Collaborative Systems – Finite State Machines

Ion IVAN, Cristian CIUREA, Sorin PAVEL
Economic Informatics Department,
Academy of Economic Studies, Bucharest, Romania
ionivan@ase.ro, cristian.ciurea@ie.ase.ro, pavelsorin@gmail.com

In this paper the finite state machines are defined and formalized. There are presented the collaborative banking systems and their correspondence is done with finite state machines. It highlights the role of finite state machines in the complexity analysis and performs operations on very large virtual databases as finite state machines. It builds the state diagram and presents the commands and documents transition between the collaborative systems states. The paper analyzes the data sets from Collaborative Multicash Servicedesk application and performs a combined analysis in order to determine certain statistics. Indicators are obtained, such as the number of requests by category and the load degree of an agent in the collaborative system.
Keywords: *Collaborative System, Finite State Machine, Inputs, States, Outputs*

1 Finite state machines

In [1] are defined the finite state machines as autonomous systems, which are evolving at a time, automatically, depending on the signals applied at that moment and the state in which the system is. The finite state machines are inspired from reality, where everything has a limited life cycle. This life cycle means an initial state, an intermediary state and a final one.
In [2] is considered that a finite state machine is a system with a finite number of states, having a model of behavior composed by states, transitions and actions. A state stores information about the past, meaning that reflect the changes from the system initialization to the present. A transition involves a change of state and is described by a condition that must be satisfied in order to start the transition. An action is a description of an activity, which must be performed at a given time.
A finite state machine is represented with the state diagram, given by the transition table from one state to another.

Table 1. The state diagram for a finite state machine

Condition/State	State 1	State 2	...	State i	...	State n
Condition 1						
Condition 2		State 3				
...						
Condition j				State k		
...						
Condition m						

As Table 1 represents, the finite state machine goes from state 2 to state 3 after the condition 2 is triggered. It also goes from state *i* into *k* by applying the condition *j*.
Figure 1 is a representation of a finite state machine with two states, namely *closed* and *opened*, passing from one state to another being accomplished by commands *open* and *close*.

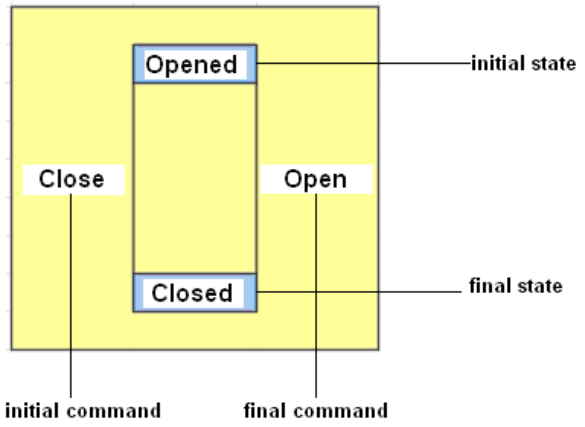

Fig. 1. Finite state machine

In the case of finite state machine from Figure 1, the output signal is not linked to the state in which the system is, but to a certain transition. The system passes from one state to another depending on the initial state and input signal applied [1].

Finite state machines are classified as *acceptors* and *transducers*, deterministic and nondeterministic. In case of the deterministic state machines, from each state shall be exactly one transition for each existing entry. In case of the nondeterministic state machines, for a given state and some input, there are several transitions with non-zero probability of producing, or none, with zero probability.

Starting from the classical theory of finite state machines, a specific formalization is made. The representation of an acceptor finite state machine is achieved by a quintuple [2]:

$< a_1, a_2, a_3, a_4, a_5 >$, where:

a_1 – the input alphabet containing a finite and non-empty array of symbols;

a_2 – a finite and non-empty array of states;

a_3 – initial state, element of a_2;

a_4 – transition function $a_4 : a_2 \times a_1 \rightarrow a_2$;

a_5 – the final states array, a component of a_2.

