Bridging Socially-Enhanced Virtual Communities

Daniel Schall, Florian Skopik, Harald Psaier, Schahram Dustdar
Distributed Systems Group
Vienna University of Technology
{lastname}@infosys.tuwien.ac.at

ABSTRACT

Interactions spanning multiple organizations have become an important aspect in today's collaboration landscape. Organizations create alliances to fulfill strategic objectives. The dynamic nature of collaborations increasingly demands for automated techniques and algorithms to support the creation of such alliances. Our approach bases on the recommendation of potential alliances by discovery of currently relevant competence sources and the support of semi-automatic formation. The environment is service-oriented comprising humans and software services with distinct capabilities. To mediate between previously separated groups and organizations, we introduce the *broker* concept that bridges disconnected networks. We present a dynamic broker discovery approach based on interaction mining techniques and trust metrics. We evaluate our approach by using simulations in real Web services' testbeds.

Categories and Subject Descriptors

H.2.3 [**Languages**]: Query languages; H.3.4 [**Systems and Software**]: Information networks; H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Design, Experimentation, Human Factors, Languages

Keywords

Broker discovery, Social networks, Interaction mining

1. INTRODUCTION

The rapid advancement of ICT-enabled infrastructure has fundamentally changed how businesses and companies operate. Global markets and the requirement for rapid innovation demand for alliances between individual companies [5]. Web services and service-oriented computing offer well established standards and techniques to model and implement interactions spanning multiple organizations. Collaborative service-based systems are typically knowledge intensive covering complex interactions between *people* and *software services*. In such ecosystems, flexible interactions commonly take place in different organizational units. The challenge is that top-down composition models are difficult to apply in constantly changing and evolving service-oriented collaboration system. There are two major obstacles hampering the establishment of seamless communications and collaborations across organizational boundaries: (i) the dynamic discovery and composition of resources and services, and (ii) flexible and context-aware interactions between people residing in different departments and companies.

Theories found in social network analysis are promising candidate techniques to assist in the formation process and to support flexible and evolving interaction patterns in cross-organizational environments. In social networks, relations and interactions typically emerge freely and independently without restricted paths and boundaries. Research in social sciences has shown that the resulting social network structures allow for relatively short paths of information propagation (the *small-world phenomenon*, e.g., see [11]). While this is true for autonomously forming social networks, the boundaries of collaborative networks are typically restricted due to organizational units and fragmented areas of expertise. We propose social network principles to bridge segregated collaborative networks. The theory of structural holes is based on the idea that individuals can benefit from serving as intermediaries between others who are not directly connected [4]. Thus, such intermediaries can potentially *broker* information and aggregate ideas arising in different parts of a network [12].

In this work, we present the following key contributions:

- We introduce *brokers* to establish connections between independent subgroups in professional virtual communities (PVCs). Our approach enables the dynamic selection of brokers based on changing interest profiles.
- We define metrics and their application to support the discovery and selection of brokers including *social trust* in service-oriented collaborations.
- Our approach is to introduce the *Broker Query and Discovery Language* (BQDL) to discover suitable brokers based on query preferences (discovery policies). The novelty of BQDL is the ability to query social network data considering information obtained from mining results to fulfill the requirements for broker discovery in PVCs.

This paper is structured as follows. In Section 2, we overview related work to provide the background for our approach. In Section 3, we present a motivating scenario for the discovery of brokers and introduce broker behavior patterns. In Section 4, we introduce supporting concepts to realize flexible interactions and the selection of brokers. We define BQDL in Section 5 followed by a discussion on the implementation and evaluation in Section 6. Finally, we conclude the paper in Section 7.

2. BACKGROUND AND RELATED WORK

In service-oriented environments, standards have been established to model human-based process activities and tasks (WS-HumanTask [7]). However, these standards demand for the precise definition of interaction models between humans and services. In our approach, we combine SOA concepts and social principles. We consider **open service-oriented environments** wherein services can be added at any point in time. Following the open world assumption, humans actively shape the availability of services. We adopt the concept of Human-Provided Services (HPS) [20] to support flexible service-oriented collaborations across multiple organizations and domains. Similarly, emergent collectives as defined by [17] are networks of interlinked valued nodes (services). Open service-oriented systems are specifically relevant for future crowdsourcing applications. While existing platforms (e.g., Amazon's Mechanical Turk[1]) only support simple interaction models (tasks are assigned to individuals), social network principles support more advanced techniques such as formation and adaptive coordination.

We focus on **strategic formation** in social networks and communities [22]. The theory of structural holes was developed by Burt [4] and is based on the hypothesis that individuals can benefit from serving as intermediaries between others who are not directly connected. A formal approach to strategic formation based on advanced game-theoretic broker incentive techniques was presented in [12]. Our approach is based on interaction *mining and metrics* to dynamically discover brokers suitable for connecting communities in service-oriented collaborations. The availability of rich and plentiful data on human interactions in social networks has closed an important loop [11], allowing one to model social phenomena and to use these models in the design of new computing applications such as crowdsourcing techniques [2]. A wide range of computational trust models have been proposed [1, 16]. We focus on social trust [8, 21, 24] that relies on user interests and collaboration behavior.

Technically, the focus of BQDL is to provide an intuitive mechanism for **querying data from social networks**. These networks are established upon *mining and metrics*. Thereby, properties of such networks are under constant flux and changes. BQDL is not a generic graph query language such as SPARQL [23], which has been designed to query ontological data. Instead, BQDL addresses the specific requirements for the discovery of actors such as brokers by accounting for (weighted) paths and metrics obtained from mining results. In [18], a query language for social networks was presented. The language in [18] has some similarities with BQDL (e.g., path functions), however, without supporting the discovery of complex sub communities based on metrics and interaction mining techniques.

