Optimized Cross-Organizational Business Process Monitoring: Design and Enactment

Marco Comuzzi[a,*], Irene Vanderfeesten[a], Tingting Wang[a]

[a]*School of Industrial Engineering, Eindhoven University of Technology, P.O. Box 513, 5600MB, Eindhoven, The Netherlands*

Abstract

Organizations can implement the agility required to survive in the rapidly evolving business landscape by focusing on their core business and engaging in collaborations with other partners. This entails the need for organizations to monitor the behavior of the partners with which they collaborate. The design and enactment of monitoring, in this scenario, must become flexible and adapt as the collaboration evolves. We propose an approach to flexibly design and enact cross-organizational business process monitoring based on Product-Based Workflow Design. Our approach allows organizations to capture monitoring requirements, optimize such requirements, e.g. choosing the monitoring process with lowest cost or highest availability, and enacting the optimal monitoring process through a service-oriented approach. Optimization, in particular, is made efficient by adopting an Ant-colony optimization heuristic. The paper also describes a prototypical implementation of our approach in the ProM framework.

Keywords: Business Process Monitoring, Product-Based Workflow Design, Ant Colony Optimization, Service-based Systems.

1. Introduction

Information is produced and travels across the world at an ever increasing speed making the world increasingly connected. In such a connected world, nothing remains local and any economically relevant event in a specific geographical area, e.g. economic downturns, natural disasters, or political upheavals, may hold significant effects on organizations on the other side of the world. Only agile businesses, that can flexibly redesign or reconfigure their operations, can survive in such an environment. Organizations often implement agility and flexibility through simplification, focusing on their core business, and engaging in collaborations with partners to maintain and possibly improve the required level of quality and cost-effectiveness.

Collaboration, in the form of delegation, outsourcing, or partnership, requires control. As stated by agency theory, when principals delegate tasks to agents, they have to prevent the agents' possible opportunistic behavior and to detect the possible unsuitability of agents to execute the task assigned to them [14]. In a governance perspective, collaboration entails the need for control. The technology supporting collaboration must therefore support risk management in collaborative settings. This is especially true in highly regulated industries, such as banking or healthcare, where companies strive to achieve the balance between internal simplification, increasing collaborations with external partners, and compliance to regulations and service levels required by the industry [23]. In such an environment, consumer trust can dramatically decrease if organizations do not guarantee the declared service levels by continuously monitoring their business environment. Control in collaborative settings represents a challenging task, since many of the assumptions

*Corresponding author

Email addresses: `m.comuzzi@tue.nl, Tel: +31 40 247 2183` (Marco Comuzzi), `i.t.p.vanderfeesten@tue.nl, Tel: +31 40 247 4366` (Irene Vanderfeesten), `tingting.t.wang@gmail.com` (Tingting Wang) characterizing intra-organizational settings, e.g. relatively stable business process specifications and homogeneous IT landscapes, do not longer hold. In other words, control and, therefore, the technology required to implement it, must adapt to dynamically changing business and IT environments.

Control in collaborative settings is first established through contracts, which define the agreements among partners, making them liable if the specified obligations are not met during the enactment of the collaboration. Contracts, however, represent a passive method for coordination, and they should always be coupled with the active monitoring of the actual enactment of the collaboration [26]. Monitoring in cross-organizational settings should reflect the agility and flexibility typically characterizing such contexts. The monitoring of cross-organizational processes should be flexibly redesigned as the collaboration evolves, e.g. processes (or parts of them) are outsourced to new partners, partners are substituted, or new contracts to regulate the collaboration are deployed.

In this context, this paper proposes an approach and a prototype for designing and enacting flexible monitoring of cross-organizational business processes. The approach is based on Product-Based Workflow Design (PBWD), a scientifically-grounded method to design business processes using a clean-sheet approach [25, 31, 33]. We use PBWD to design the monitoring process, that is, the orchestration of sources of monitoring information in the collaboration to detect the correct execution of cross-organizational business process. In order to do this, we extend PBWD with features to derive the best monitoring process for a given partner in the collaboration in non-functional terms, ~~i.e.~~ monitoring at the lowest cost, with the highest availability of monitoring information, and with the highest quality of monitoring information, or a combination thereof. In particular, we propose an Ant Colony Optimization (ACO)-based heuristic to maximize the utility function of the monitoring stakeholder under generic constraints specified on cost, availability, and quality of the monitoring information.

To the best of our knowledge, our approach is the first to consider the issue of designing *optimal* business process monitoring processes and to apply evolutionary techniques for obtaining a solution to the problem. The paper also contributes the prototypical implementation of a new plugin of the ProM framework specific for ACO-based optimization in PBWD and a Web service-based system for the enactment of optimal monitoring processes.

The paper is organized as follows. The next section contextualizes the problem of cross-organizational process monitoring. Then, Section 3 describes our approach from the methodological and architectural perspectives. Section 4 discusses the implementation of the approach, while conclusions are drawn in Section 5.

2. Related Work on Process Monitoring

Monitoring of cross-organizational business processes has been investigated, from a requirements engineering perspective, in [10] and [21]. In order to achieve a successful collaboration, both papers stress the importance of process- and communication-oriented mechanisms to transmit relevant information to interested parties across the network. Similarly, [6] considers the need to define external information requirements, i.e. information required by a consumer from its providers, in order to correctly monitor and enforce a multiparty contract. The business process management literature approaches the issue of cross-organizational business process monitoring mainly from a modeling point of view. Several are the examples, in fact, of approaches and methodologies to capture cross-organizational business processes requirements, and, more specifically, monitoring or process tracking requirements (e.g. [8]). Concerning the monitoring execution, architectural support for cross-organizational business processes has been considered in the CrossFlow [18] and CrossWork [19] projects. Their focus, however, is on the single organization and on how to (automatically) derive the infrastructure required to capture monitoring information.

Our approach fills the gap between the modeling and architectural approaches to cross-organizational business process monitoring. We combine a method for capturing monitoring requirements, based on PBWD, with the automated generation of the monitoring infrastructure, obtained as the orchestration of monitoring services. When monitoring requirements are stable, our approach allows their optimization and, consequently, the derivation of an optimized monitoring infrastructure for the partners involved in the collaboration. As monitoring requirements evolve, our approach allows the seamless reconfiguration of the monitoring infrastructure.

Monitoring has been extensively investigated also in the context of Web-service based business processes. Web service-based processes can be intrinsically considered cross-organizational, since each orchestrated Web service can in principle be exposed by a different organization. In this context, we can distinguish between intrusive and non-intrusive monitoring [20]. The former interleaves service and monitoring activities at runtime, whereas the latter separates the business from the monitoring logic, since information relevant for monitoring can be captured non-intrusively while a process is executing. Our approach builds on non-intrusive monitoring, encapsulating relevant monitoring services in monitoring Web services. These are orchestrated according to the logic captured in the monitoring requirements.