Finite state machines are characterized by the finite character of the input alphabet, the output alphabet and the set of states. The transducer finite state machine is a sextuple:

$< b_1, b_2, b_3, b_4, b_5, b_6 >$, where:

b_1 – the input alphabet containing a finite and non-empty array of symbols;

b_2 – the output alphabet, a finite and non-empty array of symbols;

b_3 – a finite and non-empty array of states;

b_4 – initial state, element of b_3;

b_5 – transition function $b_5 : b_3 \times b_1 \rightarrow b_3 \times b_2$;

b_6 – the output function.

A complex finite state machine is defined by a set of states $\boldsymbol{S}$, an input $\boldsymbol{I}$ and an output $\boldsymbol{E}$. Finite state machines increase in flexibility as it diversified the set of input symbols and the set of states.

In the field of applied computer science, finite state machines are used in modeling the behavior of applications, systems design, software engineering, compilers, and the study of formal languages [2].

2 Collaborative systems with finite number of states

A collaborative system must be treated as a finite state machine, because the portal of a collaborative system is a finite state machine.

Collaboration means more than two agents working together. It requires defining a shared goal and, in order to achieve this goal, the agents should create an agreement upon their courses of actions. Such an agreement is only achievable through negotiation [3].

It is considered the finite set of states that a collaborative system passes trough, namely $\boldsymbol{S_1, S_2, ..., S_n}$. The transfer matrix from the state $\boldsymbol{S_i}$ to state $\boldsymbol{S_j}$ is achieved by a message, a command $\boldsymbol{C_{ij}}$ or a document $\boldsymbol{D_{ij}}$.

A *bank* collaborative system passes through the following states: S_1 – opened, S_2 – receiving money, S_3 – credit acceptance, S_4 – issuing money, S_5 – exchange, S_6 – closed. Table 2 presents transitions between the six states of the *bank* collaborative system.

Table 2. Transition between the states of the *bank* collaborative system [4]

	S_1	**S_2**	**S_3**	**S_4**	**S_5**	**S_6**
S_1	null	yes	yes	yes	yes	yes
S_2	yes	null	yes	yes	yes	yes
S_3	no	yes	null	yes	yes	yes

S_4	no	yes	yes	null	yes	yes
S_5	no	yes	yes	yes	null	yes
S_6	yes	no	no	no	no	null

Switching from one state to another involves providing the system with an output. For an usual collaborative system is not possible to switch from state S_i to state S_j, for whatever i and j in the range $1..n$. The system transits from one state to another, but it doesn't pass from each state to all others. To exemplify this situation, it is considered a simple collaborative system represented by a freight warehouse. Possible states of this system are: opened, closed, supply, sales. Transition from *opened* to *closed* state is achieved through *close warehouse* command, the transition between *supply* and *opened* by *open warehouse* command. From *opened* and *supply* states to *sale* state passing is done through *merchandise* command.

To this kind of collaborative system probabilities are assigned, such as: probability of passing from state S_i to state S_j, the probability of obtaining output x in passing from one state to another. These probabilities are conditioned by a number of factors such as: commodity stocks and storage capacity, working program, client portfolio and the number of daily orders. At one point, the probability that the system goes from *opened* to *closed* state is influenced by the following situations:

- is the end of the daily working hours;
- freight stock is exhausted, following the intermediate state *closed* and then the final state *supply*;
- customer orders are no longer honored for various reasons.

Probabilities of these transitions are useful in determining the frequency of occurrence of different system states at a time. It is not possible to change from the state *supply* in the same state of *supply*. In certain circumstances, a large order from a client for example, the system passes from one *supply* state to another to meet that customer order. Probability of such a situation is quite small, given the fact that after a supply, the warehouse is loaded at the maximum capacity.

