Case Study of Self-Organization Processes in Information System Caching Components

Pavel Kurnikov[1], Nina Krapukhina[2]
National University of Science and Technology MISiS
IT and Automated Control Systems
Moscow, Russia

***Abstract*—Most modern Information Systems (IS) are designed with Object-Relational Mapping components (ORM). Such components bring down the number of queries to the database server and therefore increase the system performance. Caching mechanisms in software components are complex dynamic systems of open type. A simulation model of cache-application interaction has been created to assess the critical modes of the system. The authors suggest accumulating the cache elements states during the application run at discrete moments of time and present them as multi-variable time series. This work also suggests a method for reconstruction of phase-plane portrait of the system with multidimensional dynamics of the cache elements states. The article shows self-organization processes in caching components of information systems and illustrates the variability of system states for various initial conditions followed by transition to steady-state conditions. In particular, the paper illustrates dissipative structures as well as deterministic chaos with the complete determinism of queries in simulated information systems.**

Keywords—Caching mechanisms; ORM; phase space reconstruction; self-organization; dissipative structures; deterministic chaos

I. Introduction

Modern Information Systems use caching mechanisms, as the process of their design almost always use the technology for mapping relational data into object model [1]. This technology allows leveling down semantic gaps during system development, but also creates additional expenses in processor resource and memory consumption during the operational phase [2, 3, 4, 5].

Works [6, 7, 8, 9] demonstrated that caching mechanisms in IS are non-linear open-type systems. For instance, article [7] identifies the caching element functioning with the Brusselator model, described through a system of differential equations of high order with non-linear terms. Analytical solution of these systems is not always possible though. The research conducted in work [8] revealed statistical relation between dynamic system running modes and the time for processing user's query. The openness of the system lies in information exchange with the environment. The users working with the IS become sources of "energy", and their actions helps build up structural inhomogeneity of the system. In this case "energy" is the flow of queries to the server that enables modification of the IS. It is inevitable that this process results in cache build-up. Cache release is the zone for dispersion of energy which leads to smoothing the structural inhomogeneity of the system. Open system of such type can absorb external actions and remain in constant change at the same time.

Caching systems have feedback of both positive and negative types. Input flow of queries from a user can change the inner state of the system. In this case the system can give feedback to the input flow to increase or decrease its force, depending on the inner state of the system. In this regard it is important to take into account the development of processes in the caching system itself, as it is not only the external actions that make it change, but also inner processes that bring the system to self-organization.

In the previous works [4, 9] the authors demonstrated the possibility for extraction of cache system modes while IS model is running and expressing the results as multi-variable time series. Based on the research in works [10-14] the authors suggest using phase space reconstruction methods that are based on Takens embedding theorem [15]. This article aims at giving quality assessment to processes in the observed dynamic system. Attractor reconstruction is made on the basis of changes in caching system variables available for observation in time.

The rest of the paper is organized as follows: Section II briefly describes the caching mechanisms in ORM systems. Section III presents a simulation model of interaction between the caching mechanism and the application. Sections IV and V present the research results and their discussion, respectively.

II. Caching Mechanism

Modern ORM frameworks (object-relational mapping) build IS conceptual layer, and then every stored table is assigned an object from the conceptual model. Caching mechanism [16] handles a list of entities, each in one of the following states:

- Detached (from the cache).
- Added.
- Unchanged.
- Modified.
- Deleted.

The object state manager maintains the state of each entity in the cache – detached (from the cache), added, unchanged, modified, deleted and tracks their state transitions.

After the users' interaction with the information in IS the process of transformation of modified entities corresponding with the stored tables of the database is initiated by method SaveChanges. A list of changes is then formed for each entity. This list is formed straight from cache. Next, the list is transformed into a series of algebraic expressions (relational operations with the tables corresponding to the entities) to be realized on the database server [16]. After the transactions are complete, cache mode is synchronized with the new state of the database.

This article uses discrete-event model for simulating scenarios of multiple users interacting with the database via high-level interface [9]. Such a model allows simulating the load of several users on a server and serializes the cache modes in discrete moments of time.

