

Office Activity Procedure Exception Handling Realization Difficulties

Dovile Vojevodina, Genadijus Kulvietis

Vilnius Gediminas Technical University, Information Technology Department, Sauletekio
11, Vilnius 2040, Lithuania

d.vojevodina@erp.lt, Genadijus_Kulvietis@gama.vtu.lt

Abstract. Office activity procedure automation is getting very popular in major organizations. During the automation then workflow method is chosen frequently, because of its possibility to reflect real world business processes in a transparent and informative way. By automating such activities, organization gains a possibility to share resources between distant company departments, shortens the activity fulfillment time, and reaches work efficiency. It's a pity, but workflow doesn't support real world deviations, which often occur during document or project preparation. Every possible exception or deviation must be foreseen during the process build-time. The most difficult to solve are run-time exceptions. They are difficult to catch, their impact to the process can be determinant to the final goal and they irritate the user most of all. This paper focuses on the business process, modeled using workflow exception handling, and exception impact to the business process and user reliance on the system. The proposed decomposed workflow method shows how an office activity automation application could reach flexibility and support run-time exceptions. Combining the solutions of all mentioned above problems into a consistent solution, the exception handling could be achieved in a flexible way.

Introduction

Every organization success in recent days depends on timely received precise and genuine data, which could be reached by any remote user even in a disconnected mode [10]. The great part of automated business processes takes office activity procedure transformation into automated workflow processes. It's a pity, but these processes have a tendency to change and this directly impacts the activated business process instance.

Definition of the business process says that it is a set of one or more linked procedures or activities, which collectively realize a business objective or policy goal, normally within a context of an organizational structure defining functional roles and relations.

In other words, any disturbed activated business process instance can be directly and critically impacted so that the final result will not be reached at all. Major expected deviation or exception handling procedures are encapsulated into core

business process during the built-time, but it's a run-time exception, that usually corrupts the activated process instance.

The run-time exception nature is to occur in any place at any time. In order to solve them correctly and timely, the business process application must be built in such a way, that it could be possible to reach any of its components without major difficulties and change them if that is necessary. This paper proposes a method when an application is divided into separate, independent modules, which can function solely or in group and share common resources.

The first part of the paper introduces us to the exception occurrence context and office activity conception. Then the exceptions classes are shortly introduced and discussed. The third part introduces the proposed model conception. The fourth part focuses on the impact of the exception handling system to the business process and human reliance on the system. The paper is finished with the conclusions, which were made referring to the made research.

Office Activities

The dominating activity processes are clearly described. They have activity rules, which can be regulated by appropriate document or by achieved agreement between activity process participants. All the participants understand their duties and responsibility and can easily describe them [12].

That leads to an assumption that the first step of workflow modeling can be easily achieved in this context. The step is to model the organizational structure consisting of departments, users and roles. [13]. Finishing that, the activities ordering can be made using dedicated graph which includes every possible activity as a separate branch. The possible deviations or exceptions in this case can be represented as sub-processes to which the main process refers when needed.

Office activity process is clear activity procedure, which can be structured using workflow models and instruments. These processes can be: common office administration and management processes, separate issue execution coordination, organization layer document management, manager's accountability and usage of common resources [12].

Every office activity is assigned an executor using official instructions according to its availability. The executors are grouped by assigning those roles, which later describe the user task list. These executors communicate according to the activity process graph.

The example of the office activity procedure could be such: "There is a new user in the office activity system, he is registered in the system resource database, but wasn't assigned the user name yet. If such person tries to register the document in the system and to address it for review to someone, he receives such an exceptional situation:

- addressee was not informed about the document,
- user is disturbed by an error message,
- Workflow is stopped or even not activated.

Exceptions

A business activity, which is unable to finish and achieve the goal, is called an exception. The workflow requires the specification of both: the normal flow and the possible variations due to exceptional situations that can be anticipated and monitored [2].

Basic exception classification is very simple [3]:

- Useful exceptions – a “key to effective and flexible” processes, usually easily handled;
- Unanticipated exceptions – the result of an unexpected, infrequent and non-repetitive event.

The useful exceptions are always solved during the build-time. The dedicated process graph is added with the special branches which would be taken in the initiated process instance if need be. Those are predefined exceptions. This paper focuses on the unanticipated, in other, words, unknown exceptions, which handling can't be predefined.

