrating GIS capabilities into Web-based applications via Web services, developers can add spatial analysis and mapping capabilities to their applications. This allows GIS capabilities to be integrated into a wide range of applications, such as urban planning, emergency management, transportation, and natural resource management. GIS Web services can also be used to make GIS data and functionality available to other systems through basic or semantic-based discovery capabilities [66]. This allows other systems to expose and consume GIS data and functionality for Mashups such as MashMaker (cf. Section MashMaker).

Most of the described work considers Web services from a system perspective. However, in recent years, end-users have become the focus of various technologies, including Web services composition, as explained in the next section.

4.2. User Perspective

With the emergence of Web 2.0 and related technologies, composing services have left the traditional frontiers of enterprises. SOA concepts need to shift to this new area in order to take into account end-users, which represents a new opportunity for evolution for these concepts. In fact, with the growing number of services available through the Web, the introduction of the end-users in the loop is taking more and more importance. In fact, the end-user needs to use a certain kind of composition in different situations, especially now that Web 2.0 has brought a set of technologies that make it easy to create or collaborate on new services or use other services (for example, Mashups as described in Section 5).

This new perspective brings interesting challenges for researchers in the area of service composition. In this section, we discuss existing research from the user's point of view. This will show, in particular, the limitations of conventional methods (called manual) because they require significant skills in languages, formalisms, and protocols related to the composition of services reserved for experienced users (developers). In addition, this section highlights the limitations of the automatic approach, which decouples the composition from users. This approach is facing complex problems that are hard to resolve (even undecidable in some cases [67]). The hybrid approach, called semi-automatic, involves the user in the composition process and represents an interesting alternative. Eventually, it provides tools for the simplification and abstraction of the different tools and techniques of the composition and also provides functionalities to support the end-user.

4.2.1. Manual Web Services Composition

The first approach is based on manually composing multiple services by the user. This operation must be entirely and manually performed by the end-user. Formal languages such as SDL can be used. Alternatively, textual editors and GUI-based tools that are based on technical protocols and formalisms such as BPEL-based IDEs can also be used. Needless to say, both alternatives require a high level of technical knowledge and experience that the user does not have. Because the majority of end-users are not programmers, this approach is highly criticized for requiring an unrealistic technical level on the end-users, which dramatically limits its use.

4.2.2. Automatic Web Services Composition

The second approach is the automatic services composition. This approach aims at automatically building composite services that are in response to a user context or request. Except for the request, the end-user does not provide any more information about the composition process. Below, we cite some works that fall into the category of the automatic approach and summarize the overall landscape of contributions made in this area. The most common technique used in this approach is based on the so-called goal-driven service composition, particularly the inputs/outputs matching. In other terms, from a defined goal definition (set by the user and/or their context), this technique uses the matching between output and input interfaces (data types) in order to define the most likely pair of services

that can be composed together. Step by step, this operation aims to build the composition pattern that matches the defined goal.

In [68], the authors propose a method based on semantic matching between the input parameters (respectively, pre-condition properties) of a service with the output parameters (respectively, the post-condition properties) of its predecessor. In a similar way [69] introduces a framework for service composition based on functional aspects, in which services are chained according to their functional description. The suggested framework uses the causal link matrix (CLM) formalism in order to facilitate the computation of the final service composition as a semantic graph.

Moreover, context-aware service composition is considered as another way to automatically compose services. The authors in [70] argue that incorporating context awareness into Web service composition mechanisms increases the relevance and robustness of produced compositions. Zhovtobryukh proposes a Petri-net-based approach to enhance core composition mechanisms. In particular, to address privacy issues, Ref. [71] focuses on the use of *Ambient* in the pervasive system systematically with different levels of abstractions. Similar to final state automata, other formal modeling tools [35] are used to perform automatic service composition. Context information can also be used to select the appropriate services for a target composition [72] (although the proposed approach is for network services and not for end-user services).

The full automation of the composition process is not without inconveniences. Practically, in the absence of user involvement validation, the automated operation offers few guarantees about the relevance of the selected and composed services, and can even lead to an end product that does not match the initial goal.Moreover, automation includes a significant complexity that can lead to situations of indecision (in a formal-based approach). Indeed, Ref. [67] shows that checking an e-service composition model is undecidable in some cases. The authors argue that undecidability is due to unbounded FIFO queues. The transaction sequential consistency problem provides another perspective for understanding the queue effect, where independent transactions are allowed to commute.

4.2.3. Semi-Automatic Web Service Composition

The third approach is the semi-automatic service composition, which aims to provide end-users with an enhanced service creation environment. This environment offers support for the automated processing of the composition, where the end-user operates in a more-or-less manner. This approach has gained more interest as the automated service composition approach presents serious limitations. The semi-automatic composition comes to resolve the situation by involving the end-user in the composition process by addressing particular issues; for instance, the difficulty of selecting a relevant service among the many available. The semi-automatic composition has taken several forms that has evolved over time. A current evolution of the semi-automatic composition is what is now commonly called Mashups [73]. This latter evolution incarnates the emergence of Web 2.0 and, more specifically, its user-generated content (UGC) aspect.

More generally, based on existing related works, we can see the emergence of a multitude of methods for semi-automatic composition that are identified and explained hereafter. Generally, from the user perspective, semi-automatic service composition includes composition frameworks with graphical or textual interfaces, semantic-based tools such as tagging techniques, or even social features such as sharing or rating services (both basic and composed). These characteristics are detailed in the next section (Section 5). However, beyond the simple and direct user involvement (participation) through selecting and scheduling services and still from the user perspective, we have identified three major ways of considering the user in the composition process. In fact, some systems focus on an individual end-user, tracking their interests or preferences to use them later to define the best service that they might need. An alternative way is to consider the user as part of a community. Consequently, the system tracks the interest of this community in order to build a list of preferences used to help in the composition process. A third emerging way is

a social-network-oriented approach that is based on leveraging the social aspect of how end-users operate in the service composition environment [74]. These three approaches are detailed in the following.

User-Centric Approach

This first approach aims at building a profile of the user or involving them in the indecision points by providing tools and interfaces to facilitate the service composition process. In this approach, we can find numerous user-driven composition tools such as in [75], where semantic service discovery facilities are provided based on user preferences. A similar approach is presented in [76], where the author introduces a system called Koala (currently Co-Script). This system, materialized by a "sidebar" in the Firefox browser, learns from the user behavior when browsing a Web page, and transforms this behavior into a series of actions. This system's objectives are (i) to parametrize the following abstract actions and make them executable and (ii) to allow end-users to share their composed actions. The script can also be modified by other users or adapted to their profiles. Even if this approach's goal is to provide the end-user with support tools for service composition, we may notice that it does not take advantage of the whole information available about how users use services in a semi-automatic service composition environment.

Community-Centric Approach

The second kind relies on the knowledge produced in communities or in specific domains. A community can be involved in the process of composition in two ways. The first one is tagging or annotating basic and composite services in order to provide advanced descriptions of services (semantics, classification). This first way allows for an improvement in the discovery and selection operations, which could be leveraged to support users forward and downstream the composition process itself (what we name "a priori or posterior support"). The second method of community involvement is to extract the generated knowledge in a community or a specific domain in order to define a set of rules considered as "best practices" in this community or specific domain [77]. These rules are used to build recommendation systems to assist users in the composition process.

In this regard, the authors in [78] provide an interesting introduction to domain-knowledge for services composition. First, they explain the observed lack of it in the services composition structural approach (UDDI, WSDL, SOAP). In fact, the latter does not address the issue of the coordination and scheduling of services. Several industry standards such as BPEL and WSFL offer solutions for "a priori" composition. According to the authors, this is unsuited to a domain-specific approach (targeting a specific area; for instance, scientific computing).

The other approach addressing this problem is the semantic approach based on ontologies, which enables a "sophisticated" service discovery. Some researchers describe the possibility of using this technique of discovery (semantic matching) to manage the service composition. The lack of this method is due to the indeterminism that may arise during the selection phase, which is based on the semantic description of service functionalities. The authors stress the fact that the e-science domain implies certain dynamic processes that the structural and semantic approaches cannot cover. Hence, the domain-specific knowledge is established to support the services' dynamic selection and configuration. Methods such as CommonKADS and OilEd have been introduced to interpret the domain-specific knowledge provided by experts in a list of rules and actions. This list allows for building a service recommendation system. These recommendations can be provided to a software agent (such as that proposed in [79]) or the end-user through the development environment. This will help to proactively improve the services selection and composition processes.