The only applications of evolutionary optimization techniques to the case of (Web-service based) business process management can be found in the context of Web service composition, that is, choosing a set of concrete Web services among a set of candidates to fulfill a given process specification, under a set of constraints on cost and/or QoS. Genetic algorithms have been applied in [22], ant colony optimization in [35], and hybrid evolutionary techniques [34, 4]. We were not able to find specific applications of evolutionary optimization techniques to the case of process monitoring optimization.

To summarize, this paper tackles a relevant problem in the area of cross-organizational business process management, i.e. the definition of *optimal* monitoring processes, given the requirements of the stakeholders involved in the collaboration. To solve our problem, we use a specific evolutionary optimization techniques, i.e. Ant Colony Optimization (ACO). A more thorough discussion of the suitability of ACO for our problem is presented later after having introduced our architectural framework and the optimization problem.

This paper extends a preliminary version of the framework presented in [7] by discussing the implementation of the framework within the ProM framework and by proposing an Ant Colony Optimization-based solution to the optimization of monitoring processes.

3. A Framework for Cross-Organizational Process Monitoring

Our framework is shown in Figure 1. It combines the steps for the design and enactment of monitoring processes and the required architectural support. The framework comprises two types of actors: the Monitoring Stakeholder (MS) and the Service Providers (SPs). MS is the actor willing to build a monitoring process for a given collaborative business process. In most cases, MS is the customer of such process, but generally it can be any actor in the collaboration having interest in monitoring the execution of the process, e.g. a 3rd party auditor, a public agency, or one of the collaborating actors. SPs are the actors collaborating for provisioning the business process that MS needs to monitor.

SPs capture monitoring information on their internal infrastructure and make it available to MS to build a customized monitoring business process. For instance, MS can use the progress information made available by SPs to achieve a global vision on the status of its orders along a complex supply chain. In the information system or process engine of a service provider SP, monitoring information can be captured from various sources and through different mechanisms, ~~such as (i)~~ native APIs of the SP's ERP system, e.g. SAP monitoring architecture, workflow or BPEL engine, or IBM Websphere business monitor [9]~~, and (ii)~~ ad-hoc instrumentation of SP's business process management infrastructure, e.g. through the development of event captors or other online process inspection techniques [20]. Irrespective of how monitoring information is captured, SP makes such information available to MS through a Web service, exposing an operation for each information product that may be required by MS for monitoring purposes.

In a service-oriented architecture, SPs publish their monitoring services in a service registry (Step 0 in Figure 1) and MS browses the registry to get service descriptions and to compose the complete monitoring information from its components~~, e.g. t~~he progress information on a holiday booking may be built from the progress information of the flight booking, the hotel booking and the travel insurance booking. Such a hierarchical structure of monitoring information is similar to the concept of a product data model from the PBWD approach, and is therefore modeled in a monitoring product data model, i.e. a MON-PDM. The composition of the MON-PDM is supported by a component of our framework, i.e. MON-PDM Design (Step 1). As we will show, there can be different ways to satisfy monitoring requirements of MS characterized by different values of non-functional properties, such as cost, and the MON-PDM allows for specifying these different ways of obtaining monitoring information.

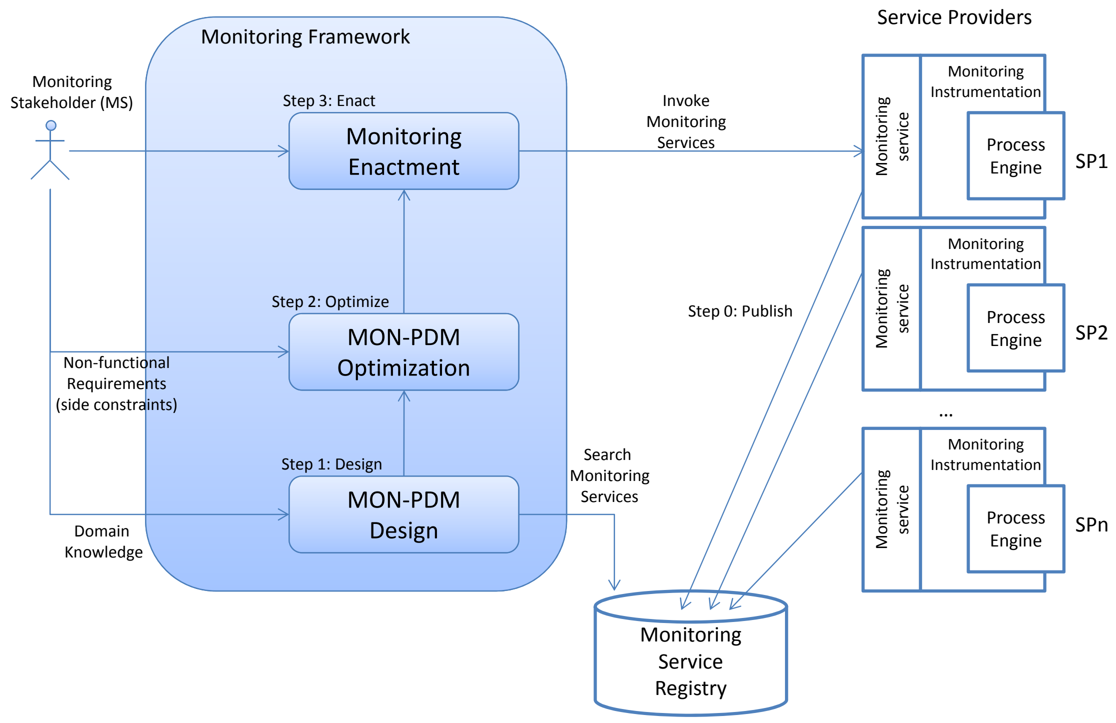


Figure 1: The Cross-Organizational Business Process Monitoring Framework

The second step in our framework therefore involves the optimization of MON-PDM. Optimization concerns finding the optimal path in MON-PDM according to the requirements of MS (Step 2). For instance, SPs can attach price information to their monitoring service in order for MS to build a monitoring process with minimum cost. Eventually, the selected path in MON-PDM can be executed by MS to monitor the execution of the business process. Execution of the monitoring process involves navigating the MON-PDM along the optimal path in MON-PDM through the invocation of the Web services implementing the operations belonging to such a path (Step 3). In the remainder these three steps are further elaborated upon.

3.1. PBWD for capturing Cross-Organizational Process Monitoring Requirements (Step 1)

PBWD is a scientifically grounded method for business process (re)design [25]. The focus of this method is on the design of processes that deliver informational products, the so-called workflow processes. The PBWD methodology takes the structure of the informational product, which is described in a Product Data Model (PDM), as a starting point to derive a process model. Informational products are, for instance, a decision on an insurance claim, the allocation of a subsidy, or the approval of a loan. Based on the input data provided by the client or retrieved from other systems, the end (informational) product is constructed step-by-step. In each step new information is produced based on the specific data available for the case. Over recent years, PBWD has shown to be a successful business process (re)design method [25, 32].

The PDM describes the composition of the informational product by explicitly modelling the information elements and the operations that compile new information products based on input information products. The information needed to monitor a business process may be modelled in a similar way. The PDM allows to easily model the different ways a certain information product can be composed, or the different sources an information product may originate from by including alternative paths to *assemble* the final informational product. This is illustrated with an example after elaborating on the formal definition of a PDM.

