Cost-Informed Operational Process Support

M. T. Wynn[1], H. A. Reijers[2,3], M. Adams[1], C. Ouyang[1], A. H. M. ter Hofstede[1,2], W. M. P. van der Aalst[2,1], M. Rosemann[1], and Z. Hoque[4]

[1] Queensland University of Technology, Brisbane, Australia.
{m.wynn,mj.adams,c.ouyang,a.terhofstede,m.rosemann}@qut.edu.au
[2] Eindhoven University of Technology, Eindhoven, The Netherlands.
{h.a.reijers,w.m.p.v.d.aalst}@tue.nl
[3] Perceptive Software, Apeldoorn, The Netherlands.
hajoalexander.reijers@perceptivesoftware.com
[4] La Trobe University, Melbourne, Australia.
z.hoque@latrobe.edu.au

Abstract. The ability to steer business operations in alignment with the true origins of costs, and to be informed about this on a *real-time basis*, allows businesses to increase profitability. In most organisations however, high-level cost-based managerial decisions are still being made separately from process-related operational decisions. In this paper, we describe how *process-related decisions at the operational level can be guided by cost considerations* and how these *cost-informed decision rules can be supported by a workflow management system*. The paper presents the conceptual framework together with data requirements and technical challenges that need to be addressed to realise cost-informed workflow execution. The feasibility of our approach is demonstrated using a prototype implementation in the YAWL workflow environment.

Keywords: Cost-Informed Process Enactment, Business Process Management, Workflow Management, Process Modelling, Prototype

1 Introduction

Organisations are eager to implement cost-based considerations in their day-to-day operations. The Gartner EXP worldwide surveys of CIOs showed process management and reducing enterprise costs to be the top two priorities for many organisations [6]. An insurance company, for instance, may want to approve a claim quickly and skip further investigations if the administrative cost incurred at a certain point is deemed to be excessive in comparison to the potential payout amount. In a similar vein, when an organization is seeking an external service provider, the candidate who gives the cheapest quote for a job is preferred provided that the candidate is deemed capable of doing the job.

In most organisations, however, tying cost considerations to process-related decisions forms a challenge. Our observation is that most Workflow Management Systems (WfMSs) offer no support for cost considerations beyond the use of generic attributes (e.g. FileNet Business Process Manager) or some basic cost

recognition and reporting (e.g. TIBCO Staffware Process Suite). Detailed cost information is typically not available at runtime and, as a result, cost information is not used for monitoring or for operational decision support.

Our motivation for this paper is to provide a conceptual framework to enable WfMSs[1] to achieve a higher level of support for cost-informed operational decisions. More specifically, such a cost-aware WfMS is able to record historical cost information and makes use of it for (real-time cost) monitoring and escalation purposes, as well as supporting simulation and cost prediction capabilities. Ideally, it can also support process improvement decisions based on cost considerations, such as determining cost profiles of different processes/process variants and using them for selection/redesign purposes. To this end, we propose methods for the capture of cost-based decision rules for process, activity and resource selections within business processes and how automated support could be provided for cost-informed process enactment within a WfMS. Our contribution is that we lay the groundwork for organizations to more easily translate cost strategies into their operational fulfilment activities using a WfMS.

It is worth noting that cost is traditionally considered as one of many non-functional requirements (NFR) for a software system or service in the same manner as maintainability, usability, reliability, traceability, quality or safety [3]. Researchers have explored how these requirements can be elicited from users and how to incorporate NFRs in conceptual models [2, 5, 11, 13]. However, the cost perspective has a very close and direct link with BPM/WfM, much more so than most other NFRs. First of all, consider that cost is relevant from the viewpoint of individual activities, resources, and entire processes – all of which are in scope for a WfMS. This *versatility* typically does not hold for many other NFRs. Quality, for example, is relevant in the context of a whole process, but not necessarily for a single activity; usability can be tied to a single activity, but not to resources; reliability may be relevant for a single activity, but is too fine-grained for cross-functional processes. Secondly, when we refer to the *dynamic* nature of cost we mean that it is relevant for both design and run time decisions. This aspect differs from NFRs such as maintainability and usability, which are important concerns at design time, but out of scope for operational decision making. Again, both the design and run time perspectives are in scope for a WfMS. In summary, a WfMS is a natural platform to manage cost concerns since it connects the many levels of cost interests and allows for implementing cost-informed design and operational decisions.

