Baltic J. Modern Computing, Vol. 9 (2021), No. 4, 466-489
https://doi.org/10.22364/bjmc.2021.9.4.06

Analysis of Concurrent Execution of Business Processes

Janis BICEVSKIS[1], Zane BICEVSKA[2], Girts KARNITIS[1], Anastasija NIKIFOROVA[1], Ivo ODITIS[2]

[1] University of Latvia, Raiņa bulvāris 19, Riga, LV-1586, Latvia
[2]DIVI grupa, Avotu iela 40A-33, Riga, LV - 1009, Latvia

janis.bicevskis@lu.lv, Zane.Bicevska@di.lv, girts.karnitis@lu.lv, anastasija.nikiforova@lu.lv, Ivo.Oditis@di.lv,

Abstract. The authors offer a method for detecting potentially incorrect execution of concurrent business processes. It is achieved by using symbolic execution of business process descriptions. The proposed method provides six steps: create a detailed business process description, define transactions, define the incorrect business process execution, create a tree of executable scenarios, calculate the results of the concurrent execution and identify scenarios leading to incorrect results. The proposed algorithm applies to both formally and informally described processes. The method was applied to analysis of different concurrent processes in e-commerce solutions, ticket distribution systems and hotel bookings.
Keywords: Concurrent transactions, symbolic execution, risks of e-commerce processes.

1. Introduction

In the real world, many processes are run concurrently, i.e., multiple instances of processes are executed simultaneously using different input data and sharing resources (such as data in a database). If the multiple processes do not share information, they can be executed correctly in a single computing system at the same time. The computing system itself ensures that the execution of one process does not affect the execution of others.

The situation is fundamentally different when multiple processes use shared information resources and are executed concurrently. In such a case, the global data used by one process may be modified by another process whilst an execution of the first process is paused or interrupted. And this may lead to incorrect results in the execution of the processes.

The concurrent execution of processes is necessary if processes have several steps which may take longer, and other processes cannot be stopped at this time. For example, if one customer chooses a product in an internet shop, it should not stop other customers from shopping in that store - both processes must be executed concurrently, at the same time. This may lead to two customers choosing the same item of the product, although

only one is available. This situation is caused by the concurrent execution of the processes, and it is not possible for the sequential/linear execution. The scientific novelty of the study: a method is proposed to identify the incorrect execution of concurrently executed business processes.

The research contains two interrelated parts. Theoretical studies of the concurrent execution of the processes have been carried out first. It is assumed that the processes to be analyzed are described in a programming language where the language semantics is strictly determined. In addition, the programming language includes a transaction execution mechanism like that used in database management systems. In general, when traditional programming languages with cycle and arithmetic operations (bidirectional counters) are used incorrect concurrent executions of business processes cannot be detected because the problem in general is algorithmically unsolvable. If the simplified business process description language CPL-1 (Concurrent Process Language) language is used, the problem can be resolved. The developed algorithm determines for every two processes defined in CPL-1 the possibility of an incorrect execution of concurrent processes. The results of the respective theoretical studies are detailed and published in (Bicevskis and Karnitis, 2020).

The second part of the research is devoted to the use of the algorithm described in the theoretical part for the analysis of e-commerce risks caused by the concurrent at such a level of accuracy that it is possible to determine the feasibility of any process scenarios and calculate the result of the scenario execution. Process execution correctness conditions are formulated in the next step of the algorithm, and it allows each scenario to determine the compliance of its execution result with the process correctness conditions. The obtained result of scenario execution allows identifying the process risks, which in turn let both buyer and seller to choose an e-commerce solution acceptable to them.

The research is an extension of previous studies published in (Bicevskis and Karnitis, 2020), (Nikiforova et al., 2020) and (Bicevskis et al., 2021). This paper leads the theoretical studies to their practical application, identifying e-commerce risks caused by the concurrent execution of processes.

The paper is structured as follows: a theoretical background of concurrent processes execution and the respective analysis methods (Section 2), risk analysis of selected e-commerce business cases (Section 3), a short discussion on the usage of the risk analysis approach (Section 4), conclusions and the future work (Section 5).

2. Theoretical background

2.1. Insolvability of the Problem

Theoretical studies of the problem of identifying incorrect execution of the concurrent process have been published in (Bicevskis and Karnitis, 2020). This paper will briefly repeat the assumptions and algorithms that will be used in e-commerce risk analysis.

First, there should be noted that identifying of incorrect execution of concurrent processes is an algorithmically unsolvable problem. It follows from the theory of algorithms that the Turing machine halting problem cannot be resolved algorithmically. If the business process description language is rich enough to simulate the functioning of a Turing-machine (Turing-complete language), the halting problem can be reduced to the

problem to identify incorrect execution of concurrent processes. Turing-complete are the languages with two bidirectional counters and loops. Hence, the identifying of incorrect execution of concurrent processes is algorithmically unsolvable if a language with two bidirectional counters and loops is used to describe the business processes. A similar problem for setting up full sets of tests was addressed by Kalnins with co-authors (Barzdins et al., 1975).

The following chapter will offer a simple process description language CPL-1 that includes a transaction processing mechanism, and an algorithm will be offered that determines whether an incorrect concurrent execution of any two processes described in CPL-1 is possible. The algorithm requires two steps: (1) for any process execution scenario, using symbolic execution, determine whether the given scenario is enforceable, and (2) as a result of the scenario, it shall be assessed whether the scenario is permissible/correct in return for the business process to be analyzed.

2.2. Process description language CPL-1

The language CPL-1 is defined by using a simplified hypothetical computational system (Bicevskis and Karnitis, 2020), and it consists of:

- processes are programs written in the process description language CPL-1; a process consists of several successive transactions, the transactions are completely executed, they are not interrupted during their execution and no other process is executed at that time; local variables can be used in the programs; you can assign a real number to the variable; the variables can form numeric and logical expressions; numeric expressions include only addition and subtraction operations, there are operators to assign values to variables in the programs,
- input data are parameters and global resources (global variables); parameter values are passed to the transaction, parameters can contain real numbers; a global resource is a variable, which is available for multiple transactions that can run concurrently and whose values can be read and written by multiple transactions,
- execution configurations specify which transactions with which parameters can be executed concurrently and what will be the value of the global resource at the start of an execution,
- executor is a processor for executing the commands specified in processes; concurrent execution of transactions is possible; execution is carried out according to execution configurations.

The CPL-1 contains the following commands:

START PROCESS – start process
END PROCESS – finish process
BEGIN TRANSACTION – start transaction
COMMIT TRANSACTION – commit/finish transaction
READ (x,r) – read the value of the global resource r and write it into variable x.
WRITE (x,r) – write the value of the variable x into global resource r.

There are logical constructions available:

IF L THEN
BLOCK1
ELSE
BLOCK2
ENDIF

If the logical expression L is true, then the BLOCK1 is executed and execution continues with the command after ENDIF; if not, the BLOCK2 is executed. BLOCK1 and BLOCK2 contain one or more commands.

y = EXPR(x_1, x_2,… x_n) is an expression of parameters x_1, x_2,… x_n (local variables), and y is the result.

A concurrent execution of one or more processes is a session. A global resource value r is defined at the beginning of each session. Processes are initiated by passing parameters to them (transaction call). Each process may contain multiple transaction calls. The final value of execution is stored in the global resource r, and all concurrently run processes are allowed to change it.

