A Social Network-based Process-aware Task Management System

Seyed Alireza Hajimirsadeghi Hye-Young Paik John Shepherd
Anthony Kosasih

University of New South Wales, Australia
{seyedh,hpaik,jas}@cse.unsw.edu.au,ajkosasih@gmail.com

Technical Report
UNSW-CSE-TR-201320
August 2013

THE UNIVERSITY OF NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

In modern society, we are frequently required to perform administrative or business processes in order to achieve our personal goals. We call these kinds of ad hoc processes, carried out towards a personal goal, *personal processes*. For almost all of our personal goals, from applying for a research position in academia or a job in industry to organising a marriage ceremony or buying a house, we seek help from social networks. Social networks are the prevalent medium for sharing notes, documents, images, videos, etc and they also provide the basic infrastructure for exchanging opinions. However we believe social networks are not particularly helpful when it comes to *personal process management (PPM)* as there remain significant problems in discovering and integrating the sets of tasks that are typically required in order to achieve many useful outcomes. This is mainly because social networks do not possess an integrated and structured framework for sharing "process knowledge". In this paper, we propose *Processbook*, a social network-based management system for *personal processes*. A simple modelling interface is introduced based on ToDoLists to help users plan towards their goals. We describe how the system can capture a user's experience in managing their ToDoList and the associated personal process, how this information can be shared with other users, and how the system can use this information to recommend process strategies. We exemplify the approach by a sample administrative process inside the University of New South Wales.

1 Introduction

In modern society, we are frequently required to perform administrative or business processes in order to achieve our goals. Examples could be simple processes such as planning a party, booking a theatre ticket, or more complicated and long lasting ones such as planning to get postgraduate admission from a top-ranked university, applying for a job in computer engineering or planning for immigration.

While the last decade has seen many of these individual processes codified via online services, there remain significant problems in discovering and integrating the tasks that are typically required to achieve useful outcomes. An important aspect of the problem is that processes frequently span organisational boundaries and there are few mechanisms to carry information and outcomes from processes in one organisation to those in the next organisation. Another major factor is that it is sometimes difficult to identify precisely which organisations and which processes within those organisations are required to accomplish a stated goal.

Discovering which business processes are relevant is frequently achieved either by searching on the Web or by being given information from friends who have previously accomplished these goals. As a consequence, each individual needs to integrate the heterogeneous sources of information themselves to obtain a useful set of tasks to reach their goal.

Integrating a collection of unfamiliar processes to achieve a goal causes many difficulties. First, it may be difficult to determine all the tasks required to complete the whole process. Second, finding out the dependencies between tasks is time consuming and potentially error prone. Third, people cannot generally predict how a particular task will affect the overall process of achieving the final outcome. Finally, tracking and managing on-the-fly changes in organisations' policies and procedures is difficult for individuals. Many of these problems stem from the fact that process and data are spread over multiple business units and the required knowledge to proceed with the whole process is not stated formally in the documents or in the guidelines from either of those business units. By contrast, overall knowledge about the process *is* often available in the minds of those who have previously gone through the process steps.

A *personal process* is made up of a number of tasks which need to be carried out in order to achieve a goal; a task may be as simple as one individual activity or may be as complex as a complete business process. Personal processes often require an integration of so-called "long tail" business processes from one or more organisations. The knowledge of such integration and consequently the knowledge of achieving a goal is referred to as *process knowledge* in this paper.

Our goal is to provide support for individuals to manage their personal processes. One possible approach for this would be to convert all personal processes into codified business processes, but this is neither plausible nor cost-effective [17]. Instead, we aim to assist users in discovering the tasks they have to do to reach their goals, the order they need to do the tasks and the sets of constraints and rules they should follow in doing those tasks. This is accomplished in a context where other users have already successfully carried out the processes and have recorded the method by which they did so.

Our system to support this (*Processbook*) adopts a social-based approach, aimed at non-expert users, who describe their personal processes via *ToDoList*s.

Processbook collects, manages, merges and shares process descriptions and allows users to follow other users who are carrying out similar tasks. It can also recommend future steps in a given process based on how others users previously achieved the same goal.

In Section 2, we elaborate the problem area with examples. Section 3 describes related work in the space of personal processes and using social software in business process management. Section 4 gives an overview of *Processbook* framework highlighting its main components. Section 5 presents the details of *Processbook* concentrating on how process knowledge is captured and then recommended to users. In Section 6 we investigate the implemented tool with an example in academic domain. Finally we conclude the paper in Section 7 with future work.

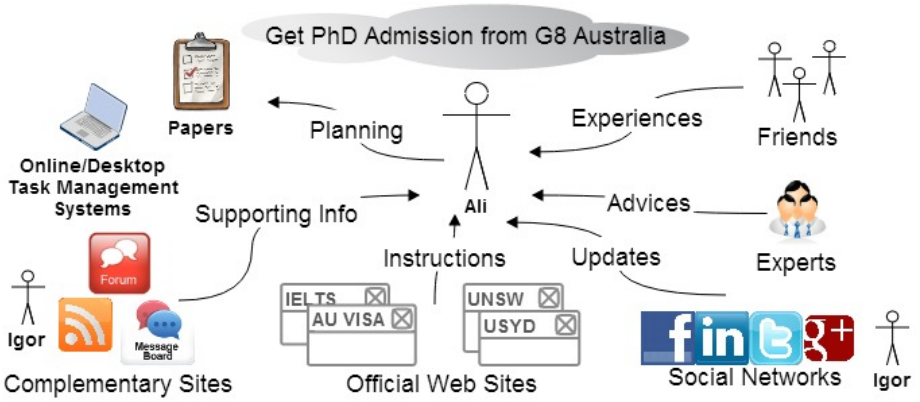
2 Motivating Scenario

We consider the problem of carrying out personal processes via an example: Ali, a student from a non English-speaking country, wishes to study for a PhD in one of the top eight Australian universities (known as the Go8). Ali has two primary objectives: find a university that would accept him and maximise the amount of funding to assist his study. Additional constraints and preferences might include: a PhD topic in the service oriented computing area, a PhD program commencing after July 2013, a department that has close relationships with industry, etc.