The remainder of the paper is organised as follows. Section 2 describes the proposed conceptual framework to support cost-informed decisions within a workflow environment. Section 3 provides an overview of technical challenges that need to be addressed to make WfMSs cost-aware. Section 4 discusses the prototype implementation of a *Cost Service* within *YAWL*, a well-known open-source WfMS. Section 5 presents related work and section 6 concludes the paper.

[1] In the remainder, we use the term WfMS to refer to all process-aware information systems, including Business Process Management Systems (BPMSs).

2 A Framework for Cost-Informed Decisions

This section defines a framework to support cost-informed process execution in a WfMS. We propose that in addition to the ability to specify cost-informed control flow definitions and resource allocation rules *at design time*, a cost-informed WfMS should also provide support for system-based decisions and system-supported user decisions *at runtime*. As such, different types of actions can be performed by a WfMS or by a resource interacting with a WfMS to support *cost-informed decision making during process execution*. Figure 1 depicts our conceptual framework which describes 1) *data input*, i.e. the information requirements to enact actions that can be undertaken by or with a WfMS to support cost-informed decision making, 2) the *actions* that can be taken on the levels of process, activity, and resource (work distribution), and 3) the *cost-informed support* that is delivered, either through decisions by the WfMS itself or people using its support.

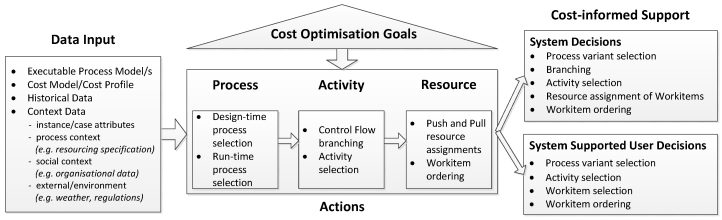

Fig. 1. A framework supporting cost-informed process execution with a WfMS.

2.1 Data Input

A number of key objects need to be provided to a WfMS as *data inputs* to support cost-informed actions. In addition to an *executable process model*, we need access to a *cost model* that can be associated with different elements within a process model (e.g., the cost rates of activities and resources). Cost data could be as simple or as complex as an organisation requires it to be. For instance, it could be a variable cost that describes the hourly rate of a resource, but it could also be a dynamic scheme that ties overhead costs to each case depending on seasonal factors. A cost profile of a process is made up of all these cost rates/data associated with activities within a particular process. Cost information, together with *historical data* as stored in a so-called *process log* regarding past executions, can be used to determine the cost of process executions as illustrated in our earlier work [23]. Since a business process is always executed in a particular context, we also adopt the four levels of *context data* described in [20]: case attributes, process context, social context, and the environment. Environmental information, for example, is needed to accurately determine the seasonal influence in the dynamic cost scheme we mentioned.

2.2 Actions

All cost-informed actions are based on the data inputs that we discussed on the one hand, while they are governed by the strategic considerations within an organisation on the other. We refer to these as *cost optimisation goals*. Typical examples are: cost minimisation, cost overrun prevention, profit maximisation, incorporation of opportunity cost, etc.

The concrete cost-informed actions supported by a WfMS, informed by data input and governed by cost optimisation goals, can then be classified into three levels: process, activity, and resource.

Process. The *process* level is concerned with carrying out *process selection* based on cost information of processes or process variants at design time or at runtime. This may involve the selection among different processes or selection among different process variants (which are created from individual processes during the execution phase). The variants of a process all share a kernel of similar behaviour, but may subtly differ to make them appropriate to deal with subtypes of cases, differing conditions across markets, or other concerns. For example, a rigorous variant for damage claim processing may be in place for normal operations, while in emergency situations a variant may be selected that relaxes or postpones certain checks. It should also be possible to assign a (whole) process or process variant to a certain resource team for execution (i.e. outsourcing) based on the cost profile.

Activity. For cases that have been started under the control of a WMFS, it is necessary to decide at certain points about the *activity* (or activities) to be executed next. Note that a process instance is known as a *case* and an instance of an activity in a case is known as a *workitem*. In its coordination capability, a WfMS may decide on which workitems are enabled in a specific case, based on the branching conditions specified in the control-flow of the underlying process model. As such, the WfMS must be aware of or needs to have access to cost relevant information for branching decisions. A WfMS could also start, skip, and cancel a workitem, among other actions, based on that cost information. For example, a WfMS can choose to cancel a particular workitem which might cause a potential budget overrun.