Collaborative systems differ one from each other in complexity. The problem of complexity is discussed similarly to the problem of simplicity. Software complexity is a new concept that requires careful definition to measure the level of complexity, in order to compare systems [5].

The complexity indicators allow comparisons of complex collaborative applications and portals. Homogeneity of portal components generates stable variations of the complexity indicators.

The complexity of the portal, having the structure represented by a graph with a fixed number of arcs and nodes, is equivalent with the complexity of a finite state machine.

The portal corresponds to collaborative system description and collaborative system complexity is approximately equal to the complexity of finite state machine, which is approximately equal to the complexity of the portal:

$CCS \sim CFSM \sim CP$, where:

CCS – complexity of collaborative system;

CFSM – complexity of finite state machine;

CP – complexity of the portal.

In [6] is considered that the software for space applications is a complex system with several components. Such software uses finite state machines in order to model the software specification from which test sequences are generated for a black box test approach.

3 The collaborative banking system as a finite state machine

In [7] is proposed a composition algorithm in order to solve the task of designing a collaborative business process while respecting a set of primary and recovery goals. In this model, each business process is described as a finite state machine.

Many scientific workflows are collaborative, because they result from some collaborative

research projects that involve a number of geographically distributed organizations. In [8] is achieved the model of a scientific workflow using a hierarchical state machine. There are presented techniques for verifying and controlling information propagation in scientific workflow environments based on hierarchical state machines.

A collaborative banking system, *CBS*, seen as finite state machine, is formalized as $<CB_1, CB_2, CB_3, CB_4, CB_5, CB_6>$ and is characterized by:

CB_1 – the input alphabet, containing an array of input messages that determine the conduct of activities and the coverage of process steps;

CB_2 – the output alphabet, defined as messages that accompany the finished products, services;

CB_3 – a finite and non-empty array of states that require action, consumption of resources, operations, equipment, people;

CB_4 – the initial state, element of CB_3;

CB_5 – the transition function $CB_5 : CB_3 \times CB_1 \rightarrow CB_3 \times CB_2$;

CB_6 – the output function.

The multitude of messages consists of homogeneous subsets of messages in relation to one or more criteria. The message lot is composed of subsets $SM_1, SM_2, ... SM_K$. The array of states consists of states $S_1, S_2, ... S_n$. The array of outputs: $E_1, E_2, ... E_m$.

If the system is in the S_i state and receives the message M_i, then it will go to S_t state. When changing to S_t state, it will issue the output E_r.

The *bank* collaborative system has the following entries: M_1 - cash deposit, M_2 - cash withdrawal, M_3 - create account, M_4 - payment making.

Consider S_0 - the standby mode of the *bank* collaborative system. The combination (M_1, S_0) determines the system transition to state S_7 representing depositing money. It is applied a certain procedure for this activity to deposit cash in customer accounts.

The S_7 state generates E_{12} output, receipt handed to the customer confirming the depositing money in the account.

It is considered Px_{ij} the probability of the *bank* collaborative system to provide output ***x*** when shifting from the state S_i to state S_j. The Px_{ij} probability of system to provide output x when changing from S_i to state S_j is determined using the relationship:

$$Px_{ij} = \frac{CFx}{CPx}, \text{ where:}$$

Pxij – the probability that the collaborative system provides output x when shifting from S_i state to S_j state

CFx – number of favorable cases to obtain the output *x* passing from S_i to S_j;

CPx – number of possible cases to obtain output *x* passing from S_i to S_j.

It is considered the collaborative banking system component represented by Multicash electronic payment service.

The array of inputs of the electronic payment system consists of:

MM_1 - payment orders in RON;

MM_2 - payment orders in foreign currency;

MM_3 - payment orders to state budget;

MM_4 - text files for direct debit;

MM_5 - LORO accounts payments.

