

The Auto Optimizer

**Robert M. Shapiro and Hartmann Genrich,
Global 360, United States**

ABSTRACT

Continuous Process Improvement is touted as a feature of many Business Process Management suites. Usually this means the provision of analytical techniques for measuring performance. These include Business Activity Monitoring, Balanced Score Cards, real-time measurement of Key Performance Indicators and the capture and analysis of event streams generated by the running system.

Some BPM suites provide a what-if simulation capability which allows the evaluation of changes to the system. This leaves the task of coming up with proposed solutions to the user, an often daunting task.

In this paper we describe an automated, goal-driven technology for process improvement. By focusing on the common characteristics of business processes in typical BPM applications, we have developed an integrated set of algorithms for generating and evaluating proposed solutions. The user specifies the desired goals in terms of performance or cost or KPIs. The algorithms run until the goal is achieved or no further improvement is found.

INTRODUCTION

Business Process Management Suites frequently include an analytics component for collecting and analyzing the log events generated by the process execution engine(s) and a simulation component that allows the exploration of alternative scenarios where the resourcing and/or the processes and/or the workload are altered in an effort to improve the overall performance of a complex business process.

The analytics offers a window in which one can explore what has happened or 'will' happen. Process improvement experiments are suggested by the analytics data and evaluated by the new data they generate. It is often the case that OLAP technology is employed to allow an analyst to rapidly explore the data and from the exploration come up with suggestions for improvement which can then be tried out (evaluated) using the simulator to generate new data.

The improvement process can be characterized as the following set of steps:

1. Analyze the historical (or simulation) data to determine where improvements are needed. If the evaluation is satisfactory, stop.
2. From the data, come up with an idea for improvement.
3. Create a simulation scenario incorporating that idea and run a simulation.
4. Evaluate the results to determine whether it has resulted in improvement and repeat the cycle.

Step 2 in the improvement process is typically a challenging manual step. In this paper we describe an approach to automating the entire cycle described above.

The optimization technology is intended to be used to improve the performance of a Business Process Management System. We provide a brief overview of the elements in such a system. Data collection and analysis is critical in Process improvement. We describe log events which are the source of the data and a brief overview of Analytics. Simulation plays a critical role in evaluating alternatives.

We provide a brief overview of Simulation. We review the basic Optimizer. It provides methods for improving throughput, lowering costs and satisfying Service Level Agreement criteria. It uses critical path analysis, resource load balancing and shift assignments to accomplish this. We present an overview of the Auto Optimizer. It uses the technology in the optimizer to achieve goals provided by the user. The construction and evaluation of alternative scenarios is automated and the optimization cycle repeats until the user-stated goals are achieved or no further improvement can be found. We follow this with details of a particular analysis method that we employ in the Auto Optimizer: Critical Path Analysis. Finally we present a summary and outlook on future developments.

BPMS OVERVIEW

A Business Process Management Suite typically includes an analytics component for collecting and analyzing the log events generated by the process execution engine(s) and a simulation component that allows the exploration of alternative scenarios.

We have added an optimizer component to make recommendations for changes to business operations.

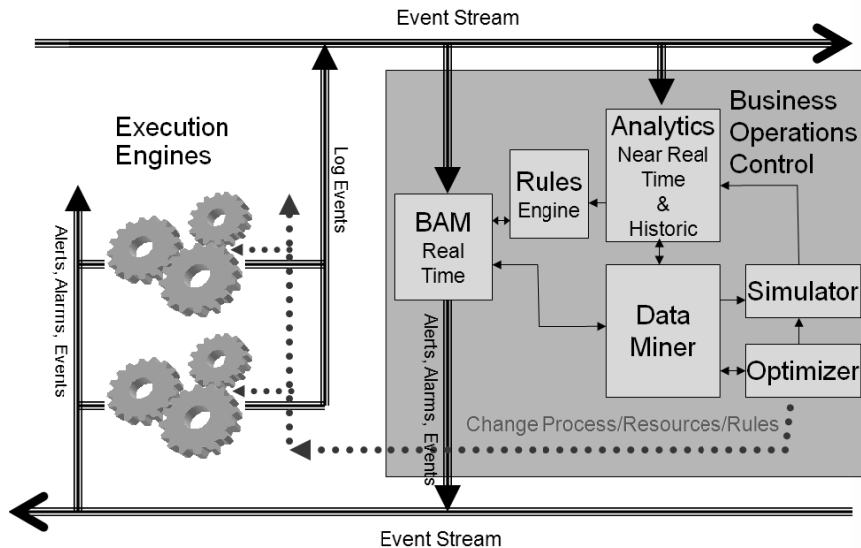


Figure 1: BPMS Modules

A stream of log events is generated by various execution engines, which include:

- Workflow enactment (BPM systems)
- BPEL
- Supply Chain Management
- ERP Systems (e.g. SAP)

The events are processed by:

- A BAM module to display KPI dashboards and generate alerts in real time. Makes use of a Rules Engine.
- An Analytics module which collects and organizes the event data.