Resource. After a workitem has been enabled, further choices related to distributing work to *resources* become possible. For workitems that need to be carried out by a user, both "push" and "pull" patterns of activity-resource assignment [17] should be supported. With respect to the "push" patterns, a WfMS would need to support cost-informed resource allocation rules for selecting a resource to offer, allocate, and/or start a workitem. With respect to the "pull" patterns, a WfMS would support a user to make cost-informed workitem selection decisions.

Figure 2 shows possible cost-based decision points within the lifecycle of a workitem. After a workitem is created, the system can offer the workitem to one or more resources for execution (which is depicted as "S:offer_s" and "S:offer_m" decisions). An additional "C:selection" annotation indicates that it is possible for this system decision to be cost-informed. i.e. a resource could be selected based on its cost characteristics. After a workitem is offered to multiple resources, one of these resources can decide to work on a workitem. For example, the "R:allocate_m" transition represents a resource making a selection and the "C:selection" annotation indicates that this system-supported user decision can be cost-informed. After a workitem is started by a resource, it can still be suspended/resumed or cancelled by a resource. The "R:suspend", "R:resume", and "R:cancel" transitions reflect these possibilities and similarly the "C:decision" annotations in these transitions indicate that these user decisions can be guided by cost information. In Figure 2, transitions that can be cost-informed are depicted using bold arrows.

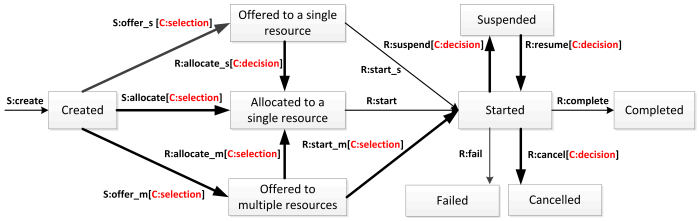

Fig. 2. Lifecycle of a workitem (based on [17]) – enriched with potential cost-based rules for system decisions and system-supported user decisions.

When more than one workitem is assigned to a resource and/or when a workitem is offered to multiple resources, a WfMS can provide support for the prioritisation of workitems based on cost information. Figure 3 illustrates a scenario where a resource has multiple workitems on his/her worklist. These workitems can be (re)ordered either by the resource ("R:re-order") or by the system itself ("S:re-order") based on cost information. In the case of a workitem being offered to multiple resources, the system can withdraw an offered workitem ("S:withdraw"), which could lead to reordering of the remaining workitems on a worklist. The "C:ordering" annotations indicate that these transitions can be cost-informed.

2.3 Cost-informed Support

As we mentioned, our framework identifies the two types of *cost-informed support* that result from the discussed ingredients: *systems decisions*, which can be taken by the WfMS itself, and *system-supported user decisions*, which are taken by resources on the basis of information provided by the WfMS. These decisions reflect the different action types supported by the framework. For instance, it

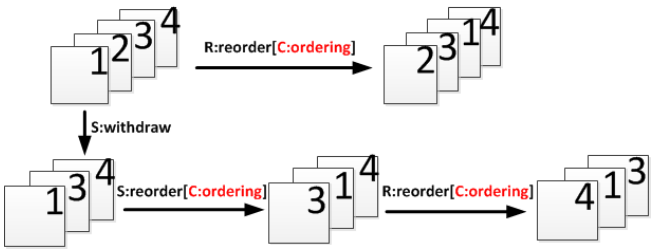

Fig. 3. Cost-based (re)ordering of a worklist of a resource with multiple workitems

is possible for the WfMS to make an automated selection of the process variant based on its cost profile and context information. Alternatively, the WfMS can provide the resource with cost profiles of different process variants and the resource can make the selection. This is also true for decisions on which activities to execute. The WfMS can either make a cost-informed decision based on a pre-defined business rule to enable/start an activity or allow the resource to start/skip/suspend/resume/cancel a particular activity based on cost information. Decisions on which paths to choose in a process are exclusively taken care of by the WfMS using predefined cost-informed business rules. Workitems can be assigned by the WfMS or can be selected by a resource based on their cost (historical or predicted values). Finally, a WfMS can order multiple workitems or a resource can decide to reorder his/her workitems using cost information. The example in Section 4.2 will illustrate these types of support, along with an explanation of the technical capabilities these depend upon.

