WISHFUL AND WISDOM AWARE COMPOSING
A Full User-centric Approach to Create NGM

Luis Javier Suarez Meza[1], Luis Antonio Rojas Potosi[1], Juan Carlos Corrales[1] and Oscar Rodriguez Rocha[2]

[1] *Grupo de Ingenieria Telematica (GIT), Universidad del Cauca, Popayan, Colombia*

[2] *Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy*

Keywords: Wishful and Wisdom Aware Composing, User-centric, Next Generation Mashups.

Abstract: Nowadays, it's possible to find huge amounts of steadily increasing web resources. Service composition languages and tools are widely used for creating compositions of multiple services in order to meet the user's requirements in the cases when a single Web service does not perform a required task. However, despite the relative intuitiveness of currently the available tools, they still require a careful manual assembly of a composition flow by the end-user, thus requiring some technical knowledge on the functioning of each individual component. We provide a full user-centric approach to create NGMs (Next Generation Mashups) interactively through W2AC (Wishful and Wisdom-Aware Composing): First, end-user wishes are considered, then a composition knowledge is extracted from existing reputable mashups created by skilled users. Thus, end-users are able to easily generate their own NGM in a fully customized fashion.

1 INTRODUCTION

The Wishful and Wisdom concepts are the foundations of our work. Generally, in many related works (Chowdhury et al., 2010; Riabov et al., 2008; De Angeli et al., 2011; Casati, 2011), these concepts are considered in a separated fashion. We take advantage of the potential of this combination (users' desire + collective knowledge), to embrace a resource composition focused on the end-user. To explain our W2AC vision, some concepts are introduced.

Our approach is based on BDI systems, which has a modal component to reason about propositional attitudes: beliefs, desires and intentions (BDI) (Dastani and Steunebrink, 2009). In this sense, we cover these aspects under the *Wishful* concept. Thus, we aim to determine what the user really wants from an explicit request made in natural language (Pedraza et al., 2011), since it is difficult to propose to the user a service that meets his/her search requirements, without knowing the true meaning of what he/she really wants.

Wisdom is a concept closely related to the paradigm of collective intelligence (CI) (Agarwal et al., 2010; Dalal, 2008). Generally, some approaches that have considered CI, argue that the users do not need a high level of expertise in service creation platform for composing mashups, if they have an adequate assistance, advice, or help by another users, who have previously solved the same (or similar) problem (Szuba et al., 2011; Maries and Scarlat, 2011).

On the other hand, many research efforts have been done on automatic composition, especially within the AI planning and Semantic Web communities. Other work uses process models or formal representation (e.g., Graphs) (Maaradji et al., 2011; Chowdhury et al., 2011). Further, current approaches about composition work on resources of the same type (i.e., just on Web services, like BPEL) (Riabov et al., 2008). The review on the state of the art shows there are currently no approaches that dare to make a combination of the great diversity of existing resources. Wasting the diversity of available resources and limiting the emergence of novel and interesting resources. In this context, we propose a combination of web resources called NGM. NGM is an hybrid integration of different types of available resources created by end-users. We provide an approach to create NGM's interactively through W2AC: First, end-user wishes are considered, then a composition knowledge is extracted from existing reputable mashups or different resources created by other more technically skilled users.

Suarez Meza L., Rojas Potosí L., Corrales J. and Rocha O..
WISHFUL AND WISDOM AWARE COMPOSING - A Full User-centric Approach to Create NGM.
DOI: 10.5220/0003940108030806
In *Proceedings of the 8th International Conference on Web Information Systems and Technologies* (WEBIST-2012), pages 803-806
ISBN: 978-989-8565-08-2

2 THE USERS' DESIRES COMPOSER: W2AC

This section presents the formal definition of the main components of an NGM: a query model that describes the formal user request and a Mashup model to derive the semantic description of an NGM, based on the descriptions of its individual components. A fundamental characteristic of our model is that it captures not only the semantics of inputs/outputs (and its functional dependency), and operations, but also the semantics of control operator structures (i.e., composition structure patterns).