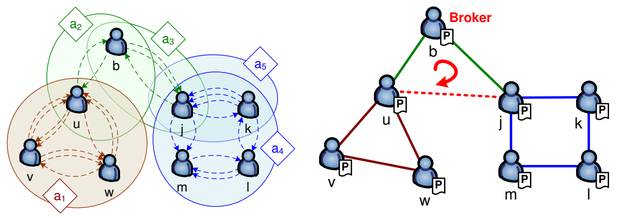
[1]Amazon MTurk: `http://www.mturk.com`

3. EMERGING VIRTUAL COMMUNITIES

A PVC is a virtual community [5] that consists of experts who interact and collaborate supported by ICT to perform their work. In today's systems, service-oriented technologies are increasingly used to realize PVCs. The support of loose coupling, sophisticated discovery, dynamic binding and various composition mechanisms make SOA the ideal technical grounding for Web-enabled PVCs.

3.1 Collaboration Scenario

Let us discuss an actual collaboration scenario in PVCs as depicted in Figure 1. Various member groups collaborate in the context of five different activities a_1, a_2, a_3, a_4 and a_5 (see Figure 1(a)). These groups intersect since members may participate in different activities at the same time. The color of the activity context determines the expertise areas an activity is related to. Such activities are, for instance, the creation of new specifications or the discussion of future technology standards. Activities (e.g., see [15]) are a concept to structure information in flexible collaboration environments, including the goal of ongoing tasks, involved actors, and utilized resources such as documents or services. They are either assigned from the outside of a community, e.g., belonging to a higher-level process, or emerge by identifying collaboration opportunities. PVC members use SOA technologies to interact in the context of ongoing activities. The HPS Framework [20] allows human participation in a service-oriented manner. Humans can provide their capabilities and expertise as *services* to enable human interactions using standardized messages (i.e., SOAP). Interactions are logged for analysis. Relations emerge from interactions as illustrated in Figure 1(b), and are bound to particular scopes (*expertise areas*). The context in which interactions take place is based on tags applied to various artifacts exchanged between collaboration partners. Tags are used to combine similar activities to create scopes (i.e., boundaries of activities). In the given scenario, a scope comprises relations between PVC members regarding help and support activities in different expertise areas (reflected by tags of exchanged messages). Scopes are used for different purposes. First, by analyzing the interaction context (i.e., using message tags), we determine users' centers of interest. Frequently used keywords are stored in the actors' profiles (see symbol P) and later used to determine their interests and expertise areas. Second, we aggregate interactions that occurred in a predefined scope, calculate metrics (numerical values describing prior interaction behavior), and interpret them as *social trust* that is based on reliability, dependability and success.



(a) Member interactions. (b) Emerging relations.

Figure 1: Collaboration model for service-oriented PVCs: (a) interactions between PVC members are performed in the context of activities; (b) social relations and profile areas emerge based on interactions.

3.2 Brokering and Compositions

Consider a scenario in the given PVC in Figure 1(b). Suppose u wants to set up an activity that requires at least one additional expert from the *brown* $\{u, v, w\}$ and *blue* domain $\{j, k, l, m\}$. Since u personally knows v and w from previous collaborations, which is reflected by Friend-of-a-Friend (FOAF) [3] `knows` relations, u is well-connected to the *brown* expertise area. However, u does not know any member from the *blue* domain. The broker concept helps to solve this problem. Actor u collaborated with b in the *green* domain, who is connected to j. Therefore, b could potentially act as a broker and forward requests or invitations to join u's current activity to j. We argue that establishing personal contacts in socially-oriented environments is of high importance compared to the traditional SOA domain, where services are mostly composed based on their properties (i.e., features and QoS) only.

Assuming one is able to infer meaningful social relations between network members, such relations have major impact on future collaborations in different scenarios: (i) *Supporting the Formation of Expert Groups.* Successfully performed compositions of actors should not be dissolved but actively facilitated for future collaborations. Thus, tight trust relations can be dynamically converted to FOAF relations (i.e., **discovery of relevant social networks**). (ii) *Controlling Interactions and Delegations.* Discovery and interactions between members can be based on FOAF relations. People tend to favor requests from well-known members compared to unknown parties. (iii) *Establishment of new Social Relations.* The emergence of new personal relations is actively facilitated through brokers. The introduction of new partners through brokers (e.g., b introduces u and j to each other) leads to future trustworthy compositions.

4. SOCIALLY-ENHANCED SOA

We adopt various concepts to realize the before mentioned collaboration communities, and consider various mechanisms to enable brokering of requests, including flexible service-oriented collaboration models and the automatic management of social trust relations.

4.1 Flexible Service-Oriented Collaboration

Web services play a fundamental role in supporting flexible, cross-enterprise collaboration scenarios. We discuss human interactions in SOA as introduced in our previous work (see HPS approach [20]). HPS enhances the traditional 'SOA-triangle' approach by enabling people to provide services using the very same technology as implementations of software-based services (SBS) use. By following the SOA paradigm, three essential steps are performed:

(1) *Publish.* Users have the ability to create HPSs and publish the services on the Web using a registry. Publishing a service is as simple as posting a blog entry on the Web. It is the association of the user's profile with an activity described as a service (WSDL). Interfaces provide the needed metadata support for the discovery of suitable HPSs. (2) *Search.* The service requester performs a keyword-based search (reflecting expertise areas) to find Human-Provided or Software-Based Services. Ranking is performed to find the most relevant HPS based on, for example, the expertise of the user providing the service. Expertise is determined automatically by the HPS framework through context-sensitive interaction mining techniques [19]. (3) *Interact.* The framework supports automatic user interface generation using XML-Forms technology[2]. Thus, personalized interaction interfaces can be generated and rendered for different devices. The HPS framework can be used for interactions between humans and also for interactions between SBS and HPSs.

