Collaborative mashup development in Enterprise 2.0

Devis Bianchini, Valeria De Antonellis, Michele Melchiori

Università degli Studi di Brescia – Dip. di Ing. dell'Informazione
Via Branze 38
25123 Brescia, Italy

{bianchin|deantone|melchior}@ing.unibs.it

Abstract. Modern organizations support and promote internal collaboration to improve performances of their processes. In this paper, we focus on collaboration in software development processes where the process activities are simplified by discovering and exploiting specific knowledge available inside the organization. Specifically, we consider the mashup application development. Mashup has been recently introduced as a development approach for situational fast-to-implement Web-oriented applications. A mashup integrates software components, called Web APIs that can provide access to complex functionalities and rich data sources. Collaboration can simplify and make more effective the mashup development process, in terms of time and quality, by exploiting the knowledge of the developers operating inside the organization. To this aim, we propose a model as part of the Enterprise 2.0 paradigm that has been recently introduced as specialization of the Web 2.0 concepts and technologies to the enterprise. In the discussed model, a mashup developer is supported in searching for assistance from developers owning specific knowledge, according to typical collaboration patterns.

1 Introduction

Modern organizations support and promote internal collaboration to improve performances of their processes. In this software development, mashup has been recently introduced as a development method inside organizations for situational fast-to-implement applications. Enterprise 2.0 [1] is the specialization of Web 2.0 concepts and technologies to the enterprise with the aim of boosting the collaboration both inside and outside the enterprise. In this context, a number of collaboration platforms is currently appearing [2,3,4] to provide enterprises with tools that allow for social linking and tagging of resources, that support the user feedback/opinions, and user-produced content management platforms (blogs and Wikis). For example, the YAMMER Web platform[1] allows an enterprise to create a private social network implementing collaborative workspaces for project team members and external selected partners. Another recent trend in enterprises is mashup [5] that has been introduced as development approach for quick-to-build applications which are created to

[1] https://www.yammer.com/

satisfy situational short term business needs by combining more Web APIs into a single lightweight Web application.

Mashup design and implementation may leverage on collaboration to discover, understand and integrate Web APIs. In fact, collaboration supported by the Enterprise 2.0 tools can simplify and make more effective the development process, in terms of required time and quality improvement, by exploiting the specialized knowledge of developers operating inside the organization. On the one hand, developers can exploit a large and always growing collection of Web APIs made available inside enterprises by means of searchable catalogs (e.g., IBM Mashup Center Catalog[2]). Beside technical documentation, Web registries provide also information about how APIs are used in mashups and feedback from the user community. On the other hand, mashup development is hindered by the heterogeneity of Web APIs and related documentation.

Generally, the steps in mashup development that can be supported by collaboration are: i) searching and identifying the most suitable API functionalities to build a new application; ii) understanding functional and nonfunctional features of the selected APIs with the purpose to compose them according to the requirements. Accordingly, research has been done to define search tools based on semantic/functional/non functional characterization of APIs [6], on their social characterization/tagging [7,8], on past use/collective knowledge of APIs [9], on techniques that mix social and functional features [10].

The proposal described in this paper focuses on collaborative development in the Enterprise 2.0 contexts. To this purpose we propose a model that includes: i) description of Web APIs as typically found on sharing Web sites, ii) information about developers based on a specialization of FOAF ontology and, iii) relationships among developers and Web APIs.

Specifically, the cited search tools focus on recommending Web APIs to the developer. Our proposal is complementary to these approaches and comes as a successive phase by recommending to the developer those colleagues inside the enterprise that have specific knowledge about the Web APIs she/he has selected. To this purpose, we have identified typical collaboration patterns: i) search for collaboration on the use of specific Web APIs; ii) search for collaboration on specific Web API technologies; iii) search for competencies in developing specific types of mashups.

The paper is organized in the following way. A simple example to illustrate the considered scenario is given in the following of the Sect. 1. In the Sect. 2, we introduce the model for mashup development based on collaboration. The collaboration patterns that exploit this model are defined in the Sect. 3. A different way to make explicit and representing the skills of the enterprise developers according to different perspectives is discussed in the Sect. 4. Conclusions and some possible extensions of the work presented in this paper are given in the Sect. 5.