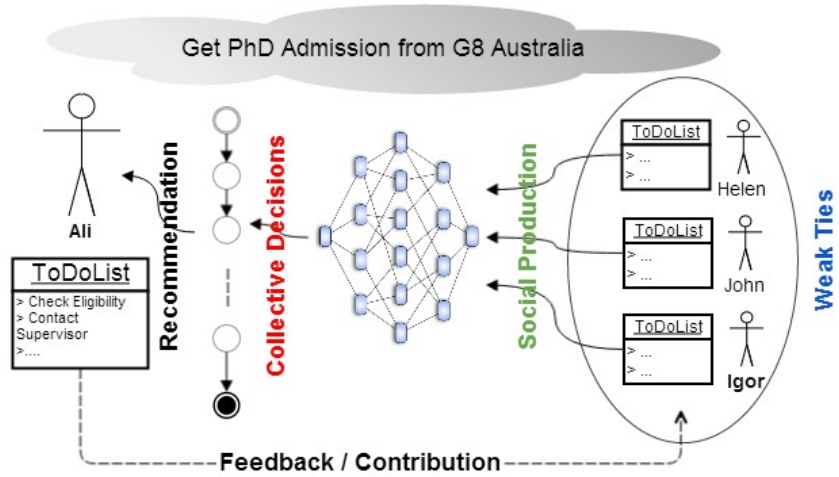
Figure 2.1a illustrates how Ali plans to reach his goals and what sources he utilises for his purpose. He first tries to identify universities that satisfy his constraints by asking friends or by searching on the Web. Once he identifies the universities, Ali collects and collates information about the entry requirements and scholarship availability for each university from its official web site. This might identify further subgoals, documents that need to be provided, timelines for applications, etc. He might also join relevant communities on popular social networks such as facebook and twitter to keep up with the latest news and updates about the institutions he is dealing with.

In carrying out the above, questions would arise at each stage for Ali. For example, the web site at some university might specify that a student needs to provide an undergraduate transcript and English proficiency test results, but might not mention the kind of visa that the student requires or how to obtain such a visa. Other typical questions that might arise are: what step should I take next, what do I do at each step, which organisation should I deal with, etc. To find the answers to such questions, Ali would seek answers from his friends, experts, social networks or other data available in online forums, blogs, etc. Ali uses a to-do list approach to organise his findings and to write down the tasks he has to do. He may do so by simply writing on paper, using computer-aided task management tools or registering in an online task management website where he is offered more gadgets to organise his tasks.

Getting advice from someone experienced with the specific process would be extremely useful, but finding such an expert might be difficult. Experts bring domain and process knowledge, but also need to have your personal circumstances communicated to them. A more effective approach might be to have the process information available online, and have a system that understands



(a) Example of a Personal Process Management



(b) Personal Process Management in *Processbook*

Figure 2.1: From Personal Involvement to Active Social Participation

both the process information and your personal constraints and situation (in terms of progress through the process), and can offer sufficient information to enable you to determine how to proceed. In practice, a number of difficult issues need to be dealt with before such a system can be realised:

Invalid, incomplete or inconsistent data: We may be faced with untrusted sources of information, or conflicting items of information, or may be given out-of-date information. Sometimes, we simply do not know certain parts of the process. In other cases, there may be hidden (or ignored) pieces of information. For example, Middle Eastern students may face a wait of up to three months in applying for an Australian student visa.

Individuals as process integrators: Individuals are responsible to collect all the relevant data and integrate it effectively while planning their goals. This is a challenging task as the process domain is usually new to individuals

and they do not have an overview of the process when progressing step by step. Heterogeneous data sources and data formats, numerous data dependencies over multiple organisations and constantly changing policies and workflows makes it even more difficult to play the role of process integrator.

Inability to predict task effects: Sometimes it is difficult to know what kind of effect accomplishing a specific task will have on the process as a whole. For example, while either of the IELTS and TOEFL English competency tests are accepted world wide, it is better to have IELTS scores if applying for Australian universities because they are better regarded.

Isolated individuals: Although people join communities on social networks and discuss issues with peers in forums and e-how sites, they are ultimately progressing in an isolated manner regarding the personal process as a whole. In Figure 2.1a, Igor is a member of some Go8 communities on Facebook, is contributing on an IELTS message board, and is also reflecting his experiences in his own blog. There is no established way for Ali to be aware of all Igor's contributions on the process, to contact him or to track his progress. People with similar goals and interests may not be able to find each other easily and each individual's progress is not necessarily recorded for future reuse.

In *Processbook*, we consider merging social software principles with process management basics to overcome the above issues. Social software is based on three core principles: social production [2], weak ties [7] and collective decisions [15]. Social production is the creation of artefacts by combining input from individual contributors without predetermining the way to do this. It abstains from the top-down planning, enables co-creation and increases innovative contributions. Weak ties are spontaneously established contacts between individuals that allow identifying competencies in organizations across departmental boundaries. Collective decisions decorrelates errors by aggregating a large number of independent judgements.

As Figure 2.1b illustrates, people are grouped based on common goals and their peer contributions are recorded and integrated automatically to produce a general view of the process pathways to achieve each goal. Finally we are able to recommend to Ali a path that most likely will guide him to reach his goal. Moreover we provide a feedback loop that makes each individual's involvement reusable and part of the social production.

A major goal of *Processbook* is to improve the management of personal processes by employing the social software principles noted above:

Weak ties: Establishing a goal-based social network removes the isolation barrier, helping individuals find peers with similar interests more easily. This way Ali could easily follow and make use of Igor's progress.

Social production: Users' knowledge and experience is captured unobtrusively while they are planning for their goal via a simple modelling interface that requires no prior process modelling knowledge. Merging individuals' knowledge on processes for achieving a certain goal allows new users to identify the major task flows and data flows while observing how each task effects the final process outcome.

Collective Decisions: Applying users' votes and feedback on socially produced process knowledge can be used to recommend process models that minimize the adverse effect of invalid, inconsistent and incomplete data.

3 Related Work

Personal process management has so far received limited attention from the academic research community. A vision statement can be found on the blog posted by Michael Rosemann [12]. Two possible implementations of personal process management are discussed in [17] and [3]. [17] mainly focuses on sequential and conditional constraints by introducing a formal personal process modelling language. The proposal in [3] is based on parallel executions, tries to simplify BPM techniques, and pays attention to the role of social aspects of process management (such as sharing and assigning tasks). Both works remain at preliminary level and are yet to realise any significant improvement over personal process management.

On the other side, a large number of commercial online tools exist for personal task management. These tools are end-user oriented and provide a plethora of features including task creation, sharing, social network integration, notification, etc. However, as [3] notes, none of them is concerned about the "process" concept; they do not embrace the practices of BPM, thus losing many beneficial aspects of structuring dependencies and constraints between tasks.

In terms of requiring flexibility and agility, personal process management is closely related to the agile BPM field. The need to remain competitive in today's fast-changing business environment has made enterprises introduce flexibility into their process models. [13] investigates four distinct approaches to gain flexibility within a Process-Aware Information System (PAIS): flexibility by design, by deviation, by underspecification and by change. All of these approaches trade off user support for flexibility. Moreover PAIS suffers from lacking a systematic approach for reusing and sharing knowledge. [17] also expresses the need of a new management system for personal processes - business processes as experienced and described by a single person.