III. Discrete-Event Model

Let us take as *R* (the data stored in the database) the multitude of stored states of the initial database. ORM creates an entities scheme *E* (entity schema) from the database scheme (relational schema). Let us assign entity *E* on the object layer of the application to each object from *R*. There is a cache vector *C* for each type of entity. Test on range of entities is defined by predicate formula from range F^E. Transformation between relational and object data is done through bi-direct connection illustrated by formulas 1 and 2.

$$\Delta R_i = \Delta UV(E_i, \Delta^{+/-}E_i), i = 1,..,I, \Delta^{+/-}E_i \neq \varnothing \tag{1}$$

where $\Delta^{+/-}E$ is a result of the test from range F^E on entity E_i.

Here the difference in entities state is assessed with *UpdateViews*, which shows how to extract information about the modified state of the stored database [16]. (In Step 1, view maintenance algorithms are applied to update views. This produces a set of delta expressions, UpdateViews, which tell us how to obtain delta tables from delta entities and a snapshot of Entities).

In the case when conceptual (entity) level test from *F* has not modified the entity due to the presence of the modification object in cache, there will be no query to the server. (no update is required).

$$\Delta E = QV(R), \tag{2}$$

here *QV – query views* transforms objects from relational scheme to entities; (The query views express entities in terms of tables).

The general rule which a user uses to work with the common resource *C* can be presented as:

$$C_i(t_k) = UpdateObjectCache(F_j^E(E_i, t_{k-1}) - F_j^E(E_i, t_k)),$$
$$j = 1,..., J, k = 1,...., K \tag{3}$$

This process can be compared to negative and positive feedback. If the list of entities changes received from cache after a certain input test is empty, there will be no query sent to the server. This equals restriction of input flow (negative feedback). On the other hand, if the result of a certain test run lead to cache flush, that can potentially increase the input flow, and this will result in extra costs for the server.

Let us define a user from range $U = \{u_1,...,u_M\}$. A user is defined by a subset of tests $F_{U_m}^E \subset F_E$ which he works with as a full group of disjoint events with probabilities $\langle P \rangle$ as well as a law of test distribution ω and time of work *T* with the system:

$$U = \langle F_U^E, P, \omega, T \rangle \tag{4}$$

In other words, user U_m is a source of events flow in [0, *T*]. The events flow is an external influence towards the system. Let us set a multi-layer density of probability for intervals between test completion time:

$$\omega_{U_m}(\tau_1,...,\tau_K), \tau_k = t_k - t_{k-1}, t_0 = 0, t_K = T \tag{5}$$

where ω_{U_m} is an *n*-dimensional density function for user U_m defining the law of random vector distribution $\|\tau_k\|, k = \overline{1,K}$ for user U_m.

Multi-dimensional random variable is simulated on an ECM with Neumann construction generalized for a multi-dimensional event [17]. The choice of test to be run by a user at the moment of time t_k is made based on a random discrete variable generated in the interval [0, 1]:

$$X = \begin{pmatrix} x_1 & ... & x_L \\ p_1 & ... & p_L \end{pmatrix}, \sum_{l=1}^{L} p_l = 1, \tag{6}$$

where $P = \{p_1, \ldots, p_L\}$ is a probability vector for a full set of events (tests from range F_U^E) with indexes $\{1, \ldots, L\}$.

Test with index x_l where $r_k \in \Delta_l$ will be performed at the moment in time t_k.

$$r \in \begin{cases} \Delta_1, & r \leq p_1 \\ \Delta_l, & p_1 + \ldots + p_{l-1} \leq r \leq p_l + \ldots + p_l, (l \geq 2) \end{cases} \tag{7}$$

where Δ_l is a part of a segment [0, 1] with length equal p_l

Server part storing the database is a shared resource for the system. Inner state of the system is cache vector *C*. This resource is also shared for every user from *U*.

Let us analyze the dynamics of cache vector *C*. For this we will inquire the state of vector *C* at [0, *T*] every Δ*t*. As a result we have a phase space described by *D*-dimensional time series $X_1, X_2, ..., X_D$ where $X_d = (x_{d,1}, x_{d,2}, ..., x_{d,K})$, $d = 1, 2, ..., D$, $K = T / \Delta t$, $x_{d,k}$ is the size of *d* element of cache *C* at moment in time *k*Δ*t*. $D = dim(C)$. For the systems containing *D* variables $x_d(t) = [x_{d,1}, x_{d,2}, \ldots, x_{d,N}]^T \in \mathrm{R}^D$, $d = 1,..., D$ phase space is suggested to be simulated with the method described in work [13]. This approach allows quality simulation of phase space from initial data, and it also has relatively low calculation complexity compared to the methods described in works [10, 11]. The process involves consecutive calculation of time lapses [18, 19] for each series. Correlation dimensions [20, 21] are found for each variable based on the calculated parameters.

