S

Social Interaction Analysis for Team Collaboration

Ognjen Scekic, Mirela Riveni, Hong-Linh Truong and Schahram Dustdar
Distributed Systems Group, Institute for Information Systems, TU Wien, Vienna, Austria

Synonyms

Collaboration analysis; Collaboration metrics; Collaboration platforms; Collective adaptive systems; Crowdsourcing; Human-based services (HBS); Interaction patterns; Process mining; Rewarding; Social computing; Social trust; Socio-technical systems; Task assignment; Team collaboration; Team formation

Glossary

Actor: Entity (human or computer) possessing a capability to act intelligently and process specific assignments (activities/tasks).

Atomic task: Task that can be handled by an individual actor.

CAS: Collective adaptive system.

Collaboration system (platform): Information system supporting execution of collaborative processes.

Collaborative process (collaboration): Joint effort of a (limited) number of actors with the goal of performing a task. A collaborative process has a limited duration and requires coordination among actors (due to task dependencies).

Composite task: Task that must be handled by multiple actors due to size or complexity. A composite task can be broken down into atomic tasks.

CSCW: Computer-supported cooperative work.

HBS: Human-based service.

Metric: Precisely defined, context-specific measure of some properties.

QoD: Quality of data.

QoS: Quality of service.

SOA: Service-oriented architecture.

SOC: Service-oriented computing.

Task assignment: The art to divide a (composite) task into (sub)tasks and assign

R. Alhajj, J. Rokne (eds.), *Encyclopedia of Social Network Analysis and Mining*,
DOI 10.1007/978-1-4614-7163-9_257-1

	them to appropriate actors.
Task:	Piece of work to be solved, typically complex enough to require knowledge or processing power of a large number of individual actors.
Team formation:	Process consisting of identifying appropriate actors for performing all atomic tasks and establishing of internal coordination and functioning rules in the team.
Team:	Set of actors taking part in a collaborative process. Team lifetime is considered equal to the lifetime of the collaborative process.
WS:	Web service.

Definition

In this article, we look into different types of collaboration systems. We describe team structures and discuss different forms of collaborations they support. In particular, we focus on interaction processes that are supported by the system and discuss different metrics used to describe and analyze such systems. We discuss three main team collaboration types: static, ad hoc, and open collaboration. We then focus on interaction analysis and discuss appropriate interaction metrics.

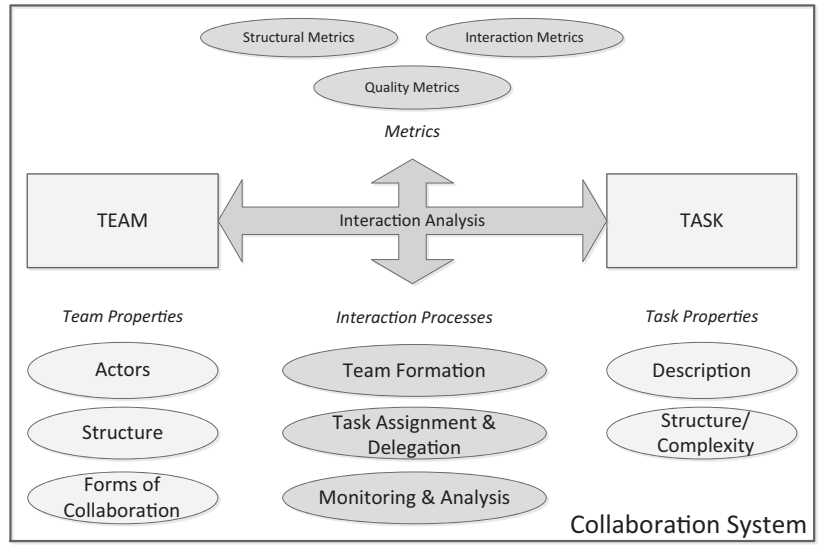
Introduction

With the advent of Web 2.0 and social networks, millions of users around the world were given the opportunity to collaborate, share ideas, and coordinate their efforts easier than ever before. These developments lead to an increased interest to exploit these opportunities, both in the research community and in the industry. Such collaborative efforts are supported by different types of *collaboration systems,* providing automated or semiautomated actor management (e.g., modeling, reputation, and rewarding), task management (e.g., modeling, creation, division, scheduling, aggregation, and monitoring), and process execution environment (including actor communication and coordination).

Collaboration systems enable different collaboration types. Depending on the type of system and type of problem to be solved, different team structures are possible. The team structure guides the interactions and collaboration among team members and consequently plays an important role in a team's performance. Figure 1 depicts the fundamental elements of a collaboration system that we discuss in this article.

Key Points

- *Human computation systems* – Systems in which human actors perform assigned tasks in a precisely defined sequence (e.g., by following an algorithm). The execution is explicitly controlled and coordinated by the system and expected to yield precise results (Law 2011).
- *Workflow management systems* – Systems that allow modeling of tasks and their execution scenarios. Notable representatives of such systems are the various business process management (BPM) systems. Although tasks can be performed by human actors, the traditional understanding of the notion of a workflow system does not include an integrated management of human-performed tasks.
- *Mixed systems* – Systems where both human and computer actors process the tasks. Humans are deeply integrated into the system, making both types of actors first-class citizens of it. The decision on who processes a particular task can be made by the system. While computer-performed tasks are accurate, employing humans for certain tasks requires dealing with uncertainties both in terms of human behavior and the quality of results.



Social Interaction Analysis for Team Collaboration, Fig. 1 Elements of a collaboration system

- *Crowdsourcing systems* – Systems in which the task is offered, rather than assigned explicitly, to an unknown and usually large group of people who can freely accept and perform the tasks.

Historical Background

The idea of combining research on how humans work, communicate, and cooperate and the research on how computer systems can efficiently support such collaborations led to the creation of an interdisciplinary research area known as *computer-supported cooperative work* (CSCW) in the 1980s (Grudin 1994). Initially, the research was focused on small-scale collaborations, e.g., within companies or interest groups. With the wide adoption of Internet technologies, service-oriented architectures (SOA), mobile and cloud computing, and especially social networks, nowadays it is possible to carry out large-scale collaborations, possibly involving thousands of collaborators across boundaries of multiple organizations and countries. In the absence of an agreed naming taxonomy, a number of nuanced terms are commonly used in literature to generally denote systems and platforms for managing collaborations: *social computing systems*, *socio-technical systems*, *hybrid collective adaptive systems (hCAS)*, and *cyber-human systems.* Some examples of today's well-established types of such collaboration systems are given as follows.

