EAI Endorsed Transactions
on Industrial Networks And Intelligent Systems

Research Article ICST.ORG

How to Make Business Processes "Socialize"?

Zakaria Maamar[1,*], Noura Faci[2], Ejub Kajan[3], Sherif Sakr[4], Mohamed Boukhebouze[5], and Ahmed Barnawi[6]

[1]Zayed University, Dubai, U.A.E
[2]Claude Bernard Lyon 1 University, Lyon, France
[3]State University of Novi Pazar, Novi Pazar, Serbia
[4]University of New South Wales, Sydney, Australia & King Saud bin Abdulaziz University for Health Sciences, Riyadh, Saudi Arabia
[5]CETIC, Charleroi, Belgium
[6]King Abdulaziz University, Jeddah, Saudi Arabia

Abstract

This paper presents an approach that builds upon social computing principles to make business processes "socialize". First the approach identifies the main components of a business process that are task, person, and machine. A task is a work unit that forms with other tasks a business process and that a person and/or machine execute. Afterwards the approach enriches a business process with details captured from the (execution and social) relations that connect tasks together, persons together, and machines together. While execution relations are widely reported in the literature, there is a growing interest in studying the role of social relations in business processes. The approach uses social relations to build configuration network of tasks, social network of persons, and support network of machines. These networks capture the ongoing interactions that arise when business processes are executed. A system illustrating how these networks are developed is also demonstrated in the paper.

Received on 02 May 2015; accepted on 29 July 2015; published on 17 September 2015

Keywords: Business process; Machine; Network; Person; Social relation; Task.

doi:10.4108/eai.17-9-2015.150284

1. Introduction

Social software, exemplified by Web 2.0 applications like social networks, Wikis, and blogs, has forced companies to review their ways of doing business. Many, if not all, companies have an online social presence so they can reach out to more customers, open up new communication channels with stakeholders, and also, showcase their embracement of the latest IT advances and gadgets [1]. Many companies recognize the need of rethinking their strategies and reevaluating their operation models as the world is getting more "social" [2].

Despite the Web 2.0 "fever", a recent study by Gartner reveals that *"...many large companies are embracing internal social networks, but for the most part, they're not getting much from them"* [3]. Social software does not work like an enterprise resource planning application where procedures are defined and employees are told to comply with these procedures. Employees' commitments (also participation in[1]) are critical to the success of social software, i.e., employees must opt-in rather than be forced [5]. On top of employees' commitments we argue that other elements contribute to this success, for instance (*i*) establishing guidelines and techniques to assist IT practitioners integrate social elements into business processes and (*ii*) demonstrating the social software's benefits through tangible results (e.g., number of new customers attracted because of a Facebook campaign).

*Corresponding author. Email: zakaria.maamar@zu.ac.ae

[1]Four factors drive user motivations to community contribution [4]: expectation of help in return, increase in positive reputation, sense of efficiency, and commitment to the community.

According to Gartner narrowing down the social-software view to social networks, only does not shed the light on other systems like business process management systems that exhibit some social aspects [2]. Various interactions take place during the execution of business processes, so we map some of these interactions onto specific social relations between these processes' components. We refer to these components as task, machine, and person. Indeed tasks are put together to form processes, persons collaborate together on complex tasks, and machines replace each other in the case of failure, are examples of social relations that business process management systems exhibit and hence, can be captured. While we acknowledge that tasks and machines cannot "socialize" (in the strict sense), combining tasks together and machines together presents some similarities with how people behave daily. Different initiatives already demonstrate the successful blend of social software with many disciplines such as learning [6], healthcare [7], and commerce [8]. Supporting our proposal of socializing tasks and machines, Tan et al. state that *"Currently, most social networks connect people or groups who expose similar interests or features. In the near future, we expect that such networks will connect* ***other entities, such as software components, Web-based services, data resources, and workflows****. More importantly, the interactions among people and nonhuman artifacts have significantly enhanced data scientists' productivity"* [9].

Our contributions in this paper include

1. Definition and specification of a set of social (execution as well) relations that permit to connect tasks together, persons together, and machines together in a business process;

2. Development of a set of networks built upon social relations; these networks are referred to as configuration for tasks, support for machines, and social for persons.

3. Demonstration through a proof-of-concept of configuration, support, and social networks development.

4. Brief discussion of the role of configuration, support, and social networks in addressing some issues that hinder business process execution.

The remainder of this paper is organized as follows. Section 2 is an overview of social software. A case study is presented in Section 3. Section 4 introduces the approach to manage social business processes. Prior to concluding in Section 6, a prototype system is discussed in Section 5.

2. Overview of social software

We report on what social software refers to and then, on some initiatives that blend social software with business processes. Dustdar and Bhattacharya stress 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"* [10].

In the literature there is not a common definition of social software. Warr states that *"social software includes a large number of tools used for online communication, e.g., instant messaging, text chat, internet fora, weblogs, Wikis, social network services, social guides, social bookmarking, social citations, social libraries, and virtual worlds"* [11]. For Schmidt and Nurcan, social software supports productivity by raising the level and scope of interactions because of the use of computers and networks [12]. Erol et al. note that the roots of social software can be traced back to the 40s and add 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 does not necessarily know each other and are a priori not organized in a hierarchy. Furthermore, social software follows a rather egalitarian approach; decisions are not made by small elites but by combining a multitude of inputs from different users"* [13]. For Liptchinsky et al., social software *"fosters collaboration of individuals who work across time, space, cultural, and organizational boundaries"* [14]. People engage in conversations and transactions so that common deliverables are produced promptly and with minimum of conflicts. Finally, Bruno et al. identify the four characteristics of social software [15]: (*i*) weak ties are spontaneously established contacts creating new views on problems and allowing competency combination, (*ii*) social production breaks with the paradigm of centralized a-priori planning of production and promotes unforeseen and innovative contributors and contributions, (*iii*) egalitarianism abolishes hierarchical structures, merges the roles of contributors and consumers, and introduces a culture of trust, and (*iv*) mutual service provisioning changes the cooperation model from a client-server model to a model based on exchanging services.

The blend of social software with business processes is reported throughout the literature. In [16], Rito Silva et al. describe the AGILIPO project that embeds social features into business process tools. The AGILIPO modeling and execution environment includes three roles known as executor, modeler, and developer that stakeholders take over. Executor carries out business processes either by making use of specified activities or by creating generic activities whenever the specified activities are not sufficient. Modeler changes the business process model by specifying new non-automated activities. Finally developer may consider automating the non-automated activities. To foster collaboration between these stakeholders, social software features such as tagging, versioning, comments, and rating are adopted. In [17], Brambilla et al. propose a specific notation to design social business processes. Social networking helps organizations harness the value of information relations and weak ties without compromising the consolidated business practices that are found in conventional business process management solutions. Despite these benefits there is a lack of appropriate notations that can be used to reflect social aspects on business process models. Brambilla et al.'s notation includes event and task types like broadcast, posting, and invitation to activity. In [18], Koschmider et al. demonstrate how social networks help enhance trust between users. Two networks are built upon a set of business processes and recommendations. The first network provides an organizational view of business processes by suggesting for instance, the average distance between performers who participated in existing business processes and those who are now participating in developing business processes. The second network shows the relations among modelers who use the recommendation system to build the business process model. In [19], Grim-Yefsah et al. reveal the existence of informal networks that people at work rely on to conduct their business. These networks perfectly coexist with regular networks where formal relations like supervision are reported. Grim-Yefsah et al. discuss how the "official" executor of a task informally seeks other persons' help in the organization known as contributors. The help takes different forms like asking for advices or confirming technical details. The informal networks back the work of regular networks and do not compete with them.

The aforementioned initiatives on blending social software with business processes develop different solutions such as tagging business processes, using social networks to enhance trust, and mixing formal and informal networks. However what social relations connect a business process's components, how business processes are adjusted in response to these relations, and what benefits these relations offer to companies, are left unanswered. In this paper we address the first question by showing how to connect tasks together, machines together, and persons together using social relations.

