Market-Based Workflow Management

Andreas Geppert, Markus Kradolfer, Dimitrios Tombros

Technical Report 97.15
November 1997

Department of Computer Science, University of Zurich
{geppert|kradolf|tombros}@ifi.unizh.ch

Abstract

This paper presents market-based workflow management, a novel approach to workflow specification and execution which regards activities contained in a workflow as goods traded on an electronic market. Information about *expected* cost and execution time is considered for activity specifications, and is used at runtime to execute workflows such that *actual* cost and execution times are balanced and optimized. To that end, task assignment uses a bidding protocol, in which each eligible processing entity specifies at which price and in which time interval he/she can execute the activity. The winner of a specific bidding process is requested execute the activity, and earns the amount specified in the corresponding bid. Market-based workflow management thus not only allows to optimize workflow executions with respect to execution time and overall cost; but the trading of activities represents an incentive for processing entities to engage in a workflow.

1 Introduction and Motivation

Workflow management has recently found great attention in the information systems field, as it allows to capture knowledge about business processes, to define workflows in a formal language/framework, and to enact workflows according to their specification. Hereby, a *workflow specification* defines the structure of workflows (e.g., atomic activities/steps), processing entities responsible/capable of executing these activities, and further constraints such as execution or temporal dependencies. *Workflow management systems* (WFMS) are the software systems that support the specification and the execution of workflows.

Workflow management is often pursued as a consequence of business process reengineering, which attempts to restructure and optimize processes, to improve the quality of services and products, and to decrease time to market. Inherent to these objectives is the notion of optimality in the sense that *a process should produce the same result at a lower cost and/or in a shorter time* than was possible before process redesign. Alternatively, a result of a higher quality may be produced, in which case quality has to be defined in a business-dependent way.

Some WFMS allow the definition of expected execution times and costs for workflows (e.g., Ultimus [21]) which can be used for simulation purposes. Others are able to recognize that a workflow will probably not meet a deadline, and can then try to "escalate" the workflow execution [15]. However, the objective of cost/time optimization is typically not addressed by WFMS, neither at the workflow specification nor at the execution level.

In this paper, we present a framework for the consideration of cost/time optimization strategies that is based on the principle of *agoric open systems* [13] (or *computational ecologies* [8]). In such systems, (software) components are regarded to act as buyers or sellers in a market. The basic idea of the approach is that objects provide and consume computational resources (such as processor time or memory). Objects that consume a resource have to pay for it in some notion of electronic currency, and sellers are compensated for providing goods and services. Prices can thereby depend on how fast a service is provided (i.e., the faster a service is provided, the higher its price).

We show that these principles can be beneficially applied to workflow management. Concretely, services that a processing entity provides (i.e., activities it can execute) are treated as resources that must be paid for. At specification time, execution cost of required services are specified. At execution time, an open market mechanism based on bidding is used to assign activities to specific processing entities (PE). A PE capable of providing a service can bid for it by naming a price and an expected response time. Depending on the accumulated time and cost of the execution of the workflow until bid time (i.e., depending on whether cost or time is critical), an appropriate PE can then be chosen.

This market mechanism must be supported at different levels. First, the workflow specifications must allow the calculation of expected costs and execution times. Second, the processing entities (both customers and service providers) must be aware of the bidding process and follow a certain service request and provision protocol. Third, the workflow must be executed in an environment which provides support for market-based workflow execution.

The contributions of this paper, thus, are:

- the application of a market-based approach which allows to consider cost and time characteristics of workflows in their specifications,
- the implementation of a market mechanism in the workflow engine which uses information on execution cost and time and balances them for the entire workflow execution,
- the use of payments for service executions as incentives for processing entities to execute activities.

The remainder of this paper is organized as follows: in the following section, we give an overview of market-based workflow management and identify the underlying assumptions. In section 3, we introduce the specification of (market-based) workflows. Section 4 describes our approach towards the specification of processing entities as well as its extensions for market-based workflow management. Section 5 presents extensions for the execution level. Section 6 surveys related work, and section 7 concludes the paper.

2 Market-Based Workflow Management

In this section, we give an overview of agoric workflow management and identify assumptions that have to hold in order to make the market mechanism work.

The principle of agoric workflow management is to establish a market in which activities (or services) contained in a workflow are traded goods. Thus, workflows act as consumers by buying services from processing entities (the sellers). Each service offered by a seller is characterized by the time period the PE needs to execute it, and by a price. Prices are specified in units of some electronic currency, and PE have an account storing the money they have earned.

The market mechanism then helps to optimize each single workflow execution in such a way that total execution time and overall aggregated costs are balanced and remain within predefined limits. This objective is achieved by using a bidding protocol for task assignment, and by taking the expenditures and execution time of the workflow into account when selecting the best bidder for a concrete activity to be assigned.

