UDC 007:658:004, DOI:10.2298/CSIS090608006M

Concept of the Exception Handling System for Manufacturing Business Processes

Dragan Mišić[1], Dragan Domazet[2], Miroslav Trajanović[1], Miodrag Manić[1] and Milan Zdravković[1]

[1] Faculty of Mechanical Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia
draganm@masfak.ni.ac.rs,traja@masfak.ni.ac.rs, mmanic@masfak.ni.ac.rs,milan.zdravkovic@gmail.com
[2] Faculty of Information Technology, Tadeuša Košćuška 63, 11000 Beograd, Serbia
dragan.domazet@fit.edu.rs

Abstract: Business processes managed by information systems rarely operate according to the pre defined scenario. Exceptions to the pre defined workflows occur frequently. This especially applies to the production processes, which are very complex and require a constant human involvement. Workflow management systems should be capable of responding adequately to the exceptions caused by the process environment. Moreover, the response should be automatic, if possible, i.e. the workflow should automatically adapt to the new situation, or otherwise, the system administrator should be informed, so that he could take appropriate actions. This paper presents workflow management system MD, which is capable of offering a satisfying solution to the detected exceptions.

Keywords: Workflow Management Systems; Exception Handling.

1. Introduction

Workflow can be defined as the automation of a business process, as a whole or in parts, which passes documents, information or tasks from one participant to another, according to a set of procedural rules.

Workflows are managed by Workflow Management Systems. Workflow Management System (WfMS) is the system that defines, creates and manages the execution of workflows through the use of software running on one or more workflow engines, which are capable of interpreting the process definition, interacting with workflow participants and, when required, capable of invoking the use of other IT tools and applications [25].

Developing a workflow definition which completely encompasses all the variations and features of a business process is very difficult, sometimes even impossible, because of unpredictable events. Workflow definitions are

usually developed on a higher, conceptual level. Since the processes defined in such a way are executed in real changing environment, situations in which the execution deviates from the basic business case occur very frequently.

Workflow model, which has been defined in advance, represents the basic, standard business case. Workflow management system should be capable of responding to exceptions, which, in fact, are deviations from the standard (designed) business case.

Some exceptions are predictable since they are known to appear periodically. The solution to that kind of exceptions is known at the moment of developing a business process definition. Other exceptions are unpredictable and they occur due to the unpredictable changes in working environment.

Much research [5, 9, 4, 14, 3] has focused on handling exceptions in workflow management systems. Research has mostly been done in handling predictable exceptions. In order to make the basic process logic clear, exceptions aren't usually integrated into the standard business process definition. Therefore, many different ways of defining and handling those exceptions separately have been suggested.

One of the flaws of existing exception handling systems is the result of exception processing. Actions taken in that case are very rudimentary, and are usually reduced to informing users, exception ignoring, making new attempts to execute the same activity or skipping the activity. Actions leading to the change of the business process definition, such as inserting new activities or deleting the old ones, are rarely supported. Moreover, handling exceptions usually refers to handling run – time exceptions. The consequences that an exception might have on future activities are very rarely taken into consideration [15].

The existing workflow management systems are usually applied in well organized environments, such as banks, insurance companies, hospitals, etc. The advantages of such environments are well defined processes through which usually flow documents presented in a digital form.

Workflow management systems are very rarely used in manufacturing environments. Exceptions are known to occur almost regularly in manufacturing environments (material is not in accordance with specification, machine is out of order, a worker hasn't come to work etc.). According to the workflow definition, the production of a mechanical part is closely connected with the occurrences of various unpredictable situations, which are dealt with on the basis of the engineers' experience. Such processes are very suitable for applying the artificial intelligence tools and expert system, which can naturally model the expert knowledge of the area.

This paper presents the concept of MD WfMS, which is being developed in Faculty of Mechanical Engineering in Nish. This workflow management system is intended for use in production environment. The system is capable of handling the exceptions in a workflow. The system core is the workflow engine. The part which refers to handling exceptions is managed by expert system whose rules are invoked at the moment of exception detection. The

work of expert system can result in the modification of either the workflow definition or workflow instance definition.

The old definition will be changed if it has been concluded that it is no longer suitable for the new situation or, in other words, if all the future instances based on it need to be changed. An example of such a situation is when it has been decided that a part of an assembly should be procured from a business partner rather than produced in the factory.

If the exception is only temporary, then for the sake of overcoming it, the process instance rather than the process definition is changed. An example of such a case might be the situation when a casting needs repair, or when a very important machine is out of order so that it is necessary to produce temporarily that part using outsourcing.

The system, in certain situations, is capable of responding to exceptions in advance, i.e. of performing necessary adjustments before the activity in which the exception will occur is taken on.

An example of an exception of this kind might be the case when the machine that is going to be used in future activities, and which is otherwise irreplaceable, is out of order. We shouldn't wait for the activity to come. On the contrary, we should take steps to adapt the system in advance. In MD system, the procedure for handling the exception starts immediately after the exception has been detected (or has been concluded that the machine is out of order), although there is enough time left until the actual occurrence of the exception.

2. Exception Handling Problems and Related Work

An exception in the workflow can be defined as an event which prevents an activity from its being executed properly.