The given above example case exceptions are unanticipated until they are not encountered and included into the process. Once the case is reproduced and the exception handling mechanism is included into the process model, they become useful exceptions.

Exceptions can occur in process instance synchronously with the flow or asynchronously. The specified exceptions are tended to occur synchronously to the flow and don't impact the work model and the final goal is reached though it can be influenced and slightly changed. These exceptions are called useful exceptions [3]. The unexpected and unknown exceptions usually occur asynchronously with the flow and the made damage is critical to the final goal. Such exceptions are very dangerous, because they can be spotted after a lot of damage is already done. And still the exception appearance spot will not be the place there it really was raised. So the main task is to find the right spot.

In the office activity processes the exceptions are context depended, so there is a lot of reason to specify exception using JECA (Justified Event-Condition-Action) rules. Each JECA rule $r(j, e, c, a)$ consists of four parts as follows [5]:

- Justification (j) – the context there the rule will be performed
- Event (e) – even on which the rule is triggered
- Condition (c) – logic constraints, that should be satisfied, so that the action in the rule would be taken
- Action (a) – that should be done if the rule is true.

The example exception “addressee is not informed” would look like this:

If “the user creates and addresses the document to someone“

AND “the user doesn't have the user name” then

“display a user friendly error message”

AND “inform the system administrator about the user with no user name”

AND “inform the user to resend the failed document to the addressee”

These rules allow achieving an exception handling consistency. Gather exception context information, discover the exception occurrence spot, decide if it's really an exception and solve it.

Exception Handling Scenario

Any system is as easier to support, as it is flexible. Flexibility means: easy design and change, easy enactment of changes in running workflow instances, fluent and transparent support of exception handling and failure recovery, dynamic workflow schema evolution [1].

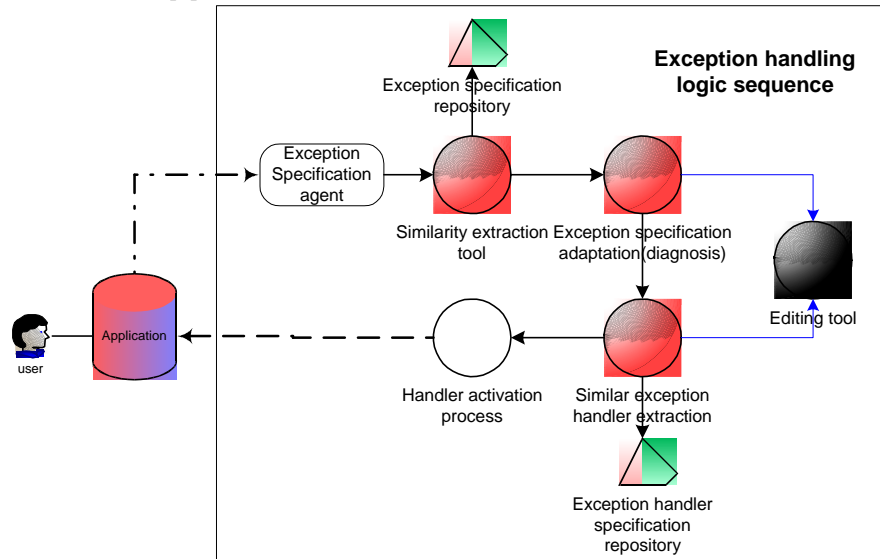


Fig 1. Exception handling logic sequence.

Discussing office activity procedure workflow exceptions, it is obvious, that the biggest work for the system is to detect possible to occur exception and if such already occurred – the occurrence spot. The system must be able to specify the exception, using JECA rule: defining context, the occurrence spot, why it occurred – the relative data, and the action, in this case - the result or damage [5]. That would be an exception detection step.

In the example case, if the document sending procedure would run constant error checking function, the exceptional case spot would be immediately reported to the system administrator and the user would not be disturbed by an unfriendly error message. In such case, we would have a friendly exception, because the exception will be solved immediately and the user will not even suspect about the exception.

The later steps depend on this step specification, so the systems must have the ability to specify this detail and timely. Also the system must be able to inform relative users about the situation and give the direct access to the specification. The user must be supplied with instruments, which should let him to extract the right handling tool and enact it in the JECA specified place.

The diagnosis step must finish with the exception handling specification, which should be directly handled to the exception-handling engine for processing. The exception handling process is finished with the handling phase.