The most notable defect of classical BPM is the so-called "model-reality divide", the distance between abstract process models and the processes executed in practice. [5] even states that agile BPM not only requires changes to the BPM life cycle, but also a paradigmatic change, and suggests that this change can be realised by applying social software features into business process management. [6] argues that more realistic models can be designed by applying social software features such as self identification, transparency, signing, logging, discussion and banning to the mechanism of process modelling. [9] contrasts the work management style in social software and business process modelling system(BPMS) and then proposes a set of guidelines suggesting how to use both in organisations, portrays an ideal modelling framework which eliminates the conventional hierarchic views of the world, includes more people in designing models, and removes *a priori* decisions on process modelling.

There have been some attempts in recent years to accommodate social features in the BPM environment. [10] targets the problem of "one person modelling tools" which has brought a general dissatisfaction among business users. As a solution, it proposes a social-based recommendation system for business

process modelling tools which improves the creation of formal process models. [14] embeds social software features, such as collaboration and wiki-like features, in the modelling and execution tools of business processes with the aim of encouraging people participate in the bottom-up design and execution of business processes. On the other hand [11] concerns of participation of end users in modelling processes, thus presenting an ad-hoc workflow system that focuses on non-intrusive capturing of human interactions. [16] takes another perspective focusing on the execution of business processes in the context of Web 2.0 and social software in a self-managed and decentralised environment. It examines the use of status feeds for supporting the execution of non-predictable business processes. [4] presents a process design methodology for addressing the extension of business processes with social features. In particular, they extend BPMN 2.0 with social roles, present a gallery of design patterns and finally propose WebRatio BPM as a technical framework for generating Social BPM applications from specifications encoded in Social BPMN.

While most of the existing work in the social BPM area focusses on adding social features to an existing BPM framework, *Processbook* intends to create a flexible personal process management environment within a social network structure. That is we make use of the BPM process view to be able to handle dependencies and constraints between tasks on one hand, while realising the three principles of social software via a goal oriented social network on the other hand. Our proposed approach can also be seen as an attempt towards implicitly involving people in processes.

4 *Processbook*: Towards Social Network-Based Personal Process Management

Processbook [8] is an architectural framework which introduces principles and guidelines for managing *personal processes* within a social network. *Processbook* aims to provide a goal-oriented social network whose users actively participate in the managing and sharing of personal processes. More specifically, (i) it supports users to, collaboratively, create and carry out personal processes, (ii) it allows users to utilise various data sources from the web to create *process fragments* as constituents of a personal process, (iii) it encourages users to share the intermediate results with others and receive feedback from them, and (iv) it creates links between people with similar goals so that they can share experiences with each other.

Figure 4.1 gives a conceptual overview of the Processbook system. It shows the different sections of the system and suggests how the social network module (top left) is integrated with the process modelling and process execution modules to form a social BPM framework. Upon registration in Processbook, users will be given a personal workspace called a process panel where they have the facilities to create processes and execute them. Once the user defines the goal/purpose of the process she wants to engage in, and any constraints (e.g., "the amount of funding needed to study PhD abroad"), she will be offered the option to join groups of people working on similar goals. She can now "follow" or "be followed by" other people, forming links and groups. *Processbook* associates a *process line* with each goal the user is attempting to acheive; the process

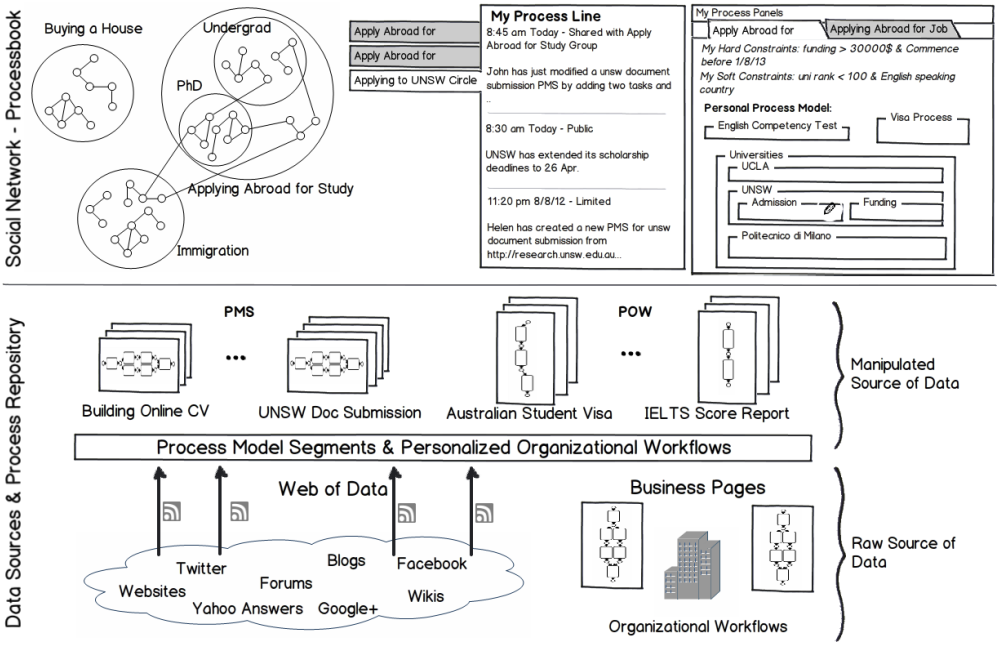


Figure 4.1: Processbook Conceptual Overview

line panel shows activity by all of the followed users as they progress towards their goal.

After the user registers their interest in a goal, the *Processbook* process modelling task (i.e., defining a personal process) continues with the user receiving a recommendation package consisting of business pages, web feeds and personal processes of her group mates. The recommendations are based on the goal and constraints specified, and the three components in the package are the main data sources from which the user may derive her own *Processbook* process. The bottom half of Figure 4.1 shows the data sources. *Processbook* users may use a combination of them to create their own personal process. Through the process panel (shown top right in Figure 4.1) a user can search through recommended items to find any process fragments that might be useful to complete her model. She is also able to browse her process line to figure out what actions other users have taken to manage their processes. The social network section in the figure illustrates circles of people in *Processbook* grouped based on their goal. Lines between users indicate that they are also following each other's work. By following a user, recommended items from that user will be prioritised and their actions could also be tracked in the process line.

The bottom half of Figure 4.1 shows the data sources. *Processbook* users may search the data sources and use a combination of them to create their own personal process. Raw data sources are either formal organisational workflows and business process models or informal guidelines that can be found over the web in blogs, forums, news pages, web sites, etc. Users of *Processbook* utilise these sources and create personalised workflows or process model segments respectively. Sources of data utilised in *Processbook* can be categorised as follows:

- **Business pages:** Organisations, institutions and business owners upload their business process and workflow models in special pages called business pages. These models can be downloaded and brought to the user's process panel to form part of her own personal process.
- **Web feeds:** Data spread over the web in blogs, forums, news pages, web sites provide a useful source of information for the descriptions of personal processes (e.g., a discussion forum on PhD applications, a university's scholarship application page). *Processbook* makes the data accessible for users in the form of web feeds. Users can search the feeds, subscribe to them and, importantly, can extract process fragments out of them and share them with others.
- **Other users' processes:** Instead of searching in raw data in feeds and business pages, users can rely on their followers or the groups they belong to, and follow the work of others. If they discover a useful process model developed by another user, they can extract the whole model or some parts of it and customise it to fit their own constraints. They can also integrate process models from two or more users.