---

[2] http://www-10.lotus.com/ldd/mashupswiki.nsf/dx/Introduction__Mashup_Center_2.0.0.2

1.1 A motivating example

Let us consider an enterprise with various branches and departments in which operate expert users and developers that need sometime to implement situational applications. For example, consider also a webmaster working for the marketing department of this enterprise that has to build an application to visualize on an interactive map, information about the customers, about sales and demographic data. The webmaster finds the APIs (developed internally, as part of the ERP, and published on the Web) that she/he needs implementing the single functional blocks. However, she/he can face difficulties in: i) using a specific API, because of a not clear API semantics or because not familiar with the I/O parameters data format; ii) integrating the selected APIs, because of heterogeneity of documentation and used languages/technologies. In fact, in spite the availability of commercial and research tools the task of integrating Web APIs requires yet specialized skills.

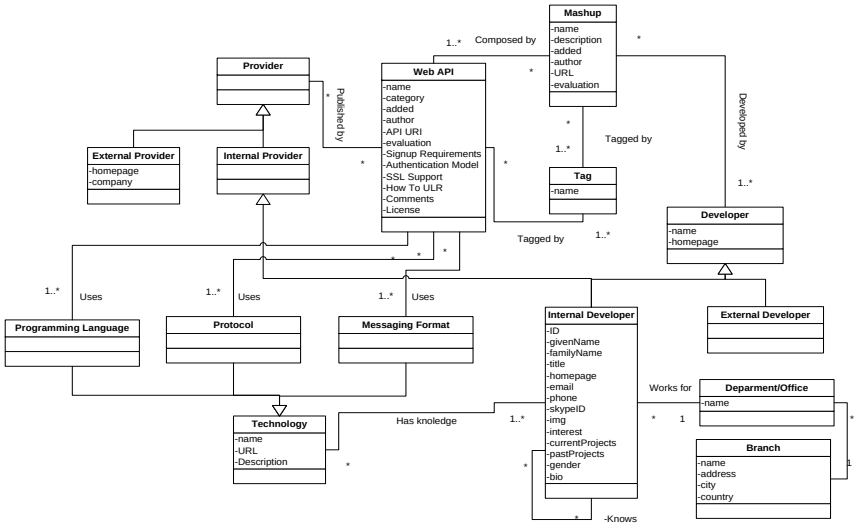

Fig. 1. UML model for supporting mashup development collaboration

2 A model for collaborative mashup development

In this section we propose a model for collaborative developing in enterprises including social and usage contexts of Web APIs. Without loss of generality, we can assume that this model maintains at least:

- Web APIs descriptions at the same level of detail of public registries like ProgrammableWeb

- Social information about developers based on a specialization of FOAF[3].

ProgrammableWeb has been chosen because it is the most complete public registry of Web APIs and mashups. The choice of FOAF is due to the fact that it is a well known and standard proposal of ontology for conceptualization of people and social relationships. The model, shown in Fig. 1, includes that information that satisfies the requirements for the collaboration scenario considered in the following section. Note that the Person class of FOAF has been here specialized in the class Internal Developer. An Internal Developer can have knowledge about technologies used by Web API. The classes External Provider and External Developer allow keeping supplementary information about providers and developer of public Web APIs and mashups.

In this model we focus specifically on the classes Internal Developer, Mashup, Web API. We associate each object of these classes with a set of descriptors. A descriptor $Des_p(o_i)$ for an object o_i according to a perspective p is defined as:

$$Des_p(o_i) = \{t_{pi}\} \quad (1)$$

where t_{pi} are terms extracted from the object descriptions, specifically from class attribute values and relationships involving the class of o_i. In Table 1 is shown the applicability of perspectives (columns) to each considered class (rows).

	Perspectives					
	Organizational	**Web API**	**Mashup**	**Technologies**	**Developer**	**SocialConnectivity**
Developer	x	x	x	x		X
Web API			x	x	x	
Mashup		x			x	

Table 1. Applicability of perspectives

A developer is described by an organizational perspective (office/department, current projects, past mashup projects), the Web APIs she/he has used/developed, the developed mashups, the technologies on which she/he has expertise and her/his social connections (developers that she/he knows, developers belonging to the same office, that have worked on the same project or on the same mashup). A Web API is described by the mashups in which it has been used, the technologies (programming languages, protocols, messaging formats), the developers that used or provided it. Finally, a mashup is specified by the Web APIs that it includes and the developer/s that developed it.