Agoric workflow management affects all levels of workflow management. On the specification level, information about cost/time of activity executions must be defined. This information is later on required in order to determine the optimal bid with respect to the actual expenditures and execution time. At the service provision level, PE must be able to participate in a market; they thus must be enabled to bid for activity executions (i.e., compute the time and cost at which they can provide their services). At the workflow execution level, the execution engine must implement market-based workflow management by keeping track of execution times and expenditures of workflows, and by acting as a trader of activity executions.

In order to put agoric workflow management to work, several underlying assumptions have to be met. We assume that for each activity type, the following information is known:
(1) its expected execution time,
(2) the expected execution cost.
These assumptions are needed in order to determine at runtime whether expenditures and/or execution time lie within the predefined limits. For PE, we assume that
(3) if multiple of them are eligible for a certain activity, then the result of executing the activity by any one of them is of equal quality,
(4) for at least several activity types, there are multiple PE capable of executing them,
(5) PE do not agree on prices for activity executions.
In the current form, our model does not consider quality of work in workflow executions. Assumption (4) ensures that no monopoly exists. In fact, this assumption is expected to be met by many workflow systems that involve human processing entities performing some intellectual task. Assumption (5) implies that PE decide on prices independently of each other. This assumption is also basic in free markets, since its violation implies the existence of cartels.

Subsequently, we introduce our approach to market-based workflow management. For each of the three levels, we first introduce its basic concepts as needed for the sake of comprehension, and then show how it can be extended to accomplish market-based workflow management.

3 Workflow Specification

3.1 Workflow Specification

In general, market-based workflow management is applicable regardless of the concrete characteristics of the underlying workflow meta-model. In the following, the TRAMs workflow specification approach [10] is used. A workflow is either a basic step, called *activity*, or a *complex workflow* consisting of subworkflows. Activities are carried out by *processing entities (PE)* —humans or software systems. Each workflow is an instance of a *workflow type*, i.e., activities are instances of *activity types* and complex workflows are instances of *complex workflow types*. A workflow type has a unique name, a set of input parameters, and a set of end states with output parameters. A complex workflow comprises a set of subworkflows (activities and/or complex workflows). For each subworkflow, an optional *start condition* specifies when it can be started. Complex workflow types can have an *end condition* for each end state, specifying when the end state can be reached. Furthermore, data flows among workflows are specified by associating parameters of different workflows. For an activity, it can be defined by which processing entity it has to be executed.

Fig. 1 shows an example of a workflow type which specifies the processing of a health insurance claim (HIC). The workflow type is represented by a black rectangle that contains a special kind of graph whose oval nodes represent the subworkflows to be executed. The end states of the subworkflows and of the workflow type are represented by gray rectangles. Black solid arrows indicate which end states of the subworkflows are referenced by which start conditions of the subworkflows. Black dashed arrows indicate which end states of the subworkflows are referenced by the end condition of the workflow type.

A HIC is processed as follows. Once a HIC is received, a clerk creates a file containing the diagnosis, the treatments, the costs, and the insurance contract number (activity `enterCase`). If no insurance contract exists for the applicant, the HIC is rejected, i.e. a notification of the rejection is produced by a clerk (`rejectHIC`). If the costs do not exceed a certain limit (e.g. 300 SFr.), the claim is accepted and a payment check is printed (`acceptHIC`). This step is also carried out by a clerk. If an insurance contract exists and the costs are above the limit, two activities are done in parallel. A software system checks whether some of the treatments are contained in a blacklist (`checkBlacklist`), and a medical specialist verifies if the treatment makes sense with respect to the patient's history (`checkHistory`). If any of these two checks fail, the HIC is rejected. Otherwise, the HIC is accepted. The workflow ends after the HIC is either accepted or rejected.

3.2 Specification of Market-based Workflows

The specification of market-based workflows requires additional information. For each activity type, it must be specified how much time it will take to execute an activity of this type. Similar-