3. Case study

Our case study refers to the electronic-patient-folder system at Anderson Hospital that handles approximately 6000 annual inpatient admissions[2]. We leverage this system to identify first, some business processes' components (i.e., tasks, persons and machines) and second, the execution nature of some tasks. When a patient shows up at the hospital, the necessary documentation is scanned into a system known as ImageNow. Upon completion the patient's MEDITECH record is updated automatically. An advantage of this update is that different stakeholders like billing staff, coders, and other authorized people have immediate, electronic access to the necessary information instead of waiting for the paper documentation to arrive. Prior to implementing the new system Anderson Hospital faced different challenges such as paper records limit access to one user at a time and paper and manual processes hamper compliance with some healthcare standards.

We use this case study to shed the light on the social dimension of business processes by establishing relations between tasks, between machines, and between persons. It is common that events "disturb" the normal completion of business processes due to call-in-sick doctors, last-minute changes in surgery dates, appointment system failure, etc. When dealing with these events we would like to assist the hospital's managers in considering doctors' recommendations when looking for substitutes, interchanging tasks to avoid policy violation, and identifying machines based on their coupling level. These are some of the benefits that networks build upon relations between tasks, between machines, and between persons will provide to these managers.

4. Our approach to social business processes

We recall that a business process is "*a set of logically related tasks performed to achieve a defined business outcome*" [20]. We also recall that the execution of some processes is strictly confined into the borders of single units (e.g., finance department), while the execution of others crosses several independent units raising security, privacy, heterogeneity, and monitoring concerns among IT practitioners and end-users as well [21].

4.1. Overview

Our approach fosters the intertwining of the three components of a business process that are task, person, and machine (Fig. 1). The success of this intertwining depends on (*i*) identifying execution and

[2]`http://www.perceptivesoftware.com/pdfs/casestudies/psi_cs_anderson.pdf`

social relations between tasks (t), between executors (i.e., persons (p) and machines (m)), and between tasks and executors; and (ii) developing categories of networks upon these relations as per the characteristics of each component. These two points are thoroughly discussed in Sections 4.2 and 4.3, respectively.

Fig. 1 illustrates an example of business process for Anderson Hospital case-study. It includes multiple tasks such as t_1: scan documentation, t_2: update records, and t_i: prepare bill. Tasks connect to each other through input and output dependencies, e.g., patient's data from t_1 are sent to t_2 so that patient records are updated. However these dependencies are primarily meant for data exchange and thus, do not help much in enriching a business process with any social element nor in shedding the light on the potential social relations between this process's components. Fig. 1 also shows the execution nature of tasks. Some tasks are completely manual (p_j: cashier executing t_i) while others are either completely automated (m_2: ImageNow executing t_2) or semi-automated/semi-manual (p_1/m_1: operator/scanner taking turns in executing t_1).

In preparation for exposing the social dimension of business processes, we associate task with *requirements* (e.g., t_2: update records must be done within one hour of scan receipt), person with *capacities* (e.g., p_1: operate scanner), and machine with *capacities* as well (e.g., m_1: produce high-resolution scan). Requirements impose restrictions on those who will execute tasks in terms of execution nature (e.g., manual), necessary expertise level for persons, reliability level for machines, simultaneous involvement of persons and machines, etc. In addition to requirements we label task as *self-contained* when its output does not require any additional processing by another task. Additional elements that make a task self-contained are discussed in [22]. Encrypt data is an example of not-self-contained task since decrypting data for proper use is required at a later stage. Unless stated a task is by default self-contained. Task assignment to executors depends on matching requirements to capacities. However the matching does not fall into this paper's scope.

4.2. Relation identification

We identify relations between tasks, between persons, and between machines from two perspectives: Execution ($\mathcal{E}$) and Social ($\mathcal{S}$). *As stated earlier making tasks and machines "socialize" is backed by first, the relations between tasks and between machines that map perfectly onto relations between persons and second, Roush's statement that computing means connecting* [23]. We exemplify the proposed social and execution relations by the case study. It is worth noting that some relations between persons are appropriate for machines and *vice-versa*. This is primarily due to the semantics of these relations.

Relations between tasks. From an $\mathcal{E}$ perspective, execution relations (*aka* dependencies) between tasks are well defined in the literature [24] such as, *prerequisite* (e.g., t_1 and t_2), *parallel prerequisite* (e.g., t_2 and t_k: synchronize patient records), and *parallel* (e.g., t_i and t_j: check pending bills). To deal with not self-contained tasks, we propose *completion* as an additional execution relation between tasks (e.g., compress scan prior to archiving and then decompress scan upon request); t_i and t_j engage in a completion relation when t_i is not self-contained and needs t_j to process its output (e.g., compress and decompress).

From a $\mathcal{S}$ perspective, we propose two social relations:

- *Interchange*: t_i and t_j engage in an interchange relation when both produce similar output with respect to similar input received for processing and their requirements do not overlap (e.g., t_1 and t'_1: enter patient details manually in the case the scanner is down). The non-overlap condition is necessary to avoid blockage when t_i's requirements (e.g., online data entry) cannot be met due to lack of executors and thus, t_i needs to be interchanged with t_j that has different requirements (e.g., offline data entry). In terms of benefits, interchange indicates how difficult a task's requirements are satisfied if the task is constantly replaced and what tasks are frequently used as replacements.

- *Coupling*: t_i and t_j engage in a coupling relation when they interact in the same business processes through one of the aforementioned execution relations (excluding completion). In terms of benefits, coupling indicates how strong or weak the connection between tasks is, which should help recommend tasks when putting business processes together at design time.

Relations between machines. In companies, machines (e.g., scanner and ImageNow) ensure the performance of automated and semi-automated tasks. Each machine (m_i) is overseen by a dedicated software component for management purposes, but this is outside this paper's scope. As stated earlier, matching capacities to requirements identifies the necessary machines that will execute tasks.

From an $\mathcal{E}$ perspective, we identify execution relations between m_i and m_j by analyzing Decker and Lesser's six coordination relations between tasks [25]. These relations are *enables*, *facilitates*, *cancels*, *constrains*, *inhibits*, and *causes*, and only two are considered as per our needs of connecting machines together to execute joint tasks: (i) *enablement* is established when m_i produces (internal) output that allows m_j to continue

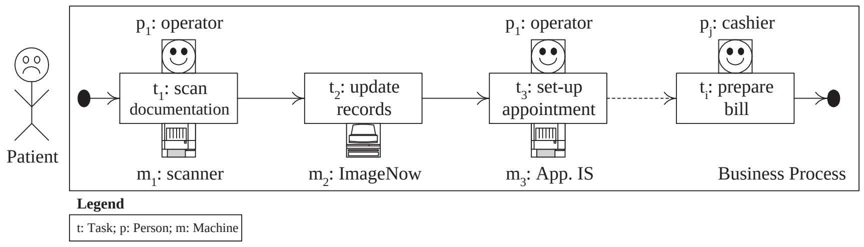

Figure 1. Business process’s components

the execution of the task (e.g., t_x: do chest Xray requiring m_x: Xray machine to take Xrays and m_y: communication app. to send the doctor Xrays), and (*ii*) *inhibition* is established when m_j executes its part of the task after a certain time elapses since m_i completed the execution of its part (e.g., t_x: do blood test where the results are sent by email some time later after blood collection and analysis using specialized equipment). It is worth mentioning that defining execution relations between independent machines in charge of dependent tasks is not relevant for our work. The previous execution relations between tasks in the $\mathcal{E}$-perspective impose automatically an execution chronology on the machines.

From a $\mathcal{S}$ perspective, we consider three social relations:

- *Backup*: m_i (e.g., scanner in main reception) and m_j (e.g., 3-in-1 printer in nurse station) engage in a backup relation when both have similar capacities (subsumption relation between capacities can also be considered using ontology). In terms of benefits, backup indicates how reliable a machine is and what machines are frequently requested as backups.

- *Cooperation*: m_i and m_j engage in a cooperation relation when both are already engaged in a backup relation (because of similar capacities) and the simultaneous combination of their respective capacities is necessary to meet a task’s requirements. In terms of benefits, cooperation indicates how often similar machines work together on executing tasks, which should help recommend machines when carrying out the matching. Though the machines are similar their collective performance might not correspond to the combination of their individual performances.