3 Collaboration patterns

Descriptors associated with developers, mashups and Web APIs enables the definition of collaboration patterns. A collaboration pattern is identified by a type of re-

[3] http://www.foaf-project.org/

quest. We define in a general way a request R for collaboration submitted by a developer D_R as follows.

$$R=< T_R, D_R, M_R, W_R> \quad (2)$$

where T_R is the type of request, $M_R=\{W^{Ri}\}$ is a set of Web APIs W^{Ri}, possibly empty, selected for implementing a mashup. For simplicity we can refer to this set as the mashup M_R. W_R is a Web API optionally specified.

With reference to the motivating example above described, we distinguish three different collaboration scenarios in which D_R can be involved.

1. Search for collaboration on the specific Web API W_R: D_R is looking for collaboration from developers that have experience with a specific Web API W_R she/he has chosen and needs to use in the mashup.

2. Search for collaboration on the technology used by WR: D_R is looking for developers that have experience with specific technologies of the Web API W_R. For example, this case is invoked, as second option, if the result of application of the previous case *Search for collaboration on a specific Web API* is empty. As a consequence, the developer shifts the request to collaboration on the W_R technologies.

3. Search for collaboration on the mashup MR: D_R is looking for collaboration with developers that have experience in mashups built with the same set of Web APIs of M_R or at least with a subset of it.

3.1 Definition of collaboration patterns

The illustrated scenarios are implemented by collaboration patterns that provide the functionalities to satisfy the request expressed in each scenario. Formally, a *collaboration pattern* is defined as a 4-uple:

$$CP_л =< R_л, m_л, \delta_л , \angle_л, > \quad (3)$$

where л is the goal of the collaboration pattern. The output of the application of a collaboration pattern is to suggest a ranked list of developers satisfying the request $R_л$. The metrics $m_л$ are used to evaluate the degree of matching between a developer and the request R, on the basis of the specified goal. The threshold $\delta_л$ is used to filter out developers with low relevance with respect to $R_л$. A developer D_j is proposed to the mashup designer if $m_л(R_л,D_j) > \delta_л$. Finally, $\angle_л$ is a ranking function to present the developers relevant for $R_л$. In particular, $D_i \angle_л D_j$, that is D_i precedes D_j in the ranking, if $m_л(R_л,D_j) \geq m_л(R_л,D_j)$. Collaboration pattern metrics $m_л$ are based on a notion of similarity between descriptors. The similarity between pairs of descriptors is defined according to the classical Dice's formula for similarity over sets.

Table 2 reports how the elements of generic collaboration pattern are defined to fulfill the three considered scenarios. Specifically, each scenario is associated with a goal and specifies a kind of request, a metrics and the perspectives considered in the evaluation of the collaboration.

In the following, for each pattern we provide the motivations for the metrics defined in the table.

