Butterfly Effect in Exception Management: Preventing Cause and Stopping Reaction

DOVILĖ VOJEVODINA, GENADIJUS KULVIETIS
Information Technology Department
Vilnius Gediminas Technical University
Saulėtekio 11, Vilnius, LT-10223
LITHUANIA

Abstract: - E-business applications are getting more complex as the demand for their usage and additional functionality is growing. This phenomenon has two sides: one is that e-business is spreading and required technologies are forced to improve, the negative side is that fast growth of technologies are leaving quite many holes for all kinds of exception appearance and starts lacking user friendliness in its functionality. Basically, if user encounters one exception while working with e-business application, this exception tends to become cyclic and initiate other exceptions in system background. In such cases, exception prevention mechanisms are vital as they are the key to the system flexibility and stability. The same is with dealing with already occurred exceptions. It is vital to help a user to finish a procedure and to deal with the already made damage. This paper is dedicated to the analysis of possible ways and mechanisms of solving this kind of problems.

Key-Words: - workflow exceptions, business processes, human factor.

1 Introduction

As e-business is rapidly spreading among companies seeking better communication quality with partners and customers, big amount of specialised technologies are being invented or upgraded. The most usual of them are workflow based, which is very convenient for constructing business processes in most comprehensible ways, as usually graphic elements and common language is used between business persons and system architects.

Definition 1.1 (Workflow) The automation of a business process, in whole or part, during which documents or tasks are passed from one participant to another for action, according to a set of procedural rules [[14]].

To this day, there are more than 200 of such technologies are available commercially. Unfortunately not every technology of this list is really workflow technology and more than that, it suites to be used as a tool for realising e-business suite. The true workflow technologies are complex and have about thirteen mandatory criteria, which it should satisfy, such as graphical maps, simulation tools, conditional routings, task queues, relationship-based routing and even exception handling [[7]]. This point to a very significant fact that many researches acknowledge exceptions as unavoidable part of any business case and thus interpret exceptions even more carefully than the business process itself. The basic e-business process is very plain and basic, but the essence of it is constructed from conditional branches which should represent real world situations handling. A good workflow management system is responsible for dealing not only with these more or less predicted business cases but also with cases which are new to us (business users) and the e-business application itself.

Definition 1.2 (Workflow Management System) A system that completely defines, manages and executes “workflows” through the execution of software whose order of execution is driven by a computer representation of workflow logics [[12]].

E-business environment, its traditions and ways of achieving it is changing every minute, including environment, business actors and rules, even demands of what e-business should accomplish. That leads to the constant cycle of e-business renewal, so the chosen tools to accomplish this task must ensure not only these thirteen criteria of workflow technology, but many more, which involves human interaction.

Basically, human is the only and most important part of this puzzle. Human drives the changes in e-business and mostly human factor is responsible for the biggest part of exceptions in e-business
applications. As e-business itself is just a set of rules, which define how every part of it should function and what to achieve, exceptions are cases for non-defined business cases. There are mostly two factors, which are the most important in any e-business applications and most difficult to control. It is application design and environment \([11]\). The lack of flexibility in controlling these parts leads to the exceptions.

Human mind is still a mystery, so most of the human driven exceptions usually are complex in their nature. Any exception which is raised in user level can cause quite big response in many parts of whole business application. The most dangerous are those which result in exceptions in the backend of the system, where business logic management engine resides. While trying to solve such situations and not noticing all the already done impact of current exception, system architects can unintentionally raise more errors and exceptions in many layers of business application. This event can be compared to the butterfly effect, when one small deviation in the current nature of the existence can raise vital changes in the entire surrounding environment.

The next section of this paper covers the analysis of exception nature, their impact to the business goals and proposes ways of achieving flexibility in handling them.

### 2 The Phenomenon of Exception

Some researches come to the conclusions that exception is a positive phenomenon as it leads to the system perfection and stability of the operations in an organisation \([1],[2]\). But more important it is to ensure the satisfaction of the client using this application, so that the executed e-business process finishes smoothly and brings the expected results. Different authors differently see the exceptionality of the case. Some defines as acceptability, frequency and degree of difference, others extends the symptoms to exceptionality, handling delay, amount of work, organisational influence, cause and rule impact \([11]\). Still it depends on the perspective used to evaluate the exception: business, technological or human factor. From the human factor point of view the exceptional case would be not recognised case, which appears unexpected, with business goal achievement delay/failure and without possibilities or tools for handling.