- *Partnership*: m_i and m_j engage in a partnership relation when their capacities are complementary and the simultaneous combination of their respective capacities is necessary to meet a task’s requirements (e.g., t_x: analyze blood sample requiring different types of machines). In terms of benefits, partnership indicates how often separate machines work together on tasks, which should help recommend machines when carrying out the matching. Since machines are different their collective performance needs to take into account the particularities of each in terms of individual performances and functional constraints, for example.

Relations between persons. In companies persons make decisions, trigger processes, exchange information, etc. As stated earlier, matching capacities to requirements identifies the necessary persons to execute tasks.

From an $\mathcal{E}$ perspective, we adopt the same execution relations for machines in order to connect p_i and p_j that are assigned joint tasks: (*i*) *enablement* is established when p_i performs other (internal) tasks whose completion allows p_j to continue the execution of the task (e.g., t_y: check patient requiring p_x: nurse to check patient’s vitals and p_y: doctor to consult patient and provide medication), and (*ii*) *inhibition* is established when p_j executes her part of the task after a certain time elapses since p_i completed the execution of her part (e.g., t_y: perform surgery requiring p_x: anesthetist and p_y: surgeon who intervenes after the anesthetic becomes effective).

From a $\mathcal{S}$ perspective, we consider three social relations:

- *Substitution*: p_i (e.g., general practitioner in family medicine) and p_j (emergency physician in emergency department) engage in a substitution relation when both have similar capacities (subsumption relation between capacities can also be considered using ontology). In terms of benefits, substitution indicates how available a person is and what persons are frequent substitutes.

- *Delegation*: p_i and p_j engage in a delegation relation when both are already engaged in a substitution relation (in term of capacity assessment) and p_i decides to assign a task that she will execute or is now executing to p_j due to unexpected changes in her status, e.g., call-in-sick or situation of overload (e.g., emergency physician transfers patient to general practitioner due to the arrival of an urgent case). In terms of benefits, delegation indicates the transfer-of-work between persons, which could help identify the right persons next time unexpected events arise.

- *Peering*: p_i and p_j engage in a peering relation when both are in different organizational units and their respective similar (e.g., managerial) and complementary (e.g., expertise) capacities are necessary to a meet a task's requirements. In terms of benefits, peering fosters cross-organization activities and indicates how often persons work together on common tasks, which should help recommend persons when the matching of capacities to requirements occurs.

Table 1. Substitution *versus* delegation

Substitution(p_i,p_j)	Delegation(p_i,p_j)
p_i stops executing ongoing tasks; p_j executes these tasks	p_i continues executing other ongoing tasks; p_j executes tasks assigned by p_i
p_j reports to whom p_i reports upon task completion; otherwise p_j waits for p_i return	p_j reports to p_i upon task completion

Table 1 compares substitution to delegation in terms of who does what and who reports to whom. Table 2 summarizes the social relations between tasks, between machines, and between persons along with their respective pre-conditions, conditions, and post-conditions. Pre-condition defines the rationale of a social relation between a process's components. Condition indicates when a network built upon a social relation is used so that solutions to conflicts that prevent a business process completion are addressed. Finally post-condition indicates the successful resolution of these conflicts so that network use is stopped. In Table 2 the occurrence of multiple relations (e.g., backup and cooperation) between the components of the same process is not exclusive if these relations' pre-conditions are simultaneously satisfied.

4.3. Network categorization

The social relations presented earlier are used for developing specialized networks, i.e., one network per relation. We group these networks into three categories: configuration network of tasks, support network of machines, and social network of persons. In this part of the paper we analyze the nodes and edges per network category and evaluate the edges that connect nodes.

Configuration network of tasks. In a configuration network of tasks, node and edge correspond to task and relation between tasks, respectively. We analyze both task and relation from $\mathcal{E}$ and $\mathcal{S}$ perspectives. On the one hand the $\mathcal{E}$ perspective sets the stage for connecting tasks together in order to form business processes and assigning tasks to executors. On the other hand the $\mathcal{S}$ perspective sets the stage for building networks of tasks based on Table 2's relations (pre-condition satisfaction) as well as using these networks when necessary (condition satisfaction).

Task. In the $\mathcal{E}$ perspective, a task (t_i) is a concrete work unit (e.g., do blood test) that acts upon the environment. We define t_i as a 5-tuple <$\mathcal{E}$-$\texttt{Req}_{t_i}$, $\mathcal{E}$-$\texttt{Pre-Condition}_{t_i}$, $\mathcal{E}$-$\texttt{Input/Output}_{t_i}$, $\mathcal{E}$-$\texttt{Condition}_{t_i}$, $\mathcal{E}$-$\texttt{Post-Condition}_{t_i}$> where $\mathcal{E}$-$\texttt{Req}_{t_i}$ is t_i's requirements, $\mathcal{E}$-$\texttt{Pre-Condition}_{t_i}$ is a set of physical and logical elements to verify so that t_i is assigned to an executor (i.e., $\mathcal{E}$-$\texttt{Req}_{t_i}$ is satisfied with respect to the executor's capacities), $\mathcal{E}$-$\texttt{Input}_{t_i}$ and $\mathcal{E}$-$\texttt{Output}_{t_i}$ identify, respectively, the data that t_i may need for execution and the data that t_i may produce after execution, $\mathcal{E}$-$\texttt{Condition}_{t_i}$ is a set of physical and logical elements to verify so that t_i can be executed after successful assignment to an executor (e.g., all necessary inputs are available), and $\mathcal{E}$-$\texttt{Post-Condition}_{t_i}$ is a set of physical and logical elements to verify so that the execution of t_i is declared either successful or failure. In the case of failure corrective actions are taken and vary from one case study to another.

In the $\mathcal{S}$ perspective, a task (t_i) is an abstract work unit (i.e., narrative description) that "signs up" in the interchange and/or coupling networks. We define t_i as a couple 1[<$\mathcal{S}$-$\texttt{Network}_{rel}$, 1[$\mathcal{S}$-$\texttt{Connect}_{rel}(t_j, w_j)$]n>]2 where $\mathcal{S}$-$\texttt{Network}_{rel}$ is either interchange network or coupling network, $\mathcal{S}$-$\texttt{Connect}_{rel}(t_j, w_j)$ lists all $t_{j=1\cdots n, j\neq i, t_j\in\mathcal{S}\text{-}\texttt{Network}_{rel}}$ that are connected to t_i through rel, and w_j is the weight of the edge from t_i to t_j. Weight assessment is given a little bit later.

Relation. In the $\mathcal{E}$ perspective, a relation ($r_{(t_i,t_j)}$) establishes a concrete dependency between t_i and t_j at run time. A dependency can be either direct (i.e., t_j follows t_i) or indirect (i.e., because t_i is not self-contained, t_j is called whenever t_i's output needs to be processed). To handle task dependencies we decompose $\texttt{Input}_t$ into either $\texttt{initialInput}_t$ (i.e., submitted to t prior to execution) or $\texttt{partialInput}_t$ (submitted to t during execution) and also decompose $\texttt{Output}_t$ into either $\texttt{partialOutput}_t$ (i.e., produced by t

Table 2. Summary of social relations

Between	Types	Pre-Conditions	Conditions	Post-Conditions
t_i,t_j	**Coupling**	t_i and t_i participated in joint business processes	review of business process design or concern over coupling level	business process design completion or coupling level satisfaction
	Interchange	t_i and t_j producing similar output in receipt of similar input	t_i lacking of executor who satisfies its requirements	executor found for t_j
m_i,m_j	**Backup**	m_i and m_j having similar capacities	m_i unexpected failure or concern over m_i reliability	backup/replacement machine found for m_i
	Cooperation	m_i and m_j having similar capacities	concern over machine collective performance	collective performance level satisfaction
	Partnership	m_i and m_j having complementary capacities	concern over machine collective performance	collective performance level satisfaction
p_i,p_j	**Substitution**[1]	p_i and p_j having similar capacities	p_i expected unavailability (e.g., annual leave and sick leave) or concern over p_i availability	substitute found for p_i
	Delegation	p_i and p_j having similar capacities	p_i unexpected unavailability (e.g., call-in-sick, urgent tasks to complete, and risk of overload)	delegate found for p_i
	Peering	p_i and p_j having similar or complementary capacities	concern over peering appropriateness	peer found for either p_i or p_j

