## И. М. Якимов, А. П. Кирпичников, В. В. Мокшин

# МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ В ИМИТАЦИОННОЙ СРЕДЕ ANYLOGIC

Ключевые слова: имитационное моделирование, сложная система, система массового обслуживания, система AnyLogic, заявка, генератор транзактов, очередь, обслуживающий аппарат.

Приводится описание методики моделирования сложных систем, включающей в себя построение имитационных моделей в системе AnyLogic, позволяющей вводить структуры моделируемых систем в графическом виде. Приводятся несколько наиболее характерных моделей систем массового обслуживания.

Keywords: imitation modeling, complex system, queuing system, the system AnyLogic, application generator TRANSACT, queue serving device.

The article describes a technique for simulating complex systems, including the construction of simulation models in the AnyLogic, allows you to enter the structure of the simulated systems in graphical form. Are some of the most typical models of queuing systems.

### Введение

Бурное развитие информационных технологий, наблюдаемое в последнее время, привело к серьёзным революционным изменениям в предметной области по моделированию систем. И если раньше в РФ доминирующим средством моделирования систем был специализированный язык GPSS и большинство специалистов по моделирования владели им, то в настоящее время появился, чуть ли не десяток других систем моделирования, позволяющих ввод структур моделируемых систем в графическом виде. Встают актуальные задачи по выбору системы моделирования под конкретную предметную область моделирования и быстрому обучению моделированию в выбранной системе.

Авторы данной статьи ранее опубликовали две статьи, посвященные вопросам обучения имитационному моделированию в системе GPSS W с расширенным редактором [1] и в системах BPWin-Arena [2]. Данная статья рассматривает вопросы обучения в системе моделирования AnyLogic. Все три статьи написаны по одной и той же методике изложения и их применение проиллюстрировано на одних и тех же имитационных моделях трёх типовых систем массового обслуживания (СМО), что по мнению авторов способствует выбору наиболее пригодных программных средств для моделирования конкретных СМО конкретными пользователями.

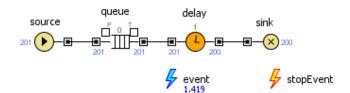
AnyLogic - программное средство для имитационного моделирования систем и процессов, разработанное российской компанией «Экс Джей Текнолоджис» (англ. XJ Technologies) [3].Первая версия системы AnyLogic 4.0 разработана в 2003 году (желательно уточнить). В 2014 году разработана версия AnyLogic 7. Система AnyLogic включает в себя графический язык моделирования и позволяет пользователю расширять созданные модели с помощью языка Java.

В процессе разработки имитационной модели пользователю доступна «Палитра компонентов моделей», приведённая на рис.1. Она включает в себя совокупность библиотек по отдельным тематическим разделам. Объекты основной библиотеки AnyLogic являются строительными блоками, с по-

мощью которых строятся структурные схемы модели. По своей функциональной принадлежности объекты подразделяются на несколько категорий, их краткое описание приведено в таблице 1. Для обработки результатов моделирования используется методика, описанная в статьях [4-7].

**Модель 1.** СМО - генератор транзактов (ГТ) (равномерный закон  $10\pm6$ ) — очередь неограниченной длины — обслуживающий аппарат (ОА) (равномерный закон  $9\pm7$ ). Остановить моделирование после вывода из модели 200 транзактов.

На рис.1 приведена структурная схема примера 1, сформированная в системе AnyLogic, с результатами моделирования.



Количество транзактов в системе: 1 Занятость delay: 0.9248734324146877

**Puc. 1 – Структурная схема примера 1 в системе AnyLogic** 

Таблица 1 – Объекты палитры основной библиотеки

таолица 1 – Ооъекты палитры основной ойолиотеки			
Катего рия	Наименование и графическое представление	Функция	
Поток заявок	Source	Создает заявки	
	Sink in	Уничтожает поступающие заявки	
	Enter inExternal	Вставляет уже существующие заявки в определенное место внутри процесса, заданного потоковой диаграммой.	
	out	потоковой диаграммой.	