When transactions are executed in a serial manner – each next transaction Ti+1 begins when the previous Ti is ended - the result is not dependent on the time dimension. But the concurrent execution of business processes using a transaction mechanism is affected by the order of the execution of multiple individual transactions. There are two sets of process execution scenarios possible – the concurrent (C) and the serial (S) (Nikiforova et al., 2020). A concurrent process execution is considered incorrect if the results of concurrent execution for the same values of parameters differ from all the results of serial process execution. This requirement is borrowed from the database theory.

Obviously, there are no loops in CPL-1 possible. Thus, the program contains only finite length pathes and the concurrent execution path of several processes is also of a finite length. To prevent an incorrect execution of transactions, the results of serial and concurrent executions must coincide for all parameter values.

2.3. Correctness of Execution of Concurrent Processes in Language CPL-1

There was an algorithm created (Bicevskis and Karnitis, 2020) to (1) find out whether incorrect concurrent execution of any two processes described in CPL-1 is possible, and to (2) determine all possible scenarios for incorrect concurrent execution of both processes. The proposed algorithm is based on symbolic execution of the program, which enables to create conditions of process execution scenarios, to execute them and to produce symbolic values of expected execution results. If symbolic values for serial and concurrent execution of processes differ for the same argument values, the potentially incorrect concurrent execution cases are detected.

Briefly, the algorithm is applied according to the following. First, process breakpoints (the points for possible interruptions) will be defined. Next, a process concurrent execution tree will be constructed using symbolic execution; the tree will contain all possible process concurrent execution scenarios. All possible concurrent execution

incorrect situations will be identified by comparing the tree branches of the scenario tree for the serial execution with those for the concurrent execution.

In the following chapters, the algorithm proposed in the theoretical study is used for the analysis of two processes described in CPL-1 – (1) payments without reserving the amount due, allowing reading, and updating of the global resource in transactions, and (2) payments with reserving the amount due, allowing reading, and updating of the global resource in transactions.

The proposed examples show that, in the first case, two processes may be incorrectly concurrently executed. In the second case, such incorrect process execution is not possible. These results point to the risks of e-commerce concurrent process execution, a detailed overview of which will follow in the next chapters.

2.4. Example of Incorrect Concurrent Process Execution

Let us have a look at a simple example of concurrent processes execution discussed in (Bicevskis and Karnitis, 2020). It supposes two payments p_1 and p_2 to be debited from the bank account r (a global resource r). Two instances of the payment process are executed - *Payment*(p_1,r) and *Payment*(p_2,r). The *Payment* process has been described with the following program:

```
START PROCESS Payment(p, global(r))

      BEGIN TRANSACTION          T1
  2                READ (x,r)
      COMMIT TRANSACTION T1

BEGIN TRANSACTION      T2
  3 IF p<=x THEN
  4          WRITE (x-p,r)
  5 ENDIF
COMMIT    TRANSACTION T2

END PROCESS
```

If we put a breakpoint after the end of the first transaction T1, the transactions from different processes can be executed concurrently. After the T1 is executed for the first time, the same process is continued with the transaction T2, and the second process is started by executing the transaction T1. Note that reading from the global resource r is in the transaction T1, while writing into the global resource r takes place in the transaction T2.

Two concurrent execution scenarios:
(Payment(2,17),T1);(Payment(5,17),T1);(Payment(2,17),T2);(Payment(5,17),T2)
(Payment(5,17),T1);(Payment(2,17),T1);(Payment(5,17),T2);(Payment(2,17),T2)
provide results 12 and 15. Such a result is impossible using the serial execution of the processes that would lead to the result 10. The result of the execution is thus considered to be incorrect. The detailed proof see in (Bicevskis and Karnitis, 2020).

2.5. Example of Concurrent Process Execution with Resource Reservation

Let us look at another example – concurrent payments with resource reservation. The process *PaymentWithReservation* receives as parameters p - the amount due, r - the saldo of the bank account, q - the reserved amount.

```
START PROCESS PaymentWithReservation (p, global(r, q))
                    reserved=FALSE
   BEGIN TRANSACTION        T1
1      READ(x,r)
       READ(y,q)
2      IF p<=x-y THEN
3                          WRITE(y+p,q)
                    reserved=TRUE
4                   ENDIF
   COMMIT TRANSACTION T1

         BEGIN TRANSACTION T2
5        READ(x,r)
         READ(y,q)
6      IF reserved=TRUE THEN
7        WRITE(x-p,r)
         WRITE(y-p,q)
8                   ENDIF
     COMMIT TRANSACTION T2
END PROCESS
```

In this example, the results of concurrent execution did not differ from any serial performance of any scenario. Accordingly, incorrect execution of concurrent processes is not possible. The proof can be found in (Bicevskis and Karnitis, 2020).

2.6. Comparison: Transaction Execution in DBMS and Concurrently

This chapter will compare the approach described in the previous chapters with the approach used in relational DBMS. The DBMS transaction mechanism has been established for a long time and is still being extensively developed, as the relational databases store and process millions of data records. The performance of data processing is crucial for a high customer satisfaction, and one of the solutions is the concurrent execution of many transactions. While reduced concurrency is generally accepted as a compromise for higher transaction isolation level needed to maintain data base integrity, it may become a problem in interactive application with high reading/ writing activity.

The method proposed in this study differs significantly from the transaction handling mechanisms traditionally implemented in DBMS. The proposed method, using the "white box" approach, offers analysis of business processes instead of executing them. As a result, the probability of an incorrect concurrent execution of processes is identified. The approach is problem oriented as different business processes have different correctness conditions.

DBMS in turn offers a universal solution – transactions are treated as “black box”, their contents are not analyzed though transaction execution results are monitored. DBMS monitors the sequence of read/write operations and undertakes a rollback operation if probably incorrect execution has been detected.

A detailed overview of DBMS operating mechanisms is given in (Nikiforova et al., 2020); let us just recall some of the aspects necessary for comparison with the proposed method.

Databases are often used to perform transactions - sets of data-dependent operations requested by the system user. The completion of a transaction is called a commitment and a cancellation before its completion is called an abort. If a transaction is aborted, all partial results, i.e., data updates resulting from those operations that were made prior to the abort decision, must be undone. This process is also called rollback.

To preserve data integrity, a DBMS must provide a set of ACID properties: atomicity, consistency, isolation and durability. One of the main properties of the transaction is an atomicity – either all operations related to a particular transaction must be carried out or none of them. If a transaction is interrupted due to a failure, the transaction must be aborted so that its partial results are rolled back and, if the transaction is completed, the results are preserved despite subsequent failures (Gostanian et al., 1998).

Perhaps the most popular concept in terms of simultaneous execution of multiple transactions is isolation - property that significantly simplify concurrent pro-gramming, since each transaction can be viewed separately, rather than having to consider all possible interleaving’s of their operations with other transactions.

Concurrency control is a coordination of concurrent access to the data base while maintaining consistency of the data (Suryavanshi and Yadav, 2012). Concurrency control techniques are divided into (1) optimistic concurrency control and (2) pessimistic concurrency control.