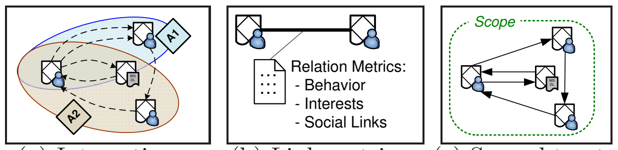
4.2 Emergence and Evolution of Trust

In contrast to a widely used security perspective on trust, we define *social trust* relying on the interpretation of previous collaboration behavior and also considering the similarity of dynamically changing interests [8, 21]. Especially in collaborative environments where users are exposed to higher risks as compared to common social network scenarios [6] and business is at stake, considering social trust is essential to effectively guide interactions [14]. Here, we define trust as follows [9, 16, 21]: *Trust reflects the expectation one actor has about another's future behavior to perform given activities dependably, securely, and reliably based on experiences collected from previous interactions.*

The fundamental approach to automatic interaction-based trust inference is depicted in Figure 2.

Interactions and Monitoring. As motivated in the introduced use case, people interact to perform their tasks. Work is modeled as activities, that describe the type and goal of work, temporal constraints, and used resources. As interactions take place in the context of activities (Figure 2(a)), they can be categorized and weighted. SOAP is the standard message format to support interactions between distributed software services. Also human interactions can be supported in a service-oriented manner using technologies such as SOAP (see HPS [20]). This technology including extensions such addressing and correlation mechanisms is state-of-the-art in service-oriented environments and well supported by a wide variety of software frameworks. This fact enables the adoption of various monitoring and logging tools to observe interactions in service-oriented systems.

Link Metrics. Interaction logs are used to infer metrics that describe the relation of single actors (Figure 2(b)). Various metrics can be calculated by analyzing interaction logs such as behavior in terms of availability and reciprocity. A simple example of a metric is the *success rate* of delegated tasks between two members (successfully processed tasks divided by the total number of delegated tasks). Relation metrics describe the links between actors by accounting for (i) recent interaction behavior, (ii) profile similarities (e.g., interest or skill similarities), (iii) social and/or hierarchical



(a) Interactions. (b) Link metrics. (c) Scoped trust.

Figure 2: Trust emerging from interactions: (a) interaction patterns shape the behavior of actors in context of activities; (b) (semi-)automatic rewarding of behavior and calculation of interaction metrics; (c) inference in scopes by interpretation of metrics.

[2]XML Forms: `http://www.w3.org/MarkUp/Forms/`

structures (e.g., role models). However, we argue that social trust relations largely depend on personal interactions. We model a community of actors with their social relations as a directed graph, where the nodes denote network members, and edges reflect (social) relations between them. Since interaction behavior is usually not symmetric, actor relations are represented by *directed links*.

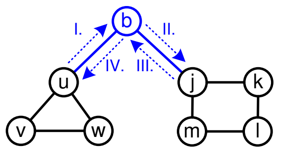
Scoped Trust. Our approach considers the diversity of trust by enabling the flexible aggregation of various interaction metrics (e.g., *success rate* and *responsiveness*) that are determined by observing ongoing collaborations. Finally, available relation metrics are weighted, interpreted, and composed by a rule engine (the detailed mechanisms can be found in [21]). The result (i.e., a linguistic representation such as *high*, *medium*, or *low*) describes trust between the actors with respect to scopes (Figure 2(c)). For instance, trust relations in a scope 'scientific dissemination' could be interpreted from interaction behavior of actors in a set of paper writing activities.

4.3 Broker Behavior Patterns

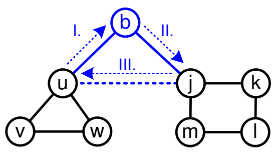
Brokers differ from other actors by their mediation capabilities. A broker acts as an intermediary node between two previously separated communities or collaboration teams. Thus, it is essential that it monitors frequently demanded contacts, updates and maintains its relations to increase and strengthen its popularity, and consequently, trust. If demand decreases, the broker must find and establish new relations. The discussed way to solve the problem is to provide the possibility of querying the social network for new contacts of interest. Of interest are, e.g., contacts to communities with high trust relations among the members and a distinct expertise.

In this work, we define different types of brokers. Considering HPS-based interactions such as delegations of online help and support requests, brokers may exhibit different behavior patterns as illustrated by Figure 3: (a) **Persistent Exogenous Interaction Pattern.** Any request and response is forwarded by the broker, thereby shielding the actually interacting nodes from each other. Thus, each network segment remains separated for the entire duration of a collaboration. (b) **Triadic Exogenous Interaction Pattern.** The broker encourages receivers of requests to establish direct connections to the initiator, and therefore, actively facilitates the emergence of new social relations.

We argue that both types of interaction patterns are applied in today's social and collaborative environments. A broker may favor one pattern over the other due to various reasons. For example, controlling the flow of interactions between personally unknown actors can strengthen a broker's reputation [12]. Establishing direct relations can significantly reduce a broker's workload. Another possible explanation for varying broker behavior patterns may be the similarity of expertise profiles.



(a) Persistent pattern.



(b) Triadic pattern.

Figure 3: Exogenous broker behavior patterns.

For example, if a broker connects similar actors, it may apply the triadic pattern to support the establishment of new social relations. However, if actor profiles diverge significantly, the broker may need to mediate interactions persistently; for example, due to the lack of a common vocabulary or understanding between communities. The proposed query language (BQDL) supports both cases. However, the discussions in the following sections mainly demonstrate the application of BQDL for persistent exogenous broker behavior patterns without detailing the peculiarities of advanced triadic patterns.