An Optimizer module makes use of the analytics data, employing data mining and simulation, to improve the business operation:

- Resource schedules are altered and resources redeployed.
- Processes are modified to improve performance, control risk.

- Rules are changed to handle emergencies.

Elements

Analytics

- Storage and processing of operations data to provide useful business information.

BAM

- Business Activity Monitoring. Real time depiction of Key Performance Indicators (**KPIs**) and generation of alerts etc.

Data Miner

- Statistical processing of business information to extract patterns, detect trends and make predictions.

Event Stream

- Flow of information packets about meaningful state changes. The log events generated by currently running processes are an example.

Execution Engine

- Any business process enactment service, including workflow engines, BPEL executor, ERP system etc.

Optimizer

- A software tool that utilizes the Data Miner and/or Simulator to make recommendations for changes to business operations.

Rules Engine

- A software system that helps manage and automate business rules. Used to detect interesting business situations automatically.

Simulator

- Supports what-if analysis of current or planned operations.

Log Events

Process execution engine(s) and simulations generate a stream of log events to be collected and analyzed by the analytics component.

- Process events occur at the start and termination of a process instance.
- Activity events occur at various points in the execution of an activity. See next section for details.
- A timed sequence is associated with two activities, possibly in different processes, through which a work item passes

Figure 1 shows an example of a single log event, WORKFLOWCREATE. Figure 2 is a list of typical events, grouped by the event category. The timing data associated with log events are depicted in Figure 4.

```
<XPDLogEvent
System="MortgageDemo"
Scenario="Mortgage Lending AsIs"
Run="8/28/2006 6:29:01 PM"
InstanceId="6505"
  ParentInstanceId=""
  WorkflowInstanceId="6505"
  Timestamp="2006-08-01T07:00:00Z"
  SequenceId="281010"
  ProcessId="Mortgage Lending AsIs"
  ProcessVersion="1"
```

```

    EventType="WORKFLOWCREATE"
    ActivitySetId="1"
    ActivityId="15" QueueId="-1"
ElapsedTimeDays="0" ElapsedBusinessHours="0"
ElapsedBusinessDays="0" AccruedWaitDays="0"
    AccruedWaitBusinessDays="0"
    AccruedWaitBusinessHours="0"
    AccruedProcessingDays="0"
    AccruedProcessingBusinessDays="0"
    AccruedProcessingBusinessHours="0">
<Participants>
    <Participant ParticipantId="System" />
</Participants>
<DataFields>
    <DataField Name="SIM_Cost" Type="FLOAT">0</DataField>
</DataFields>
</XPDLogEvent>

```

Figure 1: XML for a Single Log Event: WorkFlowCreate

| Event | Event Group |
|-----------------------|----------------|
| WORKFLOWCREATE | Process |
| WORKFLOWTERMINATE | |
| CHILDCREATE | |
| CHILDTERMINATE | |
| ARRIVEACTIVITY | Activity |
| BEGINACTIVITY | |
| COMPLETEACTIVITY | |
| SUSPENDACTIVITY | |
| CANCELACTIVITY | |
| CONTINUEACTIVITY | |
| BEGINTIMEDSEQUENCE | Timed Sequence |
| COMPLETETIMEDSEQUENCE | |
| LOGGEDEVENT | Logged Event |

Figure 2: Types of Log Events

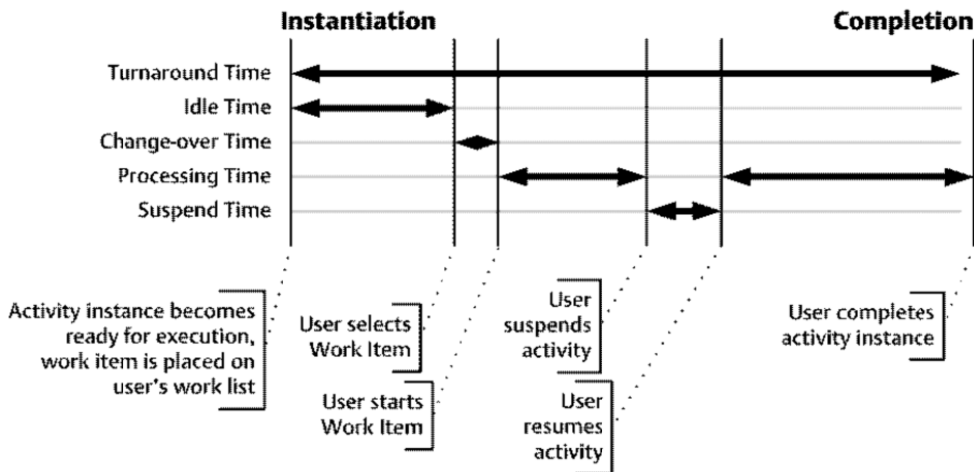


Figure 4: Timing Data from Log Events

Analytics

Figure 5 depicts a possible architecture for the analytics. The events generated by the process engine are collected in a relational data base and the data is used to populate the fact and dimension tables of OLAP cubes. Queries on the cubes support slice and dice charting and rapid reporting.