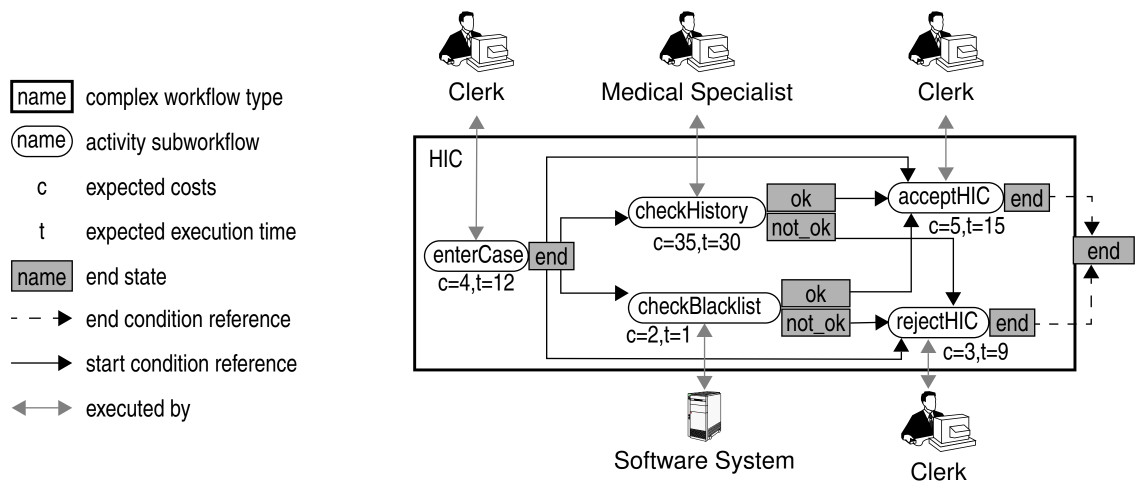

Figure 1: Workflow Type HealthInsuranceClaim

ly, the price of an activity execution of this type has to be specified in currency units. Both measures should be chosen in such a way that “normal” activity executions can stay within the cost and time limits (i.e., simply taking the average of all past execution times/costs might not be appropriate). However, as we will discuss below, either one or both values can be adjusted in case the initial specification is not appropriate.

Note that we require the specification of cost and time measures only for activity types, although at runtime we need information about the aggregated planned execution time and expenditures until each point in a workflow execution. For instance, before an execution of the activity `acceptHIC` is scheduled, we need information about real/expected execution time and cost of the entire workflow. We do not compute aggregated expected time/cost at buildtime, since this would impose rather severe restrictions on legitimate workflow structures (e.g., concerning joins and splits). Instead, this information is computed at runtime (see section 5.2).

Assignments of activities to roles or processing entities also has to be extended. In this respect it can be indicated for each assignment whether it should be market-based. If this option is specified, the appropriate PE will be selected at runtime using the bidding protocol. Note that for some tasks (e.g., simple activities or automated routine jobs), the workflow designer may decide to assign executions of it in a conventional way, e.g., using a round-robin policy.

Ultimately, note that the cost and time measures are necessary for all activity types involved in a workflow. This is because even if an activity is not scheduled based on bidding at runtime, it can still contribute to exceed time and cost budgets.

4 Brokers and Services

Workflows are executed by PE which are typically people or software systems. These PE together with their cooperation infrastructure comprise a workflow system (WS). The complete definition of a WS architecture includes:

- *components* representing PEs that take part in one of the workflows — the structural view,
- *tasks* that can be performed by those components — the functional perspective, and
- *rules* about how, when and under which constraints the components can/must perform the activities — the behavioral perspective.

The BROKER/SERVICES MODEL (B/SM) [18, 19] is used to describe the software and process architecture of the resulting WS. In B/SM, a workflow specification is enriched with the necessary elements to provide executability. From the perspective of B/SM, a WS consists of interacting, reactive components called *brokers*[1] representing participating PE. Broker behavior is defined by ECA-rules describing their reaction to simple events (e.g., service requests) or composite process-specific situations (e.g., a request within a specific time interval) in the action-part of the rules. Different types of brokers represent user agendas, organizational groups, applications with open interfaces, wrapped applications, and custom WFMS components.

Brokers also have a *state* consisting of typed variables and messages they understand. Typically, workflow activities consume, manipulate, and produce output data. This production data

1. The notion of “broker” should not be confused with “object request brokers” as proposed by the Object Management Group (OMG). For a discussion of this issue see [6].

must flow through a workflow. Most of it is stored in external systems which may not be integrated in the WFMS. Brokers may export production data to the WFMS by including it in their state declaration.

The functionality of a WS is described by a set of *services* specified by a signature consisting of the service name, a set of parameters, and the replies and exceptions the request may cause. Service execution is started by a corresponding request from the service client and is terminated when a reply or one of the exceptions defined for the service is generated by its provider. Broker interaction is based on multicasting of parameterized events. A broker needing a service generates a request event to which a server broker will react generating a reply event. In abnormal situations, an exception event will be generated. Workflows are executed through brokers by providing the requested services. A workflow instance starts executing when its initiation request event occurs.