Scenario T_R	Request $R_{Л}$	Metrics $m_{Л}$	Perspectives
1. Search for collaboration on a specific Web API W_R	R= <wApi, D_R, ∅, W_R >	$m(R,Dj) =$ $\begin{cases} 0 & \text{if } W_R \neq Des_{p_1}(D_j) \\ \alpha \cdot Sim\left(Des_{p_2}(D_R), Des_{p_2}(D_j)\right) + & \\ \beta \cdot Sim\left(Des_{p_3}(D_R), Des_{p_3}(D_j)\right) & \text{otherwise} \end{cases}$ with $\alpha + \beta \in [0..1]$ and $\alpha,\beta \geq 0$	P1='Web API', P2='Organizational', P3='SocialConnectivity'
2. Search for collaboration on the technology used by W_R	R= <tech, D_R, ∅, W_R >	$m(R,Dj) =$ $\begin{cases} 0 & \text{if } Sim\left(Des_{p_1}(D_j), Des_{p_1}(W_R)\right) = 0 \\ \alpha \cdot Sim\left(Des_{p_1}(D_j), Des_{p_1}(W_R)\right) + & \\ \beta \cdot Sim\left(Des_{p_2}(D_R), Des_{p_2}(D_j)\right) + & \text{otherwise} \\ \gamma \cdot Sim\left(Des_{p_3}(D_R), Des_{p_3}(D_j)\right) & \end{cases}$ with $\alpha + \beta + \gamma \in [0..1]$ and $\alpha,\beta,\gamma \geq 0$	P1='Technologies', P2='Organizational', P3='SocialConnectivity'
3. Search for collaboration on the mashup M_R	R= <mash, D_R, M_R >	$m(R,Dj) =$ $\begin{cases} 0 & \text{if } Sim\left(Des_{p_1}(D_j), M_R\right) = 0 \\ \alpha \cdot Sim\left(Des_{p_1}(D_j), M_R\right) + & \\ \beta \cdot Sim\left(Des_{p_2}(D_R), Des_{p_2}(D_j)\right) + & \text{otherwise} \\ \gamma \cdot Sim\left(Des_{p_3}(D_R), Des_{p_3}(D_j)\right) & \end{cases}$ with $\alpha + \beta + \gamma \in [0..1]$ and $\alpha,\beta,\gamma \geq 0$	P1='Web API', P2='Organizational', P3='SocialConnectivity'

Table 2. Definition of collaboration patterns for the scenarios

- **Pattern 1.** If the developer D_j has experience with W_R then her/his evaluation is not zero and is based on a weighted sum of two terms: i) similarity of D_j and D_R with respect the Organizational perspective, ii) similarity of D_j and D_R with respect to the SocialConnectivity perspective. So, developers that have more social/organizational connections with D_R receive a higher degree because it is supposed to be easier to contact and involve them.
- **Pattern 2.** if the developer D_j has experience with the technologies of W_R then her/his evaluation is not zero and is based on a weighted sum of three terms: i) similarity between technologies on which D_j has expertise and the technologies used by W_R, ii) similarity of D_j and D_R with respect the Organizational perspective, iii) similarity of D_j and D_R with respect to the SocialConnectivity perspective.
- **Pattern 3.** In this case, $Des_{P1}(D_j)$ includes the Web APIs used in the mashups developed by D_j. That is, if the developer D_j has used (at least one of) the Web APIs in M_R then her/his evaluation is not zero and is based on a weighted sum of three terms: i) her/his similarity with M_R with respect the Web API perspective, that is high similarity if D_j has developed mashups using the APIs in M_R, ii) similarity of D_j and D_R with respect the Organizational perspective, iii) similarity of D_j and D_R with respect to the SocialConnectivity perspective. Note that $Des_{P1}(D_j)$ and M_R are both sets of Web APIs so it is meaningful to evaluate their similarity.

For the metrics m_μ of each of these patterns, the parameters α,β,γ are initially set to the same value. During a training phase with the involvement of developers, the values are iteratively adjusted to obtain desired rankings.

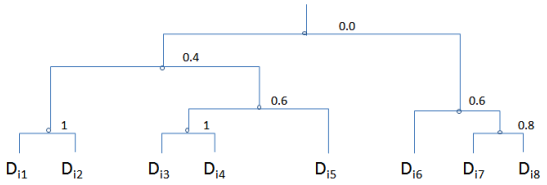

Fig. 2a. Example of Similarity tree

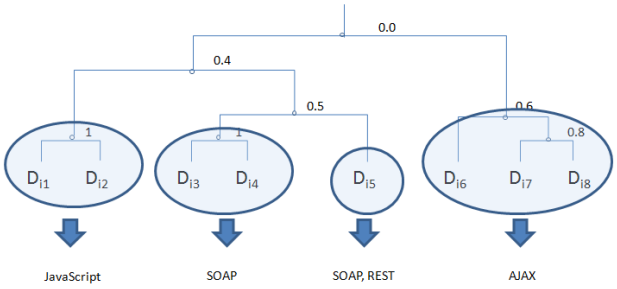

Fig. 2b. Clustering of developers by perspective 'Technology'

4 Multi-perspective organization and clustering of developers

Developer descriptions can also be organized according to the different perspectives introduced in Sect. 2. In fact, the information provided by the model in Fig. 1 and the similarity between developers descriptors can be used jointly to build representations of the enterprise developers according to the different perspectives. These organizations can be browsed by a developer to know about the available skills (technologies, Web APIs, mashups) of the other developers inside the enterprise.