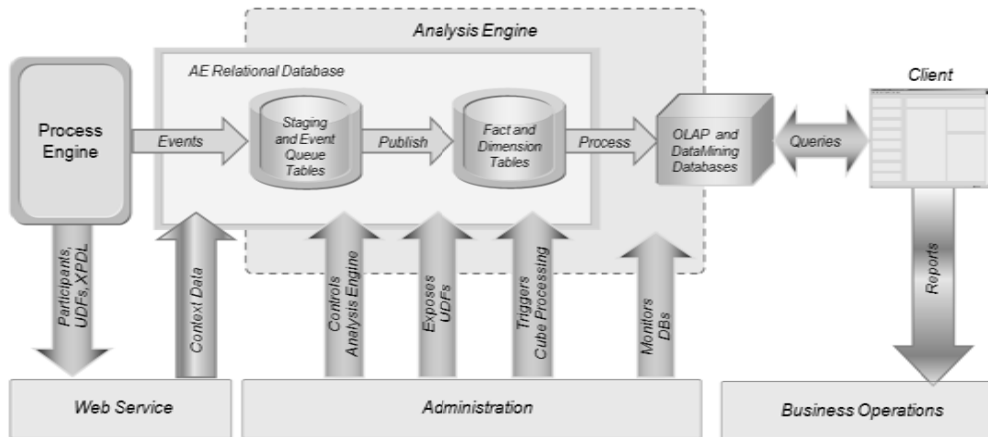


Figure 5: Analytics Overview

Simulator

The typical simulator for making predictions in a BPMS environment is a discrete event simulator. Preparation of a simulation run is commonly referred to as a scenario. The scenario consists of:

- The set of processes to be simulated. The process definitions may be represented in a format that supports process interchange, such as XPDL. Process definitions may be created by a diagram editor that is part of the BPM system, or by an independent tool such as a BPMN editor. Process definitions provide information about the activities performed, the routes taken, the rules impacting which routes and activities to perform, and the resources (human & automated) used to perform the activities. A scenario can include one or several processes for a simulation run.

- Incoming Work (Arrivals). Each scenario must involve work to be processed. The description includes information about when the work arrives, as well as all appropriate attributes of the work (e.g. region, amount, size) that may have an impact on processing.
- Resources, Roles and Shifts: As work is routed to activities in a process, resources are required to perform the activity. Resources may be human resources; they may be pieces of equipment; or they might simply be application systems. Roles are often used to describe the function performed and the skill required. Specific resources are then described as performing defined roles. The availability of Resources and Roles can also be controlled by using Shift information.
- Activity Details: In order for the simulator to reflect the real-world processing of a business process, additional information is often appended to the scenario. How long an activity takes to process is not defined in a Process Definition, but is included in the scenario information (historical data collected by the analytics may be the source of this information). This and many other details can be expressed as a single integer or a complex expression involving the attributes of the work and/or the use of statistical distribution functions for randomness. Other information appended for simulation might include routing information which tells the simulator under what conditions certain routes are taken. Routing information can come from rules that exist in the process definition, but it is often amended with percentages that, based on historical activity, reflect the likelihood of a certain outcome or path.

All scenario information is typically represented in an XML document.

A simulation run generates a log event stream – new data for the analytics engine. This can then be viewed or extracted for various purposes.

THE OPTIMIZER

The *Workflow Optimizer* contains a collection of simulation based methods to help in

- Improving throughput by reducing
- activity wait (idle) times
- activity perform times
- Lowering Costs by examining
- the effects of balancing resource utilization by cross training
- the effects of reducing staff or introducing part-time shifts
- Satisfying SLA criteria by
- assigning/changing priorities of certain work item classes

The *Optimizer's* functionality comprises three main areas:

- Activity wait time reduction through resource load balancing/cross training.
- Idle cost reduction through adjusting resource quantities and shift assignments; see section.
- Critical path/activity analysis: locates paths and activities that are 'critical' w.r.t. wait times, perform times and/or costs.

The *Optimizer* needs data to decide upon an action. The data come from two sources: the scenario document and the data base. An *Optimizer* action consists of making changes to one or more parts of the scenario document.

For example, wait time reduction by load balancing changes role assignments, and idle cost reduction changes shift assignments; both affect the resources section. Altering activity durations affects the activities section and setting work item priorities affects both the resources and the arrivals sections.

After each *Optimizer* action the altered scenario is simulated; the generated log event stream is analyzed and the results published to a set of database tables. From those tables the *Optimizer*'s analysis part extracts the data structures that feed all *Optimizer* routines. There are mainly four kinds of data.

- Step data: accumulated wait and perform times (gross/net) and costs for each step per each significant work item attribute (workflow relevant data field of the work item).
- Execution path data: identifies the process paths for individual work items and accumulates step wait times, perform times and costs along those paths.
- Productivity data: utilization (busy, idle) for each resource accumulated during the core analysis period set by the *Optimizer* user.
- Timed Sequence (TS) data: durations, gross and net, for each Timed Sequence per work item.

The data may be averaged over several simulations to balance random effects. The following table shows which data the different *Optimizer* functions employ.