4.1 Personal Process Model

The modelling interface of *Processbook* is based on a simple personal process model which is defined as a six-tuple (G, C, D, T, M, A), where G is the goal of the process, C is a set of constraints, D is a set of inputs and outputs (data), T is a set of tasks, M is a mapping that describes how tasks are connected and A is a set of annotations associated with the tasks. Each task T is either a simple activity or is a nested personal process model, thus models may be re-used in the construction of other larger models. In practice, tasks are drawn from several different sources in *Processbook*. Based on this, we can partition tasks into three sets: $T = POW \cup PMS \cup GFT$.

POW contains personalised organisational workflows. Each task in POW is derived from a standard business process model from one organisation in the Business Pages. PMS contains process model segments. Each task in PMS is derived from a source outside any organisational workflow, typically from a description of a process on a web site. Such tasks are typically found by users searching the web feeds. GFT, gap-filler tasks, are any other tasks that are necessary to guarantee the completeness of the process model, but are not included in POW or PMS for they are not part of any of the involved organisations or they are considered too variant and personal to be included in documents. Gap-filler tasks include those where the task:

- exists outside any organisation or institution and is also out of the scope of texts discussing related issues.
- is assumed to be too trivial to be modelled in business workflows or be mentioned in texts.
- may be handled in so many different ways that it makes the modelling too complicated or the texts too lengthy.

Later in Section 5.1 we propose a *ToDoList* model based on the idea of the personal process model introduced in [8].

4.2 Main capabilities of the proposed social network

There are four major capabilities that allow *Processbook* users to benefit from personal involvement in its social network: collaborative process modelling, knowledge capturing and sharing, social network based recommendation and utilising notifications.

Collaborative process modelling is realised by enabling users to copy process models created by peers to their process panel and modify them. Modification of a process model will result in a new version of that model. A set of different versions of process models which describe a single goal are kept in a pool and ranked based on the feedback given from users. Feedback is quantified using social factors such as:

- liking the model
- flagging the model as a faulty or incomplete one
- commenting on the model
- copying the model or its components to their process panel
- modifying the model by adding, deleting, renaming, annotating tasks or changing the task or data flow

Knowledge capturing and sharing aims to enhance the process management life-cycle by facilitating information exchange, which in turns speeds up modelling and execution decisions. The key point in information exchange is to find a method that automatically and non-intrusively captures users' modelling and execution experiences and then shares the captured data appropriately. Since users' actions are all performed in a web based social network framework, a web monitoring component in conjunction with a log analyser could provide users with the needed information. The extracted information could then be posted by a user and shared according to their preferences.

A built-in *recommender* system is required to filter the process knowledge repository and rank process model segments based on user's goals, preferences and constraints. An intelligent query processor module will implicitly and non-intrusively build a query from the user's goal, soft and hard constraints, past actions and the process execution state of the user's personal process model. Users would specify whether to search all the repository or limit the search to items shared by the group they belong to or by their friend circles. In addition to recommending process model elements, the recommender may also suggest a user to follow other users' process panels or to subscribe to a web feed or a business page.

Notification is used to reflect both regular changes in user-defined process models and policy changes in business environments. When an institution obsoletes a workflow, changes its policies or adds new criteria to one of its old workflows, a notification alarm should be sent to those who either have created POWs from that workflow themselves or copied an existing POW associated with that workflow. Similarly when a new PMS is extracted from a web feed or when an existing PMS is flagged as inappropriate, a notification message will be sent to those directly involved in creating that PMS and those who copied it to their process panels. Moreover when the top-ranked PMS in a pool of

PMSs - depicting the same goal - changes based on the users' feedback, it will be announced to users working on that pool to be aware of the new best practice PMS. It is also possible to get notification messages directly from one of the group or circle members stating new updates from her personal model.

5 Capturing and Sharing Personal Processes through Social Networks

In Section 4, we introduced *Processbook* as an architectural framework that gives principles and guidelines for managing personal processes within a social network. Hereafter we concentrate on two of the major capabilities of *Processbook* framework: knowledge capturing and sharing and social network-based recommendation.

Processbook aims (i) to make personal process management as effortless as possible for individuals and (ii) to utilise user participation to produce meaningful social collective data. The first step is to engage people to manage their processes through *Processbook*. Given that *Processbook* users are ordinary people rather than trained BPM designers or knowledge workers, they posses little or no knowledge or prior experience in process modelling and management. Therefore one of the main issues *Processbook* deals with is to find a way to provide support for individuals in managing their tasks while simultaneously taking advantage of their social participation to enrich its support. In Section 5.1 we propose a simple modelling interface based on the idea of ToDoLists to facilitate the modelling experience for non-technical users. The second step is to expedite the transfer of knowledge about processes among users. For this purpose, in Section 5.2 we propose a method to capture users process knowledge. Then in Section 5.3 we show how to share the process knowledge in form of process recommendation.

5.1 Modelling Interface

Traditional business process modelling lacks the required flexibility and agility when it comes to unstructured or ad-hoc processes. To break the rigidity of traditional modelling methods and to simplify their syntax for novice users, we propose a simple modelling approach that resembles the natural planning model our brain follows. [1] enumerates five steps that our minds go through to accomplish any task: defining purpose and principles, outcome visioning, brainstorming, organising, and identifying next actions. Similarly our proposed planning approach consists of five steps:

- Defining a *goal*: A goal is any desired result that requires one or more action steps. It is described in natural language and is mandatory for each plan e.g "Gain admission to a PhD degree in computer science at UNSW"

- Defining a set of *constraints*: Constraints are sets of parameters and criteria that further elaborate goals. They can be *global* to describe the general parameters e.g "Application deadline for PhD admission is 1 Dec 2013" or *local* to reflect personal visions on the goal e.g "field of study: BPM".

- Gathering all required *tasks*: Determining the set of required tasks is the first step towards the desired outcome. A task is a single unit of work in the boundaries of a particular goal. A goal is achieved when enough of the right tasks have been performed successfully and some outcomes have been created that closely enough match the initial vision.
- Elaborating tasks: A short-list of tasks are specified and elaborated by linking them to local constraints or by adding annotations to them.
- Identifying next task to do: The order in which users want to carry out their listed tasks is the final planning step and is repeated until all tasks are carried out or the desired outcome has been achieved.

