IRS-MT: Tool for Intelligent Resource Allocation.

Piotr Chrząstowski-Wachtel[1,2] and Jakub Rauch[1]

[1] Institute of Informatics, Warsaw University,
Banacha 2, PL 02-097 Warszawa, Poland
[2] Warsaw School of Social Sciences and Humanities,
Chodakowska 18/31, PL 03-815 Warszawa, Poland
pch@mimuw.edu.pl, jakub.rauch@gmail.com

Abstract. A tool for optimizing cost and time of a workflow execution with respect to allocation of multi-purpose resources is presented. The optimization is done as a result of simulations, which take into account the cost and time associated with each of the resources, when allocated to transitions in Petri nets representing a workflow. These attributes can be collected and updated based on the logs of the finished instances. The program suggests the best allocation procedures, giving the estimates of the performance of the whole run for all possible decisions.

1 Introduction

Modeling processes as workflows has become quite popular and proved its usefulness in practice. One of the problems associated with running a workflow is the resource management. By resources we mean all components required to run an activity. In our case, their necessity is described by certain requirements (e.g. skills or functions) associated with activities (transitions). With each resource we associate a bunch of skills, so we can choose, which of the resources are to be attached to certain activity at the workflow simulation run-time. The Petri net approach requires collecting all the resources necessary for a given transition before triggering an acitivity to run. One cannot reserve a resource, and keep it busy, while waiting for other resources necessary to run an activity. Only when all resources are available we can make a decision to run (fire) an activity.

It often happens that a resource is requested by different activities. A resource conflict occurs, when a single resource is shared by two enabled transitions. We must make a decision, which activity will use the resource first. We assume here that the resources are re-usable, and that after finshing a transition the resource can be used by another activity.

When we make a decision about the resource allocation, we should take into account several aspects. Usually we try to optimize some quality function, like time or cost of the workflow run. We assume here that during a workflow run we can perform several actions concurrently. Since some of them will be competing for resources, a proper allocation can improve the quality of workflow

Recent Advances in Petri Nets and Concurrency, S. Donatelli, J. Kleijn, R.J. Machado, J.M. Fernandes (eds.), CEUR Workshop Proceedings, ISSN 1613-0073, Jan/2012, pp. 235–250.

run. Engaging proper resources can diminish for instance the delays caused by lack of the only resource requested by a concurrent action and hence waiting for this resource to be released.

As described in [BPS09], there are many essential aspects of resources, which should be taken under consideration during workflow simulation. One of them is already mentioned: the multifunctionality of resources. Resources have attributes describing their skills. In other words these are the abilities to perform certain actions. Each resource is associated with the set of activities, in which it can be used. Other attributes taken under consideration are performance and cost associated with engaging the resource. So we know how fast a resource can do an activity and how much it costs to use it. By expense of a resource we mean here a value per time unit associated with involvement of the resource. A chief accountant probably has a driving license, but using him to drive the documents somewhere can be a waste of his time and precious skills. His cost per hour is much higher than that of a professional driver.

Since workflows can be quite complex, the authors do not see any analitycal approach, which would be effective in the optimization of runs. It is hard to predict all the possible outcomes of the allocations decisions, when resources are in many conflicts. Instead, we propose an experimental approach, which involves the simulation of many runs, taking into account different allocations and estimating the desired measures as a result of allocation decisions. In our prototype IRS-MT (Intelligent Resource Sharing-Modelling Tool) the manager can edit a structured workflow net and set the number of experiments. The program makes random allocations reporting times and costs of the runs. Based on this knowledge the manager can use the suggestions of the program and get a picture of possible outcomes of the decisions taken. The decisions can be made incrementally. After each decision the workflow run advances its state, and when we come to the next decision, a separate simulation will be made, basing on the actual state of the system.

2 Basic definitions

In our tool we consider resources (S), roles (R), requirements ($\mathbb{Q}$) and activities (A). Resources and roles are finite and defined a priori (by designer). For each resource we define a set of skills (roles), which it can use. This is denoted by $F_{SR} : S \to P(R)$ function. Additionally, for each role in a resource, its efficiency may differ. For every resource $s \in S$, we define the *efficiency* function $E_s : F_{SR}(s) \to \mathbb{R}_+$. This function describes how fast a resource performs each of its roles. The lower this value is, the higher is the efficiency (one is the base value). Additionaly, for every resource we define its use cost per time unit as a function $C : S \to \mathbb{N}$. On the other hand, for each activity in a workflow, a set of requirements needed to fire it, should be determined. Every requirement $Q \in \mathbb{Q}$ is a subset of roles. We define a function $F_{AQ}(a)$ which is a bag of requirements associated with the activity. It is important, that every activity has an expected duration time defined and its standard deviation value. Later, in the generation

and simulation phase, only the available resources fulfilling these requirements will be considered for taking part in such activity. And so, $s \in S$ is *fulfilling* $q = \langle R_q \rangle$, iff $R_q \subseteq F_{SR}(s)$. It is important, that if a resource is involved in some activity, it cannot be used by any other activity (there is one exception, which will be covered later in this section).

Initially all the resources are free, available in a pool of idle resources. We assume, that all the resources are reusable, so at the beginning of activity execution the needed resources are collected and at the end all the freed resources will be returned to the pool of idle resources and will be available for further use. The graphical representation of relations between the introduced notions is depicted on Fig 1.

