УДК 004.057.04 DOI: 10.18698/2541-8009-2019-07-506

МОДЕЛИРОВАНИЕ РАБОТЫ СЕРВЕРНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

А.Е. Шатунов
shatunovae@student.bmstu.ru
SPIN-код: 3291-9936

МГТУ им. Н.Э. Баумана, Москва, Российская Федерация

Аннотация

Основными задачами исследования являются моделирование работы программного обеспечения серверного слоя информационной системы и оценка адекватности полученной модели. Моделирование работы программного обеспечения является начальным этапом в исследовании вопроса об устойчивости серверного слоя к нагрузкам в виде интенсивного потока клиентских запросов. Программное обеспечение реализовано с использование сервис-ориентированного подхода к проектированию архитектуры. Функционирование программного обеспечения показано на примере дискретно-событийной имитационной модели. В качестве входных параметров модели использованы результаты обработки статистических данных о запросах, поступающих на обработку в реальное программное обеспечение. Анализ результатов моделирования и статистических данных о работе реального программного обеспечения позволил оценить качество моделирования

Ключевые слова

Имитационная модель, дискретно-событийное моделирование, серверное программное обеспечение, микросервисный стиль проектирования, система массового обслуживания, заявка, обслуживающее устройство, отказ в обслуживании

Поступила в редакцию 14.05.2019

Введение. В настоящее время наблюдается активное развитие web-технологий. Разрабатываемые web-приложения становятся более крупными, ориентированными на обработку большого количества пользовательских запросов. Для надежной работы таких приложений необходимо выполнение ряда требований, предъявляемых к алгоритмам и методам ввода-вывода данных, реализованным в программном обеспечении, а также к архитектуре приложения в целом [1].

Один из наиболее распространенных подходов, используемых для построения высоконагруженных систем [2], предполагает организацию сервис-ориентированной архитектуры. Современное представление сервис-ориентированной архитектуры представляет собой микросервисную архитектуру. Такая архитектура имеет ряд преимуществ, благодаря чему она стала востребована [3, 4], но вместе с тем существует ряд недостатков, среди которых нужно выделить усложнение процесса распределения и контроля нагрузки в системе [1, 5, 6]. Для решения задачи распределения нагрузки в сервис-ориентированной архитектуре предлагается использовать подход, основанный на моделировании работы программного обеспечения.

Структура серверного программного обеспечения разрабатываемой информационной системы. Серверная часть разрабатываемого приложения представляет собой совокупность микросервисов. Отметим, что вся бизнес-логика обработки данных выполняется на клиентской стороне, что позволяет снизить нагрузку на серверный слой.

Для хранения различных данных, требуемых для корректной обработки клиентских запросов, на серверной машине развернут сервер баз данных PostgreSQL. Выбор этого сервера обусловлен высокой загруженностью серверной части и большими объемами хранимых данных.

В существующей архитектуре каждый сервис принимает и обрабатывает специфичный для него набор запросов от клиентов. Обмен данными между серверной частью и клиентской, а также внутри серверной части между сервисами осуществляется по протоколу HTTP. Это позволяет разворачивать сервисы на разных физических машинах, что способствует более гибкому масштабированию серверной части. На уровне приложения разработан собственный протокол, определяющий формат сообщений, которыми обмениваются взаимодействующие стороны.

На серверной машине помимо микросервисов функционирует web-сервер Nginx в режиме обратного проксирования: он перенаправляет входящие клиентские запросы на соответствующие микросервисы и обеспечивает определенный уровень безопасности в сетевой инфраструктуре [7].

Все разработанные на текущий момент сервисы имеют схожую многопоточную архитектуру. В этой архитектуре выделяется главный поток выполнения и потоки-обработчики сетевых запросов по протоколу HTTP. Помимо этого в микросервисе могут выполняться другие служебные потоки, наличие которых является спецификой работы отдельно взятых микросервисов.

Логика работы серверной части выстроена таким образом, что некоторые микросервисы в ходе обработки пользовательских запросов генерируют запросы на другие микросервисы. Архитектура серверного программного обеспечения представлена на рис. 1.