5. BQDL SPECIFICATIONS

Here we define the key elements of BQDL. Table 1 lists important language elements to query interaction graphs. The language is inspired by an SQL-like syntax. It is important to note that BQDL operates on a graph defined as $G = (N, E)$ composed of a set of nodes N and edges E.

Element	Description
`satisfy`	Requires that a given condition is fulfilled by a set of nodes or edges.
`as`	Creates an alias for groupings of nodes, edges, or paths.
`<all>`	Retains all nodes/edges/subgraphs satisfying a given condition.
`[ ]`	An expression to satisfy conditions for exactly one `[1]`, one to m `[1..m]`, or one to many `[1..*]` nodes or edges.

Table 1: Important BQDL language elements.

A `Select` statement retrieves nodes and edges in G as well as aggregates of graph properties (for example, properties of a set of nodes). While traditional relational databases operate on tables, BQDL uses the `From` clause to perform queries on a graph G. A `Where` clause specifies filters and policies upon nodes, edges, and paths. To give intuitive examples, we present a set of BQDL queries along with their meaning considering a graph G and a set of subgraphs $G' \subseteq G$. We structure discussions related to a BQDL query into four essential steps: **R** the basic requirements/goal of a query, **A** the approach that is taken, **O** the output of the query, **D** the detailed description of the query.

5.1 Connecting Predefined Communities

As a first simple example in Figure 4, consider two initially disconnected communities (sets of nodes) depicted as variables `var source` $= \{n_1, n_2, \ldots, n_i\}$ and `var target` $= \{n_j, n_{j+1}, \ldots, n_{j+m}\}$ residing in the graph G. **R1:** The goal is to find a broker connecting disjoint sets of nodes (i.e., not having any direct links between each other). **A1:** Two subgraphs `G1` and `G2` are created to determine brokers which connect the source community $\{u, v, w\}$ with the target community $\{g, h, i\}$ (i.e., see `From` construct). **O1:** The output of the query is (the example shown in Figure 4) a list of brokers connecting $\{u, v, w\}$ and $\{g, h, i\}$. The lines 1-3 specify the input/output parameters of the query. **D1:** As a first step, a (sub)select is performed using the statement as shown by the lines 6-11. The statement `distinct(node)` means that a set of unique brokers shall be selected based on the condition denoted as the `Where` clause with a filter (lines 9-10). The term '`[1..*] n in source`', where `source`

```
1  Input: Graph G, var source = {n_1, n_2, ..., n_i},
2         var target = {n_j, n_{j+1}, ..., n_{j+m}}
3  Output: List of brokers
4
5  Select node From (
6      ( Select distinct(node) From G
7        Where
8          /* At least one in source 'knows' node */
9          ( [1..*] n in source ) satisfy
10           Path (n to node) as P1 With P1.length = 1 )
11                 as G1,
12     ( target ) as G2
13   )
14   Where
15     /* Retain all nodes that satisfy path filter */
16     ( <all> n in G1.nodes ) satisfy
17       /* Path to any in G2.nodes */
18       Path (n to [1..*] G2.nodes) as P2
19             With P2.length = 1
20     and
21     /* Retain all edges that satisfy edge filter */
22     ( <all> e in G1.edges ) satisfy
23       (e.relation = EPredicates.BIDIRECTIONAL) and
24       (e.trust >= MTrust.MEDIUM)
25
26 Order by node
```

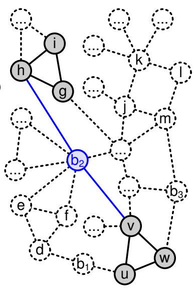


Figure 4: BQDL example 1: find broker to connect two predefined communities.

is the set of nodes passed to the query as input argument, means that at least one node $n \in G$ must satisfy the subsequent condition. Here the condition is that the node n has a link (i.e., through `knows` relations) to the source set of nodes. This is accomplished by using the `Path` function that checks whether a link between two nodes exists (the argument '`(n to node)`'). The path alias is used to specify additional constraints such as the maximum path length between nodes (here '`P1 With P1.length = 1`'). The second step is to create an alias `G2` for the target community $\{g, h, i\}$. By using the aliases `G1` (line 11) and `G2` (line 12) further filtering can be performed using the `Where` clause in line 14. The same syntax is used as previously in the subselect statement (lines 9-10). The construct `<all>` retains nodes '`n in G1.nodes`' (`G1` holding the set of candidate brokers) that are connected to at least one node in the target community `G2` with direct links ('`P2 with P2.length = 1`'). Further filtering is performed by defining lines 22-24.

Here, brokers in `G1` and both the source $\{u, v, w\}$ the target community $\{g, h, i\}$ must have edges between each other that are bidirectional. In our graph representation, this means that each relation has to be interpreted as, for example, b_2 `knows` h and h `knows` b_2. A set of different metrics is established in our system. A specific type of metric (e.g., trust) is denoted by the namespace `MTrust`. In the specified query, each actor in the result set must share a minimum level of trust depicted as '`e.trust >= MTrust.MEDIUM`'. Trust metrics are associated to edges between actors. The term `MTrust.MEDIUM` is established based on mining data to obtain linguistic representations by mapping discrete values (metrics) into meaningful intervals of trust levels. The last statement '`Order by node`' in Figure 4 implies a ranking procedure of brokers. This can be accomplished by using eigenvector methods in social networks such as the PageRank algorithm to establish authority scores (the importance or social standing of a node in the network) or advanced game-theoretic techniques based on the concept of structural holes (see for example [12]). The detailed mechanisms of this procedure are not the focus of this work.