The optimistic concurrency control assumes that conflicts are rare, and it is possible to repair the potential losses caused by such violation by delaying the synchronization of transactions until they are close to their completion. The optimistic concurrency control requires an effective repair mechanism and collisions cannot be destructive, so it is most effective when relatively rarely conflicting writing operations take place in the system (Guzek et al., 2013).

The pessimistic concurrency control synchronizes the concurrent execution of transactions at the beginning of their execution life cycle and decides whether to accept, reject or delay an operation immediately after receiving an operation (Sheikhan and Ahmadluei, 2013). The pessimistic approach is considered safer than the optimistic, as it avoids potential problems rather than resolves them (Tanenbaum and van Steen, 2007). Thus, the pessimistic approach has become the traditional concurrency control approach, and it is is usually done by blocking records, i.e., by setting locks on the database records to avoid accessing them by several transactions at the same time. The more rows are blocked, the fewer transactions can be executed without temporary locking.

The method proposed in this study offers to analyze the business process before it is completed and to identify the probability of incorrect execution without running it. Risks of incorrect concurrent execution can be solved either by modifying/changing business processes or including in the system additional mechanisms which are triggered in the

appropriate situation and protect against incorrect execution of the business processes, like those of DBMS.

The idea of additional mechanisms that monitor correctness of transaction execution refers to another study (van Beest and Bucur, 2013). The authors propose an automated framework for the runtime verification and correction of business processes to avoid interferences. It guarantees the continuous correctness of process execution in any distributed environment. Continuous detection of interferences during the process execution is achieved "on-the-fly" by means of temporal verification of the running process along with virtual external data transactions. This allows identifying a minimal set of process checkpoints where the execution of external transactions would change the outcome of the local process. To achieve continuous correction, runtime checks are made at these process checkpoints, and, whenever an interference occurs, an "intervention process" is generated to correct the local process in its current state. Subsequently, all intervention processes are continuously verified and corrected throughout the lifetime of a process. A mandatory prerequisite, as in most approaches, is the ability to pre-define critical sections of a business process that are firmly based on their liability to process interference. Though it is not still clear whether and to what extent the additional mechanisms affect the overall efficiency of business processes.

3. Identifying Risks for Informally Described Processes

In this chapter, the results of theoretical studies will be applied to the analysis of frequently occurring processes in practice. It will first analyze other authors' works about risks of business processes. They contain detailed classifications of risks but do not address the risks arising from the concurrent execution of the processes. This confirms the novelty of the current study.

The risks of internet shopping, caused by concurrent execution of business processes, in four different areas are analyzed: theater ticketing, online stores, hotel reservation, and airline ticketing. The algorithm, developed through theoretical research of correctness of concurrent execution of processes (Bicevskis and Karnitis, 2020), will be used.

3.1. Internet Shopping Risks: Overview of Related Studies

Literature review reveals that different internet shopping-related risk classifications have been established over the last decades. A total number of risks considered to have a significant impact on users' intention for online shopping ranges from two to eight (O'Callaghan, 2017).

(Forsythe and Shi, 2003) identified six types of perceived risks that may have a negative impact on the experience of buyers: financial, product performance, social, psycho-logical, physical, and time/convenience loss. Respondents found financial risk to be the most important and significant. Considering the age of this study and the development of e-commerce over the past few years, most of the identified risks have already been processed and resolved. However, some of them remain valid, for instance, financial risks.

Formerly, financial risks were primarily associated with potential losses of money due to fraudulent misuse of credit card information. Nowadays, the paradigm on financial

risks has been changed (Al-dweeri et al., 2017). The online credit card usage-related risks are thoroughly discussed in security-related studies, and practical solutions are invented in online shopping platforms, including the implementation of 128-bit RSA encryption, digital certificates, firewalls etc. (Thakur and Srivastava, 2015). Another financial side-related risk is less covered: the trust between the customer and the service or/ and shopping service provider (Rita et al., 2019; Izogo and Jayawardhena, 2018; Huseynov and Yıldırım, 2016; Pappas, 2016; Thakur and Srivastava, 2015; Al-dweeri et al., 2017). Trust and reputation are considered now to be the concepts dominating in e-commerce most (Sellami et al., 2021).

Bezes (2016) proposes a classification where the probability of bank or personal data being stolen is understood as a "transaction risk". The probability of losing money when buying from a website or a store is defined as a "financial risk". The other categories appear to be outside the scope of this paper, so we will not cover them. In general, we believe that this classification is more accurate, and we will use the term "financial risk" according to the definition of (Bezes, 2016) in this paper. Unlike theoretical studies, we aim to cover the issue of financial risks from a technological perspective.

There is a study rejecting the significance of financial risks (Wai et al., 2019). This study, based on a survey that has been carried out between 245 country residents, identifies a "convenience risk" and a "non-delivery risk" (other classifications consider both as financial risks) to be very significant, as well as the "reliability of shipper" and the "settling disputes".

Previously consumers believed that credit card data could be very easily stolen online, shopping frauders have now become a threat to be counted on. The "performance risk" is concerned with the potential failure of a product or website to meet expected performance requirements, i.e., uncertainty regarding after-sales service (Otika et al., 2019; Pappas, 2016). According to (Otika et al., 2019), "risk perceptions" and "online shopping intention" have a significant impact on online shopping intentions for internet users, as proven by the use-case with 390 internet users in Nigeria.

(O'Callaghan, 2017) revealed that risks such as privacy, source, performance, payment, and delivery risks are predominant dimensions in internet shopping. It is also in line with (Huseynov and Yıldırım, 2016). The authors have carried out an in-depth analysis of risk-relievers. Although only one rather limited example of shopping has been analyzed with a sample size of 471 respondents, the authors' research suggests that 18 risk-relievers make sense for shopper. The main risk-relievers are (1) payment security, (2) money-back guarantee, as well as possibility (3) of exchanging the item, (4) of viewing the item, (5) of seeing item in a store, (6) price and (7) website reputation.

(Huseynov and Yıldırım, 2016) has also revealed that guarantees provided by online sellers and insurance against any kind of adverse situations were assessed as the most important factors, highlighted by 88.7% of respondents as of great importance to facilitate online shopping.

Most studies highlight the particular importance of (1) perceived risks and financial risks in particular, (2) trust to retailer (Rita et al., 2019; Al-dweeri et al., 2017; Izogo and Jayawardhena, 2018).

This topic is especially relevant due to COVID-19 pandemic as online shopping become a daily phenomenon for most of the population around the world; even grocery

stores have launched their own online shops in countries and cities where they did not exist before.

3.2. Algorithm for Risk Analysis

This study considers an algorithm for the analysis of business processes that use a transaction mechanism, which may detect possibility of incorrect concurrent execution of processes. The main steps of the universal analysis algorithm are the following:

(1) **Create a description of the business process.** Firstly, a business process model should be created. If the business process is described informally, the model can be designed as a graphical diagram where the vertices of the graph represent the activities of the business process and the arcs the sequence of activities. Process analysis is possible if the feasibility of scenarios and the outcome of scenarios are also described.

The situation is different when the model consists of program code. The execution of each statement and the sequence of activities are strictly defined then. In this case, the code analysis can be performed automatically by a tool. The program code is executed symbolically: the tool compiles the conditions for the execution of the scenario and calculates the result of the execution of the scenario. In this paper, unlike (Bicevskis and Karnitis, 2020), an algorithm will be used for an informally defined business process.