Team Collaboration Analysis

Team Properties

We consider three important team properties: (a) *actors* making up the team, with their different skills, qualities, and personalities; (b) *structure,* representing the relationships, interactions, scale, and elasticity of the actor population; and (c) different *forms of collaboration* among the actors.

Actors and Team Structure

Actor teams are usually modeled as undirected or directed (multi)graphs with nodes representing people or teams of people and edges representing relationships between them (Newman 2010). Often, the edge is associated with a set of properties quantifying the interaction between the two nodes it connects and annotated with a context, representing the type of the relationship (e.g., social, professional collaboration, trust). Therefore, a team network can be modeled as a graph consisting of nodes representing actors, sets of skills forming their profiles, and edges

representing relationships and associated contexts of relationships (Caverlee et al. 2008).

Depending on the lifetime and the scale of the observed actor set, we can consider different structural properties with varying relevance. Small-scale actor teams are usually assembled relying on the compatibility of the actors and their relationships, allowing the delegation of task coordination to the actors themselves. Typical examples are expert teams where actors have previous mutual collaboration experience, of friend cliques, where the common social fabric is a promise of a successful collaboration. The lifetime of such teams is short, and the structure is mostly static and determined at assembly time. Large-scale collaborations exhibit properties of populations – an extended lifetime and complex structure changing dynamically throughout execution time, thus requiring efficient automated mechanisms for coordination. *Scale* and *turnover* (Lee and Paine 2015) are metrics used to describe the magnitude and volatility (or stability) of such collaborations. For open collaborations, the turnover represents the rate of actors joining/leaving the collaboration. For managed collaborations, the scale and turnover are important inputs for determining the next *elastic* actions to take (Riveni et al. 2014).

Forms of Computer-Supported Team Collaboration

Static Collaboration Static collaboration is characterized by well-defined, long-lasting/repetitive processes (tasks), executed by human actors with specific assigned roles. Such kind of collaborations is usually found in companies that encode and execute their daily business use cases as business processes by using workflow technologies. This collaboration type makes no use of the underlying social networks connecting the actors to alter or enhance the collaboration in any way. As such, this approach works well only in cases where the predictability of the process execution is high and where no adaptability is required.

Ad Hoc Collaboration Unlike static collaboration, the ad hoc collaboration is suitable when performing highly dynamic tasks that change in time or complex tasks that occur only once and are not repeated. In this type of collaboration, tasks are initially defined, but the actors performing them are provisioned only at runtime. Ad hoc collaborations often cross organizational boundaries and are distributed in nature – in terms of software services used, in terms of actors executing the tasks, as well as in terms of control. Actor provisioning can be fully automated or partially performed by the actors themselves, often relying on social and other underlying networks connecting the actors.

Open Collaboration In open collaborations, a task can be actively shaped by the actors. The actors (often belonging to a professional community or an interest-based community) contribute freely to the task resolution during runtime. A task is not strictly assigned to a particular actor, but instead it is editable by (m)any community members upon their wish. In this case, the coordination between the actors can affect the quality of the task (Kittur and Kraut 2008). Data quality is controlled by the system itself and/or by a designated entity, often relying on the feedback information from data users. Open collaboration is particularly suitable for longer running, best-effort tasks, with no strict quality and time constraints, but requiring distributed know-how.

Open-source development, Wikipedia, and community-based Q&A websites are among the best examples of open collaboration. Examples of open collaboration enabling technologies and platforms include cloud services (e.g., Amazon EC2), sharing and collaboration platforms (e.g., Dropbox, Google Docs, Mendeley, and some more secure and private ones such as ownCloud and ownDrive), and open-source repositories (e.g., GitHub, GitLab, SourceForge).

Most existing systems employ a combination of different collaboration types, attempting to reduce the respective limitations of individual types and offer more versatile collaborations. CrowdLang platform (Minder and Bernstein 2012) combines static and ad hoc collaboration by integrating human-provided services into workflow systems. The workflow is not fully

static but can be designed as needed by (re)combining a number of generic (simple) collaborative patterns (e.g., iterative, contest, collection, divide and conquer). The CrowdComputer platform (Tranquillini et al. 2015) combines static and open collaboration. The tasks are executed following a workflow, but the tasks are split into atomic tasks offered to individual workers through different "tactics" (e.g., marketplace, auction, mailing list). The SmartSociety platform (Scekic et al. 2015) supports combining all three collaboration types, allowing the actors to actively participate in determining and executing the team and the workflow (collaboration plan).

Task Properties

Task Description

Considering the general nature of the tasks that can be handled by a team composed of human actors, describing tasks precisely and unambiguously is extremely challenging. The difficulty lies in expressing the information that needs to be interpreted by each actor in the same way. At the same time, the effort required to properly interpret a task's objectives must be considerably smaller compared to the effort required to perform the task itself. In practice, the tasks can be described, *informally* and *formally*:

- Informally describing tasks means expressing the required outcomes in natural language, accompanied with simple examples. This approach is usually taken by today's crowdsourcing platforms that handle simple tasks. Also, informal description may be preferred in cases where tasks require aesthetic judgment or when the required outcome of the task is too vague to be expressed more precisely (e.g., on websites running creativity contests).
- Formally describing tasks means employing a specific notation that precisely defines how the task should be processed and what should the outcome be. Formal task description is usually used in specific environments, most notably in business process modeling (BPM). Initial versions of the most prominent business control-flow languages, such as BPEL, did not support specification and invocation of human interactions. An extension to BPEL, known as BPEL4People, was proposed in 2005 to allow modeling of human interactions within business processes by introducing the concept of *people activities*. A people activity can be described according to the WS-HumanTask specification. In this way, humans can be internally represented as Web services and integrated into the system.