Different workflow management systems handle the exceptions in different ways. In general, methods for exception handling can be divided into:

- methods which have the predictable exceptions built in the process definition
- methods which handle the exceptions by the well structured system

The first method, which is capable of handling exceptions that can be foreseen in advance, is based on the fact that the person who is in charge of a process definition modifies it according to the exceptional situations known at workflow build time. If, for example, an exception occurs during a process, the process control flow is diverted to the attached activities that control the new situation. This approach is good in cases with small number of possible exceptions. A direct specification of many exceptions leads to the complexity of a process model. The question is whether these cases can be considered exception handling processes since the exceptions are built in the process definition. In that case, it is more apt to call these processes well-structured and totally pre defined workflow processes. This kind of approach is used in

the WAMO [5, 6] system. For exception handling Sagas [8] and flexible transactions are used. WfMS Opera [9] for exception handling uses programming language primitives (mechanism is similar to exception handling in object oriented programming languages, as Java).

Handling exceptions according to pre defined rules requires the use of a rule-based language. This language is used for exception specification. The rules are usually in the form of ECA (event condition action) rules. The event part defines exception symptoms. The condition part compares these symptoms with the real situation, while the action part defines what it is necessary to do in that case. The existing exception handling systems usually use some special rule based languages, designed specially to satisfy the needs of a particular system. For example, WfMS system WIDE [3] uses language Chimera-Exc. ADOME [4] uses similar approach for its language. METEOR [14] uses the so called "Justified ECA rules (JECA)" where justified refers to defining a special context in which ECA rules for exception handling are applied.

On the level of active rules application, research has been carried out in temporal ECA rules. For example, Active TFL is used for defining rules in Agent Work [18] system. Active rules are applied in assembling e-service [1] (these rules are applied to XML documents). In [20] temporal ECA rules are used for defining E- business software architecture.

Some researchers are trying to apply the previous experiences in handling the similar exceptions [22, 11]. In that case, it is the most natural to use Case-Based Reasoning. One of the problems when using Case-Based Reasoning is how to determine the degree of similarity between the old cases and the new ones [2].

Based on previous analysis it can be concluded that the majority of existing workflow management systems use special languages for exception handling. These languages have been designed to satisfy the needs of particular systems. On the other hand, there are several organizations which deal with the process management system, the best known of which is Workflow Management Coalition (WfMC). This coalition has designed a special language for the definition and specification of a workflow process– XPDL [23] (Xml Process Definition Language). This language defines a Meta model that is used for defining types of constituent components of a process definition. However, XPDL does not contain all elements needed for exception definition.

MD WfMS, uses extended XPDL as the basic language for process definition. The term extended refers to extensions i.e. language constructions which are capable of exception handling. Those constructions are inserted into the standard XPDL. For the core of the system, open source system Shark [7], which had been developed in Java programming language, was chosen. Shark was used for the basic part of this WfMS system i.e. for execution engine. This system was chosen since the process is defined in XPDL which had been recommended by WfMC coalition and which is standard in the area of modeling workflows. The system also uses OMG

(ObjectManagementGroup) recommendations regarding the interface which workflow management system should implement.

The extended version of the basic XPDL, however, encompasses the constructions which are capable of handling exceptions. The extensions are predominantly related to exception detection. After the exception has been detected, the whole process definition is being sent to the expert system whose task is to propose a solution to the detected problem. That solution may be:

- a new changed process definition
- the modification of the process instance or
- the modification of the particular activity.

The process definition is modified when it is necessary to make essential changes, not the temporary ones. An example of such a situation is when the organization decides to procure a part of an assembly from a supplier, and intends to do so in the future, instead of producing the part itself.

The process instance is modified when an exception occurs as a result of sudden circumstances, which do not affect other instances derived from the same process definition. An example of this type which describes the functioning of MD system will be given later in this paper.

Modifying the activity actually means modifying the data of that activity itself. In essence, this kind of modification is the subtype of the process instance modification since neither the new activities are added nor the old ones are deleted. Therefore, the workflow remains unchanged.

The existing workflow management systems are not usually capable of handling the exceptions properly. The primary goal of such systems is defining and managing processes that flow without any exceptions. If the exception handling mechanism is the constituent part of the system, then it means that the whole system has originally been designed to be capable of handling the exceptions. MD system is an example of how a system, which has not been designed to respond to exceptions, can be extended and linked with another system, in this case the expert system in order to handle the exceptions successfully.

3. System Architecture

In MD WfMS (figure 1), workflows are defined by a process manager, i.e. the manager who is responsible for planning and managing the process. Workflows are defined by a person who is in charge of them, usually a business process manager i.e. the manager who is responsible for planning and managing the process. According to that definition, the system administrator with the assistance of an editor enters the process model. If the structure of the organization allows that, it is possible for the process manager to enter the process definitions himself. The definition is entered by means of graphic process editor. Workflow definition doesn't contain the activities that should be carried out in case of unpredictable exceptions.

Therefore, the workflow definition is not loaded with numerous additional activities, which will be carried out in case of an unpredictable situation.