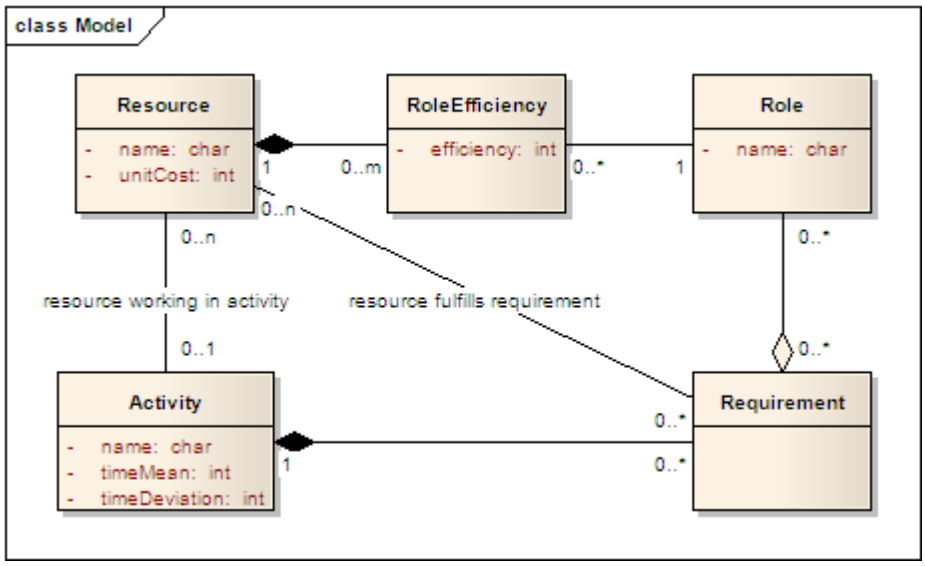

Fig. 1. Model

Our workflows are Petri nets created using five basic refinement patterns, proposed in [PCh03]: sequential-split of a place or transition, parallel-split of a place, choice-split of a transition and loop-split which attaches the spawned transition with a self-loop to a place. The additional rule for attaching a resource place is dual to the loop-split. It just glues a freshly created (resource) place to a transition by a self-loop. From the Petri net perspective we assume here that such place will contain initially a token for each physical resource available. We call the places created by such splits *resource places* or *activity places*. One cannot refine resource places. Even if the transition is refined, the resource will be allocated at the beginning of its execution and released, when it is done. Moreover, every resource place will carry the set of requirements defined for corresponding

activity. If we match the resource places with activities accordingly, we will see, that the requirement assignments are defining the F_{AQ} function.

When we use activity refinement on a transition, which is inside some other activity, we will create a *nested activity*. By its *ancestor activity* we call every other activity, in which this one is nested. Such nested activities do not differ from any other activities, except for the fact, that these can use resources from its ancestors, as long, as the resource will not have one of its roles assigned to requirements in two different activities.

3 Tool overview

The main goal of this work is to provide a a tool, which can be helpful in improving the resource management. We concentrate on optimization of the workflow execution by providing the user with relevant statistical information and letting him make decisions on resource-to-activity assignments. To achieve this, a Petri net defining a workflow with activities and resources, will be created. For this purpose we introduce a *Workflow Designer*. It is the editor for building workflows using refinement patterns. In the editor we create activities, declare the set of requirements and approximate execution duration. On the other hand, we have *Resources Editor*. We use it for defining a pool of resources, assigning roles to them, and determining their effectiveness in each role. The set of defined roles can be modified by an always-visible editor *Roles Viewer*. All these three editors form the *static part* of the tool. Its more detailed description is presented in section 3.1.

After defining the model in the *static part*, we can proceed to the *dynamic part* of the tool. We will use *Generator*, to create sufficent number of random runs. When this is done, the *Simulator*, the *Report Viewer* and the *Bucket Editor* shows up. The first one displays a copy of our workflow, where resource places contain both the requirements and lists of currently available resources, fulfilling given requirements. Here, the tool provides us with information about expected time and cost of workflow completion for each of resource-to-requirement assignment we see. Basing on this knowledge we can decide, which resource should be assigned for current requirement. Every choice causes the expected values to be recalculated, so that we can run the workflow deciding, how resources should be used in the activities. The Report View presents detailed information about expected time and cost of workflow completion. It is synchronized with current simulator state, so all the values are always up-to-date. The Bucket Editor is a tool for storing, and presenting details of the runs, which were manually performed by user in the Simulator.

For example, let us consider the following, simple case. There is a package, which must be delivered to the destination within an hour. We know, that standard travel time by a scooter in current traffic would take around fifty minutes. We have two available scooter drivers: Evan and Gregory. Both of them can do the delivery, but Gregory has got his license for much longer time than Evan, and he shortens the expected delivery time by around 20 percent. Evan, on the

other hand, is still afraid of driving fast and using tricky shortcuts, so his deliveries often take 20 percent longer than normal. However, Gregory's earnings are twice the earnings of Evan. This is the classic case, where no optimal resolution to the problem exists. It must be up to user's decision, whether time or cost is more important, and it is up to our tool to provide the user with information about expected cost and time consequences of each decision. We achieve it by simulating many thousands of runs and preparing the estimates with all aspects of the workflow taken into consideration, i.e.: activities order and dependencies, resources usage conflicts, possible time/cost variations.

3.1 Static Part

To present the tool in more details, we introduce other, more complex example. Let us suppose, that we have two rooms: A and B. We need to plaster and paint both of them. Additionaly, room A needs to be decorated. Here we assume, that a room cannot be painted, unless it is plastered and it cannot be decorated, unless it is painted. Expected time units, required to complete these tasks for each room are following:

- Room A — plastering: 20, painting: 10, decorating: 20;
- Room B — plastering: 10, painting: 20.