The array of outputs contains the following elements:

ME_1 - daily account statements;

ME_2 - interim statements;

ME_3 - treasury bulletins;

ME_4 - electronic confirmation of payments made to customs;

ME_5 - messages of rejection;

ME_6 - notifications;

The array of system states consists of:

MS_1 - in standby;

MS_2 - processing transactions;

MS_3 - in course of statements delivery;

MS_4 - in maintenance.

The Multicash service is a collaborative system that acts like the finite state machine. Combination (MM_1, MS_2) leads to output ME_5 and it determines system passage to the MS_3 state. The (MM_4, MS_2) association leads to ME_2 output and it determines the state transition to MS_3.

Figure 2 presents the input sets, the outputs and the states for the Multicash collaborative system:

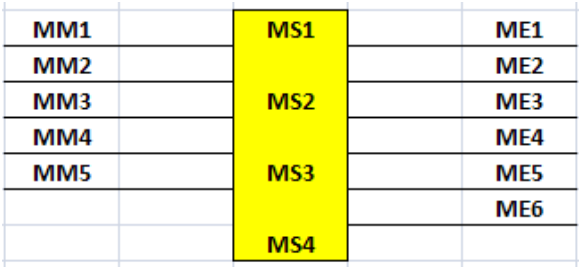

Fig. 2. The Multicash collaborative system

Starting from the definitions of transducers and acceptors finite state machines, it is demonstrated that the collaborative system, represented by the Multicash service, is a finite state transducer defined as:
< MM, ME, MS, MS1, F, E >, where:
MM – non-empty finite set of inputs;
ME – non-empty finite set of outputs;
MS – non-empty finite set of states;
MS1 – initial state;
F – transition function *F : MS x MM –> MS x ME*;
E – output function.
In [1] is considered that two finite state machines are equivalent if the two are unidentifiable, meaning that applying the same automatic entry to the input of both machines, they both produce the same sequence of outputs.
In the case of banking system as a finite state machine, in addition to the array of inputs, outputs and states, there is a set of procedures and regulations which determine certain restrictions. These restrictions prohibit the execution of activities such as: payment of a loan with money from another credit or credit given to a person who has not previously paid another loan. Based on these regulations and procedures, through restrictions imposed by regulations it is prevented any loss caused by the shift of the system in those states that led to the earlier losses. This implies that the banking system, seen as finite state machine, is dynamic, evolving and generating profits.

4 Operations on very large virtual databases as finite state machines

Collaborative systems involve daily transactions and getting data from multiple data sources. In the case of national or transnational systems, the circulated data amounts to large and very large orders, over 10^7 datasets.
Computerization of modern society, citizen-oriented software distribution and promulgation of new IT laws has led to applications that work with large data sets:

- telecommunications operators record each call or message within the network for a period of six months;
- internet and e-mail providers record accessed sites for each IP address in its administration, together with the exact date of access and data about each email message;
- government keep track of different payments for millions of people;
- national providers of utilities – gas, electricity etc. – process hundreds of millions of annual consumer bills;
- online search engines integrate content management of billions of sites;
- online banks services keep track of millions of transactions and facilitate e-payment trades.

Situations mentioned above involve many simultaneous users, using very large datasets, $10^7 \div 10^{10}$ sets, and applications for data management. Due to the large quantities of datasets to be processed, applications acquire specific properties and functionalities.
Data collections represent, but are not limited to: databases, collections of text files/XML files/multimedia, data warehouse, or any combination thereof. Administration requires specialized tools to harmonize the specific hardware and software aspects of large datasets.
Very large data collections along with the software applications that use them are seen as finite state machines as they:

- form a system with inputs, outputs and processing units; inputs consist of data, documents and measurements describing people, events or relationships; outputs are numerical results, documents or information objects valuable to the end user; processing units are algorithms, functions and procedures to convert input data into results requested by the user;

- are found in one state at a time; the state is clearly specified and known generating a certain behavior of both the application and the final user that knows the current status of the machine;
- are dynamic in terms of giving a response to actions taken upon them and effectuating state changes; transitions are triggered by some conditions affecting the entire database and allowing passage from one state to another.