(2) **Define business process transactions.** Business process activities or a set of activities are defined as a transaction in the cases when business process execution is delegated to another system (for example, a database management system) or their execution requires a time frame during which access to common resources may not be blocked for other processes.

For example, the process of selling theater tickets can be divided into three transactions: (1) read from the database the seats sold, (2) allow the client to choose a free seat in the hall, (3) let the client pay for the selected tickets. The ticket selection process should not be blocked for other remote customers, as the selecting may take longer.

(3) **Define incorrect business process execution.** This step identifies situations that are not acceptable from the business perspective. If the process execution scenario leads to a situation that does not meet the business requirements, then it is likely that the definition of the business process needs to be revised to avoid incorrect execution results. In the case of a formal model, concurrent process execution is considered incorrect if the results of concurrent execution differ from the results of serial process execution.

(4) **Construct a feasible scenario tree.** In the case of an informal model, different process execution scenarios should be identified, and their feasibility should be evaluated. There is input data necessary that will make the selected scenario executed. The result of the analysis is represented in a tree whose branches represent all possible different feasible scenarios. In general, this can be a difficult goal to achieve. If the model is defined by program code, the "white box" analysis method is used - symbolic execution of the program code, which enables to compile the conditions for the execution of pre-defined scenarios. When solving the conditions, the solution obtained is a test case that should be executed to cover the pre-defined paths. This approach has

been known since the 1970s and is currently available in the IntelliTest tool (IntelliTest, 2019). The tool generates test case sets that cover all arcs of a C# program control graph.

(5) **Calculate scenario execution results.** In the case of an informal model, the expected result of the scenario execution from the business point of view should be evaluated. In the case of a formal model, a symbolic expression is calculated using symbolic execution.

(6) **Identify scenarios that lead to incorrect business process execution.** The following algorithm for databases is proposed to detect situations where the results of concurrent process execution and serial execution differ (Nikiforova at al., 2020). First, a scenario feasibility tree is created. Then the feasibility conditions and execution result of each concurrent execution scenario with all the feasibility conditions and execution results of the serial execution path are compared, i.e., two sets of process execution scenarios are analyzed – a set of concurrent execution scenarios (C) and a set of serial execution scenarios (S). If at least one scenario S_j from S can be found for the scenario C_i from C such that the set of conditions and results of C_i coincides with the set of conditions and results of S_j, then the concurrent execution C_i is correct, otherwise it is incorrect.

All the steps of the algorithm will be demonstrated below with the help of selected examples for an informal model.

3.3. Risks in E-commerce

The development of ICT has created preconditions for replacing the traditional buying/ selling processes with the remote e-commerce solutions. The advantages of e-commerce lie in the global spread allowing entrepreneurs to develop remote marketing and sales on an unlimited geographical scale. E-commerce solutions are perceived as the future of commerce, as more and more customers want to buy a product without leaving their homes (WEB (a)).

By giving up the direct buyer-seller communication, the buying process becomes more complicated and at the same time riskier for both the seller and the buyer. The buyer is afraid of paying for goods he has not seen yet, and the seller is afraid of the risk sending the goods before receiving the payment for them. In addition, several customers are served at the same time, and the processes interact mutually, for example, a product selected but not yet paid by one customer may be sold to another customer. The processes are complicated by many different product delivery channels and product payment options as well as by many simultaneous customers

All e-commerce processes consist of three steps: (1) ordering a product/service, (2) payment for the goods/ service, and (3) delivery of goods. These steps may vary depending on the industry and the implementation. We will identify risks for both the seller and the buyer, using different purchase-sale scenarios when serving several customers concurrently. To simplify the analysis, the processes will first be considered for theatre ticket sales, then modified for other sectors.

3.4. Theatre Ticketing

Ticketing systems have been around since old times. In the past, theater tickets were sold at ticket offices, where customers were served in turn. Each client performed three steps:

(1) chose the most suitable from the available seats in the hall, (2) paid for the ticket, (3) received at the cashier a unique ticket for the specified seat in the hall. This process does not allow concurrent execution, as only one customer is served at a time. If large number of tickets had to be distributed, they were split between several box offices, each box office being allocated a certain number of tickets for distribution which were not available at other box offices. As a result, tickets were distributed concurrently but avoiding the usage of shared resources. Such a ticketing process is obviously not in line with today's technology.

Currently available ticketing systems offer e-commerce functionality: remotely connect to the ticketing system, select a ticket, pay for it and receive a copy of the ticket on a smart device, printable file, etc. The purpose of the following sections is to identify the potential risks posed by the concurrent service of several customers.

1) Defining of incorrect process execution

The correctness of the ticketing system's performance will be assessed by the status assigned to the seats in a hall. If there are seats with the status "sold" and "not paid" at the same time, then the ticket system works unacceptably. The status *seatStatus* will be determined by the values of two parameters:

1. **availability of a seat**:
   - available – the seat is available to customers,
   - reserved – the seat has been selected but not yet paid,
   - sold – a ticket for the seat is sold to a customer (the customer has chosen this seat, paid for it and received the ticket).
2. **status of payment**:
   - paid – the amount of money for the tickets has been credited to the theater's bank account,
   - not paid – no payment for the tickets was received to the theater account.

Process execution correctness is defined according to values of *seatStatus* after the ticket sales process has been completed (Table 1.):

Table 1. Definition of process execution correctness

Availability	***Payment***	***Result***
available	not paid	correct
	paid	incorrect
reserved	paid	incorrect
	not paid	incorrect
solds	paid	correct
	not paid	incorrect

The ticket system works correctly if the attribute seatStatus for any seat in the hall has either <available, not paid> or <sold, paid> as values. Any other result shall be considered incorrect / inadmissible. The result is also considered incorrect when the same seat has been sold twice (to different customers).

Process execution scenarios will be analyzed below to determine if there are possible scenarios that lead to an incorrect result.

2) Creating a description of the business process

Before analyzing the possibility of incorrect operation of the ticket system, let us define the process in more detail. The proposed operation system of the ticketing system is one of many possible and will serve only as an example to demonstrate the method of concurrent execution analysis.

The ticket distribution business process consists of three sequential phases: Select a Seat, Pay for a Ticket and Send a Ticket.

2) The first phase: *Select a Seat*

The first phase Select a Seat consists of three activities: readSeats, selectSeats, reserveSeats. A simplified graphical model of the phase is given in the Figure 1. The activity readSeats reads information from the database about the customer's chosen performance and, if the event is not sold out, shows it to the customer. The activity selectSeats allows the customer to mark his chosen seats in the hall. The activity reserveSeats changes the information on occupied seats in the database by assigning a value “reserved” to the seat. The customer may terminate the business process if, for example, he is not satisfied with the offered seat or ticket price. In this case, seatStatus does not change its value and an incorrect process is not possible.

The selectSeats operation can take a longer time and therefore, during its execution, the common resource may not be locked; information about the seats should be available to other customers. All three operations - readSeats, selectSeats, reserveSeats - will be executed as separate transactions. Thus, the ticketing system can execute many transactions from different customers' business processes "simultaneously", ensuring the execution of successive transactions for each individual process.