This process is presented on Fig. 2. We also need two resources, applicable for the work: *Steven* and *Tom*. *Steven* is an experienced *painter*, with a skill of *plastering*. *Tom*, on the other hand, is a *decorator*, who also can *paint*, but it takes him some more time that it does for *Steven*.

The purpose of the static part of the tool is to model this situation, so that it can be then simulated and later analysed in the dynamic part. We will now explain, how it can be achieved using available functionalities.

Resources Editor Defining resources consists of identifying the available set of people, machines and tools. For each of them, we can define a set of attributes like: use cost per time unit, collection of applicable roles (skills) and the effectiveness in each role. Every resource may have many skills, and many resources can have the same role. As it turns out, we often miss such knowledge during resources allocation planning and we do not take all the benefits from what a resource is capable of doing. Because of that, we can also miss the optimal resolution for given situation. Therefore we need to integrate all these aspects in the analysed context, so that we can consider consequences of particular situations with respect to most important factors. It is worth noticing, that in most popular Human Workflow management tools like Tibco [BS07] or Corel iGraphix, as well as in some academical tools like Yasper [YA06], during the workflow design and simulation phase, the roles are treated as resources. No resource, can have two skills. This, for modelling purposes, is a major limitation. It means, that one resource will never be requested for two activities with different required roles, which tightly limits analysed possibilities. In our tool there are no such boundaries.