In the making of and working with a very large database (VLDB) in banking, the database passes these states:

- formalization, the state in which the database is designed or expanded by adding new tables or a mix of existing database; formalization must take into account the restrictions of integrity already rolling and must match with the real world;
- population, the state in which the database accumulates data from multiple sources according to the default format so that the database integrity remains intact;
- consultation, the state in which the database is interrogated, searched while data is extracted and processed;
- clustering, condition in which the database is reorganized so that it best responds to users requests for consultation.

Transitions from one state to another are triggered by conditions and commands, as in Table 3.

Table 3. Commands that trigger state transitions in a VLDB

States	Formalization	Population	Consultation	Clusterization
Formalization	-	Build_DB	Consult_DB	ClusterizeDB
Population	DB_Schema	-	Consult_DB	ClusterizeDB
Consultation	DB_Schema	RetrieveD	-	ClusterizeDB
Clusterization	DB_Schema	RetrieveD	Consult_DB	-

Thus, a command is applied to several initial states, but will lead to only one final state. Depending on the condition the database is in, the condition will be fulfilled to trigger the transition to the final state.

There are considered N data sources, representing entry points of data or existing databases: S_1, S_2, ..., S_N. Data sources record data under the same format. It forms a very large virtual database by logical linking of data from different sources, as shown in Figure 3.

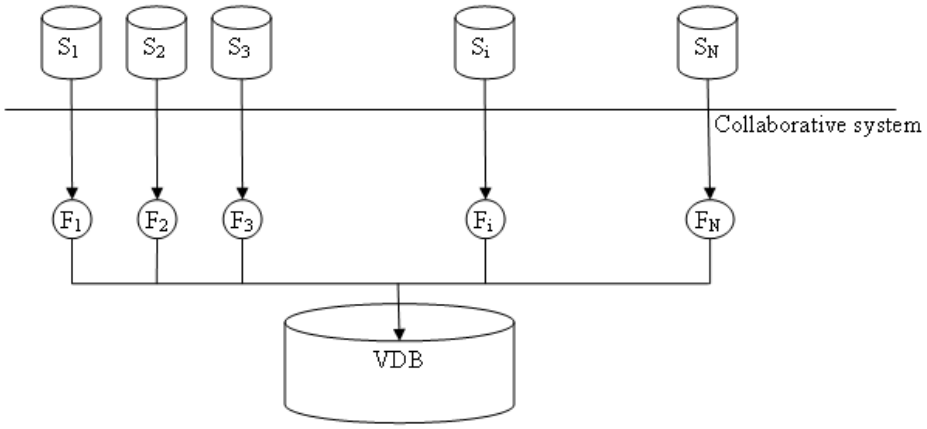

Fig. 3. The very large virtual database

The very large virtual database contains descriptions of data, without their copy. Data

is physically contained in the data source, but are united under the same logical common cover.

Operations performed on very large databases must take into account their characteristics, and should not adversely affect the terms:

- homogeneity of data, before and after operations;
- granularity of datasets, in relation to the community they belong;
- integrity of datasets, regarding their information content;
- brevity of datasets, meaning the minimization of descriptive information.

As collaborative systems and very large databases are treated as finite state machines, operations get new values:

- the results of an operation differs depending on the state in which the machine is, and hence the data; therefore, the state of data must first be consulted to ensure that they are in a fit state to run an operation; the results presented without specifying the condition in which was performed the operation are not relevant and not reliable, if the state of data is not able to produce quality results, the operation must be preceded by one or more transitions until a suitable state is founded;
- operations change the state of very large databases and, therefore, of the data contained; there are operations that act as transitions; such operations do not take into account the initial state of data in the sense of initial condition, but aimed to change the state of very large databases; the transition operations will not adversely affect the quality characteristics, but, instead, they are preparing them to generate relevant results;
- the time for making an operation is longer due to finite computer resources used for processing very large databases; the characteristic of finite state machines is that allocate the same number of hardware and software resources, regardless the amount of data that must process it; in the case of very large databases, this adversely affect the working time which increases exponentially with the size of the database.