3.1.1. Formal definitions

In our framework, a MON-PDM is a tuple $\langle IP, OP \rangle$, where $IP = \{ip_x\}_{x=1,\ldots,X}$ is the set of X information products ip_x and $OP = \{op_i\}_{i=1,\ldots,I}$ is the set of I operations op_i. The operations determine how one or more information products are combined to produce a new information product, i.e.

$$OP \subseteq \wp(IP) \times IP$$

A special information product *root* constitutes the root of the monitoring PDM, that is, the correct composition of monitoring information for MS obtained by the monitoring services.

An operation op_i in a PDM leads to one and only one information product p_x. Such an information product is the output O of the operation op_i, i.e. $O(op_i) = ip_x$. The set of information products required to execute an operation op_i is the input set IS of op_i, i.e., given $ip_x = O(op_i)$, $IS(op_i) = \{ip_y\}$ s.t. $\{\{ip_y\} \times ip_x \in OP\}$. An operation is a *leaf* operation when its input set IS is empty.

Because of alternative paths in the PDM, an information product ip_x can be produced by more than one operation $op_i \in OP$. In our framework, such set of operations is the set of preceding operations $PREC(ip_x)$, with $PREC(ip_x) = \{op_j \in OP : O(op_j) = ip_x\}$.

A complete path p is a subset of operations in the PDM leading to the production of the root information product, starting from a set of leaf operations. Formally, a complete path p satisfies the following three conditions:

$$p \subseteq OP \tag{1}$$

$$\exists_{op_i \in p}[O(op_i) = root] \tag{2}$$

$$\forall_{op_i \in p}[IS(op_i) = \emptyset \vee \forall_{ip_x \in I(op_i)} \exists_{op_j \in p}[O(op_j) = ip_x]] \tag{3}$$

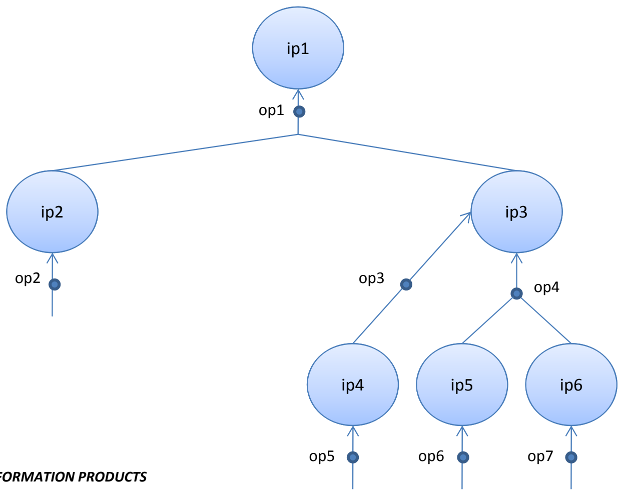
Condition 3, in particular, states that if an operation op_i belongs to p, then all operations leading to products belonging to the input information products of operation op_i have to be part of p, unless the input of an operation is the empty set, i.e. the operation op_i is a leaf operation. This condition allows the recursive definition of a complete path, from the root element to the leaf operations. In the remainder of the paper we refer to P as the set of all possible complete paths p in a given PDM.

Figure 2 shows an example of a simple monitoring PDM referring to a customer (MS) monitoring a make-to-order process that the provider (SP1) has outsourced to two contractors (SP2 and SP3). The root element ip_1 of the PDM represents the correct monitoring information required by the customer. This is the combination of information on the status of the order (ip_3) and on the expected delivery date (ip_2). While the expected delivery date can only be produced by the provider, status information can be obtained directly by the provider (ip_4) or by combining status information made available by the two contractors (ip_5 and ip_6). It can be the case, in fact, that status information of an order made available by the provider is less accurate but cheap, for instance belonging to the set $\{waiting, in\_progress, terminated\}$. Status information produced by contractors may be more detailed and up-to-date and, consequently, more expensive. If an order is in progress, for instance, a contractor may also provide the percentage of processing activities already completed. Knowing more about the internal progress of contractors, customers may be able to better synchronize their internal processes that rely on the outsourced process. The path $p_1 = \{op_1, op_3, op_5\}$ is, for instance, a complete path. The path $p_2 = \{op_1, op_2, op_4, op_7\}$ is not complete, since it does not include op_6, which would be implied by the presence of op_7 and by condition 3 of the above definition.

3.2. Optimal Paths in MON-PDM using Ant Colony Optimization (Step 2)

We define the optimal path in MON-PDM as the path that satisfies best the non-functional requirements of MS. The requirements of MS are captured by the following optimization problem.

We consider a set of C independent quality dimensions, indexed by $c = 1, \ldots, C$. For each dimension, we define a partial utility function $v_c(p)$. The utility $V(p)$ of a path p is given by the weighted sum of the partial utility functions, that is $V(p) = \sum_c [w_c \cdot v_c(p)]$. Our optimization problem ($P1$) concerns finding the



MONITORING INFORMATION PRODUCTS

id	*Information product*	*Produced by*
ip_1	Customer monitoring information	composed
ip_2	Expected delivery date	SP1
ip_3	Progress information	composed
ip_4	Provider progress information	SP1
ip_5	Contractor 1 progress information	SP2
ip_6	Contractor 2 progress information	SP3

OPERATIONS

id	*cost()*	*avai()*	*qual()*
op_1	0	1	1
op_2	0,8	1	0,5
op_3	0	1	1
op_4	0	1	1
op_5	0,2	1	0,3
op_6	0,7	1	0,8
op_7	0,3	1	0,7

Figure 2: Example of MON-PDM with non-functional characterization of operations

complete path p in the set of all possible paths P that maximizes the weighted sum of the partial utilities, given a set of side constraints:

$$P1\colon \max_{p\in P} V(p) \;\; s.t.$$
$$v_c(p) \geq T_c, \;\; c = 1,\ldots,C$$

For illustration purposes, in this paper we consider the non-functional requirements cost, availability, and (data) quality to characterize MS requirements ($C = 3$). As shown in the following, their partial utility functions $v_c(p)$ and, consequently, the utility function $V(p)$ are normalized between 0 and 1. Note that our framework could accommodate other possible non-functional requirements, as long as their normalized utility function is defined.

Cost ($cost$, $c = 1$) represents the cost of the execution of a path p in MON-PDM. It is the cost sustained by the actors in the collaboration to execute the operations in the path p to provide monitoring information. Costs are summative, that is, the cost of a path p is determined as the sum of the costs of operations op_i belonging to p [1, 24]. From a utility perspective, cost is a negative requirement, that is, the utility of an operation for MS decreases as cost increases. In formulas, the partial utility function for cost is:

$$v_1(p) = 1 - \frac{\sum_{op_i \in p} cost(op_i)}{|\{op_i \in p\}|}$$

where $cost(op_i)$ is the cost of the individual operation op_i belonging to p and, by construction, $0 \leq$

$cost(op_i) \leq 1$, for all op_i.