Table 1: Data Employment

| Optimizer Function | Simulation Data Employed |
|--|---|
| Activity wait time reduction through resource load balancing | Accumulated wait times; (Critical) path analysis; Resource utilization (accumulated busy/idle times) |
| Adjusting of activity perform times | (Critical) path analysis |
| Idle cost reduction by adjusting resource quantities and shift assignments | Productivity (accumulated busy/idle time) |
| Monitoring | Timed Sequence durations |

There is a special *Auto Optimizer* mode of operation that allows running the *Optimizer* on a given workflow scenario without user interaction. All the user has to do is to specify a 'strategy', a set of preferences and parameters, and then push the start button.

The *Auto Optimizer* puts the various components of the *Optimizer* into one, organized whole – a conceptual map that allows the user to get around in the maze of possibilities and approaches and to develop strategies for reaching specific goals.

In the sequel we use the *Auto Optimizer* as a vehicle for discussing the *Optimizer* functionality.

THE AUTO OPTIMIZER

The *Auto Optimizer* comprises most of the *Optimizer* features. There are two main groups or general goals that may well be in conflict with each other.

- Reducing activity wait times caused by workforce shortage (improving performance).
- Reducing unnecessary costs caused by workforce abundance (improving productivity).

Wait Time Reduction (Bottleneck Analysis / Critical Path Analysis)

Wait time reduction [2] is a core function of the *Optimizer*. Wait times at activities are caused by the lack of resources needed to perform an activity at the time a work item arrives at the activity. The *Optimizer* ranks the activities according to their accumulated wait times. The top one is considered the current *bottleneck* and its performer (*bottleneck role*) indicates the skill or resources of which there seems to be a shortage (*bottleneck resources*).

The search for bottlenecks may take place among all activities of the workflow or it may be restricted to a certain subset of activities: the current *critical path* or a timed sequence selected.

The *Optimizer* knows three ways of dealing with bottlenecks.

Role Removal (Specialization)

Bottleneck resources (i.e. resources that play the bottleneck role) may be set free to focus on this type of work by removing other roles of their role assignment as long as those roles can be played by other, less busy resources.

Role Addition

A bottleneck resource may be *complemented* by other, less busy resources that play roles ‘similar’ to the bottleneck role and may take over the bottleneck role, e.g. after some additional training of staff people. There are currently two ways to find such similar roles and their resources:

- The *Scenario Document* may list explicitly possible complements for the bottleneck role. For example, we say – for mere demonstration purposes – that

```
<RoleComplements>
  <Complement Name="Registrar" CanDo="Discharge" />
  <Complement Name="Lab" CanDo="XRay" />
</RoleComplements>
```

- There may be a systematic *similarity* between roles based on the syntax of their names.

For example, *NorthCal:Appraisal* and *SouthCal:Appraisal* are two roles of the *Bank Loan* workflow. They are similar in as far as the required skill – creating an *Appraisal Report* – is the same; the difference is only with respect to the region – *NorthCal* respectively *SouthCal* – from where the loan application originates.

Resource Addition

If load balancing between resources (role removal and role addition) can no longer help, there may still be ‘critical’ activities where the resources that perform those activities are unusually busy (‘maxed-out’). Then it may be in order to add resources, i.e. hire additional workforce or buy more equipment.

Excess Idleness Reduction

While the first part of the *Auto Optimizer* tries to deal with the shortage of certain types of resources, the other part deals with resource abundance. There are in essence two ways of reducing excess resource idleness:

- adjusting resource quantities and shift assignments;
- increasing the arriving workload.

Availability Reduction

In the first phase the *Auto Optimizer* tries to make the best use of the available workforce in order to improve the performance of the workflow system. Except for adding new resources if absolutely necessary, the actions taken have no effects on costs. The second phase, however, explicitly addresses cost issues. It is concerned with reducing costs for excess resources.

The *Auto Optimizer* has a rather sophisticated algorithm for removing excess resource availability[3]. The – degree of – idleness of a resource is measured in terms of the time it is available for work and the time it is actually busy. The algorithm tries to reduce resource idleness (idle costs). It looks at resources in decreasing order of their *idle costs*. Depending on the quantity and the idleness (time) of the resource at hand, it tries one of the following reduction measures.

- Set resource quantity to the quantity really needed.
- Try *removal* either temporarily by giving it *Leave of Absence* or permanently, if it was added by the *Optimizer* itself in an earlier stage.
- Try *part-time work*, possibly combined with the idleness of some other resource that can perform the same work.

There are several details and particulars to the algorithm for reducing excess resource availability; for example: whether part-time work makes sense (is enabled), in what quantity a resource occurs, whether a resource is *replaceable*, whether a resource is on part-time already.

After each action the altered scenario is simulated again. If no action is performed the next resource (ranked by *idle costs*) is tried until the list is exhausted.

Workload Increase

Excess resource idleness may indicate the capacity of the workflow system for accomplishing more work. This may be tested by increasing the arriving workload.

Currently there is only one way of increasing the workload: the arrivals of one day are added and spread evenly over the period of arrivals.