Task Structure and Complexity

Task structure directly influences the team structure. Different task structures and complexities demand specific types of collaboration in terms of communication form, coordination protocols, adaptation schemes, and outcome type. Subtask interdependencies are one of the fundamental factors determining the task structure and task complexity. One basic task structure categorization is that into *parallel* and *sequential* tasks. Parallel tasks contain subtasks that can be executed independently in parallel, while a sequential task is composed of subtasks whose execution must follow a strict order. A subtype of sequential tasks is *iterative* tasks, where the output of one actor is given as input to another actor for subsequent task execution. An experiment and analysis of parallel and iterative approaches in open systems can be found in Little et al. (2010).

Apart from subtask interdependencies, other nonstructural factors can influence a task's complexity, such as the following: (a) number of atomic tasks, (b) growth (Dustdar and Bhattacharya 2011) (the number of atomic tasks can grow in runtime, necessitating team-size adaptability), and (c) task cardinality (tasks can be designed to be executed by one or many actors in one-to-one, many-to-one, many-to-many, and few-to-one fashion). See Quinn and Bederson (2011) for details and examples.

Interaction Processes

Team Formation

The problem of team formation consists of selecting suitable actors to perform a given task

(out of a larger group of available actors) and organizing them in a collaborative structure. The first problem with identifying "suitable" actors is that suitability is highly context dependent and difficult to define precisely. Furthermore, suitability can have many different aspects. For example, the minimal suitability requirement for an actor is to possess the skills to perform the task. But, at the same time, for a successful teamwork, factors like trust, motivation, experience, and personal relations with other team members can be equally important.

Initially, the research focused on locating individual best-matching actors for a required set of skills and other individual properties. However, a group of top individuals does not guarantee the quality of their collaboration. Subsequent research efforts began taking into account the underlying social relations among the actors (e.g., friendships, managerial relations, previous business interactions, interests, connectedness, and social trust). Finally, recent systems aim to include human actors as first-class citizens allowing them to actively influence at runtime the team formation process (Rovatsos et al. 2015).

After selecting suitable actors, the following step in ensuring a successful collaboration is setting up a collaborative organization and environment. Although collaboration patterns in a team often resemble those in the underlying social networks, other factors like coordination cost, user preferences, and context are also important.

Whichever the properties considered, they are always measurable and quantifiable, meaning that the problem of team formation can be ultimately expressed as an optimization problem where we want to optimize certain performance aspects of the team as a whole (speed, quality, cost, and response time) while respecting the fixed constraints.

In general, team formation can be:

- *Self-organizing* – The actors themselves lead the team formation in a collective-intelligence fashion and set up the collaboration environment. The system assists the process (e.g., by enforcing negotiation rules, counting the votes) but does not make decisions on actor participation.
- *Centralized* – Team formation and setting up of collaborative environment is managed by the system, including the decision on participating actors.

Wikipedia and open-source community are striking examples of how self-organizing teams can perform well. The assumption is that the actors taking part in collaboration will perform best if they are given the possibility to modify and adapt the collaborative environment. This includes also the initial team formation. For example, in Gaston and DesJardins (2005) the authors investigate a system that enables actors to locally modify their collaborative environment according to their social network preferences (i.e., to *rewire* the local network topology) with the goal of achieving globally noticeable, collective performance improvement.

The most problematic aspect of self-organizing teams is the discrepancy between local and global effects. Although we rely on the collective intelligence of the actors, in practice, actors may not know how or when to modify the local network to achieve global improvements, since their actions are based upon their partial views only.

Centralized team formation is entirely handled by the system. Internally, the system can employ an algorithm or human actors to assemble the team:

- *Human-managed team formation* relies on human actors offering their referrals and recommendations via Web services, thus leveraging crowdsourcing techniques to identify the best candidates from their social networks. An example of such a system is PeopleCloud (Lopez et al. 2010).
- *Algorithmic team formation* relies on an algorithm to select actors and assemble the team. A lot of research efforts have been directed in this sense, producing a number of different algorithms. In Schall and Dustdar (2010) and Schall et al. (2012), the authors modify the well-known page ranking algorithms

PageRank and HITS to identify the best team members, based on their previous interactions. In Lappas et al. (2009) and Anagnostopoulos et al. (2012), the goal is to minimize the total coordination cost of the newly established team, while in Dorn and Dustdar (2010), the optimal team is chosen as a trade-off between skill coverage and actor connectivity. Anagnostopoulos in Anagnostopoulos et al. (2010) presents an algorithm for forming minimum size teams with minimum workload that satisfies required team skills. Kargar et al. present team formation algorithms with communication and personnel cost minimization in An et al. (2013). Sagar in Sagar (2012) has presented a formal model for task assignment and finding team collaborators in social networks by using shortest distance in the network. In Caverlee et al. (2010), the social trust between the team members is regarded as the most important factor in forming efficient collaborations.

Task Assignment and Delegation

Routing and Delegations Task delegation mechanisms are being explored as forms of coordination and load balancing in human computation. The concept of *social routing* is introduced in Dustdar and Gaedke (2011) as a form of delegation of tasks by task owners to actors from their social, professional, and other context-based community networks or the crowd. The so-called social router can be a software service that actually does the task forwarding across different types of networks depending on the requirement of the actor wishing to delegate the task.

Historical data on delegations (e.g., the executed/delegated tasks ratio, frequently used delegates) can serve as a good indicator of actor's role, performance, and social relationships. For example, a high number of task delegations testify a coordinating/managing role. However, the same information, if interpreted properly, is a potential indicator of actor's laziness. Moreover, it was shown (Sun et al. 2014) that actors favor the familiarity with the delegates over their assessed suitability for the given task. Delegation data can be used as metrics in actor selection and team formation algorithms. Moreover, delegation measures can be used in trust inference mechanisms. For example, Riveni et al. (2015) present a trust model where the rate of successfully executed delegated tasks is included in the definition of an actor's reliability metric, which in turn is included in the actors trust score. On the other hand, if the receivers of delegated tasks are considered trustworthy, new trust-based links will be created between the delegator and the delegates (Skopik et al. 2010).