A specific service in a given WS is provided by many different brokers. In general, the association between a service and its provider is established by means of *1:n* relationships called *capabilities* implying the existence of a predefined behavior for a broker whenever the service is requested. Capabilities are defined through ECA-rules of the form:

```
ON service-request-event(service, parameters)
[IF condition]
DO execute service
   provide reply
```

4.1 Adding Market-Oriented Behavior to Reactive Components

Depending on its provider, a service may have different cost and response times. In market-based workflow management, the interesting question is which service provider to choose in case there are multiple eligible ones. In order to support cost/time optimization and corresponding selection of the service provider, potential server brokers have to publish information about their service execution attributes. This takes place during the actual workflow execution through an appropriate bidding protocol. Conceptually, this protocol assumes the existence of a *service market broker* which is aware of the actual execution time and cost.

The participation of brokers in market-based workflow management is enabled by extensions to their state and their behavior. Concerning state, each broker has an *account* attribute collecting the amount of electronic money it has earned by service executions.

In addition to the previously mentioned event types, there are three new ones: *bid-request*, *bid-reply,* and *bid-confirm*. In order to participate in the bidding process, each involved broker must offer an appropriate behavior, i.e., it must be able to react to events of type bid-request, and as a response it can generate events of type bid-reply. The former prompts each eligible broker to determine its bid and to send it back to the market broker. Each instance of bid-request carries the information which is necessary to compute the bid (i.e., the service name and the request parameters). Additionally, two further parameters indicate to which extent the cost limit (planned execution time) has been overdrawn so far.

The second event type (bid-reply) is generated by bidding brokers to publish their bid. Parameters include the identification of the request and the actual bid, which is a pair (cost, time). Brokers can react to bid requests in one of two ways: humans can define that bid-requests are

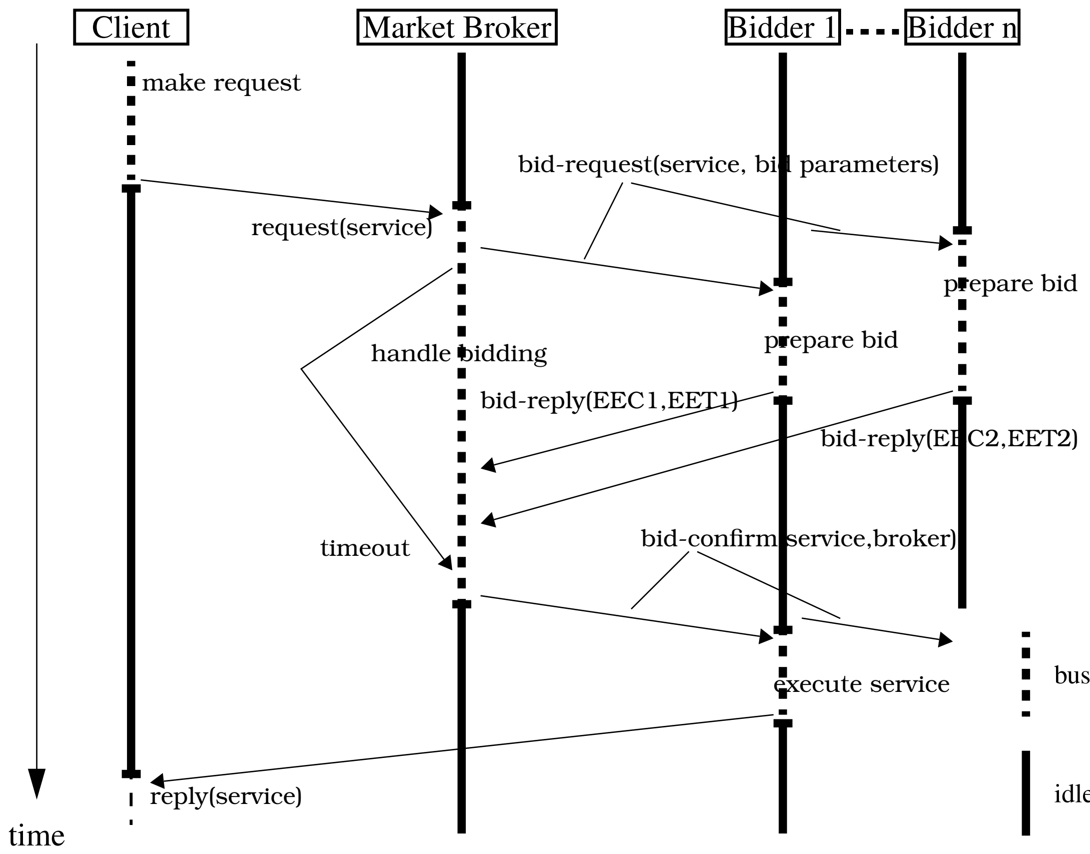

Figure 2. The bidding protocol

appended to a bid request queue, and that actual bids are determined manually. Brokers representing software systems can define reactions to bid requests in the form of ECA-rules:

```
ON bid-request(service, bid_parameters)
DO ... //compute cost and time required to execute service
   bid-reply(execution cost, execution time)
```

The bidding protocol proceeds as follows: the client posts a request to the service market by generating an appropriate request event. The service market broker reacts to this event by generating a bid request event with a bidding time limit. Brokers which have bidding rules defined for that service may then react to the bidding request. Their bids are collected by the service market broker until the time limit expires. It then chooses the winning bidder and notifies all bidders by generating a bid confirmation event. The winner then knows that it must provide the service and starts service execution. Fig. 2 shows the events generated during the protocol execution by the participating brokers. Distribution issues are abstracted at this point (i.e., a global time is assumed as well as reliable FIFO communication between brokers).

