Deadline-based Escalation in Process-Aware Information Systems

Wil M.P. van der Aalst[1,2], Michael Rosemann[2], Marlon Dumas[2]
[1] Department of Technology Management
Eindhoven University of Technology, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl
[2] Centre for IT Innovation
Queensland University of Technology, Australia
m.rosemann,m.dumas@qut.edu.au

Abstract

Process-aware information systems are typically driven by process models capturing an idealized view of the actual processes. For example, most process models assume that planned activities happen within a reasonable period. In reality such assumptions may not hold, and as a result workers may be forced to bypass the system or the designer is forced to explicitly model all exceptions that may occur when activities are not handled on time, leading to spaghetti-like models. In this paper, we acknowledge that processes may change when the organization is unable to meet deadlines and refer to such changes as *escalations*. Escalations may change the routing of work (e.g., bypass tasks), change the work distribution (e.g. allow other people to execute delayed activities), or change the requirements with respect to available data (e.g. a decision is made before all information is available). This paper proposes the 3D (Detect, Decide, and Do) approach to deal with escalations and describes a number of escalation mechanisms. The approach has been validated through case studies and simulation experiments.

1 Introduction

This paper focuses on deadline-based escalation, i.e., taking the appropriate actions when getting close to a deadline or when it becomes clear that a deadline will not be met. One of the goals of deadline-based escalation is to let process-aware information systems mimic human behavior when it comes to deadlines. Humans typically change their behavior when confronted with a deadline [11, 27, 31]. Consider for example the Yerkes-Dodson law [31] describing that increased pressure (e.g., an approaching deadline) will improve performance but too much pressure will (eventually) degrade performance. In daily life it can also be observed that humans tend to take more risks when being late (e.g., driving a car). For example, workers may skip tasks or require less information to make a decision. Clearly, such flexibility is desired in in many situations but rarely supported by IT.

Unlike humans, process-aware information systems such as workflow management systems, typically do not change their behavior when confronted with deadlines. As a result, these systems stick to an idealized view of the process even when there is no time or it is even undesirable to stick to this idealized process. Therefore, we introduce the concept of *escalation* in the context of process-aware information systems. Just like a human would "escalate" (i.e., change his behavior) whenever he is unable to meet certain deadlines, we propose the information system to escalate in a similar fashion. Escalation could imply the skipping of tasks, allowing less qualified people to do certain tasks, or making decisions based on incomplete data.

Note that escalations could also be driven by the opposite problem, i.e., the resources of an organization assigned to a process are under-utilized. In this case, escalation could mean taking on

additional activities (e.g. increased pre-sales in a consulting company). The focus of and examples within this paper will be focused on deadline-based escalations, but both cases are very similar.

As a working example, we consider the "teleclaims" process of a large Australian insurance company.[1] This process deals with the handling of inbound phone calls, whereby different types of insurance claims (household, car, etc.) are lodged over the phone. The process is supported by two separate call centers operating for two different organizational entities (Brisbane and Sydney). Both centers are similar in terms of incoming call volume (approx. 9,000 per week), average call handling time (550 seconds), number of call center agents (90) and performance objectives (90% of all calls should be answered in less than 60 seconds). Differences are the underlying IT systems, the physical locations and the modes of operation (24 hrs. versus 9-5). The teleclaims process model is shown in Figure 1. The two highlighted boxes at the top show the subprocesses in the two call centers (Brisbane and Sydney). The lower part describes the process in the back-office.

This process model is expressed in terms of an Event-Process Chain (EPC) [18, 26]. To introduce the notation let us consider the subprocess corresponding to the call center in Brisbane. The process starts with *event* "Phone call received". This event triggers *function* "Check if sufficient information is available". This function is executed by a "Call Centre Agent". Then a choice is made. The circle represents a so-called *connector* and the "x" inside the connector indicates that it is an exclusive OR-split (XOR). The XOR connector results in event "Sufficient information is available" or event "Sufficient information is not available". In the latter case the process ends. If the information is available, the claim is registered (cf. function "Register claim" also executed by a "Call Centre Agent") resulting in event "Claim is registered". The call center in Sydney has a similar subprocess and the back office process should be self-explaining after this short introduction to EPCs.

One challenge for the Australian insurance company handling the process shown in Figure 1 is dealing with an increasing number of incoming phone calls during the Australian storm season (October-March). Storms cause a higher number of damages, raising the number of incoming weekly phone calls to more than 20,000. This not only puts significant burden on both call centers, but also on the succeeding back-office processes related to evaluating and managing these claims. Overtime as one way of adjusting the available resources is applied, but typically can not cope with the entire demand. Thus, to cope with increased call traffic, the insurance company operates an "event-based response system" that differentiates four categories of situations based mainly on how severe the storms are. The first category includes localized storms and flooding and leads to a call volume of 10-50% above average for a period of at least two hours. Due to the increased call volume, customers have to wait for 5-10 minutes in the queue. The second category is triggered if strong winds, hail and structural damage occurs. This leads already to a wait time of 10-30 minutes and the call volume is 50-100% above the forecast for at least two hours. The third category covers wide-spread damage leading to waiting times of more than 30 minutes. The fourth and final category includes extreme and rare cases, in which more than 80 customers would wait on the phone for more than 30 minutes.

Individual response strategies have been defined for each of these four categories. The responses utilize additional external resources as well as a change in the way claims are lodged. First, additional resources are utilized through redeployment of employees from other departments (e.g. sales) and hiring of casual staff. While most of these people are trained, their performance in terms of average call handling time is lower than the performance of the professional call center agents. Second, a streamlined way of lodging the claims is applied in order to reduce the average call handling time and to reduce the waiting time in the queue. In this so-called rapid lodgment process, only a reduced amount of information is collected from the claimant. This leads to an average call handling time of 380 seconds. for experienced call center agents, and 450 seconds. for the additionally employed agents, down from the usual average of 550 seconds. One mechanism to deal with the different performance of these two types of agents is call routing which directs all new and straight-forward cases to the casual additional workforce, while the more complicated follow-up calls are directed to

[1] This case study (including the data provided in this section) is drawn from an interview with a call center manager.

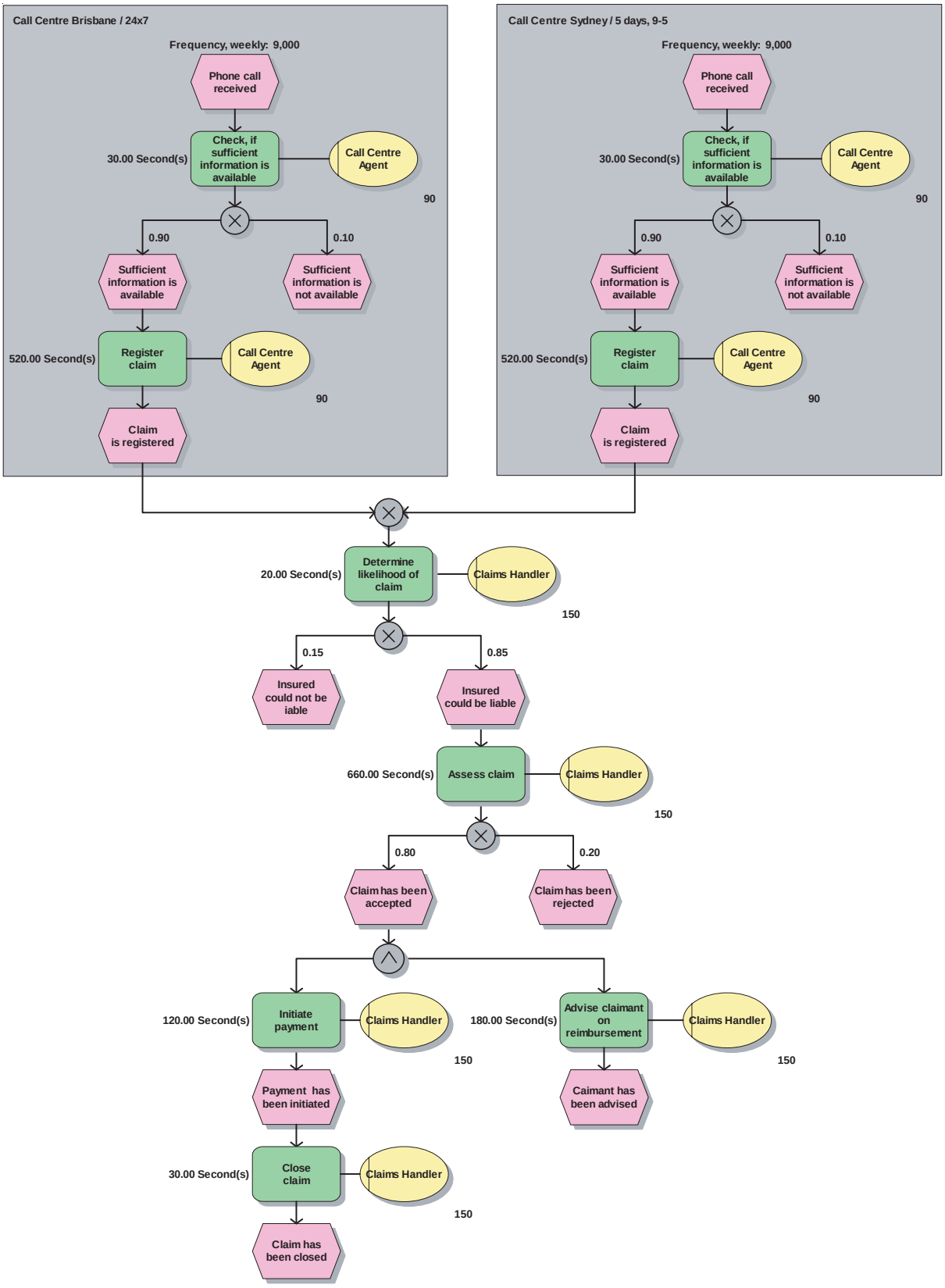

Figure 1: Insurance claim handling scenario (before escalation).

the experienced workforce.

The four categories of situations identified by the insurance company can be seen as four levels of escalation. All levels of escalation involve the rapid lodgment process but the number of additional resources varies per level. The manager in charge for claim services together with managers in charge for the related back-office processes personally evaluate the severance of the weather conditions and trigger the different escalation categories.

The teleclaims process model integrating the above escalation mechanisms (except for the "call routing" step) is shown in Figure 2. In the situation shown there are 30 additional call center agents. Moreover, for the back-office process there are 50 additional claim handlers.

