2015 IEEE International Conference on Web Services

Supporting End-User Service Composition: A Systematic Review of Current Activities and Tools

Feifei Hang, Liping Zhao
School of Computer Science
The University of Manchester
Manchester, UK
{hangf, lzhao}@cs.man.ac.uk

Abstract—**This paper presents a systematic literature review of end-user service composition. It reviews current activities performed by end users, and tools and approaches that enable them to compose and develop service systems from Web Services. The paper also highlights some key open research issues for the future.**

Keywords –Systematic literature review, end user, end-user service composition, end-user development, end-user programming, end-user computing, service composition platform.

I. Introduction

End-user service composition (EUSC) refers to activities and tools that allow end users – people who are not professional software developers – to compose service systems from Web Services without significant knowledge of a programming language. Armed with EUSC tools, end users can also create, modify or extend Web Services.

EUSC is an instance of end-user development (EUD) [2], which has been an active research topic within the field of computer science and human-computer interaction. Most cited examples of EUD include spreadsheet programming, scripting languages and programming by example[1]. The main aim of EUD is to bring programming closer to the needs of end users. EUD is also called end-user programming (EUP), end-user computing (EUC) and end-user software engineering (EUSE) [5].

In recent years, EUSC tools have begun to emerge. For example, AMICO:CALC [8] provides tools for end users to compose primitive services in a spreadsheet manner [8], whereas Hypermash [13] offers a heterogeneous service composition platform to allow end users to compose their own ad-hoc services from both SOAP and RESTful services. Easy SOA [15] enables end users to rapidly create prototypes of composite services, whereas Co-Taverna [17] and Confucius [19] provide collaborative workbenches to support the collaboration between end users.

While EUSC will continue to be an important part of service systems development, a systematic study of current activities and tools is still missing. We believe that such a study is imperative as it will not only provide a detailed picture of current activities and tools in this area, but also identify some key research issues for future development. The present paper thus aims to make a contribution to such a study.

The remaining paper has been organized as follows. Section II defines our study method, research questions, search strategy, and literature selection. Section III presents our literature review whereas Section IV analyzes the results. Finally Section V highlights some key open research issues for the future.

II. Systematic Review of EUSC

A. Method

We have adopted a systematic literature review (SLR) to our study of current activities and tools of EUSC. A SLR is a literature review approach that "synthesizes existing work in a manner that is fair and seen to be fair [21]". In comparison with conventional expert literature reviews, SLRs have the following characteristics:

- SLRs start by defining a review protocol that specifies the research question being addressed and the methods that will be used to perform the review.
- SLRs are based on a defined search strategy that aims to detect as much of the relevant literature as possible.
- SLRs document their search strategy so that readers can assess their rigor and the completeness and repeatability of the process.
- SLRs require explicit including and exclusion criteria to assess each potential primary study.

There are many reasons for undertaking a SLR. The most common reasons are [21]:

- To summarize the existing evidence concerning a treatment or technology.
- To identify any gaps in current research in order to suggest areas for further investigation.
- To provide a framework/background in order to appropriately position new research activities.

There are different processes for conducting a SLR. Our process derives from "Guidelines for Performing Systematic Literature Reviews in Software Engineering [21]" and consists of the following main steps:

1. Define the research questions.
2. Define the search strategy.
3. Select the literature.
4. Review the literature to identify current activities and tools for EUSC.
5. Answer the research questions.

1 http://en.wikipedia.org/wiki/End-user_development

TABLE I. SEARCH RESULTS OF THE SYSTEMATIC LITERATURE REVIEW

Conference/Journal Name	Acronym	Location	Result (# Papers)
IEEE International Conference on Web Services	ICWS	http://goo.gl/aGTyfE http://goo.gl/KPWK9i http://goo.gl/qBLXfE http://goo.gl/GP93WW http://goo.gl/qC76Nu http://goo.gl/LvxyJq http://goo.gl/ISXg5s http://goo.gl/Nw2vWE http://goo.gl/KvOk9d http://goo.gl/yHPgHR	18
IEEE International Conference on Services Computing	SCC	http://goo.gl/KWdRf0 http://goo.gl/0MTW4m http://goo.gl/AksZDb http://goo.gl/vRwtbn http://goo.gl/S81N6B http://goo.gl/0kYpTX http://goo.gl/TaJvDX http://goo.gl/sFhuXm http://goo.gl/P79KSk http://goo.gl/4UabcR	9
IEEE European Conference on Web Services	ECOWS ICWS-Europe	http://goo.gl/n8eMHU http://goo.gl/ZaZne7 http://goo.gl/xJ1z42 http://goo.gl/Upo9D3 http://goo.gl/Y8gOzk http://goo.gl/weUTrC http://goo.gl/ajzDfk http://goo.gl/3J1xzh http://goo.gl/lvMCz5	4
International World Wide Web Conference	WWW	http://goo.gl/1P17kD http://goo.gl/q23wgz http://goo.gl/drTwqy http://goo.gl/TqDhhf http://goo.gl/aCkqdg http://goo.gl/lPy1Bg http://goo.gl/cxRfHf http://goo.gl/ihUAt http://goo.gl/px6v95 http://goo.gl/77UevV http://goo.gl/Ab6TT0 http://goo.gl/3BgZfUl http://goo.gl/602af0 http://goo.gl/OkEER7 http://goo.gl/JI0ATj http://goo.gl/A7UyC8	5
International Conference on Service Oriented Computing	ICSOC	http://goo.gl/yiAmbh http://goo.gl/eMB8f1 http://goo.gl/pBe8tz http://goo.gl/CKFfrB http://goo.gl/xWztu9 http://goo.gl/GgToAG http://goo.gl/qIXaWz http://goo.gl/s0lNRO http://goo.gl/gkOeWM http://goo.gl/KlhvrS http://goo.gl/uHTCgV	9
IEEE Transactionson Services Computing	TSC	http://www.computer.org/csdl/trans/sc/index.html	2
Total			47

