Chapter 28
End Users Developing Mashups

Nikolay Mehandjiev, Abdallah Namoun, Freddy Lécué, Usman Wajid and Georgia Kleanthous

Abstract Mashups can open up access to the wealth of on-line information, allowing information-providing services to be discovered, integrated and presented in a manner tuned to current user needs. Their uptake is hindered by the fact that most information consumers do not have programming background and thus find it difficult to work with the current systems which are technology-driven. Many researchers attempt to help such non-programmers by replacing programming scripts with interactive visual representations to connect different information-providing service components into an assembly. However, the underlying programming techniques such as event-driven processing still shape the visual interface and make it difficult to understand for non-programmers. In contrast, we did not start with the technology but with the users—service producers and consumers, and studied the core issues which should be resolved before non-programmers can assemble meaningful mashups, over and above the presentation-level integration offered by current mashup environments. The result is an approach to assisted service composition designed for end users, which uses semantic technologies to shield users from the irrelevant complexity of service technology, from the heterogeneity of the information and from the need to

N. Mehandjiev (✉) · A. Namoun · U. Wajid · G. Kleanthous
Manchester Centre for Service Research,
University of Manchester,
Manchester M60 1QD, UK
e-mail: n.mehandjiev@manchester.ac.uk

A. Namoun
e-mail: abdallah.namoune@mbs.ac.uk

U. Wajid
e-mail: usman.wajid@manchester.ac.uk

G. Kleanthous
e-mail: georgia.kleanthous@gmail.com

F. Lécué
IBM Research, Dublin, Ireland
e-mail: freddy.lecue@ie.ibm.com

A. Bouguettaya et al. (eds.), *Web Services Foundations*,
DOI: 10.1007/978-1-4614-7518-7_28,

manually resolve dependencies between services. A tool has been developed to help us validate the approach through two observational studies of non-programmers. The studies confirmed the enabling effect of the approach, and generated suggestions for further work at the levels of both the approach and the tool.

28.1 Introduction

The empowering influence of the World Wide Web in terms of fast and convenient access to information and services from all areas of human knowledge and culture, and from any corner of the globe, is universally accepted. This is taken a step further with the idea of mashups, allowing users to combine information from different sources, process it and present it in a finely-tuned manner by composing information-providing services. These have developed from simple web pages aggregating information from different sources and presenting it side-by-side without any integration, such as iGoogle and myYahoo!, to sophisticated mashups where information is passed through a number of processing steps in a workflow-type fashion, for example Gravity[1] and MarcoFlow [11].

Given the clear potential for benefiting from such activities in terms of unleashing creativity and providing services at the point of need, the limited uptake of mashup environments is somehow puzzling. A closer look at the currently available commercial environments and research systems reveals one potential reason—the technology-driven approach underpinning them. The design of such systems would typically start from an integration technique such as event-driven processing, and a visual front-end would be constructed to present this technique to the users through a (hopefully) easy-to-understand metaphor.

However, people trained in programming and able to manage the complexity of contemporary software technologies are a small fraction of all the users who can benefit from mashup technology. Mashup environments constructed around an integration technique such as event-driven processing would not be easily accessible for the latter type of general users, despite the visual front-ends. Indeed, from the existing mashup systems (a selection of which is reviewed in Sect. 28.2), the successful ones (such as iGogle and myYahoo!) are the simplest, presenting information side-by-side in separate panels without any integration between different information sources.

In contrast to the majority of existing mashup approaches, we started from a user-driven perspective, and studied the mental models of general users with regards to mashup activities, and the issues which prevent them from assembling services into meaningful compositions processing information in a non-trivial manner. The results, reported in Sect. 28.3 of this paper, suggest we need to reduce learning costs by making the composition as transparent as possible, hide any technical details which are not relevant to the task of the user, and provide immediate feedback in

[1] Available from http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/17826.

respect to any design decisions by end users. We also gained insight into our users' mental models regarding software services and their composition.

These activities helped us to create a novel task-oriented approach to composing services into mashups, where users are shielded from the underlying technology and from the heterogeneity of the information processed, and only asked to select from a number of alternative information-providing or processing services for different parts of the composition. Semantic reasoning takes over mundane technical details such as aligning service inputs and outputs and resolving inter-service dependencies, and composition templates allow "best-practice" sharing of mashups specific to the application domain and to the tasks which are to be supported by the mashup.

The approach, called *assisted service composition for end users* and described in detail in Sect. 28.4, comprises the following two contributions:

1. A template-based process with three stages: assisted composition, template adaptation and learning (generalising user-adapted innovative applications into templates). Here we focus on the stage of assisted composition since it targets the widest audience of general users.
2. A semantic technique of service alignment, alleviating the need for data integration between constituent services, and shielding users from inter-service dependencies and from the technical complexity of service technology.

To validate the approach, we have developed a prototype tool which supports the approach and allows users to compose mashups by using "point and click" to select services. The tool uses semantic (monotonic) reasoning to resolve the dependencies between services, and advises the user regarding compatability between services. We used this tool in two observational studies with non-programmers. The observational studies, reported in Sect. 28.5, confirmed the enabling effect of the approach, and generated suggestions for further work at the levels of both the approach and the tool.

28.2 Related Research

28.2.1 Mashups and Service Composition Environments

A number of environments exist which allow the composition of information-provision services in some form of a mashup. The simplest are from presentation-level mashups which display information from different services side-by-side in widgets. The widgets support uniform user interface framework and allow the creation of personalised portal pages. These environments are mainlstream and enjoy a large number of users, for example iGoogle and myYahoo.

The next level of sophistication supports the exchange of data between different components of the mashup such as RSS data sources, information processing services, search services and display components, allowing the creation of

functionality over and above the capabilities of the individual reusable components. Some researchers start their mashup classifications from this category of data mashups, omitting the previous presentation-level maships, for example Daniel et al. consider them in the "Simple Mashups" category [10], where we have a single page, a single user and no support for processing workflow. Often cited examples here are the commercial environments Yahoo!Pipes[2] and MashMaker [13]). Yahoo!Pipes uses pipe-and-filter-style composition where information sources and processing components are linked with dataflow pipes, and focuses on facilitating reuse of composition templates by allowing users to publish the pipes they have created and to adapt the pipes of others. Intel's MashMaker takes a more direct approach, analysing the web sites visited by users to extract data which are being displayed, and suggesting how this data can be combined to achieve new output. There are also some research systems of varying degrees of maturity in this space, for example MashArt [9] uses event-based mechanisms and dataflow between different types of components to allow the construction of complex applications. Other systems using dataflow-based integration include [26, 47, 49] and others, including systems advising on appropriate next steps for the composition [18], and supporting navigation through complex mashups [12].

In contrast to the first two groups, the third group of process mashups focus on the flow of control between the different components of the mashup, using visual representation of BPEL or BPMN-style constructs linking these components. Examples of such systems are [11, 27]. At the process composition level, `bpmn.org` reports 62 tools for wiring services together using the Business Process Modelling Notation (BPMN). The approach reported here belongs to this group, yet the process representations used are hidden from the naive users. In common with other approaches in this group, we focus on the way mashup components interact in the course of a business process, and presume that the presentation integration will be handled using another existing approach.

Inspired by the underlying implementation technology and often validated by case study implementations, only a few of these systems have been evaluated in terms of usability and cognitive effectiveness. We focus on these criteria in the next section.

28.2.2 User-Centric Approaches to Service Composition

The academic field of End User Development (EUD) [25, 44] takes a user-centric approach to creating tools which can enable non-technical users to develop sophisticated applications. Main EUD results include theoretical models such as the tradeoff-based "Attention Investment Model" [4] and the lifecycle model of Meta-Design [16]. There are also a number of well-known practical successes such as spreadsheets and database form painters.

[2] http://pipes.yahoo.com/pipes/

Service-focused work in this field, however, is focused on professional programmers [2], or on web mashups rather than fully fledged service composition [51]. An exemplary user-driven design process is reported in [41], yet it is focused on conventional web applications rather than web services.

Interesting conclusions in this field include the need for supporting end users by hiding irrelevant technical details and complexity from them, providing them instead with task-oriented languages [37]; and the view of end user environments as a medium of continuous collaboration between end users and developers, resulting in the evolution of the environment itself to reflect evolving user skills and requirements [33]. The concept of "Power Users" (technology-savvy end users) as a third side to this collaboration is also important since they are often leaders of user-driven application innovation. Our studies, reported in Sect. 28.3, confirmed the validity of some of these conclusions for the domain of user-driven service composition.