Quality ($qual$, $c = 2$) represents the quality of the path p as perceived by MS. Quality of monitoring information should be intended as fit for use, i.e. the ability of a piece of information to satisfy the monitoring information requirements of its user. Fit for use quality of operations op_i belonging to a path p is aggregated using the minimum value of data quality of all operations op_i [1]. Hence:

$$v_2(p) = \min_{op_i \in p} qual(op_i)$$

where $qual(op_i)$ is the quality of the individual operation op_i as perceived by MS, with $0 \leq qual(op_i) \leq 1$.

Availability ($avai$, $c = 3$) represents the probability that the root element of MON-PDM is produced, that is, it becomes available after the execution of the path p. This is obtained by multiplying the probability of producing the results of all operations op_i in the path p [1, 24]:

$$v_3(p) = \prod_{op_i \in p} avai(op_i)$$

where $avai(op_i)$ is the probability that the individual operation op_i will produce its result, i.e. the probability that $O(op_i)$ is produced by op_i, with $0 \leq avai(op_i) \leq 1$.

Figure 2 shows possible values for the non-functional requirements of operations in our running example. For instance, the path $p_1 = op_1, op_2, op_3, op_5$ is the minimum cost path, i.e. the one that maximizes $V(p)$ with $w_1 = 1$ and $w_2 = w_3 = 0$. Specifically, $V(p_1) = 0,75$. If a constraint on quality is introduced, e.g. $v_2(p) > 0,4$, then $p_2 = op_1, op_2, op_4, op_6, op_7$ becomes the optimal cost path with respect to this constraint, with $V(p_2) = 0,64$, and $v_2(p_2) = 0,5$. In our framework, the service providers publish cost, quality, and availability information for each monitoring services in the service registry.

Since the MON-PDM is a hypergraph, the optimization problem can be seen as a (hyper)graph optimization problem. In particular, let us consider only the cost requirement, i.e. $w_2 = w_3 = 0$, without side-constraints. Since costs are summative, the utility of an operation can be seen in this case as the distance between two information products that the operation connects in the MON-PDM. Thus, our optimization problem becomes equivalent to a shortest-path problem, where the shortest path in MON-PDM links the root information product to an imaginary node that connects all leaf operations. Hypergraph shortest-path optimization with summative weights can be solved in polynomial time (see the algorithms in [2]). When quality and availability are considered, however, the weight of arcs in the graph are no longer summative and, therefore, our problem becomes NP-Hard [2]. In this paper we propose a heuristic solution to our optimization problem, based on ACO [13].

We will demonstrate that our heuristic algorithm is able to evaluate the optimal path p in reasonably complex MON-PDMs in a time that is negligible if compared to the time required for determining the optimal path through exhaustive search.

3.2.1. Ant Colony Optimization of PDM

ACO is a meta-heuristic discrete optimization strategy that mimics the behavior of ants while searching for food [2, 11]. Initially, ants wander randomly. After having found a source of food, they will return to the colony to give the news to fellow ants, leaving a pheromone trail on their path. When new ants will be looking for food, they are not likely to travel randomly, but they will follow a path with high level of pheromone, reinforcing it, since they know it will bring them to a source of food. Over time, however, pheromone starts to evaporate. The pheromone is more likely to evaporate on long (or, as in our case, low utility) paths, since it takes more time to an ant to walk through them. Conversely, the trail is likely to remain fresh on short (high utility) paths. Hence, short paths are the ones that will be marched over more frequently by ants and for which the pheromone has less time to evaporate. Following the pheromone trail dynamics, after some time the ants' behavior will converge to identify the optimal path between the colony and the source of food. As discussed in more detail later in this section, ACO optimization is also able to comfortably manage side constraints on additional quality criteria [2, 11].

Although there exist extensions for continuous optimization [28], the ACO meta-heuristic has been developed primarily to solve discrete optimization problems [11]. ACO has been applied to several real world problems [16], such as project scheduling or network routing and load balancing [27], which are combinatorial in nature, as the problem we consider in this paper. Among the approximate methods to solve combinatorial problems, ACO can be classified as a probabilistic *tree search*, i.e. constructive methods building mapping solutions to a tree structure. Tree search methods are generally best suited to problems of graph optimization. They are usually opposed on methods based on local search, such as genetic or memetic algorithms [17, 5].

Among other possible evolutionary optimization techniques, we argue that ACO minimizes the abstractions required to develop an efficient optimization algorithm to solve our problem. In order to apply an evolutionary optimization technique~~s~~ to a concrete problem, two sorts of abstraction must be thought of, i.e., the *fitness function* and the *mapping* between the actual problem and the solution space of the chosen technique [15]. The former defines the criteria that will be optimized, whereas the latter is required to translate the solutions of the problem at hand into a representation that can be handled by the optimization technique, e.g. the space of chromosomes for genetic algorithms or the position of particles in the case of particle swarm optimization. As in all graph optimization problems, in our case the solution space required by ACO coincides with the solution space of the real problem. That is, the optimal path in the PDM coincides with the optimal path discovered by ants in ACO, without the need for further abstraction, such as complex mappings to chromosome structures in the case of genetic/memetic algorithms.

Algorithm 1 shows the pseudo-code of our ACO-based algorithm for solving the optimization problem $P1$. The algorithm comprises the procedures ACOAlgorithm and ACOSearch. ACOSearch represents the search of an optimal path made by an individual ant in the colony ANT (a set of ants). ACOAlgorithm is the main procedure.

ACOAlgorithm uses a number it_n of iterations. In each iteration, the MON-PDM is walked through by the ants in the set ANT, i.e. a colony of ants, starting from its root information element. For each ant, ACOAlgorithm first calls the ACOSearch procedure to determine the complete path p for that specific ant (line 6). Then, it calculates and stores the amount of pheromone that has to be left on the complete path discovered by the ant (lines 7, 8). At the end of an iteration, i.e. after all ants in the colony have walked through the MON-PDM, the pheromone trail is updated all at once using the partial updates (line 14). Using iterations involving a population of ants is required in ACO optimization to avoid early commitment of ants to a non-optimal path. If, for instance, the complete path found by the first ant is far from optimal and the pheromone trail is updated directly by the first ant, then other ants are likely to reinforce such non optimal path.