B. *Research Questions*

Specifying the research questions is the most important part of SLRs as they drive the entire systematic review methodology [21]. Our SLR is guided by the following three research questions:

1. What service composition platforms have been proposed over the past decade?
2. What EUSC activities are supported by these platforms?
3. What approaches and tools are used to enable these EUSC activities?

C. *Search Strategy*

The aim of a SLR is to find as many primary studies relating to the research questions as possible using an unbiased search strategy [21]. The search strategy should define the search terms, search scopes, and resources to be searched.

Our search strategy was defined as follows:

Search Scope:

- Proceedings of five top international conferences on service systems: IEEE International Conference on Web Services (ICWS 2004 – ICWS 2013). IEEE International Conference on Services Computing (SCC 2004 – SCC 2013). European Conference on Web Services (ECOWS 2003 – ECOWS 2011). International Conference on Service Oriented Computing (ICSOC 2003 - ICSOC 2013). International Conference on World Wide Web (WWW 2003 – WWW 2013).
- IEEE Transactions on Services Computing (TSC 2008 – TSC 2014).

Search Resources:

- IEEE Xplore Digital Library (for ICWS, SCC, ECOWS);
- IEEE Computer Society Digital Library (for TSC);
- Springer Link (for ICSOC, ICWS-Europe 2003, and ECOWS 2004);
- ACM Digital Library (for WWW, and ICSOC 2004);

Search Terms:

- "Proceedings of IEEE International Conference on Web Services, 2007", "Proceedings of IEEE International Conference on Web Services, 2008", etc.

D. *Literature Selection*

Based on the search strategy, we located these proceedings and journal articles. We then applied the following inclusion and exclusion criteria to filter out irrelevant papers and select relevant ones:

1. Papers focusing on service composition environments and platforms were included, whereas papers concentrating on single service composition enabling technology, such as automatic composition algorithms, were excluded.
2. Long papers listed under, for example, "Research Track", "Application Track", "Industrial Track", and "Experience Track" were included.
3. For the short papers listed under "Work in Progress Track", "Poster Track", and "Demonstration Track", the ones providing sufficient details of their proposed systems to be analyzed, e.g. [25], were included, whereas the ones providing insufficient details of their works to be analyzed, e.g. [26], were excluded.
4. Literature review or survey papers were excluded as our focus was on the primary studies.
5. Preface and postface articles were excluded.

We performed this filtering process on the references hosted in the aforementioned digital libraries. We then downloaded the citations of all the relevant papers and imported them into an EndNote library. Duplicates were discarded automatically. In total, 47 papers have been downloaded. The overview of our search results is summarized in Table I. For space consideration, all the resource locations (except TSC's) in Table I are shortened by using Google URL Shortener[2].

III. Literature Review: Current EUSC Activities and Tools

We have organized our studies into five categories, according to the types of activity they describe. These categories are service composition, service design, service reuse, service testing, and service debugging. This classification is in line with the traditional software engineering lifecycle [5] and the highly-cited SOA model [28]. This section presents our review of these activities, their associated approaches and tools. .

A. Service Composition

This activity concerns creating new composite services by composing existing primitive services, data or processes to accomplish an end user's task.

According to our study, service composition is supported by two main approaches, namely static and dynamic service composition. These approaches and their associated tools are described as follows.

1) Static Service Composition. In static service composition, the primitive services, composition logics, and data and control flows are manually selected and designed by users. The composite services at runtime will strictly follow the working procedures and configurations that being specified at design time.

The most adopted approach to supporting static service composition is to allow end users define their composite service in the manner of *drawing workflow diagrams*. For example, in WSCE [1], end users can define the BPEL process diagram of their composite service with drag and drop tool. In Triana [3], users can use the graphical workflow component to design workflows at a conceptual level. Similarly, in HyperMash [13], Baya [24], graphical workbenches are provided to users to draw the workflow diagrams.

This "workflow diagram" approach has also been extensively used in scientific domain. For example, Kepler [4], Co-Taverna [17], Confucius [19], and VIEW [12] all provide graphical workflow editor to users to help them define their scientific workflows.

As another popular approach to support static service composition, *spreadsheet-based* service composition has also attracted significant research efforts. For example, AMICO:CALC [8], Mashroom [27], MashSheet [30], DataSheets [31], and the platform proposed by Kongdenfha et al. [29] all allow users to compose services in a spreadsheet manner.

In Easy SOA [15], Synthy IDE [32] and DoCoSOC [33], a *wizard- and form-based* approach has been utilized to help users define their composite service by providing the key information, such as locations of each primitive services, the invocation orders of each services, and the output format of the runtime execution result, to the systems.