Returning to [78], the author proposed a prototype for the specific domain of engineering design search and optimization (EDSO) for the modeling, analysis, and optimization of an aerodynamic object. This prototype helps less or more experienced users to build (compose) a suite of EDSO algorithms represented in Web service form to meet their

specific needs. Another study [80] suggested the same approach for service composition by upgrading this process using domain-specific knowledge (in this case, it refers to a life-science domain).

5. Mashups Editors: An End-User Services Composition Environment

Currently, we are witnessing the proliferation of Web services and APIs exposed through the Web [81]. Service composition tools propose an environment to take advantage of this proliferation by allowing users to compose services for their own interest. In addition, Web 2.0 is "cultivating" and promoting a population of creative users who generate a significant amount of content. However, as we have mentioned before, end-users have no required skills to manipulate Web services. Thus, service composition platforms and tools aim to provide features and facilities to help end-users in these operations. These efforts have led to the emergence of the so-called Mashups. As an introduction, a Mashup is defined as a Web application created by reusing existing Web resources, considered here as services. The framework and environment used to create a Mashup is named a Mashup editor (also called a Mashup creation environment or Mashup maker). This section presents existing Mashup frameworks and conducted research studies, with a special focus on features related to support for the end-user.

5.1. Mashup and Mashup Creation Environment

An application that combines content from more than one source into an integrated experience or service is called a Mashup. The process of "mashup creation" can be obviously performed at the level of a Web programming language (e.g., PHP, java) by developers, or more easily performed in frameworks (e.g., Mashup editors) by end-users. A Mashup is a more informal service composition. Service developers often have strong preferences with regard to their service creation environment. For end-users, a more user-friendly environment is more attractive, but will of course imply fewer options [82]. Because they are very intuitive, emerging service creation tools focus on how to enable the end-user themselves to create Mashups. For instance, we find, in the Internet world, Yahoo Pipes (Yahoo Pipes, http://pipes.yahoo.com/pipes/, accessed on 23 march 2010), Microsoft Popfly (Microsoft Popfly, discontinued on 24 August 2009), MashMaker [83,84], MARGMASH [85], and MARMITE [86], and, in the telecom world, eZweb [87].

5.2. Overview of Major Mashup Creation Environment

Mashup creation platforms support the user in integrating and orchestrating services for their final composite application and provide an abstract layer that hides the complexity of the underlying process model (e.g., BPEL). The growing visual programming paradigm (graphical) of Mashups is the most common way to meet those requirements. Other methods are the description of the processes via a naturally readable rule language, occasionally called a controlled natural language (CNL), or the implementation of a timeline that describes the user interaction on the basis of their chronological appearance. In order to come to a comprehensive solution for the modeling process, several other aspects, such as event-handling, dependencies between user interaction, or message flows, have to be considered.

5.2.1. Graphical Editor

The graphical Mashups editor tool allows an end-user to create simple Mashups by using the graphical user interface for drawing the workflow describing the logic of the composite service. The end-user can simply drag/drop boxes representing the available building blocks (representing Web services) and connect them to indicate the flow dependencies.

Yahoo Pipes!

Yahoo Pipes is a Web application that consists of a graphical tool that provides end-users with service composition capabilities (Mashups). Figure 8 is a screenshot of the

Yahoo Pipes tool. The left side of the figure is the service database, and the right side is the composite service created by the end-user. The composite service is defined by a set of interconnected input/output boxes, representing service interfaces, and wires, representing input/output connections between these interfaces.

MashMaker

MashMaker is a Firefox plug-in that enables the end-user to create their own Mashup from existing websites. The most important innovation here is the data extraction from Web pages that contain unstructured data. Figure 9 shows a "Facebook" Web page in which the Mashmaker plug-in automatically extracts all addresses, names, and phone numbers. Thereafter, if the user wants to display these addresses on a map, they just have to drag/drop it into a mapping service (such as Yahoo Maps or Google Maps).

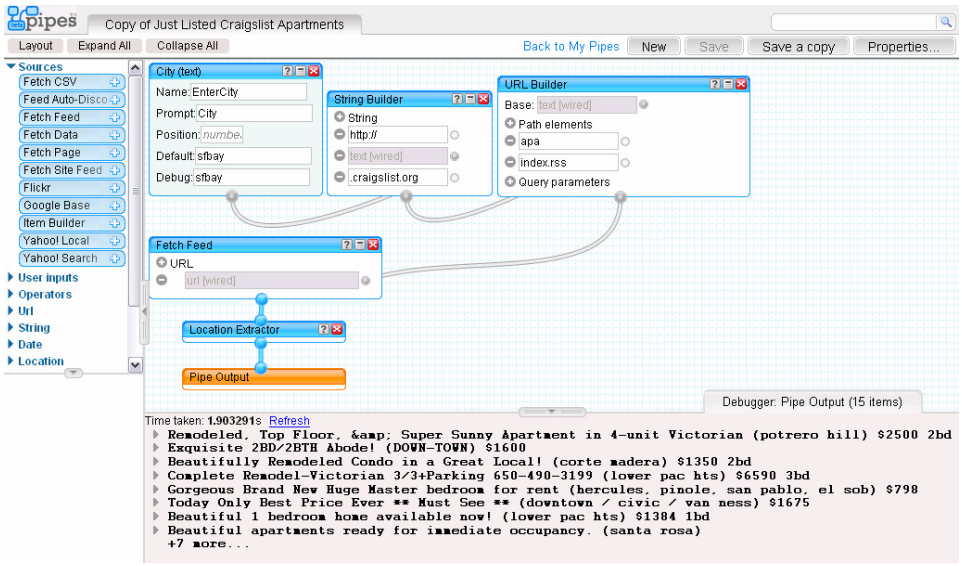

Figure 8. HousingMaps programmed in Yahoo! Pipes.

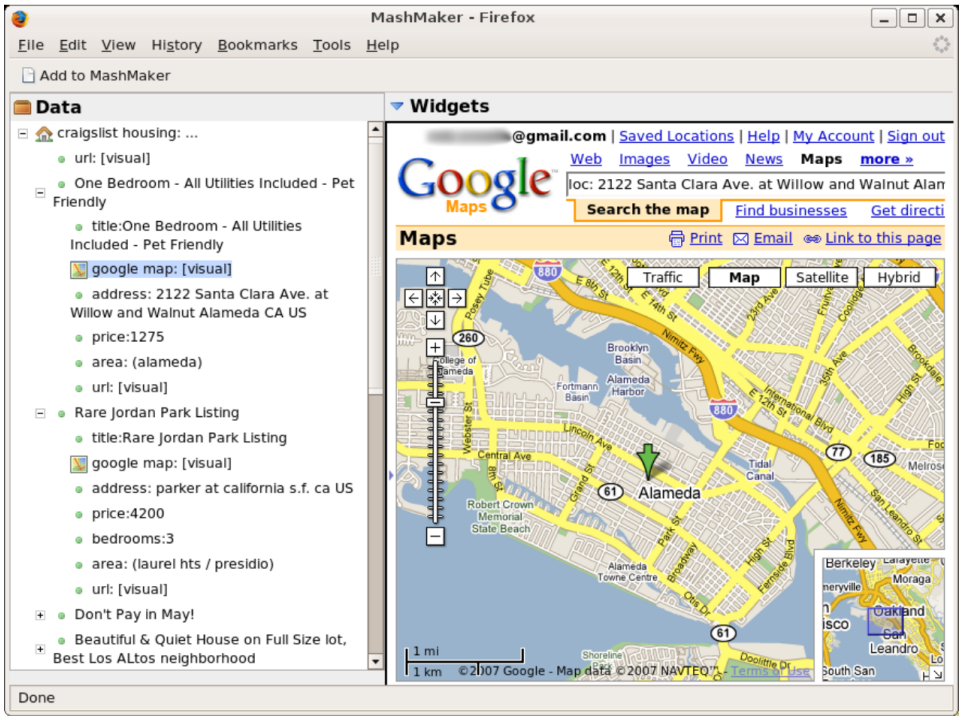

Figure 9. MashMaker example (using Google Maps).

MARMITE

MARMITE is another framework that enables the end-user to create their own Mashup with an incremental execution; users can execute a composite service step by step and see the intermediate results (see Figure 10). It is also implemented as a Firefox plug-in too. Like in Yahoo Pipes!, Marmite composite services are a set of boxes (called operators) chained with wires. However, some services can have alternative associated displays, such as a map or a video player. Users can link the output of a given service with the input of an intended successor service. MARMITE authors have tested their framework on a sample of six persons [86], where the first two are experienced programmers, two have experience with spreadsheets but not with programming, and the remaining two have no experience with either programming or spreadsheets. As a result, three out of six did not succeed in building a composite service, and those who succeeded were those who had knowledge in developing, with one having spreadsheet experience.