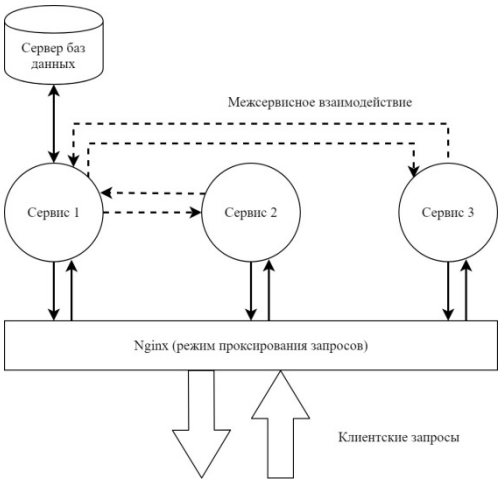

Рис. 1. Архитектура серверного ПО разрабатываемой информационной системы

Разработка имитационной модели серверного программного обеспечения. Серверное программное обеспечение (ПО) можно рассматривать как систему массового обслуживания (СМО), работа которой может быть смоделирована путем реализации дискретно-событийной модели [8]. На вход системы с некоторой усредненной интенсивностью поступают заявки, которые необходимо обслужить.

Термин «заявка» в рамках описания имитационной модели непосредственно связан с клиентским запросом, поступающим на реальный сервер. Заявка, поступившая на вход системы, идет по определенному в модели маршруту, который может включать в себя одно или несколько обслуживающих устройств, различные списки ожидания. Когда заявка проходит весь маршрут, она выводится из системы. В зависимости от того, по какому маршруту продвигалась заявка, она будет иметь один из возможных итоговых статусов обработки. Как правило, на выходе из модели заявка считается либо успешно обработанной, либо не обработанной по ряду причин.

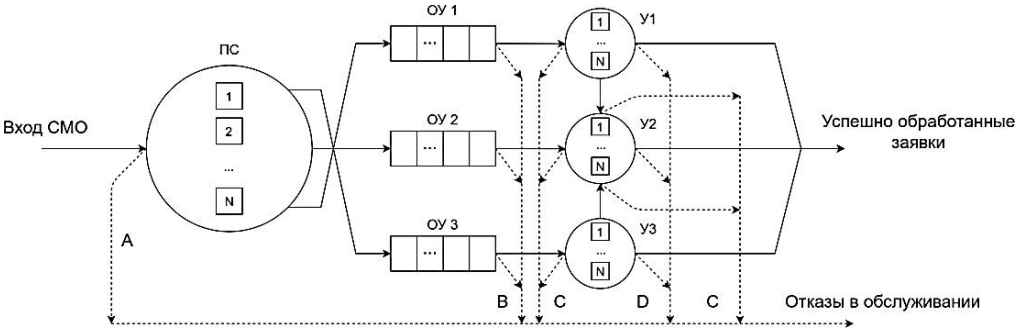

Рис. 2. Принципиальная схема СМО

Приведенное выше абстрактное описание СМО позволяет выделить следующие сущности, которые лежат в основе спроектированных классов в реализации имитационных моделей: заявка, запрос, обслуживающее устройство, списки ожидания, статусы обработки. Принципиальная схема системы массового обслуживания представлена на рис. 2. Видно, что в системе присутствует многоканальное обслуживающее устройство (прокси-сервер, ПС) с накопителями заявок (очереди устройств, ОУ1–ОУ3) и многоканальные обслуживающие устройство без накопителя заявок (устройства, У1–У3). Существует несколько маршрутов продвижения заявок, выбор того или иного маршрута обусловлен параметрами заявки и состоянием модели в текущий момент модельного времени. Обслуживающее устройство с накопителем заявок моделирует работу прокси-сервера Nginx. Остальные обслуживающие устройства моделируют работу микросервисов. Основная логика по продвижению заявок в модели реализуется в рамках симуляции.

Пунктирными линиями обозначены маршруты, которые приводят к отказу в обслуживании. Заявка может быть не обработана, если в ходе продвижения заявки в модели выполнится одно из условий:

1) все каналы обслуживания прокси-сервера заняты;

2) время ожидания в очереди для заявки истекло;

3) устройство обслуживания недоступно или все каналы устройства обслуживания заняты;

4) обслуживание заявки было прервано.

Процесс моделирования можно рассматривать с разных позиций в зависимости от выделения того или иного уровня абстракции. На более высоком уровне абстракции моделирование свелось к реализации заимствованной логики работы интерпретатора GPSS [9, 10], в которой выделяется три фазы моделирования: фаза ввода, фаза коррекция таймера и фаза просмотра. Схема алгоритма моделирования представлена на рис. 3.