Decomposed Workflow Exception Handling Logic Model

Different users can have different access to the different application modules. So it's necessary to create one common profile module where the separate personalized profiles would be created and stored for every user, which defines application personalization: access to different modules, main visualization and actions.

According to profiles, user gets individual environment for his requests input. Once request is filled, it is handled to business process logic, which defines the document processing sequence. Usually deviations occur between these two steps and during business process logic's processing. That means it is necessary to have an autonomic back-end exception handling application, which has access to business applications and user profile database.

Exception handling application must have a dedicated graph of all business logics represented as possible exception tree [3]. This application should have three running clients: exception detection, exception diagnostics and exception handling. The first of them must be active all the time and compare activated instance business logic execution with the exception description graph. Sentinels components can be used for that purpose, which would look for appropriate patterns in the behavior of basic components [6][7].

In the described example case, the sentinel would be a constantly running function, which compares the actual data with described patterns. The encountered deviation is immediately reported to the defined in the function addressee.

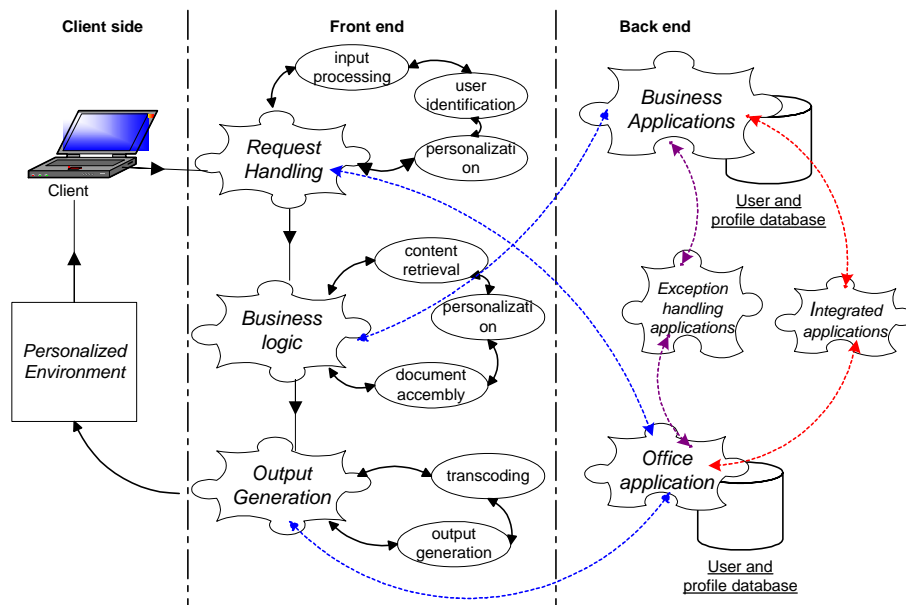


Fig 2. Logic office activity procedure cycle [9][4].

In office activity applications there is a lot of reasons to have exception ontology (referring to the [11] source, which uses taxonomy tree, but in this context ontology would be more effective), which would define appropriate input, output and processing of the common data. The sentinels could compare actual data with ontology tree. If data differs, sentinel could pass this data to the exception registration agent.

Office activity processing can involve many people with various roles, which have different task list. The exception detection sentinels should estimate that, if the ontology itself gives or links to such information: roles and task lists.

In fact, business logic description and definition module, request handling or output generation modules can have different exceptions ontology. In such way, it is possible to achieve more detail exception situation specification. Different modules could have personal sentinels and exceptions registration agents.

The Impact of Exception Handling Module To the Business Process

In the ideal model, exception-handling mechanism should be constructed as an autonomous module. The business process application should refer to exception handling module procedures or functions, which evaluates possible exceptional situations. Also, the user should not suspect or know that business process application uses exception handling application and how it works.

This model is effective and reliable but has some disadvantages:

- Every business process application element must have a reference to the exception-handling module. That slows down the applications, but gives precise data of the exceptional situation if such occurs.
- The main business process instance can be changed. If the reported exception diagnosis shows that the exception can't be reproduced or the diagnosis is indefinable, the only way to continue current process instance is to change it, so that the goal would still be achieved.
- The business process goal achievement can be late. If the business process instance has short term and is urgent, the occurred exception handling can slow down the process and the responsible for the instance person can be informed too late. That lessens the user reliance to the system.
- Exception handling mechanism is difficult to configure and support. The configuration is made one time, but requires a lot of knowledge about the exception handling and business process systems and their interaction. In case, to use exception-handling system effectively, it must be constantly renewed, that takes time and knowledge.