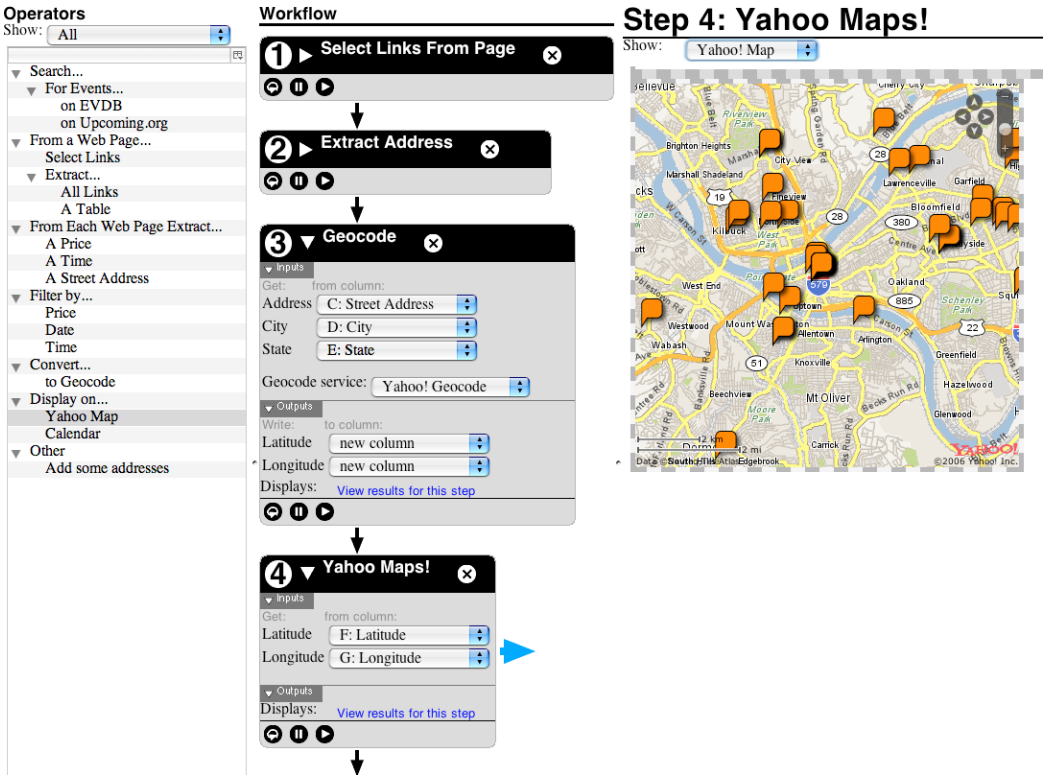

Figure 10. HousingMaps programmed in MARMITE.

Open Mashups Studio

The Open Mashups Studio (http://www.open-mashups.org/, accessed on 23 March 2008) is a Mashup creation environment introduced by Orange Labs. It is based on Open Mashups Modeling (OMM). OMM is a domain-specific language dedicated to applications based on component assembly. It uses a data flow paradigm to connect components and a very simple type of system to represent exchanged data. As Figure 11 shows, Open Mashups Studio is a Firefox plug-in and provides a similar environment to Yahoo Pipes or Marmite. In addition, Open Mashups Studio users can specify the Mashup interface.

5.2.2. Natural Language Editor

The introduction of the semantic Web paradigm in service-oriented architectures enables explicit representation and reasoning about services via a semantically rich description of their operations. Natural language composition focuses on the development of interactive service composition tools that use a textual user interface based on a natural language. For instance, Ref. [88] introduces an approach towards service selection and composition based upon the

interpretation of user requests expressed through an informal human–computer interaction interface that employs a controlled (restricted) natural language.

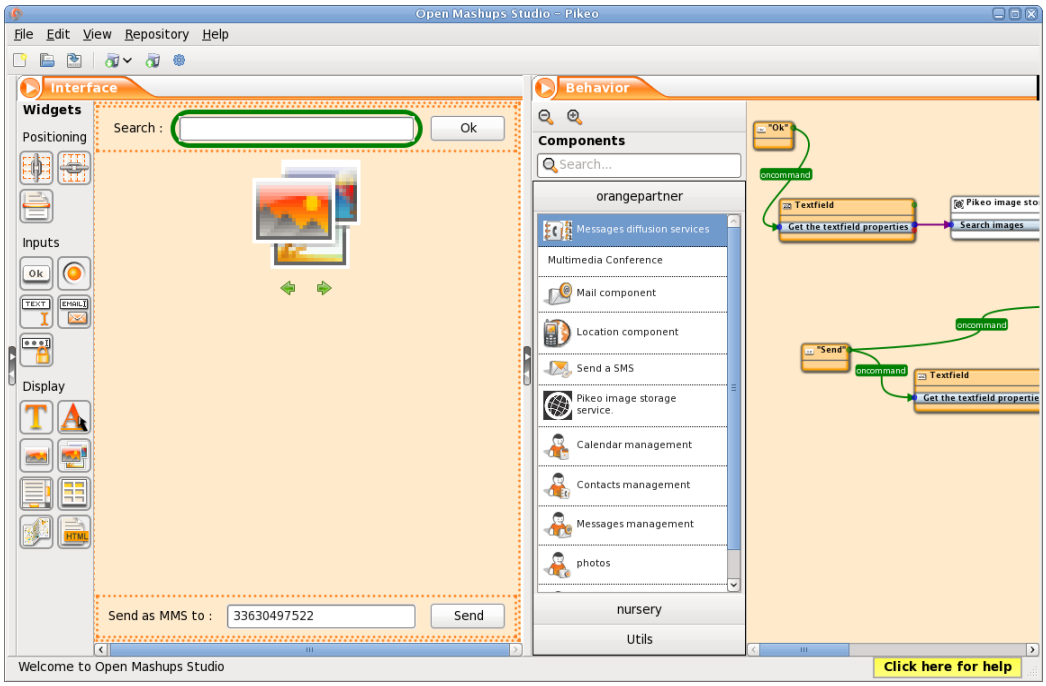

Figure 11. Open Mashups Studio screenshot.

Natural Language Composer

First introduced in [89] and then furthered in the SERVERY (http://projects.celtic-initiative.org/servery/, accessed on 27 March 2016) project, the natural language composer is used to create composite services based on the interpretation of a service request performed using a restricted natural language. This interpretation is obviously constrained by the number of service components that are annotated for natural language usage. An example of a sentence that can be interpreted is "Send by SMS Paris weather translated in English", which will result in the on-the-fly creation of a service that will sequence three basic services: the retrieval of the weather forecast from Paris, a translation of a given text in English, and, finally, SMS sending Figure 12. Four main steps are performed to make the system capable of interpreting such sentences and generating a service that can be executed:

1. Based on the natural language annotation of services in the system, the parsing of sentences is generally recursive in order to analyze and then find a possible candidate among the list of existing annotated services;
2. The interpretation graph is constructed (in an intermediate formalism);
3. Based on the interpretation graph, the system generates the orchestration script in order to create a sequence of service calls, and the arguments are appropriately assigned;
4. The script is deployed into a given execution technology.

Ubiquity

Ubiquity is an add-on for Mozilla Firefox (https://wiki.mozilla.org/Labs/Ubiquity, accessed on 06 October 2022) introduced by Mozilla labs [90]. It is an experimental interface based on a natural language input. It is a collection of quick and easy natural-language-derived commands that act as Mashups of Web services, thus allowing users to obtain information and relate it to current and other Web pages. Users' requests are based on restricted natural language commands that can be extended by the community (see

Figure 13). Basically, Ubiquity commands are small chunks of JavaScript (as an intermediate scripting language) that can be interfaced with Web services.

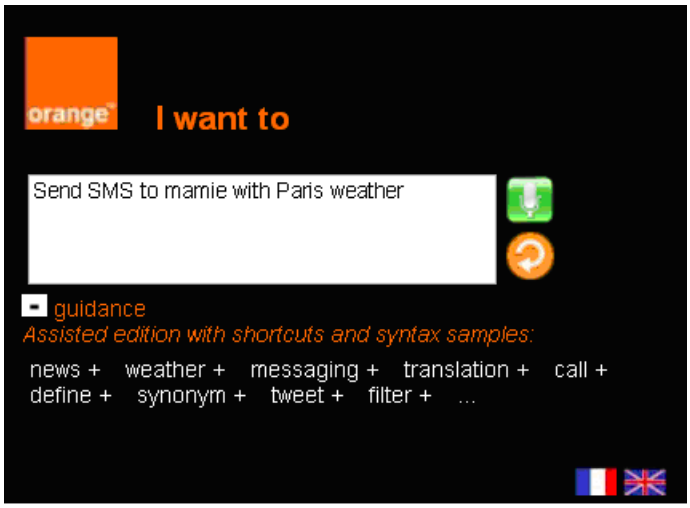

Figure 12. Interface of the natural language composer.