2 https://goo.gl/

In addition, some platforms such as iMashup [35] and the one developed by Xiang and Madey [34] provide users *WYSIWYG* editors to allow them instantly inspect on the final look and feel of their composite services.

2) Dynamic Service Composition. Dynamic service composition composes an application autonomously when a user queries for an application. Comparing with static service composition, dynamic service composition has the potential to realize flexible and adaptable applications by properly selecting and combining components based on the user request and context.

Instead of expecting users to manually write semantic service composition documents, providing users with *high-level graphical language* to easily define their desired composite services has been used as one of the approaches supporting dynamic service composition. For example, in VINCA [36], a high-level graphical editor is developed to allow end users define their services at business-level. In Flow Editor [38], a graphical editor is built to help users describe their ideal composite services in a flow-chart manner, whereas the graphical Reo Coordination Language is adopted to define composite services in the platform produced by Saifipoor et al. [37].

Similarly, a *visualization* approach is applied in the visualizer component of the system proposed by Rao et al. [39] to represent the user-defined composition logic graphically.

A *wizard-based* approach is utilized by the SMS system [40], and MARIO [41] to enable users to incrementally refine their composition requests.

Moreover, SeGSeC [42] supports dynamic service composition by providing *natural language processing* ability to analyze users requests written in natural language documents.

B. Service Design

In EUSC, this activity concerns designing services according to some needs of end users. According to our study, this activity involves design process support and design by example, described as follows.

1) Design Process. Design process concerns how requirements or design ideas are translated into design specifications and then implementation.

In the domain of service composition, some have proposed to support design processes by *constraining what can be designed to a particular domain*. For example, by applying BPEL, WSCE [1] provides end users a visual editor to create composite service with BPEL constraints, whereas the Public-oriented Healthcare Information Services Platform (PHISP) [44] adopts UML activity diagram as a manner to constraining how a composite service can be defined. Similarly, METERO-S inherits the domain constraints of BPEL4WS in its Abstract Process Designer [43]. The SMS system adopts GEM (Guidelines Element Model) and CPGA (Clinical Practice Guideline Architecture) standards as pragmatic constraints for service selection [40]. Easy SOA uses "cards", a set of predefined domain template, to constrain what can be assembled into a particular composite service [15]. Flow Editor constrains user behaviors by automatically determining if two primitive services can be composed together [38].

Moreover, other platforms, such as AMICO:CALC [8], MashSheet [30], DataSheets [31], and the mashup platform proposed by Kongdenfha et al. [29], leverage the power of spreadsheet systems to constrain what services and data processing procedures can be composed together by using the predefined spreadsheet formulas. Also, the “WireIt” approach[3] has been utilized in DISC [22] and SoCo [23] to provide domain constraints.

An alternative approach to support design processes is to *tolerate a design to be partially or imprecisely stated.* For example, Xiang and Madey [34] proposed a system that allows end users to provide only input and the goal to be achieved, and system will then attempt to automatically create a composite service for the end users. Saifipoor et al. [37] developed a service composition framework based on Reo coordination language to allow the primitive services and their coordination processes to be known explicitly only until run time. Also, Riabov et al. [41] and Shiaa et al. [45] built service composition platforms to allow users to incrementally refine their goals when creating composite services.

In addition, design processes can also be supported by *asynchronous or synchronous collaborations between users*. Triana [3], DoCoSOC [33], HyperMash [13], Co-Taverna [17], and Confucius [19] are all built to enable multiple users to work either synchronously or asynchronously on a same mashup project. It is important to understand that both asynchronous and synchronous collaborations should be explicitly assisted by the system as a function/feature, which means the collaboration achieved by, for example, making a copy of a project in an USB disk and then handing in it to another person does not count.

2) Design by Example. To better support users to produce their specifications, the approach of “*programming by example*” has been used in the existing service composition platforms to support design adaptation. For example, HyperMash [13], Coins [9], Mashroom [27], and e-BioFlow [14] can record the composition logics and configurations of a composite service defined by an end user and then automatically retrieve and reapply these logics and configurations in other composite services that have similarities to the existing one. Additionally, this feature is also witnessed in SOA4ALL Composer [20], and Baya [24].

C. Service Reuse

This activity involves compositing new services by reuse of existing services. It refers to either a form of composition, such as “gluing” together existing primitive services, or modifications, such as changing some configurations of existing composite services to achieve a new goal. We have identified three common approaches for service reuse, described as follows.

1) Reusing Services. The study of students prototyping user interface [47] showed that even if end users are able to find reusable abstractions in the form of, for example, primitive services, they may still have difficulty using them.

To address this issue, one solution is to *simply modify the configuration of an existing composite service, customizing it for a particular purpose*. For example, the Ad-hoc Editor of e-BioFlow [14] can help end users to find new tasks (i.e. either primitive services or data processing procedures) to extend or modify existing composite services to satisfy new requirements and goals. The ontology-based yellow page registry of IRIS [46] can help users to modify composite services by reusing “mediator” components.

Meanwhile, a lot of efforts have also been put on a *template-based approach* to support service reusing. With helps of template, end users can easily identify the modifiable parts of their works, and then modify them to meet their requirements. For example, in METERO-S [43], the service templates allow users to either bind to a known Web service or specify a semantic description of the Web service for their purposes. In AMICO:CALC [8], a group of services and their configurations can be abstracted as a “middleware” (i.e. a template) to be reused and modified in other mashup projects. Also, each Application Model Editor tool generated by DoCoSOC [33] is essentially a domain-specific template to be modified by users.