28.2.3 Automating Service Composition

Taken to its extreme, the idea of supporting end users in service composition would translate into the aim of fully automating the composition. Indeed, many AI-inspired approaches [3, 20, 28, 40] address the issue of automated web service composition. Full automation, however, even if it were feasible, would miss the chance for user-led innovation and fine-tuning services to user needs.

Only a handful of approaches have the users in the driving seat and support them by resolving technical details such as data integration and other service dependencies. For example, Carlson et al. [7] introduce an approach where users can drag a service onto a canvas, and this narrows down all discovered services to only those which have compatible inputs and outputs with the service thus selected by the user. A more structured support of the composition process is provided by the composition tool reported in [42], where the process is step-by-step, guided by the tool. Both approaches use semantic tagging of services, and limited semantic reasoning with the data thus available. However, neither of them uses templates and thus cannot support reuse of composition knowledge.

Semantic reasoning underpins such selection of compatible services. This can use basic semantic matching types [24, 39], the difference operator [5, 45] or Concept Abduction [8]. Different approaches differ in performance and scalability, and we need to consider the correct approach based on the expected scale of compositions and number of candidate services. This paper does not address research challenges related to ontology matching [14], which is out of the scope of this paper.

28.3 Challenges to Users Attempting to Compose Services

Our user-centric approach to enabling mashup composition by general users necessitated gaining insight into the mental models of services and service composition held by our target user groups, and understanding the main issues which may impede their uptake of service composition. This was achieved through a number of focus groups involving 64 users of mixed background—technical and non-technical. Whilst details of the full study are published elsewhere [30, 36], here we focus on the main challenges faced by users from non-programming background when attempting service composition into mashups.

28.3.1 Realistic Complexity is Overwhelming

Participants quickly extrapolated the simplistic examples used to ones of realistic complexity, which may involve up to "2000 services for each task", and a sizeable number of tasks involved. Our target end users did not consider themselves able (or indeed interested) to handle such complexity, and to manage the complex dependencies existing between different tasks. Some users also did not consider themselves at ease with having to "think in sequence".

The use of "best practice" applications and composition templates was suggested to address some of the complexity issues and allow the sharing of process knowledge between users. A further challenge associated with this approach would be to manage the evolution of both the task and the available services, indeed a successful mashup tool should be able to accommodate frequent new tasks and services.

28.3.2 Heterogeneous Data and Dependencies Between Serviecs

When participants were presented with diagrams showing the flow of data between services in a mashup, a number of them pointed out that the high number of connections linking services makes the interactions "difficult to understand" and hard to figure out "what is going on". The "spaghetti"-like nature of such diagrams made it diffucult to work out where "to put a new service". Also some participants commented on data dependencies as being not "natural".

The alternative of control-flow-based diagram connecting services was felt to "lack the level of detail that is required to make it work". Abstracting away data from these diagrams was felt by the technical users to introduce potential for errors in terms of data mismatch between services. Indeed, a number of users pointed out the different standards and formats of data (XML versus text for example), and the potential for error this would create. These problems were not foreseen by

non-technical users who ranked control flow very high and disliked the complexity which stems from explicitly representing data flow between services.

Related discussion points covered the need to specify the semantics of the services using standard semantic notations. This, however, was expected to bring complexity to service descriptions, so we also need to have different ways of representing the composition to people with different skill levels. We need the tool to be "flexible enough to allow composition without worrying about low-level details", whilst we need some "expert mode" for people with technical skills. The tool should support the users by validating the services chosen, ensuring there are no mistakes.

28.4 The Assisted Composition Approach

The approach we have developed can support non-technical end users in creating actual service compositions starting from abstract descriptions of the composition in a template. Those "power users" who are happier to engage with software are then enabled to further customise such compositions by changing the abstract templates and creating innovative variations of standard service composition tasks, customised to specific application domain or social context. In the third stage, a number of similar innovative variations would be generalised into a new reusable template by software developers, thus ensuring the growth of the overall system, allowing innovative compositions to be reused by non-technical users. This three-stage lifecycle of user-driven composition is shown in Fig. 28.1. Here we focus on the left side of the cycle, which involves end users binding concrete services to reusable templates.

Our work with end users (see Sect. 28.3) asserts that users should be shielded from technical details of service assembly such as data dependencies between services. We therefore hide from end users both control flow dependencies and data dependencies between tasks within the template processes. These aspects of the composition are instead considered behind the scene using semantic reasoning.

This section describes in further detail these aspects of our approach, using a formal model of semantic connections between services. The process of assisted composition is then presented. But first we describe a short motivating example.

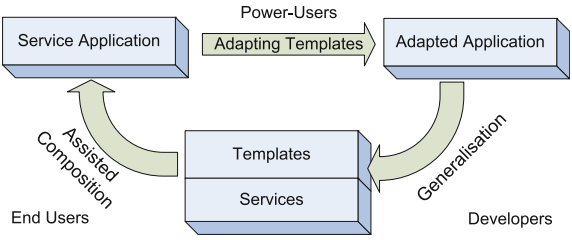

Fig. 28.1 Lifecycle of user-developed service applications

28.4.1 Motivating Scenario

The following scenario is one of the several we used within SOA4All. It targets the arrival of an overseas student to the UK. Students search for suitable universities and register for a course upon arrival. They use the acceptance letter to open a bank account and submit tax exemption letters. The bank account is then used to set up payment for University fees.

There are dependencies between the different registration tasks, often unknown to the arrivals, which cause delays and repeated visits, leading to frustration. A composite reusable service can alleviate this by guiding the process and passing the relevant data through.

Here is a list of tasks required to achieve student registration:

- `SearchForAUniversity` that returns some `Universities` and their descriptions (e.g., `UniversityID`, `Name`, `Postcode`, `Course` and their `Fees`) given a `PostCode` as a location and some subjects of courses `Subject`;
- `RegisterForACourseInUK` that returns an `AcceptanceLetter` and a `StudentID` given a `Person` and an `UKUniversity`;
- `OpenBankAccount` that returns a `BankAccount` given a `Person` and an `AcceptanceLetter`;

All input and output parameters above refer to concepts from a domain ontology, an example portion of which is shown in Fig. 28.2. The particular ontology, which is based on the Description Logics DLs $\mathcal{ALE}$—Attributive language with Atomic nega-

$$
\begin{aligned}
&University \equiv Institution \sqcap \forall hasID.UniversityID\\
&\qquad \sqcap \exists hasID.UniversityID \sqcap \forall hasName.Name\\
&\qquad \sqcap \exists hasName.Name \sqcap \forall hasPostcode.Postcode\\
&\qquad \sqcap \exists hasPostcode.Postcode \sqcap \exists hasCourse.Course\\
&UKUniversity \equiv University \sqcap \forall hasPostcode.UKPostcode\\
&\qquad \sqcap \exists hasPostcode.UKPostcode\\
&Course \equiv \exists hasFees.Fees \sqcap \exists hasID.CourseID \sqcap \exists Subject\\
&Person \equiv \exists hasPostcode.Postcode \sqcap \exists hasFirstName.Name\\
&\qquad \sqcap \exists hasLastName.Name\\
&Student \equiv Person \sqcap \exists hasID.StudentID\\
&BankAccount \equiv Account \sqcap \forall hasSortCode.SortCode\\
&\qquad \sqcap \exists hasSortCode.SortCode \sqcap \forall hasIBAN.IBAN\\
&\qquad \sqcap \exists hasIBAN.IBAN \sqcap \forall hasAgencyRef.AgencyRef\\
&\qquad \sqcap \exists hasAgencyRef.AgencyRef\\
&\qquad \sqcap \forall hasAccoubtNb.AccountNb\\
&\qquad \sqcap \exists hasAccoubtNb.AccountNb\\
&AcceptanceLetter \sqsubset Letter,\ UKPostcode \sqsubset PostCode\\
&Name \sqsubset \top, Fees \sqsubset \top, CourseID \sqsubset \top, ID \sqsubset \top, Subject \sqsubset \top,\\
&StudentID \sqsubset \top, Account \sqsubset \top, SortCode \sqsubset \top, IBAN \sqsubset \top, AgencyRef \sqsubset\\
&\top, AccountNb \sqsubset \top
\end{aligned}
$$

Fig. 28.2 Part of an $\mathcal{ALE}$ TBox

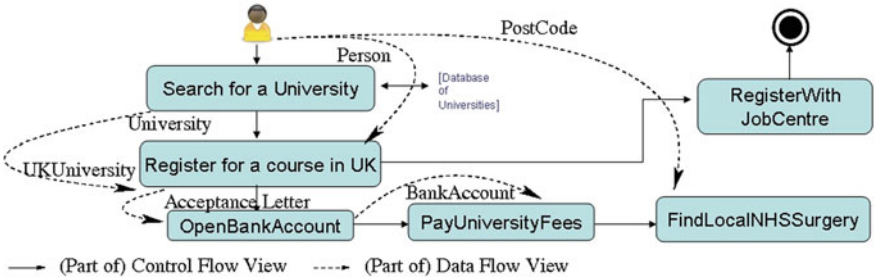

Fig. 28.3 Template-based composition view of the motivating scenario

tion, Concept intersection, Universal restrictions and Existential quantification [1], is part from a larger pair $\langle \mathcal{T}, \mathcal{A} \rangle$. $\mathcal{T}$ and $\mathcal{A}$ refer respectively to a Terminological Box (or TBox i.e., intentional knowledge) and an Assertional Box (or ABox i.e., extensional knowledge) in DL systems. In the following, we will focus on the TBox $\mathcal{T}$ that supports inference on service parameters by means of DL reasoning. An example of inference models concept is subsumption (i.e., concept-based hierarchy checking) for evaluating specialization and generalization of services parameters.