2.1 Query Model

Currently there is no a model that represents the formal request of end-users, therefore, below we present our proposal for this concern.

2.1.1 User Request (Q_0)

This kind of request it is an informal query, which represents the desired results from users. Thus, these queries are used to describe the compositions goals or to specific the input conditions for the service retrieval stage (very phase important into composition process).

2.1.2 Definition 1 (Formal User Request Q)

Let Q_0 the informal user request expressed in natural language before its analysis. A formal user request Q is defined as a tuple $Q = (userID, F, NF, C, \lambda)$ where *userID* is the identifier of the user that perform the request, F is a non-empty finite set of elements that represent *Functional Words*, such that $F = \{f_1, ..., f_n\}$. NF is a finite set of elements that represent *Non-Functional Words*, such that $NF = \{nf_1, ..., nf_m\}$. F and NF are distinct, $F \cap NF = \emptyset$. C is a pair $C = (W, P)$ where W is a non-empty finite set of words that denote *Control Words* (e.g., If, which, and, or, etc.), such that $W = \{c_1, ..., c_m\}$, and P is a finite set of elements that represent *Punctuation Marks* (,.;:). $\lambda = (F \times C)$ is the function that records the sentence meaning/structure, which is helpful to generate the logical mashup model (which is described in detail in the following section).

Elements both F and C can represent *Operators* $O = \{o\}$. A *Operator* is a basic unit both retrieval and composition phases. Generally, o represents one or more abstract services from a subset of existing real services. NF is considered a *Parameter* $P = \{p\}$ used to refine the ranking of retrieved services.

2.1.3 Definition 2 (Folksonomy)

Folksonomies are being widely used in various tagging applications of the Social Web. Folksonomies reflect through tags the collective intelligence of a crowd or a community (Wisdom of the Crowds) in giving meaning to available resources (Power Tags)(Helic et al., 2011).

Three main entities are identified in our proposal: the user $U = \{u_1, ..., u_n\}$, the resources $R = \{r_1, ..., r_m\}$ and the tags $T = \{t_1, ..., t_k\}$. Users annotate the resources with tags, creating triple associations between the user, the resource and the tag. Thus, the folksonomy can be defined by a set of annotations $A \subseteq U \times R \times T$. A folksonomy can be considered as an specific case of a *Taxonomy* τ, i.e., if τ is formed as a folksonomy by people specifying one or more tags t_j to describe certain objects (in this case *operators*, which represent web resources), the tags in τ are unrelated and τ is completely unstructured. Introducing a taxonomy structure in τ, enhances query expressivity, and also helps keep tag-based descriptions succinct(Helic et al., 2011). In this sense, according to above definitions, both *Operators* (*Resources*) o and *Parameters* p can be described by a specific set of tags $d(o) \subseteq \tau$ and $d(p) \subseteq \tau$ respectively, selected from the taxonomy τ.

2.1.4 Definition 4 (Tag Query q)

In general, a tag query $q \subseteq T \in \tau$ selects a subset ψ of an operator set $O = \{o\}$ such that each operator in the selected subset is described by all tags in q, taking into account sub-tag relationships between tags, i.e., if a tag $t_1 \in \tau$ is a sub-tag of $t_2 \in \tau$, denoted $t_1 \sim t_2$. Therefore, according to this, formally we have:

- $\psi_f(o) = \{o \in O | \forall t \in q_f \exists t' \in d(o) : t' \sim t \wedge \forall f_j \in F, \exists q_f : q_f \subseteq T\}$
- $\psi_c(o) = \{o \in O | \forall t \in q_c \exists t' \in d(o) : t' \sim t \wedge \forall c_j \in C, \exists q_c : q_c \subseteq T\}$
- $\psi_{nf}(p) = \{p \in P | \forall t \in q_{nf} \exists t' \in d(p) : t' \sim t \wedge \forall nf_j \in NF, \exists q_{nf} : q_{nf} \subseteq T\}$