5.2 Finding Communities

The broker discovery example in the previous section (Figure 4) is straightforward because the target community is already specified and passed to the query as `var target` = $\{n_j, n_{j+1}, \ldots, n_{j+m}\}$. The next example query eliminates this assumption by showing an approach to find suitable communities based on search criteria (e.g., activity or skill tags). **R2:** The goal of the query as specified in Figure 5 is to find sub-communities (or subgraphs) in G that match search criteria. **A2:** Search is performed by using a set of distinct tags specified as input parameter `var search` = $\{t_1, t_2, \ldots, t_n\}$. **O2:** The output of the query is a list of communities. **D2:** The first step is to perform a (sub)select of distinct communities (see `distinct(nodes) as G'` in line 5) to obtain non-overlapping groups of community members specified by the lines 5-14. For example, Figure 5 shows four groups of nodes $[\{d, e, f\}, \{g, h, i\}, \{l, m, j, k\}, \{u, v, w\}]$ each of them satisfying the constraints specified in the query. Each node in a specific community must be linked to at least one community member so that '`Path (n to [1..*] G'.nodes) as P1`'. Also, at least one path between nodes with '`length = 1`' satisfying trust requirements (trust level `MTrust.HIGH`) must exist in order to consider a node as a community member. Finally, a path must contain the tags specified by the search query (lines 11-12) to ensure that a member has interacted (collaborated) with other members in the context of certain activities. The alias `SG1` provides access to each community. The `Where` clause applies filtering of communities based on load conditions measured by graph metrics (`GMLoad`). For example, load conditions `G".load` are measured by the number of inbound requests and the number of pending tasks within the community.

```
1  Input: Graph G, var search = {t_1, t_2, ..., t_n}
2  Output: List of communities
3
4  Select load, nodes from (
5      ( Select distinct(nodes) as G' from G
6        Where
7          ( <all> n in G'.nodes ) satisfy
8            Path (n to [1..*] G'.nodes) as P1
9            With (
10             P1.length = 1 and P1.trust = MTrust.HIGH
11             and ( [1..*] tag in P1.tags ) satisfy
12                 (search contains tag)
13           )
14     ) as SG1
15     Where
16       ( <all> G'' in SG1 ) satisfy
17         (G''.load <= GMLoad.MEDIUM)
18
19 Order by load asc
```

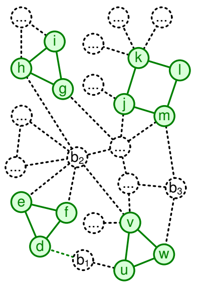


Figure 5: BQDL example 2: find ranked communities based on search criteria and metrics.

5.3 Finding Exclusive Brokers

The final BQDL example is depicted by Figure 6 to combine previously introduced concepts for broker discovery. **R3:** The basic idea of this example is to find brokers that are connected to exactly one candidate (target) community.

```
1  Input: Graph G, var source = {n_1, n_2, ..., n_i},
2         var search = {t_1, t_2, ..., t_n}
3  Output: List of brokers and communities
4
5  Select node, nodes from (
6      /* Select brokers */
7      ( /* ... */ ) as G1,
8      /* Select communities */
9      ( /* ... */ ) as SG1
10   )
11   Where
12     ( <all> n in G1.nodes ) satisfy
13       /* To one in SG1 */
14       Path (n to [1] SG1) as P1 With P1.length = 1
15
16 Order by node
```

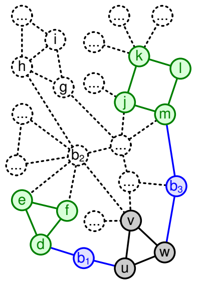
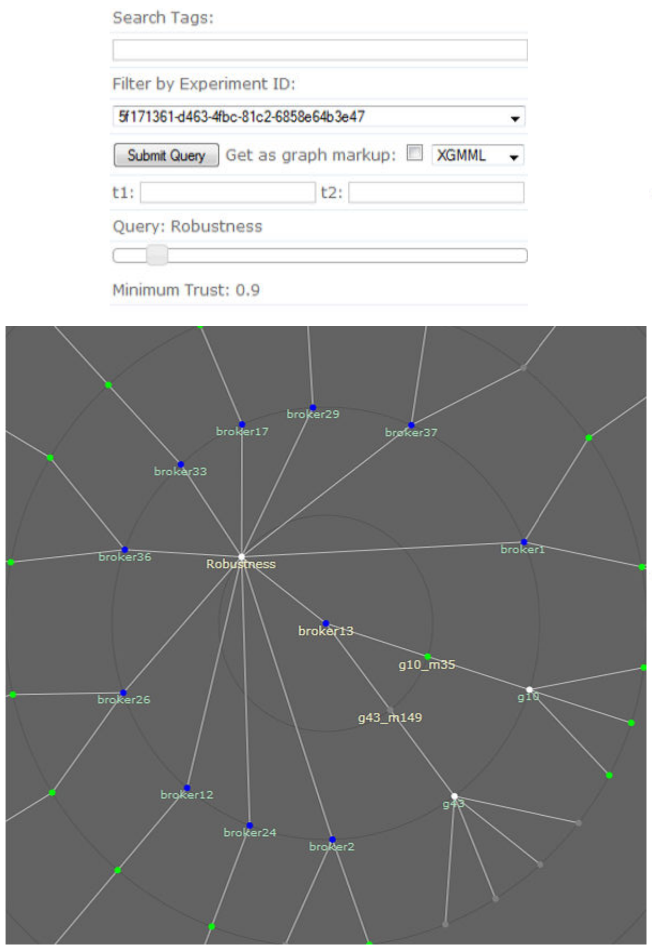


Figure 6: BQDL example 3: find exclusive brokers to connect two communities.