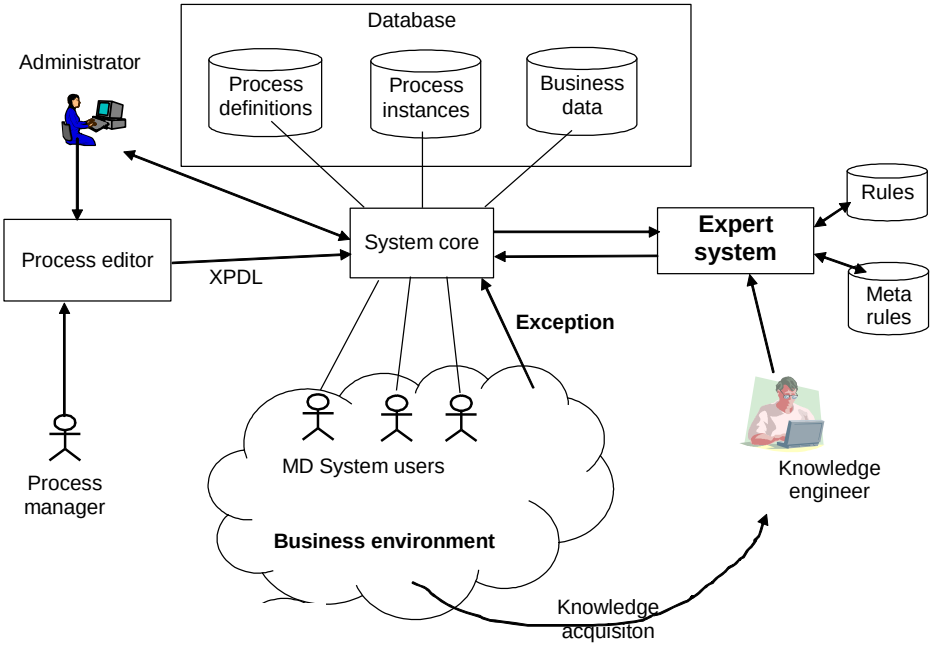

Fig. 1. MD WfMS architecture.

4. Exception Types, detection and handling

4.1. Exception types

According to the possibility of detection in the MD system, we divided the exceptions as following:

- exceptions related to data
- exceptions related to time (violation of deadlines)
- exceptions related to resources
- exceptions signalized by people.

Exceptions related to data are exceptions arising from data values that participate in the processes.

These exceptions can be further divided into complex and simple. Complex exceptions related to data are exceptions that can only be detected tracking several values.

Simple exceptions are the exceptions that can be detected based on only one value. An example is when the value of some parameter does not lie within a previously determined range.

Exceptions related to time are the exceptions that can be detected by measuring time. If the definition of process defines a specific time when certain activity must be done, measuring time enables checking whether those deadlines are met, which can be considered as an exception.

Depending on the type of time measurements used, these exceptions can be divided as following:

- Exceptions based on exceeding the duration of an activity
- Exceptions based on exceeding the deadline of an activity
- Exceptions based on exceeding the waiting time for execution of an activity
- Exceptions based on exceeding the duration of the whole process, or a group of activities
- Exceptions based on exceeding the deadline of the process, or a group of activities

Exceptions based on resources are probably the exceptions that most frequently appear in a workflow. Resources are material resources (machines and equipment necessary for some activities) and human resources.

Exceptions signaled by people: Beside various automatic ways of exception detection, the system will often report situations when it is not able to automatically detect the exception. In such situations human intervention is needed.

4.2. Exception Detection

One of the problems which arise during the application of the exception handling system is how to detect an exception. MD system can detect data exceptions, time limit exceptions with regard to exceeding time limitations, and resource exceptions.

Exceptions related to data can be detected in two ways. Simple exceptions are detected by checking the value of particular data after they have been entered by a system user or calculated by the system.

Complex exceptions related to data are the ones that are the most difficult to detect. These exceptions are the most general type of exceptions, since values and presence of different data and information can in certain situations represent the exception, in some others not, which is very hard to find out. MD system uses ASM [19] (Active Semantic Model) for the detection of these exceptions.

Linking the ASM system with MD system, as well as the way of exception detection, is described in detail in [16]. At this point, we will just mention that at the moment of entering the new information in the system (which does not exist in the model), a conclusion mechanism of ASM [27] (based on artificial intelligence), which is capable of classifying a new situation as an exception, is activated. Therefore, if a certain situation has been classified as an exception, data are sent to the expert system which afterwards makes a decision about the actions that should be taken.

Time exceptions are detected by measuring activity duration, or measuring the time that has passed until the available activity is taken on. Temporal ECA rules might have been used for the description of exceptions related to time, but, since the events whose occurrence is monitored here are quite simple (exceeding the defined duration), there has been no need to define complex conditions related to time intervals. These rules might be included later in further work on the development of this system.