This scenario illustrates the need for user involvement at the level of service customisation. Even if the mashup application was developed by software professionals, each student will have different requirements in terms of need to register with the police, desire to obtain work permit, children to enroll in local schools, etc. This variability motivates direct user control of the composition in terms of user tailoring of the tasks included and which services should be used for each task.

28.4.2 Template-Based Service Composition

An intuitive view to service composition would see it as aiming to satisfy the need for a (non-existing) service by bringing together existing ones. This integration can be done manually, yet this would involve the alignment of numerious inputs and outputs, considering a number of pre- and post-conditions, and dealing with other technical issues of grounding, etc, which are clearly outside of the skills and interests of our end users. Since we allow end users to tune a composition to their own needs, we do not need to have complete automation "from scratch" using program synthesis and AI planning techniques [28].

Instead we opt to reuse composition knowledge and provide a starting template for the users. This composition knowledge can be based on past successful compositions, and can be seeded by formalising domain-specific knowledge about how the problem addressed by the sought service would decompose into sub-problems [46], and task-specific knowledge about the core types of information processing activities [43].

Example 1 (Template-Based Service Composition)

Figure 28.3 presents the template-based composition view of the motivating scenario in Sect. 28.4.1. Tasks (or "service slots") are designed along rectangles while simple arrows are used to model a partial order on these services (i.e., its control flow). Dashed arrows refer to data flow description as possible interdependencies between tasks.

Here we focus on the stage of *template instantiation* [34, 48], where we need to allocate a specific service for each of the generic "service slots" in the template, using knowledge about the data connections and pre- and post-conditions of services. This is the 'Assisted Composition' arrow in Fig. 28.1. Note that the overall approach also includes the stages of template adaptation, where power users can create innovative solutions; and generalisation, or converting the useful innovative adaptations into new templates, helping the library of templates grow. The last stage addresses the issue of feasibility of providing a realistic library of templates—once the library is seeded for a particular domain using generic task scripts, user community will "fill the gaps" by creating numerious innovative applications, mirroring the processes currently underway on Facebook and Yahoo!Pipes.

The focus of supporting end users means that, contrary to existing work in the area (e.g. [22, 29]), we leave control in the hands of our users, and aim to provide expert guidance regarding three parts of the process: selecting suitable services for each task, ranking them according to user profile and working out compatibilities between services in terms of data flow, pre- and post-conditions. The users are involved in selecting one or more services from the shortlist for each task, according to their preferences and knowledge. Because we ensure that users have chosen compatible services, and because these services are tagged semantically, we can automate the mapping of data from one service to the other at execution time behind the scenes, without having to involve users in this.

In this paper we focus only on the aspects of selecting a set of appropriate service candidates for each task, and on working out compatibilities between services in terms of data flow in order to show to the end user the consequences of them selecting a given instance. We use semantic reasoning for both aspects, for example once a user selects a service s_i^j for task T_i, we use semantic reasoning to tag as eligible for further selection only those service candidates for the other tasks in the template which are compatible with s_j^i. Before providing further details in Sect. 28.4.4, we need to describe the semantic reasoning taking place behind the scene.

28.4.3 Semantic Connections of Services

Using tasks specifications of inputs, outputs, pre- and post-conditions of templates, we should be able to infer additional dependencies between tasks, for example we can infer data flow dependencies between tasks using their input and output specifications. In the following we present such dependencies as *semantic links* [22] between services. Then we describe our *semantic-link-based composition model*.

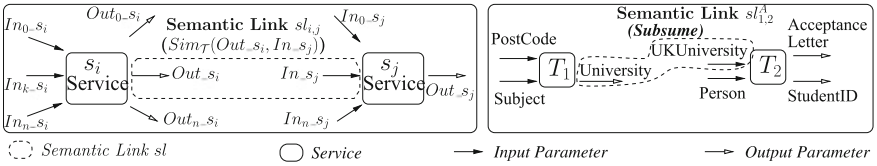

Fig. 28.4 A semantic link $sl_{i,j}$ and its illustration on the motivating scenario

28.4.3.1 Semantic Links

Since input and output parameters of semantic web services are specified using concepts from a common ontology[3] or Terminology $\mathcal{T}$ (an example of such is given in Fig. 28.2), retrieving links between output parameters $Out\_s_i \in \mathcal{T}$ of services s_i and input parameters $In\_s_j \in \mathcal{T}$ of other services s_j could be achieved by using a DL reasoner such as Fact++[4] [19]. Such a link, also known as semantic link [22] $sl_{i,j}$ (Fig. 28.4) between two functional parameters of s_i and s_j is formalized as

$$\langle s_i, Sim_{\mathcal{T}}(Out\_s_i, In\_s_j), s_j \rangle \tag{28.1}$$

Thereby s_i and s_j are partially linked according to a matching function $Sim_{\mathcal{T}}$. This function expresses which matching type is employed to chain services. The range of $Sim_{\mathcal{T}}$ is reduced to the four well known matching type introduced by [39] and the extra type Intersection [24]:

- **Exact** If the output parameter Out_s_i of s_i and the input parameter In_s_j of s_j are equivalent; formally, $\mathcal{T} \models Out\_s_i \equiv In\_s_j$.
- **PlugIn** If Out_s_i is sub-concept of In_s_j; formally, $\mathcal{T} \models Out\_s_i \sqsubseteq In\_s_j$.
- **Subsume** If Out_s_i is super-concept of In_s_j; formally, $\mathcal{T} \models In\_s_j \sqsubseteq Out\_s_i$.
- **Intersection** If the intersection of Out_s_i and In_s_j is satisfiable; formally, $\mathcal{T} \not\models Out\_s_i \sqcap In\_s_j \sqsubseteq \bot$.
- **Disjoint** If Out_s_i and In_s_j are incompatible i.e., $\mathcal{T} \models Out\_s_i \sqcap In\_s_j \sqsubseteq \bot$.

Following the definition of semantic links $sl_{i,j}$ between web service instances s_i and s_j, we also define abstract semantic links $sl^A_{i,j}$ between tasks T_i and T_j.

Example 2 (Semantic Link and Subsume Matching Type) Suppose T_1 and T_2 are respectively tasks related to `SearchForAUniversity` and `RegisterForACourseInUK` in Fig. 28.4 (right part) of the motivating scenarion in Sect. 28.4.1. In such a case, the output parameter `University` of T_1 is semantically linked to

[3] Distributed ontologies are not considered here but are largely independent of the problem addressed in this work.

[4] http://owl.man.ac.uk/factplusplus/

the input parameter `UniversityUK` of T_2. According to the example ontology in Fig. 28.2, this abstract semantic link $sl^A_{1,2}$ is valued by a Subsume matching type since $University \sqsupset UKUniversity$.

28.4.3.2 Semantic Link Composition Model

When composing services in mashups, we need to be able to reason about the quality of composition, using aggregation from the quality of individual semantic links. We conceptualise the process model of web service composition as a directed graph which has the web service specifications s_i as its nodes, and the semantic links $sl_{i,j}$ (data dependencies) as its edges.

We can generalise this model to template-based compositions, pre-computed for instance by *template-based* and *parametric-design-based* approaches [34, 48]. The composition graph there has the tasks specifications T_i as its nodes, and abstract semantic links $sl^A_{i,j}$ as its edges.