Where: $\psi_f(o)$ and is an operator subset of all operator set $O = \{o\}$ such that each operator in this subset is described by all tags in q_f (set of *functional word* tags). Thus, for each $f \in F$ there is a $q_f \subseteq T$. $\psi_c(o)$ is an operator subset of all operator set $O = \{o\}$ such that each operator in this subset is described by all tags in q_c (set of *control word* tags).). Thus, for each $c \in W$ there is a $q_c \subseteq T$. $\psi_{nf}(p)$ is an parameter subset of all parameter set $P = \{p\}$ such that each parameter in this subset is described by all tags in q_{nf} (set of *non-functional word* tags). Thus, for each $n_f \in NF$ there is a $q_{nf} \subseteq T$.

2.2 NGM Model

A data mashup model can be expressed as a tuple $m = \{userID, name, T, O, C, M, reputation\}$, where *userID* is the identifier of user that perform the request, name is the unique name of the mashup, T is the set of tags that describes it, O is the set of operators used in the mashup. C is the set of data flow connectors ruling the propagation of data among operators, M is the set of data mappings of output attributes to input parameters of connected operators, and reputation counts how many times the mashup m has been used (e.g., to compute rankings). Specifically:

2.2.1 Definition 5, Operators (O)

At a *logical level* operators O_l are defined as a set $O_l = \{O_{li} | O_{li} = (name_i, T_i)\}$ with $name_i$ being the unique name of the operator o_{li} and T_i represents a description based on tags of the o_{li} (from the some user). However, at an *executable level*, i.e., of composition patterns, which include sequence operations, parallel operations, etc. $O_p = \{O_{pi} | O_{pi} = (In_i, Out_i, Op_i)\}$ is a non-empty set of operators, where $In_i = \{in_{i0}, ..., in_{ij}\}$, $Out_i = \{out_{i0}, ..., out_{ik}\}$ and $Op_i = \{Op_{i0}, ..., Op_{il}\}$ are respectively the sets of *input*, *output*, and *operations* of an operator op_i. Thus, the set of *Operators* O is defined as: $O = O_l \cup O_p$. We distinguish three kinds of operators:

- *Source operator*s, which fetch data from the web or the local machine. They don't have inputs, i.e., $In_i = \emptyset$.
- *Typical operators*, which consume data in input and produce processed data in output. Therefore, $In_i, Out_i \neq \emptyset$.
- Control operators, which are composition structure patterns: *Sequential*, *AND-Split* (Fork), *XOR-Split* (Conditional), *AND-Join* (Merge) and *XOR-Join* (Trigger)(Yu et al., 2007).

2.2.2 Definition 6, Data Flow Connectors (C)

Let's $C = \{c_m | c_m \in O \times O : C \cap O = \varnothing\}$ the data flow connectors that assign to each operator o_j its predecessor o_k (where: $j \neq k$) in the data flow.

2.2.3 Definition 7, Data Mapping (M)

Let's M the data mapping represents the set of data mapping of the data flow from output parameters of an operator o_j to input parameters of the predecessor operator o_k (where: $j \neq k$), as follows: $M = \{m_n | m_n \in In \times Out : In \cap Out = \varnothing, In = \cup_{i,j} in_{i,j}, Out = \cup_{i,j} out_{ik}\}$ In order to better understand the formalisms defined above, the Figure 1 shows our proposal for a Mashups' meta-model, which is indeed very simple: only requires 13 concepts suffice to model its composition features at an executable level (abstractness).

