Worklr: Supporting and Capturing Business Processes from Knowledge Workers

David Martinho and António Rito Silva

[1] Instituto Superior Técnico – UL
[2] ESW - INESC-ID
`davidmartinho,rito.silva@ist.utl.pt`

Abstract. Worklr (Workflow enhanced with Live Recommendations) is a workflow tool that relies on one of the most known user experience (UX) pattern: the exchange of email messages. In organizations that do not have workflow systems guiding their business processes, knowledge workers rely on technologies like telephone, email or fax, to interact and attain their business process goals. Knowledge workers operate the business in a daily basis, and they know how to handle the most complex business situations. Worklr supports the capture of workers' knowledge following a design-by-doing approach. Workers attain business goals by producing data objects. The convergence of workers' behavior is fostered through a recommender system. This novel proposal leverages on data-driven workflow approaches.

Keywords: Ad-hoc Workflow, Operational Support, Recommender Systems

1 The Worklr Tool Approach

Our approach to capture knowledge workers' tacit knowledge is based on an ad-hoc workflow system which implements conversational acts, and where business goals are achieved when a set of data objects is produced [1]. Knowledge workers are responsible to produce data objects towards the complete generation of the information required for the process goal achievement. Additionally, knowledge workers can establish conversational acts to request the production of data objects by their co-workers. Associated to a request, the initiator may include some input data objects, and the executor is expected to reply with output data objects that she produced to fulfill the request. The flow of conversational acts (i.e. requests), supported by the ad-hoc workflow system, is driven by the knowledge workers' knowledge of their own organizational responsibilities and their co-workers responsibilities: they know who is responsible to produce what.

Consider a Travel Process case where an *employee* wants to go on a business travel to Beijing. First, he creates a process instance named `Travel Process`. As he initiates such process instance, the employee is automatically assigned to execute the first request of the process, a.k.a. root request. It is within this root request, automatically named after the process name, that the employee will create any necessary sub-requests.

The employee knows that the secretary is responsible to handle his trip, but before he sends her a sub-request, he creates a set of data objects that he knows the secretary will need. Therefore, the employee creates a data object labeled `Destination` and fills its value with Beijing, two data objects labeled `Departure Date` and `Returning Date` with the values 25/08/2013 and 30/08/2013, respectively, and a data object labeled `Motivation` where he states that he wants to attend the BPM2013 conference. For now, these four data objects are defined as the creation context of the request identified by the request label `Travel Process`[3].

Having the necessary data objects, the employee is in condition to send a sub-request to the secretary. To do so, he creates a new sub-request, chooses the secretary's queue as destination, names the request `Travel Request`, adds a little description to communicate the motivation for that request as a commentary, and selects all the data objects he created before: `Destination`, `Departure Date`, `Returning Date` and `Motivation`. As the employee sends the sub-request, that request is published in the secretary's work queue.

As one secretary claims the request for execution, she is proclaiming herself as the executor of that request, being able to see the data objects provided by the request's initiator. Having the information provided by the employee, the secretary calls some travel agencies to request some hotel and flight prices. However, before booking any hotel room or flight, the secretary needs an authorization from the *supervisor*. Similarly to the employee, the secretary also knows that the supervisor needs to know the trip details and the best tender value she got from the agencies. As such, she creates a data object named `Tender` and fills it up with the best price she got from the travel agencies. Afterwards, the secretary creates a new sub-request, which she addresses to the supervisor's queue, labels the request `Travel Authorization`, writes a comment on what she needs from the supervisor, and, as input, she selects the data objects labeled `Motivation`, `Departure Date`, `Returning Date`, `Destination` and `Tender`.

In the same way, as the request is sent, it is published in the supervisor's queue. As one of the supervisors claims this request for execution, he can see the data objects that the secretary attached as input of the request. Based in this data, the supervisor executing the request decides to either authorize or not the travel. To let the secretary know that she can proceed with the hotel and flight booking, the supervisor creates a new data object labeled `Authorization` and defines its value as *Granted*. Then, the supervisor is in condition of responding to the request and let the secretary know of his decision. Notice that the secretary will only see the data object if the supervisor explicitly selects it when he responds to the request. After the supervisor responds to the request, he cannot create any sub-request or data objects in that context[4].

[3] The root request is created automatically and named after the process name

[4] Although this requirement appears to be limitative, its for correctness purposes

After the supervisor replies to the `Travel Authorization` request, the secretary can see the responded data object `Authorization` along with its `Granted` value. Based on that positive value, the secretary calls the travel agency and books the hotel and flight, obtaining the respective reservation number and flight ticket number. With this information, she creates two data objects labeled `Hotel Reservation` and `Flight Number` and fills in the respective information. After this, she is in condition to respond to the original request initiated by the employee, providing both the `Hotel Reservation` and `Flight Number` data objects.

As the secretary responds to the request, the employee can see the hotel and flight information contained in the data objects included in the response. Since he needs no more information, and has no pending or executing sub-requests, the employee completes the process.