28.4.4 Helping Users Choose Services Through Semantic Reasoning

Given a template-based composition, semantic descriptions of the tasks in the template and of the candidate services, our approach can help users to instantiate templates with candidate services to optimise the quality of the composition. This is done using the semantic link composition models, where the data flow in the composition is automatically inferred from the DL descriptions of services parameters and from the template of how the composition breaks down into tasks. Optimising the composition models is done at the background and remains hidden from the end users, following [23]. However existing state-of-the-art approaches can be employed for performing optimization using different techniques [6, 52] on different parameters e.g., QoS only [50]. The quality estimate generated is used to provide feedback to users about their selection decisions as follows:

Once a user selects a service, the tool will grey out all service candidates which are incompatible (have a low quality of the semantic links with the selected service instance), and highlight the compatible ones (the instances for which the quality model computes high values). As illustrated in Fig. 28.5, our abstract visualisation hides all details related to control and data flow in the composition, and deals with them in the background.

Example 3 (Abstract Visualisation of Composition) Figure 28.5 illustrates a template-based composition where the user has selected a goal from the taxonomy on the left panel. All related details to data and control flow are abstracted away, and end-users could simply interpret compositions as a list of tasks (first row in Fig. 28.5) wherein each tasks could be instantiated by services (columns in Fig. 28.5). This is a

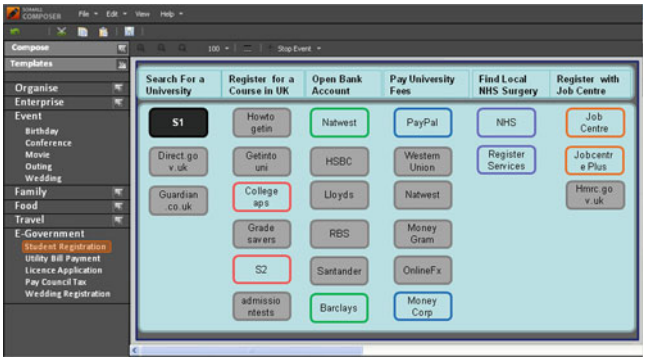

Fig. 28.5 Abstract visualisation of composition

snapshot from a low-tech mock-up we developed early on to test the idea with users, a snapshot of the actual tool is provided in Fig. 28.8.

The overall approach is descibed from two different perspectives: end users interacting with our tool, and actual back-end reasoning.

28.4.4.1 From the End-User Perspective

First of all, the user is responsible for selecting a template from a set of available ones, all organised in a domain taxonomy. The abstract visualisation of the template is then automatically generated by simply extracting its tasks and discovering relevant candidate services for each of them.

The user will proceed to select any service in any column. This selection step assigns the selected service(s) for the considered task. The system reduces the list of candidate services for each task to those which are compatible with the selection and gets back to the users with (only) services that could be assigned to other tasks. This reduction is based on both the previous selection and how the selected services can be semantically linked to candidate services of other tasks. This is repeated until each task is assigned to a service.

Example 4 (Assisted Composition from the End-User Perspective) Figure 28.5 illustrates the instantiation procedure of template after the selection of service s_1 for task `SearchForAUniversity`. Services `CollegeApps` and s_2 of task `RegisterForACourseInUK` are highlighted (in blue) because of (semantic) compatible data flow (Sect. 28.4.3) between them and s_1, while `GradeSavers` (in grey) is not because of its incompability with s_1.

During each step of the template instantiation, the end-user can backtrack and even manually remove some services from any candidates' list.

28.4.4.2 From the Back-End Perspective

Once the template is selected by the end-user, our system aims to discover candidate services that could be assigned to tasks of the template. Note that all services and templates are annotated with goals, pre- and post-conditions, and input- and output-parameter types. The addition of goals as a separate tagging element allows us to estimate their semantic proximity and differentiate between tasks and services having same inputs and outputs but achieving different things.

A task T of a template can be instantiated by a service s if and only if:

1. The service s achieves the same goal as T, assuming an ontology of goals [15].
2. The pre-conditions of s are implied by the pre-conditions of T.
3. The post-conditions of s imply the post-conditions of T.
4. The matching type between the input specification In_T of T and the input specification In_s of s i.e., $Sim_{\mathcal{T}}(In\_T, In\_s)$ is PlugIn.
5. The matching type between the output specification Out_s of s and the output specification Out_T of T i.e., $Sim_{\mathcal{T}}(Out\_s, Out\_T)$ is PlugIn.

Conditions (1)–(3) above ensure the candidate service s has the desired effect of the target task T, whilst conditions (4) and (5) ensure the semantic (functional) fit between the candidate service and the target task. Condition (4) ensures that all the data which can be passed onto T can be procesed by s. Condition (5) ensures that the output of s fits within the output specifications of T.

Once a service is selected by the user, our system retrieves its semantic descriptions and computes all its potential incoming and outgoing semantic links with services of other tasks. The computation is based on the abstract semantic links in the template and the actual service descriptions. Services can be then linked with many services depending on the data flow description of the composition. As previously mentioned, only services linked with a semantic link of value *Exact* or *PlugIn* are consided for robustness reasons. Therefore, these services are highlighted, others are greyed out in the abstract visualisation of the composition.

Example 5 (Assisted Composition from the Back-end Perspective) According to Example 4 and Fig. 28.5, s_1 has been selected to achieved task `Search-For AUniversity`. Our system dynamically reduces the candidates' list of other tasks such as `RegisterForACourseInUK` or `OpenBankAccount` depending on quality of semantic links between services. For instance, the service `GradeSavers` (for `RegisterForACourseInUK` task) is discarded because $\mathcal{T} \not\models Out\_SearchForAUniversity \sqcap In\_RegisterForACourseInUK \sqsubseteq \bot$ while the service s_2 is highlighted because $\mathcal{T} \models Out\_SearchForAUniversity \sqsubseteq In\_RegisterForACour\text{-}seInUK$.

In our approach, the user can assign more than one service to a task, implying parallel execution of services from the back-end perspective. Therefore, the control flow of the template can be even modified on the fly, by adding new parallel branches. Such a modification is transparent to the end user, who are not interested in interacting with real control flow model of composition.

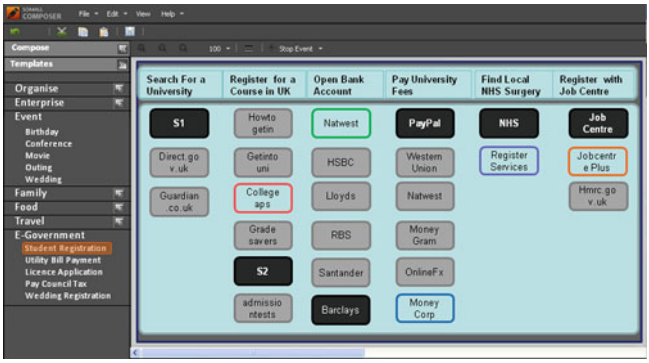

Fig. 28.6 Final composition

In case of parallel branches in the composition, the back-end tool is able to filter and merge data from these branches and connect them to the correct services. The latter is supported by the semantic link presented in (1).

Once the user has assigned services to every task of the template-based composition, the instantiation procedure is complete. Then, the composition is ready to be deployed and executed according to the the control and data flow information automatically elaborated respectively by the template description and the semantic links. Figure 28.6 depicts the final composition we obtain in our motivating scenario i.e., services in black are used to achieve the composition in Fig. 28.3.

Once the template is instantiated by the services selected by the user, the final process is ready to be deployed and then executed. In our approach the process is generated in BPEL4SWS [38] for subsequent analysis and processing by the service orchestrator. The orchestrator is responsible for scheduling, initiating, and monitoring the invocations to the tasks of the composite service during its execution, and for routing events and data items between these components.

28.5 Summative Evaluation of User Assisted Composition Tool

We have implemented a proof-of-concept implementation of the approach presented in the previous section as a module within the EC-funded integrated project SOA4All.[5] Following the technical development, we opted to test the usability and suitability of the User Assisted Composition tool (for short, UAC tool) for our target end users, non-programmers. These are people whose primary function in their jobs is not writing programming code; nonetheless, they might be involved in customising a software application to serve their personal or professional needs.

[5] More information on http://www.servicedesign.org.uk, last accessed on 30th Sept 2012.

Our selection of the participants was driven by (a) their profile which should match closely the target group of non-programmers developing process-oriented mashups, and (b) their familiarity with the applicaton domains of University enrollment and shopping. We have therefore aimed to recruit students from the Manchester Business School ensuring their IT competences are minimal. We hypothesise that with the sufficient domain knowledge and expertise other non-programmers will also be able to operate the tool comfortably, yet this requires further testing for which we need to develop specialised scenarios and services that fit with these settings.