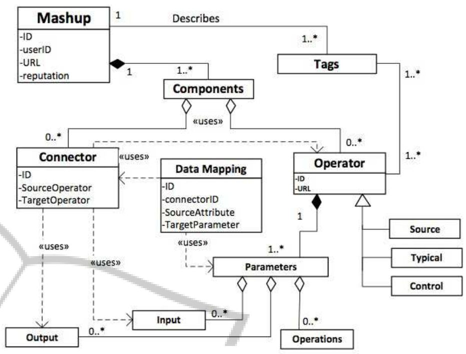

Figure 1: NGM metamodel.

Given the described models of query Q' and Mashup M we create the mashup m meets the user's request from the large number and variety of resources on the currently Web in two ways: firts, we generate a *Logical Mashup Model* (LMM) by analyzing the user's request in natural language and then, we generate an *Executable Mashup Model* (EMM abstractness) by matching *LMM* against our knowledge base. Thus, the Algorithm on Figure 2, details this strategy and summarizes the logic implemented by the generation of *EMM*.

Algorithm 1. General EMM

```
1.  INPUTS: Q0, User u                  /*Q0 informal query, which represents the desired results from User u*/
2.  OUTPUT: EMM                         /*it is the executable mashup model EMM (abstractness)*/
3.  BEGIN
4.  Q ← NLA(Q0)                                      /* get formal query from Q0*/
5.  Let LMM ← GetLogicalMashupModel(Q)               /* get LMM from Q */
6.  Let EMM ← GetEmmFromRepository(Q)                /* get EMM previously generated by the user or another*/
7.  if EMM != null then
8.     return EMM
9.  else
10.    Let Q(f) ← getFunctionalTags(Q)               /* Q(f) = {qfi, i= 1, ...,n} */
11.    Let Q(c) ← getControlTags(Q)                  /* Q(c) = {qci, i= 1, ...,m} */
12.    LMM ← getLogicalStructure(Q, Q(f)×Q(c))       /* Q(f) ∩ Q(c) = Ø, generate a logical mashup model(LMM)*/
13.    Let O = {o}                                   /* Operator Set*/;
14.    for each qfi in Q(f) do
15.       Qf (O) ← GetRankedOperatorsFromComponentRepository(qfi)   /* Qf (O) ∈ {Source, Typical}*/
16.       Let RQf ← RQf ∪ [Qf (O), qfi]               /*add Qf (O) to set RQf, ordered by qfi */
17.    end for
18.    for each qci in Q(c) do
19.       Qc (O) ← GetRankedOperatorsFromComponentRepository(qci)   /* Qf (O) ∈ {Control}*/
20.       Let RQc ← RQc ∪ [Qc(O), qci]                /*add Qc (O) to set RQc, ordered by qci */
21.    end for
22.    EMM ← genExecutableStructure(LMM, RQf, RQc)
23. end if
24. setEmmToRepository(EMM, u)                       /* stores the generated EMM in the repository*/
25. return EMM
26. END
```

Figure 2: General EMM algorithm.

In line 4, we get the formal query Q from the user's request Q_o in natural language by the *NLA* function (Natural Language Analyzer)(Pedraza et al., 2011). Then the $GetLogicalMashupModel()$ function gets the *LMM* from the Q. In line 6, the $GetEmmFromRepository()$ function gets an *EMM* abstractness, which has been previously generated by the same user or other users of our platform. If the algorithm finds an exact or similar *EMM*, it is recommended to the user, avoiding the whole process of composition. In the absence of an *EMM* that satisfies the user's request, the *EMM* is composed based on retrieved operators and the *LMM* obtained (between lines 10 and 22). Finally, the *EMM* generated is stored in a Repository of abstractness *EMM*.