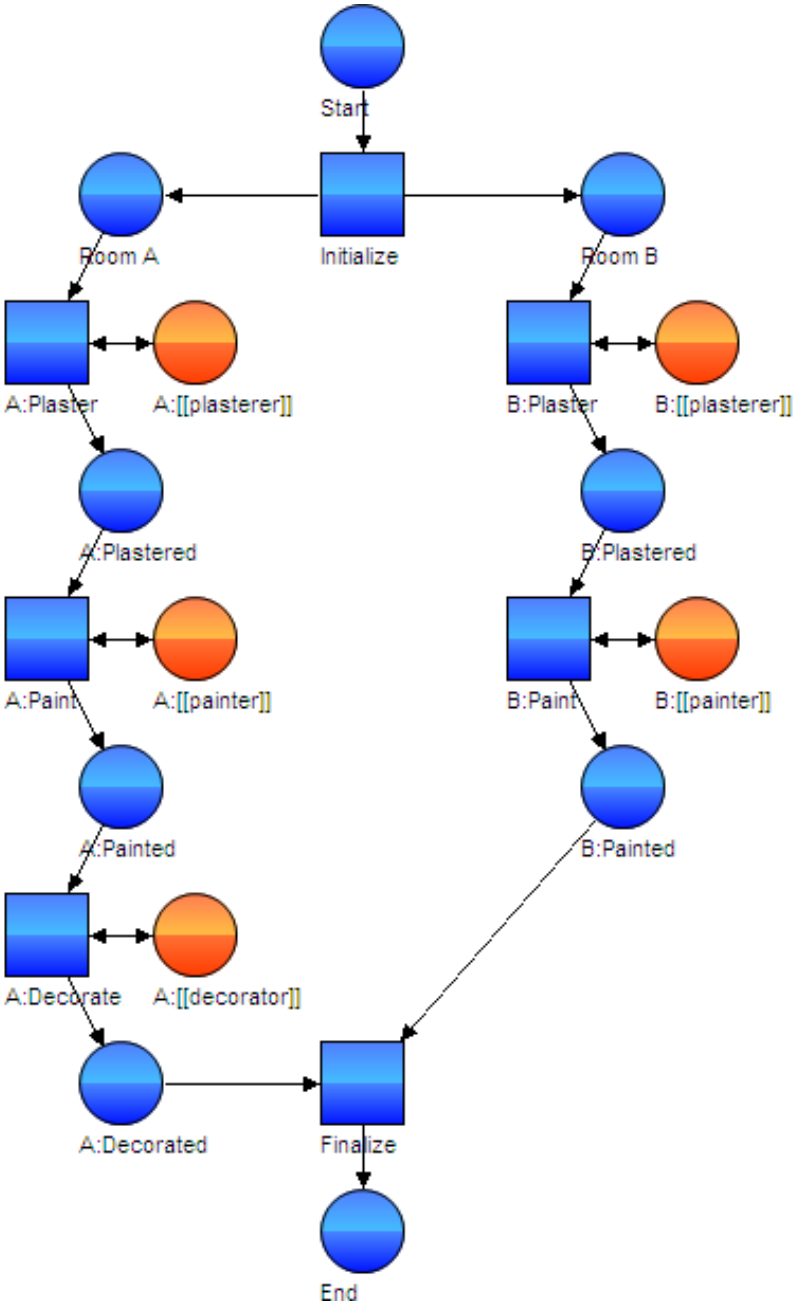

Fig. 2. Sample net

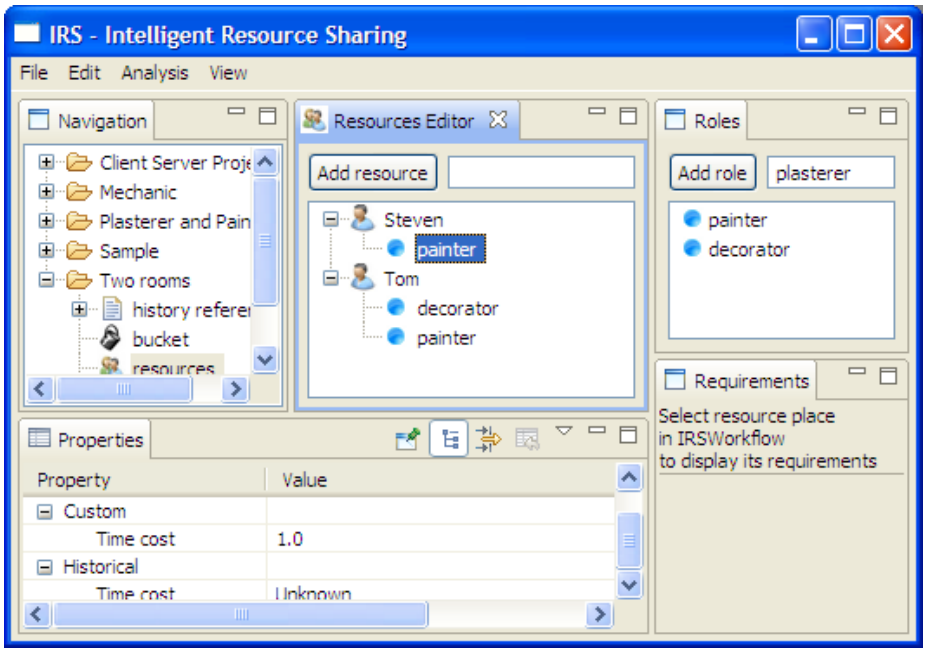

Fig. 3. Resources edition

Fig. 3 presents the pair of our resources: *Steven* and *Tom*. The interface is straightforward and it uses drag and drop features across almost every view. At the depicted state of modeling Steven lacks the plastering skill yet. In order to add it, we type *plasterer* in *Roles View*, press *Add role* button, drag the newly created role from this view and drop it over the *Steven* entry. New skill would get all of the necessary attributes initialized to default values. At the bottom of the window, we can see the *Properties* tab, which, as a context panel, allows us to modify attributes of currently selected object. The information about Steven's efficiency as a *painter* (*Time Cost* row) is displayed here. To be consistent with the descritpion, we should change this value from 1.0 to 0.8. This indicates that activity performed by Steven in a role of *painter* could take 20 percent shorter than normal. We repeat similar scenario for Tom, associating with him the role of painter and decorator and setting the efficiency coefficients for each of these roles. Expenses for each person should be defined also here.

Workflow Designer In the presented tool we can model sound workflows using refinement patterns presented in [PCh03]. To make this process easier it is possible to apply these patterns to any node the refinement tree, including the inner ones. Each node can also have its subtree truncated. The structural approach has been chosen to simplify both the design process and the inner application processing engine. In this tool we introduced the basic set of six refinements.

These might be extended in future by other patterns like communication or synchronization patterns described in [PCh03].

Editor offers additional operations for collapsing and expanding nodes, according to the information held in the refinement tree, so we can view our net at desired level of detail. The application displays all the nodes automatically in the viewer, but we can also move them around manually. The process from the example, has been created using refinement patterns, as shown on Fig. 4. Currently the *B:[[plasterer]]* resource place is selected, so that we can see, which requirements are defined for this requirement place in the bottom right tab *Requirements* (in our case all resource places will be labelled ⟨room name⟩:[[⟨required role⟩]]). As it was mentioned earlier, resource places are created by the *ACTIVITY* pattern, and cannot be further refined. Each resource place is associated with exactly one activity, so it is the right place to hold all the needed requirements for activity.

The whole net has been built using only the *SEQUENCE*, *CONCURRENCY* and *ACTIVITY* patterns. On the Fig. 5, all possible refinement operations are presented both for each standard place (not resource place) and each transition.

Every transition has got a special property, a measure, which describes its *desire to be executed*. We can modify this value to indicate transitions, which should have higher/lower probability of being executed, when in conflict with some other ones. This value is used only when at least two transitions are in conflict. By default this value is set to 1, but if we wish to make some transition to be executed more often, this value should be set accordingly. For instance if we have two active transitions, one with this value set to 3, and the second one to 1, then the first of them will be fired with 75% chance. This way we can declare, which actions are more probable or what kind of situations occur more often.

3.2 Dynamic Part

Statistics Generation Generator is a tool for preparing a list of complete runs with respect to guidelines defined in a workflow project and resources set. For this purpose, a special, extended copy of the designed workflow is created. Apart from copies of all the elements created by the designer, there are additional resource places for automatic resource availability management. It is guaranteed, that before each run, the whole net will be reset. Allocation decisions taken between two different runs are then mutually independent. It can happen that two identical runs could be generated by chance. Each run is executed and recorded using the following algorithm.

Initially, the *in* place of a workflow, is marked by a token, as well as there are tokens in the resource places. In a loop, a complete set of active transitions (activities) is constructed. If the set is empty, and the workflow is finished (a token is present on *out* place of the workflow), then the run is stored, the token from *out* place is moved to back *in* place. If there are at least two enabled transitions, one of them is randomly chosen (according to its *desire to be executed*). The selected enabled transition is fired and the loop is repeated. Note, that some