Логика продвижения заявки в модели полностью определена в фазе просмотра. Для заявки рассмотрены состояния, в которых она может находиться, и возможные переходы между состояниями. На основе множества состояний и множества переходов составлен граф переходов. Переход из одного состояния в другое определяется состоянием модели в текущий момент модельного времени. Диаграмма состояний представлена на рис. 4.

Моделирование отказов и восстановлений обслуживающих устройств осуществляется на фазе просмотра до цикла просмотра заявок. Генерация момента модельного времени, в который произойдет отказ первого устройства, происходит на фазе ввода.

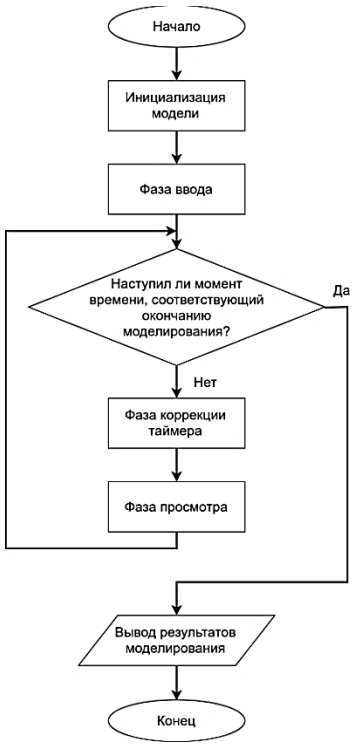

Рис. 3. Блок-схема алгоритма работы имитационной модели

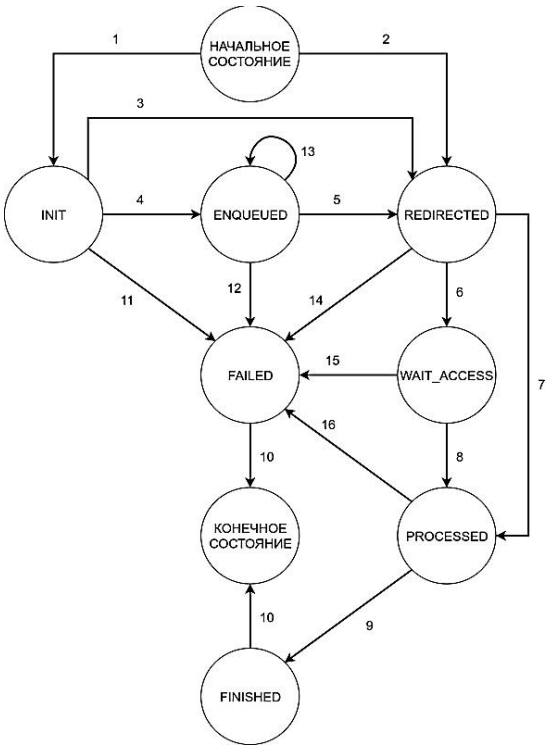

Рис. 4. Граф переходов состояний заявок

Заявка в состоянии INIT, попавшая в цепь текущих событий, вызывает генерацию новой заявки, для которой планируется ввод в модель. Таким образом моделируется периодическое обращение клиентов к серверу с заданной интенсивностью.

Состояния INIT, ENQUEUED имеют отношение к логике взаимодействия заявок с прокси-сервером Nginx. Состояния REDIRECTED, WAIT_ACCESS, PROCESSED описывают логику взаимодействия заявок с одним или несколькими вышестоящими серверами. Если заявка находится в одном из двух состояний FINISHED, FAILED, то она должна освободить ресурсы прокси-сервера и вышестоящего сервера перед тем, как покинуть модель. Заявка занимает ресурсы прокси-сервера (канал обслуживания), находясь в состоянии INIT, при условии, что прокси-сервер имеет хотя бы один свободный канал обслуживания в текущий момент модельного времени. Заявка может занять ресурсы одного из микросервисов, находясь в одном из состояний INIT, ENQUEUED, REDIRECTED.

Программная реализация модели СМО выполнена на языке программирования C++ с использованием объектно-ориентированного подхода. Спроектированы и реализованы следующие классы: «запрос», «заявка», «многоканальное устройство», «прокси-сервер», «сервер», «симуляция».