Once we recruited participants, we have conducted two consecutive evaluation studies with the following objectives in mind:

- Assess the effectiveness of the User-Assisted Composition approach by analysing composition performance.
- Test the applicability and suitability of the tool in two differing scenarios, a shopping scenario and a university scenario.
- Gauge users positive views and negative views following direct interaction with the UAC tool; thus identifying the merits of user-assisted composition approach on the one hand and the limitations and problems on the other hand.
- Capture user impressions, satisfaction and acceptance of the UAC tool through a usability questionnaire.

The UAC approach provides a number of unique features as follows:

- The composition uses activity-based templates.
- Only simple user interface interaction skills, such as clicking, selecting etc, are required to operate the tool.
- The composition is mainly system-driven using semantic reasoning.
- Services are represented via boxes. No user interface is attached to them.
- No data flow or process flow connections are required. Instead the user can re-arrange the order of activities, and assign services for each activity through "point-and-click" interaction.
- User friendly error messages are displayed in case of mistakes.

28.5.1 Evaluation Tasks

Each evaluation session in both studies took approximately 50 min, and participants went through the same steps with the only variation to the evaluation scenario, where the first study focused on a shopping context and the second study focused on an education context. In our evaluation strategy, we selected scenarios that suit the profile of our target end users by recruiting participants who have sufficient knowledge about the tasks composing the shopping and university scenarios but who have no software programming or development experience. Our participant sample contained students from Manchester Business School enrolled for undergraduate and postgraduate courses. It is worth noting that for each study we recruited a different

set of participants to ensure no learning effects are carried over to the second study. All studies were moderated by the same researcher.

The evaluation studies consisted of three common phases: training, task-undertaking, and debriefing interview phase.

- **Training phase**: in 10 min the moderator demonstrated the UAC tool, explained its various features, and encouraged participants to ask questions in case of ambiguity.
- **Task-undertaking phase**: in this phase participants read the scenario and performed the designed tasks. During the interaction with the tool, participants communicated their mental thoughts using the think-aloud protocol [21]. Think-aloud protocol is a research technique used to capture users inner thinking about the way they undertake tasks and the type of problems they encounter. Participants spent around 30 min to complete the designated tasks.
- **Debriefing interview phase**: in this phase participants reported their individual views and opinions about the UAC tool, and rated their satisfaction toward numerous aspects of the UAC tool by scoring a set of questions on a 5-point Likert scale.

28.5.2 Analysis Methods

Throughout the two studies we recorded participants opinions and interaction using SnagIt software, a screen capturing program, and their ratings using a paper questionnaire. The video recordings were reviewed and transcribed into Microsoft Word for follow-up thematic analysis [17], whilst rating scores were inserted into SPSS (i.e. a statistical software package) for calculating descriptive statistics such as the mean and standard deviation.

28.5.3 Evaluation Study One

In the first study, we recruited a total of six students who included five males and one female (mean age = 26.5). These participants were enrolled for a postgraduate degree in Manchester Business School. This study aimed to gauge initial qualitative impressions and reactions from potential end users which justifies the small sample. A pre-test questionnaire to capture participants software development background showed that our sample fits well the definition of non-programmers as depicted in Fig. 28.7. Five participants had no or basic software development background whilst one participant had a strong software development background. Participants rated pre-test questions on a 5-point Likert scale where '1' signifies 'none', and '5' signifies 'expert'. Average scores for all programming and software development experience questions were less than 2.33.

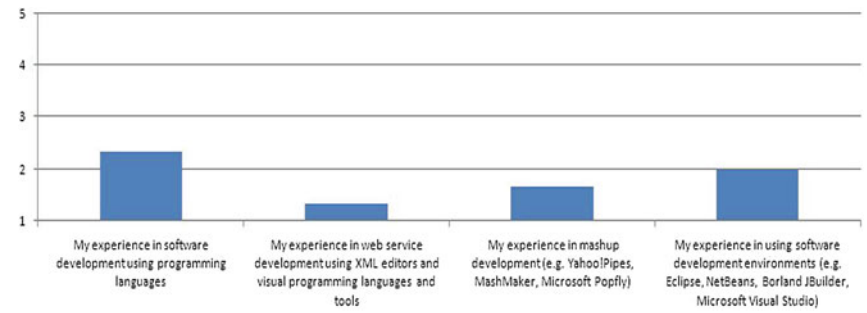

Fig. 28.7 Average scores of pre-test questionnaire

Next, each participant was instructed to go through two scenarios of varying complexity and perform the subsequent tasks:

1. **Scenario One**: suppose you own a reseller business/web shop. Your aim is to create a composition which gets the updated catalogues from clothing suppliers, aggregates the catalogues and publishes them in social networks and/or web shops. Your task is to compose a simple application which allows you to do the following:
   - Select the clothing suppliers whose catalogues you want to update.
   - Select the social networks(s)/web shop(s) where you want to aggregate and publish the desired catalogues.
2. **Scenario Two**: this time you want to build a composition which allows you to retrieve product descriptions and prices from specific suppliers and aggregate updated catalogues accordingly. Your task is to compose an application which allows you to achieve the following activities:
   - Update and aggregate catalogues from various clothing suppliers.
   - Get list of products from the desired footwear suppliers.
   - For each product get the product data and product price.
   - Aggregate the footwear products to the catalogue.
   - Retrieve list of product descriptions and prices and aggregate them to the catalogue.

In both scenarios, concurrent think-aloud protocol [21] was followed to get rich insights into the mental models of our users.

28.5.4 Results of Study One

In respect to user performance, all participants successfully completed the two scenarios using the UAC tool. It is worthwhile to note that in this study we were not

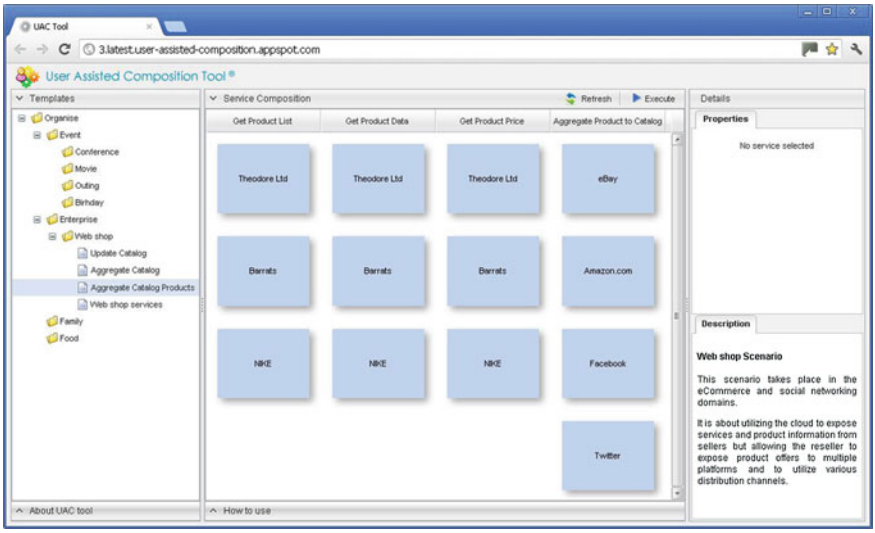

Fig. 28.8 User-assisted composition tool showing aggregate catalog products template

too concerned about the time taken to complete the tasks of the two scenarios but rather focused on user comments in relation to the composition approach and specific problems and ways to improve the tool.

Participants' feedback praised the ease by which they could operate the tool and navigate through the different sections. Also of increasing interest is the ability to compose service using the tool with no need to master programming concepts and paradigms, thus saving money and time. Instead participants were able to manage the composition with only a small number of clicks.

On the negative side participants were confused about the names of the services and activities, and found it difficult to match that to the requirements of the tasks. They also were unsure about why certain incompatible services can still be selected.

Following the composition, participants provided recommendations to help improve the tool. Among which were:

- Add a rating and description to services to empower end users to make an educated selection.
- Sort services alphabetically to facilitate search.
- Use self-explanatory names for the activities of templates to clarify their purpose.
- Enable end users to customise templates by re-arranging the sequence of activities.
- Use clear and distinct colours for selecting and de-selecting incompatible services.