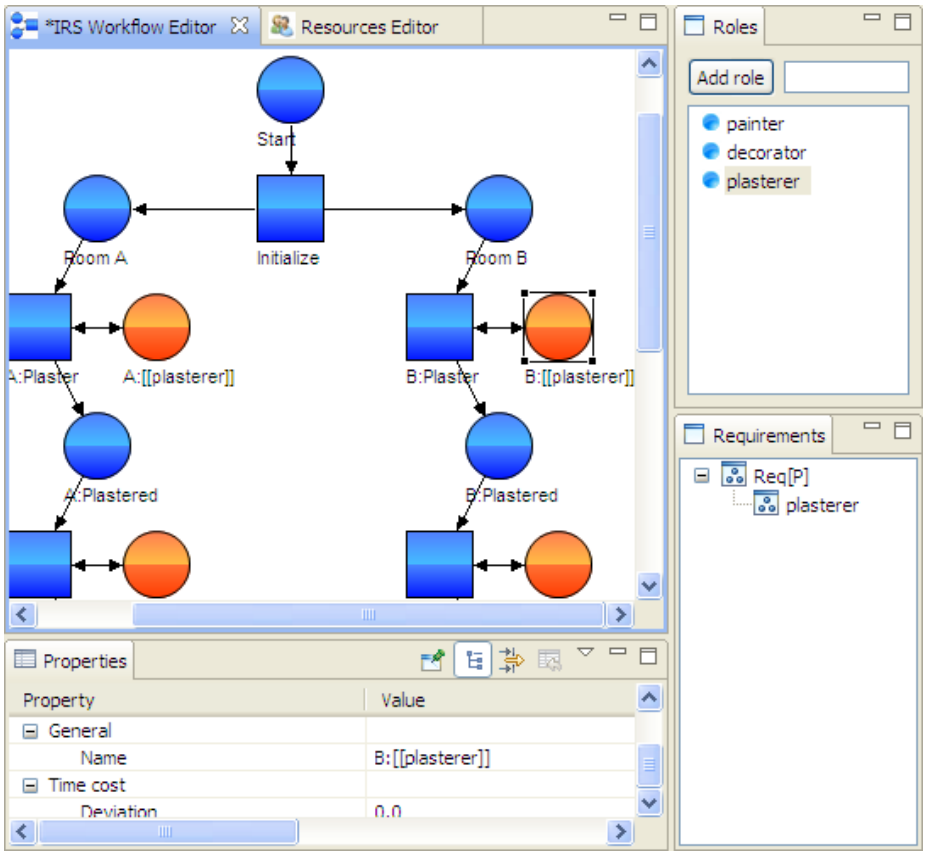

Fig. 4. Designer

of these transitions will indicate the begin or end of some activity. In such case an additional resource acquistion or return will be performed.

When the generator attempts to start an activity, all the requirements are being covered by skills owned by resources assigned to it. When we start one activity, some other can also start or end, so the tool can model various concurrent situations. When an activity ends, assigned resources are freed, the generator updates the workflow timer, the amount of time and cost counters for later analysis. A series of runs allows us to consider usefulness of certain choices in the context of further possible events.

The working time of the generator is dependent mostly on the number of allocations and releases of resources, which corresponds to one run. The computation power is also very important. In our case the generation of 10000 (ten thousand) runs on a standard computer takes no more than a few seconds.

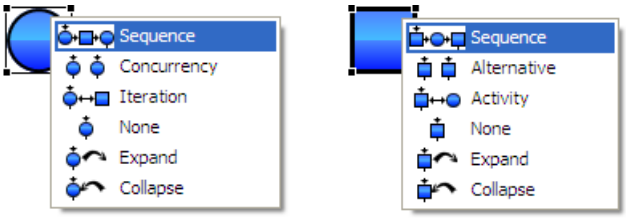

Fig. 5. Refinement patterns

Simulator Taking into account possible allocation decisions, we analyse time and cost of a workflow run. A few similar simulation tools have been reviewed in [BPM05]. On that basis some of the functionalities have been adapted to this tool, and a few flaws have been evaded. As a result, the application provides the simulator of the given net, and gives us a browser of performed runs.

In the report viewer (described later), we can browse all the runs created during generation. All these runs have been performed automatically, so using sorting capabilities of the report viever, we can easily find the optimal runs. Sometimes the differences in evaluation are very small, and the user may wish to examine the non-optimal allocation. For such purposes the simulator allows us to perform a step by step manual run of the designed net. An estimated time and cost of the run completion for every resource-to-requirement assignment at currently chosen state is presented. Thanks to this we know, which consequences are a result of our decision. The simulator remembers all the choices (resource assignments, order of activities start and end times) that a user makes during a simulation. It uses this knowledge to find all the statistical runs that are applicable to this situation, and then provides us with the estimates. Note, that in neither of Tibco, iGraphix nor Yasper, such manual interference in resource assignments is possible, because all resources in these simulators can have only one role. Moreover, up to the authors knowledge, there is no other tool, which supports simulation of resources with multiple roles, giving the user a chance to take part in a simulation at such informative level.

Let us suppose that a simulation has come up to a situation shown on the Fig. 6. The tokens presence on places *Room A* and *Room B* means that the *Initialization* phase has already ended. As we can see, enabled transitions have double-lined border. At the moment both *A:Plaster* and *B:Plaster* transitions require the same skills (in our case: *plasterer*), which have been defined in the corresponding places: *A:[[plasterer]]* and *B:[[plasterer]]*. Both places contain no tokens, which means, that no resources have been assigned to these activities yet. Currently selected place *A:[[plasterer]]* shows in the *Properties View* all the possible resource assignments for the *plasterer* skill. As it turns out, only *Steven* is suitable. Next to his name there is some information indicating the estimated cost and time of the run completion, with him taking part in this activity. In our example, *Steven's Cost* : $[1038 - 42.14]$ 970|1034|1050|1050|1084 *Time* : $[63 - 9.44]$ 50|56|70|70|72 means:

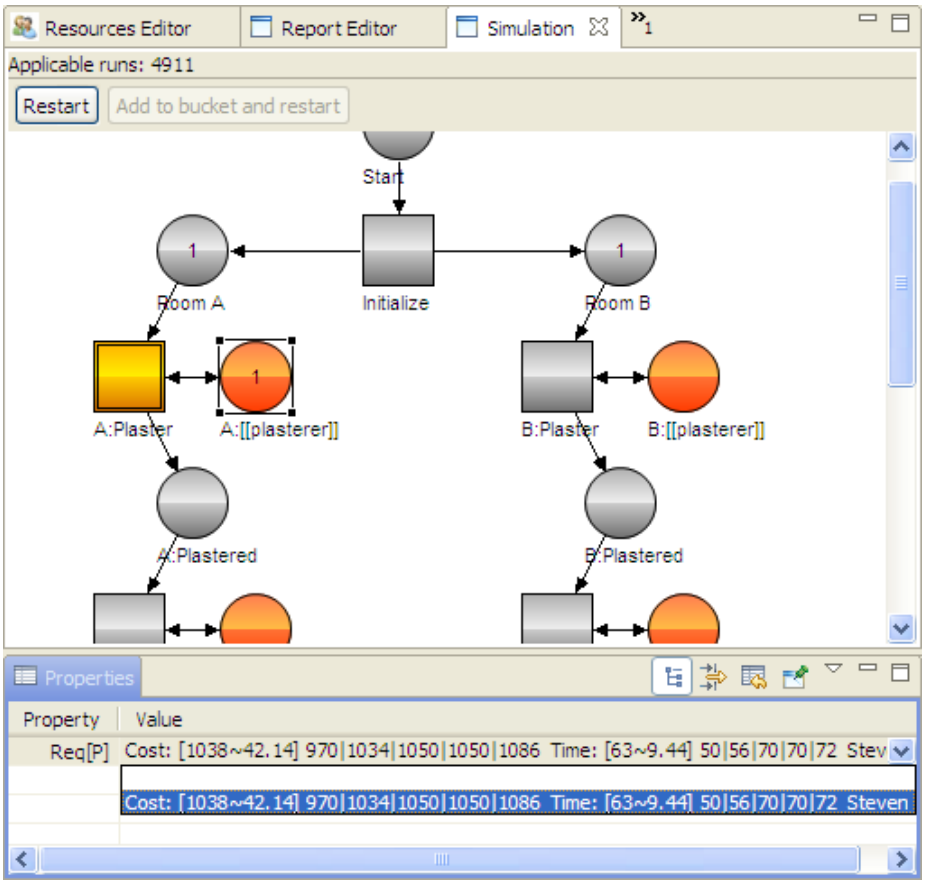

Fig. 6. Simulator

- average cost of completion is 1038 units with standard deviation of 42,14;
- average costs of completion in five successive quantiles are: 970, 1034, 1050, 1050, 1084;
- average time of completion is 63 units with standard deviation of 9.44;
- average times of completion in five successive quantiles are: 50, 56, 70, 70, 72.

When we select the *B:[[plasterer]]* place, the estimates for *Steven's* change to $Cost: \ [1004 - 44.44]\ 950|988|1000|1011|1075$ $Time: \ [65 - 8.73]\ 60|60|60|69|80$. So, starting work in *room A* will cost us more, but it will shorten the total execution time. Knowing this at such an early stage lets us undertake proper decisions from the very beginning. On that basis we may tend to choose *room A* first, if we want to save time. On the other hand, when cost is being considered crucial, we should choose beginning work from *room B*.

When all requirements are met, the corresponding activity becomes active. We can fire such enabled transition, and move to the next activity, where we will have more decisions to make. Note, that at every time, we make a decision or fire a transition, the estimates are being recalculated, to show the current situation in the net. This rule also applies to the *Report Viewer*.

Report Viewer Much more information about current situation in the simulated net can be seen on additional report view. The sample screenshot on Fig. 7 presents the state of report viewer, when *Steven* has been assigned to *A:Plaster* activity in the simulator. Thanks to synchronization between these two tabs, we can always take a look at all the computed details and expected values. Diagrams on the 7 present:

- **Total cost distribution** — average, cumulative cost of run performance in current simulator situation in 10 successive quantiles;
- **Resources cost distribution** — as above, but for each of the resource;
- **Total time distribution** — average time of run performance in current simulator situation in 10 successive quantiles;
- **Resources time distribution** — as above, but for each of the resource.

Diagrams of total cost and time provide us with information about differences between optimistic and pessimistic run executions. When viewing costs of resources, we can see, which of them have major influence on the growth in pessimistic cases. In the example shown above it is a fact that, when considering cost, in the optimistic case (on the left side of the diagram), *Steven* involvement is minimal, and the cost of *Steven's* work even goes below the cost of *Tom's* work. On that basis we can conclude that the main way to cut the cost will be to maximize *Tom's* involvement and minimize *Steven's*. It can be seen that bigger involvement of Steven means smaller involvement of Tom and vice versa. So if time does not matter we will prefer the cheaper resource.

The distribution of resource involvement time gives us somewhat different conclusions. It cannot be explicitly determined whether one of the resources should be favoured to improve the process duration. It may seem at a first glance, that in order to optimize the run we should always choose the fastest resource. But it is not the case, since it can slow down other parts of the workflow. The faster one can be the only one who can perform another action which should not be delayed.

Two additional tables at the bottom of this view present:

- list of runs, which are up to date with current situation in the simulator (including time and cost of them as well as pointing out the three most busy resources);
- list of successive concurrent events for the run selected in the first table.