While during the enactment of the monitoring process the MON-PDM is traversed bottom-up, i.e. from the leaf operations to the root information element, during the optimization ants traverse the MON-PDM top-down, i.e. starting from the root. The procedure ACOSearch implements the individual ant's logic for selecting a complete path while trying to wander from the root of the MON-PDM to its leaf operations. When visiting an information product ip_x, the ant selects the operation to follow according to a uniform probability distribution determined by the amount of pheromone currently deposited on the operations $PREC(ip_x)$ (line 2). In particular, given $PREC(ip_x)$ and the amount of pheromone $\tau_n(op_j)$ deposited on operation $op_j \in PREC(ip_x)$ at iteration n, the ant selects the operation op_j with probability $p(op_j)$ defined as:

$$p(op_j) = \frac{\tau_n(op_j)}{\sum_{op_j \in PREC(ip_x)} \tau_n(op_j)}$$

In other words, the higher the pheromone deposited on an operation, the higher the probability the ant will choose that operation. The randomness introduced by this choice in an ant's behavior prevents the algorithm to commit to early, non-optimal choices for determining the optimal path. For the first ant, all operations in the PDM are initialized with the same level of pheromone, such that the first ant has an equal

Algorithm 1 ACO optimization algorithms for MON-PDM optimization

```
1: procedure ACOAlgorithm($root$, $ANT$, $it_n$)
2: initialize: $n = 0$, $V_{opt} = 0$, $p_{opt} = \emptyset$
3: while $mcp_{m+1} \neq false$ do
4:     for all $ant \in ANT$ do
5:         $p \longleftarrow$
6:         ACOSearch($root$, $ant$, $p$)
7:         Calculate utility $V(p)$
8:         Calculate partial pheromone update
9:         if $p_{opt} = \emptyset \vee V(p) \geq V_{opt}$ then
10:            $V_{opt} \longleftarrow V(p)$                    ▷ update optimal utility value
11:            $p_{opt} \longleftarrow p$                       ▷ update current optimal path
12:        end if
13:    end for
14:    perform pheromone update
15:    $n \longleftarrow n + 1$
16: end while
17:  return $p_{opt}$
18: end procedure

1: procedure ACOSearch($ip$, $ant$, $p$)
2: select $op \in PREC(ip)$ using pheromone distribution
3: $p \longleftarrow p \cup op$
4: if $IS(op) = \emptyset$ then return                      ▷ op is leaf operation
5: else                                                    ▷ offspring the ant on $IS(op)$
6:     for all $ip' \in IS(op)$  do
7:         ACOSearch($ip'$, $ant$, $p$)
8:     end for
9: end if
10: end procedure
```

probability of choosing any operation while walking through MON-PDM.

If the chosen operation op_j is a leaf operation, then the procedure stops (line 4). Otherwise, the ACOSearch procedure is called recursively on all information elements in $IS(op_j)$ (line 7). The for loop in procedure ACOSearch is required to offspring the ant when visiting an operation $op_(j)$ for which the input set $IS(op_j)$ is not a singleton. When, in fact, an operation requires more than one information product to be executed, then all such information products have to be included in the path discovered by the current ant. The offspring of ants while exploring the MON-PDM allows our algorithm to guarantee condition 3 defining a complete path. This is a customization of the ACO meta-heuristic required to fit the problem of optimization over PDMs.

Generally, in ACO optimization the pheromone is updated using a function of fitness [13]. The fitness function captures the degree of fit between a specific direction chosen by an ant and the objective of the ant. In our case, the objective of the ant is to walk through the MON-PDM along a complete path, whereas the fit is defined by the utility that the ant finds along the path, i.e. ants look for the complete path with highest utility. Hence, we use the utility function $V(p)$ as fitness measure.

Given the complete path p discovered by an ant *ant*, the partial amount of pheromone $\psi_{ant}(op_i)$ that the ant leaves on an operation op_i belonging to p is given by the utility of the path $V(p)$, that is:

$$\psi_{ant}(op_i) = V(p), \ \ \forall op_i \in p.$$

Given the set of ants ANT in an iteration of the ACOAlgorithm procedure, the pheromone level of an operation op_i in the MON-PDM at a new iteration $n + 1$ is calculated as follows:

$$\tau_{n+1}(op_i) = \beta \cdot \tau_n(op_i) + \sum_{ant \in ANT} \psi_{ant}(op_i),$$

where β is the evaporation coefficient, comprised between 0 and 1.

In case the optimization problem includes side constraints, then the pheromone update has to keep this into account. Specifically, an ant leaves a pheromone trail only if the complete path it has found satisfies the side constraints. Hence, the partial amount of pheromone $\psi_{ant}(op_i)$ in case of side constraints is as follows:

$$\psi_{ant}(op_i) = \begin{cases} V(p) & \text{if } p \text{ satisfies side constraints} \\ 0 & \text{otherwise} \end{cases}$$

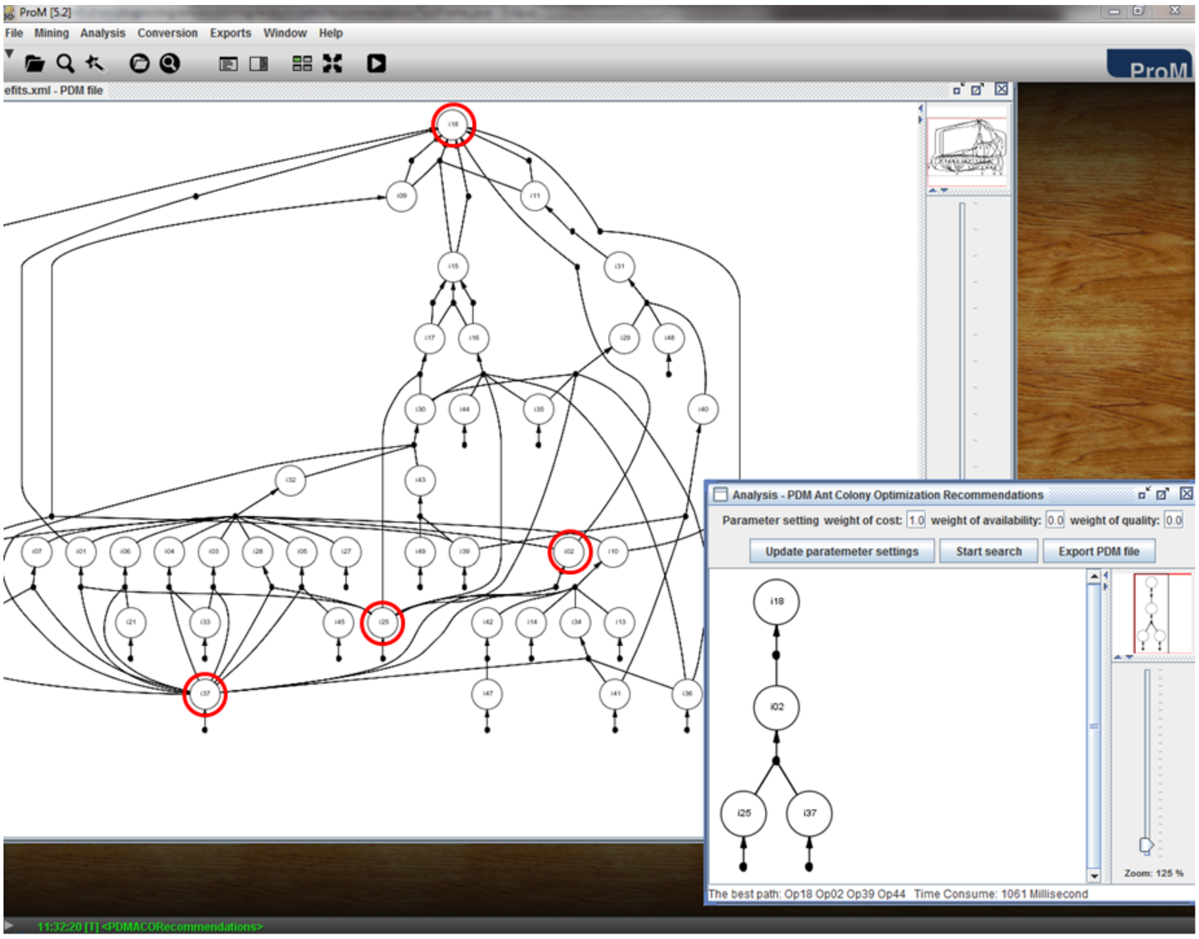
A detailed evaluation of our ACO algorithm performance is reported in the Appendix.