The efficiency of this protocol depends on the communication delays between brokers and on the length of the bidding interval. It can be improved if the bids have been prefetched so that at the request time the service market broker can directly confirm which broker makes the best offer for the service execution.

5 Market-Based Workflow Execution

5.1 Workflow Execution in EVE

Brokers and consequently, workflows are executed through the use of the distributed event engine *EVE* [6]. EVE's major purpose is to support brokers by providing event management, storage, and notification functionality. The principle by which brokers use EVE to execute workflows is the following (Fig. 3): broker ECA-rules are translated to rules stored and executed in EVE. Whenever a broker generates a primitive event, it notifies its local EvE-server about the occurrence. EVE then performs composite event detection and determines brokers that should be notified for such primitive and potentially detected composite event occurrences which may have resulted from this event. All these brokers are then appropriately informed and react as defined by their ECA-rules (whereby these reactions can in turn generate new events, and so forth).

5.2 Activity Execution in Market-based Workflow Management

In order to implement market-based workflow execution, the protocol sketched above needs to be extended. The aim is to trade activity executions in such a way that the "optimal" bidder is assigned the activity execution. This is accomplished by the procedure described below. In order to find the optimal bid, information about the current (aggregated) execution cost and time of the workflow instance have to be taken into account (and thus have to be consecutively computed at runtime). The following measures are required for each workflow instance wf:

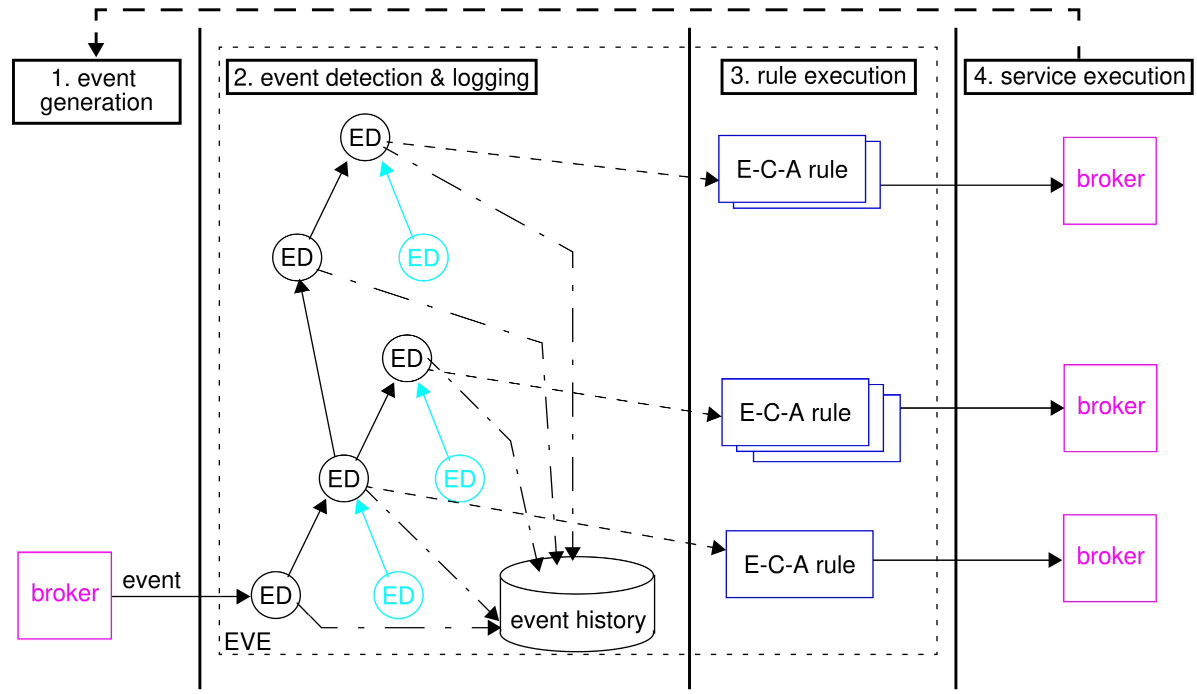

Figure 3. The workflow execution process in EVE

- C_{real}(wf, A) are the actual aggregated costs spent in wf until A's scheduling,
- $C_{planned}$(wf, A) are the aggregated costs planned for wf until A's scheduling,
- T_{real}(wf, A) is the actual execution time of wf until A is scheduled,
- $T_{planned}$(wf, A) is the expected execution time of wf until A is scheduled.