With the process completed, the employee is essentially saying that the process attained its goal, which consists in getting the travel approved and both the hotel and flight information. During the process execution, several data objects were produced and drove the process instance. Worklr considers all the created data objects and stores their set of labels, along with their cardinality, as a business process goal. This means that completing the process depicted in the example above results in a process goal with the following eight data object labels:

- 1 x `Destination`
- 1 x `Motivation`
- 1 x `Departure Date`
- 1 x `Returning Date`
- 1 x `Tender`
- 1 x `Authorization`
- 1 x `Hotel Reservation`
- 1 x `Flight Number`

Having this process goal is important to guide future business process instances under the same label, i.e. process instances labeled `Travel Process`. Worklr knows, that if in the past a particular process goal was attained, it is likely that same goal will be attained again in a future execution of that kind of process. Hence, along with other contextual information gathered during the execution of a business process instance, Worklr uses such business process goal information and inspects the current state of the process to recommend the creation of new data objects or sub-requests. Such recommendations are based on a process goal and the current execution context, i.e. the user's roles, the labels of data objects available to the user, and the overall process' data object labels.

Therefore, apart from providing operational support, Worlr is also capable of storing previous executions in a structured way, which contain information to guide future process instances with the same label. Nevertheless, as new business situations occur (e.g. if the supervisor refuses the authorization), the Worklr ah-hoc aspect supports that change, i.e. it does not enforce recommendations, and stores the new attained process goal. That new process goal is taken into consideration in the following executions of processes labeled `Travel Process`.

1.1 Features

The tool is in its first prototype stage, and its source code is not available yet due to intellectual property restrictions inherent to the PhD status of one of the authors.

The following list highlights a set of core functionalities of the Worklr application, regarding operational support:

– Create new business process instances.
– Create data objects within the execution of a request.
– Edit data objects created within the execution of a request.
– Create sub-requests and publish them into user and role queues.
– Claim requests received in the Inbox for execution.
– Communicate with the initiator of the executing request.
– Communicate with the executor of the sub-request.
– Cancel pending and executing sub-requests.
– List all sub-requests initiated and see their respective status (pending, executing, completed, canceled).
– Provide data objects as input of sub-requests.
– Respond to executing requests, specifying which data objects should be included in the response.
– Complete business process instances.

Apart from operational support, Worklr saves executions in the form of *request templates*, which are entities that abstract the different completed requests by storing some contextual information. A request template is therefore composed by:

– an *initiator role context*, which is essentially the set of organizational roles played by the initiators of that kind of request.
– an *input data context*, which is the set of data object labels, and respective cardinality, of the data objects provided as input in that kind of request.
– a *creation data context*, which is also a set of data object labels, and respective cardinality, created in that kind of request.
– an *output data context*, which, analogously, is the set of data object labels, and respective cardinality, responded in that kind of request.

Along with all the business process goals attained under a particular process label, e.g. `Travel Process`, the Worklr tool provides a recommender system that analyses the context of execution of the current user, and computes a set of recommendations of both the labels of the data objects that should be created, and the labels of the sub-requests that can also be created. To each recommendation is associated a score, as discussed in [2], from which the list of recommendations is ordered accordingly.

Hence, the list of Worklr features is extended by this recommender system, and the tool can also:

– Provide sub-request recommendations based on the current execution context.
– Provide data creation recommendations based on the current execution context.

As users perceive recommendations as useful, they are re-using the labels and behavior from other organizational parties playing a similar set of roles. Additionally, although the number of process, request and data object instances increase as more cases are handled, if recommendations are followed, the number of request templates and data object templates converges and recommendations become more accurate.

1.2 Experiment

As shown in the screencast of the tool[5], Worklr is in a functional state enough to conduct an experiment. In [3], we discuss the results of an experiment that we recently conducted, where we gathered some important feedback.

In the experiment, we had 13 participants and 1 confederate. The confederate would initiate new process instances and the root request, while the 13 participants were distributed across 4 distinct organizational roles that should cooperate to attain a particular business process goal. The business process used in the experiment was similar to the one exemplified here, but with more business rules.

A last comment on the experiment: we gathered some important results from the tool as we perceive some convergence in the use of labels by different participants. In the cold-start of the experiment, users would use different labels to identify requests and data objects, but after 16 business process instances, we identified some interesting convergence results.

Acknowledgement

This work was supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under project PEst-OE/EEI/LA0021/2013.

References

1. Martinho, D., Silva, A.: Non-intrusive capture of business processes using social software. In: BPM Workshops, pp. 207–218. Springer (2012)
2. Martinho, D., Silva, A.: A recommendation algorithm to capture end-users' tacit knowledge. Business Process Management 7481, 216–222 (2012)
3. Martinho, D., Silva, A.R.: An experiment on capturing business process from knowledge workers, submitted to the BPMS2 Workshop at BPM2013

[5] http://vimeo.com/user4862900/worklr-demo-bpm2013 - Password: BPM2013