3 CONCLUSIONS AND FUTURE WORK

The result of our research indicates that there is still a lack approaches to provide a feasible solution for end-users to mash the great diversity of existing resources. In this paper, we proposed an hybrid integration of different types of available web resources. We call this combination Next Generation Mashups (NGM). To achieve this, we define a user-centric approach to create NGMs based on W2AC (Wishful and Wisdom Aware Composing), a composition paradigm that aims at determining what ordinary users really want (Wishes) from a request in natural language, to finally deliver them the best solution that meets their needs (without requiring programming or technical skills). To do this, we define two meta-models, one to describe the user's request and another to represent the NGMs. Currently we have implemented the module that generates the *LMM* described previously. The next step of this work is to study and define new features that extend the NGM meta-model.

ACKNOWLEDGEMENTS

The authors would like to thank Universidad del Cauca, COLCIENCIAS and TelComp2.0 Project for supporting the Research of the M.Sc. Student Luis Javier Suarez.

REFERENCES

Agarwal, N., Galan, M., Liu, H., and Subramanya, S. (2010). Wiscoll: Collective wisdom based blog clustering. *Inf. Sci.*, 180:39–61.

Casati, F. (2011). How end-user development will save composition technologies from their continuing failures. In *Proc. of the 3th international conference on IS-EUD'11*, pages 4–6, Berlin, Heidelberg.

Chowdhury, S. R., Daniel, F., and Casati, F. (2011). Efficient, interactive recommendation of mashup composition knowledge. In Kappel, G., Maamar, Z., and Motahari-Nezhad, H. R., editors, *ICSOC*, volume 7084 of *LNCS*, pages 374–388. Springer.

Chowdhury, S. R., Rodríguez, C., Daniel, F., and Casati, F. (2010). Wisdom-aware computing: on the interactive recommendation of composition knowledge. In *Proceedings of the ICSOC*, pages 144–155, Berlin, Heidelberg. Springer-Verlag.

Dalal, N. (2008). Wisdom networks: Towards a wisdom-based society. pages 11–18. Springer Berlin Heidelberg.

Dastani, M. and Steunebrink, B. (2009). Modularity in bdi-based multi-agent programming languages. In *Proceedings of the 2009 IEEE/WIC/ACM*, WI-IAT '09, pages 581–584, Washington, DC, USA.

De Angeli, A., Battocchi, A., Chowdhury, S. R., Rodriguez, C., Daniel, F., and Casati, F. (2011). End-user requirements for wisdom-aware eud. In *Proceedings of the 3th IS-EUD'11*, pages 245–250, Berlin, Heidelberg. Springer-Verlag.

Helic, D., Strohmaier, M., Trattner, C., Muhr, M., and Lerman, K. (2011). Pragmatic evaluation of folksonomies. In *Proceedings of the 20th WWW '11*, pages 417–426, New York, NY, USA. ACM.

Maaradji, A., Hacid, H., Skraba, R., and Vakali, A. (2011). Social web mashups full completion via frequent sequence mining. In *Proceedings of the SERVICES '11*, pages 9–16, Washington, DC, USA. IEEE Computer Society.

Maries, I. and Scarlat, E. (2011). Enhancing the computational collective intelligence within communities of practice using trust and reputation models. LNCS, pages 74–95. Springer Berlin / Heidelberg.

Pedraza, C., Zuiga, J., Suarez, L. J., and Corrales, J. C. (2011). Automatic service retrieval in converged environments based on natural language request. In *SERVICE COMPUTATION 2011*, pages 52–56, Rome, Italy.

Riabov, A. V., Boillet, E., Feblowitz, M. D., Liu, Z., and Ranganathan, A. (2008). Wishful search: interactive composition of data mashups. In *Proceedings of the 17th WWW '08*, pages 775–784, New York, NY, USA.

Szuba, T., Polanski, P., Schab, P., and Wielicki, P. (2011). On efficiency of collective intelligence phenomena. In Nguyen, N. T., editor, *Transactions on computational collective intelligence III*, chapter On efficiency of collective intelligence phenomena, pages 50–73. Springer-Verlag, Berlin, Heidelberg.

Yu, T., Zhang, Y., and Lin, K.-J. (2007). Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. Web*, 1:1–26.