Time limitations that should be monitored are defined by, Limit and Deadline, the elements of XPDL scheme. SimulationInformation is another element within the scope of XPDL scheme which stores information related to the simulation of workflows that are being monitored. According to the original plan, such information should be used by the module for the simulation of the workflow process. Since Shark does not contain this kind of module, SimulationInformation element was used for defining the estimated duration of the activity (element Duration) and estimated waiting time for the activity to be taken on by human resources. These two elements, Duration and Waiting Time, are a part of Time Estimation element. XPDL definitions of these elements are:

```
<xsd:element name="SimulationInformation">
  <xsd:complexType>
   <xsd:sequence>
     <xsd:element ref="xpdl:Cost"/>
     <xsd:element ref="xpdl:TimeEstimation"/>
   </xsd:sequence>
   <xsd:attribute name="Instantiation">
    <xsd:simpleType>
      <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="ONCE"/>
        <xsd:enumeration value="MULTIPLE"/>
      </xsd:restriction>
    </xsd:simpleType>
   </xsd:attribute>
  </xsd:complexType>
</xsd:element>

<xsd:element name="TimeEstimation">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="xpdl:WaitingTime" minOccurs="0"/>
    <xsd:element ref="xpdl:WorkingTime" minOccurs="0"/>
```

```
      <xsd:element ref="xpdl:Duration" minOccurs="0"/>
     </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
```

Previously mentioned time limitations are monitored by means of time management module which is the constituent part of MD system. This module functions on the basis of checking violations of time limitations at regular time intervals. A time interval is adjustable.

Besides the information entered at the modeling time of the workflow, MD system uses the information obtained during the execution of the workflow process. Execution time and the acceptance time are monitored for each activity and the data are then stored in the database. At the moment of checking whether an activity has violated the predefined time limit, besides the time defined in the model, the times previously stored in the database (real times) are used too. The mean value of the times from the database is compared with current time.

In case the anticipated time has passed, it is considered that an exception has arisen.

Exceptions related to resources are detected by means of a special resource management module, which is linked with MD system. In case that a resource is no longer available, the resource management module immediately sends an appropriate message to MD system. MD system will automatically check whether that resource is used by any activities in the current process instances. If there is such an activity, it is considered that an exception occurs and the mechanism for its handling is activated. Resources in the process are defined by means of MdResource element, which is integrated in XPDL scheme. This element is a part of Activity element. XPDL definitions of added elements are:

```
<xsd:element name="Activity">
 <xsd:complexType>
    <xsd:sequence>
             ...
          <xsd:element ref="xpdl:MdResources"
                                    minOccurs="0"/>
    </xsd:sequence>
           ...
 </xsd:complexType>
</xsd:element>

<xsd:element name="MdResources">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="xpdl:MdResource" minOccurs="0"
                        maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="MdResource">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="xpdl:Description" minOccurs="0"/>
   <xsd:element ref="xpdl:MdResourceAttributes"
               minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:NMTOKEN"
                 use="required"/>
  <xsd:attribute name="Name" type="xsd:string"/>
 </xsd:complexType>
</xsd:element>
```

4.3. Exception Handling

XPDL definition, which has been created by graphical process editor, is sent to the system core (execution engine). The system core is responsible for creating process instance (based on the process definition), and also for the process execution i.e. forwarding activities to corresponding resources.

As soon as an exception is detected, both the process definition and the exception cause are sent to the expert system. We use expert system created via JESS expert system shell [12]. This is the Java rule based system, created in Sandia National Laboratories, from Livermore, California.

In the expert system Meta rules come first.

WfMS system can administer processes from different application domains, such as government, hospitals or manufacturing. Since the domains can be diverse, the predefined rules for solving the run time exceptions needn't be the constituent parts of the expert system. According to that, there are Meta rules which are to decide whether there is the domain knowledge for a particular exceptional case or not.

If an exception related to data is in question, it can happen that a system does not contain enough data for the expert system to draw a conclusion. In such cases the user is granted the right to enter additional data. In such situations, the system user is asked certain questions by defined domain rules. Answering the questions, the user provides the expert system with information necessary for drawing a conclusion. Since the questions are domain specific, certain rules are invoked only if the domain they are defined for is in question.

Considering the casting machining, there are certain rules which are capable of handling exceptions which occur due to the irregularities in casting structure. Therefore, the first question that is posed is whether it is the irregularity of the casting structure. If that is the case, the further questions which enable a more detailed exception description are posed. On the basis of the answers obtained, the expert system is capable of solving the problem.

If an exception has been detected by means of a control parameter, it may become necessary that a human participant enters some additional data, which provides a more detailed description of a particular exception. Therefore, certain rules are developed for asking process specific questions that a human participant has to answer.

Since the expert system is linked with WfMS, the human participant is not even aware of this internal communication. The user is asked questions only if the expert system does not contain enough information to draw a conclusion. If the expert system contains rules for solving a certain problem, then the facts necessary for drawing conclusions are known. The questioning mechanism is devised to enable entering such facts.

The expert system encompasses two defined classes (defftemplate): question (representing questions which the user is asked), and answer (for user's answers).

```
(deftemplate question
          (slot type)
          (slot text)
          (slot ident)
          (multislot valid)
)

(deftemplate answer
          (slot ident)
          (slot text)
)
```

Both the set of posed questions and the set of offered answers depend on the problem that is being solved. For example, in order to demonstrate the application of MD system (pump making), the following questions out of the set of questions were posed:

```
(defrule is-a-hole
=>
(assert (question (type yes-no) (text "Is it a
cavity?") (ident cavity) (valid yes no)))
)
(defrule porousness
          (answer (ident cavity) (text ?d))
=>
          (if (eq ?d yes) then
            (assert (question (type multi) (text "How
large is porousness?") (ident porous) (valid large
medium small)))
          else
            (assert (answer (ident no-solution) (text
no-solution)))
          )
)

(defrule load-pump-rules
```

```
            (answer (ident cavity) (text yes))
            (answer (ident porousness) (text ?por))
  =>
            (clear)
            (batch pump_rules.clp)
  )
```

These questions are posed during the cast processing if a user detects a hole which can't be repaired by further lathe processing.