As much as we try to design a perfect workflow to support the processes of an organization, there will always be unexpected situations. A workflow definition typically specifies the normal, or ideal, flow as well as possible deviations which can occur. The \([14]\) terminology does not include a definition of workflow exception. Therefore, many researches have come to define these unpredictable situations in their own manner. In the context of enterprise architecture, an exception is the projection of real world situations, which in some manner is of an ad-hoc origin.

**Definition 2.1 (Workflow exception)** Any business activity that can not be executed in a predefined manner is referred as an exception \([13]\).

One of the ways to classify exception is to identify cause of it: application errors, failures or deviations, and sometimes the lack of performance and efficiency. Exceptions mainly affect the business goals which a particular workflow helps to achieve. Here again we deal with the context in which a particular workflow is executed. That context usually is a reason how exceptions are classified \([11]\). The classification of exceptions in a business context pertains to either the goals and assumptions, are related to activities and how they are related to resources. These classes cover almost all main aspects of enterprise architecture. We must not also forget that exceptions occur during the runtime of workflow which is why it is most difficult to deal with. One can plan and try to cater for possible deviations but there will always be unexpected behaviour.

In \([8],[9],[10]\) a distinction is made between changes (referring to business logics and build-time) and exceptions (referring to run-time and technologies). The latter is usually a result of change which can be made in some layer of e-business architecture and by so doing initiate an exception in other layers.

Workflow is mostly related to the business layer, but workflow management system covers all layers of e-business architecture and, thus, also plays an important role in the infrastructure and application layers. \([15]\) classifies exception according to these levels and assumes that the damage of exception can be noticed in other layers than raised. And indeed it is the most important property of exception the place where it was raised. Knowing the place we can identify other exception properties, such as context, run-time input/output data and damage. In this paper we focus on exceptions which are unexpected which have to be dealt with instantly. The important issue is to construct the e-business model which could check for the validity and integrity of run-time data and such. Seeking to catch the exceptions of such nature we must think of services in each level which would register and...
define unexpected situations. Basically, an exception becomes known if it is detectable and resolvable and it becomes resolvable if it is detectable. This is illustrated in Figure 1 which represents simple three-dimensional exception specification. This specification/ classification schemes aids in dealing with exceptions [[15]].

Fig 1. The dimensions for basic exception specification: is it an exception for sure, can it be detected and if there are methods for solving it.

Though these dimensions are truly connected with each other and influence each other, while experimenting with real life situations, we come to a conclusion, that there is dependency and sequence connecting these dimensions into one life cycle. Basically, we encounter exception while executing a process. After we encounter it, the most important property of exception is detectability. If we can detect it, then we can classify it as exception and think of its handling ways, which can be reached only through supervision of encountered situation. As soon as we define the handling ways, this exception becomes known and adds knowledge to our business case.

Still, any solved exceptions, which have just become a known business case, can leave unexpected handling holes in business process, which will cause other unknown exceptions.

Fig 2. The sequence of exception life cycle.

If we want to solve and prevent the emergence of human driven exceptions, which we state mostly dangerous to a business case, we must firstly seek for the cause of them or events leading to them. The main causes would be:

- casual environmental changes – like unavailable human recourses due to sickness, vacation or mission,
- wrong decisions in business process execution – when a person is not familiar with a procedure as it should and therefore makes mistakes,
- lack of application management knowledge – a person is not an expert in using particular application,
- unpredicted turn of business rule – then the person decides to apply a new way of executing the process or task and current process lack this rule.

From here arise a new question: if human factor exceptions are really unpredicted situations or can we prescribe them to the parasitic cases of the workflow. Strictly from the application designer point of view, we can agree with such a possibility, but looking into the business process purpose itself, we can state, that they are business rules changes or exceptions. To decide if a registered non-defined case is an exception, we can check against atomicity, consistency, isolation and durability properties in particular case [[3]]. If even one property is true, we can definitely call the case an exception.

3 The Ways of Dealing with Exceptions

The important question is why we should take exceptions into account so seriously and why we should try to avoid them. There are many business cases which illustrate how unnoticed exception in workflow (which were not modelled and noticed in time) made the biggest and oldest companies bankrupt by injuring their, stable at a glance, enterprise architecture [[11]]. The source of exception initiation is always human-user or a system, in both cases the seriousness off exception damage must be taken into account. As it was stated above, the most important thing is to avoid the echo of exception in other application layers than it was encountered.