Similarly, this template-based supporting approach has also been adopted in SCE [7], ServFace [48], SOA4ALL Composer [20], and Baya [24].

2) Sharing and Distributing Services. In addition to reusing existing services, some service composition platforms also help end users to *share and distribute their works*.

In Triana [3] and Synthy IDE [32], users can package and share their whole workflows as primitive services to be composed with other services. Kepler [4] also allows users to deploy their scientific workflows as primitive services in other applications for compute-intensive tasks.

As suggested by Mackay [51], some programming efforts made by end users becomes quite long-lived, even though it is common for end users to view their ad-hoc application as “throw away”. Therefore, some systems start to provide users the feature of supporting *unplanned sharing* of their works. For example, the social-network based service recommendation component of SoCo [23] helps users distribute their mashups even implicitly, whereas all the user-created mashup patterns in Baya [24] will be potentially retrieved and reused by others through its recommendation server. Additionally, all the public mashup scripts are also being distributed among users in both HyperMash [13] and Mashroom [27].

3) Providing Right Service Composition Abstraction for End Users. Another well-known and wide-applied approach to facilitate the reuse by end users is to choose the right abstractions for their problem domains.

3 WireIt is an open-source JavaScript library, to create full-web graph editors for dataflow applications, visual programming languages, graphical modeling, or graph editors. Available at: http://neyric.github.io/wireit/docs/

Since service composition is essentially about to reuse existing services to achieve users' goals, a large number of service composition platforms provide an appropriate level of abstractions of problem domains to help users to reuse existing services. For example, by utilizing a process-oriented language, VINCA [36] can visually capture users' needs at a high abstraction level. Similarly, the natural language processing component of SeGSeC [42] allows users to only issue their service composition requests in natural language without worrying about the technical details of their composite services. The "card" concept introduced by Easy SOA [15] provides a high-level domain abstraction to users and hides the low-level implementation details of service composition technologies from users.

The approach to providing decent domain abstractions has also been widely adopted in logic-, constraint-, and semantic-driven platforms. For instance, Rao et al. [39] developed a translator component in their system to translate external (high-level) DAML-S specifications of Web services into internal (low-level) LL axioms. METERO-S [43] utilizes a constraint analyzer to dynamically select services from service repository for users. WSMX [49] allows users to only specify the high-level goals of their composite services by using Web Service Modeling Ontology (WSMO).

D. *Service Testing*

This activity concerns identifying the failures and bugs of the composite services. According to our study, exisiting service composition platforms support end-user testing through the following four approaches:

1) Immediate Feedback. Claimed by Panko [54, 55] and Hendry and Green [56], end users are notoriously overconfident about the correctness of their works. To address this issue, a widely adopted solution in service composition platforms is to *immediate feedback* about the values of a workflow returns. For example, in the SMS system [40], Triana [3], METERO-S [43], Synthy IDE [32], and AMICO:CALC [8], users are able to review the immediate feedbacks of their composite services with the help of GUIs when defining the corresponding workflows. Similarly, Web2Exchange [16], iMashup [35], Baya [24], and the automated service composition approach proposed by Liu et al. [11] also provide immediate feedback mechanisms to help users to get more objective and accurate level of confidence.

2) Maximizing Test Coverage. Research on testing tools for end users has focused on testing approaches that are integrated with users' work and are incremental in their feedback. The most notable and mature approach is the *"What You See Is What You Test" (WYSIWYT)* methodology to track and graphically represent test coverage for end users. However, according to our study, WSCE [1] is the only service composition platform providing WYSIWYT support by generating "pseudo Web services" for testing.

3) Checking Against Design Specification. Another approach to test and verify composite services is by *checking runtime statuses and execution results of composite services against predefined specifications*.

In WSCE [1], "inspectors" are used to check runtime execution results of composite services against the user-defined specifications written in low-level source code. In Astro [6] and VIEW [12], service monitors are applied to monitor the violation of the monitored properties of composite services.

Other platforms, such as WSMX [49], COLQUIDE [50], SCE [7], SCENE [57], Synthy IDE [32], and the graph-based service composition platform proposed by Shiaa et al. [45], also allow users to define detailed specifications of their composite services, and help users to check the states of their services against the predefined rules.

4) Visualization. Another way that being utilized to analyze the correctness of a composite service is to visualize its behavior. According to our study, iMashup [35] is the only platform providing visualization support to users. It visualizes the behavior of a composite service in the form of workflow diagram to help users detect design issues.

E. *Debugging*

Whereas service testing detects the presence of errors, debugging is the process of locating and removing errors. Yet, debugging is the least supported EUSC activity in all the platforms that we reviewed. To the best of our knowledge, HyperMash [13] and WSMX [49] are the only two platforms that help users to debug their works by providing change suggestions.

Table II summarizes the EUSC activities, their associated approaches and tools.

IV. Answering Research Questions

A. *What service composition platforms have been proposed over the past decade?*

According to our study, 47 service composition platforms have been proposed or built to support EUSC since 2003. A complete list of these platforms is summarized in Table II.