Productivity Index

A simple productivity index and a fuzzy scale allow watching the progress of the *Auto Optimizer*. The index is the – possibly weighted – number of work items completed during the core period, divided by the sum of resource costs and fixed daily costs. The unit cost is defined in the *Scenario document*; default: \$1000.

Auto Optimizer Strategies

Various parameters and preferences allow the *Auto Optimizer* user to define a specific strategy. Between sessions the settings are kept in the *Scenario document*. On the *Auto Optimizer* panel the fields are grouped in the following way:

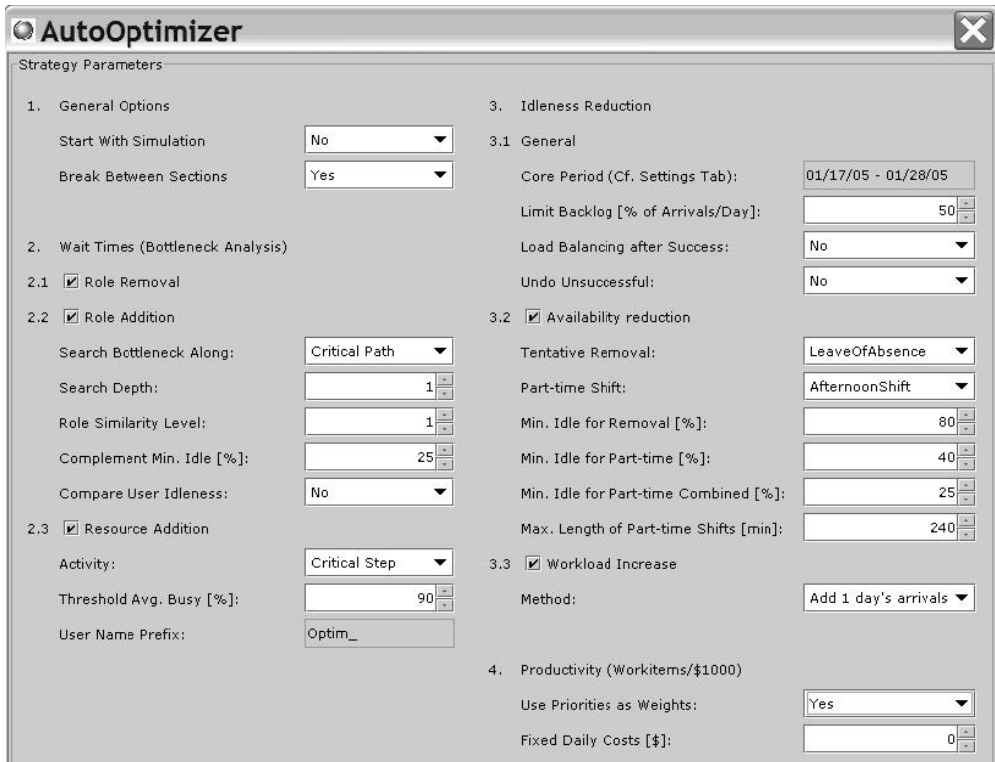


Figure 3: Auto Optimizer Settings

General Options

Between major sections the user can suspend the run and resume it later, after reviewing the results. If the current scenario has not been simulated/published after the last change the Auto Optimizer may start with simulation.

Wait Time Reduction

There are a number of options and thresholds that influence bottleneck analysis and critical path analysis and the subsequent actions.

A bottleneck can be searched within ‘All Steps’, a ‘Critical Path’ or a ‘Timed Sequence’. If there is no complement for the current bottleneck role, role addition may stop or it may go on in the list of activities ranked by accumulated wait times.

The syntax of role names may distinguish several levels of similarity. As an example

- Level 2: identity (no difference must be ignored)
- Level 1: skill must be the same but the regions may differ (regional cross training)
- Level 0: ignore all differences, both in skill and in region

A resource that receives an additional role must show some amount of idleness.

If specializing and cross training don't help a new resource may be added to perform the most critical activity.

Idleness Reduction

After wait time reduction the remaining idleness can be reduced by either reducing the availability of resources or increasing the workload. Either method is ap-

plied only as long as the backlog of unfinished work objects is not greater than the given limit (in terms of the average number of work objects arriving each day). After each step the *Auto Optimizer* may return to the load balancing phase of the wait time reduction phase.

Reducing availability can be done by removing resources or assigning part-time shifts. Currently there is only one way of increasing the workload: the arrivals of one day are added and spread evenly over the period of arrivals.

Productivity Index

If priorities are defined for the work items, they may be used as weights in the number of work items completed. Fixed daily costs may be added for each work item.

CRITICAL PATH ANALYSIS

For Wait Time Reduction the Optimizer searches for ‘critical’ roles – skills required to perform ‘critical’ activities, i.e. activities that show large accrued wait times and hence indicate a shortage of corresponding resources. Here the term ‘critical’ is used in a rather loose, informal sense. In the realm of process and project management, however, there is also quite a powerful, formal meaning associated with the term.

Critical Paths

In project methodology, the critical path is the sequence of work in all projects which, if identified and managed, always leads to success. Still, many seek the magical critical paths of their projects in hopes of making management simple.