The ACO algorithm for calculating the optimal path in MON-PDM has been implemented as a plugin of the ProM framework.

ProM is an extensible framework supporting a wide variety of *process mining* techniques in the form of plug-ins[1] [30]. Process mining is concerned with the extraction of knowledge about a business process from its process logs, that is, the traces left by process execution in the information system(s) executing such process [29]. Process mining enables the discovery and analysis of several perspectives of business processes, ranging from control flow to the structure of the social network of the actors executing processes. ProM is implemented in Java; its core is available under the GNU public license, while the more than 400 ProM plugins can be downloaded under the L-GPL license.

Although mainly related to process mining techniques, ProM plugins also implement other business process analysis and improvement techniques. In this paper, we exploit the plugins supporting the design of Product-Based Workflows. ProM already provided a plugin for the design of PDMs. We extended ProM by implementing a plugin for the optimization of PDMs. In particular, our plugin allows the specification of the parameters and side constraints of the optimization problem P1 and solves the optimization problem using the ACO heuristic presented in Section 3. On top of the support to the design and optimization of PDMs provided by the extended ProM, we also developed a lightweight Web service-based component for the enactment of the monitoring processes, which is discussed in the next section.

[1]`www.promtools.org/prom6`


ProM [5.2]
File Mining Analysis Conversion Exports Window Help
efits.xml - PDM file
Analysis - PDM Ant Colony Optimization Recommendations
Update paratemeter settings
Start search
Export PDM file
i18
i02
i25
i37
Zoom: 125 %
11:32:20 [T] <PDMACORecommendations>

Parameter setting weight of cost: 1.0 weight of availability: 0.0 weight of quality: 0.0

The best path: Op18 Op02 Op39 Op44 Time Consume: 1061 Millisecond

Figure 3: MON-PDM optimization in ProM

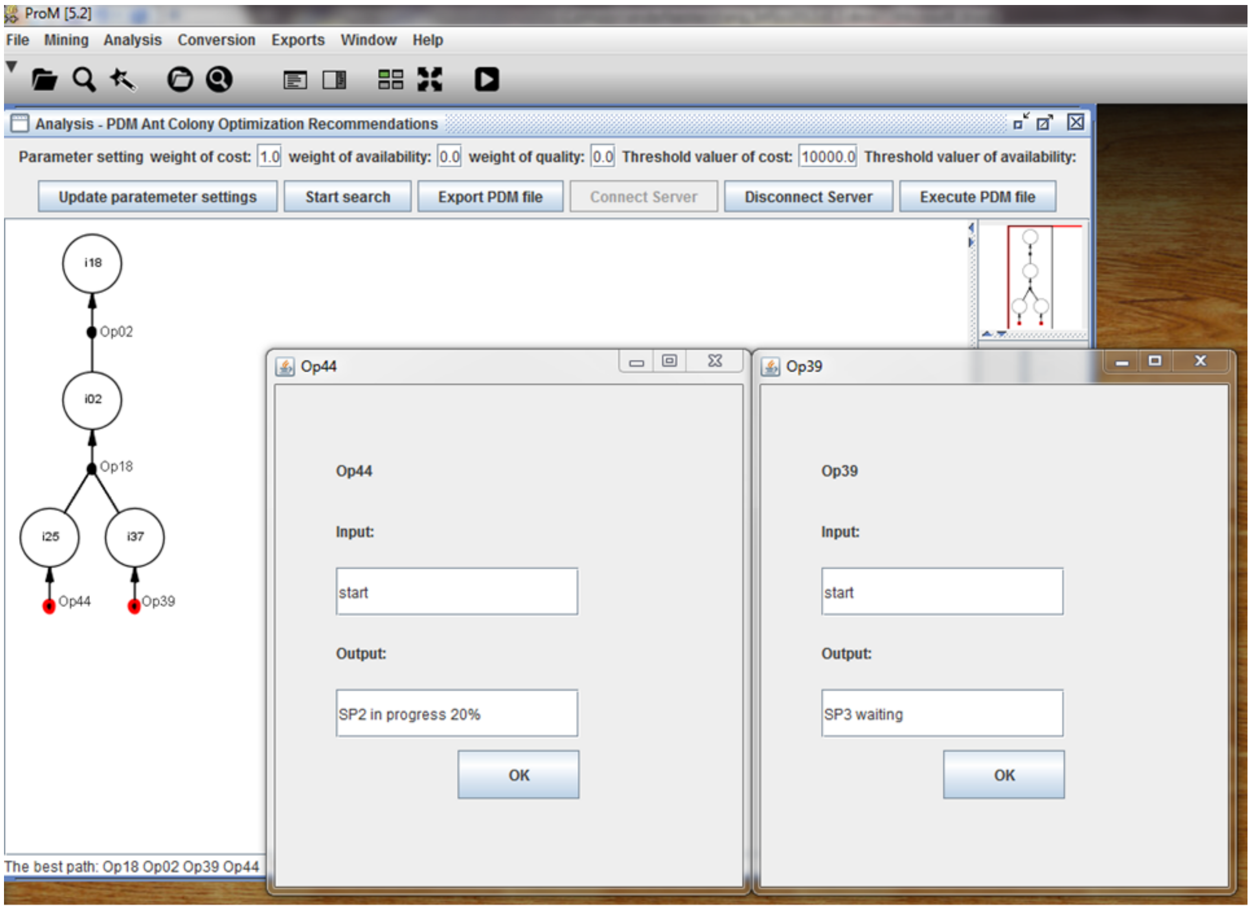
Figure 3 shows a snapshot of the MON-PDM optimization component implemented as a plugin of ProM 5.2. In particular, the upper left PDM is the sample PDM that we use for the evaluation of our algorithm, where we highlighted the information products becoming part of the optimal complete path. The lower right PDM is the optimal complete path resulting from the application of the algorithm. Note that MS can specify its non-functional requirements using the parameters setting form.

As discussed in the next section, after calculating the optimal path in MON-PDM, MS is now ready to execute the optimal MON-PDM. The tool support to this action is described in Section 4.

4. Enacting Cross-Organizational Process Monitoring (Step 3)

The optimal path in the PDM represents the blueprint for implementing the monitoring process satisfying best the requirements of MS. Once the optimal path is determined, the final step is to support its automated execution.