Formally, we define a simple modelling interface called *ToDoList* to realise the idea of natural planning. Users may own one or more ToDoLists, one for each of their goals. However there is only one ToDoList for each user under a particular goal.

Definition 1 A *ToDoList* is a triple (G, T, C_G), where

- G is a statement of the *goal* in natural language
- T is a set of tasks to be done to achieve the goal
- C_G is a set of constraints on the goal

Each task gives a natural language statement of one activity to be completed in achieving the goal. A task is a pair (D, C_T), where D is a description which can be either a natural language description of an atomic task or a reference to another ToDoList and C_T is a set of constraints on the task.

5.2 Capturing process knowledge

Tasks are first class entities in our proposed ToDoList modelling approach. The set of tasks chosen by a user to accomplish a goal in conjunction with the order she carries out them is what we call process knowledge. To be able to capture the process knowledge, it is important to be aware of task life-cycle in our system. Figure 5.1 illustrates the *task state diagram* in *Processbook*. Each task, at any given time, could be in one of the "planning", "carrying out", "carried out" and "captured" states.

- **Planning** Tasks are identified by users themselves or given to them through a recommendation mechanism (Section 5.3) Once a task is identified, it will be considered in the planning state. The planning state is similar to drafting a ToDoList and resembles the brainstorming step of [1].
- **Carrying out** Tasks are brought to "carrying out" mode on a user's decision to perform them. When a user identifies the next action she wants to take, she can simply mark the task as being in carrying out mode. The task can be reverted back to planning mode when the user pauses or stops the execution of the task. Pausing a task will temporarily prevent it from being executed, meaning that it could be resumed later. But if the user wants to apply some changes to the task, she has to stop it first, apply the modification she wants and then start it again.

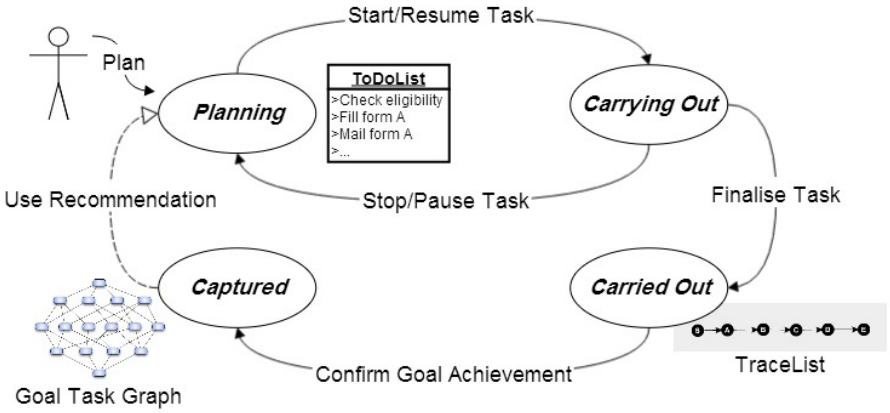


Figure 5.1: Task State Diagram

- **Carried out** When the user obtains the desired result from the running task, its completion timestamp is recorded and the task is considered finalised and in "carried out" mode. The carried out state consists of a set of completed tasks, ordered by their completion timestamp. Hence it could be regarded as a trace log for each ToDoList. It is expected that by the end of the personal planningall of the required tasks for an achieved goal are found in the carried out mode or more specifically in *TraceLists*. Associated with each ToDoList, there exists a *TraceList* that is opened after the first task in ToDoList has been carried out and is closed after the corresponding ToDoList terminates. A TraceList captures the execution of tasks in a ToDoList.

- **Captured** Once a user reaches her goal, the trace log will be aggregated with other users' traces for the same goal. If the user cancels her plan or does not achieve her specified goal, her trace log will be deleted. The aggregation of trace logs produces a *Goal-Task Graph* (GTG), where users' successful experiences for a particular goal are captured. The GTG captures the social production for a community of users who share the same goal.

TraceLists are formally defined as follows:

Definition 2 A TraceList is a triple (G, C_G, H_G) where

- G is a goal
- C_G is a set of constraints on the goal
- H_G is a set of history records of tasks and their properties

Each history record $H \in H_G$ is a quadruple (T, C_T, S_T, E_T):

- T is a task
- C_T is a set of constraints on the task

- S_T is the time when task was started
- E_T is the time when task was completed

A ToDoList terminates successfully if all its tasks have been carried out and the owner of ToDoList confirms that a desired outcome has been reached by performing those tasks. The TraceList of such a ToDoList execution is called a *complete* TraceList. Intuitively a set of complete TraceLists for a certain goal will give a useful insight on how to achieve that particular goal. We argue that each TraceList resembles a blog post describing a personal solution to reach the goal, while the aggregation of TraceLists is analogous to a Wiki that describes the general solution to reach that goal in different contexts and from different perspectives. *Processbook* realizes the concept of social production by merging all complete TraceLists of a goal into a graph structure called GTG(Goal-Task Graph).

Definition 3 A GTG is a weighted directed graph $GTG(G, V, E, W, C)$ where

- G is the goal for which complete TraceLists are aggregated
- V is the union of tasks in the complete TraceLists with goal G; the result set forms the vertices of GTG
- E is the set of edges of GTG; each edge indicates the order of execution between two tasks
- W is the set of weights associated with the edges in E, indicating the number of times a particular edge has been followed over all TraceLists of goal G.
- C is a set of constraints associated with the edges in E, indicating the circumstances under which a flow of tasks has occurred over all TraceLists of goal G.

For example, $T_i \xrightarrow{3,\{c_1 \vee c_2\}} T_j$ means task T_j preceded task T_i, three times in all complete TraceLists with either c_1 or c_2 mentioned as constraints of T_j in those complete TraceLists. Procedure 1 demonstrates the procedure of merging TraceLists into a GTG for goal G. The input of the procedure is a set of complete TraceLists that have G as their goal. `PrepareTraceList` orders tasks in each TraceList by their E_T attribute. More complicated metrics could also be applied to include the S_T and C_T attributes of history records as well, but we do not consider this further in this report.

Each precedence relationship between two tasks is then added to the graph as follows: `FindTask`(T_i, T_j, GTG) searches the graph for tasks T_i and T_j. If both tasks and their precedence relationship already exist in the GTG, we only need to increase the weight and update the constraint attributes of precedence. Otherwise if both tasks exist but they have not been connected, a new precedence should be added by (`AddPrecedence`) with its weight and constraints set. If only one of the tasks exists in GTG, we have to first add the missing task to the graph (`AddTask`), then add the corresponding precedence and finally set its weight and constraint attributes. This is similar to the case where both tasks are new to the GTG, with the minor difference that we have to add both tasks first.