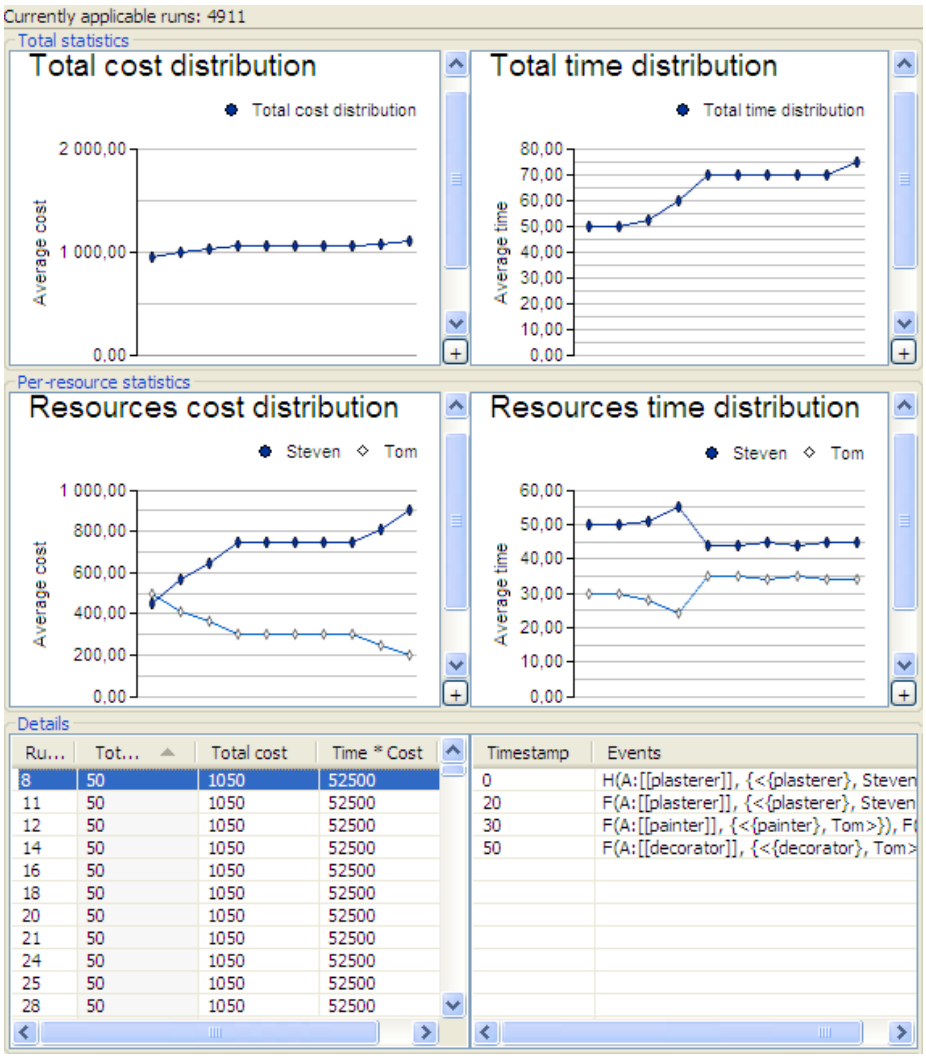

Fig. 7. Report viewer

Bucket Editor To ease the efforts, and allow the user to store the most interesting runs for further analysis, a simple *Bucket Editor* has been created. After the simulation comes to an end, the user can store the run he has just performed manually. This run will be available for later view. The bucket presents information similar to the report viewer, but because it contains information for several, manually chosen runs (not thousands like report viewer), those can be presented in a slightly different form.