In particular, by mean of a hierarchical agglomerative clustering procedure, the developers can be partitioned in groups, where each group includes the developers owning similar features according to a considered perspective. The resulting representation is a mapping *group-to-skills*.

In order to illustrate the main steps required, let us consider specifically the perspective 'Technologies'. The hierarchical clustering procedure is applied to the set of the developers. In the procedure, the similarity metrics for each pair of developer descriptors $Des_P(D_{ij})$ and $Des_P(D_{ik})$, the value of $Sim(Des_P(D_{ij}), Des_P(D_{ik}))$ normalized in the range [0,1]. The output of the procedure is a similarity tree, as the one illustrated in Fig. 2a, where the leaves represent the descriptors and the arcs connect pairs of clusters recognized as the most similar at a specific step. Setting a given minimum similarity threshold defines the clusters in the similarity tree. For example, the threshold is set to 0.6 in the Fig. 2b. The skills according to the perspective 'Technology' for each group of developer cluster are shown as the intersection of the $Des_P(D_{ik}) = \{t_{pik}\}$ for each D_{ik} in the cluster. Note, that the $\{t_{pik}\}$ are values available in the object descriptions according to the collaboration model.

5 Conclusions

In this paper, we have introduced a collaboration model for mashup development in the context of Enterprise 2.0. Specifically, the model permits to define collaboration patterns to support automatically the mashup developer in the task of discovering potential collaboration with other developers. Future work includes the development of a software prototype to be integrated in an Enterprise 2.0 platform and extension of the model by considering additional information, like tags, Web APIs categories and feedback on the collaboration. Moreover, the model has to be enriched with mechanisms for ranking and evaluating the skills and the experience of developers.

6 References

1. A. P. Mcafee, "Enterprise 2.0: The Dawn of Emergent Collaboration" *MIT Sloan Management Review*, vol. 47, no. 3, 2006.
2. IBM, "IBM Connections." http://www-01.ibm.com/software/lotus/products/connections.
3. TELLIGENT, "TELLIGENT Enterprise." http://telligent.com/social-enterprise-software.

4. J. Soriano, D. Lizcano, M. A. Cañas, M. Reyes, and J. J. Hierro, “Fostering Innovation in a Mashup-oriented Enterprise 2.0 Collaboration Environment,” *System and Information Sciences Notes*, no. 1, pp. 62–68, 2007.
5. V. Hoyer and M. Fischer, “Market overview of enterprise mashup tools”. In *Proc. of the 6th International Conference on Service Oriented Computing (ICSOC’08)*, 2008, pp. 708-721.
6. Gomadam, K., Ranabahu, A., Nagarajan, M., Sheth, A. P. and Verma, K. (2008) A Faceted Classification Based Approach to Search and Rank Web APIs,*Proc. of 6th IEEE Int. Conference on Web Services (ICWS'08).*
7. K. Goarany, G. Kulczycki, and M. B. Blake, “Mining Social Tags to Predict Mashup Patterns,” *Proceedings of the 2nd International Workshop on Search and Mining User-generated Contents (SMUC 2010), October 30, 2010, Toronto. Canada. Copyright 2010 ACM.*, pp. 71-77, 2010.
8. D. Bianchini, V. De Antonellis, and M. Melchiori, “Semantic Collaborative Tagging for Web APIs Sharing and Reuse,” in *Proceedings of the 12th Int. Conference on Web Engineering (ICWE’12)*, 2012, vol. LNCS, no. 7387, pp. 76-90.
9. O. Greenshpan, T. Milo, and N. Polyzotis, “Autocompletion for Mashups,” in *Proc. of the 35th Int. Conference on Very Large DataBases (VLDB’09)*, 2009, pp. 538-549.
10. B. Tapia, R. Torres, and H. Astudillo. "Simplifying mashup component selection with a combined similarity- and social-based technique". In *Proceedings of the 5th International Workshop on Web APIs and Service Mashups (Mashups '11)*, Agnes Koschmider, Erik Wilde, and Christian Zirpins (Eds.). ACM, New York, NY, USA, 2011.