The word critical seems to cause this misconception about the critical path. It implies pivotal importance, something crucial or indispensable.

In Critical Path Method (CPM), the critical path can be defined in two ways:

- The critical path in a project is the longest path relative to the time available to complete it.
- The critical path in a project is the path with the least project time reserves.

(Cited from [5], p.113.)

The kind of *projects* addressed by *Project Management* techniques like PERT and CPM differ in two major aspects from the *workflows* that appear in a BPMS.

- The project usually defines a single process like building a submarine or a power plant while a workflow defines a process that occurs many times, in parallel and in sequence.
- The project usually knows only sequencing and AND split/joins for structuring the set of basic tasks (steps). That makes it easy to define a *path* as a single start to end sequence of steps.

A workflow, however, may have – in addition to AND split/join – XOR split/join and loops. Each single instance/occurrence of a workflow, i.e. the processing of a single work item, when looking at it from hindsight, has the same simple structure as a project. All choices have been resolved, all loops have been unfolded. Only the AND split/joins remain and create a non-sequential course of action.

On the other side, it is the type of *Split* that affects the number of occurrences of the different paths in a larger set of work items being processed.

Loops may increase the number execution paths indefinitely and – if they contain AND clauses – quite rapidly. The path analysis component added to the analytics

avoids such combinatorial explosion by overlaying occurrences of the same cycle. The details are outside the scope of this paper (see [4]).

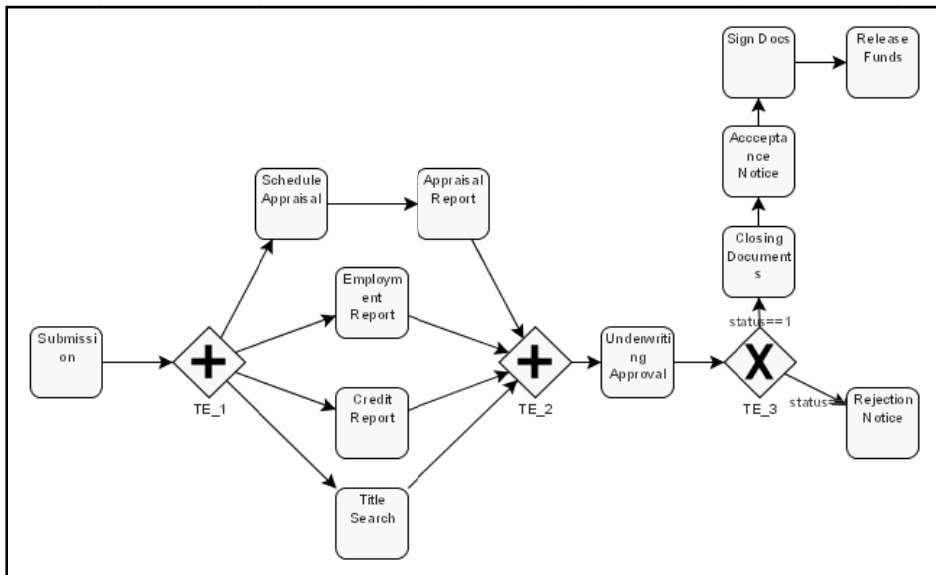
The first definition above of being *critical* can be easily applied to the paths occurring in the processing of a single workflow object. For the purpose of analyzing/optimizing workflow processing, we generalize it to include different measures.

- The *X-critical* path in processing a work item is the path that is maximal with respect to measure *X* where
- *X* is the *wait time* at activities due to the resources not being available (*not* due to the synchronization at AND joins), accumulated along all activity occurrences of the paths;
- *X* is the accumulated *perform time* at activities;
- *X* is the accumulated sum of wait and perform times;
- *X* is the accumulated *perform costs* (resource costs) of activities.

Note that the third case captures the usual notion of *critical* in CPM.

(CRITICAL) PATHS OF WORKFLOWS

Based on data collected during simulation/analysis (by a special path analysis component, (see [4]), the *Optimizer* works with the following kind of tables. The screenshots are taken from the prototype implementation of our ideas. The example is based on the following simple business process.



The workflow has both parallel (AND split/join) and alternative branches (XOR split). A loan application is sent along four parallel branches for preparing the reports needed for the underwriting decision. Then the applicant is informed accordingly.

The applications are processed in four regional centers, *Washington, Oregon, NorthCal* and *SouthCal*. This organization is reflected by the roles that the staff members may play. There are several discrepancies between the available staff and the needs induced by the actual (simulated) workload; they allow the *Optimizer* to show its power.