These measures are computed as follows.

Computation of aggregated execution time. Events have two additional attributes, one for expected execution time, and one for effective execution time. In order to compute the measures for execution time, we consider the corresponding pair of service request and reply events. When the reply event occurs, the planned execution time as specified for the activity type is added to the planned execution time given in the request event. The same computation is made for effective execution time. Both new measures are stored in the attributes of the reply event and are thus visible in other events that contain the reply event or refer to it. In case of composite events (which, e.g., represent "join"-nodes of parallel branches), the appropriate measures have to be taken from the longest of the parallel branches, i.e., in this case, the time measures are determined by the largest planned/effective execution time (this is obtained as the maximum of the measures as stored in the component events' attributes).

Computation of aggregated cost. The cost measures are also computed successively at runtime. As soon as an atomic activity a is terminated, the actual cost is computed as C_{real}(wf, a) plus the execution cost of a. Similarly, the planned costs is the sum of $C_{planned}$(wf, a) and the planned execution cost of `a`.

Summarizing, the activity execution cycle of Fig. 3 is extended as follows:
(1) In phase 2, compute the four measures described above,
(2) In phase 3, apply market-based task assignment if this is required by the workflow specification. This step will be detailed immediately.

5.3 Protocols for Market-based Task Assignment

The market mechanism actually comes into play during task assignment. Assume an activity is to be assigned to some PE, and that the assignment should be based on bids. The task assignment protocol then is defined as follows:
(1) Determine the set of eligible PEs.
(2) Send the bid request for the activity to each of them.
(3) Collect bids for a certain (pre-defined) period of time. Each bid is specified by a pair `B= (P, T)`, where `P` is the price (specified in currency units) at which the bidder is willing to execute the activity. `T` is the time (in time units) the PE will need to execute the activity.
(4) Evaluate bids, i.e., select the "optimal" bid. The PE who has sent the optimal bid is called the "winner".
(5) Notify the winner (and all other bidders).

The fourth step (bid evaluation) is based on information about the aggregated workflow cost and execution time (the four measures described above). This information is then used to compute the amount of time saved or overdrawn, and the amount of money saved or overdrawn. Based on these computations, task assignment can then stress cost or time. This is, when execution time up to a certain point has been minimized at the expense of an overdrawn cost limit, then in the subsequent steps task assignment would weight costs more than time.

These measures are used to determine whether and to which extent the workflow execution has exceeded time and or cost limits. The *cost overrun ratio* of wf_A is defined as

$$\mathrm{COR}(wf, A) = \frac{C_{real}(wf, A)}{C_{planned}(wf, A)}$$

and the *time overrun ratio* is defined as

$$\mathrm{TOR}(wf, A) = \frac{T_{real}(wf, A)}{T_{planned}(wf, A)}$$

Bids are then evaluated according to the following formula:

$$\mathrm{winning\_bid} = \min(P_i^{\mathrm{COR}(wf, A)} \times T_i^{\mathrm{TOR}(wf, A)})$$

COR and TOR are used in the exponents in order to weight cost and/or time. If cost (time) has been overrun, then the difference in price (time) component from multiple bids will be increased (while we refrain from choosing the bid with the minimal price/time regardless of the other component). Thus, even if (say) cost has been overrun, then we still might choose a bid with a non-minimal price, provided that its specified time is by far the smallest one.

In our example, we assume that within a particular HIC-instance `wf1`, the activity `enterCase` was executed within 9 time units at the price of 6 currency units. When `checkHistory` is scheduled, `COR(wf1,checkHistory)` = 1.5 and `TOR(wf1,checkHistory)` = 0.75. Three PE make the following bids for `checkHistory`: (40, 20), (30, 35), and (28, 40). The winning bid is computed as the minimum of {2392.56, 2364.47, 2356.58}, i.e., the third one is the winner.

Note that for each single bidding process, it is determined whether either cost or time are more critical for the workflow in question. This means that costs are stressed whenever actual costs have exceeded expected costs, and execution time stayed within the limit. Additionally, the fact that for a workflow cost or time are weighted more can change arbitrarily often within a workflow.

5.4 Calibration of Cost Limits, Deadlines, and Bids

For a workflow type it might happen that for many of its instances, deadlines and/or cost limits cannot be met. The reaction of EVE in case either one of them is exceeded is to optimize accordingly. If, however, both of them have been exceeded repeatedly within a workflow execution, it can be suspended and the workflow administrator will be informed. He/she can then decide whether the workflow should resume execution, or be aborted.