Finally, participants rated their agreement with a number of statements to assess the usability of the UAC tool and their overall experience with service composition on a 5-point Likert scale, where '1' signifies disagree, '3' signifies neutral and '5' signifies agree. Indeed participants perceived the UAC tool as easy to use (with a mean $m = 3.83$) and navigate ($m = 3.66$), and did not find the notations used

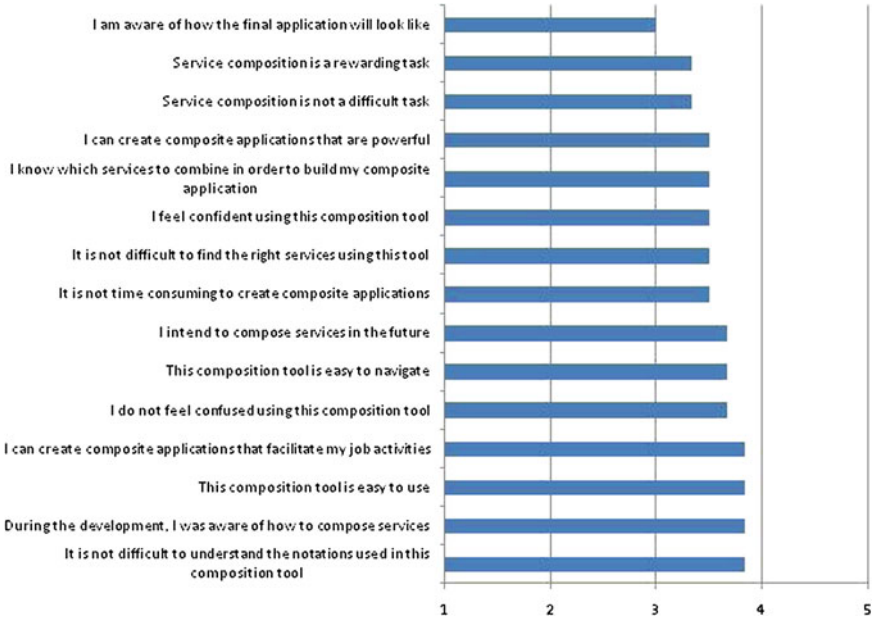

Fig. 28.9 Average rating scores of UAC tool

within the tool difficult to use. This experience improved user confidence that the tool allows end users to create composite applications which facilitate job functions (m = 3.83). However, participants expressed uncertainty in regard to the look and feel, and behaviour of the final application as depicted in Fig. 28.9. Similarly it was difficult for participants to evaluate how rewarding service composition is for they did not see the final application.

28.5.5 Evaluation Study Two

In the second study, we recruited a total of 12 Manchester University students by sending a screening questionnaire to the University student mailing list. The questionnaire collected information about programming and development experience, service modelling experience, background knowledge of software development environments and modelling tools, and general demographic information. We purposefully selected participants whose questionnaire scores match our requirements. Our sample included seven males and five females who study for non-Computer Science degrees such as: Business and Management, International Human Resource Management, Marketing, and Managerial Psychology. Participants rated their experience in respect to seven software development questions of the screening questionnaire on a 5-point Likert scale where '1' signifies Extremely poor, '3' signifies Average, and

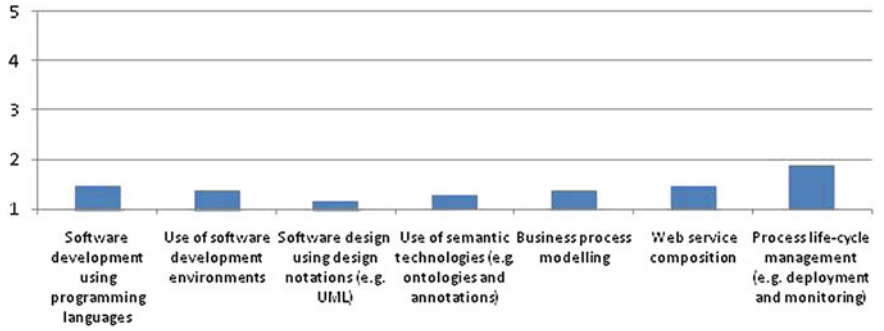

Fig. 28.10 Average scores of software development background questionnaire

'5' signifies Excellent. All scores averaged less than 1.9 as depicted in Fig. 28.10, indicating that our selected participants truly represent people who are not programmers.

We created a scenario with which our participants are familiar as it envisages the process they go through when applying to study at a particular UK university. The test scenario details the registration process overseas students go through while getting admission in UK universities as follows: Your goal is to complete an overseas student registration process. For this you need to develop a software application which allows you to search for a UK university, register for a course in the university and find an accommodation. There are two ways for paying the university fee, the first way is to open a bank account and get funds transferred into that account. The bank account can be used to make payment for the university fee. In the second way you can request a letter from a sponsor and submit that letter to the university. You must choose only one way to pay the university fee. After paying the university fee you will register with the NHS.

To accomplish the above scenario, participants were instructed to complete three primary tasks:

- **Task One**: Navigate to and load the appropriate activity-based template.
- **Task Two**: Remove the police registration activity from the template.
- **Task Three**: Select relevant services for each activity in the template according to the requirements of the test scenario.

During these development tasks, participants were continuously encouraged to express their views.

28.5.6 Results of Study Two

The objective evaluation of the UAC tool focused on measuring the average time taken to complete each task, along with the number of participants who have successfully

Table 28.1 Task completion time in seconds and number of participants who completed the task

Task	Average completion time (s)	# of users who completed the task
Finding and navigating to the right activity-based template Student Registration	21.25 (std = 12.99)	12
Removing 'Police Registration' activity from template	47.33 (std = 39.29)	11
Selecting appropriate services for activities	192.75 (std = 88.29)	12

Table 28.2 Average number of problems, positive comments and suggestions for the user assisted composition tool (STD: Standard Deviation)

Task	Average	STD
Positive comments	3.25	2.22
Overall problems	1.83	1.70
Conceptual problems	0.92	1.24
Usability problems	0.92	1.24
Suggestions	1.25	1.86

completed the tasks. The descriptive statistics revealed that participants spent the longest time inspecting the available services and selecting a relevant service for each template activity. However, participants were quicker to find the relevant template, and remove the police registration activity as summarised in Table 28.1. Of major interest is the ability of all of our participants to successfully complete the three tasks apart from one participant who did not manage to perform task 2 primarily due to the location of the remove button which was encapsulated within a pop-up menu. This high task completion rate reflects the effectiveness of the tool.

In regard to self-reported data, participants feedback and comments were analysed using thematic analysis [17], and classified into four categories; conceptual problems, usability problems, positive comments, and suggestions. Thematic analysis technique is qualitative in nature as a researcher goes through textual representation of an interaction, codes data segments, and creates general themes for the specific codes [17]. The results showed that, on average, participants were positive about the UAC tool, with 3.25 positive comments per participant as summarised in Table 28.2. As for problems, participants reported the same number of conceptual and usability problems, with an average of 0.92 negative comment per participant. Each participant provided at least one comment for ameliorating the UAC tool.

The positive feedback appreciated the presence of a 'how to use' section within the tool to help novice users grasp an understanding of the user assisted composition approach. They also found the tool intuitive, e.g. "it is easy to follow the logical steps", and easy to operate, e.g. "I just need to select the right services". The clickable nature of services and the ability to remove activities as well as services were appreciated

by our end users. Finally, participants praised the structure and the way services are laid out within the tool.

Among the conceptual problems that emerged was the ambiguity of how to start the composition process which could be attributed to participants unfamiliarity with the tool, e.g. "What should I do now? I do not know". A total of 5 participants read the how-to-use section to help them get started with the tool. Another issue participants brought up was the inability to view the outcome of their development efforts and test the developed application, e.g. "I can not see the results", "It was easy but I wish I could see the end result so I could understand what I have done". This is aligned with the findings of [35] where users emphasised the need to see runtime results as the development process unfolds. It is quite important to inspect the behaviour of the application and debug any unapparent problems. Some participants proclaimed that the terminology and language used in the 'help and how to use the tool' sections were somewhat technical and complex, for instance: "press execute to deploy composition". Finally, participants highlighted that the tool does not provide any explanation as to why certain services get excluded upon selecting others.

In respect to usability problems, participants complained about the lack of text to describe the purpose of each service. This could greatly enhance their choice of target services and allocation of those to template activities. Some participants were uncertain as to why some services were greyed out upon selecting a particular service. However, this confusion diminished as soon as the experimenter explained that the greying out feature is used to highlight any incompatible services. The context menu for removing activities from the template was not apparent to one participant, and it proved to be cumbersome to find its location. Another aspect that worried our participants is the inconvenience that could be caused by the presence of many activities and their associated services in which case they would have to scroll horizontally and vertically to view them. Table 28.3 summarises the issues participants encountered when using the UAC tool.