[1] **Substitution example:** (t_1:scan documentation) and ($t_1^{'}$:enter patient details manually) are connected together in the substitution network since the interchange pre-condition is met, i.e., producing similar output (e.g., patient's healthcare provider) in receipt of similar input (e.g., patient's number). This network will be used when the interchange condition is met, which is scanner down so t_1 cannot be executed and replaced with $t_1^{'}$. After identifying an executor for $t_1^{'}$ who is the agent at the reception, the use of this network is stopped since the interchange pre-condition is met.

during execution) or $\texttt{finalOutput}_t$ (i.e., produced by t at the end of execution). Unless stated Input and Output refer to initialInput and finalOutput, respectively. We define $r_{(t_i,t_j)}$ as a triple <t_i, t_j, $\mathcal{E}$-Type> where $\mathcal{E}$-Type is either prerequisite dependency (i.e., t_i successful execution as per $\mathcal{E}$-Post-Condition$_{t_i}$ and $\mathcal{E}$-finalOutput$_{t_i}$ $\cap$ $\mathcal{E}$-initialInput$_{t_j}$ $\neq \emptyset$), parallel-prerequisite dependency (i.e., $\mathcal{E}$-partialOutput$_{t_i}$ $\cap$ $\mathcal{E}$-initialInput$_{t_j}$ $\neq \emptyset$), parallel dependency ($\mathcal{E}$-partialOutput$_{t_i}$$\cap$ $\mathcal{E}$-partialInput$_{t_j}$ $\neq \emptyset$, and $\mathcal{E}$-partialOutput$_{t_j}$ $\cap$ $\mathcal{E}$-partialInput$_{t_i}$ $\neq \emptyset$), or completion relation ($\mathcal{E}$-Output$_{t_i}$ requires processing).

In the $\mathcal{S}$ perspective a relation ($r_{(t_i,t_j)}$) establishes a link between t_i and t_j both members of a network that is built upon this relation. We define $r_{(t_i,t_j)}$ as a 5-tuple <t_i, t_j, $\mathcal{S}$-Pre-Condition$_{(t_i,t_j)}$, $\mathcal{S}$-Condition$_{(t_i,t_j)}$, $\mathcal{S}$-Post-Condition$_{(t_i,t_j)}$> where $\mathcal{S}$-Pre-Condition$_{(t_i,t_j)}$ is a set of physical and logical elements to verify so that t_i connects t_j (i.e., either ($\mathcal{E}$-Output$_{t_i}$ and $\mathcal{E}$-Output$_{t_j}$ are equivalent with respect to similar input received) or (t_i and t_j participate in joint business processes)), $\mathcal{S}$-Condition$_{(t_i,t_j)}$ is a set of physical and logical elements to verify so that the network associated with $r_{(t_i,t_j)}$ is used, and $\mathcal{S}$-Post-Condition$_{t_i,t_j}$ is a set of physical and logical elements to verify before the ongoing use of the network associated with $r_{(t_i,t_j)}$ is stopped (i.e., executor found for t_i, business process enrichment completion, or coupling level satisfaction).

Relation evaluation. Assessing the weight (w) of $r_{(t_i,t_j)}$ is restricted to the $\mathcal{S}$ perspective, only.

- *Interchange relation*. Equation 1 measures the weight of an interchange edge where $|interchange_{(t_i,t_j)}|$ is the number of times that t_i is interchanged successfully with t_j (i.e., an executor is found for t_j) and $|failure_{t_i}|$ is the number of times that t_i is not executed due to lack of executors. A higher combined interchange weight for a task (e.g., close to 1) indicates the excessive interchange of this task, which should be taken into account when implementing critical business processes.

$$w^{\mathcal{S}}_{interchange_{(t_i,t_j)}} = \frac{|interchange_{(t_i,t_j)}|}{|failure_{t_i}|} \quad (1)$$

- *Coupling relation*. Equation 2 measures the weight of a coupling edge where $|participateJointProcess_{(t_i,t_j)}|$ is the number of times that t_i is directly coupled with t_j in joint business processes and $|participateProcess_{t_i}|$ is the number of times that t_i participates in business processes. A higher coupling weight (e.g., close to 1) indicates the "smoothness" of exchanging data between tasks due to "limited" semantic mismatches.

$$w^{\mathcal{S}}_{coupling_{(t_i,t_j)}} = \frac{|participateJointProcess_{(t_i,t_j)}|}{|participateProcess_{t_i}|} \quad (2)$$

Support network of machines. In a support network of machines, node and edge correspond to machine and relation between machines, respectively. We analyze both from $\mathcal{E}$ and $\mathcal{S}$ perspectives. On the one hand the $\mathcal{E}$ perspective sets the stage for assisting machines schedule task execution and also initiating this execution. On the other hand the $\mathcal{S}$ perspective sets the stage for building networks of machines using the relations listed in Table 2 (pre-condition satisfaction) and also using these networks when needed (condition satisfaction).

Machine. In the $\mathcal{E}$ perspective, a machine (m_i) is a concrete processing unit (e.g., printer) that receives tasks (e.g., print out document) for execution after confirming that the machine's capacities meet these tasks' requirements, i.e., tasks' pre-conditions satisfied. Like with task we define m_i as a 5-tuple <$\mathcal{E}$-Cap$_{m_i}$, $\mathcal{E}$-Pre-Condition$_{m_i}$, $\mathcal{E}$-Input/Output$_{m_i}$, $\mathcal{E}$-Condition$_{m_i}$, $\mathcal{E}$-Post-Condition$_{m_i}$> where $\mathcal{E}$-Cap$_{m_i}$ is m_i's capacities, $\mathcal{E}$-Pre-Condition$_{m_i}$ is a set of physical and logical elements to verify so that m_i schedules the execution of a task taking into account $\mathcal{E}$-Cap$_{m_i}$, $\mathcal{E}$-Input$_{m_i}$ and $\mathcal{E}$-Output$_{m_i}$ (optional) identify, respectively, the tasks that m_i receives for execution and the results (e.g., data and product) that m_i may produce after execution, $\mathcal{E}$-Condition$_{m_i}$ is a set of physical and logical elements to verify so that m_i begins executing a task (e.g., no pending tasks exist), and $\mathcal{E}$-Post-Condition$_{m_i}$ is a set of physical and logical elements to verify so that m_i detaches a task after execution and updates $\mathcal{E}$-Cap$_{m_i}$. The success or failure of this execution is dependent on the task's post-conditions.

In the $\mathcal{S}$ perspective, a machine (m_i) is an abstract processing unit (i.e., narrative description) that "signs up" in networks built upon the three possible social relations between machines. We define m_i as a couple 1[<$\mathcal{S}$-Network$_{rel}$, 1[$\mathcal{S}$-Connect$_{rel}(m_j, w_j)$]n>]3 where $\mathcal{S}$-Network$_{rel}$ is either backup network, cooperation network, or partnership network, $\mathcal{S}$-Connect$_{rel}(m_j, w_j)$ lists all $m_{j=1\cdots n, j\neq i, m_j \in \mathcal{S}\text{-Network}_{rel}}$ connected to m_i through rel, and w_j is the weight of the edge from m_i to m_j. Weight assessment is given a little bit later.

Relation. In the $\mathcal{E}$ perspective, a relation ($r_{(m_i,m_j)}$) establishes a link between two machines m_i and m_j both in charge of executing joint tasks. We define $r_{(m_i,m_j)}$ as a triple <m_i, m_j, $\mathcal{E}$-Type> where $\mathcal{E}$-Type is either enablement relation or inhibition relation.

In the $\mathcal{S}$ perspective, a relation ($r_{(m_i,m_j)}$) establishes a link between two machines m_i and m_j both members of a network built upon this relation. We define $r_{(m_i,m_j)}$ as a 5-tuple <m_i, m_j, $\mathcal{S}$-Pre-Condition$_{m_i,m_j}$, $\mathcal{S}$-Condition$_{m_i,m_j}$, $\mathcal{S}$-Post-Condition$_{m_i,m_j}$> where $\mathcal{S}$-Pre-Condition$_{m_i,m_j}$ is a set of physical and logical elements to check so that m_i connects m_j (i.e., either ($\mathcal{E}$-Cap$_{m_i}$ and $\mathcal{E}$-Cap$_{m_j}$ are equivalent (m_i and m_j do the same job) or ($\mathcal{E}$-Cap$_{m_i}$ and $\mathcal{E}$-Cap$_{m_j}$ are simultaneously required)), $\mathcal{S}$-Condition$_{(m_i,m_j)}$ is a set of physical and logical elements to verify so that the network associated with $r_{(m_i,m_j)}$ is used (i.e., either m_i fails unexpectedly, concern over m_i reliability, or concern over the collective performance of m_i and m_j), and $\mathcal{S}$-Post-Condition$_{m_i,m_j}$ is a set of physical and logical elements to check before the ongoing use of the network associated with $r_{(m_i,m_j)}$ is stopped (i.e., backup/replacement machine found for m_i or satisfaction with collective performance of m_i and m_j).