The reason for exceeding cost limits and deadlines can be that the handled case is much more complex than the average case for which the workflow is designed (e.g., a HIC with a huge number of treatments, or a complex diagnosis). However, the reason can also be that budgets/deadlines for single activities have been calculated too sharply. This kind of inappropriateness should then be remedied by adjusting the corresponding cost limits and/or execution time of the responsible activity. Alternatively, if for example deadlines cannot be extended for some reason, then a possible remedy would be to increase the cost limit for the problematic activities, or to add processing entities being capable of executing the activity. The rationale behind the latter is that bids tend to decrease whenever the set of bidders grows larger.

In order to calibrate a workflow system, the workflow administrator needs information about problematic activities, i.e., those that repeatedly exceed expected costs and deadlines. In order to provide this information, the functionality of post-mortem analysis [7] must be extended. Then, post-mortem analysis also answers queries such as “how often did activity executions overdraw their budget and deadline within executions of a certain workflow type”.

Furthermore, the processing entities also might need to adapt their bids, depending on how often they win the bidding process. A natural reaction (of humans) to repeatedly not winning bids is to lower prices in the future. Similar behavior is required for PE which are software components; they can implement adjustment of bids in ECA-rules.

6 Related Work

Current WFMS allow to specify deadlines, some even allow the adjustment of deadlines. However, we are not aware of any WFMS that allows to specify information about costs and execution times and to use this information together with the knowledge about the current workflow state for task assignment.

The work reported in [15] addresses (only) execution times and tries to minimize costs occurring due to workflow escalations (note, however, that the notion of “cost” there does not bear any budget or financial meaning as in our model). Similar to our model, during task assignment in FlowMark [11], requests are sent to all eligible participants. In contrast to our model, task assignment is then not based on bids of PE, but the task is assigned to that participant whose affirmative reply is received first.

Market-based mechanisms have been used for the assignment of resources and scheduling of tasks. The work reported in [4, 13, 22] describes how resources such as processor slices or memory can be assigned using market-based techniques. There, processes (or their users) are clients bidding for the usage of a resource, and machines (or their owners) sell these resources at some price. Thus, these approaches typically consider one “seller” and multiple buyers who bid for the resource or a service. In contrast, our model is characterized by the presence of one buyer (at a certain point in time), but several potential providers. Therefore, the techniques proposed for agoric open systems cannot be directly applied to our case.

A market-based approach has also been proposed for query processing and optimization in Mariposa [17], a distributed database management system. In Mariposa, there is also one buyer (a query) and several providers which can answer (parts of) the query. Each query has a budget, and the query optimizer then tries to answer the query as fast as possible while staying within the budget limits. The major difference between Mariposa’s approach and our model is that budget and time limits are known and considered only for a single step (i.e., a query), while in workflow management the problem is apparently to optimize the utilization of both, time and cost in such a way that the past execution time exceeded and expenditures can be taken into account (so that both limits are more likely to be met for the entire workflow).

The CORBAservices [14] specify a trader service, which however does not cover bidding or billing. Recent research in electronic markets has also focussed on trading of services in distributed object systems [e.g., 12]. Recent research in electronic markets has also focussed on trading of services in distributed object systems [e.g., 12]. However, to the best of our knowledge, this research considers only single services, but not entire workflow executions.

7 Conclusion

This paper introduced a novel approach towards market-based workflow management. We have shown how workflow specifications have to be extended by deadlines and budgets (for activities), and what additional information and behavior needs to be offered by processing entities in order to participate in market-based workflows. We have discussed extensions to an execution engine which enable it to schedule workflow executions in a market-based way and to optimize workflow executions by assigning tasks to PE based on bidding.

Thus, the contribution of this work is that it allows to consider workflows from a market perspective, and that it leverages the concept of electronic markets to workflow management. This approach is feasible for workflow systems in which many PE participate and tasks to be executed are complex and/or intellectually demanding (it would be less meaningful for workflows consisting of only small routine jobs). Typically, humans will be involved in such workflow systems. In this respect, one should be aware of the (social) impacts of this approach on humans—which then act as a system of entrepreneurs by marketing their services—involved in a workflow. One might criticize this impact (e.g., because the amount of "money" earned in workflows might be treated as a performance indicator), but we would argue that this market-based approach leads to more flexibility and self-responsibility of human PE (the lack of which has been identified as a major shortcoming of current approaches to workflow management and software processes [1, 16]). Particularly, through the bidding-based task assignment, humans can decide much more freely which tasks they like to perform (in these cases, they should send cheap bids) and which they do not (for those they send no or expensive bids).