IV. Results

The authors suggest using the previous results [4, 9] for simulating caching mechanisms in the IS. Architecture of the database on the server is illustrated by Fig. 1.

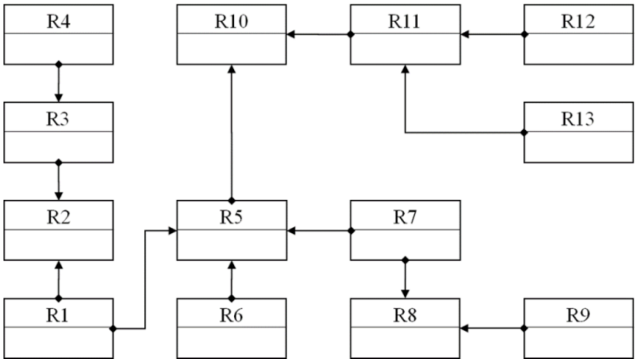


Fig. 1. Database Diagram.

The database consists of 13 tables with multiple links. Three basic tests were taken as functional base: adding data into a table or a set of tables (*insert_data*), removing data from a table or a set of tables (*remove_data*), and getting data from a table or a set of tables (*get_data*). Initial conditions of the model are the number of users working with the system simultaneously. Every user is described by a set of tests which he works with, as well as a function of their distribution in the interval [0, *T*]. A set of tests for each user is a group of disjoint events. Each event is defined by its probability. Another initial condition is the state of cache vector before the run of the model. Initial state of the database was the same for all models described in this work.

Let us look at the model with one user (*U0*) working with one database table (*R7*). Let us identify the set of test for the user (*U0*) as a sub-set {*insert_data*, *get_data*}. Time for modeling was 1000 seconds, Table I.

TABLE I. Initial Conditions for Model 1

Model	**Users**	**Tables**	**Distribution of basic tests**			**Time of simulation T, sec**
			Insert data	***Remove data***	***Get data***	
1	1	R7	0.5	0	0.5	1000

Dynamics for cache size associated with table R7 is shown in Fig. 2.

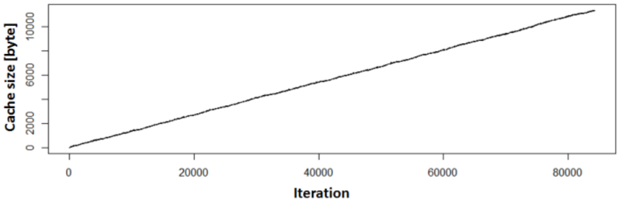


Fig. 2. Cache Dynamics for Table R7 (Model 1).

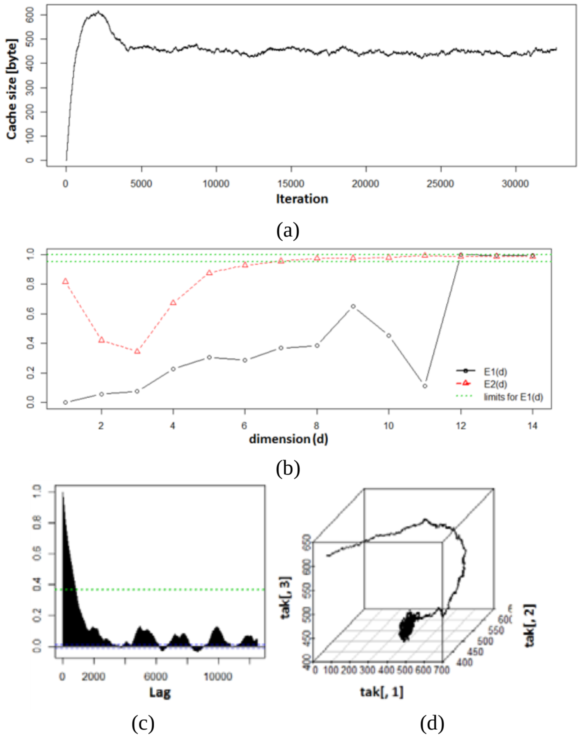
Cache dynamics for table *R7* is presented by linear growth of space. It is related to the fact that non-linear processes do not occur in single-user mode in the scenario with one object. Additionally, as framework 1 shows obvious domination of *insert_data* and *get_data* tests, each query for an entity is accompanied with caching the object. In 1000 seconds of simulation more than 80,000 queries to cache were executed. The total cache volume reached 11347 bytes. In other words, there is no reaction of the system to external factors, because there is no release of cache and input flow to the database server does not get cut. System does not compensate for these processes and stays in an unbalanced state, meaning there is no self-organization.