Figure 1 shows that only 1 service composition platform was proposed in 2003. In 2004, the number of proposed platforms dramatically increased to 9, and became the highest through out the entire decade. The second and third peeks were witnessed in 2008 and 2010, which have 7 and 6 papers published, respectively.

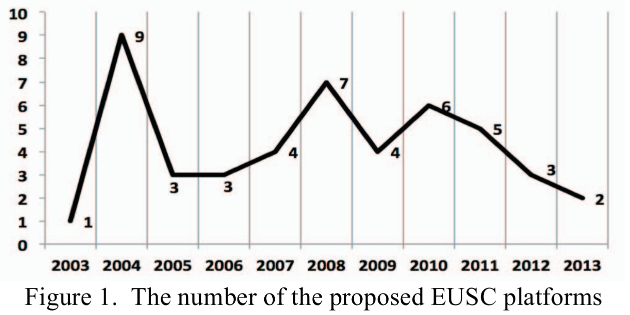

Figure 1. The number of the proposed EUSC platforms per year.

TABLE II. EUSC ACTIVITIES, APPROACHES, TOOLS, AND SERVICE COMPOSITION PLATFORMS

Activities	Approaches	Tools	Platforms / Citations
Service Composition	Static Service Composition	Workflow Diagram Editor	Yu, et al. [1], Majithia, et al. [3], Altintas, et al. [4], Trainotti, et al. [6], Braem, et al. [7], Grechanik, et al. [9], Carlson, et al. [10], Liu, et al. [11], Lin, et al. [12], Wassink, et al. [14], Srinivasmurthy, et al. [16], Namoun, et al. [18], Mehandjiev, et al. [20], Zahoor, et al. [22], Zhang [17], Maaradji, et al. [23], Roy Chowdhury, et al. [24], Imran, et al. [25], Zhang [19], Hang and Zhao [13]
		Spreadsheet	Obrenovic and Gasevic [8], Wang, et al. [27], Kongdenfha, et al. [29], Hoang, et al. [30], Lemos, et al. [31]
		Wizard / Form	Yamaizumi, et al. [15], Chafle, et al. [32], Marin, and Lalanda [33]
		WYSIWYG Editor	Xiang and Madey [34], Liu, et al. [35]
	Dynamic Service Composition	High-level Graphical Language Editor	Han, et al. [36], Saifipoor, et al. [37], Pi, et al. [38]
		Visualizer	Rao, et al. [39]
		Wizard	Lee, et al. [40], Riabov, et al. [41]
		Natural Language Processor	Fujii and Suda [42]
Service Design	Design Process	Constraint-driven Environment	Yu, et al. [1], Lee, et al. [40], Aggarwal, et al. [43], Yamaziyumi, et al. [15], Obrenovic and Gasevic [8], Kongdenfha, et al. [29], Hoang, et al. [30], Zahoor, et al. [22], Maaradji, et al. [23], Wang, et al. [44], Pi, et al. [38], Lemos, et al. [31] , Hang and Zhao [13]
		Automatic Specification Reasoning Component	Xiang and Madey [34]
		Incremental Refining Component	Saifipoor, et al. [37], Riahov, et al. [41], Shiaa, et al. [45]
		Collaboration Enabling Component	Majithia, et al. [3], Marin and Lalanda [33], Zhang [17], Zhang [19] , Hang and Zhao [13]
	Design by Example	Programming-by-Example Recorder	Grechanik and Conroy [9], Wang, et al. [27], Wassink, et al. [14], Mehandjiev [20], Roy Chowdhury, et al. [24] , Hang and Zhao [13]
Service Reuse	Reusing Services	Service Discovery Component & Service Matchmaker	Radetzki, et al. [46], Wassink, et al. [14], Roy Chowdhury, et al. [24] , Hang and Zhao [13]
		Service Template	Aggarwal, et al. [43], Braem, et al. [7], Marin and Lalanda [33], Obrenovic and Gasevic [8], Namoun, et al. [18], Mehandjiev, et al. [20]
	Sharing & Distributing Services	Composite Service Packaging/Publishing Component	Majithia, et al. [3], Altintas, et al.[4], Chafle, et al.[32], Hang and Zhao [13]
		Composite Service Retriever	Altintas, et al. [4], Wang, et al.[27], Maaradji, et al. [23], Roy Chowdhury, et al. [24] , Hang and Zhao [13]
	Providing Right Service Composition Abstraction for End Users	Decent Service Composition Abstraction	Han, et al. [36], Fujii and Suda [42], Rao, et al. [39], Aggarwal, et al. [43], Trainotti, et al. [6], Haller, et al. [49], Vargas-Solar and Peñalva [50], Yamaizumi, et al. [15], Chafle, et al. [32], Marin and Lalanda [33], Riabov, et al. [41], Carlson, et al. [10], Liu, et al. [11], Shiaa, et al. [45], Obrenovic and Gasevic [8], Wang, et al. [27], Kongdenfha, et al. [29], Srinivasmurthy, et al. [16], Hoang, et al. [30], Zahoor, et al. [22], Zhang [17], Mei, et al. [52], Zhao, et al. [53], Wang, et al. [44], Liu, et al. [35], Imran, et al. [25], Pi, et al. [38], Lemos, et al. [31], Zhang [19], Hang and Zhao [13]
Service Testing	Immediate Feedback	Immediate Feedback	Lee, et al. [40], Majithia, et al. [3], Aggarwal, et al. [43], Chafle, et al. [32], Liu, et al. [11], Obrenovic and Gasevic [8], Srinivasmurthy, et al. [16], Liu, et al. [35], Roy Chowdhury, et al. [24]
	Maximizing Test Coverage	WYSIWYT Component	Yu, et al. [1]
	Checking Against Design Specifications	Specification Editor & Runtime Monitor	Yu, et al. [1], Trainotti, et al. [6], Haller, et al. [49], Vargas-Solar and Peñalva [50], Braem, et al. [7], Colombo, et al. [57], Chafle, et al. [32], Lin, et al. [12], Shiaa, et al. [45], Hang and Zhao [13]
	Visualization	Service Specification Visualizer	Liu, et al. [11]
Debugging	Change Suggestions	Change Suggestion Provider	Haller, et al. [49] , Hang and Zhao [13]