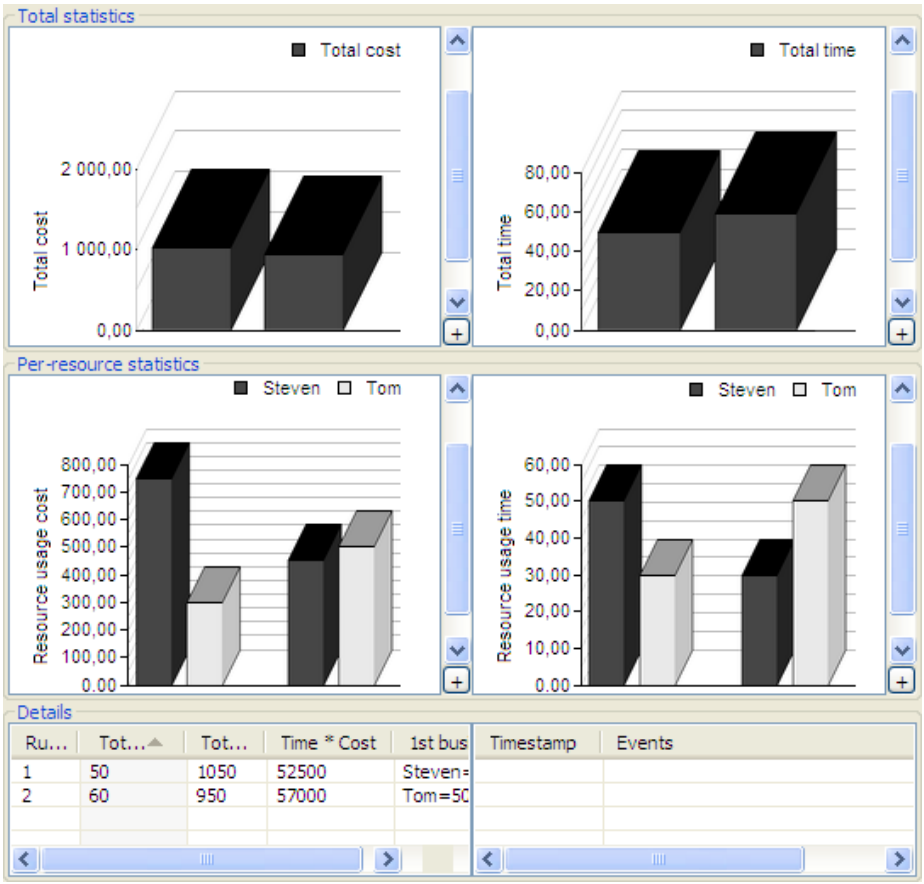

Fig. 8. Bucket viewer

Fig. 8 shows a screenshot of the bucket with two stored runs. As we can see it is organized similarly to the report viever, and the visible diagrams present the same information as those from report viewer:

– **Total cost** — total cost of execution each of the stored runs separately;
– **Resources cost distribution** — as above, but for each of resources;

- **Total time distribution** — total time required to complete each of the stored runs separately;
- **Resources time distribution** — as above, but for each of resources.

The bottom tables represent information about stored runs, showing exactly the same attributes and properties, as the report viewer does.

First of them (displayed on the left hand side of every diagram) has been performed with the goal to minimize the time. Second, though, was made to run the process as cheap as possible, ignoring the time at all. As we can see, the cost reduction requires extra time in workflow execution. Of course, increasing the execution speed increases the cost. Major differences can also be seen in the resource usage distribution in both of these cases. Cost reduction has led to a significant decrease of *Steven's* involvement. In this case, *Tom* takes some of his responsibilities. When the time was the goal, the proportions has significantly changed. *Steven* became more desired, because *Tom* seemed cause the most delays.

4 Conclusions and future work

The provided example shows, how largely the work organization can differ, depending on goals that we want to achieve. In many cases minor changes in the resource allocation can have large influence on the overall result and, conversely, sometimes major allocation changes result in very similar outcomes. Very often humans do not take under consideration all the aspects, which the presented tool does. Its role is to simplify this whole process and make it easier, to find a suitable organization plan that will fit our needs and fulfill the criteria. While the simplification of resource allocation planning is one point, the other one is the possibility to point out uncommon executions, which can lead to increased effectiveness and so increase our profits. Combining user's knowledge and computer's computation power could therefore lead to major design corrections, which would be a basis for further business improvements.

The presented tool is still a prototype, and there are some practical and theoretical issues that need to be addressed. On the theoretical side additional extensions to resource and workflow definition language should be introduced. This includes further conformance to the form of resources described in [BPS09], because only a few resource attributes have already been implemented in this tool. There is also a big potential of the currently used workflow language, which can be further extended. In the context of statistical data tooling, some additional information could be presented (e.g. the Student's t-distribution, $3 - \sigma$). Finally, no measure of reliability have yet been introduced and, in the sense of realism of modelled cases, this is one of the major flaws of the presented tool.

On the more practical side, the integration with existing systems and methods should be made. First of all the statistical runs of the net could generate logs from the run to make them readable by other applications in the same way as the YAWL does [WS09]. The conformance with CPN seems attractive, because of its formal basis and constant development. Secondly, the integration with currently

used common tools is to be made as well. The LDAP for instance is a main source for human resource information. The reporting tools like Microsoft Dynamics AX (Microsoft Business Solutions – Axapta) can be a source of information about resource experience and effectivness of each of such resource. Any other form of gathering of knowledge about past and predictable future resource usefulness can become important in such case. Therefore more integration of this tool will be examinated and developed in the future.

Acknowledgment

The authors would like to thank the reviewers of this work for the effort they put into improving the paper by the constructive reviews, which resulted in a thorough revision of the paper.

Tool description

The tool is written in Java and can be run on any machine with Java (JRE 6.0) installed. It can be downloaded from the http://duch.mimuw.edu.pl/~pch/IRSMT/irsmt.zip. The zip file contains a full version of the tool and a README file, which explains how to install and use the application.

References

[PCh03] Piotr Chrząstowski-Wachtel, Boualem Benatallah, Rachid Hamadi, Milton O'Dell, Adi Susanto, *Top-down Petri Net Based Approach to Dynamic Workflow Modelling*, Lecture Note in Computer Science. v2678. 336-353., 2003.

[BPM05] M. Laugna, J. Marklund. *Business Process Modeling, Simulation, and Design.* Prentice Hall, Upper Saddle River, New Jersey, 2005.

[YA06] Kees van Hee, Olivia Oanea, Reinier Post, Lou Somers, Jan Martijn van der Werf, *Yasper: a tool for workflow modeling and analysis*, Application of Concurrency to System Design, International Conference on, pp. 279-282, Sixth International Conference on Application of Concurrency to System Design (ACSD'06), 2006.

[BS07] Bruce Silver, Bruce Silver Associates, *The BPMS Report: TIBCO iProcess Suite 10.6*, BPMS Watch www.brsilver.com/wordpress, 2007.

[BPS09] W.M.P. van der Aalst, J. Nakatumba, A. Rozinat, and N. Russell. *Business Process Simulation: How to get it right?* In J. vom Brocke and M. Rosemann, editors, International Handbook on Business Process Management, Springer-Verlag, Berlin, 2009.

[WS09] A. Rozinat, M. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, and C. Fidge., *Workflow Simulation for Operational Decision Support.*, Data and Knowledge Engineering, 68(9):834-850, 2009.