Delegation Patterns in Business Process Activities The four main delegation patterns, detailed in Kloppmann et al. (2005), are:

- *Nomination* pattern allows predefined actor (s) to decide to whom to assign a task.
- *Escalation* pattern allows transfer of responsibility for task execution to other human actors when the originally assigned actor cannot meet task's time constraints. When an escalation is triggered, actors designated as escalation recipients are notified and allowed to decide how to proceed with the task execution, possibly reassigning it to another actor. When dealing with uncertainties related to human processing, this is an inevitable mechanism that must be supported.
- *Chained execution* pattern forces the actors to perform a specific sequence of actions, where the concrete actions may be determined only in runtime. The actors can be assigned a new action only after completing the previously assigned one. This pattern allows implementing a "bag of tasks" kind of parallelism, with each actor repeatedly removing tasks from the bag.
- *Four eyes principle* pattern allows two actors to take a public or a private decision on the same issue independently *(separation of duties*).

Algorithmic Task Life Cycle Management In cases when subtasks are clearly delimited and subtask dependencies are static and do not change in time, parallelizing a task execution is fairly

easy. Some application domains, such as crowdsourcing systems, are characterized by exactly such properties.

This has led researchers to dedicate a lot of effort to automate task life cycle management transparently for the programmer, by developing a number of programming language extensions/libraries that work on top of existing commercial crowd-sourcing systems, such as Amazon Mechanical Turk. The extensions are typically able to automatically split a task; to assign/offer the subtasks to the actors in the crowd respecting the dependency, cost, and time constraints; and to merge the processed subtasks into the final resulting task. Additionally, automated quality control processes may be also offered. Most commonly, these are based on peer reviews or on a combination of redundant processing and majority rule. For example, an image that needs to be tagged may be submitted to multiple actors, but the aggregated result will contain only tags suggested by multiple independent actors. The data quality requirements can have a direct influence on task assignment, as they may introduce assignments not explicitly required by the user but performed transparently by the system. In fact, the main purpose of algorithmic handling of task assignment is exactly to move the burden of task life cycle management from the user to the system.

Collaboration systems can manage task assignments automatically throughout the entire execution time, repeating them when needed. For example, Little et al. (2009) show a system offering the possibility of iterative task execution, by reassigning previously processed tasks a number of times in order to improve the final quality of work by incrementally building upon previous work. In Marcus et al. (2011), a system can autonomously decide when to assign pleasing tasks to specific actors in order to motivate/reward them.

Another major advantage of algorithmic task assignment is the cost optimization. For large-scale collaborations, the system is able to assign the tasks in such a way to reduce the coordination costs better than human managers could do. For example, the task can be assigned to actors possessing similar professional skills and backgrounds, or the system can adjust task prices and time allotments based on the feedback obtained from monitoring data (Barowy and Berger 2012).

Collaboration Monitoring and Analysis

Monitoring and analyzing collaborative processes is necessary to gather important metrics about the performance of teams and actors and the quality of processed tasks. Such metrics are then used to detect bottlenecks, improve performance, and decide on appropriate compensation of the actors. As these metrics play a fundamental role in determining overall collaboration efficiency and costs, every collaboration system must support some kind of monitoring and analysis functionalities.

Monitoring can be performed during the runtime of a collaborative process *(active monitoring*) or it can be performed post-runtime, e.g., by *log mining.* Log mining is usually considered a part of more complex analysis processes, known as *workflow/process mining* (van der Aalst 2011; Zhang and Serban 2007).

Active monitoring is suitable for detecting anomalies that require quick responsive actions and team adaptations. An example of a system capable of monitoring and analyzing SOA-based collaborative processes can be found in Truong and Dustdar (2009).

Log mining, on the other hand, is used to gather less obvious information about the internal functioning of the team, since it considers the backlog of all recorded actions performed during previous collaborations. This allows discovery and prediction of critical execution paths, expected workload distribution, actor performance, and identification of previously unknown collaborative social networks, e.g., the network of most trusted colleagues or the groups of workers that together collaborate most efficiently as a team.

Metrics lie at the very heart of any monitoring and analysis process. In the following section, we present an overview of the metrics used by different collaboration systems.

Collaboration Metrics and Patterns

Metrics used in collaboration systems can be divided into three major categories:

- *Structural metrics* – defining the mathematical properties of the social/collaborative network connecting the actors.
- *Interaction metrics* – defining various properties of individual actors or actor groups, emerging as the result of past interactions.
- *Quality metrics* – defining quality criteria for actor performance and for task outcome data.

Structural Metrics and Network Patterns

Structural metrics and network patterns are based on mathematical properties of the social graph connecting the actors in a collaboration network (team). They provide useful insights into the functioning and self-organization of actors in a team. Structural metrics are well researched. In the following, we briefly mention some of main structural metrics:

- *Centrality measures* – include various metrics that identify the importance of an actor within a network in different contexts of importance. Some of the *most important centrality metrics are* degree centrality, closeness centrality, betweenness centrality, *and* eigenvector.
- *Structural groups* – They refer to various group patterns that can be identified within networks, such as *core* (denoting a subset of actors within a network where each actor is connected to at least *k* other actors within the same subset), *k-component* (denoting a subset of actors in which each two actors are connected by at least *k* independent paths), and *clique* (denoting a subset of actors all directly connected to each other).
- *Transitivity and reciprocity* – Transitivity reflects the "friend-of-a-friend" concept, i.e., if an actor *a* is connected by an edge to another actor b, and *b* is connected to c, then *a* is also connected to c. Reciprocity, on the other hand, denotes the probability that actor *b* points to actor *a* if actor *a* points to *b*.
- *Similarity* – It is defined by *structural equivalence* and *regular equivalence* metrics. See Newman (2010).

Details about all these and other metrics, as well as about ranking algorithms, can be found in Newman (2010).

Interaction Metrics

Interaction metrics can be defined at two levels: *individual level* (targeting individual actors) and *group level* (targeting multiple actors or the entire team). Individual interaction metrics describe a property of an individual actor that is shaped by the interaction in which the actor has participated. Group interaction metrics describe properties of particular interactions between actors, possibly including the collaboration as a whole.