Search Tags:

Filter by Experiment ID: 5f171361-d463-4fbc-81c2-6858e64b3e47

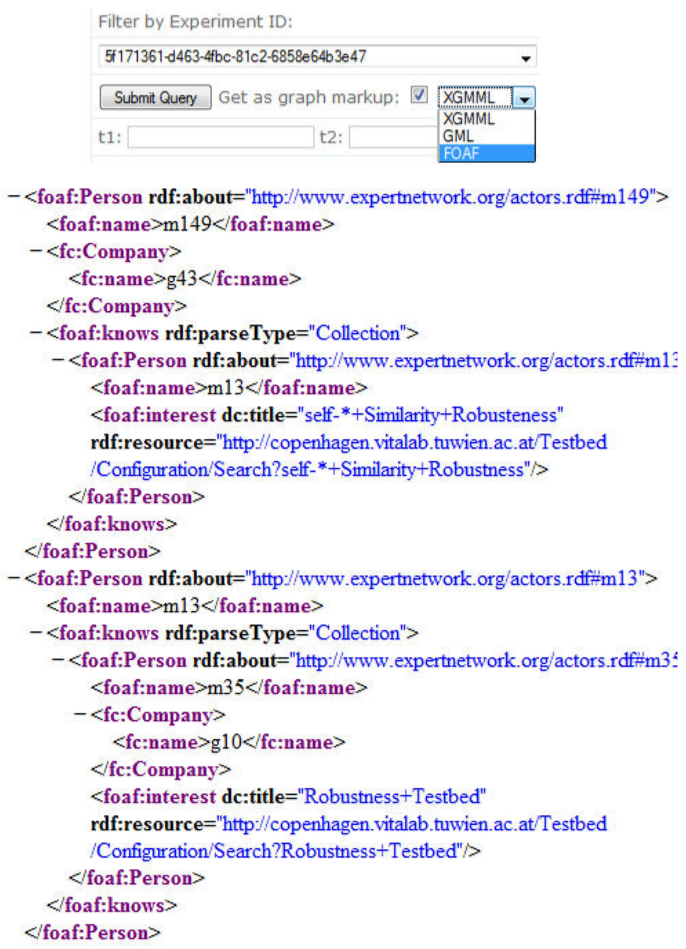
Submit Query Get as graph markup: ☐ XGMML

t1: t2:

Query: Robustness

Minimum Trust: 0.9

(a) Network visualization view.


Filter by Experiment ID: 5f171361-d463-4fbc-81c2-6858e64b3e47

Submit Query Get as graph markup: ☑ XGMML

t1: t2:

```
- <foaf:Person rdf:about="http://www.expertnetwork.org/actors.rdf#m149">
    <foaf:name>m149</foaf:name>
  - <fc:Company>
      <fc:name>g43</fc:name>
    </fc:Company>
  - <foaf:knows rdf:parseType="Collection">
    - <foaf:Person rdf:about="http://www.expertnetwork.org/actors.rdf#m1
        <foaf:name>m13</foaf:name>
        <foaf:interest dc:title="self-*+Similarity+Robusteness"
        rdf:resource="http://copenhagen.vitalab.tuwien.ac.at/Testbed
        /Configuration/Search?self-*+Similarity+Robustness"/>
      </foaf:Person>
    </foaf:knows>
  </foaf:Person>
- <foaf:Person rdf:about="http://www.expertnetwork.org/actors.rdf#m13">
    <foaf:name>m13</foaf:name>
  - <foaf:knows rdf:parseType="Collection">
    - <foaf:Person rdf:about="http://www.expertnetwork.org/actors.rdf#m3
        <foaf:name>m35</foaf:name>
      - <fc:Company>
          <fc:name>g10</fc:name>
        </fc:Company>
        <foaf:interest dc:title="Robustness+Testbed"
        rdf:resource="http://copenhagen.vitalab.tuwien.ac.at/Testbed
        /Configuration/Search?Robustness+Testbed"/>
      </foaf:Person>
    </foaf:knows>
  </foaf:Person>
```

(b) Example of FOAF profile.

Figure 7: Web-based broker discovery and network visualization tool.

Again, the source community is $\{u, v, w\}$. **A3:** Communities are retrieved along with brokers. Filtering is applied based on paths to obtain exclusive brokers. **O3:** The output of the query are brokers along with communities they are connected to (e.g., b_1, $\{d, e, f\}$). **D3:** First, a set of candidate brokers is retrieved and made available via the alias `G1` (line 7). This is the same procedure as introduced before (see Figure 4). Second, communities are retrieved and stored in `SG1` (line 9). Again, this is based on the same principle as introduced previously in Figure 5. We call brokers connecting exactly one community *exclusive brokers*. This is accomplished by the statements in 12-14 demanding for '`n to [1] SG1`'. The broker b_2 is a non-exclusive broker because it connects multiple communities $\{d, e, f\}$ and $\{g, h, i\}$, thereby making $\{g, h, i\}$ unreachable from the $\{u, v, w\}$ community perspective.

6. IMPLEMENTATION AND DISCUSSION

The implementation of BQDL is part of our initiative to create a testing environment for socially-enhanced SOA. The environment consists of a Web service-based simulation environment using the Genesis2 [10] framework and a middleware implementing user tools, logging, and eventing capabilities. Here we focus on tools assisting the users in discovering brokers based on visualized community structures.