Algorithm 1 Merging *TraceLists* to GTG

Input: TRL: A set of complete *TraceLists* having goal G
Output: GTG for goal G

```
PrepareTraceList(TRL)
for all tracelists trl in TRL do
    for all T_i → T_j in trl do
        Let w_ij = trl_weight_{T_i→T_j}
        Let c_ij = trl_constraints_{T_i→T_j}
        found = FindTask(T_i, T_j, GTG)
        switch found
            case both T_i and T_j were found in GTG
                if T_i → T_j exist in GTG then
                    Weight_{T_i→T_j} += w_ij
                    Constraints_{T_i→T_j} = Constraints_{T_i→T_j} ∨ c_ij
                else
                    AddPrecedence(T_i → T_j)
                    Weight_{T_i→T_j} = w_ij
                    Constraints_{T_i→T_j} = c_ij
                end if
            case only T_i was found in GTG
                AddTask(T_j, G)
                AddPrecedence(T_i → T_j)
                Weight_{T_i→T_j} = w_ij
                Constraints_{T_i→T_j} = c_ij
            case only T_j was found in GTG
                AddTask(T_i, G)
                AddPrecedence(T_i → T_j)
                Weight_{T_i→T_j} = w_ij
                Constraints_{T_i→T_j} = c_ij
            case neither T_i nor T_j were found in GTG
                AddTask(T_i, G)
                AddTask(T_j, G)
                AddPrecedence(T_i → T_j)
                Weight_{T_i→T_j} = w_ij
                Constraints_{T_i→T_j} = c_ij
    end for
end for
```

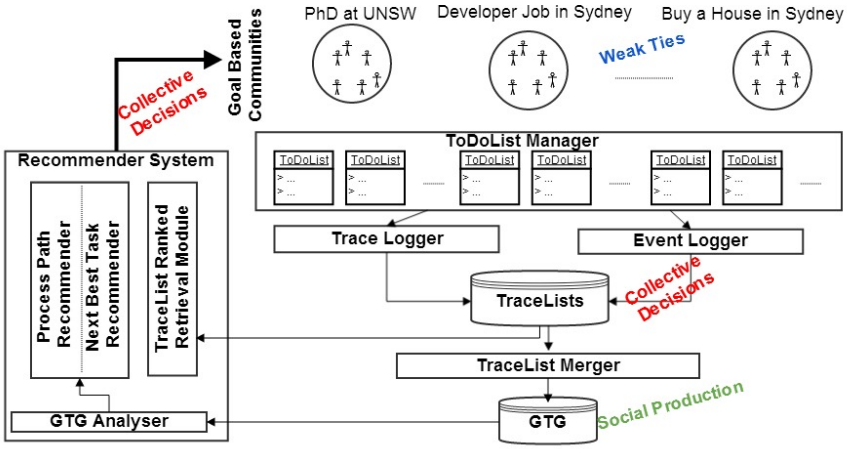


Figure 5.2: Realisation of social software principles in *Processbook*

5.3 Sharing process knowledge

Knowledge of achieving a goal, which we refer to as process knowledge, is captured when individuals carry out ToDoLists; this information is aggregated in the *GTG*. Process knowledge is the main artefact that is shared among users in *Processbook*. Process knowledge sharing happens via recommendations to targeted communities. Figure 5.2 shows how such a sharing mechanism is realised in *Processbook*. For each goal, a community is created. Users are considered members of a community once they start towards a goal. These goal-based communities, forming weak ties among users, are then used as a target group for recommendations. Users start planning their goals in the *ToDoList manager*. The executions of their plans are logged by *trace logger* in the TraceList database. Secondary data from their involvements that includes votes and comments on recommended items are also recorded by the *event logger*. The *TraceList merger* performs aggregation of TraceLists (peer products) of a goal into the *GTG* to produce social production. Collective decisions are realised by applying users' votes on TraceLists and providing recommendations.

The *recommender* system consists of two different modules separated by the data sources they utilise: (i) the *TraceList ranked retrieval module* that makes use of TraceList database and (ii) the *process recommender* that utilises the *GTG*. Ranked retrieval of complete TraceLists could be based on several different metrics:

- number of tasks in a TraceList; those with less tasks will be ranked higher implying they need less effort to achieve the goal
- execution time of a TraceList calculated by $Max(ET) - Min(ST)$; those with shorter execution time will be ranked higher implying they reach the desired outcome sooner
- popularity of a TraceList indicated by users' votes which reflects users' opinions on the usefulness and effectiveness of a TraceList

The Process Recommender uses two slightly different approaches: recommending the *next best task* or recommending a *process path*. Both approaches use the GTG as their data source. The next best task recommender gives users a task at a time while being dynamically modified as the GTG grows. The Process Path Recommender, on the contrary, provides the whole set of tasks needed to reach the goal in the form of a path. It does not reflect GTG changes unless the user explicitly asks for an updated recommendation. One advantage of the next best task recommender appears when a user wants to select a task manually, which is not the necessarily the next best one. In this case the recommender will consider calculating the best remaining path, taking into account what user has chosen so far. In both cases, the user may fully or partially accept the recommendation or may completely reject it and continue making her own plans.

The major advantage of the process recommender over the TraceLists ranked retrieval module is that it better reflects the knowledge of the crowd. This is because the next task or the process path given by the process recommender comes from merging different TraceLists and is considered to give the best possible solution from the crowd experience, while in the ranked retrieval system we are limited to the number of single TraceLists. The advantage of using a retrieval system based on single TraceLists is that what is finally recommended is already used by one or more users in real world. In the process recommender, the final recommendation path might have not necessarily been carried out before.

The basis of the process recommender is to find the minimum weighted path in the GTG. Weights in the GTG represent the popularity of a task flow but they have to be normalised so that Dijkstra's algorithm could be applied and the minimum weighted path from the starting task to the end task found. We also elaborate the weights of the GTG by taking into account users' votes in addition to the frequency measure. Therefore the final weight for an edge in GTG between tasks T_i and T_j is calculated as follows:

$$\alpha \times (10 - \frac{w_{ij}}{N} \times 10) + \beta \times (10 - vote)$$

where w_{ij} is the initial frequency weight of the edge between T_i and T_j, N is the number of complete TraceLists merged into the GTG, $vote$ indicates the average votes for the task flow, ranging from 0 to 10 (with 10 meaning the best), and α and β ($\alpha, \beta \in [0, 1], \alpha + \beta = 1$) are coefficients for popularity metrics and users' votes respectively. These coefficients are taken into account so that users would be able to customise recommended items. Increasing α will bias recommendation towards items which have been appeared more frequently in TraceLists e.g paths or tasks that majority of users have used. On contrary increasing β will guide users towards items that have a better average of users' votes regardless of the number of times they have been appeared in trace logs. While end-users are allowed to adjust these coefficients, *Processbook* is also supposed to find the optimum value for them through a learning mechanism. Design and implementation of such a mechanism is one of our future work.