Certainly, the most important actor-level metrics are *skill coverage* and *trust.* Skill coverage represents a degree to which an actor or a team possesses necessary skills to perform a task. This metric is important because it describes how much a team's set of skills deviates from the optimal one for a given task. The problem of matching skills is equivalent to the problem of functional matching in Web service compositions.

Trust, as a computational concept, was formalized in Marsh (1994) and since then it has been seen as a metric of great importance for selection of appropriate actors during the team formation phase. Trust is defined as an indicator of an actor's expectation about another actor's future behavior based on knowledge from previous interactions, and which inherently involves a degree of uncertainty about this behavior and its outcomes. Trust is highly context dependent, and one actor may have information about several scope-specific trust values for another actor. A scope can be the membership in a professional network, social network, or a collaboration team. Inferring trust is important in several cases:

- For actor discovery and team formation algorithms, when determining actor suitability for specific tasks

- For team optimization, adaptation, and risk management purposes
- For delegation mechanisms, e.g., when selecting a collaborator that may be a part of the extended team structure for the purpose of load balancing in cases of unexpected load

We can distinguish three types of trust based on the types of actors and interactions that are taken into account for its inference:

- *Local trust* or *direct trust* (sometimes also called *private reputation*) – firsthand trust, inferred from the outcome of an actor's previous interactions with the trustee
- *Recommendations* – secondhand trust inferred from the outcome of past interactions between a well-trusted entity and the trustee
- *Global trust* or *reputation* – aggregated community trust, inferred from outcomes of past interactions between third-party actors and the trustor (Skopik et al. 2009)

Other actor-level metrics include *task familiarity* and *team familiarity* (Espinosa et al. 2007). These are especially important for open collaboration where the system cannot assign a task to appropriate and trusted actors. If some of the actors within an open collaboration are already familiar with other actors, the coordination will be positively affected.

Team familiarity is important in large teams where effective team coordination is more difficult. Team familiarity is a function of multiple other metrics such as quality of prior interactions with a coworker, prior belonging to the same team, and prior experience with the same team structure and organization. Hence, this measure is closely related to *trust*.

Homophily is another metric closely related to team familiarity, where an actor chooses a collaborator based on profile similarity. Fazel-Zarandi et al. (2011) conclude that homophily (e.g., gender, same interests, tenure status) is considered more important in choosing research collaborators than, for example, their qualification level.

Task familiarity is best explained with an example of open source software development team. The bigger the number of interdependent modules, the more complex is the task. This increases the amount of information to be processed by human actors; thus, it is important that actors have a reasonable amount of task familiarity. Details of a model for performance analysis of teams based on task familiarity and team familiarity can be found in Espinosa et al. (2007).

Group-level metrics describe performance properties of a collaboration. One of the fundamental metrics describing collaborations is the team size. The bigger the number of collaborating actors, the more communication and coordination among them is needed. For example, in Kittur and Kraut (2008), the authors use Wikipedia to analyze how the number of editors and the coordination methods affect the article quality in terms of accuracy, completeness, and clarity.

A metric indicating *interaction intensity* between an actor and other important actors is measured in specific interaction contexts. It is used in the aforementioned *DSARank* ranking algorithm (Schall and Dustdar 2010).

The relevance of the connections to important actors is the most important factor in determining the reputation of an actor. The reliability of the feedback information in reputation systems depends on the reputation of actors providing the feedback. Reputation information is valuable when an actor lacks information based on direct experiences with another actor. However, when this information is available and appropriate, the private or direct trust weights more than trust values based on reputation data. In this case the weight of data from direct interactions should be determined by calculating the minimum number of direct/local trust or rating values that should be maintained by an actor for the actor providing the service/executing a task (Noorian et al. 2012).

Collaboration cost is an important metric because of its direct business influence. This metric takes into account not only the price of task processing paid to the actors, but rather the total costs, including the communication and coordination costs. It is used as the basis for the cost optimization algorithms, as shown in the

following systems – Quirk (Marcus et al. 2011) and AUTOMAN (Barowy and Berger 2012).

Automatically discovering *collaboration patterns* naturally occurring among actors opens up a possibility to identify particularly (un)successful collaboration groups or execution sequences. This information can in turn be used to optimize collaborative process. Identifying collaboration patters is one of the central topics of process mining.

Quality Metrics

Quality of Data (QoD) Metrics As collaboration systems deal with various human-performed tasks, and the data quality primarily depends on the type of tasks, trying to develop a general set of quality metrics makes little sense. For example, metrics listed in Table 1, such as data completeness, freshness, and accuracy, are well-known metrics, but their definition is highly dependent on the goal of their use. Instead, different metrics are developed for particular application domains (e.g., Hu et al. (2007) and de La Robertie et al. (2015) present models to assess article quality in Wikipedia). However, it is exactly the fact that humans participate in the collaborative processes that introduces a concept common to all the application areas – that of *uncertainty* or *inaccuracy* (Parameswaran and Polyzotis 2011). The main sources of uncertainty are caused by the dynamic and unexpected behavior of humans: humans make mistakes, are subjective, and exhibit malicious/dysfunctional behavior. Thus, approaches for dealing with uncertainty should be included in supporting systems.

Different research communities deal with uncertainty differently. However, all approaches rely on some probability metrics that quantify our belief that a single task is performed correctly. In principle, all approaches can be divided into two categories:

- *Optimistic approaches* – Processed tasks are returned along with a confidence (accuracy) estimate. The data user accepts the results but must be aware that a certain percentage of the results will be wrong.
- *Pessimistic approaches* – The system applies various mechanisms for error detection and correction and usually resubmits the task to multiple actors until the merged result satisfies the required quality threshold.

Social Interaction Analysis for Team Collaboration, Table 1 Overview of metrics and patterns used in collaboration systems

Structural metrics		Centrality measures (Degree, closeness, betweenness, eigenvector, etc.)
		Structural groups (Cores, components, cliques)
		Transitivity, reciprocity
		Similarity, equivalence
Interaction metrics	Actor level	Trust, reputation functional/skill coverage
		Task familiarity, team familiarity
	Group level	Structural groups
		Team size
		Link quality, interaction intensity
		Collaboration patterns (Delegations, escalations, redundant processing, iterative processing, etc.)
Quality metrics	Quality of data (QoD)	Uncertainty, completeness, accuracy, freshness, relevancy etc.
	Performance	Availability, response time, success rate, etc.
	Rewarding and incentives	Effort, productivity, quality of work

Actor performance quality metrics are similar to the "traditional" Web service metrics, like average execution time, number of invocations, and availability. On the group and collaboration level, these metrics measure and predict the existence of various invocation patterns, i.e., the probabilities that certain services will be called in a particular order with respect to other services. A detailed discussion on interaction metrics can be found in Truong and Dustdar (2009).