To resolve the aforementioned issues and improve the overall composition experience using the UAC tool, participants recommended and discussed a number of potential solutions as follows:

R1 Supplement the activities and services with further details (e.g. service properties and provider information) to allow users to differentiate between services and make an educated selection of services. These details could be shown, for example, in the form of tool tips when mouse hovering upon services. Indeed

Table 28.3 Conceptual and usability issues emerging from UAC tool

Conceptual	Usability
Runtime effect	Lack of text description
Terminology and language	Invisible options (e.g. removal of activities)
Compatibility of services	Scalability of the service template
	Unclear use of colour codes

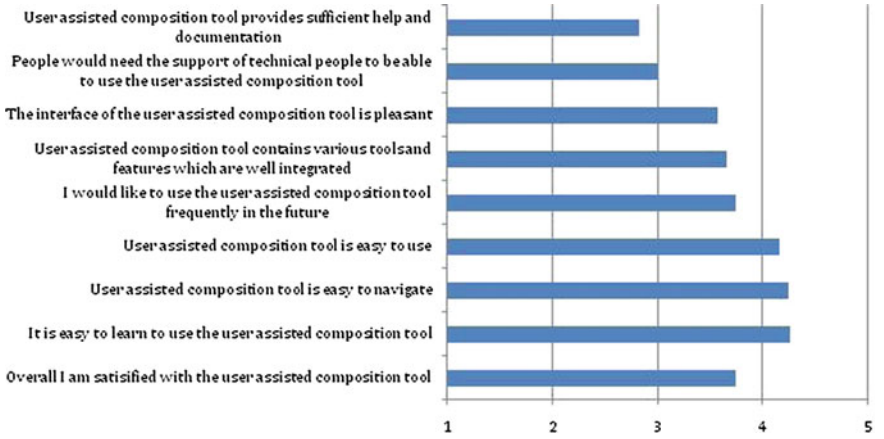

Fig. 28.11 Average rating scores of user-assisted composition tool

the composition process is highly-reliant on the careful selection of relevant services which, in turn, necessitates direct comparison of the features of multiple services accomplishing the same activity.

R2 Simplify the terminology used in the how to use section, and make it as close as possible to end user language.

R3 Provide explanation as to why certain services are incompatible, and couple this with distinct and clear use of colours.

R4 Provide a wizard to guide developers through the selection process of services and activities, especially at the start of the composition.

R5 Add a quick option (e.g. x in windows or a removal button) on the right corner of each activity column to allow fast removal of activities.

R6 Include runtime effects in the composition to empower end users to debug and test their application on the go.

Finally, participants concluded their user-assisted composition interaction tasks by rating a number of usability questions to assess ease of use, ease of learning, ease of navigation, user interface, help and documentation, and overall satisfaction with the tool on a 5-point Likert scale, where '1' signifies disagree, '3' signifies neutral, and '5' signifies agree.

There was a common consensus among our participants that the User Assisted Composition tool is easy to learn ($m = 4.27$, $std = 0.90$) and easy to use ($m = 4.16$, $std = 0.93$) as shown in Fig. 28.11. Similarly, participants agreed that the UAC tool is easy to navigate ($m = 4.25$, $std = 0.96$) and did not think that support from technical people is required to operate the tool ($m = 3.00$, $std = 1.04$). These scores confirm that end users can exploit the advantages of user-assisted composition approach without the need to master the underlying technical details of service composition and process modelling. Our participants also expressed strong willingness to use the

UAC tool more frequently in the future (m = 3.75, std = 1.28), and overall were satisfied with the tool (m = 3.75, std = 0.86).

28.6 Conclusion

We have established, through two summative evaluation studies, the effectiveness and suitability of the user assisted composition approach for non-programmers. The simplicity of point and click approach enabled our participants to complete the composition tasks with no major issues despite the short training they received. A number of interesting points emerged, including R1 to R6 listed above and especially the visibility of runtime operation. These will shape our future research directions. Every time participants were to select a service for an activity of the template, they expressed uncertainty and indicated that more service information should be displayed to assist them. In our view, text description of what the service does alone is not sufficient to overcome this issue, but rather more quality characteristics and criteria, such as reliability, reputation, and usability, should be exposed to enable informed service selection.

Our studies including the findings, however, have some limitations. First, it was not possible to demonstrate and test the final composite application which we believe would have strongly influenced user perception in regard to the overall composition experience. Indeed our participants found the composition process easy to perform and straight forward using the UAC tool; however, participants primary concern focused on showing runtime effects and ability to examine the final application.

Secondly, it is quite challenging for the moment to achieve a meaningful comparison between the UAC tool and other existing mashup and service development tools due to a number of qualifying reasons. First experimental design necessitates changing one variable whilst keeping the rest constant to study the influence of the variable. This means creating two similar versions of the UAC tool and varying one feature to test its influence. Testing the UAC tool against another totally different service development tool would not allow us to establish causal relationships but instead inspect mainly user interface issues. Moreover, the selected tools for comparison should be able to accomplish the same scenario and employ the same services for the comparison to be valid, or at least recruit scenarios of similar complexity. In addition, the purpose of this paper was to detail the approach and validate its feasibility with representative end users as an initial step rather than to compare the proposed tool against other tools. We argue that comparing different tools which support the same type of composition (e.g. service composition using service frond-ends) would not yield interesting results, but rather shed light on only usability issues. For the comparison to be scientific and valid, the plan is to extend the current UAC tool with other composition approaches. Thus, we intend in the future to conduct a series of comparative studies where we contrast overall service development experience using differing service development paradigms within a single tool. In particular we aim to investigate user interface based composition, process based composition, and

dataflow based composition. The concepts behind these paradigms and how they support the service development process are indeed interesting and worthwhile to explore.

In light of these limitations, we plan to undertake various steps to improve and qualify our research in the future in addition to these comparative studies. To start with, we will recruit participants who are domain experts and practitioners in various sectors such as the public and health sector, and increase our sample size to a satisfactory number. Moreover, in the next evaluations we will empower end users to see and test the final application of their composition, interact with it, and debug it throughout the composition.

In summary, the qualitative and the quantitative feedback obtained demonstrate that our end users understood the principles of the assisted composition approach, were positive about it and were able to accomplish the tasks set to them. Following our earlier work [31, 32], we believe that this is partially due to the benefits expected from service composition in terms of producing mashups which are finely tuned to the user needs, and partially due to the reduction of the learning costs as perceived by the user. The reduction in learning costs is attributed to our two main contributions presented in this paper: the approach of hiding technical complexity using semantic reasoning, and the reuse possible by the template-based development process.

References

1. Baader, F., Nutt, W.: In: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
2. Beaton, J.K., Myers, B.A., Stylos, J., Jeong, S.Y.S., Xie, Y.C.: Usability evaluation for enterprise SOA APIs. In: SDSOA '08: Proceedings of the 2nd International Workshop on Systems Development in SOA Environments, pp. 29–34. ACM, New York (2008). doi:10.1145/1370916.1370924
3. Berardi, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Mecella, M.: Automatic composition of e-services that export their behavior. In: Proceedings of the 1st International Conference on Service Oriented Computing (ICSOC), pp. 43–58 (2003)
4. Blackwell, A.F.: First steps in programming: a rationale for attention investment models. In: Proceedings of HCC '02, p. 2. IEEE CS, Washington (2002)
5. Brandt, S., Kusters, R., Turhan, A.: Approximation and difference in description logics. In: Proceedings of KR, pp. 203–214 (2002). http://www.citeseer.ist.psu.edu/brandt02approximation.html
6. Canfora, G., Penta, M.D., Esposito, R., Villani, M.L.: An approach for qos-aware service composition based on genetic algorithms. In: Proceedings of GECCO, pp. 1069–1075 (2005)
7. Carlson, M.P., Ngu, A.H., Podorozhny, R., Zeng, L.: Automatic mash up of composite applications. In: Proceedings of the 6th International Conference on Service-Oriented Computing, ICSOC '08, pp. 317–330. Springer, Berlin (2008). doi:10.1007/978-3-540-89652-4_25
8. Colucci, S., Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. Electron. Commer. Res. Appl. **4**(4), 345–361 (2005)
9. Daniel, F., Casati, F., Benatallah, B., Shan, M.C.: Hosted universal composition: models, languages and infrastructure in mashart. In: Laender, A., Castano, S., Dayal, U., Casati, F., de Oliveira, J. (eds.) Conceptual Modeling—ER 2009. Lecture Notes in Computer Science, vol.