Note that the four levels of escalation refer to the process as a whole. In reality it can also be the case that a single case or a limited set of cases is escalated. For example, it could be that different cases have different deadlines and that due to circumstances some cases have been delayed more than others. Consider for example an insurance case requiring medical information from some specialist and the information is still missing five days before some operation. There is no need to escalate the whole process. It suffices to escalate the single case, e.g., look for another specialist. Another example, would the reviewing process for a conference. If there is a paper that has not been reviewed by any PC member, the PC chair will escalate and try to find a last minute reviewer. If the paper is supposed to be reviewed by three reviewers and only one review is missing while the two reviews agree, the escalation could be to just continue with one review missing.

Although not explicitly mentioned by the people involved in the teleclaims process, we added an escalation mechanism to Figure 2 that escalates a case in isolation if needed. In the original process, function "Assess claim" would on average take 660 seconds. As shown in Figure 2 there are now two functions: one taking 660 second and one taking only 400 seconds. In a way this is similar to the rapid lodgment. However, the escalation does not depend upon the "global" level of escalation. Instead it depends on how long the case is already in the process. If it has been in the process for more than 1 hour, a rapid assessment is done.

In the remainder of this paper, the teleclaims process is used as a running example. Using this case study we will illustrate the challenges of workflow escalation, i.e.,

- What are the available strategies to cope with an increasing waiting time (i.e. escalation strategies)?
- What are the scope, tradeoffs and effectiveness of these escalation strategies?
- How to use simulation to support the selection of the right level of escalation and escalation strategy?

This paper is structured as follows. The next section provides an overview of process-aware information systems and how escalations effect the main perspectives of these systems. Section 3 introduces the 3D approach, i.e., detect, decide and do, as a way of categorizing the main activities of an escalation process. Alternative escalation mechanisms are differentiated for each of the perspectives in Section 4. Section 5 gives insights into the contributions process simulation can make for the evaluation of escalation mechanisms using the teleclaims process. Section 6 describes another case using different escalation mechanisms. The paper ends with an overview of related work (Section 7) and conclusions (Section 8).

2 Process-Aware Information Systems

Process-Aware Information Systems (PAISs) support business processes in organizations based on explicit knowledge of both the organization and the processes. Note that classical applications such as e-mail, spreadsheets, and databases are unaware of the processes at hand. As a result they do not

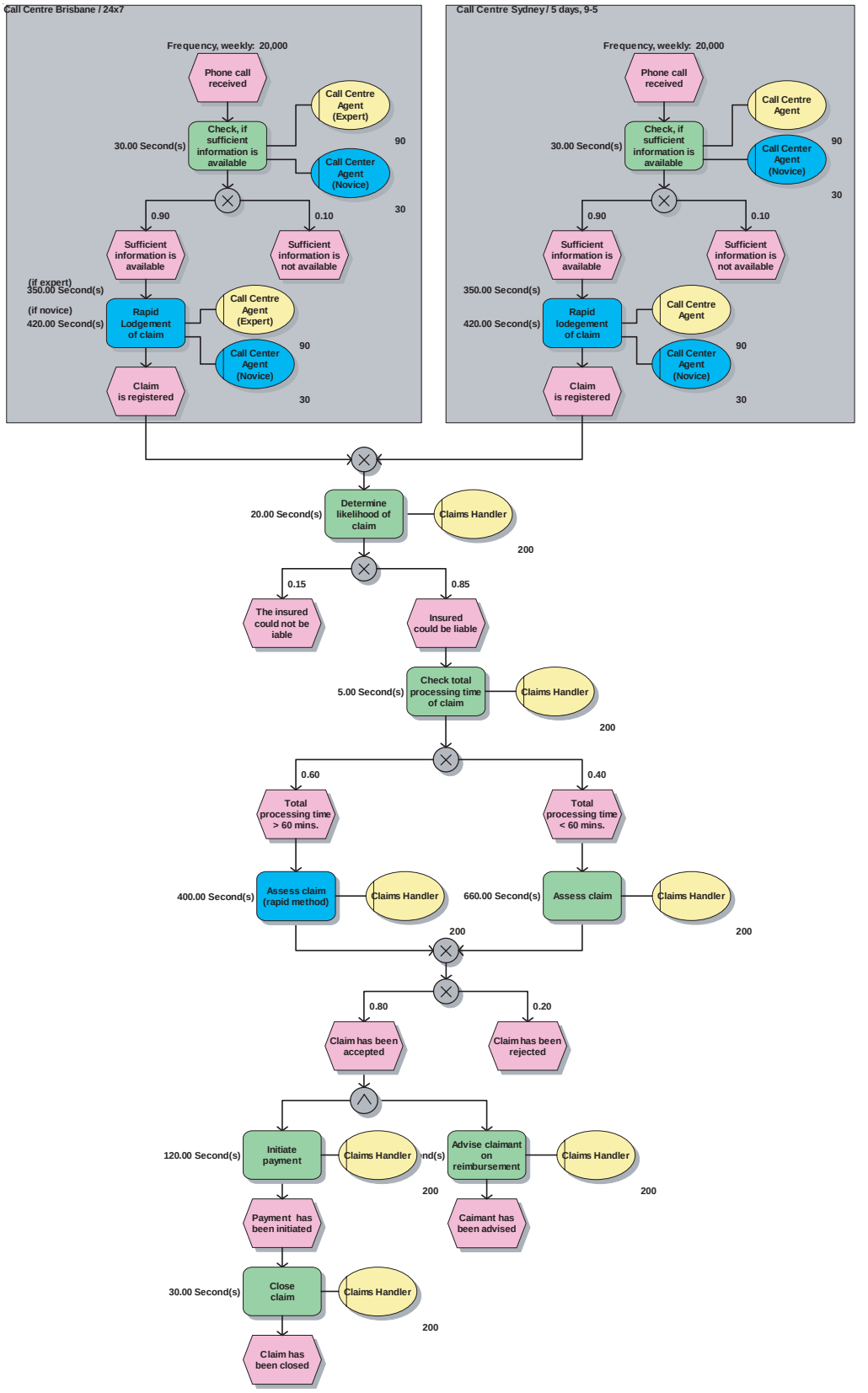

Figure 2: Insurance claim handling scenario including escalation mechanisms.

offer support for the definition, analysis, enactment, control, and monitoring of business processes. Traditionally, organizations hard-coded fragments of business processes in dedicated software. However, since the nineties more and more organizations started to use generic software, e.g. Enterprise Resource Planning (ERP) systems, Workflow Management (WFM) systems, Business Process Management (BPM) systems, etc. WFM systems such as Staffware, MQSeries, and COSA are the most typical examples of a PAIS. Based on an explicit model a process is enacted, or as the Workflow Management Coalition (WfMC) defines it: "A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications." [20]. BPM systems can be considered to be the next generation of workflow technology extending its functionality beyond automation [30]. BPM systems include functionality for monitoring, analysis, flexibility, and cross-organizational processes. ERP systems are also process aware but parts of the system are dedicated to specific processes (e.g. procurement, sales, or finance). These parts can be configured within predefined boundaries. In addition most ERP systems include a workflow component to allow for arbitrary processes. The workflow engines of SAP, Baan, PeopleSoft, Oracle, and JD Edwards can be considered as integrated BPM systems. Note that the class of PAISs is not restricted to ERP, WFM and BPM systems. There are many other systems supporting explicitly modeled processes, e.g. Product Data Management (PDM), Customer Relationship Management (CRM), and Case Handling (CH) systems. Note that systems like the PDM system Windchill and the CH system FLOWer provide a workflow component. Also note that, to date, many organizations still use self-developed PAISs. For example, many banks, insurance companies, governmental organizations have developed dedicated PAISs whose functionality is comparable to the systems just mentioned.

The topic of this paper, i.e. "dealing with escalation when getting closer to a deadline", is relevant to a wide range of PAISs. However, for presentation purposes we often focus on WFM systems as typical examples of PAISs.

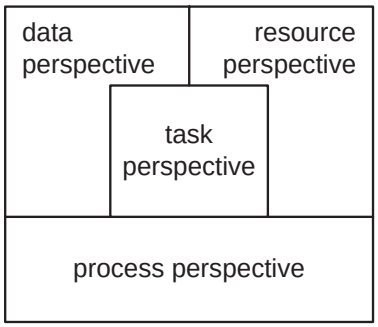

Figure 3: Perspectives of models driving PAISs.

PAISs are driven by models of processes and organizations. By changing these models, the behavior of the system adapts to its environment and changing requirements. These models cover different perspectives. Figure 3 shows some of the perspectives relevant for PAISs (for a detailed discussion of perspectives in the context of WFM systems we refer to [16]). The *process perspective* describes the control-flow, i.e., the ordering of tasks. The *data perspective*, also referred to as information perspective, describes the data that are used. The *resource perspective* describes the structure of the organization and identifies resources, roles, and groups. The *task perspective* describes the content of individual steps in the processes and thus connects the other three perspectives.

Escalations may impact one or more perspectives. Therefore, we elaborate a bit on the perspectives using the example shown in Figure 4. The figure shows four tasks: *T1*, *T2*, *T3* and *T4*. The *process perspective* shows that the four tasks are executed in a sequence, i.e., first *T1*, followed by *T2*, etc. Most real-life processes are not simple sequential processes but include parallel routing,

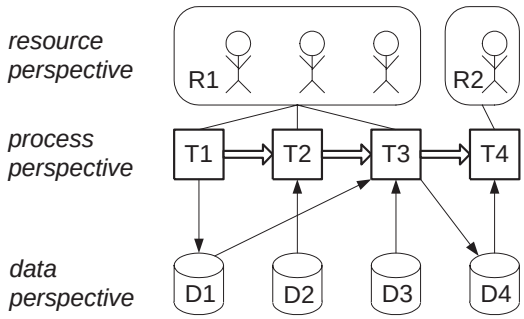

Figure 4: An example illustrating the process, data, and resource perspectives.

conditional routing, and iteration. In fact, as shown in [1], at least 20 routing patterns can be identified. Also note that the sequence shown in Figure 4 is executed for any *case*, i.e., the process may be instantiated multiple times. If Figure 4 would model the reviewing process for a conference with steps *T1*: register paper, *T2*: send to reviewers, *T3*: collect reviews and *T4*: decide and inform author, then these four tasks would be executed for all papers. We assume that the process perspective describes the life-cycle of a case, and therefore we limit ourselves to *case-driven* processes. Approving loans, processing insurance claims, billing, processing tax declarations, handling traffic violations and mortgaging, are other examples of case-driven processes.

While the process perspective is concerned with the ordering of tasks, the *resource perspective* focuses on the resources needed to execute these tasks. Resources may be human (e.g. employee) or non-human (e.g. device, software, hardware). Non-human resources may be consumable (e.g. energy) or not (e.g. a tool). In this paper, we will focus on human resources also referred to as "workers" or "users". To avoid the direct mapping of resources to tasks, resources are grouped into resource classes. An example of a resource class is a *role*, i.e., a group of workers having similar qualifications. Another example of a resource class is an organizational unit (e.g. a department, a team, or a branch). Resources classes can be linked to tasks. For example, in Figure 4 tasks *T1*, *T2*, and *T3* can be executed by any of the three workers having role *R1* and *T4* can only be executed by the worker having role *R2*. A worker may execute multiple tasks and the same task may be executed by multiple workers, however for simplicity we assume that a worker cannot work on two tasks at the same time and that for each case each of the corresponding tasks is executed by a single resource. For example, one resource may execute *T1* for one case while another resource executes the same task for the next case. Note that resource classes may overlap, e.g. Figure 4 could be changed such that the fourth worker also has role *R1* in addition to role *R2*.

Tasks may produce or require information/data. The *data perspective* is concerned with the data required to process the case. Figure 4 shows four data elements: *D1*, *D2*, *D3* and *D4*. These data elements may refer to structured (e.g. the name of a customer) or unstructured (e.g. a Word document) data. There are many ways to describe data using techniques ranging from UML class diagrams to XML schemas. In a PAIS data elements may be linked to tasks as shown in Figure 4. *T1* produces data element *D1*. This data element is again used by *T3*. Besides *D1*, task *T3* also uses *D3* and produces data element *D4*. *T2* uses *D2* and *T4* uses *D4*. Note that *D2* and *D3* are *external* date elements, i.e., they are not produced by the process shown in Figure 4 but need to be supplied by other sources (e.g. another process or another organization).

The task perspective is not shown explicitly in Figure 4. This perspective describes the content of *T1*, *T2*, *T3* and *T4*, i.e., a description of the actual work done in each of the steps.

Although providing only a simplistic view of case-driven processes, Figure 4 nicely illustrates the perspectives. In addition to these classical perspectives of PAIS, it is important for the purpose of escalation to differentiate two more perspectives. First, the *context perspective* describes the environment for which a process model has been designed. The initial example included two contexts:

storm and non-storm season. A change in the context can motivate an escalation. Potential problems with deadlines can often be anticipated based on experiences and a sound understanding of how a specific context correlates with process performance criteria. For example, it is known that a predicted storm will cause a problem in a few hours or days. In such a scenario, it would be too reactive to wait until the first deadline-related problems occur. Second, the *performance perspective* includes all relevant evaluation criteria for a certain process. Escalation may be required even if there are no problems with data, resources or tasks, but instead the performance data has changed, e.g. a customer wants delivery in two days instead of four days. On the other side, modifications in the performance perspective can be one escalation mechanism, e.g. the finish date for make-to-stock production orders can be extended if they compete with problematic make-to-order processes.

The following discussions, however, will be focused on the classical perspectives, i.e., data, resource, task and process. Before we start discussing the 3D approach to escalation, we would like to point out that Figure 4 shows a number of constraints:

- All four tasks need to be executed for any case.
- Task *T2* can only be executed after *T1* completes, etc.
- Task *T1* can only be executed by someone with role *R1*, etc.
- Task *T2* can only be executed if *D2* is available, etc.

Typically, these constraints are considered to be *hard* constraints. In general, PAISs such as WFM systems focus on supporting such constraints. However, under some circumstances it may be useful to consider some constraints as *soft* constraints, e.g. for some cases there may be good reasons to skip task *T2*, execute *T3* without *D3*, or allow someone with role *R1* to execute task *T4*.

3 Escalations: The 3D approach

A PAIS deals with case-driven processes, i.e., cases are handled following the life-cycle in the process perspective and using the data and resources specified in the other perspectives. One of the pitfalls is that PAISs are typically configured on the basis of idealized models. As a result these systems have problems dealing with situations that do not conform with the "normal flow". Often the term "exception handling" is used to refer to the things that need to be done when there are deviations between what is planned and what is actually happening. Although many authors have published interesting results on exception handling [5, 6, 10, 12, 13, 14, 19, 21, 25, 29] (cf. Section 7), today's PAISs still have problems dealing with exceptions. In this paper, we focus on a particular kind of exception: *for one or more cases the organization cannot meet its deadlines.*

We assume that each case c has a deadline D_c. If some cases do not have a deadline, we simply set the deadline to infinity, i.e., $D_c = \infty$. The deadline may be absolute or relative to the time the case started. However, the result is an absolute timestamp. Similarly, tasks may have deadlines, e.g. D_c^t is the deadline of task t for case c.[2] The completion time of a case c, denoted C_c, is the actual time the case was completed. Similarly, C_c^t is the completion time of task t for case c. Preferably, $C_c \leq D_c$ and $C_c^t \leq D_c^t$ for all cases c and all tasks t. A case c is *late* if $C_c > D_c$ and a task t is *late* for case c if $C_c^t > D_c^t$. Note that C_c and C_c^t are only known *after* the completion of the case or task. However, it is often possible to predict the completion time of a case or task. For a given case c, let P_c be the predicted completion time of c and P_c^t be the predicted completion time of task t for case c. Case c is *predicted to be late* if $P_c > D_c$ and a task t is *predicted to be late* for case c if $P_c^t > D_c^t$. Section 3.1 discusses how P_c and P_c^t could be computed.

[2]Note that the notation assumes that there are no loops, i.e., tasks cannot be executed multiple times for the same case. This can be solved in several was. For example, we can assume D_c^t to be the deadline for the last iteration of t for c.

If a case or task is late or predicted to be late, it may be wise or even necessary to take special measures. These measures are called *escalations*. Consider for example the reviewing process for a conference. If close to the deadline for informing the authors still many reviews are missing, then the program chairs may decide to ask additional reviewers, send reminders or base the decision on fewer reviews. To date, such escalations are typically not supported by PAISs, i.e., workers have to work around the system to deal with escalations and are not supported at all. There may be several reasons for escalations, e.g. there may be seasonal influences in the number of cases (e.g. in summer there will be more insurance claims related to fire), the number of available resources may vary (e.g. during the Christmas holidays there are not enough workers to cope with the workload), or there may be some kind of emergency (e.g. a catastrophe or a new law generating more work). These are only few examples of circumstances that may cause cases or tasks to be late. Note that many organizations depend on other organizations (e.g. for information). As a result, a problem in one organization can cause escalations in other organizations.

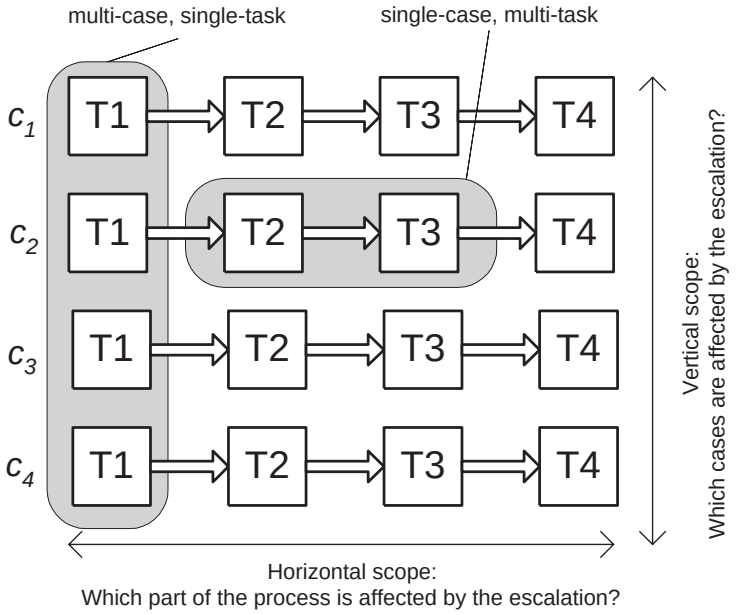

Figure 5: Scope of escalations.

Figure 5 shows that the scope of an escalation may vary in at least two dimensions. First of all, an escalation may involve a single case or multiple cases. An example of *single-case escalation* is a permit request that must be processed within two weeks and a couple of days before the deadline, it is found that there are still several tasks to be done, or worst, that the deadline has expired. On the other extreme, there is the *multi-case escalation* that involves all cases of a given process. An example is the introduction of a new law that forces organizations to handle cases within a certain time-frame. Note that multi-case escalation may also refer to selected cases in a given process, e.g., a cases involving a claim of more that one million euros. Second, the scope of an escalation may refer to a single task or to multiple tasks (i.e. a portion of the process). A *single-task* escalation focuses on a particular task in the process. For example, in the reviewing process of a conference a escalation may result in the skipping of a review step but the author will always be informed, i.e., the escalation is limited to the review task. On the other hand, an example of a *multi-task escalation* is that where the manager of the department is on holidays and he is responsible for a number of tasks while no other resources are allowed to execute these tasks. As a result some tasks are late and work is piling up. An escalation mechanism may be to delegate these tasks to another person.

The teleclaims case described in the introduction proposes a multi-case escalation. A severe storm

does not lead to the escalation of a single case but of an entire process. Note that in the introduction four possible escalation levels were mentioned. These refer to the whole process. However, in the back office there is also a single-case escalation: cases that are delayed more than one hour get a rapid assessment. The multi-case escalation in the teleclaims example is a single-task escalation scenario since it focuses on the initial "lodgment" task of the claim handling process. The single-case escalation (rapid assessment for delayed cases) is also an example of a single-task escalation.

To support escalation resulting from cases and/or tasks that are too late (or are predicted to be late), we propose the 3D approach: *Detect*, *Decide*, and *Do*. First, one needs to detect that there is a problem, i.e., that a case or task is (expected to be) late. Then one needs to make a decision on what to do (i.e., decide which escalation to apply). Finally one needs to execute the escalation that was selected. There are some similarities between the 3D approach and the well-known ECA (Event-Condition-Action) rules. In fact, ECA rules have often been proposed to deal with workflow exceptions [5, 6, 10, 12, 13, 14, 21, 25, 29]. However, the proposed 3D approach differs from the ECA rules approach in several ways. First of all, detecting whether there will be delayed cases is quite different from catching an event and evaluating a condition. Second, the decision process may involve human judgment. Finally, as will be shown in the sequel, we consider a special kind of actions: *mode switching*. The various parts of a process (including all perspectives) may be in different modes. In case of an escalation, the process switches from one mode to another. We refer to this mode as an *escalation mode*. Just like the US Homeland Security Advisory System with its alert levels green (low), blue (guarded), yellow (elevated), orange (high), and red (extreme), we envision processes operating at various levels. Instead of a color code we use numbers where 0 is the normal mode of operation and higher numbers indicate modes corresponding to escalation. Consider for example task $T3$ in Figure 4. This task should be executed by a person with role $R1$ and requires data elements $D1$ and $D3$. At level 0 the PAIS enforces that $T1$ is indeed executed by a person having role $R1$ and that it can only start if both data elements $D1$ and $D3$ are available. Suppose that for a case c, task $T3$ is not executed before its deadline. This is detected and the mode is set to 1. In this mode, task $T3$ may be executed, even if $D3$ is unavailable. If this does not help, i.e., after some time $T3$ is still not executed for case c, the mode is set to 2. In mode 2, $T3$ is offered to all people having role $R2$ where $R2$ includes role $R1$. If this does not help, the mode is set to 3. In mode 2, $T3$ is simply skipped. These examples illustrate the differences between ECA rules and the 3D approach.

Note that in the teleclaims case four levels of escalation were identified. These correspond to escalation modes. Also note that an escalation mode has a certain scope (cf. Figure 5). A process may switch from one escalation mode to another for a single case or multiple cases and for an individual task or multiple tasks.

In the remainder of this section we discuss the three steps *Detect*, *Decide*, and *Do* in more detail.

3.1 Detect

The goal of deadline-based escalations is to avoid cases or tasks that are too late, i.e. $C_c > D_c$ or $C_c^t > D_c^t$, and if they happen to be too late to take rectification measures. We consider detection at the level of a single case and at the level of the process or even the whole organization. As indicated, detecting also includes monitoring the relevant context. However, this type of monitoring is outside the scope of this paper.

Let us first consider detection at the level of a single case c. There are four possible situations that result in a deadline-based escalation.

- $C_c > D_c$ or *time()*$> D_c$, i.e. the case is completed too late. (Note that we use *time()* to denote the current time.) In this situation there is not much that can be done, i.e. the case is too late anyway and only rectification measures to try and compensate for this are possible.

- $C_c^t > D_c^t$ or *time()*$> D_c^t$, i.e. task t is executed too late. In this situation, the case is delayed

with respect to task t. If t is not at the end of the process, this may be a signal to try and speed-up the case. For example, if t was not executed yet, it may be skipped.

- $P_c > D_c$, i.e. the case is predicted to complete too late and an escalation may circumvent this.
- $P_c^t > D_c^t$, i.e. it is predicted that task t is executed too late, and this may trigger some escalation.

To be able to detect this we need to have concrete values for D_c, D_c^t, C_c, C_c^t, P_c, and P_c^t. The first four values are easy to measure. The latter two estimates (P_c and P_c^t) may be calculated in various ways. Some examples:

- Based on historic information one can calculate the average time needed to execute a task (waiting time + processing time). By calculating the longest path from the current state of a case to the desired state (i.e. completion of the whole case or a specific task), we get a rough estimate for the time needed to reach that state. The "prediction engine" of Staffware [28] uses such an approach to estimate the completion time of cases. A major drawback of this type of approaches is that they do not take into account the actual workload, i.e. if there are many cases in the pipeline, then predictions based on historical data of flow times may be too optimistic.
- Instead of using a "static" calculation based on the longest path from the current state of a case to the desired state, it is also possible to use more sophisticated techniques such as queueing analysis or simulation. This is more complicated but the results will be more accurate.
- Most approaches based on queueing analysis or simulation focus on averages, i.e. the normal behavior of the flow in "steady-state". However, these techniques can also take the current state of the process (i.e. all cases) into account and do a transient analysis. For example, the current state of the process can be used to initialize a simulation model. This approach has been successfully applied using the WFM system COSA, the BPM tool Protos, and the simulation tool ExSpect [24]. It results in more accurate predictions for P_c and P_c^t but is more involved.

Factors resulting in delayed cases often do not delay a single case but multiple cases at the same time. Therefore, it may be more suitable to consider multiple cases at the same time for detection purposes. There are two ways to achieve this: (1) aggregate the results for individual cases (e.g. monitor the value of $\sum_c max((P_c - D_c), 0)$); or (2) focus on monitoring the utilization of resources relative to their capacity. The latter approach is attractive because it is relatively simple (if people are too busy then escalate). It relies on the principle that high utilization levels usually indicate that there is a lot of queueing and therefore this can serve as a trigger for resource-based escalations (e.g. increasing the number of staff). It must be noted though that this approach may not detect the need for escalation in some situations. Specifically, if cases are waiting excessively long for external data, then it may happen that the utilization is low and yet there is a need to escalate in order to prevent deadline violations.

Just like the flow time of a case, the expected utilization can be predicted using a wide range of techniques. For example, based on the routing probabilities and the number of cases arriving one can calculate the number of times each task needs to be executed. This combined with historic information about the average processing time can be used to estimate future utilization levels. Also more advanced techniques are possible, e.g. a simulation based on the current state [24].

In Section 2 we discussed a number of perspectives on workflows, including the context perspective and the performance perspective. In the general case, prediction techniques need to take these two perspectives into account. In the teleclaims scenario for example, the decision to do rapid lodgement is not based on the timing of a single case but rather on a human interpretation of the weather forecast, which is part of the context perspective. Similarly, changing performance targets can influence the ability to meet deadlines.

3.2 Decide

Through the detection mechanisms just described, the cases and/or tasks that are (predicted to be) too late are identified. The detection step is followed by a decision step were the escalation measure is selected. There are three possible decision mechanisms: manual, automated, and semi-automated.

For *manual decision making*, the fact that a case or task is (predicted to be) too late is forwarded to a human actor that decides on the actions that need to be taken. The actor can choose from a wide range of actions as will be discussed in Section 4 (e.g. skipping a delayed task). The advantage of human judgment is that a human can take into account "fuzzy" information ranging from the weather forecast to gossip. Experienced workers can make excellent decisions and select the right level of escalation. A drawback is that the human actor may be unavailable or too busy. This way important decisions may be postponed. Note that for manual decision making it is important to forward the escalation detection to the proper actors.

Automatic decision making results in escalations without any human involvement. Based on a set of rules, the right escalation is selected. For example, if the head of the department does not confirm within two weeks, the case is routed to her replacement. Note that automatic decision making requires a rule language, e.g. some variant of RuleML [4]. The advantage of automated decision making is that there are no delays and using the right set of rules the quality of the decision may be high and consistent. The drawback is that rules typically have problems interpreting the circumstances of the escalation, e.g. is the delayed case the "tip of the iceberg" or an isolated case.

To combine the best of both worlds, *semi-automated decision making* may be used, e.g. if a case or task is (predicted to be) too late, first some automatic decisions are made. If these escalations do not help and the situation gets worse, a human may get involved. It is also possible to do it the other way around, i.e. if a case or task is (predicted to be) too late, first a human actor is notified. If this actor does not respond in time, an automated rule is applied.

3.3 Do

The last step in the escalation process (i.e. the "Do step") is the actual escalation. In the next section, we describe possible escalation mechanisms. The intent is not to be complete but rather to provide a framework that process designers can apply to specific scenarios.

4 Escalation mechanisms

An escalation is a deviation from a normal course of action. In the case of a priori escalation, an escalation mechanism implements a tradeoff between on the one hand the amount of time required to complete a task, a case, or a set of cases, and on the other hand the level of service or the resource utilization (e.g. an escalation may result in service degradation or resource redeployment). Different escalation mechanisms strike different tradeoffs. Accordingly, we adopt a model of escalation in which each task has an *escalation mode* and in each mode the task may behave differently with respect to the process perspective, the data perspective and the resource perspective. In the remainder of this section, we examine some escalation mechanisms with respect to these perspectives. For each mechanism, we provide an example and discuss its scope (i.e. single-case and/or multi-case), cost and tradeoffs.

Though we will discuss in the following escalation strategies for each perspective separately, it is important to note that there are two types of interrelationships between the perspectives in terms of workflow escalation. First, a problem in one perspective can trigger an escalation in another perspective as a compensation mechanism. For example, missing information in a lodged claim (e.g. no reports from witnesses of a car accident), may require a specialist for the claim assessment. Second, escalation mechanisms in two different perspectives can be alternatives. In our initial example, rapid

lodgement (process perspective) and the utilization of additional staff members (resource perspective) could also be seen as alternatives.

4.1 Process Perspective

4.1.1 Alternative path selection

Description When defining a process it is possible to specify that certain paths are conditional upon the potential violation of a deadline. In other words, there are different alternatives for performing a part of the process: one corresponding to the normal course, and the others corresponding to different escalation modes striking different cost tradeoffs. Two particular forms of this mechanism are: (i) *alternative task selection*, where a choice is made between executing a given task or executing an alternative (less desirable but faster) task; and (ii) *task skipping* (Figure 6) where a choice is made between executing a task (or set of tasks) and doing nothing (i.e. the task(s) in question is/are optional).

Example In the teleclaims process, the "claim lodgment" task is replaced by an alternative "rapid claim lodgment" task. This is an example of alternative task selection.

Scope This mechanism applies to both single-case and multi-case escalation.

Cost and tradeoffs This mechanism aims at speeding up the process execution in exchange of a degraded level of service (i.e. lower quality of service) or to push some work to later stages of a process (or to other processes) in order to meet a deadline. A cost may be assigned to this mechanism to reflect the loss in quality of service or the impact that it may have at later stages of the process.

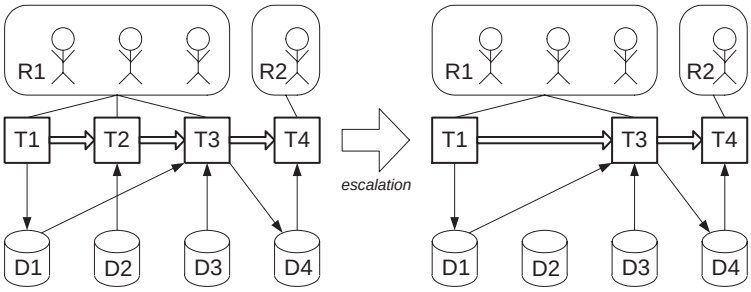

Figure 6: An example of an escalation using task skipping.

4.1.2 Escalation sub-process

Description When it is predicted that a deadline will be violated, or when the deadline is actually violated, a sub-process is spawned off to perform actions specifically related to the deadline violation, such as notifying the appropriate stakeholders, re-negotiating a new deadline, or performing compensation actions and cancelling the case. During the execution of this sub-process, the rest of the process may be suspended.

Example When a deadline violation occurs, a procedure is spawned to notify the customer. The customer is offered the choice to have the process stopped and be reimbursed, or to continue with a new deadline (which can be seen as a modification in the performance perspective).

Scope This is typically a single-case escalation mechanism, however, one can also imagine that a sub-process is spawned off to deal with multiple cases.

Cost and tradeoffs The costs of this mechanism are dependent on the nature of the escalation sub-process.

4.1.3 Task pre-dispatching

Description Under some circumstances, it is possible to start preparing for the execution of a task prior to completion of a previous task. This idea can be found in modern computer architectures, where instructions may be started before completion or preceding instructions in order to exploit otherwise idle resources. Two forms of task pre-dispatching can be identified: pipelining and predictive branching.

In the *pipelining* mechanism, a task B that immediately follows another task A is enabled (i.e. placed in the worklist) as soon as the execution of A starts (i.e. after A has been picked from the worklist and the preparation phase for A has been completed). However, B is flagged as pre-dispatched, in such a way that whenever a resource picks this task from the worklist, it will not be allowed to proceed up to completion until A has completed, thereby ensuring that the control-flow dependency (and any underlying data dependency) is preserved.

Predictive branching applies when a decision point D that immediately follows a task C is reached. The workflow system then attempts to "guess" which branch will be taken (e.g. based on past history), and pre-dispatches the first task in the chosen branch. After completion of C, the branching condition is evaluated, leading to two possible scenarios: (i) the previously chosen branch is the one that should be taken, in which case the branch is allowed to proceed; or (ii) a different branch is taken, in which case the pre-dispatched task needs to be retracted (i.e. the task is withdrawn from the worklist, and if a resource has already picked it, the resource is notified and any preparation actions are undone). A variant of predictive branching is *predication*[3], whereby all the branches are taken in parallel, rather than a "guess" being made for one of them. In this case, the first task of each branch is pre-dispatched, and some of these tasks are retracted when the branching condition is evaluated.

Example Before a clean-up team is authorized to enter an area it may be necessary to wait for approval from an inspection team, however, the preparation of the clean-up (e.g. setting up the clean-up equipment) could be pre-dispatched after a certain point in the inspection process.

Scope This mechanism applies to both single-case and multi-case escalation. The mechanism is only applicable when the tasks in the process have an explicitly defined preparation phase and in the case of predictive branching and predication, the resources must be able to undo any preparation actions.

Cost and tradeoffs The cost of this mechanism is determined by two factors: (i) heavier resource utilization as resources prepare tasks and then hold until they can start the actual execution; and (ii) in the case of failed predictive branching, the cost of undoing the preparation actions.

4.1.4 Overlapping

Description Overlapping can be applied for workflow escalation in the case of large batches and involves two sequential activities. The main idea behind overlapping is that two activities can be accelerated, if they are parallelized. This goes further than pipelining as the following task will be started (not just prepared) while the preceding is still processing.

[3]This technique has parallels with the one used in the Intel Itanium processor (`http://www.devx.com/Intel/Article/20218/2217`).

Example A specialist works on 10 claims in one batch, before he forwards the entire batch for further processing to the next resource. Overlapping would mean, that he forwards smaller batches, e.g. in the size of two each.

Scope This escalation mechanism is related to a single case escalation.

Cost and tradeoffs The benefit of an accelerated processing of two sequential activities has to be compared with the increasing costs related to increased coordination efforts between these two activities.

4.1.5 Prioritization

Description Higher priorities are given to certain tasks or cases, letting them overtake other cases in the consumption of resources.

Example If there are twenty customers with orders for a particular service and it is predicted that half of them will result in deadline violation, the potentially late cases are given higher priority.

Scope This is a multi-case mechanism.

Cost and tradeoffs Giving higher priorities to some tasks or cases necessarily means lowering the priorities of others. This may result in deadline violations for certain tasks or cases that would not have occurred had the priorities been left unchanged. Apart from this, and the usual overhead on the process execution environment of detecting, deciding and triggering the escalation procedure, the cost of this mechanism is neutral. This mechanism may be attractive in cases where being late by a small amount of time or being late by a larger amount of time have more or less the same implications. (Note that often *service levels* are defined as the percentage of cases on time.) For example, if there are several cases running late, one may choose to focus on some of them in order to meet their deadlines, and neglect the others, even if this makes these other cases violate the deadline by more time than they would otherwise have done.

4.2 Resource Perspective

4.2.1 Resource redeployment

Description The idea of resource redeployment is to increase the capacity of the resources associated to cases or tasks that are running late as illustrated in Figure 7. Resource redeployment can take many forms including: *adding more resources* (e.g. moving people between departments), *extending the scope of the roles associated with a task* (e.g. allow people with a lower role to execute the task,), *increasing the capacity per resource* (e.g. overtime) or *changing the allocation of tasks to achieve load balancing*. Therefore the "Resource redeployment" mechanism can be seen as a collection of mechanisms aiming at achieving an increase in resource capacity.

Example In the teleclaims process, an escalation immediately leads to overtime being requested from the call center operators. If this is not enough, employees from the sales and service departments are redeployed to the teleclaims process, and if necessary, casual workforce is called upon.

Scope This is a multi-case escalation mechanism.

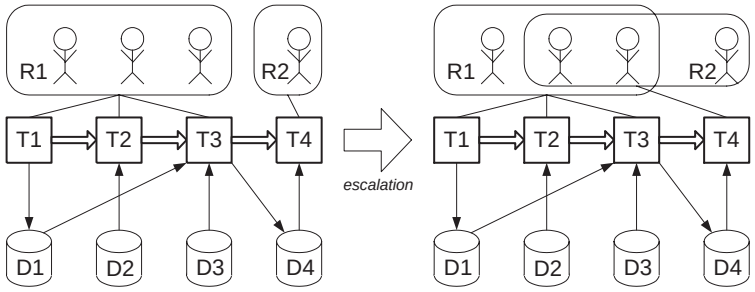

Figure 7: An example of escalation using resource redeployment.

Cost and tradeoffs Increased costs including variable costs (overtime and hiring additional workforce) and fixed costs (need to train people to be able to be redeployed or to train a pool of potential casual workers). Furthermore, the average performance per resource may be negatively impacted as in the teleclaims scenario. Besides increased costs there is the risk of lower quality levels. If resources start doing tasks outside of their normal routine or expertise area, the impact on quality may be negative (e.g., more errors or a less professional service to the customer).

4.2.2 Batching

Description In some circumstances it may be possible to group together tasks that would be more efficiently treated as a batch assigned to a single resource. This approach can be effective in reducing the number of deadline violations when the tasks being batched are predicted to have deadline violations. For example, in location-based batching, tasks from several process instances are clustered based on their associated location and the location of the available resources. Task instances "close" to each other in space are executed in batch by a resource, before the resource moves to another location.

Example In the insurance claim handling process, when an event takes places at a given location (e.g. a bushfire), all on-site assessment tasks related this event are batched and assigned to one or a group of dedicated assessors.

Scope This is a multi-case escalation mechanism.

Cost and tradeoffs Batching accelerates activities by eliminating setup times for individual activities. Costs can occur for the efforts related to batching activities.

4.2.3 Splitting

Description Splitting is the opposite of batching and refers to the case, in which finalizing the work for an entire batch of cases would take too long. In these cases, it might be possible to split the batch into smaller batches, which are worked on in parallel.

Example In a typical claims process, legal experts are involved as specialist. However, such an expert can easily become the bottleneck. Instead of one full-time legal expert, it might be useful in some case to have two (or more) experts working part-time and in parallel.

Scope This escalation mechanism converts a single case escalation into multi-case.

Cost and tradeoffs This approach reduces the processing time by splitting a batch over a number of resources, who work in parallel. This benefit has to be compared with the additional setup time/costs at each resource, possible additional transportation costs and the efforts related to consolidating the cases again to one batch for the next activity.

4.3 Data perspective

4.3.1 Deferred data gathering

Description The gathering of certain data is postponed until the point in the process where it is actually needed. In other words, data items that would normally be produced by a given task are not produced when this task completes, but instead, they are gathered by the (first) task that needs them.

Example In the teleclaims scenario, a simplified version of the claim creation form is used during escalation.

Scope Applies both to single and multi-case escalation.

Cost and tradeoffs Deferred data gathering usually results in work being pushed to a later point in the process. In the case of the teleclaims process, when the simplified version of the claim creation form is used, some relevant data is not gathered. Some of these data is not necessary in some cases, and when it does become necessary, a call is made by the relevant claim handling department to the customer. In the case of claims where an on-site assessment needs to be made, the missing data may be gathered by the assessor. Note that data degradation can be used in conjunction with "alternative path selection". For example, in the teleclaims scenario an alternative version of the "lodge claim" task is associated to the simplified version of the claim creation form.

4.3.2 Data degradation

Description Tasks are allowed to be executed with less or different data. For example, if a document is not available, the decision can be taken without it. It is also possible to look for other sources of less reliable or more costly data.

Example During a paper review process, the acceptance/rejection decision is usually taken on the basis of three reviews. However, if one of the reviews is missing by a given deadline, the decision may be taken with only two reviews.

Scope Applies both to single and multi-case escalation.

Cost and tradeoffs The strategy results in loss of quality of service, and in certain cases, it may result in some work being pushed to a later step in the process.

5 Simulation study: Teleclaims processing in the storm season

Let us now return to the teleclaims process shown in Figure 1. To illustrate the effect of different escalations, we take this process and evaluate different scenarios using simulation. Note that some of these scenarios describe the way the Australian insurance company is escalating during the storm season. Other scenarios are merely used to provide a better coverage of the escalation mechanisms discussed in the previous section.

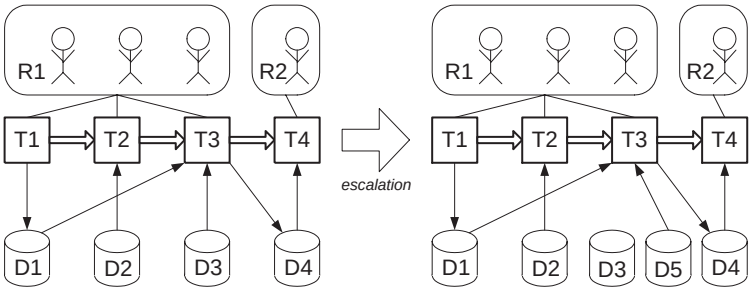

Figure 8: An example of an escalation in the data perspective.

For our simulation study we use CPN Tools [7] which is based on Colored Petri Nets [17] as a modeling and analysis language. The reasons for using CPN Tools are its expressiveness (it is easy to model all escalation mechanisms), its theoretical basis (allowing for different types of analysis), and its simulation speed (close to a classical programming language).

Let us first simulate the process shown in Figure 1. We assume the arrival process to be Poisson (i.e., negative exponential interarrival times). Since the distribution of the call volume and number of resources over the day was not given, we assumed these to be constant over an 8 hour period per day. All activities (i.e., the functions in the EPC diagram) are assumed to have a negative exponential service time. The average service times are indicated in the diagram and so are the routing probabilities. For example, 10% of the incoming claims stop after the first step in the process. The numbers of resources are also shown in Figure 1: there are 90 call center agents in each call center and 150 claims handlers in the back office. Note that the same resource will execute all steps in the process for a given claim in one of the call centers or the back office, i.e., transfer of work only takes place in between a call center and the back office.

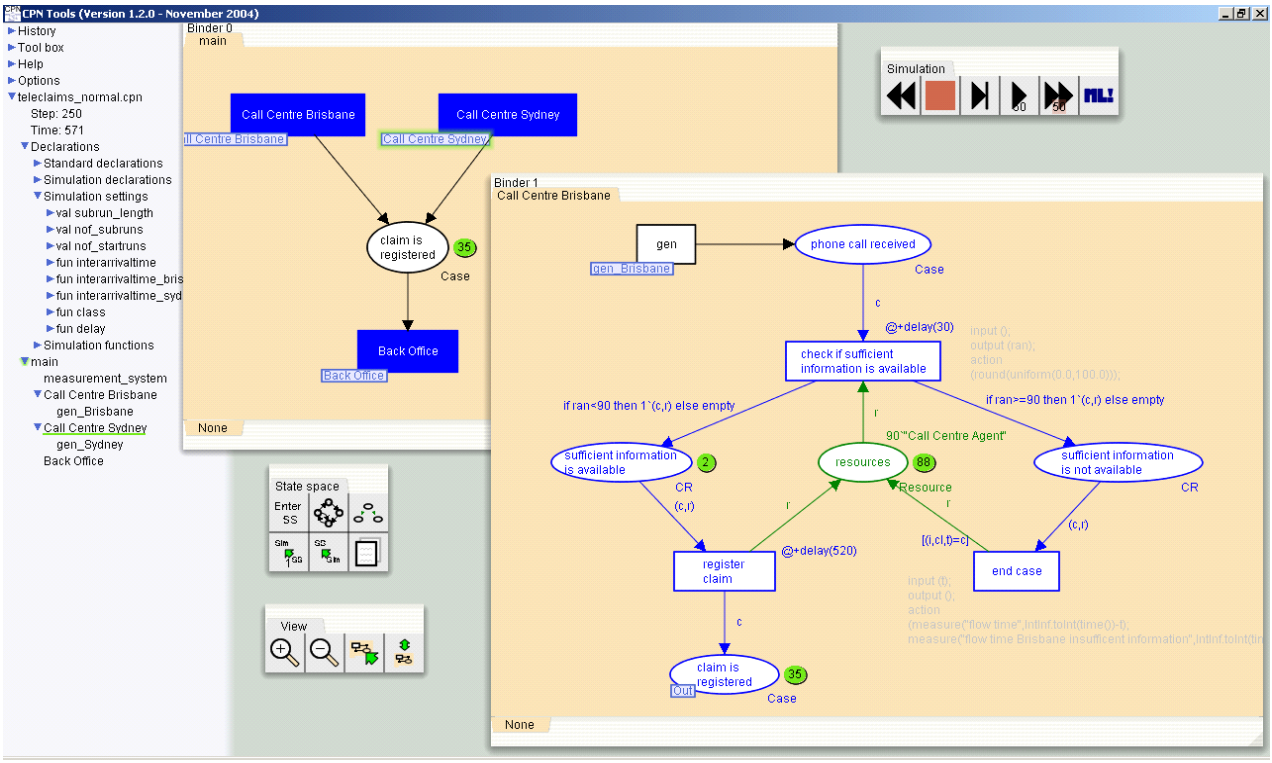

Figure 9: A screenshot of CPN showing the top-level model and the Brisbane call center.

Figure 9 shows a screenshot of the top-level model and the Brisbane call center in CPN Tools while simulating the model. Figure 10 shows the CPN model of the back office. A detailed description of the CPN models is outside of the scope of this paper. However, a superficial comparison of Figure 1 and the figures 9 and 10 will assist in obtaining a basic understanding of the CPN models.

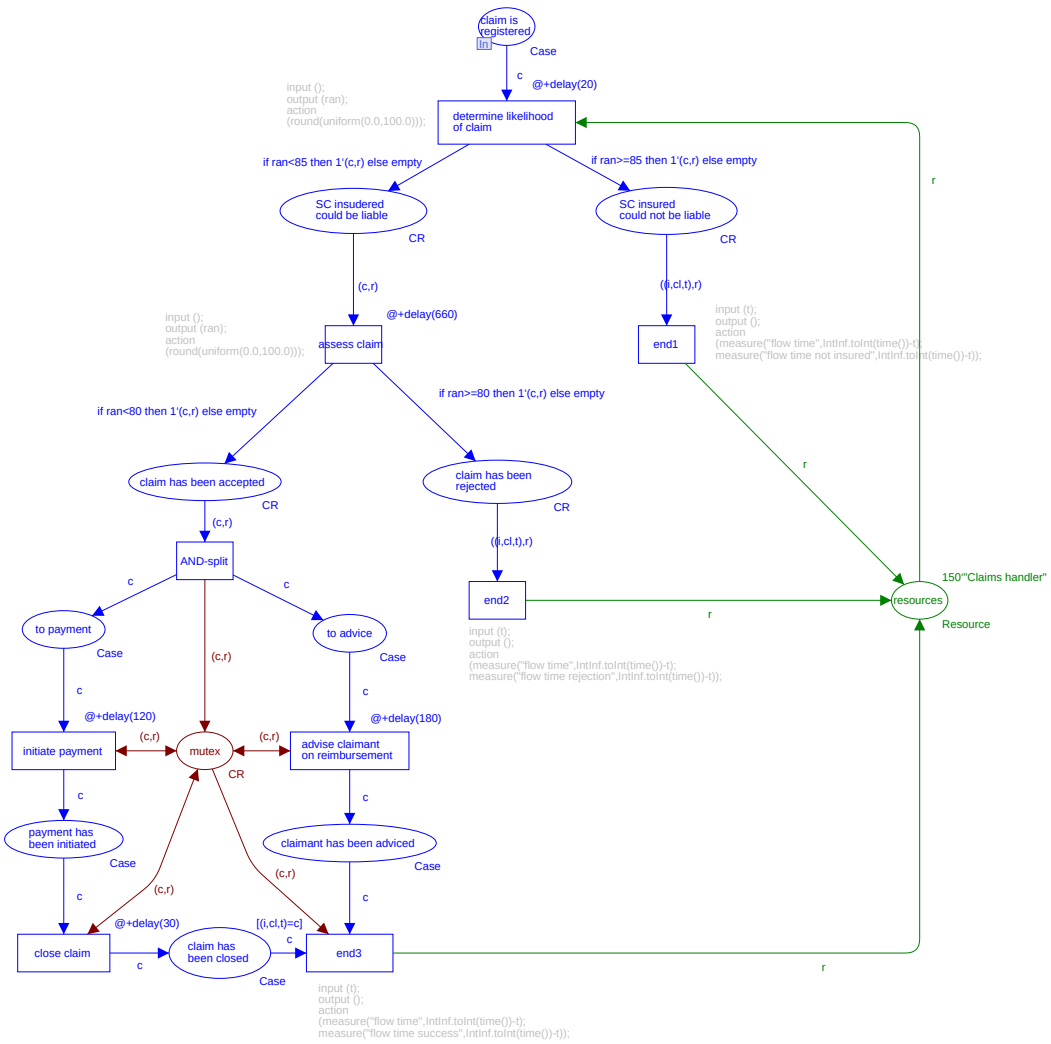

Figure 10: The CPN model of the back office.

For calculating confidence intervals we assume a start run of 2 days and 10 subruns of 1 day (8 hours). The simulation shows that the average flow time is 1271 seconds, i.e., about 21 minutes. The variance of the flow time is 928266 (with a 95% confidence interval of [911853,944679]), i.e., the standard deviation is approximately 963. This implies that there are considerable differences between individual flow times. The average flow time of successful cases is 1667 [1660,1673]. The utilization of the call center agents is 0.26 and utilization of the claims handlers is 0.60.

The process shown in Figure 1 cannot deal with the incoming claims in the storm season where the average number of claims per week goes up from 2*9000=18000 to 2*20000=40000. The utilization of the claims handlers would be more than 1.0 indicating that it is impossible to cope with the flow of work.

Figure 2 shows the way the insurance company deals with 40000 per week. (Note that the model contains both a single-case and a multi-case escalation.) The changes are the addition of resources to both the call centers and the back office (i.e., the Resource redeployment mechanism), the rapid lodging (i.e., the Alternative path selection mechanism), and rapid claim assessment (also

the Alternative path selection mechanism but now applied as a single-case escalation). As a result of these escalations, the organization can cope with the incoming claims and the average flow time is 937 seconds [932,943].

6 Another simulation study: Reviewing journal papers

The teleclaims simulation study used the *Resource redeployment* mechanism and *Alternative path selection* mechanism to escalate. In the study no escalation mechanisms corresponding to the data perspective were used. Moreover, the study was primarily a multi-case escalation (during a storm the whole process is escalated). Therefore, we provide another simulation study focusing on single-case escalation and also including escalation mechanisms corresponding the data perspective.

In this section, we consider the process of reviewing process for a journal. Each paper submission to the journal is registered by the editorial officer (this takes on average 2 hours). The editor of the journal will ask three reviewers to review each submission. The request of the editor to the reviewer will generate a response of the reviewer who either accepts or declines the invitation. If the reviewer accepts to review the paper, the editorial officer wil send it to him/her. If the reviewer declines the invitation, the editor will invite another reviewer. This process is repeated until for each paper three people have accepted the invitation. A reviewer that accepts the invitation, will review the paper and propose to accept it or reject it the paper. (For simplicity, we assume that the outcome of any review is "accept" or "reject".) The editor will make a decision when all three review reports have been returned. As long as a review is missing, the decision is postponed. If two reviewers propose to accept, the editor will accept the paper. Otherwise, the paper is rejected. The editorial officer will inform the author of the result (this takes on average 2 hours of working time). Finally, the case is closed.

For the simulation, we assume that on average 4 new papers arrive per week. The arrival process is Poisson (i.e., negative exponential interarrival times with a mean of 7/4=1.75 days). The task execution times are also sampled from a negative exponential distribution. Handling the response of a reviewer takes on average 1 hour of working time from the editorial officer. Reviewers on average reply to an invitation within one week (negative exponential distribution with mean of 1 week) and 70% accepts to review the paper. This implies that for the other 30% the editor appoints another reviewer, etc. The average reviewing time is 6 weeks after accepting the invitation and sending the paper. On average, 60% of papers are rejected by the reviewer. Since the editor "follows" the reviewers, on average only 40% is accepted. The editorial officer is assumed to be continuously available. However, the editor only checks on the papers at the end of every month (every four weeks to be precise). At that point in time, he can decide to appoint new reviewers and accept/reject papers.

Figure 11 shows the top level of the CPN model. Note that the model has four subprocesses: *gen_papers* (for arrival of new papers), *check_papers* (the handling of the papers by the editor at the end of each month), *reviewer_contact* (the actions related to the management of reviewer responses), and *reviewers* (an external process modeling the reviewer base). In this paper, we will not show or describe these subprocesses in detail. The textual description given above should resolve most ambiguities. For our simulation experiments we use a start run (to avoid start-up effects) and 10 subruns. Each subrun takes one year (52 weeks) and the subruns are used to calculate 95% confidence intervals. For the base scenario (i.e., without escalation) the average flow time is 131 days (with a 95% confidence interval of [129,133]), i.e., approximately 19 weeks. The average variance of the flow time is 2733 (with a 95% confidence interval of [2587,2880]). This implies that frequently the review process takes half a year or longer.

Taking this base scenario (without any form of escalation) as a point of departure, we now investigate the effects of various escalation mechanisms:

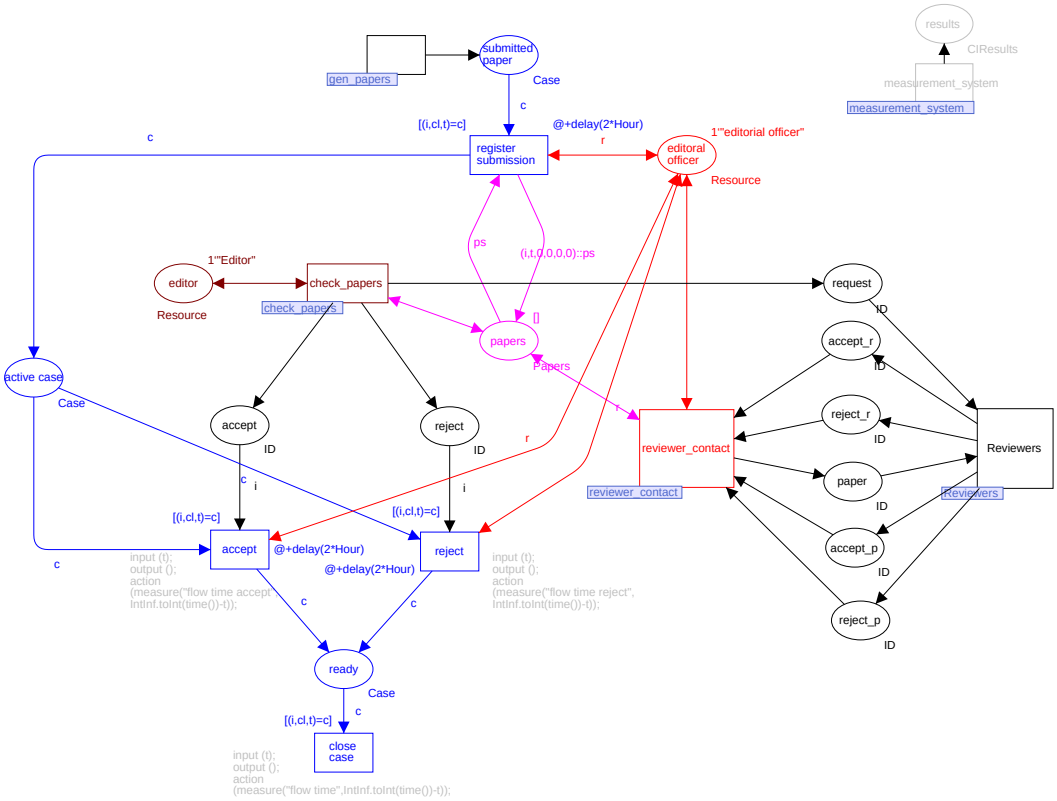

Figure 11: The top-level CPN model of the reviewing process.

Escalation 1 For the first escalation, we invite an additional reviewer if the reviewing process takes more than one month. This means that in the end 4 people may review the same paper, but after three reviews arrive a decision is made. If there happen to be 2 accepts and 2 rejects, the paper will still be accepted. Clearly, this escalation uses the *Resource redeployment* mechanism described in Section 4.

Escalation 2 The second escalation is similar to the first one but now up to two reviewers may be added if there is no conclusion after one month.

Escalation 3 Again the *Resource redeployment* mechanism is used to allow one additional reviewer. However, now, if needed, the additional reviewer is only invited after two months.

Escalation 4 The fourth escalation is a combination of the previous two: Potentially up to two additional reviewers are invited after two months.

Escalation 5 Now we focus on the invitation of reviewers. If after one month there are still reviewers that have to accept or decline the invitation, we invite one additional reviewer.

Escalation 6 Again we focus on the invitation of reviewers. If after one month there are still reviewers that have to accept or decline the invitation, we invite up to two additional reviewers. This means that in the end 5 people may review the same paper.

Escalation 7 We now apply the *Data degradation* mechanism, i.e., make decisions with less information. Note that if a paper gets two positive reviews, it is certain that it will get accepted. Therefore, two positive reviews or two positive reviews will result in a final decision and there is no need to wait for the third review.

Escalation 8 Escalation 7 may discard the third review even if the third reviewer only takes a bit longer than the other two. Therefore, we refine the *Data degradation* mechanism with a temporal component. Now two positive reviews or two negative reviews may trigger a final decision, but only after one month has passed.

Escalation 9 Combines Escalation 1 and Escalation 7, i.e., a combination of the *Resource redeployment* mechanism and the *Data degradation* mechanism is used.

Escalation 10 Combines Escalation 1 and Escalation 8, i.e., as in the previous escalation but now with the temporal component.

For each escalation we conducted a simulation experiment using CPN Tools.[4] The results are shown in figures 12 and 13.

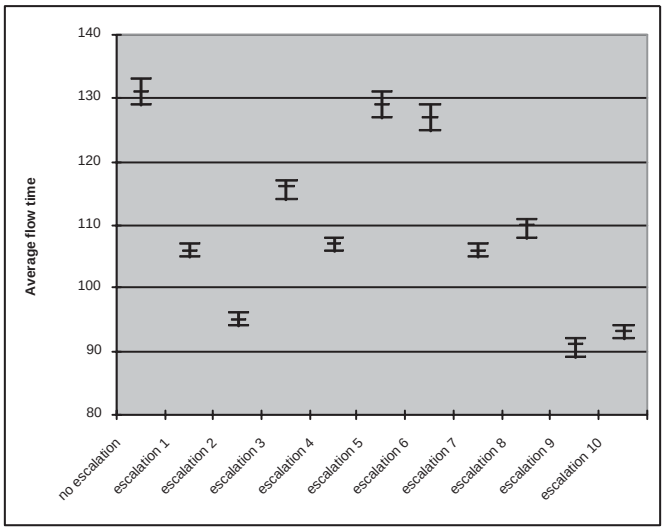

Figure 12: The average flow time and 0.95 confidence intervals for the base scenario and the 10 escalations.

Figure 12 shows the average flow time for the base scenario and each of the 10 escalations. In each case, the average and the 95% confidence interval are given. The diagram shows that each of the 10 escalations provides an improvement. However, for Escalation 5 and Escalation 6 the improvements are not significant, and at best only marginal. This shows that the problem is not in the first part of the process, i.e., the invitation of reviewers. All other 8 escalations provide a clear improvement in terms of flow time. If we focus on the first four escalations, we see that the *Resource redeployment* mechanism works. Moreover, the earlier and the more reviewers get involved, the better the end result. Escalation 7 and Escalation 8, i.e., the use of *Data degradation*, have a positive effect on the flow time. Combining the *Resource redeployment* and *Data degradation* mechanisms, further reduces the flow time. As could be expected, Escalation 9 is slightly more effective than Escalation 10 because in the first month one does not have to wait for the third review if the first two agree.

Figure 13 shows the variance of the flow times for each strategy. Here it is surprising to see that the *Resource redeployment* mechanisms significantly reduce the variance while the *Data degradation* mechanisms do not. For example, Escalation 7 and Escalation 8 have a variance comparable to the base scenario and the less effective strategies Escalation 5 and Escalation 6. This can be explained as follows. There will be cases with two positive or two negative reviews that are handled quickly. However, for cases with a positive and a negative review, the reviewing process will take as long as

[4]Note that a simulation run of 11 years only takes about 10 seconds on a laptop (Intel Pentium M Processor, 1.40GHz, 512MB).