6.1 Broker Discovery Application

The implemented prototype includes a Web-based broker discovery tool helping users in analyzing various BQDL queries and corresponding parameters. Figure 7 shows screenshots of the tool and an example FOAF profile that can be retrieved from the Web application. The users access information captured from the PVC environment. The network view is obtained by mapping raw SOAP-interactions into a graph representation composed of nodes (services) and edges (interaction links). In our implementation, this is performed by selecting a particular set of logs which are associated with an `Experiment ID`. After issuing the corresponding (BQDL) query, a graph is visualized consisting of several brokers connecting communities. By default, the collaboration network is visualized as a graph view as depicted in Figure 7(a). The user is able to select a trust threshold by moving a slider bar. A reduced (demanded) trust threshold results in more target communities being added to the visualization. *Color online*: target communities matching search criteria are depicted using a node that is labeled with the community identifier (white color) and a set of green colored nodes (labeled with the node's name) linked to the central community node (to indicate a node's membership to a community). Interactions can be retrieved as FOAF profiles (see Figure 7(b)) that include `<foaf:interest>` tags.

Experiment	# Req.	MIN	AVG	MAX	Total
1 (RP=10)	50	3167	9083	10368	52543
	100	1669	9369	10576	101244
	200	1825	9211	10748	190647
1 (**RP=50**)	50	1606	15955	29952	50762
	100	1482	27440	48562	98685
	200	1638	36313	47689	188573
1 (RP=100)	50	1606	15955	29952	50762
	100	1544	28560	57501	105331
	200	1591	55185	100370	202394
2 (RP=50)	100	2308	37891	63258	123677
3 (RP=50)	100	2854	42041	67516	136266
4 (RP=50)	100	3276	55058	84739	167778

(a) BQDL processing time.

Applied Tags in Exp. 4 (n=1029 and groups=230)	Frequ.
self-*	295
Robustness	306
Testbed	311
DB	314
Healing	321
Trust	322
WS	327
Autonomic	335
Similarity	341
Logging	353

(b) Tag frequency.

Query ID	BQDL query keywords	# Brokers	AVG proc. time
Q1	Robustness Logging	105	3993
Q2	Robustness Logging DB Testbed	134	3666
Q3	Robustness Logging DB Testbed Similarity	146	3478

(c) BQDL queries in Exp. 4, number of discovered brokers and AVG processing time.

Figure 8: BQDL processing statistics in simulated environment (in milliseconds).

6.2 SOA Testbed Environment

Our evaluations were gathered using the logging features of the Genesis2 framework [10]. Genesis2 has a management interface and a controllable runtime to deploy, simulate, and evaluate SOA designs and implementations. A collection of extensible elements for these environments are available such as models of services, clients, registries, and other SOA components. Each element can be set up individually with its own behavior, and steered during execution of a test case. For the experiments in this work, we deployed Genesis2 Backends to the *Amazon Elastic Compute Cloud*[3]. We launched, depending on the amount of involved service instances, two or three *Community AMIs* of the type *High-Memory Extra Large Instance* (17.1GB of memory) running a Linux OS. In the following, we provided each instance with the same Genesis2 Backend snapshot via mountable volumes from the *Elastic Block Store*. Finally, we deployed the following environment setup from a local Genesis2 Frontend. It included SOA-based PVCs established by Genesis2 Web services equipped with simulated behavior and predefined relations to provide communication channels and instantiate communities. Services act like HPSs when delegating each other new tasks, processing tasks, re-delegating existing tasks, or reporting tasks' progress status. Tasks are not delegated arbitrarily but must match the receivers capabilities. Therefore, they are tagged by three keywords one of which must match the picked receivers capabilities. As an intermediate, a broker combines capabilities of the two communities it connects. The broker avoids task processing and only forwards tasks. The finally deployed environments are variable in number of services, number of participants per group (2-5 services) and consequently also in number of communities and required brokers that connect at least each community with another (see also [13] for *minimum spanning trees* in social networks). Task processing and delegation decisions happen individually and in random time intervals (1-8 seconds).

[3]Amazon EC2: `http://aws.amazon.com/ec2/`

6.3 BQDL Performance Aspects

We conducted several experiments to test the performance of our BQDL implementation under varying characteristics such as varying number of nodes and groups. The results are summarized in Figure 8. We simulated environments with different numbers of nodes and interactions to obtain insights in performance aspects. BQDL tools (Figure 7) and BQDL related graph libraries implemented in C# have been deployed on our local (lab-based) blade servers equipped with Intel Xeon 3.2GHz CPUs (quad core) and 10GB RAM hardware. Interaction logs are managed by MySQL 5.0 databases. A client request pool (RP, see Table 8(a)) is created on a separate machine (Intel Core2 Duo CPU 2.50 GHz, 4GB RAM) to perform parallel invocations of the BQDL query Web service. Clients are connected with the server via a local 100MBit Ethernet.