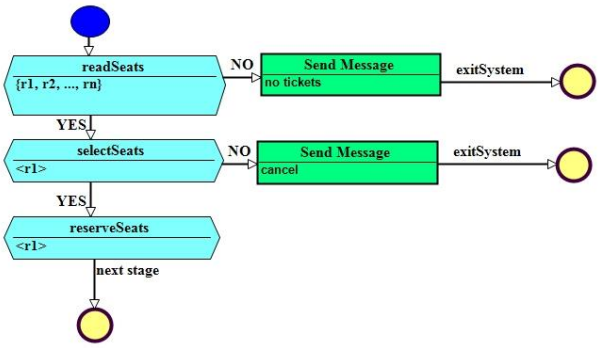

Figure 1. The simplified process *Select a Seat*

Unfortunately, the simultaneous service of several customers can lead to incorrect execution of the process. This is shown by the algorithm described in the previous sections. Let us construct a concurrent execution scenario of two processes P1 and P2:

P1(readSeats,YES)=>P2(readSeats,YES)=>
P1(readSeats,YES,reserveSeats,nextStage) =>
P2(readSeats,YES, reserveSeats,nextStage)

This scenario is feasible but there is a risk of selling the same seat to two customers if customers from both processes P1 and P2 choose the same seat. This is unacceptable, and the simplified seat selection process given in Figure 1 is risky.

The situation changes drastically when seat reservation is used: a control mechanism checks whether the seat is already booked by another process. Figure 2 gives a correct process model, in which the data on free seats in the hall are re-read before reserving a seat and in case the seat selected by P1 is already reserved in another process P2, the seat selection step is repeated. The business process is changed by adding additional controls before the actual reserving in reserveSeats.

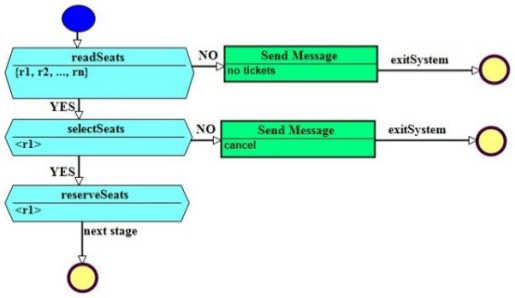

Figure 2. The improved process *Select a Seat*

A similar situation for bank transactions is analyzed in detail in the paper (Bicevskis and Karnitis, 2020). It proves that if bank payments are allowed to be made at the same time reading the account balance and recording the changed value into the account are organized in separate transactions, it may lead to an incorrect result. It has also been proven that the process of payments with a reservation cannot lead to an incorrect result.

3) The second phase: Pay for a Ticket

Banking systems offer many ways to pay for the tickets bought. Let us look at three options:

- **Direct payment** - payment from the ticket system directly using the internet banking options. Direct payment option should be integrated into the ticketing solution for each specific bank.
- **Card** - credit card payment, where the customer discloses his credit card details to the specific information system, such as Wordline or PayPal. In this case, the customer assumes a serious risk, because the card information can reach fraudsters both by hacking the information system and intercepting the card data during data transfer, and the customer may not be aware of security breaches and may not be able to mitigate them.
- **Bill** - payment of the invoice sent to the customer. In this case, payment will take longer because the customer has to wait and pay the bill but it is much safer from the point of view for both the customer and the seller.

In all these cases, the step *Pay for aTticket* must be performed as a separate transaction, because the service of other customers may not be interrupted until the end of the ticket payment process.

Step *Paying for a Ticket* poses risks to both the seller and the customer: the seller reserves tickets for a certain period, preventing them from buying other customers, and the customer, in turn, pays for the ticket, believing that he/ she will receive the ticket on time.

It is even more difficult for the ticket system to get a secure ticket payment if it is done by an external (bank's) payment system. Different banks have different payment solutions, which makes it difficult for the ticket system to unify payment processes. The execution of the payment remotely is subject to all the most common risks, including an unstable internet connection that may cause delays in communication between the ticketing system, the customer, and the bank. Banks are often reluctant to allow remote payments to information systems of little-known partners, as they can pose a threat to the banking system. Respectively, some customers may not have access to *Direct payment* and *Card* payment services.

The phase *Pay for a Ticket* contains three activities: *readAccount*, *checkValue* and *writeAccount* (Figure 3). The activity *readAccount* reads the customer's account balance from the bank's database, the activity *checkValue* checks whether the customer has enough money to pay the ticket price. If payment can be made, the activity *writeAccount* deducts the amount payable for tickets from the account balance and stores the new account balance in the bank's database.

In the case of Direct payment, the ticket seller does not see the payment process – the internet bank sends the request for the amount should be paid to the bank where the customer has the respective account. The bank returns information about the transferred amount to the ticketing system and the internet bank connects the customer back to the ticket solution. If the bank does not respond for a long time, a timeout occurs, and this is often the situation when involvement of the ticket system staff is needed.

Payments with credit cards are made by third-party payment systems, such as *PayPal*. Like direct payments, the solution ensures the execution of the payment from the buyer's account. The operation *Pay for the Ticket* returns "1" to the ticketing system if the payment is successful and "0" if the payment is unsuccessful.

Activities *readAccount* and *writeAccount* are executed as independent transactions. And it leads to risks that the payments may be executed incorrectly if run concurrently (Bicevskis at al., 2020). However, as concurrent payment execution from a common resource (from one bank account) for several customers is unlikely, there are grounds to assume that the *Pay for the Ticket* transaction is executed as one individual transaction and it cannot affect the service of other customers.

Payment for a ticket in the traditional way (after receiving an invoice) cannot be done fully automatically. The customer pays the invoice autonomously, the ticketing system must identify incoming bill payments, for example by invoice number. Only after the payment identification the paid tickets may be delivered to the customers. This solution poses risks to the seller when payment is not made on time or message transmission is delayed - the ticket may remain reserved, even though unpaid. If the ticket is sent to the customer before payment confirmation, the seller risks not receiving payment for the ticket at all. In real ticketing systems, this problem is solved by including additional controls and functions in the information system. Successful.

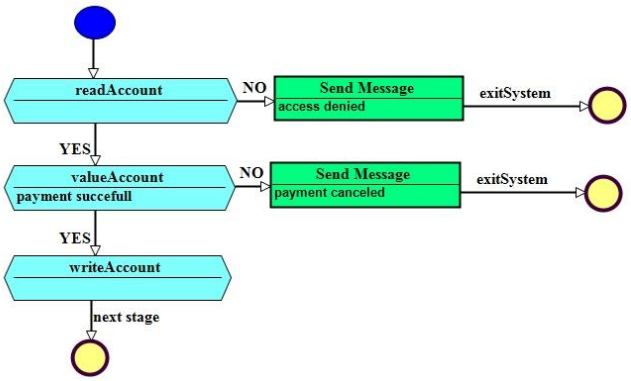

Figure 3. Process *Pay for a Ticket*

4) The third phase: Send a Ticket

Send a Ticket consists of three activities: waitAnswer, sendTicket, setStatus (Figure 4). The activity waitAnswer is waiting for a message from the banking system confirming the payment. Three situations are possible:

- **payment completed successfully -** the ticket system prepares a ticket and the activity *sendTicket* sends it to the customer, the corresponding seats in the hall are marked as sold by using the activity *changeStatus.* If *Direct payment* is used, the payments received will be identified automatically by a ticketing system. If tickets are paid with a card, the ticketing system waits for payment confirmation from the card payment provider. If the customer has paid a bill, the ticketing system must identify the incoming payments.
- **payment was not completed successfully** - this situation may occur if the customer does not have enough money in the account to pay for the tickets. The ticketing system sends a message to the customer about the refusal to purchase tickets, the corresponding seats in the hall may be released for sale by the activity *changeStatus,* and the seller may sell the ticket to another customer.
- **the payment was not confirmed timely** - this situation may occur if the message has not been received from the payment system timely. In such cases, the staff must be involved to find out the reasons and to complete the ticket purchase manually. If the bill is not paid for a long time, the information system must be able to react to it. Different solutions are possible: (1) resend the invoice to the customer, (2) cancel the purchase, release the corresponding seats for repeated sale (*changeStatus)*. This solution runs the risk that the ticket is actually paid for, but the ticket is resold to another customer due to a delay in reporting.