The literature proposes two solutions for executing a PDM. The first solution translates the PDM into a process model using the algorithms described in [33]. The process model can then be deployed and enacted by a workflow engine [31]. The second solution directly executes the PDM, using a process engine for the execution of operations in the PDM and a recommendation system to suggest the next operation to the user [32]. Both solutions have to tackle the problem of uncertainty in the execution of a PDM. An information product in the PDM, in fact, can be produced by executing different operations, and it is up to


ProM [5.2]
File Mining Analysis Conversion Exports Window Help
Analysis - PDM Ant Colony Optimization Recommendations
Update paratemeter settings
Start search
Export PDM file
Connect Server
Disconnect Server
Execute PDM file
i18
Op02
i02
Op18
i25
i37
Op44
Op39
OK
OK

Parameter setting weight of cost: 1.0 weight of availability: 0.0 weight of quality: 0.0 Threshold valuer of cost: 10000.0 Threshold valuer of availability:

Op44

Input:

start

Output:

SP2 in progress 20%

Op39

Input:

start

Output:

SP3 waiting

The best path: Op18 Op02 Op39 Op44

Figure 4: Enactment of optimal path for business process monitoring

the user to decide which operations to choose to produce a certain information product when multiple ways are possible. The first solution captures such uncertainty in the specification of the process model, whereas the second solution delegates the management of uncertainty to the recommendation system acting in pair with the process engine.

Our optimization of the MON-PDM removes this uncertainty. With the algorithm described in Algorithm 1, in fact, we obtain a complete path p in which each information product is *deterministically* produced by the execution of one specific operation or a set of operations. As a consequence, we decided to develop our own specific, lightweight support to the MON-PDM execution. In summary, we developed:

- a component to optimize the MON-PDM, as a plugin of ProM 5.2;
- a lightweight engine directly executing the optimal monitoring process orchestrating the monitoring services as specified by the optimal path in the MON-PDM.

As depicted in our framework in Figure 1, each operation in the MON-PDM is implemented by a Web service published in the Service Registry. Our Monitoring Enactment component receives the complete path from the MON-PDM Optimization component. Each operation in the MON-PDM contains a reference to the Web service operation implementing it.

As shown in Figure 4, our implementation involves a form for the execution of operations, i.e. invoking monitoring Web services, and contextual information for MS, i.e. highlighting the operation that is currently being executed (two leaf operations in Figure 4). Note that while executing leaf elements MS is asked to fill in the input of the operation to start the monitoring process, whereas for non-leaf operations input information is already filled in using the output of previously executed operations. The operation is physically executed by invoking the corresponding operation in the monitoring Web services published by SPs. In case of non-leaf operations the value returned by the preceding Web service invocations is temporarily stored and reused when necessary for subsequent invocations to other Web services required for executing the MON-PDM.

Our automated derivation of monitoring processes allows managing the evolution of both monitoring requirements and monitoring services. If the monitoring requirements of MS change, for instance MS wants to possibly increase the quality of monitoring information or new monitoring services become available, then MS can re-run the optimization using a modified optimization problem and reconfigure the monitoring process. Given the fact that we use a service-oriented approach, new versions of monitoring services can be plugged-in a monitoring process as long as their interface is consistent with the existing version.

Note also that in our framework the execution of a PDM fails when an operation fails to produce the output element. This is a limitation of our approach when compared to direct PDM execution [32], which can deal with operations failure by executing alternative operations, if available, to obtain the required information elements.

Concerning the technical architecture, we use Java Web services. Examples of instrumentations of the OpenESB BPEL engines and the YAWL workflow engine for building monitoring services are discussed in [8].

5. Conclusions

Delegation and outsourcing of non-core activities represent viable ways for companies in dynamic business settings to increase their business value. In order to support the need for control over outsourced operations emerging from such a scenario, in this paper we presented a framework to support the definition and implementation on-the-fly of monitoring infrastructures for cross-organizational business processes. Monitoring requirements (functional and non-functional) are captured in a PDM. The PDM is then optimized to satisfy the monitoring stakeholder requirements. Optimization is achieved through an ACO-based heuristic, which performs reasonably well even on complex product data models resulting from the analysis of a real world practice case study. Eventually, our framework supports also the enactment of cross-organizational monitoring through an engine, implemented as a plugin of the ProM framework, which can directly execute the optimal path discovered in the PDM.

We envision two directions for future work. First, we will focus on the support to design the product data models, designing and implementing advanced methods for annotating monitoring services, retrieving those from the registry, and combining them in a product data model. Concerning the PDM optimization, we will focus on improving our heuristic solution of the product data model optimization problem by considering different heuristics, e.g. based on genetic algorithms. In the same direction, we want to extend the type of monitoring utility functions, considering also not separable utility functions.

Acknowledgments

The authors would like to thank Gianni Acampora for the discussions about possible evolutionary techniques to exploit in our framework.

Appendix - Evaluation of the PDM Optimization Algorithm

We evaluated our ACO algorithm keeping as reference the optimal solution calculated using exhaustive search of all possible complete paths.

The evaluation considers a fairly complex PDM, with 45 operations and 50 information products ($I = 45$, $X = 50$), showing a total of 13 complete paths to produce the root element. This PDM represents a real world PDM defined for the process of awarding unemployment benefits in the Netherlands [31]. Since the

Table 1: Experimental evaluation of the MON-PDM optimzation algorithm.

Experiment Number	**Parameters** $[it_n, \|ANT\|]$	**Performance** pi_1	**Performance** pi_2	**Execution time** (ms)
1	$[1, 25]$	100%	100%	≤ 2
2	$[1, 20]$	$99, 61$%	98%	≤ 2
3	$[1, 10]$	$98, 33$%	80%	≤ 1
4	$[1, 5]$	$91, 57$%	20%	≤ 2
5	$[2, 5]$	$94, 10$%	78%	≤ 3
6	$[3, 5]$	$99, 17$%	95%	≤ 4
7	$[4, 5]$	$99, 50$%	97%	≤ 7
8	$[5, 5]$	100%	100%	≤ 9

original PDM contained only information on the cost of the operations, we generated randomly all missing values of availability and quality.

An experiment in our evaluation is determined by a specific configuration of the number of iterations, i.e. the value of it_n, and the number of ants in a colony, i.e. the cardinality of the set ANT. We run $R = 50$ replications in each experiment. In each experiment, we use the value $\beta = 0.5$ for the pheromone evaporation, which has been found to be experimentally *good* in [12].

Given p_r and p_{opt} as the optimal paths obtained in a replication r and through the manual exhaustive search, respectively, we defined the performance indicators pi_1 and pi_2 of the PDM optimization algorithm as follows.