Let us simulate a model with three users. The object of the queries is still *table R7*. Let us add test *remove_data* to the *model 1* set of tests. New initial conditions are shown in *Table II*.

TABLE II. Initial Conditions for Model 2

Model	**Users**	**Tables**	**Distribution of basic tests**			**Time of simulation T, sec**
			Insert data	***Remove data***	***Get data***	
2	3	R7	0.3	0.35	0.35	1000

It is worth mentioning that Table II only shows the general distribution of test running in the model during time period T. Fig. 3 illustrates the cache growth dynamics, additionally there are visuals on results of calculations for correlation dimension and time lapse for the simulated attractor. All the calculations were done in RStudio in language R with non-linear Tseries package. Overview of numeral methods for this package is presented in work [22].



(a)

(b)

(c)

(d)

Fig. 3. Simulation Results for 3 users in Accordance with Initial Data for Table II: (a) Cache Dynamics for Table R7; (b) Auto-Correlation Function; (c) Embedding Dimention Calculation; (d) Attractor Visualisation.

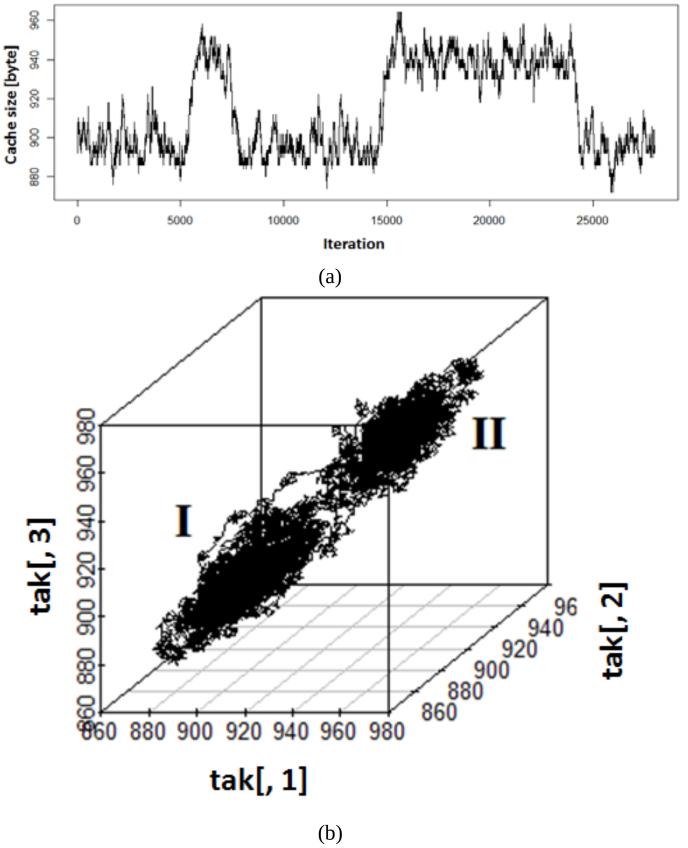
Here non-linear dynamics in cache behavior can be seen. This is due to parallel input flows of queries to the system and inner non-linear caching mechanism. Apart from that, this model shows stable increase of allocated memory, as there is a cache release mechanism at work. This mechanism is a zone for information dispersion, and as a result of its work the system structural inhomogeneity is smoothed. Open system of such type processes external influence, but remains in permanent change. Peak memory use in this case only reached 657 bytes. This system is in unstable stationary mode of work.

Let us add another influence to demonstrate the 'fragility' of this balance, as shown in Table III. In *model 3* the new external influence to caching mechanism is user *U4* with an only test *get_data*. This time the object of queries will be new table R8 linked to R7 by the key. Time of simulation was kept unchanged. Query *get_data* to object R8 linked to R7 will introduce changes to cache associated with object R7. This will lead to the increase in query flow to the system.