5829, pp. 428–443. Springer, Berlin (2009). http://dx.doi.org/10.1007/978-3-642-04840-1_32. doi:10.1007/978-3-642-04840-1_32
10. Daniel, F., Koschmider, A., Nestler, T., Roy, M., Namoun, A.: Toward process mashups: key ingredients and open research challenges. In: Proceedings of the 3rd and 4th International Workshop on Web APIs and Services Mashups, Mashups '09/'10, pp. 9:1–9:8. ACM, New York (2010). doi:10.1145/1944999.1945008. http://doi.acm.org/10.1145/1944999.1945008
11. Daniel, F., Soi, S., Casati, F.: Distributed user interface orchestration: on the composition of multi-user (search) applications. In: Ceri, S., Brambilla, M. (eds.) Search Computing, Lecture Notes in Computer Science, vol. 6585, pp. 182–191. Springer, Berlin (2011). http://dx.doi.org/10.1007/978-3-642-19668-3_17. doi:10.1007/978-3-642-19668-3_17
12. Deutch, D., Greenshpan, O., Milo, T.: Navigating in complex mashed-up applications. Proc. VLDB Endow. **3**(1–2), 320–329 (2010). http://dl.acm.org/citation.cfm?id=1920841.1920885
13. Ennals, R.J., Garofalakis, M.N.: Mashmaker: mashups for the masses. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD '07, pp. 1116–1118. ACM, New York (2007). doi:10.1145/1247480.1247626
14. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Berlin (2007)
15. Fensel, D., Kifer, M., de Bruijn, J., Domingue, J.: Web service modeling ontology submission, w3c submission (2005)
16. Fischer, G., Nakakoji, K., Ye, Y.: Metadesign: guidelines for supporting domain experts in software development. IEEE Softw. **26**(5), 37–44 (2009). doi:10.1109/MS.2009.134
17. Guest, G., MacQueen, M.K., Namey, E.: Applied Thematic Analysis. SAGE Publications Inc, New Delhi (2012)
18. Han, J., Han, Y., Jin, Y., Wang, J., Yu, J.: Personalized active service spaces for end-user service composition. In: IEEE International Conference on Services Computing, 2006, SCC '06, pp. 198–205 (2006). doi:10.1109/SCC.2006.80
19. Horrocks, I.: Using an expressive description logic: Fact or fiction? In: Proceedings of KR, pp. 636–649 (1998)
20. Hull, R., Benedikt, M., Christophides, V., Su, J.: E-services: a look behind the curtain. In: Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '03, pp. 1–14. ACM, New York (2003). doi:10.1145/773153.773154
21. Kuusela, H., Paul, P.: A comparison of concurrent and retrospective verbal protocal analysis. Am. J. Psychol. **113**(3), 387–404 (2000)
22. Lécué, F., Léger, A.: A formal model for semantic web service composition. In: Proceedings of ISWC, pp. 385–398 (2006)
23. Lécué, F., Mehandjiev, N.: Seeking quality of web service composition in a semantic dimension. IEEE Trans. Knowl. Data Eng. **23**(6), 942–959 (2011)
24. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: Proceedings of WWW, pp. 331–339 (2003)
25. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm, Human-Computer Interaction Series, vol. 9. Springer, Netherlands (2006). doi:10.1007/1-4020-5386-X_1. http://dx.doi.org/10.1007/1-4020-5386-X_1
26. Liu, X., Hui, Y., Sun, W., Liang, H.: Towards service composition based on mashup. In: Proceedings of IEEE Congress on Services, pp. 332–339 (2007). doi:10.1109/SERVICES.2007.67
27. Martinez, A., Patino-Martinez, M., Jimenez-Peris, R., Perez-Sorrosal, F.: Zenflow: a visual web service composition tool for BPEL4WS. In: Proceedings of VLHCC'05, pp. 181–188. IEEE Computer Society, Washington, (2005). doi:10.1109/VLHCC.2005.74
28. McIlraith, S.A., Son, T.C.: Adapting Golog for composition of semantic web services. In: Proceedings of KR, pp. 482–496 (2002)
29. Mehandjiev, N., Lécué, F., Wajid, U.: Provider-composer negotiations for semantic robustness in service compositions. In: Proceedings of ICSOC/ServiceWave, pp. 205–220 (2009)
30. Mehandjiev, N., Namoun, A., Wajid, U., Macaulay, L., Sutcliffe, A.: End user service composition—perceptions and requirements. In: Proceedings of 8th IEEE European Conference on Web Services ECOWS'2010 (2010, to appear)

31. Mehandjiev, N., Stoitsev, T., Grebner, O., Scheidl, S., Riss, U.: End-user development for task management: Survey of attitudes and practices. In: Proceedings of VLHCC '08, pp. 166–174. IEEE Computer Society, Washington (2008). doi:10.1109/VLHCC.2008.4639079
32. Mehandjiev, N., Sutcliffe, A., Lee, D.: Organizational view of end-user development. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, Human-Computer Interaction Series, vol. 9, chap. 17, pp. 371–399. Springer, Netherlands (2006). doi:10.1007/1-4020-5386-X_17. http://dx.doi.org/10.1007/1-4020-5386-X_17
33. Mørch, A.I., Mehandjiev, N.D.: Tailoring as collaboration: the mediating role of multiple representations and applicationunits. Comput. Support. Coop. Work **9**(1), 75–100 (2000). doi:10.1023/A:1008713826637
34. Motta, E.: Parametric Design Problem Solving—Reusable Components for Knowledge Modelling Case Studies. IOS Press, Amsterdam (1999)
35. Namoun, A., Nestler, T., De Angeli, A.: Service composition for non-programmers: prospects, problems, and design recommendations. In: Proceedings of IEEE 8th European Conference on Web Services (ECOWS), pp. 123–130 (2010). doi:10.1109/ECOWS.2010.17
36. Namoun, A., Wajid, U., Mehandjiev, N.: A comparative study: application development by ordinary internet users and it-professionals. In: Proceedings of ServiceWave'2010. Springer, Berlin (2010, to appear)
37. Nardi, B.A.: A Small Matter of Programming: Perspectives on End User Computing. MIT Press, Cambridge (1993)
38. Nitzsche, J., Norton, B.: Ontology Based Data Mediation in BPEL (for Semantic Web Services). Springer, New York (2008)
39. Paolucci, M., Kawamura, T., Payne, T., Sycara, K.: Semantic matching of web services capabilities. In: Proceedings of ISWC, pp. 333–347 (2002)
40. Pistore, M., Roberti, P., Traverso, P.: Process-level composition of executable web services: "on-the-fly" versus "once-for-all" composition. In: Proceedings of ESWC, pp. 62–77 (2005)
41. Rode, J., Rosson, M.B., Pérez-Qui nones, M.A.: End-users' mental models of concepts critical to web application development. In: Proceedings of VLHCC '04, pp. 215–222. IEEE Computer Society, Washington (2004). doi:10.1109/VLHCC.2004.25
42. Sirin, E., Hendler, J.A., Parsia, B.: Semi-automatic composition of web services using semantic descriptions. In: Proceedings of WSMAI, pp. 17–24 (2003)
43. Sutcliffe, A.: Domain Theory: Patterns for Knowledge and Software Reuse. L. Erlbaum Associates Inc., Hillsdale (2002)
44. Sutcliffe, A., Mehandjiev, N.: Introduction. Commun. ACM **47**(9), 31–32 (2004). doi:10.1145/1015864.1015883
45. Teege, G.: Making the difference: a subtraction operation for description logics. In: Proceedings of KR, pp. 540–550 (1994). http://www.citeseer.ist.psu.edu/teege94making.html
46. ten Teije, A., van Harmelen, F., Wielinga, B.: Configuration of web services as parametric design. In: Motta, E., et al. (ed.) Proceedings of EKAW-2004, LNAI, vol. 3257, pp. 321–336. Springer, Heidelberg (2004). ISBN 3-540-23340-7
47. Westerski, A.: Integrated environment for visual data-level mashup development. In: Proceedings of WISE '09, pp. 481–487. Springer, Berlin (2009). doi:10.1007/978-3-642-04409-0_47
48. Wielinga, B., Schreiber, G.: Configuration-design problem solving. IEEE Expert Intell. Syst. Appl. **12**(2), 49–56 (1997)
49. Wong, J., Hong, J.I.: Making mashups with Marmite: towards end-user programming for the web. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '07, pp. 1435–1444. ACM, New York (2007). doi:10.1145/1240624.1240842
50. Yu, T., Lin, K.J.: Service selection algorithms for composing complex services with multiple QoS constraints. In: Proceedings of ICSOC, pp. 130–143 (2005)
51. Zang, N., Beth, R.M.: What's in a mashup? And why? Studying the perceptions of web-active end users. In: Proceedings of VLHCC'08, pp. 31–38. IEEE Computer Society, Washington (2008). doi:10.1109/VLHCC.2008.4639055
52. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality driven web services composition. In: Proceedings of WWW, pp. 411–421 (2003)