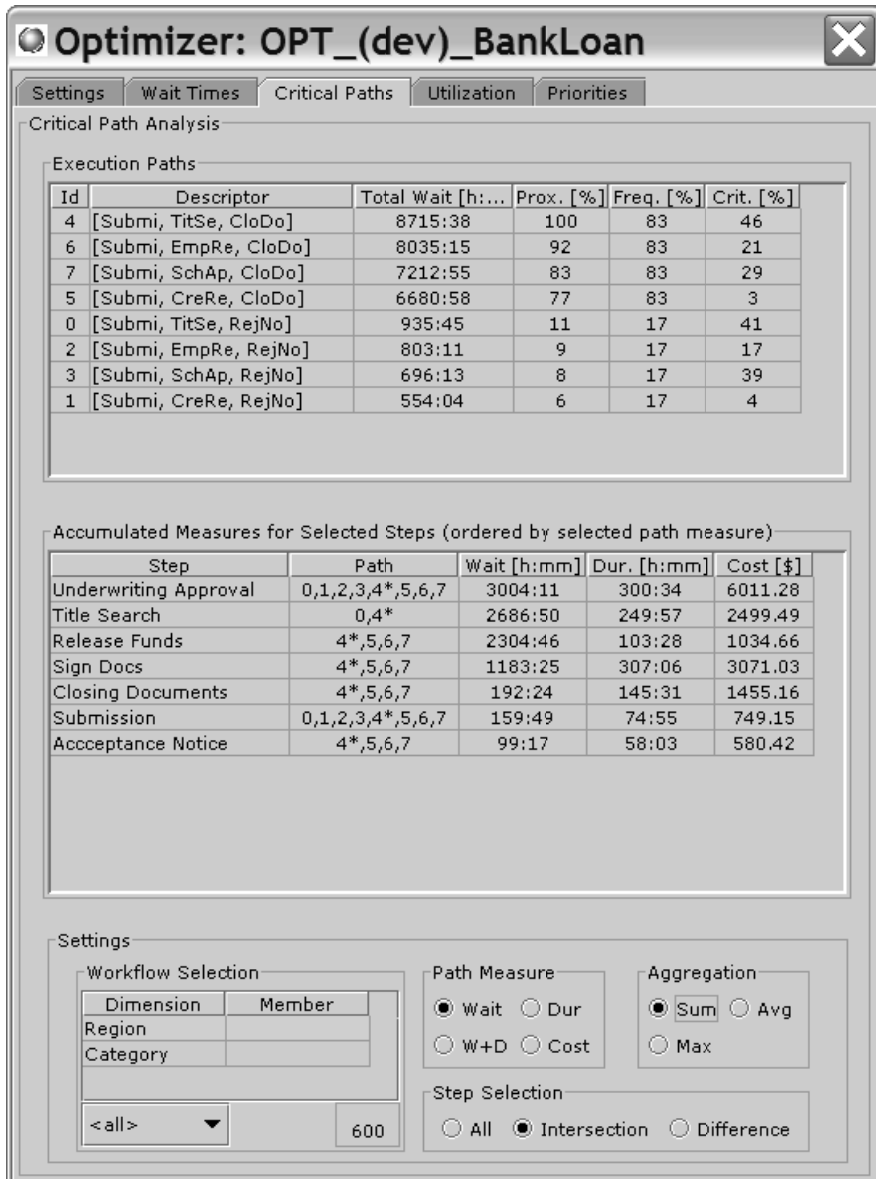
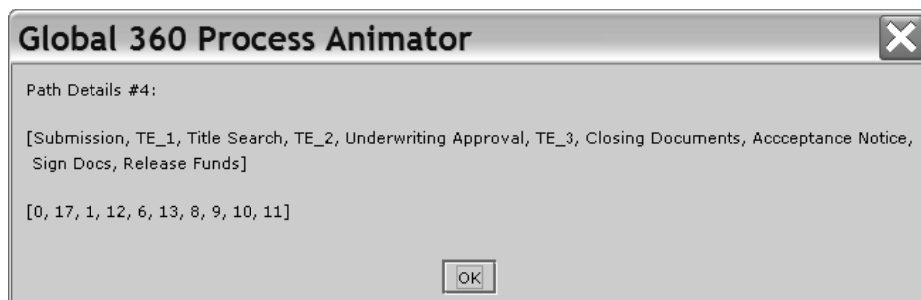


Figure 4: Critical Path Analysis

Execution Paths

The Execution Paths table above lists all paths that occur in one of the processes for the simulated set of work items. Each path is identified by the – abbreviated – names of the activities that mark the outcome of a split (initial activity, first activity after each split). So path #4 starts at *Submission* (as all paths do), at the first (AND) split *TE_1* it continues with *Title Search* and after the next and final (XOR) split *TE_3* it continues with *Closing Documents*.

A double-click on the path row brings up more detailed information about the path #4.



The paths are ordered according to the selected measure *total wait time* (see settings on critical path panel Figure 4, lower right corner). Path #4 is closely followed by path #6 whose 'proximity' is 92%, i.e. whose measure is 92% of the top path. Both paths occur in 83% of **all** selected processes/work items (see settings on critical path panel Figure 4, lower left corner). Path #4 is the critical one in 46% of all work items, #6 in 21%.

Steps and Basic Measures

The second table depicts the steps belonging to the selected path (#4). It shows the list of paths to which they belong and all their basic measures (*wait time*, *perform time* and *perform costs*) in the selected mode of aggregation (*sum* in this case). The steps are ranked according to the measure selected for the path table.