TABLE III. INITIAL CONDITIONS FOR MODEL 3

Model	**Users**	**Tables**	**Distribution of basic tests**			**Time of simulation T, sec**
			Insert data	***Remove data***	***Get data***	
3	4	R7, R8	0.3	0.35	0.35	1000

Cache dynamics for table R7, R8 is shown on Fig. 4.



(a)

(b)

Fig. 4. Simulation Results For 4 users in Accordance with Initial Data for Table III: (a) Shared Cache Dynamics; (b) Attractor Visualisation.

Cyclic transitions of the dynamic system from zone I to zone II are shown in Fig. 4. The system has two attractors. The received data is not stationary as the system is bi-stable. This is an open system, so part of external queries is redirected to cache release, which results in caching mechanism self-regulation.

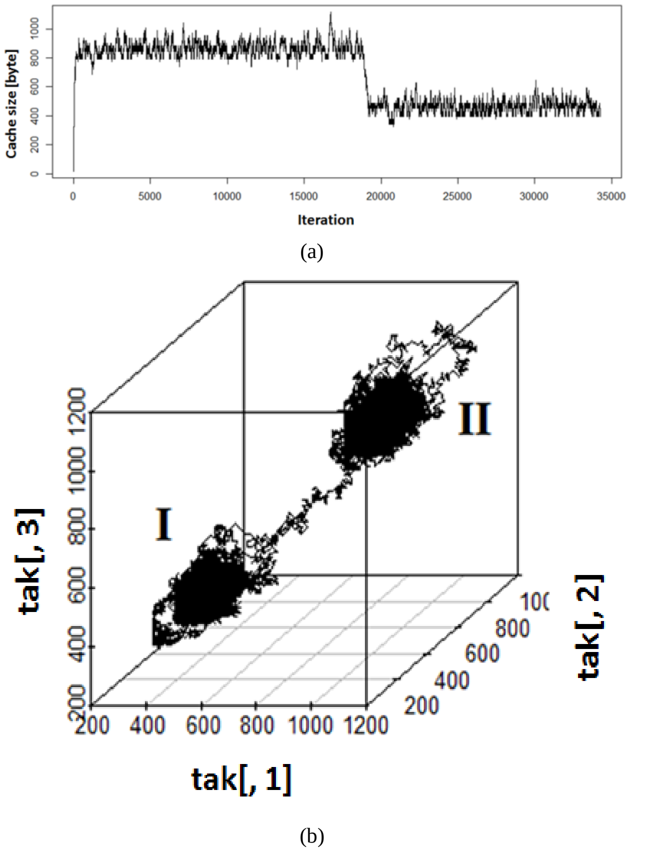
Let us increase input queries flow to the system by adding new users while keeping the test distribution and query objects the same, see Table IV.

TABLE IV. INITIAL CONDITIONS FOR MODEL 4

Model	**Users**	**Tables**	**Distribution of basic tests**			**Time of simulation T, sec**
			Insert data	***Remove data***	***Get data***	
4	10	R7, R8	0.3	0.35	0.35	1000

Increase in number of users from 4 to 10 affected the total number of completed transactions on the server and queries to cache (Fig. 5a). This did not though change the number of equilibrium conditions of the system (Fig. 5b).

Let us look at the system with two groups of users. Within its group each user will be described by a group of tests it interacts with, as well as a unique function of their distribution in the interval [0, *T*]. Let us also change the objects of the users' queries, see Table V.



(a)

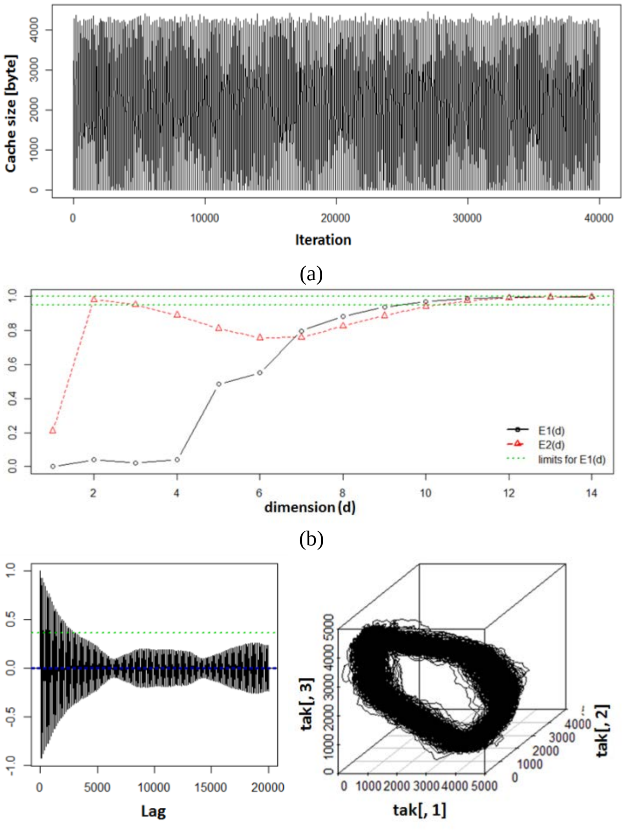
(b)