Operations on very large databases, seen as finite state machines, processes data according to initial state of them and, in some cases, play the transition role.

It is considered the database of Collaborative Multicash Servicedesk - CMS application, in which are stored the requests of bank customers, regarding the problems that they have in using the Multicash electronic payment service.

The database of Collaborative Multicash Servicedesk application is designed like a virtual database, in which data are taken from different databases of Raiffeisen Bank information system. There is a database from which are taken the customers names and accounts, from another database are extracted the users details and the database of CMS application contains all the data aggregated.

The CMS application is used effectively within Raiffeisen Bank, in its database being introduced over two thousand requests per month. The application, having the structure represented in Figure 4, can be seen as a finite state machine that works with very large data volumes.

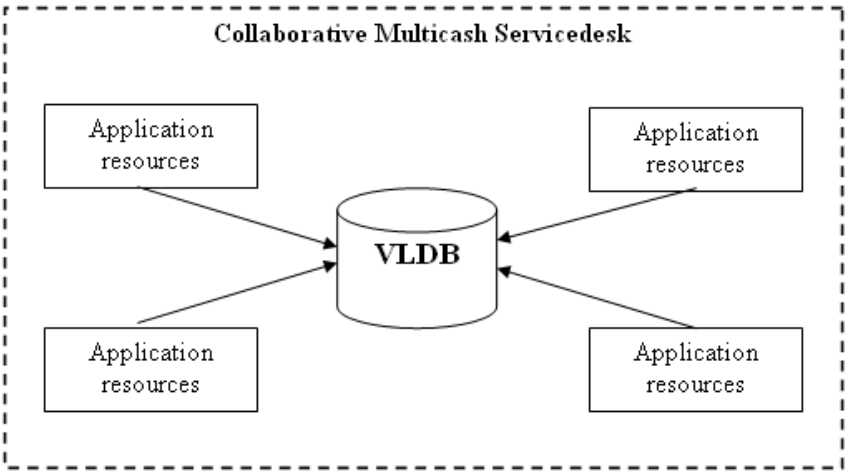

Fig. 4. The distributed CMS structure

Initial data volume is used to estimate the physical resource requirements for data storage and the workload required for data preparation and database creation. When estimating the volume of a large virtual database, it is necessary to be considered as more elements so that whatever changes will occur to the virtual database, it's structure remains stable.

The Collaborative Multicash Servicedesk application is structured in two modules:

- the module for online registration of bank customers requests;
- the module for recording phone requests by Multicash Helpdesk analysts.

In the module for online registration of bank customers' requests, each customer receives from the bank a username and password with which he will authenticate in the application. The associated customer interface allow the customer to send a written request to the Helpdesk department, by framing the issue in the appropriate category and subcategory, but also to register a priority request in exchange of a fee.

In the module for recording phone requests by Multicash Helpdesk analysts, after authentication in the application, the analyst see the page from which is made the registration of requests in the database.

The situation of requests on categories, recorded in the period January 01 to June 30, 2010, is presented in Figure 5.