Incentives and rewarding are important and effective mechanisms for indirectly influencing quality and motivation of human actors in collaborations. The principal metrics in use in today's computer-supported collaboration systems are:

- *Effort* – It measures an actor's determination to perform a task. The main purpose of this metric is to provide a way to compare the performance of both experienced and inexperienced actors. For example, an inexperienced actor may put in a lot of his time and resources only to perform a task worse or slower than an experienced actor. However, for the purpose of incentivizing, a higher effort level should be compensated with a higher reward, because it will ultimately lead to better experienced actors.
- *Productivity* – It expresses the number of units processed in a time period. This metric is suitable for piecework and easily quantifiable tasks (e.g., bug reporting, image tagging, text translation). Kasunic (2008) has defined effort and productivity metrics for software projects.
- *Quality of work* – This metrics expresses the quality of the working process of an actor. It should not be confused with the quality of data (QoD) of processed tasks. This metric is used to assess actors when the task's QoD cannot be easily determined or when it cannot say much about the actor. For example, actors that help other actors, waste less resources, provide creative ideas, or take responsibility should be also rewarded. In such cases, the subjective opinions of other relevant actors (i.e., *peers*) can be used to quantify these elusive actor qualities.

In order to acquire the rewarding metrics, collaborative systems use different *evaluation methods*, relying both on human and machine actors:

- Individual evaluation methods
  - *Quantitative methods* – They represent a quantitative measurement of an individual actor's contribution as measured by the system itself. Such metrics can represent the number of processed tasks, average speed, responsiveness, acceptance rate, etc. These methods are considered fair and cheap to implement, but unfortunately they are applicable only in cases where actors work on easily quantifiable tasks.
  - *Subjective methods* – In cases where the quality of work is a property understandable to humans only, a quantitatively expressed subjective assessment by a human actor replaces a quantitative metric measured by the system itself. This is the case with artistic, design, and engineering tasks. The advantages are the simplicity and cost, but a serious drawback is the inevitable lack of objectivity.
- Group evaluation methods
  - *Peer evaluation methods* – They are used to express an aggregated opinion of an interest group. The members of evaluation group usually express their votes by scoring tasks or actors on a fixed scale or by investing amounts of virtual credits expressing their confidence (placing bets). The quality and effectiveness of these methods are influenced by the size of the composition of the evaluation group.
  - *Indirect evaluation methods* – In certain situations, human actors can be evaluated by comparing the status of the artifacts they previously produced with the status of the artifacts produced by other members of the same community. The artifacts can be Web pages, projects, articles, photos, and programming code. These comparisons are usually performed with the help of sophisticated algorithms. An example is the Google's PageRank algorithm, impact

factor for scientific publications, or Klout's algorithm for measuring social network influence. Advantages and disadvantages of these methods are dependent on the properties of the algorithm.

Key Applications

Taken individually, many of the described techniques and algorithms have found practical used in today's systems: Structural and interaction metrics are used in social network platforms and recommender systems for determining trusted groups, which in turn are used for improving the quality of recommendations. The same techniques are at the same time a useful tool for sociological, ethnological, medical, and forensic research. The described evaluation methods are used in software development industry and crowdsourcing platforms, where a nonautomated evaluation would be impractical due to the scale of the collaborative effort. The task assignment and delegation patterns are used in workflow management systems, which are standard tools for automating complex or critical business processes in medium and large companies. Many of the team formation algorithms are adapted from the algorithms originally used in service-oriented architectures for service composition.

The wish of the authors of this article was to draw attention to the prospective benefits that these techniques can bring when used together in the context of emerging collaborative systems. These systems (e.g., Scekic et al. 2015; Minder and Bernstein 2012; Tranquillini et al. 2015) need to manage the entire cycle of human participation and will thus require in future Kittur et al. (2013) application of many of the presented methods and metrics and will hopefully drive the development of novel ones.

Future Directions

Although a considerable amount of work is done in the area of interaction analysis in social networks, there is much less work conducted on team-based metrics and analysis. Many open questions still remain to be tackled. Some of them are (i) understanding the interdependencies between metrics for better analysis of different collaboration systems, testing, and evaluating these team-based metrics and (ii) utilizing these metrics in the most appropriate way for task adaptation. Another future research direction in team collaboration in mixed systems is to develop metrics that can be used to compare human and software-based actors.

Cross-References

- ▶ Analyzing Link Dynamics in Scientific Collaboration Networks A Social Yield Based Perspective
- ▶ Behavior Modeling in Social Networks
- ▶ Collaboration Patterns in Software Developer Network
- ▶ Collective Intelligence, Overview
- ▶ Community Detection and Recommender Systems
- ▶ Community Detection in Social Network: An Experience with Directed Graphs
- ▶ Community Detection, Current and Future Research Trends
- ▶ Extracting and Inferring Communities via Link Analysis
- ▶ Incentives in Collaborative Applications
- ▶ Modeling Social Behavior
- ▶ Modeling Social Preferences Based on Social Interactions
- ▶ Network Structure Analysis
- ▶ Recommender Systems based on Social Networks
- ▶ Recommender Systems in Crowd Sourcing
- ▶ Reputation Systems
- ▶ Rewarding
- ▶ Social Groups in Crowd
- ▶ Social Influence Analysis
- ▶ Web Service Composition
- ▶ Web Service Infrastructure Patterns

References

An A, Kargar M, ZiHayat M (2013) Finding affordable and collaborative teams from a network of experts. In: Proceedings of the 13th SIAM international conference on data mining, Austin, 2–4 May 2013, pp 587–595. doi:10.1137/1.9781611972832.65. http://dx.doi.org/10.1137/1.9781611972832.65

Anagnostopoulos A, Becchetti L, Castillo C, Gionis A, Leonardi S (2010) Power in unity: forming teams in large-scale community systems. In: Huang, Jimmy & Koudas, Nick & Jones, Gareth J. F. & Wu, Xindong & Collins-Thompson, Kevyn and An, Aijun. CIKM, pp 599–608

Anagnostopoulos A, Becchetti L, Castillo C, Gionis A, Leonardi S (2012) Online team formation in social networks. In: Proceedings of the 21st international conference on World Wide Web-WWW '12. ACM Press, New York, p 839. doi:10.1145/2187836.2187950. http://dl.acm.org/citation.cfm?id=2187836.2187950

Barowy D, Berger E (2012) AUTOMAN: A platform for integrating human-based and digital computation. http://www.cs.umass.edu/~emery/pubs/AutoMan-UMass-CS-TR2011-44.pdf

Caverlee J, Liu L, Webb S (2008) Socialtrust: tamper-resilient trust establishment in online communities. In: Proceedings of the 8th ACMIEEECS joint conference on digital libraries. ACM, pp 104–113. http://portal.acm.org/citation.cfm?id=1378889.1378908

Caverlee J, Cheng Z, Eoff B, Hsu CF, Kamath K, Kashoob S, Kelley J, Khabiri E, Lee K (2010) SocialTrust++: building community-based trust in social information systems. In: The 6th international conference on collaborative computing: networking, applications and worksharing, CollaborateCom 2010, Chicago, 9–12 Oct 2010, pp 1–7. doi:10.4108/icst.collaboratecom.2010.40

de La Robertie B, Pitarch Y, Teste O (2015) Measuring article quality in wikipedia using the collaboration network. In: Proceedings of the 2015 IEEE/ACM international conference on advances in social networks analysis and mining 2015, ASONAM 15. ACM, New York, pp 464–471. doi:10.1145/2808797.2808895. http://doi.acm.org/10.1145/2808797.2808895

Dorn C, Dustdar S (2010) Composing near-optimal expert teams: a trade-off between skills and connectivity. In: On the move to meaningful Internet systems: OTM 2010, pp 472–489. http://www.springerlink.com/index/H434570G12787H57.pdf

Dustdar S, Bhattacharya K (2011) The social compute unit. IEEE Internet Comput 15(3):64–69. doi:10.1109/MIC.2011.68. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5755601

Dustdar S, Gaedke M (2011) The social routing principle. IEEE Internet Comput. Vol. 15, No. 4. pp. 80–83, doi:10.1109/mic.2011.97

Espinosa JA, Slaughter SA, Kraut RE, Herbsleb JD (2007) Familiarity, complexity, and team performance in geographically distributed software development. Organ Sci 18(4):613–630. doi:10.1287/orsc.1070.0297. http://dx.doi.org/10.1287/orsc.1070.0297

Fazel-Zarandi M, Devlin HJ, Huang Y, Contractor N (2011) Expert recommendation based on social drivers, social network analysis, and semantic data representation. In: Proceedings of the 2nd international workshop on information heterogeneity and fusion in recommender systems, HetRec 11. ACM, New York, pp 41–48. doi:10.1145/2039320.2039326. http://doi.acm.org/10.1145/2039320.2039326

Gaston ME, DesJardins M (2005) Agent-organized networks for dynamic team formation. In: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems – AAMAS '05, p 230. doi:10.1145/1082473.1082508. http://portal.acm.org/citation.cfm?doid=1082473.1082508

Grudin J (1994) Computer-supported cooperative work: history and focus. Computer 27(5):19–26. doi:10.1109/2.291294. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=291294

Hu M, Lim EP, Sun A, Lauw HW, Vuong BQ (2007) Measuring article quality in wikipedia: models and evaluation. In: Proceedings of the sixteenth ACM conference on conference on information and knowledge management, CIKM 07. ACM, New York, pp 243–252. doi:10.1145/1321440.1321476. http://doi.acm.org/10.1145/1321440.1321476

Kasunic M (2008) A data specification for software project performance measures: results of a collaboration on performance measurement. Technical report, Carnegie Mellon University, Software Engineering Institute. http://books.google.at/books?id=Un3NSgAACAAJ

Kittur A, Kraut RE (2008) Harnessing the wisdom of crowds in wikipedia. In: Proceedings of the ACM 2008 conference on computer supported cooperative work – CSCW '08. ACM Press, New York, San Diego, pp 37–46. doi:10.1145/1460563.1460572. http://portal.acm.org/citation.cfm?id=1460563.1460572; http://portal.acm.org/citation.cfm?doid=1460563.1460572

Kittur A, Nickerson JV, Bernstein M, Gerber E, Shaw A, Zimmerman J, Lease M, Horton J (2013) The future of crowd work. In: Proceedings of the 2013 conference on computer supported cooperative work, CSCW '13. ACM, pp 1301–1318. doi:10.1145/2441776.2441923

Kloppmann M, Koenig D, Leymann F, Pfau G, Rickayzen A, Schmidt P, Trickovic I (2005) WS-BPEL extension for people (July):1–18. http://public.dhe.ibm.com/software/dw/specs/ws-bpel4people/BPEL4People_white_paper.pdf

Lappas T, Liu K, Terzi E (2009) Finding a team of experts in social networks. In: Proceedings of the 15th ACM international conference on knowledge discovery and data mining, vol 7120(4). p 467. http://portal.acm.org/citation.cfm?doid=1557019.1557074

Law E (2011) Defining (human) computation. In: Proceedings of the Workshop on Crowdsourcing and Human Computation, in conjunction with CHI

Lee CP, Paine D (2015) From the matrix to a model of coordinated action (MoCA): a conceptual framework

of and for CSCW. In: Proceedings of the 18th ACM conference on computer supported cooperative work & social computing, CSCW 15. ACM, New York, pp 179–194. doi:10.1145/2675133.2675161. http://doi.acm.org/10.1145/2675133.2675161

Little G, Chilton LB, Miller R, Goldman M (2009) TurKit: tools for iterative tasks on mechanical turk. IEEE. doi:10.1109/VLHCC.2009.5295247. http://ieeexplore.ieee.org/ lpdocs/epic03/wrapper.htm?arnumber=5295247