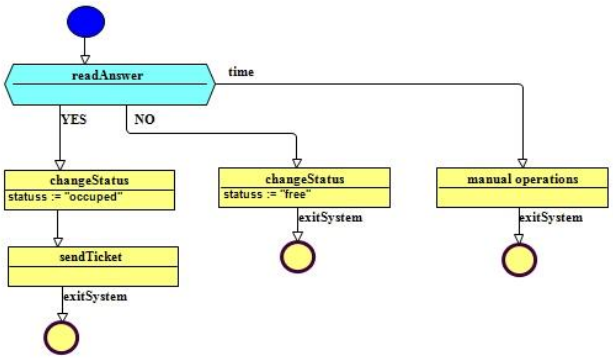

Figure 4. Process *Send a Ticket*

Of course, ticketing systems may use different ticketing processes depending on the wishes of the developer and customers. The proposed process just demonstrates a concurrent performance analysis method, which will be discussed in more detail in the next section.

3.5. Process Execution Scenarios

This chapter discusses an algorithm that allows to identify the incorrect concurrent execution of two business processes P1 and P2 in the case when the business process is described by a partially formalized model. The model consists of two parts, where the first represents the execution of process P1 and the second - the execution of process P2. The Figure 5 shows the case when the process P2 is started concurrently with P1 while the *selectSeat* activity is executed.

According to the algorithm given in the chapter 2, the analysis of concurrent execution of P1 and P2 requires creating of a tree for all possible P1 and P2 concurrency execution scenarios. This would require an analysis of very many scenarios that would exceed the scope of this paper. Therefore, just few scenarios will be analyzed to clarify the proposed method.

There are two requirements for the model – (1) activities are unambiguously defined at such a level that for any process execution scenario the conditions for the implementation of the scenario can be drawn up and it can be assessed whether the scenario is feasible; (2) the result of the scenario execution can be calculated, and it allows to decide whether the scenario is acceptable/correct for/to the analyzed business process or not.

An improper concurrent execution of the processes P1 and P2 can be detected using the following scenario:

P1(readSeats,YES,selectSeats,YES) =>
P2(readSeats,YES,selectSeats,YES,reserveSeats,
readAccount,YES,valueAccount,YES,writeAccount,
readAnswer,YES, changeStatus, sendTicket,exit) =>
P1(reserveSeats,readAccount,YES,valueAccount,YES,
writeAccount,readAnswer,YES,changeStatus, sendTicket,exit)

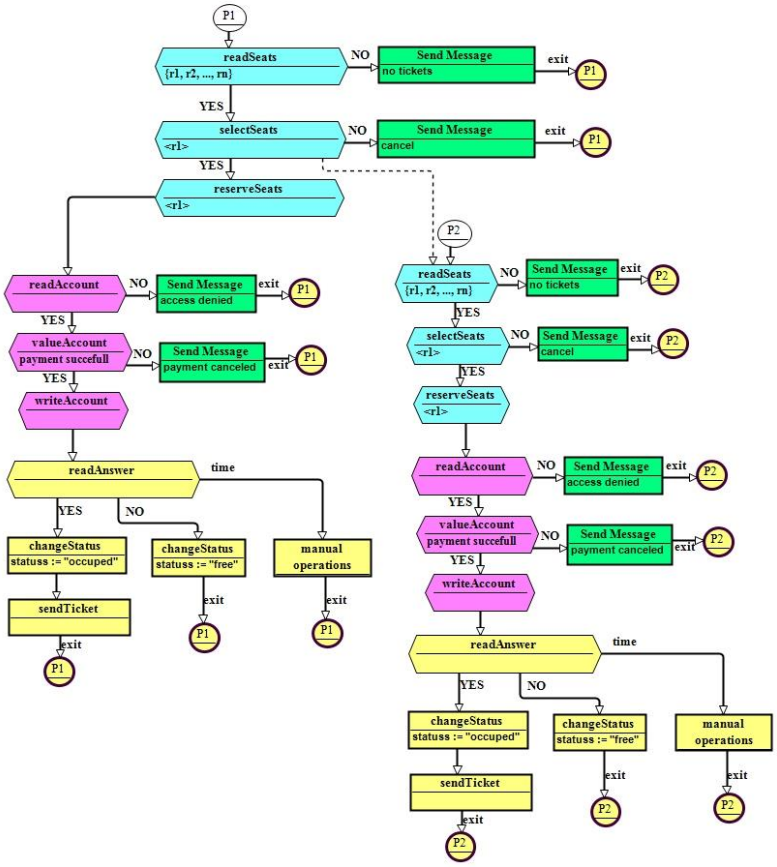

Figure 5. Fragment of the P1 and P2 concurrent execution scenarios tree.

As a result, if two customers choose the same seat in the hall, tickets for the seat in the hall will be sent to both customers. This defect could be corrected by not allowing re-booking of seats, as shown in the Figure 2. If the first process lasts so long that the second process reserves the seat still reserved by the first process, then refreshing the status of seats would reveal that the seat, initially selected by the first process, is already taken by the second process and the first process must select another seat.

Similarly, process defects can also be detected by analyzing the scenarios when payment confirmations arrive late in the ticket system and the ticket is already resold. In this case, the ticket can be purchased by another customer, even though the first customer has paid for the ticket but is late in notifying the ticketing system.

Another process defect is possible if readAccount and writeAccount are separate transactions. The saldo of a bank account then could be changed by another process at the same time. Although a situation where the balance of one customer's account is changed by several processes is unlikely, it is recommended not to leave such a risk in

the ticketing system. The defect can be avoided by using payment booking like that for booking of seats.

Sending a ticket to the customer without receiving feedback from the payment process (relying on the stable operation of the internet) is debatable. Due to technical issues, the sent file may not reach the customer, and the ticketing system will not find it out without intervention of the staff. This defect can be remedied by requiring a customer's confirmation about receiving the tickets.

There is also a process defect possible when the customer is unable to pay for the ticket but the seatStatus remains <reserved, not paid> thus excluding the ticket from the pool of available tickets.

The proposed algorithm creates an executable scenario tree, performs symbolic execution of scenarios, and calculates the result of concurrent execution to discover both process defects and incorrect execution of scenarios.