3 Technical Challenges

For a WfMS to be capable of cost-informed enactment, execution and support across the three levels (process, activity and resource), the following key criteria would need to be satisfied:

1. *Association of cost data and cost-based rules with a process/workflow.* This support is required prior to the execution phase. Relevant cost rates for different process elements such as activities and resources must be specified in advance. Some values would include salary and incidental costs for human resources, the costs of materials required, fixed costs associated with activity enactments, rentals, depreciation, and so on.
2. *Runtime calculation of the cost of execution of each process instance and its component activity instances.* Such calculations may be:
   - *time-based*, for example salary costs for the time a human resource spends on workitem execution, or timed charges for interaction with an external service, or the cost of insurance for a period;
   - *usage-based*, for example forklift hire, or the use of an MRI machine, or payment of a set fee for an expert witness;

- *measurement-based*, for example per tonne costs of a raw material, or per millilitre of a pharmaceutical, or per kilowatt-hour of a power supply;
- *invocation-based*, for example costs involving in retooling an assembly line for a product run;
- *a fixed cost*, for example an overhead cost of commencing an activity, or a building approval application fee;
- *a combination of the above*, for example a truck rental may involve an initial hire cost plus a fee per kilometre, or a usage-based fee for a machine hire may also involve a time-based insurance fee.

3. *Logging and analysis of cost data.* The ability to archive all calculated costs for each process instance (incorporated into the process event logs) and to perform extrapolated calculations over archived data.
4. *Support for cost-informed decisions.* The ability to use the calculated cost for the current process instance, and/or those of all previous instances of the process, to:
   - make human-based and system-based *cost-informed control-flow decisions*. These decisions would include providing real time calculated values for use as input into branching predicates; to continuously monitor for cost overruns and, when detected, manually or dynamically skip unnecessary or low priority workitems, or cancel workitem and/or cases; and to notify administrators when cost thresholds are being approached;
   - *allocate work* to resources based on decisions about their costs;
   - provide human resources and administrators with *cost information* about a process and its component activities, to enable them to make cost-informed decisions about subsequent process executions and process re-engineering;
   - support for *cost-informed process variant selections*.

4 Realisation

We have developed a prototype implementation for the YAWL workflow environment [18] that addresses the technical challenges outlined in the previous section. YAWL was chosen as the implementation platform because it is built on an expressive workflow language that provides extensive support for identified workflow and resource patterns, together with a formal semantics. The environment is open-source and offers a service-oriented architecture, allowing the prototype to be implemented completely independent of the core workflow (enactment) engine.

4.1 The Cost Service

Our prototype, known as the *Cost Service*, has been realised as a YAWL Custom Service. It provides two interfaces: one which receives notifications from the workflow engine and participating services at various points in the life-cycle

of a case, and the other which allows the engine and services to query cost-information, either to request a calculation and have the result returned, or to return a complete cost-annotated log of a process instance (or instances). The latter interface also supports the import and export of *cost models*.

A cost model is an XML document that describes all the base cost data and formulae to be associated with a particular process model as defined in [23]. In brief, each cost model consists of three core descriptor sets:

- *Drivers:* Each cost driver defines how cost is associated with one or more process elements (resource, activity, case) together with the relevant cost rate for each element. A cost rate is defined as a data pair of a value and a *per* amount, for example $50 per hour, $70 per tonne, $20 per invocation (fixed) and so on, applied to a process element.
- *Functions:* Each function defines an expression for aggregating various cost elements. For example, a function may aggregate a fixed cost, and costs for salaries, insurance and machine hire for all resources involved in an activity.
- *Mappings:* Each mapping provides a way to relate terms used in management accounting to terms used in a WfMS.

The Cost Service is also responsible for the logging of all cost data for all process instances and their activities. The workflow engine and other interested services such as the *Resource Service*, which manages all resourcing allocations, notify the Cost Service throughout the life-cycle of each process instance, passing to it the appropriate data so that it can (i) perform the required cost calculations by applying the data to the relevant cost model components; and (ii) store all interactions and results in its process logs.

The workflow engine has been extended to accommodate control-flow predicates that include cost-based expressions. When process execution reaches a control-flow predicate that contains a cost-based expression, the workflow engine will call the *Cost Service*, passing the expression, along with all associated data. The *Cost Service* will use that data to evaluate the expression against the appropriate cost model components, and return the result. The engine will then embed the result into the predicate (replacing the cost-based expression), which it will then continue to evaluate in the usual manner, as required.