The main problem of every system designer is to construct the system in such a way, that the user should not suspect about the occurred exception. If exceptional situations cannot be solved without the help of the user, the systems should present the user with understandable questions and give the user a chance to choose, how to solve the situation. The main requirement held from the user to the system is consistency and easy to use environment. That means, even exception handling must be presented as recognizable element of the system.

Usually system reengineering (in this case, adding exception handling mechanism) leads to the more complicated interface. The user expects simplicity. If he doesn't get simplicity, he starts avoiding using the system, or doesn't use all of its capabilities.

Given above model tries to avoid the interaction of the user and the exception handling system by using e-mail notifications to the system administrator, who takes care of occurred exceptional situations. If user interrupt is needed, he is presented with the informative dialog boxes.

Conclusions

As research shows, the office activities are quite simple procedures, but they involve many actors. The occurring exceptions in this environment are quite common but sometimes need user interrupt. Office document processing result usually depends on the time needed to fulfill the business goal. If this process is interrupted, it must be handled very quickly with minor consequences. While using decomposed workflow method, it is possible to reach system support transparency and flexibility, collect reusable information for the future. This method was tested in the office activity procedure processing application. The tests confirmed that exceptions source could be detected in its occurrence spot, which quicken the salvation process noticeably.

The tested method helped to minimize the number of possible exceptional situations, made the system more friendly to the user (the user no longer receives the error messages), helped the administrators to solve exception quicker and better (they no longer have to look for the exception occurrence spot).

References

1. Hu J., Grefen P., "Conceptual framework and architecture for service mediating workflow management", *Information and Software Technology*, Number 45, 2003: pp 929-939.
2. Casati F., Fugini M. G., Mirbel I., "An Environment for designing exceptions in workflow", *Information Systems*, Volume 24, Number 3, 1999: pp 255-273.
3. Sadig W. S., Orlowska M. E., "On Capturing Exceptions in Workflow Process Models", *The university of Queensland, Australia, Department of Computer Science and Electrical Engineering*.
4. Ennsler L., Leo P., Meszaros T., Valade E., IBM Redbooks, "The XML Files: Using XML for Business-to-Business and Business-to-Consumer Applications", *ITSO, IBM Corp.*
5. Zongwei Luo, A. Sheth, K. Kochut and J. Miller., "Exception Handling in Workflow Systems", *Applied Intelligence*, Volume 13, Number 2. September/October, 2000: pp125-147.
6. Sadiq S.W., Marjanovic O., Orlowska M.E., "Managing Change and Time in Dynamic Workflow Processes", *International Journal of Cooperative Information Systems*, World Scientific Publishing Company.
7. Dellarocas C., "Toward Exception Handling Infrastructures in Component-Based Software", *Proceedings of the International Workshop on Component-based*

Software Engineering, 20th International Conference of Software Engineering (ICSE), Kyoto, Japan, April 25-26, 1998.

8. Workflow Management Coalition, "Terminology & Glossary". *WFMC-TC-1011. Issue - 3.0. Winchester Hampshire, United Kingdom. 1999.*
9. IBM Corp., "DB Magazine: Strategies & Solutions for Database Professional", *Volume 8, Number 2, Quarter 2, 2003, pages: 16-19.*
10. G. Alonso, R. Gunthor, D. Agrawal, A. El Addadi, C. Mohan, "Exotica/FMDC: Handling Disconnected Clients in a Workflow Management System", *Proceedings of the 3rd Conference on Cooperative Information Systems, Viena, May, 1995.*
11. Klein M., Dellarocas C., "Towards a systematic repository of knowledge about managing multi-agent system exceptions", Working paper AES-WP-200-01, Center for Coordination Science, Massachusetts Institute of Technology, Cambridge MA USA, February, 2000.
12. Jankeviciute A., "Method of structuring of hierarchical office activity procedures based on workflow patterns", MCs Thesis, Information System Department, Fundamental Sciences Faculty, Vilnius Gediminas Technical University, 2001.
13. Kappel G., Lang P., Rausch-Schott S., Retschitzegger W., "Workflow Management Based on Objects, Rules and Roles", *Bulletin of the Technical Committee on Data Engineering, Volume 18, Number 1, March 1995: pp 11-19.*