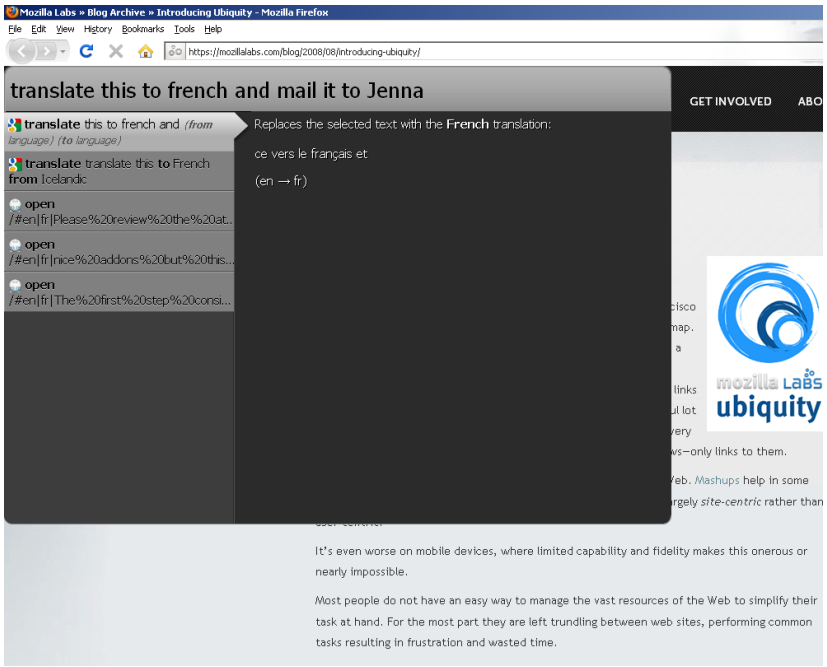

Figure 13. Example of user's request using Ubiquity.

5.3. General Properties Analysis

In [91], the authors present an overview of tools and environments for creating Mashup to identify research issues. The authors point out and explain the difference between Mashup development and classic component-based application development. The Mashup targets specific situational needs (typically a use case). To perform this analysis, the authors selected some Mashup creation environments (Yahoo Pipes, Google Mashup Editor, Microsoft Popfly, etc.). They proposed reviewing these tools instantiated in a particular Mashup sort that is the "housing maps application". They identified the conceptual and practical features that will help to structure the analysis. At the conceptual level, two paradigms were distinguished: (i) the basic components that will be used to create a Mashup, which could be either data, application logic, or the user interface. This classification results in a layered view of Mashups creation that will include three layers: a

presentation layer (interface), data layer, and functional processes layer. (ii) The second identified paradigm is the composition logic; in other words, how the components are assembled. This operation depends on several parameters, which include: the output type (data, application logic, or interface), the orchestration style (flow-based, event-based, or layout-based), inter-component communication (one-to-one interface, centralized communication media), and the composition execution (instance-based or continuous).

At the Mashup creation environment level, several characteristics have been identified and classified here through two concepts: (i) the user interface, which can be browser-based (sometimes plug-ins), and is characterized by an environment type (drag and drop, textual, or hybrid) in order to provide facilities for the user, who could be a Web user, an advanced user, or a programmer. (ii) The execution environment, which is an important parameter to consider since it stands for delivering Mashups for users. It is characterized by the deployment type (hosting: local, Mashup provider, or a third party), the integration operation, which may occur on the server side (engine-based or Web-app-based implementation style) or on the client side (for instance, within the browser via JavaScript), and, finally, the scalability of the execution environment (number of data sources, composition models, or users). This structured analysis allows for a detailed comparison of different Mashup makers according to various criteria. However, unlike [92], this analysis mainly highlights the Mashup environment's technical aspects from the service providers' viewpoint. It helps to identify the technical issues to consider when implementing a Mashup maker for social networking matters (e.g., scalability). Nevertheless, this study does not provide elements that help to identify the requirements that each Mashup framework has to meet to become as user-friendly as possible.

In [93], a similar study highlighted that Mashup creation can be separated into several conceptual levels. This has introduced the concept of "lightweight composition", which is just another name, from the end-user point of view, for the Mashup creation process. Furthermore, the authors focused on Mashup makers, with a special focus on community-related and social network properties, which they named "mass collaboration" features.

From the end-user point of view, Grammel et al. [92] investigated tools and environments for creating Mashups, which they called "Mashup makers". This investigation provided an advanced analysis of the main characteristics and properties provided by these environments from the end-users' point of view. The authors defined a Mashup as "an end-user driven recombination of Web-based data and functionalities". In this study, six Mashup makers were selected and classified into three categories: information Mashup, process Mashup, and Web site customization. Seven dimensions were defined in order to analyze the selected Mashup makers, including the support for the community features dimension, which represents a particular interest in our context. Indeed, community members provide elements that can be reused by other members, create examples, and help each other. Some features were identified and classified as: (i) Mashup sharing, (ii) collaborative classification, notation, or marking, and (iii) exchanges and discussion forums. Accordingly, the proposed analysis can be applied to the social network (of friends) case. For instance, this analysis could be useful for the specification of a "Mashup maker" in order to optimize end-users' support features. We may notice that the authors have highlighted the need to introduce social networking features at the heart of the Mashup creation process.

5.4. End-User Support

After reviewing the general properties of the Mashup creation environment, the next section highlights, based on existing studies, the growing need for supporting the end-users in order to help them compose services. Table 1 summarizes the main features provided by the Mashup creation environments cited above. These features could potentially be used as support for end-users at several levels. Moreover, in order to facilitate the service composition for end-users, current Mashup editors provide an abstraction layer that hides the technical specifications and simplifies them for the users. For example, providing a Web service with an abstract description in the form of an input/output black box

and a composed service in the form of a graphic flow or sequence of services. Most Mashup editors also allow for the reuse of created composed services as building blocks to compose other services. In addition, to help end-users to compose services, Mashup creation environments provide learning materials such as videos, tutorials, and forums for assistance. Learning by example is also an approach that allows new users to reuse and edit Mashups that have been created by others. We categorize the features listed above as indirect support for users in the process of composition.

To provide direct assistance to end-users, most Mashup editors tend to ease the end-user intervention in the process of composition. This intervention can take place at three levels:

- Pre-composition support: by facilitating the selection of services by features that are either service categorization, textual, or contextual selection.
- Post-composition support: by providing the ability to tag or rate basic services. This information is used later on by recommendation systems (collaborative filtering or content-based) at the pre-composition phase in order to allow for the automatic selection of services that fit with users' preferences.
- In-composition support (at the services scheduling phase): for this case, no direct features have been identified in the current Mashup editors that help end-users in selecting services when they are creating a composite service (connecting services).

Table 1. Summary of the most relevant features offered by some Mashups environments.

		MS. Popfly	**Yahoo! Pipes**	**IBM Mashup Center**	**Intel Mashmaker**
Abstraction Level	*Reuse of complete Mashup*	Y	Y	Y	Y
	Visual data-flow languages	Y	Y	Y	N
Learning support		Y	Y	Y	Y
Sharing Mashups		Y	Y	Y	Y
Community features	*Tagging*	Y	Y	Y	Y
	Rating	Y	Y	Y	N
Discovery and selection	*Text-Based Search*	Y	Y	Y	N
	Categorization of services	Y	Y	Y	N
	Context	Y	N	N	N

Nevertheless, several studies have shown the potential of exploiting the interactions of users with services as a basis for supporting features to the end-user. In this same direction, through a use-case approach, Floyd et al. [94] highlighted the APIs proliferation on the Web in parallel with the number of creative Web users. The study shows the benefits of the collaboration between end-users and developers, which combines the innovation and creativity of end-users with the expertise of developers. Automating this collaboration is an important challenge that we are looking to tackle. In that regard, an interesting study [95] describes the interactions of Yahoo! Pipes' users. This can be used to extract social structures based on an analysis of user interactions. Furthermore, these users interact with services through the Mashups that they create. Soriano et al. [96] emphasize the growing importance of the user–service relationship in a service-oriented architecture for composing services. In fact, the authors introduce EZWeb, an environment for sharing Mashups between colleagues, as a basis for co-production in an enterprise context. In addition, Refs. [97,98] emphasize the phenomenon of what they call "social interaction"

between services. In fact, the aspects of trust and reliability between services may impact the service selection for the composition. Yu and Woodard [81] propose a very interesting view of the ecosystem of Mashups. This study, on the Programmableweb API repository (http://programmableweb.com, accessed on 14 October 2021), has truly shown that the utilization of services follows a long-tail effect (power-law distribution), one of the major and interesting properties in social networks [99]. We believe that service recommendation is a solution for disseminating expertise between users to enable them to compose services.