Unfortunately, even for two simple business processes, a tree of all possible concurrent scenarios can reach a significant volume. A tool for scenario analysis is needed. A prototype of such a tool was developed in Python, and it is used to analyze business processes defined in CPL-1.

3.6. Online Store

The operation of online store is determined by four steps: marketing, selection of goods, payment for goods and delivery of goods. We will not consider marketing issues in this work; the other three steps are like those in theatre ticketing systems. The industry specifics of online stores are often related to the efficiency of delivery processes.

The activity selectItem differs from the choice of tickets because the customer wants to choose the product personally, look at it and evaluate it in detail. The online stores display samples from catalogs remotely but the customer will be able to make the final assessment of the product only after bying and receiving the product. In addition, if the payment for the goods is not prompt, then online stores that do not reserve the ordered goods may sell them to another customer who has paid for the goods sooner. Such situations occur regularly in practice.

Even more risky is the customer's cooperation with the online store in the cases when the online store orders goods at a wholesaler only after placing a customer's order or receiving the payment. In these cases, it may take several days to receive the product from the wholesaler. The problem is often addressed by using own warehouses with reserve stocks of goods.

In summary, online stores without warehouses are quite risky in terms of delivery. If the goods are not reserved upon receipt of the customer's order, the delivery of the goods to another customer who has made the payment earlier is not excluded, thus extending the delivery time of the goods.

The activity payItem does not differ significantly from the payment of tickets, however, additional risks are expected if goods are expensive and standard payment limits for internet banking transactions are in use. The promptness of the payment has a significant impact on the process, as the delivery terms of the goods depend on it. Payment via internet banking at the time of ordering is not only the safest, but also the most modern payment method for shopping in online stores. The customer is redirected

to the payment website of the selected bank with an automatically prepared payment order. The order is confirmed immediately as soon as the payment is received.

Payment by payment card or bank transfer is less efficient as the product can be bought by another customer who has paid for the product sooner.

The risks of the activity deliveryItem are like the risks of ticketing processes – (1) the customer receives a product the quality of which has not been checked yet, (2) product delivery terms are determined by the customer's product payment efficiency and online store processes, (3) the online store can sell the product ordered by the customer to another customer who has paid for the product sooner, thus extending the delivery time.

3.7. Hotel Reservation

The hotel reservation specifics are availability (or non-availability) of a hotel room for a specific time frame. It is not possible to assign physically the same room to two customers, but you can still sell it twice for the same tome period, just like in any other internet store. There is also a risk that the room reserved for the customer will not be released in time, for instance, because the previous customer has extended his stay in a hotel. Such situations can be resolved by the hotel staff.

An insignificant risk exists if the customer searches the most advantageous offer from many, and, in the moment of the final booking, the special offer may no longer be valid because another customer has already booked it.

Payment for hotel services usually is made at check-out. By directly contacting the hotel staff, the customer acknowledges the services received and pays for them. Unfortunately, it may happen that the credit card has not enough coverage to pay for the services received. A similar situation arises when a customer has left a hotel without paying the hotel.

3.8. Airline ticketing

As in the case of hotel reservations, the concurrent execution of air ticket purchases is risky. When searching for the best flight route for a long time, the route chosen by the customer may not be available anymore because another customer has already booked this route. Airline tickets distributors try to reduce this risk by providing customers with additional information about the number of available tickets.

Payment for tickets contains the same risks as payment for goods in the online store. Direct payments are secure and fast. Unfortunately, this form of payment is only available when the ticket distributor has had a contract with the respective bank and has implemented direct payments into his information system. The customers with accounts in foreign banks are usually forced to use cards. The Card and Bill payments may take longer, and it arises risks for timely delivery of tickets before the flight. Efforts are being made to reduce these risks by setting up flexible information systems, communicating promptly with customers if payments not received timely, and sending information on ticket reservations to airports directly.

4. Aspects of Use

4.1. Applying of Analysis

The paper offers a method that allows to identify purchase and sale risks, when serving several customers concurrently. The analysis of concurrent e-commerce execution processes is required:

1. For online store customers to identify the risks of purchasing: Does the quality of the selected goods meet the customer's requirements? Will the goods be delivered on time? Is there a risk of non-delivering for pre-paid goods?
2. For online store owners to check the customer's solvency and reliability as well as to develop and improve the business processes allowing increase the volume of goods sold.
3. For online store developers to implement correct e-commerce systems. However, additional work is needed to describe the business processes to be implemented in sufficient detail - to create a scenario tree and to reveal scenarios that lead to an incorrect result.

Unfortunately, fraudulent transactions when buying goods in online stores are still common. This is evidenced by the warnings provided by the Latvian Security Incident Prevention Center about the risks when purchasing goods in online stores (CERT, 2020). The warnings propose to do the following before using an online store:

- **Check the security of supported payment methods.** The payment methods offered in online stores are different. It is recommended to use credit cards and payment methods that offer consumer protection, allowing consumers to get their money back if the product is not delivered, such as PayPal.
- **Contact the online store**. A good online store knows that customers want to communicate in a variety of ways. Check if the company provides a phone number, email address, chat facilities, or just a non-personal form of communication. If in doubt, call the company or send a request for more information via email or social networks. A professional online store usually responds within an hour or no more than two business days, depending on the environment and time zone.

These instructions of the leading security authority in Latvia confirm the need for online store risk assessment once again.

4.2. Limitations for Use

The proposed algorithm can find all possible concurrent scenarios and parameter value conditions for an arbitrary number of processes and transactions leading to incorrect execution. It operates in accordance with the principles of “white box”. Hence, the knowledge of business processes is required to gain benefits from the method. Although both formal and informal definitions are allowed, the level of details of the knowledge can affect the accuracy of the output, i.e., detection of all potentially incorrect executions of processes. The involvement of experts with the specific knowledge of the business processes is essential.

This limitation applies to all existing solutions on this issue and there are no objective reasons to assume that it will be resolved. Thus, the proposed approach can be mainly

used for internal needs, i.e., to inspect the existing system(s) and relevant processes that run concurrently. It can also be used when a new e-commerce system is being developed to avoid implementation of incorrect business processes.

The proposed methodology supports both formal and informal definitions of business processes. The informal definition is suited for many external users because it does not require defining the processes in CPL-1 but there is still a deep understanding of business processes necessary, as this affects the accuracy of the analysis. The proposed approach may reduce the number and level of risks, but it does not guarantee the elimination of them.

There is another technical limitation: An appropriate tool is needed to support concurrent execution analysis. The number of processes, transactions, breakpoints, complexity of the programs may vary, and it has a significant impact on the size of the scenario tree.

This tool is currently being developed and it serves as the future work for our research project. The interpreting tool will use a definition of processes in CPL-1 to build the tree representing all execution paths implying from these processes. The final step will be an analysis of these paths, where the result of the serial execution of respective processes will serve as correctness criteria. This will automate the execution of the algorithm described in this paper.

5. Conclusions

The paper summarizes the research devoted to risk identification for concurrently executed business processes. First, theoretical studies of the concurrent execution of processes were carried out, followed by identifying risks of e-commerce business processes based on theoretical research results.