Relation evaluation. Assessing the weight (w) of $r_{(m_i,m_j)}$ is restricted to the $\mathcal{S}$ perspective, only. In the following $\mathcal{T}_m$ is the set of all tasks assigned to m.

- *Backup relation*. Equation 3 measures the weight of a backup edge where $\mathcal{T}^{fail}_{m_i} \subseteq \mathcal{T}_{m_i}$ is the set of tasks that m_i failed to execute and $|replaceSuc_{\mathcal{T}^{fail}_{m_i},(m_i,m_j)}|$ is the number of failed tasks in $\mathcal{T}^{fail}_{m_i}$ that m_j completes successfully. A higher combined backup weight for a machine (e.g., close to 1) indicates its reliability, which should be taken into account prior to finalizing task assignments.

$$w^{\mathcal{S}}_{backup_{(m_i,m_j)}} = \frac{|replaceSuc_{\mathcal{T}^{fail}_{m_i},(m_i,m_j)}|}{|\mathcal{T}^{fail}_{m_i}|} \quad (3)$$

- *Cooperation relation*. Equation 4 measures the weight of a cooperation edge where $\mathcal{T}^{coop}_{m_i} \subseteq \mathcal{T}_{m_i}$ is the set of all tasks that m_i executes in cooperation with other machines and $|cooperateSuc_{(\mathcal{T}^{coop}_{m_i} \cap \mathcal{T}^{coop}_{m_j})(m_i,m_j)}|$ is the number of tasks in $\mathcal{T}^{coop}_{m_i} \cap \mathcal{T}^{coop}_{m_j}$ that m_i and m_j execute successfully together. A cooperation weight reveals the collective performance of similar machines executing joint tasks.

$$w^{\mathcal{S}}_{cooperation_{(m_i,m_j)}} = \frac{|cooperateSuc_{(\mathcal{T}^{coop}_{m_i} \cap \mathcal{T}^{coop}_{m_j})(m_i,m_j)}|}{|\mathcal{T}^{coop}_{m_i}|} \quad (4)$$

- *Partnership relation*. Equation 5 measures the weight of a partnership edge where $\mathcal{T}^{part}_{m_i} \subseteq \mathcal{T}_{m_i}$ is the set of all tasks that m_i executes in partnership with other machines and $|partnerSuc_{(\mathcal{T}^{part}_{m_i} \cap \mathcal{T}^{part}_{m_j}),(m_i,m_j)}|$ is the number of tasks in $\mathcal{T}^{part}_{m_i} \cap \mathcal{T}^{part}_{m_j}$ that m_i and m_j complete successfully together. A partnership weight reveals the collective performance of different machines executing joint tasks.

$$w^{\mathcal{S}}_{partnership_{(m_i,m_j)}} = \frac{|partnerSuc_{(\mathcal{T}^{part}_{m_i} \cap \mathcal{T}^{part}_{m_j}),(m_i,m_j)}|}{|\mathcal{T}^{part}_{m_i}|} \quad (5)$$

Social networks of persons. In a social network of persons, node and edge correspond to person and relation between persons, respectively. We analyze both from $\mathcal{E}$ and $\mathcal{S}$ perspectives. On the one hand the $\mathcal{E}$ perspective sets the stage for assisting persons plan task execution and also initiating this execution. On the other hand the $\mathcal{S}$ perspective sets the stage for building networks of persons using the relations listed in Table 2 (pre-condition satisfaction) and also using these networks when needed (condition satisfaction).

- **Person.** In the $\mathcal{E}$ perspective, a person (p_i) is a concrete "processing unit" (e.g., doctor) who receives tasks (e.g., check patient) for execution after confirming that the person's capacities meet these tasks' requirements, i.e., tasks' pre-conditions satisfied. Like with tasks and machines we define p_i as a 5-tuple <$\mathcal{E}$-Cap$_{p_i}$, $\mathcal{E}$-Pre-Condition$_{p_i}$, $\mathcal{E}$-Input/Output$_{p_i}$, $\mathcal{E}$-Condition$_{p_i}$, $\mathcal{E}$-Post-Condition$_{p_i}$> where $\mathcal{E}$-Cap$_{p_i}$ is p_i's capacities, $\mathcal{E}$-Pre-Condition$_{p_i}$ is a set of physical and logical elements to verify so that p_i plans the execution of a task taking into account $\mathcal{E}$-Cap$_{p_i}$, $\mathcal{E}$-Input$_{p_i}$ and $\mathcal{E}$-Output$_{p_i}$ identify, respectively, the tasks that p_i receives for execution and the tasks/results that p_i might generate/produce after execution, $\mathcal{E}$-Condition$_{p_i}$ is a set of physical and logical elements to verify so that p_i begins executing a task (e.g., no-higher priority tasks exist), and $\mathcal{E}$-Post-Condition$_{p_i}$ is a set of physical and logical elements to verify so that p_i detaches a task after execution and updates $\mathcal{E}$-Cap$_{p_i}$. The success or failure of this execution is dependent on the task's post-conditions.

In the $\mathcal{S}$ perspective, a person (p_i) is an abstract "processing unit" (i.e., narrative description) who signs up in networks built upon the three possible relations between persons namely, substitution, delegation, or peering. We define p_i as a couple 1[<$\mathcal{S}$-Network$_{rel}$, 1[$\mathcal{S}$-Connect$_{rel}(p_j, w_j)$]n>]2 where $\mathcal{S}$-Network$_{rel}$ is either substitution network, delegation network, or peering network, $\mathcal{S}$-Connect$_{rel}(p_j, w_j)$ lists all $p_{j=1\cdots n, j\neq i, p_j \in \mathcal{S}\text{-Network}_{rel}}$ connected to p_i through rel, and w_j is the weight of the edge from p_i to p_j. Weight assessment is given a little bit later.

- **Relation.** In the $\mathcal{E}$ perspective, a relation ($r_{(p_i,p_j)}$) establishes a link between two persons p_i and p_j in charge of executing joint tasks. We define $r_{(p_i,p_j)}$ as a triple <m_i, m_j, $\mathcal{E}$-Type> where $\mathcal{E}$-Type is either enablement relation or inhibition relation.

  In the $\mathcal{S}$ perspective, a relation ($r_{(p_i,p_j)}$) establishes a link between two persons p_i and p_j both members of a network built upon this relation. We define $r_{(p_i,p_j)}$ as a 5-tuple <p_i, p_j, $\mathcal{S}$-Pre-Condition$_{p_i,p_j}$, $\mathcal{S}$-Condition$_{p_i,p_j}$, $\mathcal{S}$-Post-Condition$_{p_i,p_j}$> where $\mathcal{S}$-Pre-Condition$_{p_i,p_j}$ is a set of physical and logical elements to check so that p_i connects p_j (i.e., $\mathcal{E}$-Cap$_{p_i}$ and $\mathcal{E}$-Cap$_{p_j}$ are similar or complementary), $\mathcal{S}$-Condition$_{(p_i,p_j)}$ is a set of physical and logical elements to verify so that the network associated with $r_{(p_i,p_j)}$ is used (i.e., p_i is unavailable, concern over p_i or p_j availability, or concern over peering p_i and p_j together), and $\mathcal{S}$-Post-Condition$_{p_i,p_j}$ is a set of physical and logical elements to check before the ongoing use of the network associated with $r_{(p_i,p_j)}$ is stopped (i.e., substitute/delegate was identified for p_i or peer found for either p_i or p_j).