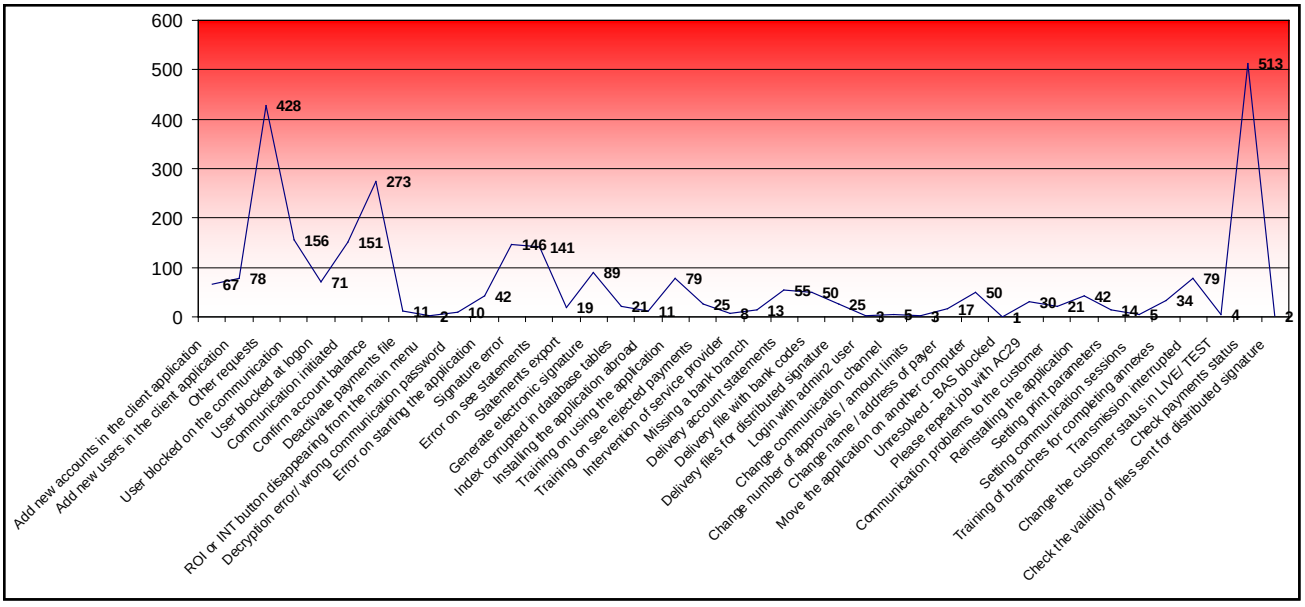

Fig. 5. The number of requests by category

Having the database of all customers' requests, it is realized the analysis of the types of problems faced by Multicash service users and are determined the strategies to address each customer, according to the history of problems he encountered.

According to the graphic representation in Figure 5, the first three categories with the biggest number of requests are *Checking payments status*, *Other requests* and *Confirm account balance*. The difference between the number of requests registered on these categories and the number of requests from other categories is significant. To reduce the number of requests in these categories should be:

- improved the Multicash service, in order to allow real time view of operations performed;
- reviewed the requests recorded in the category *Other requests* for their reclassification in existing categories or in order to create new categories of problems;
- updated accounts balances in real time.

The available states for CMS application, seen as finite state machine, are as follows: $MCMS_1$ – in standby, $MCMS_2$ – generating reports, $MCMS_3$ – recording requests, $MCMS_4$ – in maintenance.

The application transition from one state to another is given by a command or a set of commands. The application passes from the $MCMS_1$ state to the $MCMS_3$ state, but can go through $MCMS_2$ state to $MCMS_4$ state.

From the database of CMS application data sets are identified and is performed a combined analysis to determine certain statistics. The combined analysis involves correlations between data sets, for the calculation of quality indicators.

It considers A_1, A_2, A_3 and A_4 the names of four analysts who actually work with the CMS application within the Multicash Helpdesk department of Raiffeisen Bank.

Is determined the load degree of each agent in the system and is made a redistribution of operations so that do not exist a situation in which an agent is overloaded and another do not have enough operations which fill the working time.

5 Conclusions

In a bank, million of transactions take place daily, representing transfers between existing accounts, opening new accounts, building or liquidation of deposits, loans giving. These transactions require the existance of an advanced database management system and an integrated computer system. Electronic transactions that takes place in a bank are saved in databases and are never deleted. Each bank has well tuned procedures for backup and disaster recovery, to avoid the loss of database records, even for natural disasters events.