Fig. 5. Simulation Results for 10 users in Accordance with Initial Data for Table IV: (a) Shared Cache Dynamics; (b) Attractor Visualisation.

TABLE V. INITIAL CONDITIONS FOR MODEL 5

Model	**Users**	**Tables**	**Distribution of basic tests**			**Time of simulation T, sec**
			Insert data	***Remove data***	***Get data***	
5	1-4	R11 – R13	0.35	0.3	0.35	5000
	5	R1 – R4	0.2	0.4	0.4	

Here a group of users U1-U4 interacts with tables R11-R13. User U5 interacts with tables R1-R4.



(a)

(b)

(c)

(d)

Fig. 6. Simulation Results for 20 users in Accordance with Initial Data for Table V: (a) Shared Cache Dynamics; (b) Auto-Correlation Function; (c) Embedding Dimention Calculation; (d) Attractor Visualisation.

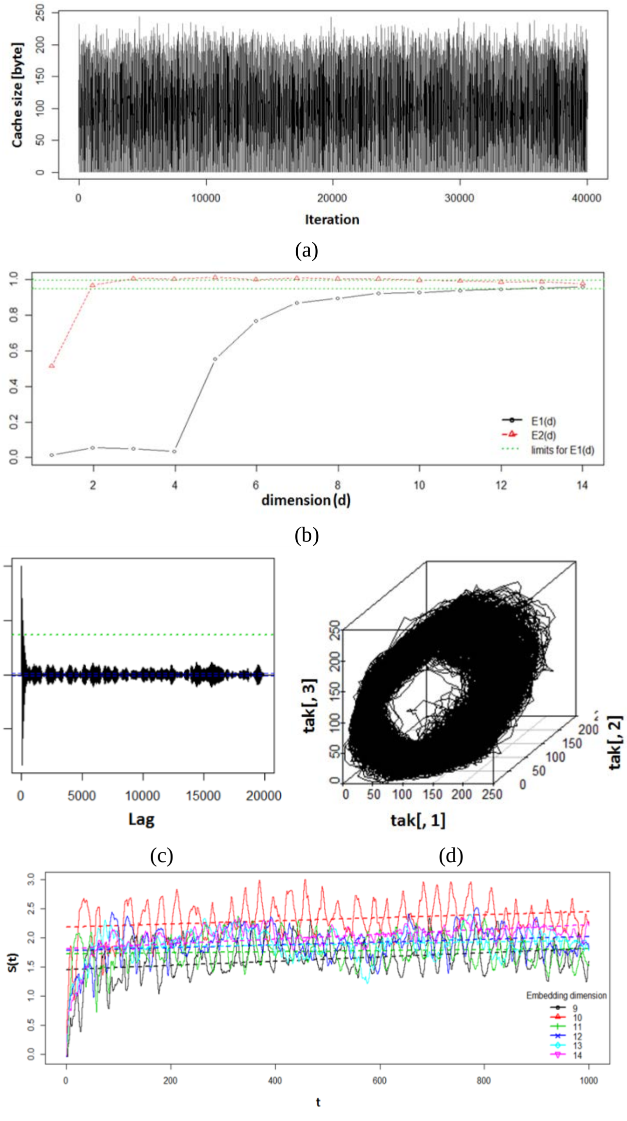
Despite the presence of inquiries to linked entities (R12, R13 and R1, R3, R4), Fig. 6d. shows closed-loop isolated trajectory in phase space corresponding with cyclic movement. This movement is asymptotically stable and in the process of self-organization the system will be attracted to this trajectory regardless of the initial cache state. Despite minor presence of chaotic processes caused by a large number of users interacting with the system, they do not disturb the end cycle. In other words, the system is far from the point of stochastic bifurcation.