- **Relation evaluation.** Assessing the weight (w) of $r_{(p_i,p_j)}$ is restricted to the $\mathcal{S}$ perspective, only. In the following $\mathcal{T}_p$ is the set of tasks assigned to p.

  - *Substitution relation*. Equation 6 measures the weight of a substitution edge where $\mathcal{T}_{p_i}^{sub} \subseteq \mathcal{T}_{p_i}$ is the set of all tasks assigned to p_i but then are assigned to a different person due to p_i expected unavailability and $|substituteSuc_{\mathcal{T}_{p_i}^{sub},(p_i,p_j)}|$ is the number of tasks in $\mathcal{T}_{p_i}^{sub}$ that p_j completes successfully. A higher combined substitution weight for a person (e.g., close to 1) indicates her availability, which should be taken into account prior to finalizing task assignment.

$$w^{\mathcal{S}}_{substitution_{(p_i,p_j)}} = \frac{|substituteSuc_{\mathcal{T}_{p_i}^{sub},(p_i,p_j)}|}{|\mathcal{T}_{p_i}^{sub}|} \tag{6}$$

  - *Delegation relation*. Equation 7 measures the weight of a delegation edge where $\mathcal{T}_{p_i}^{del} \subseteq \mathcal{T}_{p_i}$ is the set of all tasks assigned to p_i but then are delegated to p_j due to her unexpected unavailability and $|delegateSuc_{\mathcal{T}_{p_i}^{del},(p_i,p_j)}|$ is the number of tasks in $\mathcal{T}_{p_i}^{del}$ that p_i delegates. A higher combined delegation weight for a person (e.g., close to 1) indicates her work-of-transfer level to other persons.

$$w^{\mathcal{S}}_{delegation_{(p_i,p_j)}} = \frac{|delegateSuc_{\mathcal{T}_{p_i}^{del},(p_i,p_j)}|}{|\mathcal{T}_{p_i}^{del}|} \tag{7}$$

  - *Peering relation*. Equation 8 measures the weight of a peering edge where $\mathcal{T}_{p_i}^{peer} \subseteq \mathcal{T}_{p_i}$ is the set of all tasks that p_i executes with other peers and $|peeringSuc_{(\mathcal{T}_{p_i} \cap \mathcal{T}_{p_j}^{peer}),(p_i,p_j)}|$ is the number of tasks assigned to p_i and p_j that both complete successfully together in different business processes. A higher combined peering weight for a person (e.g., close to 1) reveals the appropriateness of having this person execute joint tasks with other persons.

$$w^{\mathcal{S}}_{peering_{(p_i,p_j)}} = \frac{|peerSuc_{(\mathcal{T}_{p_i} \cap \mathcal{T}_{p_j}^{peer}),(p_i,p_j)}|}{|\mathcal{T}_{p_i}^{peers}|} \tag{8}$$

4.4. Role of networks in addressing conflicts

In [26] we detail the role of the networks in addressing conflicts that hinder BP execution. This role refers to BP social coordination that includes four steps: categorize resources that tasks require for their execution, define how tasks/machines/persons of a BP bind to resources during this execution, categorize conflicts on resources that arise between tasks, between machines, and between persons, and finally analyze the appropriateness of certain networks of tasks/persons/machines for addressing these conflicts.

First we categorize resources into (*i*) logical, i.e., their use/consumption does not lead into a decrease in their reliability/availability level and (*ii*) physical, i.e., their use/consumption does lead into a decrease in their reliability/availability level. This decrease requires resource replacement[3]/replenishment at a certain stage. For the sake of illustration in this paper we restrict discussions to logical resources, only. Afterwards we define a set of properties that allow to describe a resource (Table 3). These properties are *unlimited* (ul, by default), *limited* (l - when a resource use/consumption is measured or a resource ceases to exist due to temporal constraints, for example), and *limited but renewable* (lr - when a resource use/consumption either hits a certain threshold or is subject to temporal constraints, for example; in either case renewal is possible). Additional properties could be considered if need be, such as *non-shareable* (ns - when a resource simultaneous use/consumption has to be scheduled) and *shareable* (s, by default).

Table 3. Examples of logical resources per property type

Resource		Examples
Category	Property	
Logical	Unlimited (ul)	Data (read mode), software (no cap on number of licences)
	Limited (l)	Thread
	Limited but renewable (lr)	File access right (valid for a certain time with possible extension), password (valid for a certain time with possible extension)

$\mathcal{T}$-Conflict$_1$ is an example of conflicts between tasks with emphasis on resource consumption and not data (inputs and outputs) and policy incompatibilities between these tasks. $\mathcal{T}$-Conflict$_1$ arises when (*i*) a prerequisite relation between t_i and t_j exists, (*ii*) consume(t_i, r_i) $\rightarrow$ produce($t_i, r_{i,j}$), and (*iii*) t_j needs $r_{i,j}$ (i.e., $t_j \nrightarrow r_j$, no r_j is made available for t_j). Potential conflicts on $r_{i,j}$ (and eventually $r_{\{i,k,\cdots\},j}$ and $r_{i,\{j,k,\cdots\}}$) because of the *limited* property of $r_{i,j}$, include:

- l: two cases result out of the prerequisite relation between $t_{\{k,\cdots\}}$ (e.g., complete necessary paperwork) and t_j (e.g., direct patient to appropriate department) on top of the same relation between t_i (e.g., check patient vitals) and t_j:

    a) $r_{i,j}$ (e.g., report on vital levels) ceases to exist (e.g., blood sample no longer valid) before the execution of t_j begins; t_j waits for $t_{\{k,\cdots\}}$ to produce $r_{\{k,\cdots\},j}$ (e.g., insurance provider approval); (at least one) $t_{\{k,\cdots\}}$ either is still under execution (e.g., due to delay in receiving approval from insurance provider) or failed.

    b) Only one consumption cycle of $r_{i,j}$ is permitted (per type of property) but it turns out that several consumption cycles of $r_{i,j}$ are required to complete the execution of t_j and finish the consumption of $r_{\{k,\cdots\},j}$ that $t_{\{k,\cdots\}}$ produce.

[3]Replacement can be the result of degradation.

After identifying the different task conflicts on resources, we suggest solutions for these conflicts based on the aforementioned networks. These solutions consider the fact that tasks are associated with transactional properties (e.g., pivot, retriable, and compensatable) that limit their re-execution in the case of failure [27]. The following examines briefly how the interchange and coupling networks of tasks are used to address $\mathcal{T}$-Conflict$_1$-Case a.

a) $r_{i,j}$ ceases to exist before the execution of t_j begins; t_j waits for $t_{\{k,\cdots\}}$ to produce $r_{\{k,\cdots\},j}$; at least one t_k either is still under execution or failed. Current statuses of tasks and resources are: state(t_i): done; state($r_{i,j}$): withdrawn; state(t_j): not-activated; and state(t_k): either activated (still under execution) or failed with focus on the latter state below. Because t_i now takes on done state, pivot (canceling t_i) and retriable (re-executing t_i) transactional properties are excluded from the analysis of developing solutions to address resource conflicts. This analysis is given in Table 4. The objective is to reproduce $r_{i,j}$ (or produce $r_{i',j}$ with $t_{i'}$ being obtained through the interchange network of t_i). Because of t_k failure, $r_{k',j}$ is produced using $t_{k'}$ that is obtained through the interchange network of t_k.

5. Implementation

In this section we describe the architecture of the **S**ocial-based b**U**siness **P**rocess manag**E**ment f**R**amework (SUPER) that supports our research on the social enterprise (or enterprise 2.0) [28]. SUPER is a Business Process Management System (BPMS) that uses social computing principles (i.e., connecting entities together through relations, developing networks upon these relations, and analyzing these networks) to model and develop business processes. Fig. 2 illustrates the architecture of SUPER that is built upon multiple components discussed below.