In our future work, we will extend this work in various ways. First, so far *atomic activities* can be scheduled according to cost/time considerations. In inter-organizational workflow management it is however conceivable that (sub)workflows are "outsourced" to other companies. In such cases, it should be possible to apply the market mechanism to entire subworkflows, too. While the bidding protocol itself would still apply without extensions, workflow specifications need to be extended by the definition of expected time/cost for subworkflows (on the client/superworkflow side). On the "server" side (i.e., the subworkflow), the WFMS should be able to participate in bidding, i.e., to compute cost and time of the requested subworkflow execution. For that matter, the specification of expected cost and execution time needs to be leveraged from single activities to entire workflows.

The second extension will address quality of service in specifications and in workflow executions.

8 References

1. K.R. Abbott, S.K. Sarin. Experiences with Workflow Management: Issues for the Next Generation. Proc. ACM 1994 CSCW, Chapel Hill, NC, October 1994.
2. N. Barghouti, E. Koutsofios, E. Cohen. Improvise: Interactive Multimedia Process

Visualization Environment. Proc. 5th ESEC, Sitges, Spain, September 1995.
3. C.J. Bussler. Policy Resolution in Workflow Management Systems. Digital Technical Journal 6:4, Fall 1994.
4. K.E. Drexler, M.S. Miller. Incentive Engineering for Computational Resource Management. In [8].
5. D. Georgakopoulos, M. Hornick, A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. Distributed and Parallel Databases 3:2, April 1995.
6. A. Geppert, D. Tombros. EvE, an Event Engine for Workflow Enactment in Distributed Environments. Technical Report 96.05, Department of Computer Science, University of Zurich, May 1996.
7. A. Geppert, D. Tombros. Logging and Post-Mortem Analysis of Workflow Executions based on Event Histories. *Proc. 3rd Intl. Workshop on Rules in Database Systems* (RIDS), Skoevde, Sweden, June 1997. LNCS 1312, Springer.
8. B.A. Huberman (eds). *The Ecology of Computation*. Studies in Computer Science and Artificial Intelligence, North-Holland, Amsterdam 1988.
9. S. Jablonski, C. Bussler. Workflow Management. Modeling Concepts, Architecture, and Implementation. International Thomson Computer Press, London 1996.
10. M. Kradolfer, A. Geppert. Modeling Concepts for Workflow Specification. Technical report, Departement of Computer Science, University of Zurich, April 1997.
11. F. Leymann, W. Altenhuber. Managing Business Processes as an Information Resource. *IBM Systems Journal* 33:2, 1994.
12. M. Merz, K. Mueller-Jones, W. Lamersdorf. Agents, Services, and Markets: How do they integrate? Proc. IFIP/IEEE Int'l Conf. on Distributed Platforms, Dresden, Germany, February 1996.
13. M.S. Miller, K.E. Drexler. Markets and Computation: Agoric Open Systems. In [8].
14. CORBAservices: Common Object Services Specification. OMG, July 1997 (http://www.omg.org/corba/sectran1.htm).
15. E. Panagos, M. Rabinovich. Reducing Escalation-Related Costs in WFMSs. NATO Advanced Study Institute (ASI) on Workflow Management Systems and Interoperability, Istanbul. Turkey, August 1997.
16. I. Sommerville, T. Rodden. Human, Social and organizational Influences on the Software Process. In A. Fuggetta, A. Wolf (eds). *Software Process*. Trends in Software, John Wiley & Sons, Chichester, 1996.
17. M. Stonebraker, P.M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, A. Yu. Mariposa: A Wide-area Distributed Database System. VLDB Journal 5:1, 1996.
18. D. Tombros, A. Geppert, K.R. Dittrich. Brokers and Services: Constructs for the Implementation of Process-Oriented Environments. In B. Freitag, C.B. Jones, C. Lengauer, H.-J. Schek (eds). Object-Orientation with Parallelism and Persistence. Kluwer Academic Publishers, 1996.
19. D. Tombros, A. Geppert, K.R. Dittrich. The Broker/Services Model for the Design of Cooperative Process Oriented Environments. Technical Report 97.06, Department of Computer Science, University of Zurich, June 1997.
20. D. Tombros, A. Geppert, K.R. Dittrich. Semantics of Reactive Components in Event-

Driven Workflow Execution. *Proc. 9th Intl. Conf. on Advanced Information Systems Engineering*, Barcelona, Spain, June 1997.
21. Ultimus Product Guide. Ultimus Inc., Raleigh, NC, 1997 (http://www.ultimus1.com/).
22. C.A. Waldspurger, T. Hogg, A. Huberman, J.O. Kephart, W.S. Stornetta. Spawn: A Distributed Computational Economy. IEEE Trans. on Software Engineering 18:2, Feb. 1992.