The most important step of the exception handling process is exception detection. This phase can be achieved in many ways, such as making the
ideal graph of workflow and comparing every run-time step to it and if we find distinction, we register exception. The other way is to build sentinels, specific services, which would be active all the time and gather information/data on workflow instance run-time and compare it to predefined ontology or taxonomy. [8] always stress that the one who wants to deal with exceptions must go behind the system. The detection stage should be related to going to the back of workflow management system and gathering information there, which is usually a cause of unpredicted and unknown exception.

After identifying the right spot of exception occurrence and gathering all possible data, which will help to deal with raised situation we can pass to the second stage: the exception diagnosis stage. Where the detection phase tends to be similar for various contexts of e-business architecture, the diagnosis phase is highly dependent on context and (therefore) differs from case to case.

This is particularly true if we care to catch not only infrastructure and application exceptions but also business logics exceptions. The best way here is taxonomies, which defines the context in detail and predefined hierarchy according to specifics of the context.

The result and actions taken to deal with the exception depends on how precise the diagnosis was made. So if detection of exception is a projection on ideal workflow model graph, then diagnosis is a method to identify the distinction of this projection and the reasons/sources why they occurred. After approval of the exception diagnosis, the last step of the cycle is to select the appropriate handler. This phase is the most simple as system or human has guidelines, specification and just need to execute them correctly and finish the case. The fulfilment of business goal is not at stake during this stage though it may seem so. The decision of the impact on business goal must also be made in diagnosis phase, leaving the handling phase only to confirm the correctness of the diagnosis.

The handling phase can, none the less, be tricky as other unpredicted exceptions can be raised if diagnosis or detection phases failed. However, knowing the reasons and the cause of the exception, one only needs to select the right procedure, method, function or modification and enact it on the predefined spot, which is also identified in diagnosis phase. Note that one should not forget that the spot of exception occurrence will not always be the spot of fixing if we care about the fulfilment of business goal. The whole exception handling cycle is illustrated in Figure 3.

![Diagram of exception handling cycle](image)

**Figure 3.** The basic lifecycle of an exception, involving its detection, human decisions (if need arise) and handling procedures.

Figure 3 clearly shows that for any unpredicted case we should find more or less predictable handling case. The handling of exception is almost always achieved through the additional procedures or functions or other program elements which are newly created for this particular case or reused older handling procedure elements. Once again we should notice that if we bring into the e-business
application new elements, which were not natively included, we bring a new area for new exception emergence. Therefore, we should always avoid new elements in any part of e-business application, as one change in business logic can not always be prepared to meet this change in application or infrastructure level.

4 The Key is the Flexibility

Flexibility has always been an object of discussion. Many researches are curious of the true nature and influence to the process of flexibility. The different means of flexibility has been noted. The difference ways and places of achieving flexibility have been found. In this paper we think of flexibility as of:

- Easy design and change;
- Easy enactment of changes in running workflow instances;
- Fluent and transparent support of exception handling and failure recovery; and
- Dynamic workflow schema evolution [[16]].

Such definition of flexibility should be the essence of any application architect as it points to the ways of coming up to the truly supportable and exception non receptive business environment.

From the analysis of the previous sections, it is obvious that human factor exceptions are unavoidable due to the surrounding environment changeable nature. We can state that exception is already a part of the application, as it starts changing from the very first day if its implementation. That leads to the assumption that exception is already a part of the application, but not yet activated and thus must be regarded as mere deviation from the current business state and must be dealt with using “if condition then conclusion” method [[4]]. Still, such method can be not enough between distributed processes, of which e-business application is constructed, when processes interact with external applications and have side effects [[5]].

Some authors suggest using an integrated perspective when exception specification is projected on several layers of evaluation, like strategic, tactical and operational level, each of them pointing to particular side of exception properties [[16]]. Basically this points to the application ability to support ad-hoc processes and ad-hoc processing.

As every time the distinction in properties is found, we must consider changing the particular workflow instance or whole process.

Table 1. This table depicts results of the experiment gathering and analysing human involved exceptions in business application constructed as workflow.