At design time, process engineers (or designers) define new Business Processes (BP)s using the BP modeling component. This component is an extension of the Yaoqiang BPMN editor [29] with new operations that for instance, assign executors (persons and/or machines) to tasks and specify the requirements and capacities of tasks, persons, and machines accordingly. Fig. 3 shows

Table 4. Possible coordination actions in the case of $\mathcal{T}$-Conflict$_1$/limited property/case a

Transactional property		Coordination actions	Network involved
t_i	t_k		
Null	Null	– re-execute t_i to re-produce $r_{i,j}$ – re-execute t_k to produce $r_{k,j}$	N/A
	Pivot	Deadlock	N/A
	Compensatable	Deadlock	N/A
	Retriable	– re-execute t_i to re-produce $r_{i,j}$ – replace t_k with $t_{k'}$ then execute $t_{k'}$ to produce $r_{k',j}$	Interchange(t_k, $t_{k'}$)

Compensatable	Null	– compensate t_i; either re-execute t_i to re-produce $r_{i,j}$ or replace t_i with $t_{i'}$ then execute $t_{i'}$ to produce $r_{i',j}$
		– either re-execute t_k to produce $r_{k,j}$ or replace t_k with $t_{k'}$ then execute $t_{k'}$ to produce $r_{k',j}$
			Interchange(t_i, $t_{i'}$)
			Interchange(t_k, $t_{k'}$)

	Pivot	Deadlock	N/A

	Compensatable	– compensate t_i; either re-execute t_i to re-produce $r_{i,j}$ or replace t_i with $t_{i'}$ then execute $t_{i'}$ to produce $r_{i',j}$
		– replace t_k with $t_{k'}$ then execute $t_{k'}$ to produce $r_{k',j}$
			Interchange(t_i, $t_{i'}$)
			Interchange(t_k, $t_{k'}$)

	Retriable	– compensate t_i; either re-execute t_i to re-produce $r_{i,j}$ or replace t_i with $t_{i'}$ then execute $t_{i'}$ to produce $r_{i',j}$ – re-execute t_k to produce $r_{k,j}$	Interchange(t_i, $t_{i'}$)

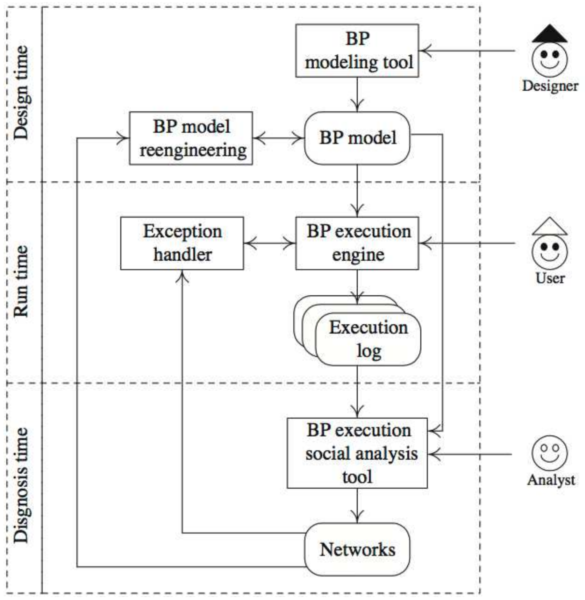

Figure 2. SUPER architecture

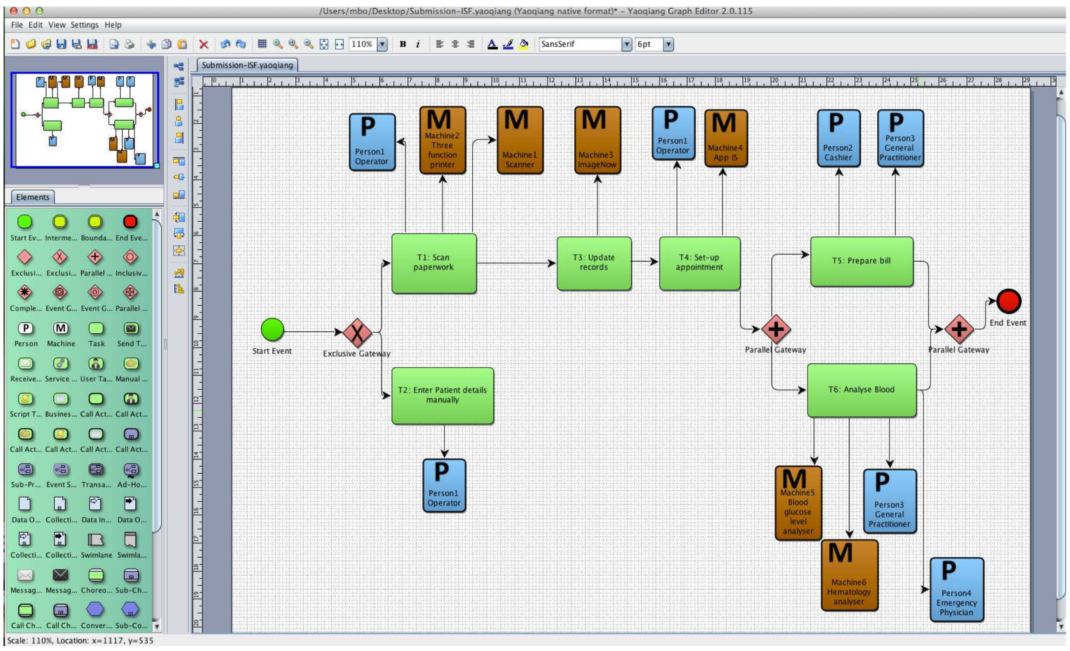
/Users/mbo/Desktop/Submission-ISF.yaoqiang (Yaoqiang native format)* – Yaoqiang Graph Editor 2.0.115
File Edit View Settings Help
Submission-ISF.yaoqiang
Elements
Start Event
Exclusive Gateway
T1: Scan paperwork
T2: Enter Patient details manually
T3: Update records
T4: Set-up appointment
T5: Prepare bill
T6: Analyse Blood
Parallel Gateway
End Event
Person1 Operator
Machine2 Three function printer
Machine1 Scanner
Machine3 ImageNow
Machine4 App IS
Person2 Cashier
Person3 General Practitioner
Machine5 Blood glucose level analyser
Machine6 Hematology analyser
Person4 Emergency Physician
Scale: 110%, Location: x=1117, y=535

Figure 3. Screenshot of Yaoqiang BPMN editor

a screenshot of the Yaoqiang BPMN editor extension in which a part of the electronic-patient-folder system BP is modeled (Section 3).

At run time, BPs are executed using appropriate engines for instance, Jboss JBPM engine[4]. This engine interprets the specification of the BP model that is generated from the Jboss JBPM editor. The different execution traces accumulated during the execution of BP instances are stored in a log file. These traces contain information about events referring to the execution of tasks in terms of execution time (timestamp) and executors. The traces are in e**X**tensible **E**vent **S**tream (XES) format[5] [30]. XES is the de facto standard for process execution log expression and adopted by

[4]www.jboss.org/jbpm

[5]http://www.xes-standard.org/

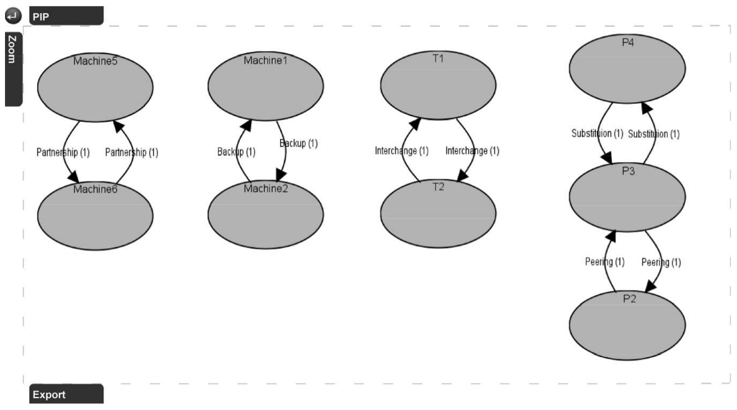

Figure 4. Screenshot of the BP execution social analysis component, a demo video is available at https://www.youtube.com/watch?v=Py5oGPQot64

several log analyzer tools such as the popular process mining framework, ProM[6].

At diagnosis time, a BP execution social analysis component discovers and builds the necessary category of networks (configuration, machine, and social) based on process execution logs as well as BP models. This component is developed as a plugin of the ProM framework and represents the discovered networks using a well-known XML-based format for representing graphs, GraphML[7]. This format is supported by several graph drawing and analyzing tools such as yED Graph Editor[8] and Eclipse Plugin Postfuse[9]. Therefore, such tools can also be used to visualize and analyze the discovered networks by the BP execution social analysis component.