If the expert system has the domain knowledge for this kind of exception, then the whole process definition (Java objects) is sent to it. These objects are recorded into the facts base of the expert system. In addition to the process definition, expert system receives the data related to the specific process instance within which the exception occurred, as well as the exception description, no matter whether it was generated automatically or by a user.

The expert system offers two types of solutions: the first type of solutions refers to the specific process instance and the second type of solutions refers to process definition.

Disregarding the solution type, the expert system creates a new process definition. This definition, conditioned by a particular problem, may contain some new activities added by the expert system, or may delete some old ones. The fact that the exception can affect future activities within the same workflow may be also taken into consideration. New process definition is sent back to the WfMS system. Together with the process definition, expert system forwards data defined for particular process instance.

The workflow instance, within which an exception has occurred, is in the meantime suspended, and its execution stopped. The new process instance is automatically created on the basis of the new process definition coming from the expert system. The new process instance is automatically processed to the workflow part within which the exception has occurred. From then on the execution of the workflow continues according to the new workflow definition.

If the newly created solution affects the remaining process instances (for example, the machine that is going to be used in further activities is out of order), then the remaining process instances derived from the old process definition are modified too. This happens only if the particular process instance hasn't come yet to the activity within which the problems are detected. If that activity has already been executed, the modification of the process definition is not necessary.

The old process definition remains the constituent part of the WfMS and may be used later. Whether the new process definition or the old one will be used depends both on the type of exception and the expert system conclusions. These conclusions are stored in the data base. The data base also contains the new process definition and the old one, information whether the changes are to be applied only to the current process instance, all the new instances or to the instances whose execution has not come to a

particular activity yet. The transfer of the information from the expert system to MD system is performed by means of a special class which contains previously mentioned data.

New rules for solving some new problems can be defined within the expert system at any time. New rules can be added to the rule database independently from the WfMS because they are not directly related to the system. Consequently, the WFMS can be applied in new domains and the knowledge about the predictable problems within the system can be modified and adjusted.

The set of solutions which the expert system proposes will grow in time in accordance with the extension of the knowledge domain. In spite of that, new exceptions, for which the solution does not exist, will arise. In that case, WfMS allows the system user to offer a solution himself.

If the problem for which the Expert System doesn't have the solution occurs, the graphic process editor is opened and the actual process definition is automatically loaded. In that case, the user is granted the right to modify the process in order to find a solution to the problem. The procedure after the new process has been defined in this way is similar to the one after the solution has been offered by an Expert system. The new process definition is sent back to the WfMS, which automatically applies new process instance (derived from the new process definition), and the data from the earlier process instance are still present.

Let's take a look at the example of processing mechanical part of great dimensions. Let's assume that there is only one machine available in the factory for processing such a big part. If that machine were out of order, an exception in the process would occur since the activity of lathe processing couldn't be performed. In that case solutions can be different, ranging from waiting for the mechanical part to be repaired, to sending the part to a business partner to process it. If there aren't rules in the system which will handle this exception, the user can choose the option of opening the editor with process definition. Then the user, at his own discretion, can change the process. Since the process instance that is being executed is in question, the activity that is being performed is marked in the open process. It is up to the user to adjust the process to new circumstances. In this case it means that the new activities related to processing the part by a business partner will be added. After he has devised a new process definition, the user should also define the data related to the application of the new process definition. These data are the ones that the expert system issues at the moment of automatic problem solution (whether the new definition applies to all the instances or just to the current one).

5. An example

As an example we will observe pump making process in the pump factory. The simplified schema of the whole process is shown in the figure 2.

The pump impeller is usually made from the casting, which is afterwards processed on cutting machine tools until the final form is obtained. A problem in pump making process can occur due to the inappropriate casting process, which may result in obtaining the casting with cavities. Manufacturing the pump from such a casting cannot be continued until some corrections have been made. Corrections (if it is possible for the part to be corrected) are made by filling the cavities by welding.

Visual irregularity detection in the casting structure is not usually possible until the process comes to cutting. Only after the removing of material layers has started, the irregularities can be detected. In that case the cutting process stops.

Depending on the material porosity, an engineer makes a decision whether the part will be overhauled or declared as waste.

The following activities can be used for overhauling the part: inserting a new element, welding, filling the cavities with glass water or multimetal.

The decision regarding which of the aforementioned activities will be used depends on the cavities dimension and conditions of utilization.