In this regard, context-based frameworks for service selection and composition have been proposed in [100,101]. Moreover, the authors in [102] provide a context-aware service discovery framework based on social knowledge. Social relationships and potential behavioral similarities between users, or, in general, among users with similar interests, allow for the inference of further user's interests. Such an approach is more widely related to the intersection between information retrieval and social network analysis as exhaustively stated in [103], and has been applied to connected application domains such as augmented virtual environments [104].

5.5. Other Related Technologies

Some of the recent technologies in software engineering such as *Low-code* can be related to Web service composition and mashups in different ways. Low-code is a development approach that allows developers to create complex applications with minimal coding. This can be used in Web service composition by allowing developers to easily create and compose Web services using visual drag-and-drop interfaces and pre-built components, rather than writing code from scratch. Low-code platforms often include a visual development environment similar to Mashups, where developers can drag and drop pre-built components, such as Web services, forms, and user interface elements, to create a functional application. They also include pre-built connectors and integrations with other systems, such as databases and third-party services, to allow developers to quickly and easily connect to other systems and services.

Low-code and Mashups are both related to Web service composition, but they have some key differences. A Mashup is a Web application that combines data or functionality from multiple sources into a single, integrated experience. Mashups can be created by combining Web services, APIs, and other data sources together, usually by writing code to connect and integrate them. In our context, we refer to Mashups that do not require writing code to create the integration between different services. On the other hand, low-code is a development approach that allows developers to create complex applications with minimal coding. It uses visual drag-and-drop interfaces, pre-built components, and other abstraction layers to allow developers to create applications without having to write extensive amounts of code. Low-code platforms often include a visual development environment, where developers can drag and drop pre-built components, such as Web services, forms, and user interface elements, to create a functional application.

6. Discussion and Conclusions

We presented in this paper a literature review of Web service composition and end-user oriented composition environments (Mashup editors). In this regard, we discussed the concepts of service, SOA, Web services composition, and its key concepts. We classified the different approaches to service composition either from system or end-user perspectives.

In particular, we pointed out the concepts of SOC and composition of services, which were originally developed for enterprise application integration, and have recently evolved for end-users usage, typically Web users. These end-users are characterized by limited technical and programming skills, but are nevertheless producing Web content. In fact, in the Web 2.0 context, one of the interesting properties of end-users is their ability to produce or participate in producing content. Web 2.0 has brought a set of different technologies dedicated to end-users (even in an enterprise context), so it has become very easy for such users to publish or annotate resources (user-generated content (UGC)). Furthermore, these

end-users are tying new relationships based on interests to the generated content, and stay in touch with their social relatives through online social networks and collaborative environments. Consequently, the composition of services should currently be driven by end-user needs, as it is encouraged by online environments of sharing and social interactions through the Internet.

Mashup editors have emerged as an answer to this evolution in order to overcome the technical complexity that the end-users are facing and to ease the composition process for them. In fact, through this Mashup concept, existing works have provided (i) abstraction features such as visual workflow language, (ii) community features such as rating and tagging, and (iii) service selection facilities such as text-based search. Even if these features are absolutely necessary, we hardly believe that they are sufficient. In particular, during the composition process itself, and as we have pointed out previously, existing features do not currently provide any direct support to end-users. In fact, the users have to manually select and connect all services in order to compose them according to specific requirements and the composition logic. This phase of the composition represents a relatively painful phase of the process due to the lack of support.

Regarding the evolution of the Web from Web 1.0 to Web 4.0, Web 1.0 was about connecting to the Internet and retrieving information, Web 2.0 was the advent of social media and user-generated content, Web 3.0 was the advent of the semantic Web, where computers generate and think about new information instead of humans, and Web 4.0 is the Internet of Things, or, as we call it, the intelligently connected Web.

The objective was to propose a comprehensive and consistent classification framework by conducting a thorough analysis of the different Web service composition approaches and providing a clear and detailed explanation of the proposed classification framework. However, as in most classification approaches, our taxonomy was limited because it was intended for the end-user perspective of Web service composition, and hence did not cover many other classification criteria, such as QoS, expressiveness, scalability, security, and interoperability.

Different emerging approaches promote the continuous assistance of the end-user when they are composing services. This concept is sometimes presented as a service dynamic recommendation [105], where, in a Mashup environment, such a feature is called *Mashup completion* [106]. In other interesting recent studies [107–109], new environments for Web service composition (such as on mobile devices) have been explored.

While Web 2.0 was a central motivation of this work since we considered service composition as user-generated content (UGC), it is interesting to keep in perspective the evolution of the Web from Web 1.0 to Web 4.0. Going beyond the Web 1.0 paradigm, which was about connecting to the Internet and retrieving information, and the Web 2.0 paradigm, which was about social media and user-generated content, Web 3.0 represents the emergence of the *semantic Web* in order to relate semantic information and unleash its potential. On the other hand, Web 4.0 is considered as the *Internet of Things*, where billions of devices are expected to be connected through the *Intelligent Web*, providing a new set of paradigms when it comes to Web service composition [110].

The Web 3.0 and Web 4.0 technologies and capabilities associated with these concepts are expected to provide more personalized and intuitive experiences for users, which can improve the usability and user-friendliness of Web service compositions [111]. For example, the "semantic Web" aspect of Web 3.0 is expected to enable Web services to understand the meaning and context of the data that they process, which can provide more accurate and useful results for end-users. Similarly, the "intelligent Web" aspect of Web 4.0 is expected to include technologies such as artificial intelligence and natural language processing, which can provide more personalized and intuitive experiences for end-users. Additionally, with the help of AI and ML, Web services are able to understand the needs and preferences of the users and provide a more personalized and intuitive composition of service. Overall, Web 3.0 and Web 4.0 are expected to bring new technologies and capabilities that will improve

the usability and user-friendliness of Web service compositions, which will be beneficial for end-users.

With respect to emerging technologies in machine learning (ML) and artificial intelligence (AI), Web service composition can be used to build more sophisticated and intelligent systems. For example, a Web service that uses natural language processing (NLP) to understand and respond to user requests could be combined with a Web service that uses ML to make recommendations, resulting in a more powerful and user-friendly service. Additionally, AI-based Web services can be integrated into Web service compositions to provide a more accurate and efficient processing of data. As these technologies continue to evolve, the possibilities for Web service composition will likely expand further.