Theoretical studies assume that the processes to be analyzed are described in a programming language with an integrated transaction mechanism and strongly determined language semantics. The main conclusions are:

- In general, incorrect concurrent executions of business processes cannot be detected if traditional programming languages are used which include loops and arithmetic operations (bidirectional counters).
- Incorrect concurrent executions can be detected if processes are described using the simplified business process description language CPL-1 which contains a database transaction mechanism.
- Use of the algorithm for two concurrent payment processes shows: the concurrent execution of processes without reservation but using shared resource may lead to incorrect execution of processes, but processes with reservation cannot lead to incorrect concurrent execution of processes.

The algorithm was used for identifying risks in e-commerce solutions even if business processes are defined informally, though at such a level of accuracy that it is possible to determine the feasibility of all process scenarios and to calculate results of scenario executions. When process execution correctness conditions are formulated, it is possible to determine each scenario the compliance of its execution result with the process

correctness conditions. The obtained scenario execution results allow identifying the business process risks.

Although problems of concurrent business process execution are effectively addressed in different DBMS, the issue of correct concurrent process execution in distributed systems is still open. If the architecture of the system does not include usage of a central data base, the research topic remains topical, and further research will be carried out on it.

Acknowledgments

The research leading to these results has received funding from the research project "Competence Centre of Information and Communication Technologies" of EU Structural funds, contract No. 1.2.1.1/18/A/003 signed between IT Competence Centre and Central Finance and Contracting Agency, Research No. 1.6 “Concurrence analysis in business process models”.

References

Al-dweeri, R. M., Obeidat, Z. M., Al-dwiry, M. A., Alshurideh, M. T., Alhorani, A. M. (2017). The impact of e-service quality and e-loyalty on online shopping: moderating effect of e-satisfaction and e-trust. *International Journal of Marketing Studies*, **9**(2), 92-103. http://doi.org/10.5539/ijms.v9n2p92.

Barzdins, J., Bicevskis, J., Kalnins, A. (1975). Construction of complete sample systems for correctness testing. In *Mathematical Foundations of Computer Science*. Berlin: Springer. pp. 1-12.

Bicevskis, J., Karnitis G. (2020). Testing of Execution of Concurrent Processes. In *International Baltic Conference on Databases and Information Systems* (pp. 265-279). Springer, Cham. https://doi.org/10.1007/978-3-030-57672-1_20.

Bicevskis, J., Nikiforova, A., Karnitis, G., Oditis, I., Bicevska, Z. (2021). Risks of Concurrent Execution in E-Commerce Processes. *Proceedings of the 16th Conference on Computer Science and Intelligence Systems,* M. Ganzha, L. Maciaszek, M. Paprzycki, D. Ślęzak (eds). ACSIS, Vol. **25**, pages 447–451 DOI: http://dx.doi.org/10.15439/2021F70

Bezes, C. (2016). Comparing online and in-store risks in multichannel shopping. *Internat. Journal of Retail & Distribution Management.* https://doi.org/10.1108/IJRDM-02-2015-0019.

CERT (2020). How to recognize a secure Internet store (In Latvian) https://cert.lv/lv/2020/11/ka-atpazit-drosu-interneta-veikalu.

Forsythe, S. M., Shi, B. (2003). Consumer patronage and risk perceptions in Internet shopping. *Journal of Business research*, **56**(11), 867-875.

Guzek, M., Danoy, G., Bouvry, P. (2013). System design and implementation decisions for ParaMoise organizational model. *Proceedings of the Federated Conference on Computer Science and Information Systems*, pp. 999-1005.

Huseynov, F., Yıldırım, S. Ö. (2016). Internet users’ attitudes toward business-to-consumer online shopping: A survey. *Information Development*, **32**(3), 452-465.

IntelliTest, (2019). Overview of Microsoft IntelliTest. https://docs.microsoft.com/en-us/visualstudio/test/intellitest-manual/?view=vs-2019.

Izogo, E. E., Jayawardhena, C. (2018). Online shopping experience in an emerging e-retailing market. *Journal of Research in Interactive Marketing.* Vol. 17, pp. 379-392.

Nikiforova, A., Bicevskis, J., Karnitis, G. (2020). Towards a Concurrence Analysis in Business Processes. In *2020 Seventh International Conference on Social Networks Analysis, Management and Security* (SNAMS) (pp. 1-6). IEEE. http://dx.doi.org/10.1109/SNAMS52053.2020.9336566.

Otika, U., Olise, E., Oby, O. B. (2019). Risk Perceptions and Online Shopping Intention among Internet Users In Nigeria. *Global Journal of Management And Business Research.* Vol. XIX, pp. 1-12.

O'Callaghan, E. (2017). The International Review of Retail, Distribution and Consumer Research. *The International Review of Retail, Distribution and Consumer Research*. Vol. 27, no. 5, pp. 435–436. https://doi.org/10.1080/09593969.2017.1391957.

Pappas, N. (2016). Marketing strategies, perceived risks, and consumer trust in online buying behaviour. *Journal of retailing and consumer services,* **29**, 92-103. https://doi.org/10.1016/j.jretconser.2015.11.007.

Rita, P., Oliveira, T., Farisa, A. (2019). The impact of e-service quality and customer satisfaction on customer behavior in online shopping. *Heliyon,* **5**(10), e02690. https://doi.org/10.1016/j.heliyon.2019.e02690.

Sellami C., Baron M., Bechchi M., Hadjali A., Jean S., Chabot D. (2021). Towards a Unified Framework for Computational Trust and Reputation Models for e-Commerce Applications. In: Cherfi S., Perini A., Nurcan S. (eds) Research Challenges in Information Science. RCIS 2021. *Lecture Notes in Business Information Processing*, vol **415**. Springer, Cham. https://doi.org/10.1007/978-3-030-75018-3_44

Sheikhan, M., Ahmadluei, S. (2013). An intelligent hybrid optimistic/pessimistic concurrency control algorithm for centralized database systems using modified GSA-optimized ART neural model. *Neural Computing and Applications*, **23**(6), 1815-1829.

Suryavanshi, R., Yadav, D. (2012). Modeling of multiversion concurrency control system using Event-B. In *2012 Federated Conference on Computer Science and Information Systems* (FedCSIS) (pp. 1397-1401). IEEE.

Tanenbaum, A. S., Van Steen, M. (2007). Distributed systems: principles and paradigms. Prentice-Hall.

Thakur, R., Srivastava, M. (2015). A study on the impact of consumer risk perception and innovativeness on online shopping in India. *International Journal of Retail & Distribution Management*. Vol. 43 No. 2, pp. 148-166. https://doi.org/10.1108/IJRDM-06-2013-0128.

van Beest, N., Bucur, D. (2013). Continuous correctness of business processes against process interference. In 2013 IEEE 6th International Conference on Service-Oriented Computing and Applications (pp. 110-117) https://doi.org/10.1109/SOCA.2013.39.

Wai, K., Dastane, O., Johari, Z., Ismail, N. B. (2019). Perceived risk factors affecting consumers' online shopping behaviour. *The Journal of Asian Finance, Economics and Business*, **6**(4), 246-260. https://dx.doi.org/10.2139/ssrn.3498766.

WEB (a). Ecommerce Statistics. https://ecommerceguide.com/ecommerce-statistics.

Received November 22, 2021, revised December 19, 2021, accepted December 21, 2021