Let us introduce additional inquiry objects to the system, as well as new groups of users for running the tests. The time of simulation will be kept unchanged, see Table VI.

TABLE VI. INITIAL CONDITIONS FOR MODEL 6

Model	**Users**	**Tables**	**Distribution of basic tests**			**Time of simulation T, sec**
			Insert data	***Remove data***	***Get data***	
6	1–4	R11–R13	0.35	0.3	0.35	5000
	5	R1–R4	0.2	0.4	0.4	
	6–14	R1, R5, R6	0.25	0.25	0.5	
	15–20	R1, R5, R6	0	0	1	

New groups of users U6-U14 and U15-U20 interact with tables R1, R5, R6, but the latter only accesses the database for information without making any changes.



(a)

(b)

(c)

(d)

(e)

Fig. 7. Simulation Results for 20 users in Accordance with Initial Data for Table VI: (a) Shared Cache Dynamics; (b) Embedding Dimention Calculation; (c) Autocorrelation Function; (d) Attractor Visualisation; (e) Estimating Maximal Lyapunov Exponent.

Information dispersed and injected into the system is strictly compensated, despite minor presence of chaotic processes caused by a large number of users interacting with the system. This points at dissipative type of the observed system. Fig. 7(e) demonstrates the estimates for maximal Lyapunov exponent [23, 24, 25] for space several special dimensions. Despite the high dispersion of this exponent, it has positive value which is another confirmation for the presence of chaotic processes.

V. Conclusion and Further Research

A discrete-event model was developed as a result of this research. The model allows simulation of various scenarios for users interacting with the database server through caching mechanisms of object-relative representation. Data on states of each system cache valuable in time were reflected as variables in multi-dimensional time series. Each component of the time series was assigned necessary parameters for phase space reconstruction. Analysis of trajectories data revealed the mode of work for simulated dynamic systems.

The authors demonstrated linear processes of cache dynamics for single user (*U0*) interacting with one database table (*model 1*). With the randomly requested data in test *get_data* and the absence of *remove_data* all the entries were fully cached. In case of collective work of users with the system, combined with non-linear connections in caching mechanism, the latter demonstrates complex non-linear dynamics both on macro- and micro-levels. This is the key element in self-organization process.

Unstable stationary dynamic mode was demonstrated for the system that three users were interacting with (*model 2).* Set of tests and distribution were selected in a way that the model does not show stable increase of allocated memory. It can be described as dissipation of information by the caching mechanism, which leads to the open system of this type processing external influence, while staying in permanent change.

Non-equilibrium of the stationary state of *model 2* was confirmed by the data from *model 3* run, where user *U4* with the sole test *get_data* became the additional external influence on the system. The system demonstrated bi-stable transitions between two attractors I and II, but it still remained in constant self-organization. Adding new users in *Model 4* affected the total number of transactions completed on the server during the simulation time *T*, but it did not affect the number of equilibrium conditions of the system.

Models 5 and 6 demonstrated presence of self-oscillating processes. High load on the caching mechanism was provided by several groups of users simultaneously interacting with most of the objects of the initial database. As a result, the system had stable limit cycle without a stable focus. Observations of *model 5* showed that convergence of phase trajectory does not depend on the initial state of cache. Besides, even though the users in *model 6* had determined behavioral rules, non-linear dynamic system was subject to determined chaos, which was revealed with calculating highest Lyapunov exponent. Despite this, such behavior of caching mechanism is an optimum option for multi-user regime of work with IS due to the system’s resistance to initial conditions.

To reproduce a larger number of information systems, one should expand the functional test base of user-application server interaction. Research into self-organization processes in caching components of information systems will allow comparative analysis of function composition at IS development stage and enable choice of a more stable version.

References

[1] S. Bhatti, Z. Abro, F. Rufabro. “Performance evaluation of java based object relational mapping tool,” Mehran University Research Journal of Engineering and Technology, 2013, pp. 159-166.

[2] A. Gruca, P. Podsiadło. “Performance Analysis of NET Based Object–Relational Mapping Frameworks,” International Conference: Beyond Databases, Architectures and Structures, Springer, Cham, 2014, pp.40-49.

[3] R. Jacobson, S. Misner, Microsoft SQL Server (TM) 2005 Analysis Services Step by Step, Microsoft Press, 2006.