The YAWL Resource Service has a pluggable framework for adding new resource *allocation strategies*, which at runtime receive the set of potential users that may be allocated an activity, and use a defined strategy to select one user from the set. The standard set of YAWL allocators (e.g., Random Choice, Shortest Queue, Round Robin) has been extended with a number of cost-based strategies, such as *Cheapest Resource, Cheapest to Start, Cheapest Completer* and so on. When the Resource Service enacts a cost-based allocator at runtime, the allocator will directly query the *Cost Service*, requesting a calculation based on previous case histories (stored within the process logs) for the resources involved, based on the particular allocation strategy in question. The allocator will then use the result of the query to determine the appropriate resource to whom to allocate the activity, fulfilling a *push*-based resource interaction.

A user or administrator interacting with the YAWL worklist, which is also managed by the Resource Service, may invoke a query request to the *Cost Service* for data about a particular activity, which will then be displayed on their work queue. The user can then use this information to make ad-hoc cost-informed decisions regarding which activity to choose from their worklist to perform next, fulfilling a *pull*-based resource interaction.

With regards to process variants, the standard YAWL environment contains a service called the *Worklet Service* that allows for the selection of process variants based on the current case context, available data values and an extensible rule set [18]. Future work will extend this service to also support cost-based rule expressions, which may then be used to determine which process variant is the ideal selection for the current context of a case.

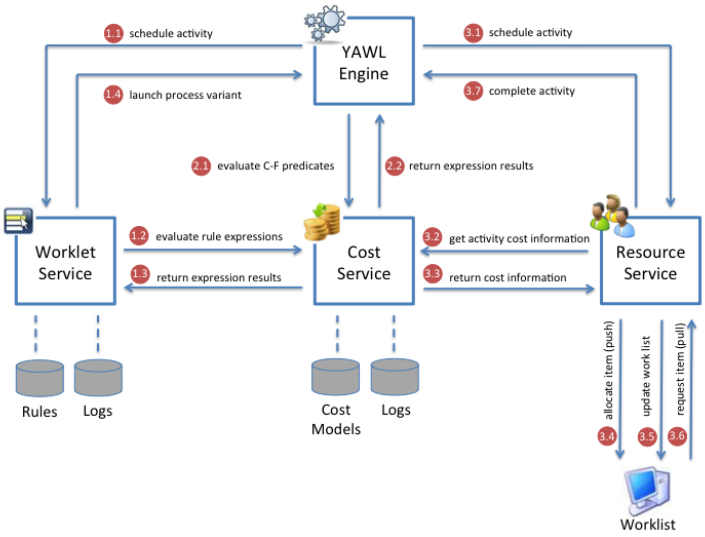

Fig. 4. Prototype architectural flow in the YAWL environment.

Figure 4 shows the flow of information through the prototype for each level of cost-informed support. At the *process* level, the workflow engine schedules an activity for execution by the Worklet Service (1.1). The Worklet Service traverses its rule set for the activity, querying the Cost Service to evaluate cost-based rule expressions (1.2). The Cost Services evaluates and returns the results (1.3), which the Worklet Service uses to select the appropriate process variant for the activity, and launches the variant in the engine (1.4).

At the *activity* level, when the workflow engine encounters an branching construct in the control-flow of a process instance, it queries the Cost Service to evaluate the predicate of each outgoing branch (2.1). The engine then uses the results of the predicate evaluations to fire the branch that evaluates to true (2.2).

At the *resource* level, where the distribution of work takes place, the workflow engine schedules an activity for a (human) resource (3.1) with the Resource

Service. The Resource Service then queries the Cost Service for all cost information pertaining to the activity (3.2), which the Cost Service returns (3.3). If the activity is configured for system-based allocation (push pattern), the specified allocation strategy (e.g. Cheapest Resource) is employed using the cost information in its calculations, then the activity is routed to the worklist of the selected resource (3.4). If the activity is configured for resource-based allocation (pull pattern), the affected resources' worklists are updated with the retrieved cost information (3.5) allowing a resource to select the appropriate activity based on the cost information presented to them (3.6).