<table>
<thead>
<tr>
<th>Exception</th>
<th>Exception is recognised</th>
<th>Exception has a handler</th>
<th>Business result change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lack of needed workflow branch</td>
<td>80%</td>
<td>5%</td>
<td>50%</td>
</tr>
<tr>
<td>Promotion of lifecycle not in</td>
<td>95%</td>
<td>90%</td>
<td>20%</td>
</tr>
<tr>
<td>time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wrong data usage</td>
<td>50%</td>
<td>95%</td>
<td>10%</td>
</tr>
<tr>
<td>Resource unavailable</td>
<td>80%</td>
<td>70%</td>
<td>15%</td>
</tr>
<tr>
<td>Lack of system usage knowledge</td>
<td>40%</td>
<td>10%</td>
<td>5%</td>
</tr>
</tbody>
</table>

Here we come to the conclusion, that ad-hoc property is the greatest e-business application advantage seeking flexibility in managing human factor exceptions. Still, this property can be enhanced and generate better results if we could manage the knowledge of the whole exception lifecycle, like the cause of appearance, processes, tools or even actors usually involved, screenshot of current situation and data log of processed data at the exception time. Surely this information is nothing without proper analysing tools, so here we once again face the proper workflow management system features, such as workflow metrics and monitoring, workflow statistics, exception handling.

The table shows that as more the exception is recognised as less it is an impact the final business goal, and as much it does impact the goal, there already exist the proper exception handlers for such cases.

The described e-business architecture (see figure 4) is a model for achieving flexibility in human driven exception handling, as exception handling layer is like a glue between human (system user) and business logics. In so doing we can achieve the direct communication and human decision transfer to business logic, through the exception layer which help to evaluate the properties of possible exception/deviation. This is achieved by using current situation specification and projection on predefined taxonomies.
5 Case Study

While analysing the nature of exceptions, the possible their cure and avoiding ways, we still must not forget that exceptions can not be fully avoided and handled. This shows everyday practice while using business applications. To this day, many people are using e-business technologies for paying bills, shopping, getting certified or mostly important managing their own accounts. It is obvious that the processes which involved our own money are mostly important for us and we are always double cautious in such situations.

The figure 5 shows the most common exception result in e-business application. The exception was achieved after basic human mistake – wrong data entry. It would seem that such case should be predicted in such an application. Still the user gets the most uninformative error message, which points to only two deductions in this case:

- the user account lacks money,
- some serious problem was raised in the application.

The first assumption terrifies the user, as he/she knows the state of his/her funds and get worried if his/her money is in its place. The second assumption leads to a conclusion that user action raised some great fail of the application. In both cases user doesn’t get enough information, which is mostly needed for him/her at the moment:

- is everything is in order with user money account,
- was the transaction achieved,
- what the user should do next.

Unfortunately, user does not get any information nor from the system directly, nor from the application support team, while the transaction was actually made, but the process suspended as it does not finished registration and informative procedures.
What would any user definitely do in such case is to use the buttons available to him/her and try again to full fill the transaction. But such actions only raise more uncertainty in the actions and results, as we do not know at the moment the already made damage in user profile, account and business process itself.

Figure 6 clearly describes that such unexpected exception can initiate other back-end system exceptions, which are not usually noticed at once, but maybe after some time. In response such left and unnoticed back-end system exceptions can again raise the same or other exceptions in user level and initiate cyclic exception occurrence effect. That is why it is always vital to remember to start handling any exception case from going to the back of the system and starting checking from the very roots of whole application logic.

The same raised unpredicted exception can lead to two more uncertain conclusions: the account balance should be checked and the business procedure itself should be finished. If we leave the current procedure as it is, the result would be, that some data can be left open for intruders (if sessions were not finished properly), the user profile can be messed up and contain wrong information which would prevent him/her from making other transactions in the future, and current transaction will be questionable (was it achieved as it should). Many of these questions could be avoid if the user would be supplied with proper information and help at the moment. While the user didn’t get it, he/she initiates other doubtful cases, which can later raise unknown exceptions.

The bright side of this situation is only one, if we manage to detect the roots of current case and find a proper handling way; we gain knew valuable
knowledge of this business process and related technologies. This unpredicted business case becomes known exception.

4 Conclusion

What truly worries every e-business user and supplier is a simple and reliable process delivery from its start point to the end. Still, the changing environment which influence the e-business itself, emerging technologies and human actions ways, is great danger for any e-business application results achievement. The success of any e-business application depends on the result and user satisfaction. It is our, as system architects mission, to insure the safety and flexibility of this process at any point of its execution. The practice shows, that great peril lies not only in unpredicted exceptions but also in business logic optimisation holes, which eventually can raise the same unpredicted exceptions. Therefore it is very important to classify and gather exception information, not only about already happened events, but also about possible to occur exceptions and to analyse this data in purpose to avoid them. The good e-business application should not only be flexible in solving exceptions but also in predicting them and automatically avoiding. Current research result leads to the conclusion, that human driven exceptions are most difficult to predict and therefore to fully avoid. The only way is to gather knowledge of human behaviour and help system to take proper decision in execution.

References: