The Network-Based Business Process

Ejub Kajan • State University of Novi Pazar, Serbia
Noura Faci • Université Lyon 1, France
Zakaria Maamar • Zayed University, UAE
Alfred Loo • Lingnan University, Hong Kong
Aldina Pljaskovic • State University of Novi Pazar, Serbia
Quan Z. Sheng • University of Adelaide, Australia

Business processes can be structured or unstructured. The former are well formed, often repetitive, and call for intense coordination among stakeholders. The latter are loosely defined and utilize stakeholders’ personal experiences and social interactions. Here, the authors demystify the social interactions on which network-based business processes are developed, structure them for better tracking, and analyze the value they add to enterprise operations.

Today, “bailout,” “inflation,” and “recession” are terms that people frequently hear and read about. Enterprises (and even certain governments) are caught in the middle of a major financial crisis that is jeopardizing their growth, profit, and sometimes survival. To respond to this crisis, many enterprises have initiated various “lifting” initiatives aimed at improving their business processes, aligning their development strategies with market needs, and strengthening their core competencies. The severity of the crisis requires that enterprise executives be extremely proactive by relying quite often on personal contacts and informal sources when making decisions.

Parallel to the financial crisis, enterprises are also getting the message about the value Web 2.0 adds to their operations: “Enterprise spending on Web 2.0 technologies will grow strongly over the next five years, reaching US$4.6 billion globally by 2013, with social networking, mashups, and RSS capturing the greatest share” (www.forrester.com/Global+Enterprise+Web+20+Market+Forecast+2007+To+2013/fulltext/-/E-RES43850?docid=43850). Practically speaking, executives should be aware of how social software (built on Web 2.0 technologies) could shape their business practices, as in the case of using crowd-sourcing to tap into external expertise.1,2 (See the “Social Software in Brief” sidebar for more about this trend.)

Before we can measure the value social software adds, however, we must demystify the social relations that arise from the interactions among a business process’s three components — task, person, and machine. A task is a work unit that, with other tasks, constitutes a business process and is assigned to machines (hardware or software) or people for execution. An inability to identify appropriate social relations among these components mitigates the role social software can play in enterprises. Gartner reports that some 80 percent of social business software projects won’t achieve their intended benefits through 2015 (www.computerworld.com/s/article/9236323).

Simply put, a business process is “a set of logically related tasks performed to achieve a defined business outcome.”3 Although the relations (or dependencies) between tasks are critical to the
Although social software might sound simple to define, no consensus exists in the literature on a common definition. According to Selim Erol and colleagues, the roots of social software can be traced back to the 1940s. The authors note that “impressive results are created without a central plan or organization. Instead, social software uses a self-organization and bottom-up approach where interaction is coordinated by the ‘collective intelligence’ of the individuals; the latter do not necessarily know each other and are a priori not organized in a hierarchy.” For Vitaliy Liptchinsky and colleagues, social software “fosters collaboration of individuals who work across time, space, cultural, and organizational boundaries.” People engage in conversations and transactions so that common deliverables are produced promptly and with minimum conflict. For Rainer Schmidt and Selmin Nurcan, social software supports interaction and production using computers and networks. Last but not least, Giorgio Bruno and colleagues propose four properties of social software: weak ties are spontaneously established contacts that create new views on problems and allow competency combination; social production breaks with the paradigm of centralized a priori production planning and promotes unforeseen and innovative contributors and contributions; egalitarianism abolishes hierarchical structures, merges contributors’ and consumers’ roles, and introduces a culture of trust; and mutual service provisioning changes the cooperation model from a client-server one to a model based on exchanging services.

The following events provide more information on social software: the Workshop on Business Process Management and Social Software (BPMS), held with the International Conference on Business Process Management (BPM); and the Workshop on Business Applications of Social Networks Applications (BASNA), held with the IEEE/ACM Conference on Advances in Social Networks Analysis and Mining (ASONAM).

### References

### Example Scenario
Purchase-order business processes consist of multiple tasks: submit order, manage customer, prepare bill, check inventory, ship product, and request supply. When a customer places an order for products, a staff member from the purchasing department extracts the customer’s details from a data repository to verify discount eligibility prior to finalizing the bill. After verification, she sends a detailed purchase order to inventory and the bill to the customer, if any downpayment is required. Products that are in stock are shipped to the customer. Otherwise (not in stock), the staff member contacts suppliers and notifies the customer as well.

Often, unforeseen events disturb this process’s normal completion — for instance, an accountant calling in sick, a last-minute strike of customs officials, or the inventory system going down. To ensure that the process completes, operations managers must be able to determine alternatives based on previous, similar experiences, and answer several possible questions. Who substituted for the accountant last time and how did that person perform? Did the accountant delegate her tasks to a colleague due to a last-minute meeting request? Which tasks provide the same results and are hence interchangeable? What impact does replacing a failing machine have on other machines? We address all these questions in the following discussion. In line with these questions, Schahram Dustdar and Kamal Bhattacharyya point out “the huge gap between business process management technologies, usage patterns, and workflows on the one hand, and social computing as it is known today.”

### Developing Network-Based Business Processes
Contrary to existing approaches that use specific notations to give business
processes a social flavor (such as broadcasts and posting, or tags and comments), we demystify the social relations that exist between tasks and machines using an analogy to those that exist between people. Social relations are different from execution relations that bind tasks to people or machines (as when accountant executes prepare bill, or inventory-app executes check inventory). We might consider using tasks’ input and output dependencies to identify social relations, but such dependencies are inappropriate; they are meant only for data exchange.

In preparation for demystifying social relations, we associate tasks with requirements (for example, request supply must occur online) and people and machines with capacities (or, alternatively, capabilities) for instance, person X approves US$5,000+ bills, and crane-233 lifts loads up to 500 tons. Requirements impose restrictions on those who execute tasks in terms of execution type (such as manual), necessary authorization level (people), reliability level (machines), and so on. In addition to requirements, we refer to a task as self-contained when its output doesn’t require any additional processing by another task. Additional elements that make a task self-contained are listed elsewhere. Unless stated, a task is by default self-contained. Compress data is not a self-contained task; decompression is necessary for the compressed data to be used at a later stage. Business-process engineers assign tasks to executors by matching requirements to capacities. Further discussion of the topic is out of this article’s scope, but a good technique to ensure this matching can be based on crowdsourcing, as discussed elsewhere.

We identify and analyze relations between tasks, machines, and people from two perspectives: execution and social (our focus). In particular, each social relation is the basis of a specialized network that will fall into one of the following categories: configuration (tasks), support (machines), and social (people).

Tasks
From an execution perspective, relations (that is, dependencies) between tasks (t_i and t_j) are well established in the literature, including prerequisite, parallel prerequisite, and parallel. To deal with tasks that aren’t self-contained, we propose completion as an extra execution relation; t_i and t_j engage in a completion relation when t_i isn’t self-contained and needs t_j to process its output.

From a social perspective, we consider two relations.

Interchange. t_i (submit order online) and t_j (submit order by fax) engage in an interchange relation when both produce similar output (order received) with respect to similar input received for processing, and their requirements (that is, Web versus fax) don’t overlap. The non-overlap condition avoids blockage in cases where t_i’s requirements aren’t satisfied. Thus, t_i interchanges with t_j because their requirements are different. In terms of benefits, interchange indicates how difficult it can be to satisfy a task’s requirements if it’s replaced continuously. For a business-process engineer, difficulty would arise given a lack of executors with appropriate capacities (such as an unreliable website).

Coupling. t_i (submit order) and t_j (manage customer) engage in a coupling relation when they interact in the same business processes through an execution relation (excluding completion). In terms of benefits, coupling indicates how strong or weak the connection between tasks is, which should help engineers recommend tasks during business-process design. For a business-process engineer, strong coupling ensures smooth interaction between tasks (for instance, less data-semantic conflicts to address), but could restrict opportunities for finding replacement tasks.