It is likely that tasks that have been repeatedly used in complete TraceLists are excluded from the final recommendation due to existence of some unrealistic short paths in the graph. To avoid ignoring such tasks and thus improving the recommendation results, we define the concepts of *required tasks* and *required ratio*. The required ratio for task T_i is calculated by $^{w_{*i}}/_{N}$ where w_{*i} is the

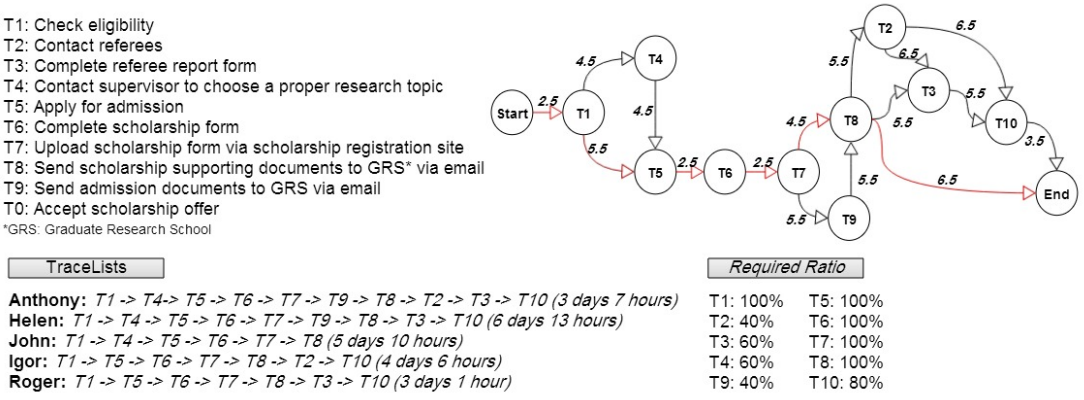


Figure 6.1: "Applying for UNSW PhD scholarship" TraceLists and *GTG*

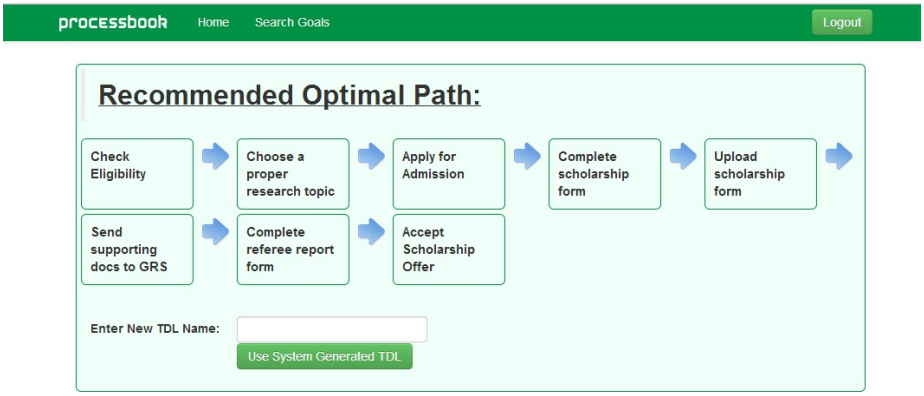
sum of the weights of ingoing edges to task T_i and N is the number of complete TraceLists merged into the *GTG*. A threshold for required ratio is set to force the inclusion of task T_i. Tasks whose required ratio is above the threshold are called required tasks, implying they are the necessary steps if the process is to achieve the goal. To take into account required tasks, the recommendation mechanism should also be modified. To this purpose, the *GTG analyser* shown in Figure 5.2 is responsible for marking required tasks in each *GTG*. The final recommended paths will be filtered to avoid ignoring required tasks.

6 Implementation and Example

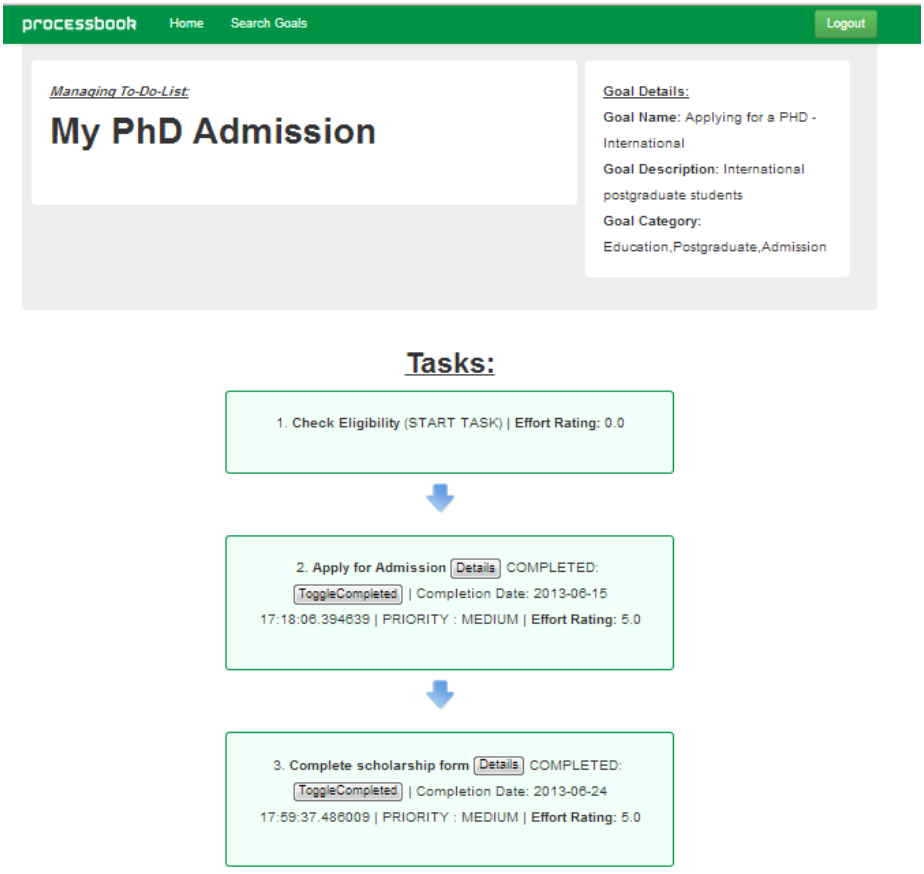
A prototype of the proposed system is implemented as a Java-based Web application and is available at our server [1]. In this section, we demonstrate our solution by presenting a simplified version of test cases we have run with *Processbook*. We have asked five students in our school to plan for the case "Applying for UNSW PhD scholarship" through the ToDoList modelling interface of *Processbook*. Based on this input, we show how the system generates recommendations for Ali, who also wants to apply for a PhD scholarship.