Моделирование работы серверного программного обеспечения. Для моделей принято допущение о том, что обработка заявки может завершиться неудачно только в случае отказа обслуживающего устройства. В контексте предметной области разрабатываемой информационной системы это означает, что все запросы составлены корректно с точки зрения соответствия формату и типу передаваемых данных. При моделировании отказов также были приняты допущения, связанные с тем, что генерация последующего отказа происходит только после восстановления работоспособности очередного устройства; при моделировании хотя бы раз должен произойти отказ наиболее загруженного сервера; отказать могут только вышестоящие сервера. В модели серверного ПО наиболее загруженным устройством обслуживания считается является сервис аутентификации и авторизации.

В качестве исходных параметров для моделирования выбраны следующие:

1) время моделирования;

2) функция плотности вероятности случайной величины, характеризующей промежуток времени между поступлением на вход системы двух заявок;

3) функция плотности вероятности случайной величины, характеризующей промежуток времени между восстановлением одного обслуживающего устройства и отказом следующего;

4) функция плотности вероятности случайной величины, характеризующей время восстановления обслуживающего устройства;

5) количество каналов обслуживания прокси-сервера;

6) ограничение на максимальный размер очередей прокси-сервера;

7) ограничение на время ожидания заявок в очередях прокси-сервера;

8) количество вышестоящих серверов;

9) количество каналов обслуживания для каждого вышестоящего сервера.

Вероятностные распределения случайных величин в процессе моделирования применяются в генераторах последовательностей случайных чисел для выполнения двух основных действий: определение момента модельного времени

поступления очередной заявки на вход системы; определение вероятностных характеристик заявки. В модели используются равномерное и экспоненциальное распределения.

Экспоненциальное распределение имеет функцию плотности вероятности

$$f(x) = \lambda e^{-\lambda x}. \quad (1)$$

Параметр λ в формуле (1) определяет интенсивность (частоту) возникновения того или иного события. Применительно к процедуре генерации заявок, этот параметр определяет среднюю частоту поступления заявок на вход системы. Значение параметра было получено в результате обработки статистических данных по запросам, поступающим на реальное серверное программное обеспечение. Значение параметра λ = 0.152. Это означает, что среднее время между поступлением двух заявок на вход СМО составляет 6.574 секунд.

Равномерное распределение имеет функцию плотности вероятности

$$f(x) = \frac{1}{b-a}. \quad (2)$$

Значения параметров *a* и *b* в формуле (2) выбраны равными 0,0 и 1,0 соответственно. Равномерное распределение необходимо для реализации механизма случайного выбора сервиса, в который попадет данная заявка, а также для корректировки суммарного времени обработки заявки на основе вероятностных характеристик запроса.

Разработанная модель серверной инфраструктуры является упрощенной, однако она способна адекватно отражать реакцию системы при различных значениях входных параметров.

Для определения достоверности результатов моделирования были выполнены следующие действия:

1) организован сбор статистики о работе серверного ПО;

2) реализована обработка статистических данных с целью определения параметров заявок, поступающих на вход модели, а также некоторых параметров модели;

3) проведено моделирование работы серверного ПО;

4) выполнен анализ результатов моделирования.

Сбор статистики организован путем внедрения в сервисы программного модуля, основное назначение которого заключалось в регистрации различных параметров запросов в ходе их обработки. Значения параметров запросов для имитационной модели представлены в табл. 1, где введены следующие обозначения параметров заявок:

1) вероятность поступления заявки данного типа в систему $P_з$;

2) математическое ожидание времени обработки заявки $\overline{T}_з$, мкс;

3) среднеквадратичное отклонение $\sigma_з$ величины $\overline{T}_з$, мкс;

4) математическое ожидание времени обработки запроса в базу данных $\overline{T}_{БД}$, мкс;

5) среднеквадратичное отклонение $\sigma_{БД}$ величины $\overline{T}_{БД}$, мкс;

6) вероятность порождения запроса в базу данных в ходе обработки заявки $P_{БД}$;

7) вероятность взаимодействия с другим сервисом в ходе обработки заявки $P_{вз}$.