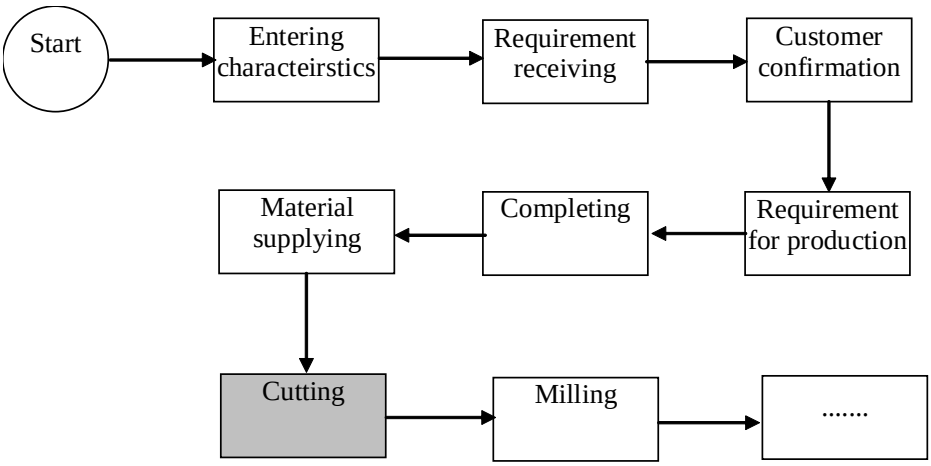

Fig. 2. Example of a pump making process.

Welding process is preceded by digging the material i.e. broadening the hole. This means that two new activities, digging and welding, are added before lathe process. Therefore, the original process is extended with two new activities. After these two activities, the execution of the workflow continues according to the previous plan (figure 3).

Consequently, the XPDL process definition is modified and sent back to the system core which continues the execution of the process according to the new process definition. The new process definition is applied only in that particular instance without affecting other instances of the same process.

The following example refers to exception detection using the control parameter whose value is entered by a user. After the exception has been detected, the questioning mechanism, enabling a more detailed description of the situation, is activated. According to that, in the cavity case, the user first marks that the cavity has been detected, and then he is questioned about its dimension. On the basis of the information obtained, the new rules, which decide the way the problem will be solved, are applied. The modified process definition is sent back to WfMS according to the scheme which has already been presented.

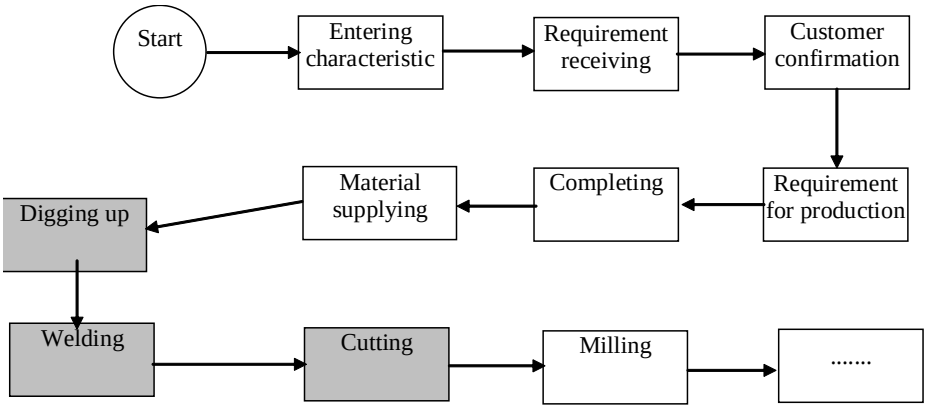

Fig. 3. Changed pump making process.