The first performance indicator pi_1 captures the percentage ratio of the expected value of the utilities of p_r and p_{opt} on the set of replications defining one experiment, that is:

$$pi_1 = \frac{E[V(p_r)]}{E[V(p_{opt})]} \cdot 100.$$

The second performance indicator pi_2 is the average number of times within an experiment in which a replication succeeded in finding the optimal path p_{opt}, that is:

$$pi_2 = \frac{\sum_{r=1}^{R} suc_r(p_r)}{R} \cdot 100.$$

where the function $suc(p_r)$ is defined as:

$$suc_r(p_r) = \begin{cases} 1 & \text{if } p_r = p_{opt} \\ 0 & \text{otherwise} \end{cases}$$

Table 1 shows the results of our experimental evaluation. Optimal performance in terms of both quality and accuracy is achieved by using either one iteration with 25 ants or 5 iterations with 5 ants. Experiments involving one iteration, however, show an overall better performance in terms of execution time. From the point of view of the application of our framework, the time required to optimize the MON-PDM remains in the order of the tens of milliseconds, which is negligible when compared to the time required for executing the monitored business process or, of course, the time needed for manually calculating the optimal path through exhaustive search.

Note that, while convergence of ACO algorithms to the optimal solution has been analytically demonstrated [3], the analysis of the time required to find an optimal solution and the extraction of guidelines for the design of ACO algorithmic components, e.g. optimal number of iterations or ants in an iteration, are still open research questions [13]. In our problem, we found that, while optimal accuracy (pi_2) is achieved by a minimum set of ants (25 in our case), used either in one large or several small iterations, increasing the number of ants in an iteration generally shows better performance than increasing the number of iterations maintaining a fixed ant population. Theoretically, these results apply only to our specific problem and cannot be generalized to other contexts without further investigation.

References

[1] D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *IEEE Transactions on Software Engineering*, 33(6):369–384, 2007.

[2] G. Ausiello, G. Italiano, U. Nanni, and L. Brim. Hypergraph traversal revisited: Cost measures and dynamic algorithms. *Mathematical Foundations of Computer Science*, 1450:1–16, 1998.

[3] A. Badr and A. Fahmy. A proof of convergence of ant algorithms. *Information Sciences*, 160:267–279, 2004.

[4] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz. Heuristics for QoS-aware web service composition. In *Int. Conf. on Web Services (ICWS)*, pages 72–82, 2006.

[5] C. Blum. Beam-ACO - hybridizing ant colony optimization with beam search: an application to open shop scheduling. *Computers & Operations Research*, 32:1565–1591, 2005.

[6] D. Chiu, K. Karlapalem, Q. Li, and E. Kafeza. Workflow view based E-contracts in a cross-organizational E-services environment. *Distributed and Parallel Databases*, 12:193–216, 2002.

[7] M. Comuzzi and I. Vanderfeesten. Product-based workflow design for monitoring of collaborative business processes. In *Proc. 23rd Int. Conf. Advanced Information Systems Engineering*, pages 154–168, 2011.

[8] M. Comuzzi, J. Vonk, , and P. Grefen. Measures and mechanisms for process monitoring in evolving business networks. *Data and Knowledge Engineering*, 71:1–28, 2012.

[9] I. Corporation. Websphere business monitor, 2011.

[10] M. Daneva and R. Wieringa. A requirements engineering framework for cross-organizational ERP systems. *Requirements Engineering*, 11:194–204, 2006.

[11] M. Dorigo and C. Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344:243–278, 2005.

[12] M. Dorigo and G. Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1470–1477, 1999.

[13] M. Dorigo and T. Stuzle. *Ant Colony Optimization.* MIT Press, Cambridge, MA, 2004.

[14] K. Eisenhardt. Agency theory: An assessment and review. *Academy of Management Review*, 14:57–74, 1989.

[15] E. Elbeltagi, T. Hegazy, and D. Grierson. Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, 19:43–53, 2005.

[16] B. Fox, W. Xiang, and H. Lee. Industrial applications of the ant colony optimization algorithm. *International Journal of Advanced Manufacturing Technology*, 31:805–814, 2007.

[17] M. Ginsberg. *Essentials of artificial intelligence.* Morgan Kaufmann Publishers, 1993.

[18] P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig. CrossFlow: cross-organizational workflow management in dynamic virtual enterprises. *Comput. Syst. Sci. & Eng.*, 5:277–290, 2000.

[19] P. Grefen, R. Eshuis, N. Mehandijev, N. Kouvas, , and G. Weichart. Internet-based support for process-oriented instant virtual enterprises. *IEEE Internet Computing*, 13:65–73, 2009.

[20] S. Guinea, O. Nano, G. Spanoudakis, and L. Baresi. Comprehensive monitoring of BPEL processes. *IEEE Internet Computing*, 14:50–57, 2010.

[21] V. Kartseva, J. Hulstijn, J. Gordijn, and Y.-H. . Tan. Control patterns in a health-care network. *European Journal of Information Systems*, 19:320–343, 2010.

[22] H. Liu, F. Zhong, B. OuYang, and J. Wu. An approach for QoS-aware Web service composition based on improved genetic algorithm. In *Int. Conf. on Web Information Systems and Mining (WISM)*, pages 123–128, 2010.

[23] S. Muller and C. Supatgiat. A quantitative optimization model for dynamic risk-based compliance management. *IBM Journal of Research and Development*, 51:295–307, 2007.

[24] J. Oh, N. W. Cho, H. Kim, Y. Min, and S.-H. Kang. Dynamic execution planning for reliable collaborative business processes. *Information Sciences*, 181:351–361, 2011.

[25] H. Reijers, S. Limam Mansar, and W. van der Aalst. Product-based workflow design. *Journal of Management Information Systems*, 20:365–391, 2003.

[26] W. Robinson. A requirements monitoring framework for enterprise systems. *Requirements Engineering*, 11:17–41, 2006.

[27] K. Sim and W. Sun. Ant colony optimization for routing and load-balancing: survey and new directions. *IEEE Transactions on Systems Science and Cybernetics, Part A*, 33:560–572, 2003.

[28] K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, 185:1155–1173, 2008.

[29] W. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[30] B. van Dongen, A. de Medeiros, H. Verbeek, A. Weijters, and W. van der Aalst. The ProM framework: A new era in process mining tool support. In *Applications and Theory of Petri Nets*, pages 444–454, 2005.

[31] I. Vanderfeesten. *Product-Based Design and Support of Workflow Processes.* PhD thesis, Eindhoven University of Technology, 2008.

[32] I. Vanderfeesten, H. Reijers, and W. van der Aalst. Product-based workflow support. *Information Systems*, 36:517–535, 2011.

[33] I. Vanderfeesten, H. Reijers, W. van der Aalst, and J. Vogelaar. Automatic support for product based workflow design: Generation of process models from a product data model. In *Proc. OTM 2010 Workshops*, pages 665–674, 2010.

[34] L.-J. Zhang, B. Li, T. Chao, and H. Chang. On demand Web services-based business process composition. In *IEEE Int. Conference in Systems, man, and cybernetics*, pages 4057 – 4064, 2003.

[35] W. Zhang, C. Chang, T. Feng, and H.-Y. Jiang. QoS-based dynamic Web service composition with ant colony optimization. In *IEEE Computer Software and Applications Conference (COMPSAC)*, pages 493–502, 2010.