Machines
Machines (m_i, whether hardware or software) execute automated tasks. From an execution perspective, we identify relations between m_i and m_j using Keith Decker and Victor Lesser’s relations (namely enables, facilitates, cancels, constrains, inhibits, and causes), which address coordination problems between tasks. Two of these relations connect machines assigned the same task: Enablement is established when m_i produces (internal) output that lets m_j continue task execution (for example, deliver product, requiring traffic system to report on traffic status and truck to ensure transportation). Inhibition is established when m_j executes its part of the task after a certain time has elapsed since m_i executed its part (for instance, inspect goods after a 48h quarantine in warehouse). When connecting separate machines, the execution relations between tasks automatically impose a chronology on these machines. Consequently, business-process engineers don’t need to identify dedicated execution relations between separate machines.

From a social perspective, we consider three relations (some authors use the terminology of social machines).

---

**Strong coupling ensures smooth interaction between tasks, but could restrict opportunities for finding replacement tasks.**

---

*The Network-Based Business Process*
Backup. $m_i$ (a scanner) and $m_j$ (a three-function printer) engage in a backup relation when both have similar capacities (the subsumption relation can also be used, but ontology use is necessary). In terms of benefits, backup indicates how reliable a machine is and what machines are frequent backups. For a business-process engineer, concern over reliability can trigger changes in machines prior to confirming their task assignment.

Cooperation. $m_i$ and $m_j$ engage in a cooperation relation when both are already engaged in a backup relation (in terms of similar capacities), and combining their respective capacities is necessary to satisfy a task’s requirements (for instance, several cranes to lift heavy items). In terms of benefits, cooperation indicates how often similar machines work together on tasks, which should help recommend machines when business-process engineers carry out the matching. Because the machines are different, their collective performance must consider individual performance and functional constraints. For a business-process engineer, concern over collective performance can trigger changes in machines prior to confirming their task assignment.

Partnership. $m_i$ (a crane) and $m_j$ (a truck) engage in a partnership relation when their capacities are complementary, and their respective capacities combined are necessary to satisfy a task’s requirements. In terms of benefits, partnership indicates how often separate machines work together on tasks, which should help recommend machines when business-process engineers carry out the matching. Because the machines are different, their collective performance must consider each machine’s specificities as regards individual performance and functional constraints. For a business-process engineer, concern over collective performance can trigger changes in machines prior to confirming their task assignment.

People

In enterprises, people ($p_i$) make decisions, initiate processes, and so on. From an execution perspective, we adopt the same execution relations between machines, namely, enablement (load product requiring crane operator and supervisor) and inhibition.

From a social perspective, we consider the following three relations.

Substitution. $p_i$ and $p_j$ engage in a substitution relation when both have similar capacities (we can also consider a subsumption relation if we use an ontology). In terms of benefits, substitution indicates how available a person is and which people are frequent substitutes. For a business-process engineer, concern over availability can trigger changes in people prior to confirming their task assignment.

Delegation. $p_i$ and $p_j$ engage in a delegation relation when both are already engaged in a substitution relation, and $p_i$ assigns a task that she will execute or is executing to $p_j$ due to unexpected changes in status — for instance, calling in sick or risking overload (truck dispatcher handles the job of truck driver due to last-minute changes). In terms of benefits, delegation indicates the transfer of work between people, which could help identify the right people next time unexpected changes happen. Crowdsourcing also helps achieve this identification. For a business-process engineer, concern over continuity drives the prompt availability of delegates.