Little G, Chilton LB, Goldman M, Miller R (2010) Exploring iterative and parallel human computation processes. In: Proceedings of the 28th of the international conference extended abstracts on human factors in computing systems – CHI EA '10, p 4309. doi:10.1145/1753846.1754145. http://portal.acm.org/citation.cfm?doid=1753846.1754145

Lopez M, Vukovic M, Laredo J (2010) PeopleCloud service for enterprise crowdsourcing. In: 2010 I.-E. International conference on services computing, pp 538–545. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5557275

Marcus A, Wu E, Karger DR, Madden S, Miller RC (2011) Platform considerations in human computation. In: Workshop on crowdsourcing and human computation

Marsh SP (1994) Formalizing trust as a computational concept. University of Stirling, Stirling

Minder P, Bernstein A (2012) Crowdlang: a programming language for the systematic exploration of human computation systems. In: Aberer K, Flache A, Jager W, Liu L, Tang J, Guret C (eds) Social informatics, LNCS, vol 7710. Springer, Berlin/Heidelberg, pp 124–137

Newman MEJ (2010) Networks: an introduction. Oxford University Press, New York, NY, USA

Noorian Z, Fleming M, Marsh S (2012) Preference-oriented QoS-based service discovery with dynamic trust and reputation management. In: Proceedings of the 27th annual ACM symposium on applied computing – SAC '12, p 2014. doi:10.1145/2245276.2232111. http://dl.acm.org/citation.cfm?doid=2245276.2232111

Parameswaran A, Polyzotis N (2011) Answering queries using humans, algorithms and databases, Tech. rep. Stanford University. http://ilpubs.stanford.edu:8090/986/

Quinn AJ, Bederson BB (2011) Human computation: a survey and taxonomy of a growing field. In: Proceedings of the 2011 annual conference on human factors in computing systems – CHI '11. ACM Press, New York, p 1403. http://dl.acm.org/citation.cfm? id=1978942.1979148

Riveni M, Truong HL, Dustdar S (2014) On the elasticity of social compute units. In: Jarke M, Mylopoulos J, Quix C, Rolland C, Manolopoulos Y, Mouratidis H, Horkoff J (eds) Advanced information systems engineering. Lecture notes in computer science, vol 8484. Springer International Publishing, Cham, Switzerland, pp 364–378 http://dx.doi.org/10.1007/978-3-319-07881-6_25

Riveni M, Truong HL, Dustdar S (2015) Trust-aware elastic social compute units. In: Trust- com/BigDataSE/ISPA, IEEE, vol 1, pp 135–142

Rovatsos M, Diochnos D, Craciun M (2015) Advances in social computing and multiagent systems. In: 6th International workshop on collaborative agents research and development, CARE 2015 and Second international workshop on multiagent foundations of social computing, MFSC 2015, Istanbul, Turkey, May 4, 2015, Revised Selected Papers. Springer International Publishing, Cham, chap Agent Protocols for Social Computation, pp 94–111. doi:10.1007/978–3–319-24804-2_7

Sagar AB (2012) Modeling collaborative task execution in social networks. In: Potdar V, Mukhopadhyay D (eds) CUBE. ACM, pp 664–669. http://dblp.uni-trier.de/db/conf/cube/ cube2012.html#Sagar12

Scekic O, Schiavinotto T, Diochnos DI, Rovatsos M, Truong HL, Carreras I, Dustdar S (2015) Programming model elements for hybrid collaborative adaptive systems. In: 2015 I.E. Conference on Collaboration and Internet Computing (CIC), pp 278–287. doi:10.1109/CIC.2015.17

Schall D, Dustdar S (2010) Dynamic context-sensitive PageRank for expertise mining. In: Proceedings of the second international conference on social informatics, Springer, Berlin/Heidelberg, pp 160–175. http://dl.acm.org/citation.cfm?id=1929326.1929338

Schall D, Skopik F, Dustdar S (2012) Expert discovery and interactions in mixed service-oriented systems. IEEE Trans Serv Comput 5(2):233–245. doi:10.1109/TSC.2011.2. http://doi.ieeecomputersociety.org/10.1109/TSC.2011.2; http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5710867

Skopik F, Schall D, Dustdar S (2009) The cycle of trust in mixed service-oriented systems. In: 35th Euromicro conference on software engineering and advanced applications, pp 72–79. doi:10.1109/SEAA.2009.20. http://ieeexplore.ieee.org/lpdocs/epic03/ wrapper.htm?arnumber=5349860

Skopik F, Schall D, Dustdar S (2010) Modeling and mining of dynamic trust in complex service-oriented systems. Inf Syst 35(7):735–757. doi:10.1016/j.is.2010.03.001. http://linkinghub.elsevier.com/retrieve/pii/S0306437910000153

Sun H, Srivatsa M, Tan S, Li Y, Kaplan LM, Tao S, Yan X (2014) Analyzing expert behaviors in collaborative networks. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, KDD 14, ACM, New York, pp 1486–1495. doi:10.1145/2623330.2623722. http://doi.acm.org/10.1145/2623330.2623722

Tranquillini S, Daniel F, Kucherbaev P, Casati F (2015) Modeling, enacting, and integrating custom crowdsourcing processes. ACM Trans Web 9(2):1–43

Truong HL, Dustdar S (2009) Online interaction analysis framework for ad-hoc collaborative processes in SOA-based environments. Framework 260–277. doi:10.1007/978-3-642-00899-3_15. http://www.springerlink.com/index/u1075446t8qr727q.pdf

van der Aalst WMP (2011) Process mining. Springer, Berlin/Heidelberg. doi:10.1007/978–3–642-19345-3. http://www.mendeley.com/ research/no-title-avail/; http://www.sciencedirect.com/science/article/pii/ S0166361503001945; http://www.springerlink.com/index/10.1007/978-3-642-19345-3

Zhang P, Serban N (2007) Discovery, visualization and performance analysis of enterprise workflow. Comput Stat Data Anal 51(5):2670–2687. doi:10.1016/j.csda.2006.01.008 http://linkinghub.elsevier.com/retrieve/pii/S0167947306000132