Currently the most critical step is *Underwriting Approval*; it has the largest total wait time and it occurs in all paths (in all processes). It is followed by *Title Search* which is on one of the parallel branches and hence occurs in two paths only (acceptance and rejection). Hence it is no surprise that path #4 is *the most critical* one.

Why Steps Are Wait Time Critical: Available Resources

By double clicking on an activity row of the step table one gets a look into the resource situation of the activity. For the most critical step, *Underwriting Approval*, we get:

| CPA Selected Step (Total Measures) | | | | | |
|------------------------------------|------------------|-------------|-------------|-----------|--|
| Step | Path | Wait [h:mm] | Dur. [h:mm] | Cost [\$] | |
| Underwriting Approval | 0,1,2,3,4*,5,6,7 | 3004:11 | 300:34 | 6011.28 | |

| Roles and Users (Average Measures) | | | | | |
|------------------------------------|-------------|-------|-----------------|----------|-------------|
| Role | Wait [h:mm] | User | Avail. [h:mm/d] | Busy [%] | Cost [\$/h] |
| *NorthCal:Underwriting | 12:25 | Gale | 8:00 | 99 | 20 |
| SouthCal:Underwriting | 0:06 | Pat | 8:00 | 83 | 20 |
| | | Roger | 8:00 | 64 | 20 |
| Oregon:Underwriting | 0:06 | Penny | 8:00 | 74 | 20 |
| Washington:Underwriting | 0:00 | Yvon | 8:00 | 61 | 20 |
| | | Dick | 8:00 | 46 | 20 |

The lower part lists all performers (roles) that are required by some occurrence of *Underwriting Approval*. The * in front of the top most *NorthCal:Underwriting* indicates that this is also the current bottleneck role; it has an average wait time of 12.5 hours. The only resource to play that role is *Gale* who works 8 hours per day and is busy 99% of its time.

The situation shown is when no load balancing has happened yet. The table above suggests that *Dick* who is the least busy resource doing *Underwriting Ap-*

proval– in region *Washington* – is trained to do *Underwriting Approval* also in region *NorthCal*.

If all possible load balancing had been depleted already, a new resource *Optim_1_(Gale)* – a ‘clone’ of *Gale* – would be hired in the resource addition phase of the *Auto Optimizer*, as the average percentage of being busy of all resources that can do the most critical step is above the selected threshold of 90% (cf. optimizer settings, Figure 3).

When Steps Are Perform Time Critical

Currently the *Auto Optimizer* offers no way of automatically reducing the perform time of a critical activity. However, a simple right-click on the step row in the CPA step table opens a dialog for editing the activity’s properties, e.g. the perform time or a participant reference to specially trained performer. And in the table showing the activity’s resource situation, a right-click on a resource opens a dialog for editing its properties, e.g. its proficiency.

Another measure may be re-designing the workflow such that the critical activity is divided into two parts performed in parallel. Though conceivable in principle, no effort has been made to program this kind of change of a workflow definition.

SUMMARY, OUTLOOK

The *Auto Optimizer* puts the *Optimizer* components into one, organized whole – a conceptual map. It orients the user in the maze of possibilities and approaches for optimization and helps developing strategies for reaching specific goals.

There are a few unexplored possibilities worth mentioning.

Activity wait time reduction has come up with two competing notions: *bottleneck* and *critical*. It seems that all may be based on critical paths and activities alone. Bottleneck search can already be constrained to the respective wait-time critical path. In many cases the most critical role (i.e. the top role in the critical activities role/resource table) is also the bottleneck role. And if not, the bottleneck may not be that critical after all, i.e. on a parallel branch with a lot of slack (time reserve).

Currently the perform time of critical activities must be adjusted manually. This seems unsatisfactory, an unfinished business, and more research is needed. One option, for example, is to split a perform time critical activity into two parallel ones to allow for more work to be done in parallel.

The *Optimizer* uses statistics about timed sequence durations to monitor the effects of its actions. Watching/monitoring throughput in terms of TS durations may seem rather ‘expensive’ (complex); workload/backlog and cycle times, however, are not fine enough a measure. The TS statistics could serve as an alternative way of controlling the *Auto Optimizer*:

- High ‘goal satisfaction’ stops load balancing;
- Low ‘goal satisfaction’ stops idleness reduction.

Another unfinished yet quite promising business is the integration of some commercial workforce management system. With unlimited availability of resources/qualifications, a simulation run can create an ideal pattern of performer requirements. This pattern can be fed into a workforce management system to provide an optimal schedule for a given workforce. See [6] for an example of this.

REFERENCES

- [1] *Automated optimization templates*. Discussion paper. Global 360, Sept 2004.
- [2] *Reducing wait time*. Technical Report. Global 360, Aug 2005.

- [3] *The Workflow Process Optimizer*. Technical Report. Global 360, March 2006.
- [4] *Critical paths of workflows*. Technical Report. Global 360, Dec 2006.
- [5] Dennis H. Busch: *The new critical path method*. Chicago, Illinois. 1991
- [6] Robert Shapiro: Integration of Workforce Management with a Business Process Management Suite. 2008 BPM and Workflow Handbook, published by Future Strategies Inc.