To illustrate, the following equation assesses the weight of an edge in a delegation network, where $T_{\text{del}}^{p_i}$ is the set of all tasks assigned to $p_i$ but then delegated due to her unexpected unavailability, and $\text{delegateSuc}_{T_{\text{del}}^{p_i}(p_i,p_j)}$ is the number of tasks in $T_{\text{del}}^{p_i}$ that $p_i$ delegates to $p_j$:

$$W_{\text{delegation}_{p_i,p_j}} = \frac{\text{delegateSuc}_{T_{\text{del}}^{p_i}(p_i,p_j)}}{T_{\text{del}}^{p_i}}.$$ 

Peering. $p_i$ and $p_j$ engage in a peering relation when $p_i$ and $p_j$ are with different organizational units, and their respective similar (managerial) and complementary (expertise) capacities are necessary to meet a task’s requirements. In terms of benefits, peering fosters cross-organizational collaboration and indicates how often people work together on common tasks, which should help recommend people when performing matching. For a business-process engineer, concern over appropriate peering among people helps reduce conflicts and ensure better homogeneity between peers.

Table 1 compares substitution to delegation in terms of who executes what and who reports to whom.

Table 2 summarizes the social relations between tasks, machines, and people along with their respective pre-condition, condition, and postcondition. Precondition defines the rationale for establishing a social relation between
Table 2. Summary of social relations.

<table>
<thead>
<tr>
<th>Entities involved</th>
<th>Social relation types</th>
<th>Preconditions</th>
<th>Conditions</th>
<th>Postconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i, t_j$</td>
<td>Coupling</td>
<td>Participated in joint business processes</td>
<td>Review of business process design or concern over coupling level</td>
<td>Business-process design completion or coupling level satisfaction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$t_j$ lacks executor who satisfies its requirements</td>
<td>Executor found for $t_j$</td>
<td></td>
</tr>
<tr>
<td>$m_i, m_j$</td>
<td>Backup</td>
<td>Have similar capacities</td>
<td>$m_i$ has unexpected failure or concern over $m_i$ reliability</td>
<td>Backup/replacement machine found for $m_i$</td>
</tr>
<tr>
<td></td>
<td>Cooperation</td>
<td>Have similar capacities</td>
<td>Concern over machine collective performance</td>
<td>Collective performance-level satisfaction</td>
</tr>
<tr>
<td></td>
<td>Partnership</td>
<td>Have complementary capacities</td>
<td>Concern over machine collective performance</td>
<td>Collective performance-level satisfaction</td>
</tr>
<tr>
<td>$p_i, p_j$</td>
<td>Substitution</td>
<td>Have similar capacities</td>
<td>$p_i$ expected unavailability (for example, annual leave or sick leave) or concern over $p_i$ availability</td>
<td>Substitute found for $p_i$</td>
</tr>
<tr>
<td></td>
<td>Delegation</td>
<td>Have similar capacities</td>
<td>$p_i$ unexpected unavailability (called in sick, urgent task to complete, or risk of overload)</td>
<td>Delegate found for $p_i$</td>
</tr>
<tr>
<td></td>
<td>Peering</td>
<td>Have similar or complementary capacities</td>
<td>Concern over peering appropriateness</td>
<td>Peer found for either $p_i$ or $p_j$</td>
</tr>
</tbody>
</table>

Figure 1. System overview. Business-process engineers, users, and analysts use the system, respectively, to design business processes, execute them, and mine networks that contain details on how executions occurred.
components. Condition indicates when a network built on a social relation is used to solve conflicts that prevent a business process from completing. Finally, postcondition indicates when to stop using a network.

Showcasing a Network-Based Business Process

Figure 1 shows our system, which extends the Yaoqiang BPMN Editor (www.sourceforge.net/projects/bpmn) to handle operations such as assigning tasks to executors, defining tasks as either self-contained or not, and recommending tasks for interchange. The system analyzes the following example scenarios:

- replacement, with ship-product task flagged in red due to a last-minute railway strike, so ship-product-by-air task is recommended;
- enhancement, with prepare-bill task flagged in yellow, so bill-convert task is recommended; and
- delay, with request-supply task flagged in green due to call-in-sick, so a delegate is recommended.