B. What EUSC activities are supported by these platforms?

As shown in Figure 2, 41 out of 47 platforms have been proposed to support service composition, and it makes service composition being the most supported EUSC activities. As the second most supported activity, service reuse is supported by 39 out of 47 platforms, whereas service design is assisted by 33 platforms. There are also 18 platforms aiming at helping users to test their services, whereas only 2 platforms concern the debugging activity of end users.

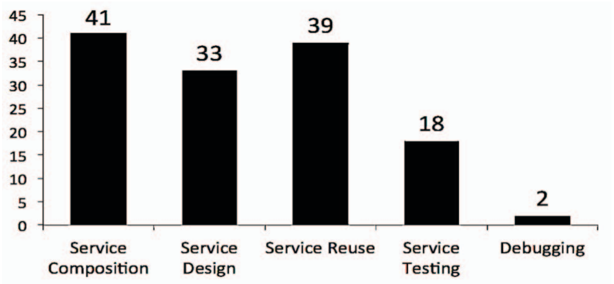

Figure 2. The five EUSC activities and the number of the proposed service composition platforms that support these activities.

C. *What approaches and tools are used to enable these EUSC activities?*

As shown in Table II, there are 12 approaches in total and 23 corresponding tools used to support the five EUSC activities. For service composition, workflow diagram editors are the most popular tool for end users, and have been adopted in 20 platforms. Constraint-driven environments (13 out of 47) are the most adopted tools to support the service design activity, whereas providing service composition abstraction (30 out of 47) is the main concern for supporting service reuse. For service testing, most platforms (10 out of 47) choose to use runtime monitors to check the execution status of composite services against their corresponding specifications. For debugging, the both reviewed platforms support this activity by providing end users with change suggestions.

IV. Open Research Issues

Looking forward, we have identified the following open research issues for the future:

- *Development of EUSC benchmarks*. First and foremost, we believe it is essential to develop benchmarks for objective comparisons and evaluation of end-user service composition platforms.
- *Provision of EUSC front-end.* For dynamic service composition platforms, instead of writing composite request manually, more end user-friendly tools need to be developed to help end users specifying composition requests.
- *Support for EUSC debugging.* According to our literature review, more research efforts need to be made to assist end users debugging their composite services.

To conclude, this paper has achieved its aim by presenting a systematic literature review of end-user service composition. It has reviewed 5 current activities performed by end users, 23 tools and 12 approaches that enable end users to compose and develop service systems from Web Services. The paper has also highlighted some key open research issues for the future.

References

[1] X. Yu, L. Zhang, Y. Li, and Y. Chen, "WSCE: a flexible Web service composition environment," in Web Services, 2004. Proceedings. IEEE International Conference on, 2004, pp. 428-435.

[2] H. Lieberman, F. Paternò, M. Klann, and V. Wulf, End-user development: An emerging paradigm: Springer, 2006.

[3] S. Majithia, M. Shields, I. Taylor, and I. Wang, "Triana: A graphical web service composition and execution toolkit," in Web Services, 2004. Proceedings. IEEE International Conference on, 2004, pp. 514-521.

[4] I. Altintas, E. Jaeger, K. Lin, B. Ludaescher, and A. Memon, "A web service composition and deployment framework for scientific workflows," in 2013 IEEE 20th International Conference on Web Services, 2004, pp. 814-814.

[5] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, et al., "The state of the art in end-user software engineering," ACM Computing Surveys (CSUR), vol. 43, p. 21, 2011.

[6] M. Trainotti, M. Pistore, G. Calabrese, G. Zacco, G. Lucchese, F. Barbon, et al., "Astro: Supporting composition and execution of web services," in Service-Oriented Computing-ICSOC 2005, ed: Springer, 2005, pp. 495-501.

[7] M. Braem, N. Joncheere, W. Vanderperren, R. Van Der Straeten, and V. Jonckers, "Guiding service composition in a visual service creation environment," in Web Services, 2006. ECOWS'06. 4th European Conference on, 2006, pp. 13-22.

[8] Z. Obrenovic and D. Gasevic, "End-User Service Computing: Spreadsheets as a Service Composition Tool," IEEE Transactions on Services Computing, vol. 1, pp. 229-242, 2008.

[9] M. Grechanik and K. M. Conroy, "Composing integrated systems using gui-based applications and web services," in Services Computing, 2007. SCC 2007. IEEE International Conference on, 2007, pp. 68-75.