Author Contributions: All authors have contributed equally. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lemos, A.L.; Daniel, F.; Benatallah, B. Web service composition: A survey of techniques and tools. *ACM Comput. Surv. (CSUR)* **2015**, *48*, 1–41. [CrossRef]
2. Oh, S.C.; Lee, D.; Kumara, S.R. A comparative illustration of AI planning-based web services composition. *ACM SIGecom Exch.* **2006**, *5*, 1–10. [CrossRef]
3. Chan, K.; Bishop, J.; Steyn, J.; Baresi, L.; Guinea, S. A fault taxonomy for web service composition. In Proceedings of the Service-Oriented Computing-ICSOC 2007 Workshops: ICSOC 2007, International Workshops, Vienna, Austria, 17 September 2007; Springer: Berlin/Heidelberg, Germany, 2009; pp. 363–375.
4. Alamri, A.; Eid, M.; El Saddik, A. Classification of the state-of-the-art dynamic web services composition techniques. *Int. J. Web Grid Serv.* **2006**, *2*, 148–166. [CrossRef]
5. Nahrstedt, K.; Balke, W.T. A taxonomy for multimedia service composition. In Proceedings of the 12th Annual ACM International Conference on Multimedia, New York, NY, USA, 10–15 October 2004; pp. 88–95.
6. Spohrer, J.; Maglio, P.P.; Bailey, J.; Gruhl, D. Steps Toward a Science of Service Systems. *Computer* **2007**, *40*, 71–77. [CrossRef]
7. Kotler, P.; Turner, R. *Marketing Management: Analysis, Planning, and Control*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1984.
8. Zeithaml, V.; Bitner, M.; Gremler, D. *Services Marketing, International Edition*; McGraw-Hill: New York, NY, USA, 1996.
9. Soldani, J.; Luthmann, L.; Gottwald, N.; Lochau, M.; Brogi, A. Compositional testing of management conformance for multi-component enterprise applications. *Serv. Oriented Comput. Appl.* **2022**, *16*, 209–225. [CrossRef]
10. Kirda, E. Engineering of Web services with XML and XSL. In Proceedings of the 8th European Software Engineering Conference Held Jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Vienn, Austria, 10–14 September 2001; pp. 318–319.
11. Papazoglou, M.P. Service -Oriented Computing: Concepts, Characteristics and Directions. In Proceedings of the International Conference on Web Information Systems Engineering, Roma, Italy, 10–12 December 2003; p. 3. [CrossRef]
12. Erl, T. *Service-Oriented Architecture: Concepts, Technology, and Design*; Prentice Hall PTR: Upper Saddle River, NJ, USA, 2005.
13. Yuan, Y.; Wen, J.; Li, W.; Zhang, B. A Comparison of Three Programming Models for Telecom Service Composition. In Proceedings of the Third Advanced International Conference on Telecommunications (AICT'07), Morne, Mauritius, 13–19 May 2007; IEEE Computer Society: Washington, DC, USA, 2007.
14. Yelmo, J.; del Alamo, J.; Trapero, R.; Falcarm, P.; Yi, J.; Cairo, B.; Baladron', C. A user-centric service creation approach for Next Generation Networks. In Proceedings of the 2008 First ITU-T Kaleidoscope Academic Conference-Innovations in NGN: Future Network and Services, Geneva, Switzerland, 12–13 May 2008; pp. 211–218. [CrossRef]
15. Papazoglou, M.P.; Georgakopoulos, D. Introduction: Service-oriented computing. *Commun. ACM* **2003**, *46*, 24–28. [CrossRef]
16. Juric, M.B.; Mathew, B.; Sarang, P.G. *Business Process Execution Language for Web Services: An Architect and Developer's Guide to Orchestrating Web Services Using BPEL4WS*; Packt Publishing Ltd.: Birmingham, UK, 2006.
17. Wang, P.; Chao, K.M.; Lo, C.C.; Huang, C.L.; Li, Y. A fuzzy model for selection of QoS-aware web services. In Proceedings of the 2006 IEEE International Conference on e-Business Engineering (ICEBE'06), Shanghai, China, 24–26 October 2006; pp. 585–593.
18. Peltz, C. Web services orchestration and choreography. *Computer* **2003**, *36*, 46–52. [CrossRef]
19. Pistore, M.; Barbon, F.; Bertoli, P.; Shaparau, D.; Traverso, P. Planning and monitoring web service composition. In Proceedings of the Artificial Intelligence: Methodology, Systems, and Applications: 11th International Conference (AIMSA 2004), Varna, Bulgaria, 2–4 September 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 106–115.
20. Clarke, E.M.; Grumberg, O.; Peled, D. *Model Checking*; The MIT Press: Boston, MA, USA, 1999.
21. Gordon, M. From LCF to HOL: A short history. In *Proof, Language, and Interaction: Essays in Honour of Robin Milner*; The MIT Press: Boston, MA, USA, 2000; pp. 169–185.

22. Fernandez, J.; Jard, C.; Jéron, T.; Viho, C. An experiment in automatic generation of test suites for protocols with verification technology* 1. *Sci. Comput. Program.* **1997**, *29*, 123–146. [CrossRef]
23. Zhao, X.; Yang, H.; Qiu, Z. Towards the formal model and verification of web service choreography description language. In Proceedings of the Web Services and Formal Methods: Third International Workshop, WS-FM 2006, Vienna, Austria, 8–9 September 2006.
24. Broy, M. Towards a Formal Foundation of the Specification and Description Language SDL. *Form. Asp. Comput.* **1991**, *3*, 21–57. [CrossRef]
25. Bond, G. An introduction to ECharts: The concise user manual. *Transition* **2006**, *4*, 2.
26. Curbera, F.; Duftler, M.; Khalaf, R.; Nagy, W.; Mukhi, N.; Weerawarana, S. Unraveling the Web services web: An introduction to SOAP, WSDL, and UDDI. *IEEE Internet Comput.* **2002**, *6*, 86–93. [CrossRef]
27. Kennedy, S.; Molloy, O.; Stewart, R.; Jacob, P.; Maleshkova, M.; Doheny, F. A Semantically Automated Protocol Adapter for Mapping SOAP Web Services to RESTful HTTP Format to Enable the Web Infrastructure, Enhance Web Service Interoperability and Ease Web Service Migration. *Future Internet* **2012**, *4*, 372–395. [CrossRef]
28. Christensen, E.; Curbera, F.; Meredith, G.; Weerawarana, S. *Web Services Description Language (WSDL) 1.1*; Citeseer: Gaithersburg, MD, USA, 2001.
29. Box, D.; Ehnebuske, D.; Kakivaya, G.; Layman, A.; Mendelsohn, N.; Nielsen, H.; Thatte, S.; Winder, D. Simple Object Access Protocol. 2000. Available online: http://www.w3.org/TR/SOAP/ (accessed on 18 March 2020).
30. Fielding, R.T. *Architectural Styles and the Design of Network-Based Software Architectures*; University of California: Irvine, CA, USA, 2000.
31. Andrews, T.; Curbera, F.; Dholakia, H.; Goland, Y.; Klein, J.; Leymann, F.; Liu, K.; Roller, D.; Smith, D.; Thatte, S.; et al. Business Process Execution Language for Web Services. 2003. Available online: http://xml.coverpages.org/BPELv11-May052003Final.pdf (accessed on 10 July 2020)
32. Standard, OASIS Web Services Business Process Execution Language. 2007. Available online: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf (accessed on 7 March 2020).
33. Turner, K. Formalising web services. In Proceedings of the Formal Techniques for Networked and Distributed Systems-FORTE 2005: 25th IFIP WG 6.1 International Conference, Taipei, Taiwan, 2–5 October 2005; pp. 473–488.
34. Martin, D.; Burstein, M.; Hobbs, J.; Lassila, O.; McDermott, D.; McIlraith, S.; Narayanan, S.; Paolucci, M.; Parsia, B.; Payne, T.; et al. OWL-S: Semantic markup for web services. *W3C Memb. Submiss.* **2004**, *22*, 2007-04.
35. Milanovic, N.; Malek, M. Current solutions for Web service composition. *IEEE Internet Comput.* **2004**, *8*, 51–59. [CrossRef]
36. Elenius, D.; Denker, G.; Martin, D.; Gilham, F.; Khouri, J.; Sadaati, S.; Senanayake, R. The OWL-S editor–a development tool for semantic web services. In Proceedings of the Semantic Web: Research and Applications: Second European Semantic Web Conference (ESWC 2005), Heraklion, Greece, 29 May–1 June 2005; pp. 78–92.
37. Scicluna, J.; Abela, C.; Montebello, M. Visual modeling of owl-s services. In Proceedings of the IADIS International Conference WWW/Internet, Madrid, Spain, 6–9 October 2004; Citeseer: Gaithersburg, MD, USA, 2004.
38. Roman, D.; Keller, U.; Lausen, H.; de Bruijn, J.; Lara, R.; Stollberg, M.; Polleres, A.; Feier, C.; Bussler, C.; Fensel, D. Web service modeling ontology. *Appl. Ontol.* **2005**, *1*, 77–106.
39. Lausen, H.; de Bruijn, J.; Polleres, A.; Fensel, D. Wsml-a language framework for semantic web services. In Proceedings of the W3C Rules Workshop, Washington, DC, USA, 27–28 April 2005.
40. Paolucci, M.; Srinivasan, N.; Sycara, K. Expressing wsmo mediators in owl-s. In Proceedings of the Workshop on Semantic Web Services: Preparing to Meet the World of Business Applications Held at the 3rd International Semantic Web Conference (ISWC 2004), Hiroshima, Japan, 7–11 November 2004; Citeseer: Gaithersburg, MD, USA, 2004.
41. Dimitrov, M.; Simov, A.; Momtchev, V.; Konstantinov, M. WSMO Studio—A Semantic Web Services Modelling Environment for WSMO. In Proceedings of the Semantic Web: Research and Applications: 4th European Semantic Web Conference (ESWC 2007), Innsbruck, Austria, 3–7 June 2007; pp. 749–758.
42. Filipowska, A.; Haller, A.; Kaczmarek, M.; van Lessen, T.; Nitzsche, J.; Norton, B. Process ontology language and operational semantics for semantic business processes. In *Deliverable 1.3. Project IST 026850 SUPER*; Campus Essen: Essen, Germany, 2007.
43. Bansal, S.; Bansal, A.; Gupta, G.; Blake, M.B. Generalized semantic Web service composition. *Serv. Oriented Comput. Appl.* **2016**, *10*, 111–133. [CrossRef]
44. Liang, Q.A.; Chung, J.Y.; Miller, S. Modeling semantics in composite Web service requests by utility elicitation. *Knowl. Inf. Syst.* **2007**, *13*, 367–394. [CrossRef]
45. Medjahed, B.; Malik, Z.; Benbernou, S. On the composability of semantic web services. In *Web Services Foundations*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 137–160.
46. Paganelli, F.; Ambra, T.; Parlanti, D. A QoS-aware service composition approach based on semantic annotations and integer programming. *Int. J. Web Inf. Syst.* **2012**, *8*, 296–321. [CrossRef]
47. Paganelli, F.; Ambra, T.; Parlanti, D.; Giuli, D. A semantic-driven integer programming approach for QoS-aware dynamic service composition. In Proceedings of the 2011 50th FITCE Congress-" ICT: Bridging an Ever Shifting Digital Divide", Palermo, Italy, 31 August–3 September 2011; pp. 1–6.
48. Hassine, A.; Matsubara, S.; Ishida, T. A Constraint-Based Approach to Horizontal Web Service Composition. In Proceedings of the 5th International Semantic Web Conference (ISWC 2006), Athens, GA, USA, 5–9 November 2006.