Fig. 4 depicts examples of networks related to the electronic-patient-folder system that are generated using the BP execution social analysis component. For example, a network of machines is built to specify the backup relation between m_1 (`scanner`) and m_2 (`three-function-printer`) since both machines have the similar capacities (using subsumption). A configuration network of tasks is also constructed to express the interchange relation between t_1 (`scan-documentation`) and t_2 (`enter-patient-details-manually`). Both tasks produce similar output and their requirements do not overlap. Last and not least, a social network of persons is built to describe the peering relation between p_2 (`cashier`) and p_3 (`financial-manager`) since both persons have complementary capacities that are necessary to achieve t_5 (`prepare-bill`). This social network expresses also a substitution relation between p_3 (`general-practitioner`) and p_4 (`emergency-physician`) since both have the same capacity.

[6]www.promtools.org
[7]http://graphml.graphdrawing.org/
[8]www.yworks.com
[9]http://postfuse.macrolab.de

6. Conclusion

This paper discussed business processes from a social perspective. Different relations with a "social flavor" have been proposed such as interchange, substitution, delegation, and peering. They permitted to connect the components of a business process that are task, person, and machine together. A case study illustrated these components as well as these relations. For instance, tasks can be substituted when their requirements are not met. Persons delegate their tasks when unexpected changes occur in their schedules. Last but not least machines act as backups when their peers fail. The role of all these networks in conflict resolution is detailed in [26]. In term of future work, we would like to conduct some acceptance tests by end-users and to extend the analysis component of SUPER to handle cases like load balancing and resource management.

References

[1] Badr, Y. and Maamar, Z. (October 2009) Can Enterprises Capitalize on Their Social Networks? *Cutter IT Journal* 22(10).

[2] Chandler, S. (November 2011), Social BPM: Gateway to Enhanced Process Efficiency. Http://www.virtusa.com/blog/2011/11/, visited September 2014.

[3] Kanaracus, C. (2013), Gartner: Social Business Efforts Largely Unsuccessful so Far. Http://www.computerworld.com/s/article/9236323/.

[4] Killock, P. (1999) The Economies of Online Cooperation: Gifts and Public Goods in Cyberspace. In A., S.M. and Kollock, P. [eds.] *Communities in Cyberspace* (Routledge, London), 259–262.

[5] Farzan, R., DiMicco, J.M., Millen, D.R., Brownholtz, B., Geyer, W. and Dugan, C. (2008) Results from Deploying a Participation Incentive Mechanisms within

the Entrerprise. In *Proceedings of the 2008 Conference on Human Factors in Computing Systems (CHI'2008)* (Florence, Italy).

[6] Cress, U., Held, C. and Kimmerle, J. (2013) The Collective Knowledge of Social Tags: Direct and Indirect Influences on Navigation, Learning, and Information Processing. *Computers & Education* **60**(1).

[7] Domingo, M.C. (July 2010) Managing Healthcare Through Social Networks. *Computer* **43**(7).

[8] Maamar, Z., Faci, N., Kouadri Mostéfaoui, S. and Akhter, F. (2011) Towards a Framework for Weaving Social Networks Into Mobile Commerce. *International Journal of Systems and Service-Oriented Engineering* **2**(3).

[9] Tan, W., Blake, M.B., Saleh, I. and Dustdar, S. (September/October 2013) Social-Network-Sourced Big Data Analytics. *IEEE Internet Computing* **17**(5).

[10] Dustdar, S. and Bhattacharya, K. (May/June 2011) The Social Compute Unit. *IEEE Internet Computing* **15**(3).

[11] Warr, W. (2008) Social Software: Fun and Games, or Business Tools? *Journal of Information Science* **34**(4).

[12] Schmidt, R. and Nurcan, S. (2008), BPM and Social Software. Proceedings of the First Workshop on Business Process Management and Social Software (BPMS2'2008), Milan, Italy.

[13] Erol, S., Granitzer, M., Happ, S., Jantunen, S., Jennings, B., Koschmider, A., Nurcan, S. *et al.* (October-November 2010) Combining BPM and Social Software Contradiction or Chance. *Journal of Software Maintenance and Evolution: Research and Practice* **22**(6-7).

[14] Liptchinsky, V., Khazankin, R., Truong, H.L. and Dustdar, S. (2012) A Novel Approach to Modeling Context-Aware and Social Collaboration Processes. In *Proceedings of the 24th International Conference on Advanced Information Systems Engineering (CAiSE'2012)* (Gdansk, Poland).

[15] Bruno, G., Dengler, F., Jennings, B., Khalaf, R., Nurcan, S., Prilla, M., Sarini, M. *et al.* (2011) Key Challenges for Enabling Agile BPM with Social Software. *Journal of Software Maintenance and Evolution: Research and Practice* **23**(10).

[16] Rito Silva, A., Meziani, R., Magalhães, R., Martinho, D., Aguiar, A. and Flores, N. (2010) AGILIPO: Embedding Social Software Features into Business Process Tools. In *Proceedings of the Third International Workshop on the Business Process Model and Notation (BPMN'2010)* (Ulm, Germany).

[17] Brambilla, M., Fraternali, P. and Vaca, C. (2011) A Notation for supporting Social Business Process Modeling. In *Proceedings of the Fourth Workshop on Business Process Management and Social Software (BPMS2'2011) held in conjunction with The Seventh International Conference on Business Process Management (BPM'2011)* (Lucerne, Switzerland).

[18] Koschmider, A., Song, M. and Reijers, H. (2010) Social Software for Modeling Business Processes. *Journal of Information Technology* **25**(3).

[19] Grim-Yefsah, M., Rosenthal-Sabroux, C. and Thion, V. (2011) Using Information of an Informal Network to Evaluate Business Process Robustness. In *Proceedings of the International Conference on Knowledge Management and Information Sharing (KMIS'2011)* (Paris, France).

[20] Davenport, T.H. and Short, J.E. (1990) The New Industrial Engineering: Information Technology and Business Process Redesign. *Sloan Management Review* .

[21] Alfaro Saiz, J.J., Rodríguez-Rodríguez, R., Verdecho, M.J. and Ortiz, A. (2009) Business Process Interoperability and Collaborative Performance Measurement. *International Journal Computer Integrated Manufacturing* **22**(9).

[22] Carstensen, A., Holmberg, L., Sandkuhl, K. and Stirna, J. (2008) Integrated Requirement and Solution Modelling: An Approach Based on Enterprise Models. In T., H., J., K. and E., P. [eds.] *Innovations in Information Systems Modeling: Methods and Best Practices* (IGI Global), 89–105.

[23] Roush, W. (August 2005) Social Machines - Computing means Connecting. *MIT Technology Review* .

[24] Limthanmaphon, B. and Zhang, Y. (2003) Web Service Composition with Case-Based Reasoning. In *Proceedings of the 14th Australasian Database Conference (ADC'2003)* (Adelaide, Australia).

[25] Decker, K. and Lesser, V. (1992) Generalizing the Partial Global Planning Algorithm. *International Journal Cooperative Information Systems* **1**(2).

[26] Maamar, Z., Noura, F., Kouadri Mostéfaoui, S. and Kajan, E. (2013) Network-based Conflict Resolution in Business Processes. In *Proceedings of the 10th IEEE International Conference on e-Business Engineering (ICEBE'2013)* (Coventry, UK).

[27] Little, M. (2003) Transactions and Web Services. *Communications of the ACM* **46**(10).

[28] Faci, N., Maamar, Z., Kajan, E. and Benslimane, D. (2014) Research Roadmap for the Enterprise 2.0 Ű Issues & Solutions. *Scientific Publications of the State University Of Novi Pazar Journal, Series A: Applied Mathematics, Informatics & Mechanics* **2**(2).

[29] Kajan, E., Faci, N., Maamar, Z., Loo, A., Pljaskovic, A. and Sheng, Q.Z. (March/April 2014) The Network-based Business Process. *IEEE Internet Computing* **18**(2).

[30] Verbeek, H., Buijs, J., van Dongen, B. and van der Aalst, W. (2011) XES, XESame, and ProM 6. *Information Systems Evolution* **72**.