4.2 Illustrative Example

We consider a simplified home loan application and approval process as an illustrative example to demonstrate the support for cost-informed process enactment within the YAWL environment. Figure 5 depicts a YAWL model of the process which has been annotated with role-based resource assignments. Most of the tasks in the process are assigned to one specific role, except for three parallel tasks in the "(Re-)Assess Loan Application" sub-process, which can be performed by either a resource performing the role of Mortgage Underwriter (MU) or Underwriting Assistant (UA). Note that the "Engage Broker" activity will be delegated to the Worklet Service (WS) for execution at runtime. The execution of the "Need Mortgage Insurance" activity will be automatically supported by the YAWL engine using the information provided in the loan application.

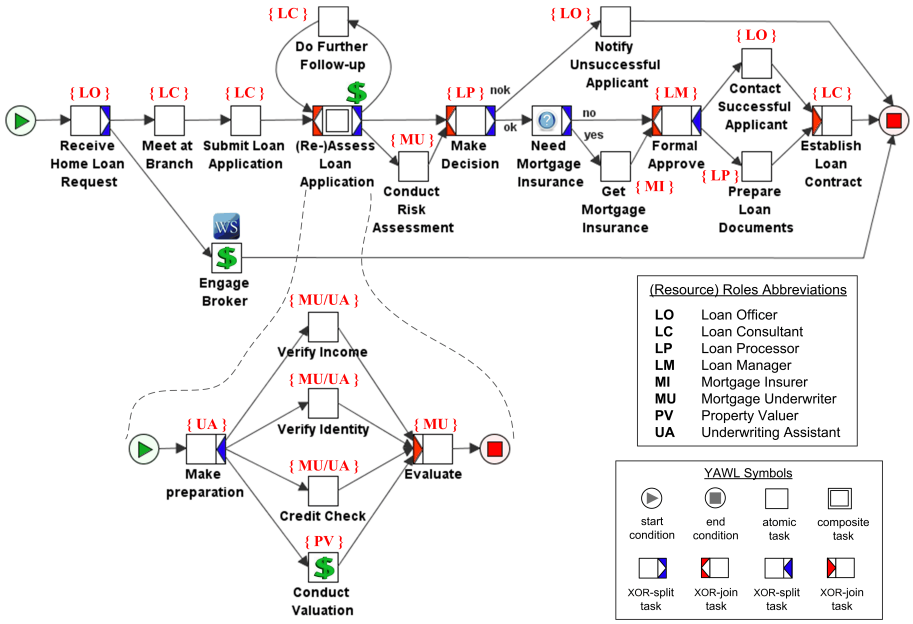

Fig. 5. A home loan process in YAWL (annotated with resource assignments).

With this example, we assume that the cost optimisation strategy is to minimise the labour cost for processing a loan case when possible. The cost model includes the cost rates of resources and activities at a bank, and of mortgage broker services. These can be categorised as follows.

- Role-based (variable) cost rate of a resource, e.g., a bank employee in a certain role has a salary of $50 per hour.
- Fixed cost rate of a resource for a given activity, e.g., a property valuer charges $300 to conduct a property valuation activity.
- Fixed cost rate for an activity, e.g., a mortgage insurance processing fee is $50 per loan application.
- Case-based (variable) cost rate, e.g., a mortgage broker service charges a commission of 0.5% of the loan amount.

Support for cost-informed decisions at each of the three levels during the process enactment can be demonstrated through examples as follows.

Process level. At runtime the WS handling the "Engage Broker" activity maintains a number of process variants corresponding to various broker services that charge the bank different commissions and/or fees for service provision. Cost-informed rules or criteria to guide the selection of the process variant to execute can be specified as predicates to pass to the Cost Service for evaluation. Depending on information from the home loan application such as loan amount, the applicant's ability to repay, the property location, and planning requirements, together with costs incurred from previous variant instances, the variant that represents the best value for money, while covering the necessary regulatory and risk requirements, will be selected.

Activity level. The XOR-split after the "(Re-)Assess Loan Application" activity embeds a cost-based predicate. With a clear positive or negative assessment result, the "Make Decision" activity will be carried out immediately afterwards. Otherwise, further follow-up will be required for another round of assessment. However, if the processing cost to this point reaches a certain limit (can e.g. be set as a certain percentage of the loan amount), then instead of conducting further follow-up, a risk assessment will be performed based on the current evaluation data to make a decision.