49. Maamar, Z.; Lahkim, M.; Benslimane, D.; Thiran, P.; Sattanathan, S. Web Services Communities-Concepts & Operations. In Proceedings of the 3rd International Conference on Web Information Systems and Technologies (WEBIST'2007), Barcelona, Spain, 3–6 March 2007.
50. Maamar, Z.; Wives, L.K.; Al-Khatib, G.; Sheng, Q.Z.; De Vit, A.R.D.; Benslimane, D. From communities of web services to marts of composite web services. In Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, Perth, Australia, 20–23 April 2010; pp. 958–965.
51. Marques, V.; Casado, A.C.; Moreno, J.I.; Aguiar, R.L.; Chaher, N. A simple QoS service provision framework for beyond 3rd generation scenarios. In Proceedings of the 10th International Conference on Telecommunications, Tahiti, Papeete, 23 February–1 March 2003; Volume 2, pp. 1475–1481.
52. Li, M.; Zhu, D.; Deng, T.; Sun, H.; Guo, H.; Liu, X. GOS: A global optimal selection strategies for QoS-aware web services composition. *Serv. Oriented Comput. Appl.* **2013**, *7*, 181–197. [CrossRef]
53. Lan, C.W.; Chen, R.C.; Su, A.Y.; Huang, A.F.; Yang, S.J.; Chung, J.Y. A multiple objectives optimization approach for QoS-based web services compositions. In Proceedings of the 2009 IEEE International Conference on e-Business Engineering, Macau, China, 21–23 October 2009; pp. 121–128.
54. Zhang, Y.; Panahi, M.; Lin, K.J. Service process composition with QoS and monitoring agent cost parameters. In Proceedings of the 2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, Washington, DC, USA, 21–24 July 2008; pp. 311–316.
55. Paganelli, F.; Parlanti, D. A dynamic composition and stubless invocation approach for information-providing services. *IEEE Trans. Netw. Serv. Manag.* **2013**, *10*, 218–230. [CrossRef]
56. Parejo, J.A.; Segura, S.; Fernandez, P.; Ruiz-Cortés, A. Qos-aware web services composition using grasp with path relinking. *Expert Syst. Appl.* **2014**, *41*, 4211–4223.
57. Ramírez, A.; Parejo, J.A.; Romero, J.R.; Segura, S.; Ruiz-Cortés, A. Evolutionary composition of QoS-aware web services: A many-objective perspective. *Expert Syst. Appl.* **2017**, *72*, 357–370. [CrossRef]
58. Kazmi, S.A.; Tran, N.H.; Ho, T.M.; Hong, C.S. Hierarchical matching game for service selection and resource purchasing in wireless network virtualization. *IEEE Commun. Lett.* **2017**, *22*, 121–124. [CrossRef]
59. Al-Fuqaha, A.; Rayes, A.; Guizani, M.; Khanvilkar, M.; Ahmed, M. Intelligent service monitoring and support. In Proceedings of the 2009 IEEE International Conference on Communications, Dresden, Germany, 14–18 June 2009; pp. 1–6.
60. Do, C.T.; Tran, N.H.; Huh, E.N.; Hong, C.S.; Niyato, D.; Han, Z. Dynamics of service selection and provider pricing game in heterogeneous cloud market. *J. Netw. Comput. Appl.* **2016**, *69*, 152–165.
61. Jih, W.R.; Hsu, J.Y.J.; Wu, C.L.; Liao, C.F.; Cheng, S.Y. A multi-agent service framework for context-aware elder care. In Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, 8–12 May 2006.
62. Greenshpan, O.; Milo, T.; Polyzotis, N. Autocompletion for mashups. *Proc. VLDB Endow.* **2009**, *2*, 538–549. [CrossRef]
63. Foster, I.; Kesselman, C.; Nick, J.M.; Tuecke, S. Grid services for distributed system integration. *Computer* **2002**, *35*, 37–46. [CrossRef]
64. Chao, K.M.; Younas, M.; Griffiths, N.; Awan, I.; Anane, R.; Tsai, C.F. Analysis of grid service composition with BPEL4WS. In Proceedings of the 18th International Conference on Advanced Information Networking and Applications, Fukuoka, Japan, 29–31 March 2004; Volume 1, pp. 284–289.
65. Chen, Z.; Lin, H.; Chen, M.; Liu, D.; Bao, Y.; Ding, Y. A framework for sharing and integrating remote sensing and GIS models based on Web service. *Sci. World J.* **2014**, *2014*, 354919. [CrossRef] [PubMed]
66. Lemmens, R.; Wytzisk, A.; de By, R.; Granell, C.; Gould, M.; Van Oosterom, P. Integrating semantic and syntactic descriptions to chain geographic services. *IEEE Internet Comput.* **2006**, *10*, 42–52. [CrossRef]
67. Balbiani, P.; Cheikh, F. *Computational Analysis of Interactiong Web Services: A Logical Approach*; IRIT Institut de Recherche en Informatique de Toulouse: Toulouse, France, 2006.
68. Zhang, R.; Arpinar, I.; Aleman-Meza, B. Automatic composition of semantic web services. In Proceedings of the 1st International Conference on Web Services, Las Vegas, NV, USA, 23–26 June 2003; Citeseer: Gaithersburg, MD, USA, 2003; pp. 38–41.
69. Lécué, F.; Silva, E.; Pires, L. A framework for dynamic web services composition. *Emerg. Web Serv. Technol.* **2008**, *2*, 59–75.
70. Zhovtobryukh, D. *Context-Aware Web Service Composition*; University of Jyvaskyla: Jyvaskyla, Finland, 2006.
71. Waluyo, A.B.; Taniar, D.; Rahayu, W.; Srinivasan, B. A Dual Privacy Preserving Approach for Location-Based Services in Mobile Multicast Environment. *Mob. Netw. Appl.* **2018**, *23*, 34–43. [CrossRef]
72. Paganelli, F.; Ulema, M.; Martini, B. Context-aware service composition and delivery in NGSONs over SDN. *IEEE Commun. Mag.* **2014**, *52*, 97–105. [CrossRef]
73. Liu, X.; Hui, Y.; Sun, W.; Liang, H. Towards service composition based on mashup. In Proceedings of the 2007 IEEE Congress on Services (Services 2007), Salt Lake City, UT, USA, 9–13 July 2007; pp. 332–339.
74. Meira, S.R.; Buregio, V.A.; Nascimento, L.M.; Figueiredo, E.; Neto, M.; Encarnação, B.; Garcia, V.C. The emerging web of social machines. In Proceedings of the 2011 IEEE 35th Annual Computer Software and Applications Conference, Munich, Germany, 18–22 July 2011; pp. 26–27.