[4] N. Krapuhina, P. Kurnikov, I. Tarkhanov. “Multi-criteria method of estimating performance ORM components in information systems,” Trudi Instituta sistemnogo analisa Rossiyskoy akademii nauk ISA RAN, 65(22), 2015, pp. 105-109.

[5] D. Zmaranda, L. Pop-Fele, C. Gyorodi, R. Gyorodi, G. Pecherle. “Performance comparison of crud methods using net object relational mappers: A case study,” International Journal of Advanced Computer Science and Applications, (11), 2020.

[6] S. Melnikov. “Formation of dissipative structures in ORM-components of high-loaded portals,” Modern problems of science and education. Surgery, 6(5), 2013.

[7] S. Kudzh, S. Melnikov. “Unsteady operation of a software system with the active “caching” to access the database,” Perspectives of Science and Education, 8(2), 2014, pp. 51-55.

[8] S. Melnikov. “Dissipative structures of the "cycle" in the software, which exists in the mode of multi-threaded execution of the executable code,” Modern problems of science and education. Surgery, 10(6), 2012.

[9] P. Kurnikov, N. Krapuhina. “Phase space reconstruction of high-loaded caching mechanism dynamics in information systems,” Journal of information technologies and computing systems, 2019, pp. 49-65.

[10] S. Garcia, J. Almeida. “Multivariate phase space reconstruction by nearest neighbor embedding with different time delays,” Physical Review E - Statistical, Nonlinear, and Soft Matter Physics, 72(2), 2005.

[11] Y. Hirata, H. Suzuki, K. Aihara. “Reconstructing state spaces from multivariate data using variable delays,” Physical Review E - Statistical, Nonlinear, and Soft Matter Physics, 74(2), 2006.

[12] J. Barnard, C. Aldrich, M. Gerber. “Embedding of multidimensional time-dependent observations,” Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics, 64(4), 2001.

[13] R. Wang, J. Gao, Z. Gao, X. Gao, H. Jiang, L. Cui. “Data Fusion Based Phase Space Reconstruction from Multi-Time Series,” International Journal of Database Theory and Application, 8(6), 2015, pp. 101-110.

[14] S. Palit, S. Mukherjee, D. Bhattacharya. “A high dimensional delay selection for the reconstruction of proper phase space with cross auto-correlation,” Neurocomputing, 113 (2013), pp. 49-57.

[15] F. Takens. “Detecting strange attractors in turbulence, Dynamical systems and turbulence,” Warwick 1980, Springer, Berlin, Heidelberg, 1981, pp. 366-381.

[16] A. Adya, P. Bernstein, S. Melnik. “Generation of query and update views for object relational mapping,” U.S. Patent No. 7,647,298. Washington, DC: U.S. Patent and Trademark Office, 2010.

[17] V. Bykov. “Digital modeling in statistical radio engineering,” STIN, 75 (1973), p. 10294.

[18] J. Theiler. “Spurious dimension from correlation algorithms applied to limited time-series data,” Physical Review A, 34(3), 1986, p. 2427.

[19] H. Kantz, T. Schreiber. “Nonlinear time series analysis,” vol. 7, Cambridge University Press, 2004.

[20] D. Turcotte. “Fractals and Chaos in Geology and Geophysics,” Cambridge University Press, 1992.

[21] G. Malinetskii, A. Potapov, A. Rakhmanov. “Limitations of delay reconstruction for chaotic dynamical systems,” Physical Review E, 48(2), 1993, p. 904.

[22] T. Parker, L. Chua. “Practical Numerical Algorithms for Chaotic Systems,” Mathematics of Computation, 56(193), 1991, pp. 377-379.

[23] L. Arnold, W. Kliemann, E. Oeljeklaus. “Lyapunov exponents of linear stochastic systems,” In Lyapunov exponents. Springer, Berlin, Heidelberg, 1986, pp. 85-125.

[24] M. Rosenstein, J. Collins, C. De Luca. “A practical method for calculating largest Lyapunov exponents from small data sets,” Physica D: Nonlinear Phenomena, 65(1-2), 1993, pp. 117-134.

[25] J. Eckmann, S. Kamphorst, D. Ruelle, S. Ciliberto. “Liapunov exponents from time series,” Physical Review A, 34(6), 1986, pp. 4971-4979.