Таблица 1

Параметры запросов для микросервисной модели ПО

Микросервис	Запрос	Параметр						
		$P_з$	$\overline{T}_з$	$\sigma_з$	$\overline{T}_{БД}$	$\sigma_{БД}$	$P_{БД}$	$P_{вз}$
Сервис 1	запрос 1	0.0510	1483	20	1117	21	1.000	0.0
	запрос 2	0.0770	324	7	1068	19	0.221	0.0
	запрос 3	0.3310	1986	21	1263	20	0.085	0.0
	запрос 4	0.2300	14782	47	4320	22	1.000	0.0
	запрос 5	0.2740	108	5	1444	26	0.150	0.0
	запрос 6	0.0003	856	9	1319	14	0.863	0.0
	запрос 7	0.0040	2653	19	0	0	0.000	0.0
	запрос 8	0.0300	93	6	0	0	0.000	0.0
	запрос 9	0.0030	598	14	0	0	0.000	0.0
Сервис 2	запрос 10	0.1460	1331	15	0	0	0.000	1.0
	запрос 11	0.8540	1602	18	0	0	0.000	1.0
Сервис 3	запрос 12	0.0436	284	8	0	0	0.000	1.0
	запрос 13	0.0436	1107	14	0	0	0.000	1.0
	запрос 14	0.1529	332	7	0	0	0.000	1.0
	запрос 15	0.1529	904	15	0	0	0.000	1.0
	запрос 16	0.6070	575	15	0	0	0.000	1.0

Вероятность поступления заявки в систему рассчитывали как отношение количества заявок данного типа к общему количеству всех заявок, поступивших на вход системы:

$$P_з = \frac{N_з}{N_{общ}}.$$

Для расчета математического ожидания времени обработки заявок использована простая рекуррентная формула

$$a_i = \frac{a_{i-1} i + d_i}{i+1},$$

где a_i — среднее значение величины на очередном шаге просмотра статистики; d_i — очередное значение из выборки на текущем шаге просмотра статистики.

Для расчета среднеквадратичных отклонений была использована формула на основании смещенной оценки дисперсии

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2},$$

где x_i — i-е значение параметра из статистической выборки; $\overline{x}$ — математическое ожидание значения параметра; n — количество значений в статистической выборке.

Среднее время перенаправления запроса прокси-сервером было выбрано приближенно с учетом результатов экспериментальных замеров. Это значение составляет 500 мкс.

Количество каналов обслуживания на вышестоящих серверах было принято равным 10. Именно такое число потоков-обработчиков запросов порождается в микросервисах на реальном сервере. Количество каналов обслуживания для прокси-сервера в модели было принято равным 65536. Это число соответствует максимальному числу соединений, которые способен обрабатывать Nginx в реальной серверной инфраструктуре. Время моделирования было принято равным пяти рабочим дням, поскольку сбор статистики осуществлялся в течение такого промежутка времени.

Моделирование проводили несколько раз, в результате чего была сформирована выборка результатов моделирования, на основе которой получены усредненные результаты, представленные в табл. 2, где приняты следующие обозначения: X — параметр, $M[X]$ — математическое ожидание параметра, σ — среднеквадратичное отклонение параметра.

Таблица 2

Результаты моделирования работы серверного ПО

X	Общее количество заявок	Количество межсервисных взаимодействий	Количество отказов в обслуживании
$M[X]$	130 218	30 147	216
σ	41	37	8

Для оценки качества моделирования полученные результаты были сопоставлены с результатами обработки статистических данных по работе реального серверного программного обеспечения. Результаты обработки статистических данных представлены в таблице 3.

Таблица 3

Результаты обработки статистики по отказам на реальном сервере

X	Общее количество заявок	Количество межсервисных взаимодействий	Количество отказов в обслуживании
$M[X]$	119 803	28 613	105
σ	35	28	6

Заключение. Как видно из результатов моделирования, относительная погрешность по числу обработанных запросов составляет 0,087. Это дает основание полагать, что модель на данном этапе исследований уже способна адекватно отражать процессы, протекающие в ходе обработки заявок серверным программным обеспечением.