75. Lord, P.; Alper, P.; Wroe, C.; Goble, C. Feta: A light-weight architecture for user oriented semantic service discovery. In Proceedings of the Semantic Web: Research and Applications: Second European Semantic Web Conference, ESWC 2005, Heraklion, Greece, 29 May–1 June 2005; pp. 17–31.
76. Law, T. Social Scripting for the Web. *Computer* **2007**, *40*, 96–98.
77. Kengne Kungne, W.; Kouamou, G.E.; Tangha, C. A rule-based language and verification framework of dynamic service composition. *Future Internet* **2020**, *12*, 23. [CrossRef]
78. Chen, L.; Shadbolt, N.; Goble, C.; Tao, F.; Cox, S.; Puleston, C.; Smart, P. Towards a knowledge-based approach to semantic service composition. In Proceedings of the Semantic Web-ISWC 2003: Second International Semantic Web Conference, Sanibel Island, FL, USA, 20–23 October 2003; pp. 319–334.
79. Abuhussein, A.; Shiva, S.; Sheldon, F.T. CSSR: Cloud services security recommender. In Proceedings of the 2016 IEEE world congress on services (SERVICES), San Francisco, CA, USA, 27 June–2 July 2016; pp. 48–55.
80. DiBernardo, M.; Pottinger, R.; Wilkinson, M. Semi-automatic web service composition for the life sciences using the BioMoby semantic web framework. *J. Biomed. Inform.* **2008**, *41*, 837–847. [CrossRef] [PubMed]
81. Yu, S.; Woodard, C.J. Innovation in the Programmable Web: Characterizing the Mashup Ecosystem. In Proceedings of the Service-Oriented Computing—ICSOC 2008 Workshops: ICSOC 2008 International Workshops, Sydney, Australia, 1 December 2009; pp. 136–147. [CrossRef]
82. Yu, J.; Benatallah, B.; Saint-Paul, R.; Casati, F.; Daniel, F.; Matera, M. A framework for rapid integration of presentation components. In Proceedings of the 16th International Conference on World Wide Web, Banff, AB, Canada, 8–12 May 2007; pp. 923–932.
83. Ennals, R.; Garofalakis, M.N. MashMaker: Mashups for the masses. In Proceedings of the SIGMOD Conference, Beijing, China, 11–14 June 2007; pp. 1116–1118.
84. Ennals, R.; Gay, D. User-friendly functional programming for web mashups. *SIGPLAN Not.* **2007**, *42*, 223–234. [CrossRef]
85. Díaz, O.; Pérez, S.; Paz, I. Providing personalized mashups within the context of existing web applications. In Proceedings of the Web Information Systems Engineering–WISE 2007, Nancy, France, 3–7 December 2007; pp. 493–502.
86. Wong, J.; Hong, J. Making mashups with marmite: Towards end-user programming for the web. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, 30 April–3 May 2007; pp. 1435–1444.
87. Soriano, J.; Lizcano, D.; Cañas, M.; Reyes, M.; Hierro, J. Fostering innovation in a mashup-oriented enterprise 2.0 collaboration environment. *UK sai: sisn* **2007**, *24*, 62–68.
88. Bosca, A.; Ferrato, A.; Corno, F.; Congiu, I.; Valetto, G. Composing Web services on the basis of natural language requests. In Proceedings of the 2005 IEEE International Conference on Web Services, Orlando, FL, USA, 11–15 July 2005; p. 2. [CrossRef]
89. Shiaa, M.; Falcarin, P.; Pastor, A.; Lécué, F.; Silva, E.; Pires, L.F. Towards the automation of the service composition process: Case study and prototype implementations. In Proceedings of the ICT Mobile and Wireless Communications Summit, Stockholm, Sweden, 10–12 June 2008.
90. Erlewine, M. Ubiquity: Designing a Multilingual Natural Language Interface. In Proceedings of the SIGIR 2009 Workshop on Information Access in a Multilingual World, Boston, MA, USA, 23 July 2009; pp. 45–48.
91. Yu, J.; Benatallah, B.; Casati, F.; Daniel, F. Understanding Mashup Development. *IEEE Internet Comput.* **2008**, *12*, 44–52. [CrossRef]
92. Grammel, L.; Storey, M. *An End User Perspective on Mashup Makers*; Tech. Rep. DCS-324-IR; University of Victoria: Victoria, UK, 2008.
93. Hoyer, V.; Fischer, M. Market Overview of Enterprise Mashup Tools. In Proceedings of the 6th International Conference on Service-Oriented Computing, Sydney, Australia, 1–5 December 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 708–721.
94. Floyd, I.R.; Jones, M.C.; Rathi, D.; Twidale, M.B. Web Mash-ups and Patchwork Prototyping: User-driven technological innovation with Web 2.0 and Open Source Software. In Proceedings of the 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07), Big Island, HI, USA, 3–6 January 2007; IEEE Computer Society: Washington, DC, USA, 2007; p. 86. [CrossRef]
95. Jones, M.C.; Churchill, E.F. Conversations in developer communities: A preliminary analysis of the yahoo! pipes community. In Proceedings of the Fourth International Conference on Communities and Technologies: Communities and Technologies 2009 (CCT '09), State College, PA, USA, 25–27 June 2009; ACM: New York, NY, USA, 2009; pp. 195–204. [CrossRef]
96. Soriano, J.; Lizcano, D.; Hierro, J.J.; Reyes, M.; Schroth, C.; Janner, T. Enhancing User-Service Interaction through a Global User-Centric Approach to SOA. In Proceedings of the Fourth International Conference on Networking and Services (ICNS '08), Gosier, Guadeloupe, 16–21 March 2008; IEEE Computer Society: Washington, DC, USA, 2008; pp. 194–203. [CrossRef]
97. Maamar, Z.; Wives, L.; Badr, Y.; Elnaffar, S. Even Web Services Can Socialize: A New Service-Oriented Social Networking Model. In Proceedings of the 1st International Conference on Intelligent Networking and Collaborative Systems (INCoS 2009), Barcelona, Spain, 4–6 November 2009; pp. 24–30.
98. Skraba, R.; Beauvais, M.; Stan, J.; Maaradji, A.; Daigremont, J. Developing compelling social-enabled applications with context-based social interaction analysis. In Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining, Athens, Greece, 20–22 July 2009; pp. 206–211.
99. Wasserman, S.; Faust, K. *Social Network Analysis: Methods and Applications*; Cambridge University Press: Cambridge, MA, USA, 1994.
100. Wen, S.; Li, Q.; Yue, L.; Liu, A.; Tang, C.; Zhong, F. CRP: Context-based reputation propagation in services composition. *Serv. Oriented Comput. Appl.* **2012**, *6*, 231–248. [CrossRef]

101. Lee, J.; Zhang, J.; Huang, Z.; Lin, K.J. Context-based reputation management for service composition and reconfiguration. In Proceedings of the 2012 IEEE 14th International Conference on Commerce and Enterprise Computing, Hangzhou, China, 9–11 September 2012; pp. 57–61.
102. Suppa, P.; Zimeo, E. A context-aware mashup recommender based on social networks data mining and user activities. In Proceedings of the 2016 IEEE International Conference on Smart Computing (SMARTCOMP), St. Louis, MO, USA, 18–20 May 2016; pp. 1–6.
103. Bouadjenek, M.R.; Hacid, H.; Bouzeghoub, M. Social networks and information retrieval, how are they converging? A survey, a taxonomy and an analysis of social information retrieval approaches and platforms. *Inf. Syst.* **2016**, *56*, 1–18. [CrossRef]
104. Hacid, H.; Hebbar, K.; Maaradji, A.; Saidi, M.A.; Ribière, M.; Daigremont, J. Enhancing navigation in virtual worlds through social networks analysis. In Proceedings of the International Symposium on Methodologies for Intelligent Systems, Warsaw, Poland, 28–30 June 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 146–152.
105. Maaradji, A.; Hacid, H.; Daigremont, J.; Crespi, N. Towards a Social Network Based Approach for Services Composition. In Proceedings of the 2010 IEEE International Conference on Communications (ICC 2010), Cape Town, South Africa, 23–27 May 2010.
106. Maaradji, A.; Hacid, H.; Skraba, R.; Vakali, A. Social web mashups full completion via frequent sequence mining. In Proceedings of the 2011 IEEE World Congress on Services, Washington, DC, USA, 4–9 July 2011; pp. 9–16.
107. Mesfin, G.; Ghinea, G.; Grønli, T.M.; Younas, M. Web Service Composition on Smartphones: The Challenges and a Survey of Solutions. In Proceedings of the International Conference on Mobile Web and Intelligent Information Systems, Barcelona, Spain, 6–8 August 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 126–141.
108. Zhang, W.; Zhang, Y.; Wu, Q.; Peng, K. Mobility-enabled edge server selection for multi-user composite services. *Future Internet* **2019**, *11*, 184. [CrossRef]
109. Nunes, D.; Tran, T.D.; Raposo, D.; Pinto, A.; Gomes, A.; Silva, J.S. A web service-based framework model for people-centric sensing applications applied to social networking. *Sensors* **2012**, *12*, 1688–1701. [CrossRef] [PubMed]
110. Nath, K.; Iswary, R. What comes after Web 3.0? Web 4.0 and the Future. In Proceedings of the International Conference and Communication System (I3CS'15), Shillong, India, 28–30 April 2015; Volume 337, p. 341.
111. Algosaibi, A.A.; Albahli, S.; Melton, A. World Wide Web: A survey of its development and possible future trends. In Proceedings of the 16th International Conference on Internet Computing and Big Data-ICOMP, Las Vegas, NV, USA, 27 July 2015; Volume 15, pp. 79–84.