[10] M. Carlson, A. H. Ngu, R. Podorozhny, and L. Zeng, "Automatic Mash Up of Composite Applications," in Service-Oriented Computing – ICSOC 2008. vol. 5364, A. Bouguettaya, I. Krueger, and T. Margaria, Eds., ed: Springer Berlin Heidelberg, 2008, pp. 317-330.

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

[12] C. Lin, S. Lu, Z. Lai, A. Chebotko, X. Fei, J. Hua, et al., "Service-oriented architecture for VIEW: a visual scientific workflow management system," in Services Computing, 2008. SCC'08. IEEE International Conference on, 2008, pp. 335-342.

[13] F. Hang and L. Zhao, "HyperMash: A Heterogeneous Service Composition Approach for Better Support of the End Users," presented at the Web Services (ICWS), 2013 IEEE 20th International Conference on, Santa Clara, CA, USA, 2013.

[14] I. Wassink, M. Ooms, and P. van der Vet, "Designing workflows on the fly using e-BioFlow," in Service-Oriented Computing, ed: Springer, 2009, pp. 470-484.

[15] T. Yamaizumi, T. Sakairi, M. Wakao, H. Shinomi, and S. Adams, "Easy soa: Rapid prototyping environment withweb services for end users," in Web Services, 2006. ICWS'06. International Conference on, 2006, pp. 931-932.

[16] V. Srinivasmurthy, S. Manvi, R. Gullapalli, D. Sathyamurthy, N. Reddy, H. Dattatreya, et al., "Web2exchange: A model-based service transformation and integration environment," in Services Computing, 2009. SCC'09. IEEE International Conference on, 2009, pp. 324-331.

[17] J. Zhang, "Co-Taverna: a tool supporting collaborative scientific workflows," in Services Computing (SCC), 2010 IEEE International Conference on, 2010, pp. 41-48.

[18] A. Namoun, T. Nestler, and A. De Angeli, "Service composition for non-programmers: Prospects, problems, and design recommendations," in Web Services (ECOWS), 2010 IEEE 8th European Conference on, 2010, pp. 123-130.

[19] J. Zhang, D. Kuc, and S. Lu, "Confucius: a scientific collaboration system using collaborative scientific workflows," in Web Services (ICWS), 2010 IEEE International Conference on, 2010, pp. 575-583.

[20] N. Mehandjiev, F. Lecue, U. Wajid, and A. Namoun, "Assisted Service Composition for End Users," presented at the IEEE 8th European Conference on Web Services (ECOWS), 2010.

[21] Keele Staffs, "Guidelines for performing systematic literature reviews in software engineering," Technical report, EBSE Technical Report EBSE-2007-012007.

[22] E. Zahoor, O. Perrin, and C. Godart, "Disc: A declarative framework for self-healing web services composition," in Web Services (ICWS), 2010 IEEE International Conference on, 2010, pp. 25-33.

[23] A. Maaradji, H. Hacid, R. Skraba, A. Lateef, J. Daigremont, and N. Crespi, "Social-Based Web Services Discovery and Composition for Step-by-Step Mashup Completion," in Web Services (ICWS), 2011 IEEE International Conference on, 2011, pp. 700-701.

[24] S. Roy Chowdhury, C. Rodríguez, F. Daniel, and F. Casati, "Baya: assisted mashup development as a service," in Proceedings of the 21st

international conference companion on World Wide Web, 2012, pp. 409-412.

[25] M. Imran, F. Kling, S. Soi, F. Daniel, F. Casati, and M. Marchese, "ResEval mash: a mashup tool for advanced research evaluation," in Proceedings of the 21st international conference companion on World Wide Web, 2012, pp. 361-364.

[26] C. Cappiello, M. Matera, M. Picozzi, A. Caio, and M. T. Guevara, "MobiMash: end user development for mobile mashups," in Proceedings of the 21st international conference companion on World Wide Web, 2012, pp. 473-474.

[27] G. Wang, S. Yang, and Y. Han, "Mashroom: End-User Mashup Programming using Nested Tables," presented at the Proceedings of the 18th international conference on World wide web, Madrid, Spain, 2009.

[28] M. P. Papazoglou and W.-J. van den Heuvel, "Service-Oriented Computing: State-of-the-Art and Open Research Issues," IEEE Computer. v40 i11, 2003.

[29] W. Kongdenfha, B. Benatallah, J. Vayssière, R. Saint-Paul, and F. Casati, "Rapid development of spreadsheet-based web mashups," in Proceedings of the 18th international conference on World wide web, 2009, pp. 851-860.

[30] D. D. Hoang, H.-Y. Paik, and A. H. Ngu, "Spreadsheet as a generic purpose mashup development environment," in Service-Oriented Computing, ed: Springer, 2010, pp. 273-287.

[31] A. L. Lemos, M. C. Barukh, and B. Benatallah, "DataSheets: A Spreadsheet-Based Data-Flow Language," in Service-Oriented Computing, ed: Springer, 2013, pp. 616-623.

[32] G. Chafle, G. Das, K. Dasgupta, A. Kumar, S. Mittal, S. Mukherjea, et al., "An integrated development environment for web service composition," in Web Services, 2007. ICWS 2007. IEEE International Conference on, 2007, pp. 839-847.