Resource level. For each role, there are multiple resources who may have different cost rates (e.g. different salary levels between a junior and a senior loan processor). For example, for the execution of the "Conduct Valuation" activity, the "Cheapest Resource" resource allocation strategy is selected. Figure 6 includes screenshots of design-time support for allocation strategy selection in the YAWL designer, while Figure 7 shows screenshots of the organisational model, the related cost model, and the automatic resource allocation (as expected) during the process execution in the YAWL runtime environment.

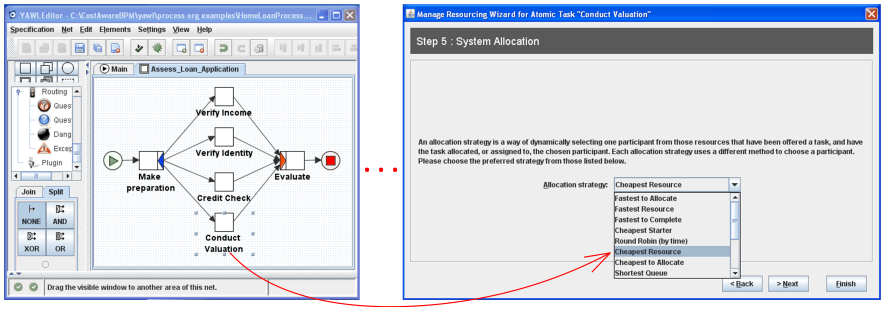

Fig. 6. At design time: The system is configured to allocate "Conduct Valuation" activity to a resource with Property Valuer role with the cheapest cost.

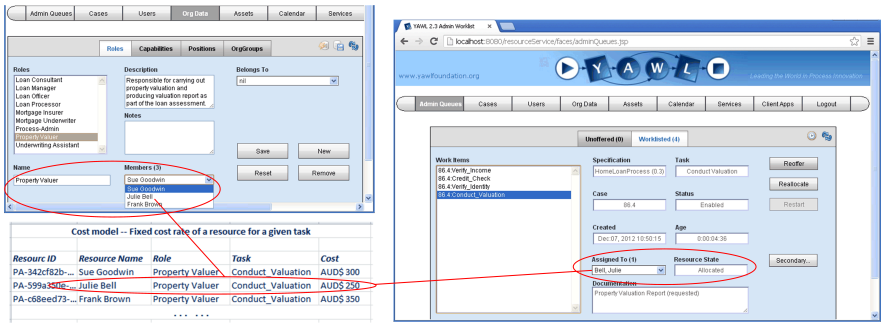

Fig. 7. At runtime: The system finds the resource with "Property Valuer" role (from organisational data) with the cheapest cost (from the cost model) and assigns the workitem (instance of activity) "Conduct Valuation" to this resource.

5 Related Work

Cost has always been one of the key factors under consideration in the context of business process reengineering [10] and process improvements [16]. In [10], the authors discussed the findings from a survey conducted of commonly used business process reengineering techniques by consulting organisations and the findings highlighted the importance of cost/benefit/risk analyses at different stages, and in particular identified Activity Based Costing (ABC) [4] as one of the representative techniques in the evaluation stage. Through the iterative application of BPM techniques, processes can be improved in terms of quality, flexibility, time and/or cost [16, 12]. Although WfMSs support planning, execution, (re)design and deployment of workflows [22], direct support for cost-informed execution is currently lacking. We have previously taken a first step by proposing a generic cost model [23], which is one of the ingredients of the encompassing framework we presented and demonstrated in the current paper.

The interrelationships between processes, resources and cost are also highlighted in the reports produced by the International Federation of Accountants [14, 15]. Notwithstanding these works, few studies exist where a structured

approach to the analysis of cost factors in a process-aware information system is undertaken. Since the introduction of ERP systems, a number of studies have been conducted on the effects of ERP systems on traditional management accounting practices [1, 7, 8]. Recently, Vom Brocke et al. proposed an information model to link the ARIS accounting structure with ARIS process semantics using Event Driven Process Chains (EPC) [21].

Cost-informed operational process support is related to the notion of operational support studied in the context of process mining [19]. Examples are predictions and recommendations learned over historic event data and provided in an online setting. Current approaches are based on regression, decision tree analysis, annotated transition systems, and so on [19]. Operational support is also mentioned as one of the main challenges in the Process Mining Manifesto [9]. As shown in this paper, operational support based on cost considerations can be provided though an external cost service tightly coupled to the WfMS.