Литература

[1] Клеппман. М. Высоконагруженные приложения. Программирование, масштабирование, поддержка. СПб., Питер, 2018.
[2] Di Francesco P., Lago P., Malavolta I. Research on architecting microservices: trends, focuses, and potential for industrial adoption. *IEEE ICSA*, 2017. DOI: 10.1109/ICSA.2017.24 URL: https://ieeexplore.ieee.org/document/7930195
[3] Namiot D., Sneps-Sneppe M. On micro-services architecture. *INJOIT*, 2014, vol. 2, no. 9, pp. 24–27.
[4] Richards M. Microservices vs. service-oriented architecture. O'Reilly Media, 2016.
[5] Niu Y., Liu F., Li Z. Load balancing across microservices. *IEEE INFOCOM*, 2018. DOI: 10.1109/INFOCOM.2018.8486300 URL: https://ieeexplore.ieee.org/document/8486300
[6] Newman S. Building microservices. O'Reilly Media, 2015.
[7] Айвалиотис Д. Администрирование сервера NGINX. М., ДМК Пресс, 2013.
[8] Bangsow S. Use cases of discrete event simulation. Springer, 2012.
[9] Кельтон В., Лоу А. Имитационное моделирование. СПб., BHV, 2004.
[10] Кудрявцев Е.М. GPSSWorld. Основы имитационного моделирования различных систем. М., ДМК Пресс, 2004.

Шатунов Алексей Евгеньевич — студент кафедры «Системы автоматизированного проектирования», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Научный руководитель — Берчун Юрий Валерьевич, старший преподаватель кафедры «Системы автоматизированного проектирования», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

SIMULATION OF SERVER SOFTWARE WORK

A.E. Shatunov shatunovae@student.bmstu.ru
SPIN-code: 3291-9936

Bauman Moscow State Technical University, Moscow, Russian Federation

Abstract

The main objectives of the study are to simulate the software operation on the server layer of the information system and to assess the adequacy of the model obtained. The software simulation is the initial stage in the study of the server layer stability to loads in the form of an intense flow of client requests. The software is implemented using a service-oriented approach to architecture design. The software functioning is shown on the example of a discrete event simulation model. The results of processing statistical data on requests received for processing in real software were used as input parameters of the model. Analysis of simulation results and statistical data on the real software operation made it possible to assess the quality of the simulation

Keywords

Simulation model, discrete event simulation, server software, microservice design style, queuing system, application, server, denial of service

Received 14.05.2019

References

[1] Kleppman. M. Vysokonagruzhennye prilozheniya. Programmirovanie, masshtabirovanie, podderzhka [High-load applications. Programming, scaling, support]. Sankt-Petersburg, Piter Publ., 2018 (in Russ.).

[2] Di Francesco P., Lago P., Malavolta I. Research on architecting microservices: trends, focuses, and potential for industrial adoption. *IEEE ICSA*, 2017. DOI: 10.1109/ICSA.2017.24 URL: https://ieeexplore.ieee.org/document/7930195

[3] Namiot D., Sneps-Sneppe M. On micro-services architecture. *INJOIT*, 2014, vol. 2, no. 9, pp. 24–27.

[4] Richards M. Microservices vs. service-oriented architecture. O'Reilly Media, 2016.

[5] Niu Y., Liu F., Li Z. Load balancing across microservices. *IEEE INFOCOM*, 2018. DOI: 10.1109/INFOCOM.2018.8486300 URL: https://ieeexplore.ieee.org/document/8486300

[6] Newman S. Building microservices. O'Reilly Media, 2015.

[7] Ayvaliotis D. Administrirovanie servera NGINX [NGINX server administration]. Moscow, DMK Press Publ., 2013 (in Russ.).

[8] Bangsow S. Use cases of discrete event simulation. Springer, 2012.

[9] Law A.M., Kelton W.D. Simulation modelling and analysis. McGraw-Hill Education, 2000 (Russ. ed.: Imitatsionnoe modelirovanie. Sankt-Petersburg, BHV, 2004.)

[10] Kudryavtsev E.M. GPSSWorld. Osnovy imitatsionnogo modelirovaniya razlichnykh system [Fundamentals of simulation modeling of different systems]. Moscow, DMK Press Publ., 2004 (in Russ.).

Shatunov A.E. — Student, Department of Systems of the Automated Designing, Bauman Moscow State Technical University, Moscow, Russian Federation.

Scientific advisor — Berchun Yu.V., Senior Lecturer, Department of Systems of the Automated Designing, Bauman Moscow State Technical University, Moscow, Russian Federation.