The Collaborative Multicash Servicedesk application is a collaborative auto-adaptive system that allows auto-configuration based on information entered by users. The CMS application adapts to input data and change the components, so as to provide maximum utility and support to its users, regardless the category they belong to.

Collaborative systems are concrete models of finite state machines, having the properties of these automata and being treated accordingly in order to increase their effectiveness.

Acknowledgements

This article is a result of the project POSDRU/6/1.5/S/11 „Doctoral Program and PhD Students in the education research and innovation triangle". This project is co funded by European Social Fund through The Sectorial Operational Programme for Human Resources Development 2007-2013, coordinated by The Bucharest Academy of Economic Studies, project no. 7832, Doctoral Program and PhD Students in the education research and innovation triangle, DOC-ECI.

References

[1] E. Nicolau and A. Popovici, *Algoritmi. Automate finite. Calculatoare electronice*, Editura Stiintifica, Bucharest, 1970.

[2] F. Wagner, *Modeling Software with Finite State Machines: A Practical Approach*, Auerbach Publications, 2006.
[3] S. O. Oguz, A. Kucukyilmaz, T. M. Sezgin and C. Basdogan, "Haptic negotiation and role exchange for collaboration in virtual environments," *2010 IEEE Haptics Symposium*, 25-26 March 2010, pp. 371-378.
[4] I. Ivan, C. Ciurea and S. Pavel, "Very Large Data Volumes Analysis of Collaborative Systems with Finite Number of States," *Journal of Applied Quantitative Methods*, Vol. 5, No. 1, 2010.
[5] I. Ivan and F. Alecu, *Structuri HTML*, ASE Publishing House, Bucharest, 2005.
[6] A. O. Arantes, N. L. Vijaykumar, V. A. de Santiago and D. Guimaraes, "Test Case Generation for Critical Systems through a Collaborative Web-Based Tool," *2008 International Conference on Computational Intelligence for Modelling Control & Automation*, 10-12 Dec. 2008, pp. 163-168.
[7] J. Lemcke and A. Friesen, "Composing Web-service-like Abstract State Machines (ASMs)," *2007 IEEE Congress on Services*, 9-13 July 2007, pp. 262-269.
[8] Y. Ping, Y. Zijiang and L. Shiyong, "Formal Modeling and Analysis of Scientific Workflows Using Hierarchical State Machines," *IEEE International Conference on e-Science and Grid Computing*, 10-13 Dec. 2007, pp. 619-626.

Ion IVAN has graduated the Faculty of Economic Computation and Economic Cybernetics in 1970. He holds a PhD diploma in Economics from 1978 and he had gone through all didactic positions since 1970 when he joined the staff of the Bucharest Academy of Economic Studies, teaching assistant in 1970, senior lecturer in 1978, assistant professor in 1991 and full professor in 1993. Currently he is full Professor of Economic Informatics within the Department of Computer Science in Economics at Faculty of Cybernetics, Statistics and Economic Informatics from the Academy of Economic Studies. He is the author of more than 25 books and over 75 journal articles in the field of software quality management, software metrics and informatics audit. His work focuses on the analysis of quality of software applications.

Cristian CIUREA has a background in computer science and is interested in collaborative systems related issues. He has graduated the Faculty of Economic Cybernetics, Statistics and Informatics from the Bucharest Academy of Economic Studies in 2007. He is currently conducting doctoral research in Economic Informatics at the Academy of Economic Studies. Other fields of interest include software metrics, data structures, object oriented programming in C++ and windows applications programming in C#.

Sorin PAVEL has graduated the Faculty of Economic Cybernetics, Statistics and Informatics from the Bucharest Academy of Economic Studies in 2008. He is currently following Master's in Software Project Management and the Doctoral School in Economic Informatics, both at the Academy of Economic Studies.