Users' knowledge and experience in planning the scholarship case has been captured in the TraceList database. Task descriptions, complete TraceLists and vote statistics are provided in Figure 6.1. Distinct tasks detected in our sample scenarios are: check eligibility (T_1), contact referees (T_2), complete referee report form (T_3), pay application fee (T_4), proceed with apply online (T_5), apply for admission online (T_6), send admission documents to graduate research school (GRS) via email (T_7), complete scholarship form (T_8), register in international scholarship system (T_9), upload scholarship form via international scholarship system (T_{10}), send supporting document to GRS via email (T_{11}), accept scholarship offer (T_{12}), complete acceptance form (T_{13}), find potential supervisor (T_{14}), finalise a PhD topic with the supervisor (T_{15}). *Processbook* aggregates TraceLists and builds the *GTG* shown in Figure 6.1. We assume

[1] Available at: http://pepe.cse.unsw.edu.au:8080/Processbook



(a) Process Path Recommendation



(b) ToDoList Manager

Figure 6.2: Screenshot of *Processbook*

that votes for each edge by users are equal (e.g 5 out of 10). Moreover since the effect of constraints is not yet implemented in the recommender system of *Processbook*, we ignore the constraint labels of the *GTG* edges. For simplicity

weights are calculated considering coefficients α and β set equally at 0.5.

In terms of number of tasks, John's TraceList ranks higher; in terms of execution time, Anthony's TraceList would be recommended to Ali. However, if Ali decides to use a process path recommender, he will be offered a new path which does not exist in either of those TraceLists. The recommended path excludes T_9 which represents a common misunderstanding among applicants. However, the path shown in Figure 6.1 also excludes T_3, T_4 and T_{10}, all of which are mandatory tasks according to UNSW policies. To avoid this undesirable elimination, we have to tune the required threshold to 50%, to enforce inclusion of such required tasks.After doing this, *Processbook* filters out the path illustrated in Figure 6.1 and returns the optimal path as can be seen in an screenshot of the system in Figure 6.2a. A user may then choose to accept this recommended path and start managing their ToDoList via the ToDoList manager as shown in Figure 6.2b.

7 Conclusion and Future Work

Our proposed solution for personal process management is to create a flexible process management environment within a social network structure. We maintain the process awareness of traditional BPM to be able to handle dependencies and constraints between tasks, but at the same time we follow the principles of social software: we have established weak ties in *Processbook* goal-based communities to break the top-down management paradigm of traditional BPM, we have facilitated social production to allow innovative solutions to appear, and we have utilised collective decisions to minimize the risks in choosing those solutions. To realise all of these, we have proposed a novel method for modelling personal processes based on the idea of ToDoLists and have implemented a mechanism to unobtrusively capture users' experience separately and then aggregate them in a graph structure that can be used as a source for process recommendation.

Our future work includes: (i) enabling context-aware process recommendation, (ii) enforcing trustworthy recommendation by utilising more social software features and (iii) enriching our capturing mechanism by enabling parallelism in task flows. As well, we intend to evaluate *Processbook* in real world scenarios by conducting comprehensive user studies. We believe that *Processbook* can be employed in many knowledge intensive domains in addition to personal process management.

Bibliography

[1] David Allen. *Getting things done.* penguin books, 2001.

[2] Yochai Benkler. *The wealth of networks : how social production transforms markets and freedom.* Yale University Press, 2006.

[3] Marco Brambilla. Application and Simplification of BPM Techniques for Personal Process Management. In Marcello Rosa and Pnina Soffer, editors, *Business Process Management Workshops*, volume 132 of *Lecture Notes in*

Business Information Processing, pages 227–233. Springer Berlin Heidelberg, 2013.

[4] Marco Brambilla, Piero Fraternali, and Carmen Vaca. BPMN and Design Patterns for Engineering Social BPM Solutions. In *Business Process Management Workshops (1)*, pages 219–230, 2011.

[5] Giorgio Bruno, Frank Dengler, Ben Jennings, Rania Khalaf, Selmin Nurcan, Michael Prilla, Marcello Sarini, Rainer Schmidt, and Rito Silva. Key Challenges for Enabling Agile BPM with Social Software. *Journal of Software Maintenance*, 23(4):297–326, 2011.

[6] Selim Erol, Michael Granitzer, Simone Happ, Sami Jantunen, Ben Jennings, Paul Johannesson, Agnes Koschmider, Selmin Nurcan, Davide Rossi, and Rainer Schmidt. Combining BPM and Social Software: Contradiction or Chance? *Journal of Software Maintenance*, 22(6-7):449–476, 2010.

[7] M.S. Granovetter. The Strength of Weak Ties. *The American Journal of Sociology*, 78(6):1360–1380, 1973.

[8] SeyedAlireza Hajimirsadeghi, Hye-Young Paik, and John Shepherd. Processbook: Towards social network-based personal process management. In Marcello Rosa and Pnina Soffer, editors, *Business Process Management Workshops*, volume 132 of *Lecture Notes in Business Information Processing*, pages 268–279. Springer Berlin Heidelberg, 2013.

[9] Paul Johannesson, Birger Andersson, and Petia Wohed. Business Process Management with Social Software Systems - A New Paradigm for Work Organisation. In *Business Process Management Workshops*, pages 659–665, 2008.

[10] Agnes Koschmider, Minseok Song, and Hajo A. Reijers. Social Software for Modeling Business Processes. In *Business Process Management Workshops*, pages 666–677, 2008.

[11] David Martinho and António Rito Silva. Non-intrusive Capture of Business Processes Using Social Software - Capturing the End Users' Tacit Knowledge. In *Business Process Management Workshops (1)*, pages 207–218, 2011.

[12] Michael Rosemann. Personal Process Management. Rosemanns blog, 2011. http://www.michaelrosemann.com/uncategorized/113/.

[13] Helen Schonenberg, Ronny Mans, Nick Russell, Nataliya Mulyar, and Wil M. P. van der Aalst. Process Flexibility: A Survey of Contemporary Approaches. In *EOMAS*, pages 16–30, 2008.

[14] António Rito Silva, Rachid Meziani, Rodrigo Magalhães, David Martinho, Ademar Aguiar, and Nuno Flores. AGILIPO: Embedding Social Software Features into Business Process Tools. In *Business Process Management Workshops*, pages 219–230, 2009.

[15] Don Tapscott and Anthony D. Williams. *Wikinomics: How Mass Collaboration Changes Everything*. Portfolio Hardcover, 2007.

[16] Simon Vogt and Andreas Fink. Using Status Feeds for Peer Production by Coordinating Non-predictable Business Processes. In *Business Process Management Workshops (1)*, pages 253–265, 2011.

[17] Ingo Weber, Hye-Young Paik, and Boualem Benatallah. Forms-based service composition. In *Service-Oriented Computing*, pages 627–635. Springer Berlin Heidelberg, 2011.