The results of the first experiment are based on 198 nodes, 200 edges, and a total number of 10 distinct tags applied to interactions between nodes. The BQDL processing time for this environment is shown in Table 8(a). We vary the number of concurrent requests, denoted as RP, by launching multiple threads. Given a size of **RP=50** and a total amount of # 100 requests to be processed, setting RP=100 does not speed up the processing time of requests (i.e., the total time needed to process a number of requests). The average processing time increases by comparing RP=100 and RP=50 due to the overhead when handling a larger amount of requests simultaneously. Thus, we use RP=50 for all further experiments. Also, by processing a larger amount of requests, say # 200, the total processing time linearly increases with the number of requests. We increased the number of nodes and interactions to understand the scalability of BQDL under different conditions: experiment 2 with 579 nodes, experiment 3 comprising 774 nodes, and experiment 4 with 1029 nodes in the testbed. HPSs in the testbed have been deployed equally on multiple hosts, e.g., 3 cloud hosts in experiment 4 to achieve scalability. In subsequent experiments detailed in Figure 8 (experiments 2-4) we focus on a request pool with RP=50 and 100 requests to be processed by the BQDL service using different keywords (see Table 8(c)). To compare the experiments 1-4, we query the interaction graph using the keywords `Robustness Logging`. Increasing the number of nodes by a factor ≈ 3 (see experiment 1 and 2), the processing time of BQDL raises by 30%. Comparing the experiments 2 and 3 (node addition of $\approx 30\%$), the processing time increases by 10%. By comparing the experiments 3 and 4 (node addition of $\approx 30\%$), the processing time increases by 20%. Our experiments show that BQDL scales with larger testbed environments linearly. Furthermore, we used different BQDL query keywords as shown in Table 8(c). The number of discovered brokers increases given mutliple keywords (see Table 8(b) for the set of available tags). The average BQDL processing time is not significantly influenced by the number of used keywords.

7. CONCLUSION AND FUTURE WORK

In this paper we introduced the notion of *brokers* in socially-enhanced service-oriented environments. The idea of our broker approach is derived from theories found in social sciences (structural holes). Brokers can be modeled as Human-Provided Services to support the seamless integration of human capabilities in service-oriented infrastructures. The novelty of our approach is that brokers are not discovered based on static policies or static broker capabilities. In this work, we proposed the discovery of brokers based on mining techniques and the automated computation of periodically updated metrics based on interaction logs. This not only helps to find suitable brokers but also relevant communities and social networks to which brokers are connected to. Furthermore, we introduced the *Broker Query and Discovery Language* (BQDL) enabling the definition of discovery and interaction policies. BQDL operates on a graph structure that is maintained and updated through mining. Furthermore, we discussed the implementation and performance aspects of BQDL. Future work will include the implementation of a link-based reputation ranking algorithm for brokers. Further scalability analysis will be performed in service-based testbeds.

Acknowledgment

This work received funding from the EU FP7 program under the agreements 215175 (COMPAS), 215483 (SCube) and 216256 (COIN). The authors thank Lukasz Juszczyk for providing the Genesis2 framework.

8. REFERENCES

[1] D. Artz and Y. Gil. A survey of trust in computer science and the semantic web. *J. Web Sem.*, 5(2):58–71, 2007.

[2] D. Brabham. Crowdsourcing as a model for problem solving: an introduction and cases. *Convergence*, 14(1):75, 2008.

[3] D. Brickley and L. Miller. Foaf vocabulary specification 0.98, 2010. http://xmlns.com/foaf/spec/.

[4] R. S. Burt. Structural holes and good ideas. *American Journal of Sociology*, 110(2):349–399, Sept. 2004.

[5] L. M. Camarinha-Matos and H. Afsarmanesh. Collaborative networks. In *PROLAMAT*, pages 26–40, 2006.

[6] C. Dwyer, S. R. Hiltz, and K. Passerini. Trust and privacy concern within social networking sites. In *AMCIS*, 2007.

[7] M. Ford et al. Web Services Human Task (WS-HumanTask), Version 1.0., 2007.

[8] J. Golbeck. Trust and nuanced profile similarity in online social networks. *TWEB*, 3(4), 2009.

[9] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 3(4), 2000.

[10] L. Juszczyk and S. Dustdar. Script-based generation of dynamic testbeds for soa. In *ICWS*, pages 195–202, 2010.

[11] J. Kleinberg. The convergence of social and technological networks. *Commun. ACM*, 51(11):66–72, 2008.

[12] J. Kleinberg, S. Suri, E. Tardos, and T. Wexler. Strategic network formation with structural holes. *ACM Conference on Electronic Commerce*, 7(3):1–4, 2008.

[13] P. J. Macdonald, E. Almaas, and A.-L. Barabási. Minimum spanning trees of weighted scale-free networks. *Europhys. Lett.*, 72(2):308–314, 2005.

[14] M. J. Metzger. Privacy, trust, and disclosure: Exploring barriers to electronic commerce. *J. Computer-Mediated Communication*, 9(4), 2004.

[15] T. P. Moran, A. Cozzi, and S. P. Farrell. Unified activity management: Supporting people in e-business. *Com. of the ACM*, 48(12):67–70, 2005.

[16] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation for e-businesses. In *HICSS*, page 188, 2002.

[17] C. Petrie. Plenty of room outside the firm. *IEEE Internet Computing*, 14, 2010.

[18] R. Ronen and O. Shmueli. Soql: A language for querying and creating data in social networks. In *ICDE*, pages 1595–1602, 2009.

[19] D. Schall and S. Dustdar. Dynamic context-sensitive pagerank for expertise mining. In *Social Informatics*, volume 6430 of *LNCS*, pages 160–175. Springer, 2010.

[20] D. Schall, H.-L. Truong, and S. Dustdar. Unifying human and software services in web-scale collaborations. *Internet Comp.*, 12(3):62–68, 2008.

[21] F. Skopik, D. Schall, and S. Dustdar. Modeling and mining of dynamic trust in complex service-oriented systems. *Information Systems*, pages 735–757, 2010.

[22] W. Tsai. Social capital, strategic relatedness, and the formation of intra-organizational strategic linkages. *Strategic Management Journal*, (21):925–939, 2000.

[23] W3C. Sparql query language for rdf, 2008. Online: http://www.w3.org/TR/rdf-sparql-query/.

[24] C.-N. Ziegler and J. Golbeck. Investigating interactions of trust and interest similarity. *Decision Support Systems*, 43(2):460–475, 2007.