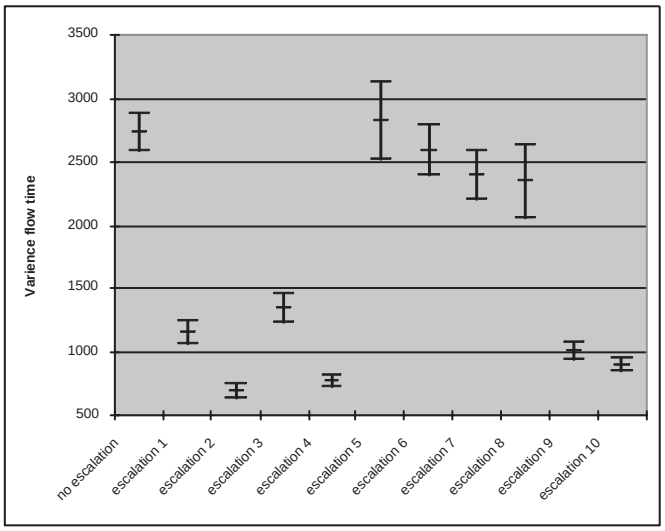

Figure 13: The variance of the flow times for each scenario.

in the base scenario. Hence, there will be a cases that take a long time and that do not benefit from the escalation. The first four escalations are more robust because one or two more reviewers are involved. Therefore, their variance is smaller.

The above discussions illustrate the benefits of simulation as a tool to investigate the effects of the escalation mechanisms described in Section 4 in a given scenario.

7 Related work

In social sciences, researchers have investigated the effect of pressure (e.g., an approaching deadline) on the performance of people [11, 27, 31]. These studies suggest that workers change the way of working when confronted with a deadline. In most cases the effect is positive, i.e., people manage to get the work done in time. Unfortunately, current workflow management systems do not support or mimic human behavior in the presence of deadlines. A notable exception is the the FlowConnect system of Shared Web Services [15]. FlowConnect supports the definition of milestones that have planned values and actual values. These milestones are used to generate escalations and timeouts. An escalation in the context of FlowConnect means that a user is signaled that it is now critical to perform an action, i.e., the corresponding work-item is highlighted in the user's worklist. A time-out means that an action is executed if a milestone was not reached in time.

Deadline escalation can be seen as a special type of exception handling. Many proposals have been put forward to address various aspects of exception handling in workflow systems [5, 6, 10, 12, 13, 14, 19, 21, 25, 29]. Some of these previous proposals (e.g. [5]) advocate an ECA rules-based approach, which we contrasted with the 3D approach in Section 3. Other proposals such as [12] focus on failures and rare or unexpected events rather than the ability to meet deadlines. Finally, a number of generic frameworks for structuring exception handling knowledge have been put forward. In particular, parallels can be drawn between the phases of the 3D approach (Detect, Decide and Do) and the phases for exception handling proposed by Klein & Dellarocas [19], namely "Preparing for exceptions", "Diagnosing exceptions" and "Resolving exceptions". In this respect, the 3D approach can be seen as a refinement of the general exception handling framework of Klein & Dellarocas. In the terminology of Klein & Dellarocas, what the 3D approach brings is an ontology (including resolution mechanisms) for a specific subclass of exceptions (namely deadline escalations).

Some researchers have proposed techniques addressing the specific issue of deadline escalation in workflow systems. For example, Panos & Rabinovich [22] describe an approach to dynamically adjust

deadlines based on costs, expected execution times, and available slack time. In [23] this approach is refined and supported by simulation experiments. In [8] the topic of capturing time constraints in workflow definitions is considered, and a PERT-like technique for the analysis of the temporal behavior of a workflow is proposed. This technique is similar to the one employed by the "prediction engine" of Staffware [28]. Unfortunately, these techniques implicitly assume that processing times are independent of the workload and resource capacity.

The notion of "process fragments" in Staffware [28] allows composite activities to be bound at runtime to specific subprocesses. This concept can be used to incorporate escalation mechanisms into a workflow description. For example, based on the escalation level a specific subprocess may be selected at a specific point in a workflow.

There is also a link between detection for deadline-based escalation and Business Activity Monitoring [9]. For example process mining techniques [2, 3] can be used to measure, predict and explain escalations. To the best of our knowledge, the application of these techniques (generally aimed at offline analysis) to support the "Decide" phase of deadline escalation at runtime, is an open question.

8 Conclusion

Although organizations are forced to escalate regularly, today's (process-aware) information systems offer little support for this. In this paper, we focused on escalations triggered by the (predicted) inability to meet deadlines. Using an example of the teleclaims process of an Australian insurance company, we identified and analyzed issues related to deadline-based escalation. We then proposed a general approach to deadline-based process escalation and we presented various escalation mechanisms. The effectiveness of these mechanisms was evaluated through simulation studies.

Future work will aim at designing and evaluating cost models for escalation. Such cost models are key during the decision phase, when the cost of applying an escalation needs to be weighed against: (1) the extent to which it decreases the probability of violating certain deadlines (or violating them to lesser degrees than without escalation); and (2) the cost of these deadline violations. On the basis of such model, it would then be possible to answer other key questions such as: (1) when to apply a given escalation mechanism (individually); and (2) which combinations of mechanisms are most likely to work effectively together. Finally, it is necessary to design ways to seamlessly incorporate the cost model and the escalation mechanisms into existing process-aware information systems, and in particular workflow systems. Today's systems are typically unable to predict future problems and modeling the various escalation mechanisms result in spaghetti-like diagrams that cannot be maintained easily. A process modeling language that allows to clearly distinguish between a "base" process model and "escalated" versions of this model would be desirable.

Acknowledgments The third author has been funded by a Queensland Govenerment "Smart State" Fellowship co-sponsored by SAP Australia Pty Ltd. The author would also like to thank Greg Bird from Advanced Data Integration for our fruitful discussions on deadline escalation mechanisms. Many thanks also to the anonymous organization that kindly provided the teleclaims case study.

References

[1] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[2] W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.

[3] W.M.P. van der Aalst and A.J.M.M. Weijters, editors. *Process Mining*, Special Issue of Computers in Industry, Volume 53, Number 3. Elsevier Science Publishers, Amsterdam, 2004.

[4] H. Boley. The Rule Markup Language: RDF-XML Data Model, XML Schema Hierarchy, and XSL Transformations. In O. Bartenstein, U. Geske, M. Hannebauer, and O. Yoshie, editors, *Web Knowledge Management and Decision Support, 14th International Conference on Applications of Prolog*, volume 2543 of *Lecture Notes in Computer Science*, pages 5–22. Springer-Verlag, Berlin, 2003.

[5] F. Casati, S. Ceri, S. Paraboschi, and G. Pozzi. Specification and Implementation of Exceptions in Workflow Management Systems. *ACM Transations on Database Systems*, 24(3):405–451, 1999.

[6] D. Chiu, Q. Li, and K. Karlapalem. A Meta Modeling Approach to Workflow Management Systems Supporting Exception Handling. *Information Systems*, 24(2):159–184, 1999.

[7] CPN Group, University of Aarhus, Denmark. CPN Tools Home Page. http://wiki.daimi.au.dk/cpntools/.

[8] J. Eder, E. Panagos, and M. Rabinovich. Time Constraints in Workflow Systems. In M. Jarke and A. Oberweis, editors, *Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE '99)*, volume 1626 of *Lecture Notes in Computer Science*, pages 286–300. Springer-Verlag, Berlin, 1999.

[9] Gartner. Gartner's Application Development and Maintenance Research Note M-16-8153, The BPA Market Catches another Major Updraft. http://www.gartner.com, 2002.

[10] D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.

[11] J.M.P. Gevers, W. van Eerde, and C.G. Rutte. Time pressure, potency, and progress in project groups. *European Journal of Work and Organizational Psychology*, 10(2):205–221, 2001.

[12] D. Grigori, F. Casati, U. Dayal, and M.C. Shan. Improving Business Process Quality through Exception Understanding, Prediction, and Prevention. In P. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. Snodgrass, editors, *Proceedings of 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 159–168. Morgan Kaufmann, 2001.

[13] C. Hagen and G. Alonso. Flexible Exception Handling in the OPERA Process Support System. In *International Conference on Distributed Computing Systems*, pages 526–533, 1998.

[14] S.Y Hwang and J.Tang. Consulting Past Exceptions to Facilitate Workflow Exception Handling. *Decision Support Systems*, 37(1):49–69, 2004.

[15] A. Iordachescu. *FlowConnect: Process Timing and Distribution Concepts.* Shared Web Services, Ultimo, NSW, Australia, 2004.

[16] S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation.* International Thomson Computer Press, London, UK, 1996.

[17] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1.* EATCS monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1997.

[18] G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.

[19] M. Klein and C. Dellarocas. A Knowledge-Based Approach to Handling Exceptions in Workflow Systems. *Journal of Computer-Supported Collaborative Work*, 9("3-4"):399–412, 2000.

[20] P. Lawrence, editor. *Workflow Handbook 1997, Workflow Management Coalition*. John Wiley and Sons, New York, 1997.

[21] Z. Luo, A. Sheth, K. Kochut, and J. Miller. Exception Handling in Workflow Systems. *Applied Intelligence*, 13(2):125–147, 2000.

[22] E. Panagos and M. Rabinovich. Escalations in workflow management systems. In *Proceedings of the workshop on Databases: Active and Real Time (DART-96)*, pages 25–28. ACM Press, 1997.

[23] E. Panagos and M. Rabinovich. Reducing Escalation-Related Costs in WFMSs. In *Workflow Management Systems and Interoperability*, pages 107–127. Springer-Verlag, Berlin, 1998.

[24] H.A. Reijers and W.M.P. van der Aalst. Short-Term Simulation: Bridging the Gap between Operational Control and Strategic Decision Making. In M.H. Hamza, editor, *Proceedings of the IASTED International Conference on Modelling and Simulation*, pages 417–421. IASTED/Acta Press, Anaheim, USA, 1999.

[25] H. Saastamoinen and G.M. White. On handling exceptions. In *Proceedings of the ACM Conference on Organizational computing systems*, pages 302–310. ACM Press, 1995.

[26] A.W. Scheer. *ARIS: Business Process Modelling*. Springer-Verlag, Berlin, 2000.

[27] A. Seers and S. Woodruff. Temporal pacing in task forces: Group development or deadline pressure. *Journal of Management*, 23:169–187, 1997.

[28] Staffware. *Staffware Process Suite Version 2 – White Paper*. Staffware PLC, Maidenhead, UK, 2003.

[29] D.M. Strong and S.M. Miller. Exceptions and exception handling in computerized information processes. *ACM Transactions on Information Systems*, 13(2):206–233, 1995.

[30] M. Weske, W.M.P. van der Aalst, and H.M.W. Verbeek. Advances in Business Process Management. *Data and Knowledge Engineering*, 50(1):1–8, 2004.

[31] R.M. Yerkes and J.D. Dodson. The relation of strength of stimulus to rapidity of habit-formation. *Journal of Comparative Neurology and Psychology*, 18:459–482, 1908.