[33] C. Marin and P. Lalanda, "DoCoSOC- Domain Configurable Service-Oriented Computing," in Services Computing, 2007. SCC 2007. IEEE International Conference on, 2007, pp. 52-59.

[34] X. Xiang and G. Madey, "A semantic web services enabled web portal architecture," in Web Services, 2004. Proceedings. IEEE International Conference on, 2004, pp. 834-835.

[35] X. Liu, Q. Zhao, G. Huang, H. Mei, and T. Teng, "Composing data-driven service mashups with tag-based semantic annotations," in Web Services (ICWS), 2011 IEEE International Conference on, 2011, pp. 243-250.

[36] Y. Han, H. Geng, H. Li, J. Xiong, G. Li, B. Holtkamp, et al., "VINCA–A visual and personalized business-level composition language for chaining web-based services," in Service-Oriented Computing-ICSOC 2003, ed: Springer, 2003, pp. 165-177.

[37] S. Saifipoor, B. T. Ladani, and N. Nematbakhsh, "A Dynamic Reconfigurable Web Service Composition Framework Using Reo Coordination Language," in Web Services, 2007. ECOWS'07. Fifth European Conference on, 2007, pp. 203-212.

[38] B. Pi, G. Zou, C. Zhong, J. Zhang, H. Yu, and A. Matsuo, "Flow Editor: Semantic Web Service Composition Tool," in Services Computing (SCC), 2012 IEEE Ninth International Conference on, 2012, pp. 666-667.

[39] J. Rao, P. Kungas, and M. Matskin, "Logic-based Web services composition: from service description to process model," in Web Services, 2004. Proceedings. IEEE International Conference on, 2004, pp. 446-453.

[40] Y. Lee, C. Patel, S. A. Chun, and J. Geller, "Towards intelligent Web services for automating medical service composition," in Web Services, 2004. Proceedings. IEEE International Conference on, 2004, pp. 384-391.

[41] A. V. Riabov, E. Boillet, M. D. Feblowitz, Z. Liu, and A. Ranganathan, "Wishful search: interactive composition of data mashups," in Proceedings of the 17th international conference on World Wide Web, 2008, pp. 775-784.

[42] K. Fujii and T. Suda, "Dynamic service composition using semantic information," in Proceedings of the 2nd international conference on Service oriented computing, 2004, pp. 39-48.

[43] R. Aggarwal, K. Verma, J. Miller, and W. Milnor, "Constraint driven web service composition in METEOR-S," in Services Computing, 2004.(SCC 2004). Proceedings. 2004 IEEE International Conference on, 2004, pp. 23-30.

[44] P. Wang, Z. Ding, C. Jiang, and M. Zhou, "Web service composition techniques in a health care service platform," in Web Services (ICWS), 2011 IEEE International Conference on, 2011, pp. 355-362.

[45] M. M. Shiaa, J. O. Fladmark, and B. Thiell, "An incremental graph-based approach to automatic service composition," in Services Computing, 2008. SCC'08. IEEE International Conference on, 2008, pp. 397-404.

[46] U. Radetzki and A. B. Cremers, "Iris: A framework for mediator-based composition of service-oriented software," in Web Services, 2004. Proceedings. IEEE International Conference on, 2004, pp. 752-756.

[47] A. J. Ko and B. A. Myers, "Designing the whyline: a debugging interface for asking questions about program behavior," in Proceedings of the SIGCHI conference on Human factors in computing systems, 2004, pp. 151-158.

[48] A. Namoun, T. Nestler, and A. De Angeli, "Service Composition for Non-programmers: Prospects, Problems, and Design Recommendations," presented at the IEEE European Conference on Web Services (ECOWS), 2010.

[49] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler, "Wsmx-a semantic service-oriented architecture," in Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on, 2005, pp. 321-328.

[50] G. Vargas-Solar and P. L. Peñalva, "Building WEB services portals: implementation experiences," in Services Computing, 2005 IEEE International Conference on, 2005, pp. 217-223.

[51] W. E. Mackay, "Patterns of sharing customizable software," in Proceedings of the 1990 ACM conference on Computer-supported cooperative work, 1990, pp. 209-221.

[52] L. Mei, Y. Wang, Q. Li, J. Wang, and Z. Zhu, "A Service-Oriented Framework for Hybrid Immersive Web Applications," in Web Services (ICWS), 2011 IEEE International Conference on, 2011, pp. 556-563.

[53] Z. Zhao, S. Bhattarai, and N. Crespi, "An event-based functionality integration framework," in Web Services (ICWS), 2011 IEEE International Conference on, 2011, pp. 720-721.

[54] R. R. Panko, "What we know about spreadsheet errors," Journal of Organizational and End User Computing (JOEUC), vol. 10, pp. 15-21, 1998.

[55] R. R. Panko, "Spreadsheet errors: What we know. what we think we can do," arXiv preprint arXiv:0802.3457, 2008.

[56] D. G. Hendry and T. R. Green, "Creating, comprehending and explaining spreadsheets: a cognitive interpretation of what discretionary users think of the spreadsheet model," International Journal of Human-Computer Studies, vol. 40, pp. 1033-1065, 1994.

[57] M. Colombo, E. Di Nitto, and M. Mauri, "Scene: A service composition execution environment supporting dynamic changes disciplined through rules," in Service-Oriented Computing–ICSOC 2006, ed: Springer, 2006, pp. 191-202.