These recommendations are extracted from the networks of tasks, people, and machines, respectively.

We focus hereafter on two specific scenarios: load balancing among four people \(\{p_1, p_2, p_3, p_4\}\) and analyzing their manpower. Figures 2a and 2d show the delegation-driven data structures for these scenarios. We populated two tables known as delegation and acceptance level with details generated randomly at start-up, but which will change over time. The delegation table tracks a person’s previous delegation experiences in terms of either success or failure. Delegation requests are assigned to people with 

\[
\left( w_{ij} \text{ that is, } w_{delegation,p_i,p_j} \right)
\]

in the delegation network, at the risk of overloading them. The acceptance-level table processes the combined weights \((cw_j)\) of people acting as delegates. Figure 2b shows an acceptance threshold of 0.1.

Load balancing ensures that the same people don’t continuously act as delegates; for example, \(p_2\) in Figure 2d is considered a good delegate.
On receiving the next purchase order, the system directs all delegation requests to $p_2$ (Figure 2e). Obviously, this increases his or her acceptance level, but isn’t “healthy” from a productivity perspective. To address this issue, we temporarily exclude anyone with an acceptance level higher than a given threshold and find other appropriate delegates (that is, the person with the next-highest edge weight). Figures 2c and 2f identify the moment when $p_2$ is excluded temporarily as a delegate.

Manpower analysis oversees productivity when specific events happen, such as $p_1$ being sick with $p_2$ still excluded. Figures 2h and 2i show that $p_1$ and $p_4$ took over the tasks initially assigned to $p_1$ and $p_2$. Independently of how $p_1$ and $p_4$ perform, they can’t handle all the tasks alone. Thus, continuously monitoring business-process execution lets managers take action (such as adjust the load balancing threshold or hire temp workers) when a manpower bottleneck (no executors available in the networks) occurs.

Today’s economic and political contexts pose new challenges to those who make decisions by relying on personal contacts and unstructured information sources such as social networks. There is no doubt that enterprise executives’ reliance on social networks raises concerns over drawing the line between professional life and social life.

References

Acknowledgments
Zakaria Maamar’s work was partially supported by the ZU Research Incentive Funds Program (grant no. R12045).

Ejub Kajan is an assistant professor in the Department of Technical Sciences at the State University of Novi Pazar, Serbia. His research interests include e-business, interoperability, Web 2.0, Semantic Web, Web services, and decision support systems. Kajan received a PhD in computer science from the University of Niš, Serbia. Contact him at ekajan@ieee.org.

Noura Faci is an associate professor in the Department of Computer Science at Université Lyon 1, France. Her research interests include social networks, dependable e-business systems, and service-oriented computing. Faci received a PhD in computer science from Reims University, France. Contact her at noura.faci@univ-lyon1.fr.

Zakaria Maamar is a full professor in the College of Information Technology at Zayed University, UAE. His research interests include Web services, social networks, and context-aware computing. Maamar has a PhD in computer science from Laval University, Canada. Contact him at zakaria.maamar@zu.ac.ae.

Alfred Loo is an associate professor in the Department of Computing and Decision Sciences at Lingnan University, Hong Kong. His research interests are in distributed computing, wireless security, and peer-to-peer systems. Loo has a PhD in computing from the University of Sunderland, UK. Contact him at alfred@ln.edu.hk.

Aldina Pljaskovic is a PhD student in computer sciences in the Faculty of Electronic Engineering at the University of Niš and also a teaching assistant at the State University of Novi Pazar, both in Serbia. Her research interests include e-business, e-learning, computer graphics, and software engineering. Contact her at apljaskovic@np.ac.rs.

Quan Z. Sheng is an associate professor in the School of Computer Science at the University of Adelaide, Australia. His research interests include service-oriented architectures, Web of Things, distributed computing, and pervasive computing. Sheng has a PhD in computer science from the University of New South Wales, Australia. Contact him at michael.sheng@adelaide.edu.au.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.