Knowledge base for resolving this problem is:

```
(defrule welding1 "porousness corrects by welding”
 ?f <- (answer (ident cavity) (text yes)); entered by
user
(answer (ident porousness) (text large)); entered by
user
(answer (ident place) (text accessible));entered by
user
   (Process-data (fluid-type ?type) (temperature ?t))
    (test (or (and (eq ?type water) (> ?t 120)) (eq
?type acid)))
=>
          (assert (action welding))
          (assert (place before current))
   (retract ?f)
)

(defrule welding2 " porousness corrects by welding"
 ?f <-    (answer (ident cavity) (text yes)) ;entered by
user
```

```
          (answer (ident porousness) (text large))
;entered by user
          (answer (ident place) (text accessible))
;entered by user
          (answer (ident part-type) (text rotational))
;entered by user
=>
          (assert (action welding))
          (assert (place before current))
        (retract ?f)
)
(defrule navarivanje3 "porousness corrects by welding"
 ?f <-    (answer (ident cavity) (text yes)) ;entered by
user
          (answer (ident porousness) (text large))
;entered by user
          (answer (ident place) (text accessible))
;entered by user
          (answer (ident stress) (text dynamical))
;entered by user
=>
          (assert (action welding))
          (assert (place before current))
        (retract ?f)
)
(defrule reject1
  (answer (ident cavity) (text yes))
  (answer (ident porousness) (text large))
  (answer (ident stress) (text dynamical))
  (answer (ident place) (text inaccessible))
=>
  (assert (answer (ident reject) (text part-rejected)))
)
(defrule welding-change-process
  (action welding)
  (place before current);place for inserting new
activities.
  (current-activity ?x); current activity id
  (process-definition-id ?processDefId);process Id from
XPDL definition
=>
  (bind ?prevAct (get-previous-activities ?x));list of
previous activities
  (bind ?packageDefinition (fetch PackageDefinition));
XPDL package definition
  (bind ?activities (get-activities-from-process
?processDefId)); all process activities (list of Java
classes(
  (bind ?process (get-process-from-definition
?processDefId)); Java object of process
```

```
  (bind ?transitions (call ?process get
"Transitions")); list of Java objects with transitions
between acitivities
  (bind ?currentAct (call ?activities getActivity (get-
activity-definition-id ?x))); Java object with current
acitivity
  (bind ?prevActObject (call ?activities getActivity
?prevAct)) ;Java object with previous activity
  (bind ?b1 (call org.enhydra.jawe.MisicProba
createActivity "raskopavanje" "raskopavanje"
    ?activities ?process "Opis" 1 ""
"FreeTextExpressionParticipant" 1  1 "" "" ""
nil));Java object for new activity
  (bind ?b2 (call org.enhydra.jawe.MisicProba
createActivity "navarivanje" "navarivanje"
    ?activities ?process "Opis" 1 ""
"FreeTextExpressionParticipant" 1  1 "" "" "" nil))
;Java object for new activity
  (call ?activities add ?b1)
  (call ?activities add ?b2)
  (bind ?tNew1 (call Utility createTransition
?transitions ?prevActObject ?b1)); new transition
  (bind ?tNew2 (call Utility createTransition
?transitions ?b1 ?b2)) ; new transition
  (bind ?tNew3 (call Utility createTransition
?transitions ?b2 ?currentAct)) ; new transition
  (call ?transitions add ?tNew1); adding new transition
to list of all transitions
  (call ?transitions add ?tNew2)
  (call ?transitions add ?tNew3)
  (call Utility removeTransition ?prevActObject
?currentAct ?transitions); removing unnecessary
transition
(bind ?strategy (call org.enhydra.jawe.MisicProba
createStrategyClass ?processDefId ?process
?currentProcess ?currentActivity "change"
"CURRENT_INSTANCE_ONLY")); creating an object of class
;strategy with information about using new process
;instance. Attributes are: old XPDL process definition,
;new XPDL process definition, Java object with current
;process instance, java object with current activity
;instance, change type, and fact that new process
;definition should apply to current instance only
)
```

6. Conclusion

Today's workflow management systems are not so robust because they lack the capability to handle exceptions which lead to the deviation of the workflow from the pre defined workflow model. If a workflow management system is capable of handling the exceptions, it means that the exceptions have become the constituent part of it at build time of workflow definition. Such systems use special process defining languages which have the constructions for exceptions handling built in.

MD system, which is presented in this paper, treats the exception handling a little differently. We have tried to extend the existing WfMS without changing it radically. Therefore, we have linked the system core to the separate expert shell capable of defining rules for handling exceptions. Any system can be extended in similar way.

Now it is possible to define exceptions independently from the main workflow. Consequently, the workflow is no longer overloaded with redundant information. Furthermore, since WfMS can be applied to different environments, it has become easy to add new rules for handling the exceptions under new conditions, and this contributes to scalability of the system. On the other hand, the practical application of the WfMS will lead to the extension of the existing exception handling knowledge. The proposed workflow model makes the extension of the existing system possible without affecting the WfMS core. In that way, the system acquires modularity which enables it to be applicable in various domains. When applying the WfMS in a completely new domain, it is necessary only to define the knowledge base of the Expert system. Modification of the knowledge domain does not affect the system core.

The exception handling system can resolve controversial situations in different ways. Depending on its expert knowledge, the WfMS can be extended in such a way that it becomes capable of handling various exceptions. Therefore, some new activities can be added, or the existing ones modified or deleted. The thorough analysis of the workflow enables us not only to predict the influence the exception will have on future activities, but also to make proper corrective actions in advance.

Acknowledgment

The paper is part of project TR12010 – “Active product semantic data model” that is sponsored by Ministry of Science and Technology for the period of 2008-2010.