6 Conclusion and Future Work

The paper proposes a conceptual framework to enable workflow management systems to be cost-informed during enactment. In particular, we proposed how cost-based decision rules for process variant selections, activity related decisions (e.g., execution, cancellation, deferment), and resource assignment decisions can all be supported within a WfMS. We presented the technical challenges that need to be addressed to realise this level of support for a WfMS and proposed an architecture for cost-informed process execution. We also presented a realisation of such a cost-informed workflow environment using the YAWL workflow management system. We believe that our approach will enable organizations to more easily translate cost strategies into operational fulfilment using a WfMS and we have plans to evaluate the framework with stakeholders' input (e.g. through interviews and case studies).

This work takes an important step towards achieving a higher level of support for WfMSs in terms of the cost perspective. For the future, we are interested in the development of predictive capabilities that may help to project the cost that is incurred by alternative operational decisions. Furthermore, we are at this point reflecting on the incorporation of other non-functional concerns, besides cost, to enhance the operational support provided by a WfMS. We hope that our work inspires other researchers to more closely connect organisational strategic concerns with the practices and systems on the work floor.

References

1. P. Booth, Z. Matolcsy, and B. Wieder. The impacts of enterprise resource planning systems on accounting practice–the Australian experience. *Australian Accounting Review*, 10(22):4–18, 2000.
2. L. Chung and J. C. S. do Prado Leite. On non-functional requirements in software engineering. In *Conceptual modeling: Foundations and applications*, pages 363–379. Springer, 2009.

3. L. Chung, B. Nixon, E. Yu, and J. Mylopoulos. *Non-functional requirements in software engineering*. Kluwer, 2000.
4. R. Cooper and R. Kaplan. Measure costs right: Make the right decisions. *Harvard Business Review*, September - October:96–103, 1988.
5. L. M. Cysneiros, J. C. S. do Prado Leite, and J. d. M. S. Neto. A framework for integrating non-functional requirements into conceptual models. *Requirements Engineering*, 6(2):97–115, 2001.
6. Gartner. Improving business processes, 2010.
7. S. Grabski, S. Leech, and A. Sangster. *Management accounting in enterprise resource planning systems*. CIMA Publishing, 2009.
8. T. Hyvönen. *Exploring Management Accounting Change in ERP Context*. PhD thesis, University of Tampere, 2010.
9. IEEE Task Force on Process Mining. Process Mining Manifesto. In *BPM 2011 Workshops*, volume 99 of *LNBIP*, pages 169–194. Springer, 2011.
10. W. Kettinger, J. Teng, and S. Guha. Business process change: a study of methodologies, techniques, and tools. *MIS quarterly*, 21(1):55–80, 1997.
11. J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *Software Engineering, IEEE Transactions on*, 18(6):483–497, 1992.
12. M. Netjes, H. Reijers, and W. van der Aalst. On the formal generation of process redesigns. In *Business Process Management Workshops*, pages 224–235. Springer, 2009.
13. J. J. O'Sullivan. *Towards a precise understanding of service properties*. PhD thesis, Queensland University of Technology, 2006.
14. Professional Accountants in Business Committee. Evaluating and improving costing in organizations, July 2009.
15. Professional Accountants in Business Committee. Evaluating the costing journey: A costing levels continuum maturity model, July 2009.
16. H. Reijers and S. Mansar. Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega*, 33(4):283–306, 2005.
17. N. Russell, W. van der Aalst, A. ter Hofstede, and D. Edmond. Workflow resource patterns: Identification, representation and tool support. In *Proceedings of the Conference on Advanced Information Systems Engineering*, volume 3520 of *LNCS*, pages 216–232. Springer, 2005.
18. A. ter Hofstede, W. van der Aalst, M. Adams, and N. Russell, editors. *Modern Business Process Automation: YAWL and its support environment*. Springer, 2010.
19. W. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.
20. W. van der Aalst and S. Dustdar. Process mining put into context. *IEEE Internet Computing*, 16:82–86, 2012.
21. J. vom Brocke, C. Sonnenberg, and U. Baumoel. Linking Accounting and Process-Aware Information Systems - Towards a Generalized Information Model for Process-Oriented Accounting. *European Conference on Information Systems*, pages 1–13, 2011.
22. M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag, New York, Inc., Secaucus, NJ, USA, 2007.
23. M. T. Wynn, W. Z. Low, and W. Nauta. A framework for cost-aware process management: Generation of accurate and timely management accounting cost reports. In *Conferences in Research and Practice in Information Technology (CRPIT)*, 2013.