References

1. Bonifati, A., Ceri, S., Paraboschi, S.: Active rules for XML: A new paradigm for E-services. The VLDB Journal 10, 1, 39-47 (2001)
2. Burkhard, H.D.,: Case Completion and Similarity in Case-Based Reasoning, International Journal, Computer Science and Information Systems, Volume 1, Number 2. (2004)
3. Casati, F.,: Specification and Implementation of Exceptions in Workflow Management Systems. ACM Transactions on Database Systems, vol.24, no.3, pp.405–451. (1999)
4. Chiu, D. K. W., Li, Q., Karlapalem, K.,: ADOME-WFMS: Towards Cooperative Handling of Workflow Exceptions. ECOOPWorkshop 2000: Advances in Exception Handling Techniques, Lecture Notes in Computer Science, vol.2022, pp.271–288. (2001)
5. Eder, J., Liebhart, W.,: Contributions to Exception Handling in Workflow Management. EDBT Workshop on Workflow Management Systems. Valencia, Spain. (1998)
6. Eder, J., Liebhart, W.,: The workflow activity model WAMO. In Proceedings of the International Conference on Cooperative Information Systems, Vienna, Austria. (1995)
7. Enhydra Shark: Open Source Java/XML Workflow Engine, http://shark.enhydra.org/
8. Garcia-Molina, H., Salem, K.,:"Sagas". Proc. ACM SIGMOD (1987).
9. Hagen, C., Alonso, G.,: Exception Handling in Workflow Management Systems. IEEE Transactions on Software Engineering, vol.26, no.10, pp.943–958. (2000)
10. Hwang, G.H., Lee, Y.C., Wu, B.Y.,: A New Language to Support Flexible Failure Recovery for Workflow Management Systems, Springer-Verlag Berlin Heidelberg. (2003)
11. Hwang, S., Tang, J.: Consulting past exceptions to facilitate workflow exception handling. In: Decision Support Systems 37, 1, pp. 49-69. (2004)
12. JESS, the Rule Engine for the JavaTM Platform, Sandia National Laboratories, http://herzberg.ca.sandia.gov/jess/.
13. Kumar, A., Wainer,J., Zhang, Z.,: Meta-workflows and ESP: A Framework for Coordination, Exception Handling and Adaptability in Workflow Systems. Springer-Verlag Berlin Heidelberg. (2004)
14. Luo, Z., Sheth, A., Kochut, K., Miller, J.,: Exception Handling in Workflow Systems. Applied Intelligence: the International Journal of AI, Neural Networks, and Complex Problem-Solving Technologies, vol.13, no.2, pp.125–147. (2000)
15. Miler, R.: Event-Oriented Dynamic Adaptation of Workflows: Model, Architecture, and Implementation, dissertation, University of Leipzig. (2002)
16. Mišić D., Stojković M., Domazet D., Trajanović M., Manić M. and Trifunović M.: Exception detection in business process management systems, Journal of Scientific and Industrial Research, in press.
17. Muehlen, M.: Workflow-based Process Controlling: Foundation, Design and Application of workflow-driven Process Information Systems. Logos, Berlin. (2004)
18. Müller, R., Greiner, U., and Rahm, E.: AGENT WORK: a workflow system supporting rule-based workflow adaptation. Data Knowl. Eng. 51, 2, 223-256. (2004)

19. Stojkovic, M., Manic, M., Trajanovic, M., Korunovic, N.: Semantic Structures In The Product Data Model. Proceedings of International Conference on Product Lifecycle Management, PLM-SP3, 227-234. (2007)
20. Wan, H., Li, L.:. E-business software architecture based on temporal ECA rules and actions conflicts management. Int.Conf on E-Business Engineering (ICEBE), Beijing China, pp 208-211. (2005)
21. Weber, B., Rinderle, S., Reichert, M.: Change Support in Process-Aware Information Systems - A Pattern-Based Analysis, Tech. Rep. TR-CTIT-07-76, CTIT, University of Twente, The Netherlands. (2007)
22. Weber, B., Wild, W., Breu, R.: CBRFlow: Enabling Adaptive Workflow Management Through Conversational Case-Based Reasoning, Springer-Verlag Berlin Heidelberg. (2004)
23. WfMC (XPDL): Workflow Process Definition Interface - XML Process Definition Language. Document Status - 1.0 Final Draft. Document Number WFMC-TC-1025. Workflow Management Coalition 2002.
24. Workflow Management Coalition, The Workflow Reference Model
25. Workflow Management Coalition, Terminology and Glossary, Workflow Management Coalition 1999.
26. Wu, S., Guo, B.: Design of an exception handling algorithm of workflow. ISECS International Colloquium on Computing, Communication, Control, and Management, pp.281-284. (2008)
27. Misic, D., Stojkovic, M., Domazet, D., Trajanovic M., Manic, M., Trifunovic, M. : Exception detection in business process management systems. Journal of Scientific and Industrial Research, pp. 188-193. (march 2010)

Dragan Mišić is research and teaching assistant at Faculty of Mechanical Engineering, University of Niš. He works with workflow management and business process management systems and their application in the manufacturing business environments. He is the author of more than 30 scientific and professional papers.

Dragan S. Domazet is a full professor at the Faculty of Information Technology in Belgrade, of which he is dean and founder. He has worked on the application of object-oriented bases in the product development by applying the STEP standard and on the application of PDM (Product Data Management) systems. His areas of work and research are eLearning systems, analysis and reengineering of business processes, making feasibility studies for the introduction of new information systems, development of systems for collaboration engineering, etc

Miroslav Trajanovic, professor at Mechanical Engineering Faculty, University of Nis, Nis, Serbia, has had got 30 years of experience in application of IT in mechanical engineering and education. He is expert for computer programming, CAD, finite element method and expert systems. He is the author of more than 140 scientific and professional papers. Professor Trajanovic was also project leader for 10 projects mainly in IT and Mechanical Engineering, as well as two FP6 and two FP7 projects.

Miodrag Manić is professor at Faculty of Mechanical Engineering, University of Niš. He is head of Laboratory for Intelligent Manufacturing Systems. Science fields are using of ICT in design of product and manufacturing technology, especialy integrated CAD/CAPP/CAM and CNC programing machine tools. He is participated in eighteen scientific-research projects, he is author of 120 scientific-technical papers, and co-author of two university textbooks.

Milan Zdravković is research and teaching assistant at Faculty of Mechanical Engineering, University of Niš. His work is focused to using semantic web tools, languages and service oriented architectures for facilitating interoperability in inter-organizational collaborative forms, such as manufacturing supply chains.

Received: June 08, 2009; Accepted: March 08, 2010.