рия прафическое представление  Exit  outExternal  in □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	TC	11	<i>A</i> .
Ехіт ооцЕхтепа	Катего	Наименование и	Функция
Ехіт ош'Ехтепа по в объект заявки из процесса, заданного потоковой диаграммой, позволяя пользователю самому решать, что следует сделать с этими заявками.  Ноід по ош поток заявок на определенном участке структурной схемы.  Хріт по ош данное число новых заявок (копий) и посылает их дальше.  Сотвіпе по ош	рия	* *	
в объект заявки из процесса, заданного пото-ковой диаграммой, позволяя пользователю самому решать, что следует сделать с этими заявками.  Ноід по оцтору ет/разблокировывает поток заявок на определенном участке структурной схемы.  Зрііт оцтору смемы.  Для поступающей заявки из профиссовном участке структурной схемы.  Для поступающей заявки из объект создает заданное число новых заявок (копий) и посылает их дальше.  Сотвіпе по оцтору по низ дарх заявок в порты по и порядке), а затем создает новую заявку и направляет ее на выходной порт.  Вејестоцтри оцтору оцтору по по оцтору по по оцтору по по порядке в один из двух выходных портов в зависимости от выполнения заданного условия.  SelectOutput оцтору по			
цесса, заданного пото-ковой диаграммой, по-зволяя пользователю самому решать, что следует сделать с этими заявками.  НоІд іп оцт — поток заявок на определенном участке структурной схемы.  Для поступающей заявки объект создает заданное число новых заявок (копий) и посылает их дальше.  Сотвіпе іп оцт — дожидается поступления двух заявок в порты in 1 и іп 2 (в произвольном порядке), а затем создает новую заявку и направляет ее на выходной порт.  SelectOutput оцт іп оцт — оцт заявки в один из двух выходных портов в зависимости от выполнения заданног условия.  SelectOutput — оцт			
песса, заданного положовой диаграммой, по- зволяя пользователю самому решать, что следует сделать с этими заявками.  Ноід по оцтору ет/разблокировывает поток заявок на определенном участке структурной схемы.  Для поступающей заявки и объект создает заданное число новых заявок (копий) и посылает их дальше.  Сотвіте по оцтору прети ния двух заявок в порты по и по двух заяви в один из двух выходных портов в зависимости от выполнения заданного условия.  SelectOutput оцтору по оцтору п			в объект заявки из про-
зволяя пользователю самому решать, что следует сделать с этими заявками.  Ноіd in out		In	цесса, заданного пото-
зволяя пользователю самому решать, что следует сделать с этими заявками.  Ноіd in out			ковой диаграммой, по-
Сомый оит опторацие оптор			зволяя пользователю
По оит по опток заявками.  Влокирует сделать с этими заявками.  Влокирует стразблокировывает поток заявок на определенном участке структурной схемы.  Для поступающей заявки объект создает заданное число новых заявок (копий) и посылает их дальше.  Сотвот домилается поступления двух заявок в порты по ит по ит по порядке), а затем создает новую заявку и направляет ее на выходной порт.  SelectOutput Направляет входящие заявки в один из двух выходных портов в зависимости от выполнения заданного условия.  SelectOutput5 Объект направляет входящие заявки в один из пяти выходных портов в зависимости от выполнения заданных (потов вероятностей) условий.  Хранит заявки в определенном порядке. Моделирует очередь заявок, омидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Обозначает вход в областы процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd областы процесса, в которой одновременно может находиться ограниченное количество заявок.			
НоІд іп оцт поток заявок на определенном участке структурной схемы.  При поцтору поток заявок на определенном участке структурной схемы.  При поцтору поток заявок на определенном участке структурной схемы.  При поцтору поток заявок на определенном участке структурной схемы.  При поцтору поток заявок на определенном участке структурной схемы.  При поцтору поток заявок на определенном и объект создает заявим в порты іп і и іп 2 (в произвольном порядке), а затем создает новую заявку и направляет ее на выходной порт.  При поцтору поток заявки в один из двух выходных портов в зависимости от выполнения заданного условия.  При поцтору поцтору поток заявки в один из двух выходных портов в зависимости от выполнения заданнох (детерминистических или заданных (детерминистических или заданных с помощью вероятностей) условий.  При поцтору поцтору поцтору почередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  При поцтору почередь заянотока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  При поцтору почередь заянотока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  При поцтору почередь заяном вероятностей) условий.  При почет заявки в один из двух выходных портов заявок вероятностей условий.  При почет заявки в один из двух выходных портов заявок вероятностей условий.  При почет заявки в один из двух выходных портов зависимости от выполнения заданных (детерминистических или заданных с помощью вероятностей) условий.  При почет заявки в один из двух выходных портов зависимости от выполнения заданных (детерминистических или заданных (детерминис			
НоІ поит поит поток заявок на определенном участке структурной схемы.    Для поступающей заявки объект создает заданное число новых заявок (копий) и посылает их дальше.    Дожидается поступления двух заявок в порты поит порядке), а затем создает новую заявку и направляет ее на выходной порт.    SelectOutput outT по поит поит поит поит поит поит поит п			
по оит поток заявок на определенном участке структурной схемы.    Для поступающей заявки объект создает заданное число новых заявок (копий) и посылает их дальше.   Дожидается поступления двух заявок в порты по ит по порядке), а затем создает новую заявку и направляет е на выходной порт.   SelectOutput оит заявки в один из двух заявки в один из двух выходных портов в зависимости от выполнения заданных (детерминистических или заданных (детерминистических или заданных с помощью вероятностей) условий. Хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой днаграмме.   Маtch   Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.   Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.   Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.   Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.   Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.   Собозначает выход из область процесса, в которой одновременно может находиться ограниченное количество заявок.			заявками.
тото ваявок на определенном участке структурной схемы.    При		Hold	Блокиру-
Поток заявок на определенном участке структурной схемы.  Для поступающей заявки объект создает заданное число новых заявок (копий) и посылает их дальше.  Сотвот ния двух заявок в порты по ит по порядке), а затем создает новую заявку и направляет е на выходной порт.  SelectOutput оutт оutт заявки в один из двух выходных портов в зависимости от выполнения заданного условия.  SelectOutput5 оut1 оut1 оut1 оut1 оut2 оut2 оut5 оut0 пяти выходных портов в зависимости от выполнения заданного условия.  Queue оut0 оut0 оut0 пяти выходных портов в зависимости от выполнения заданных (детерминистических или заданных с помощью вероятностей) условий.  Хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd in out области процесса, в которой может находиться ограниченное количество заявок.		in out	1 2
Пенном участке структурной схемы.    Для поступающей заявки объект создает заданное число новых заявок (копий) и посылает их дальше.    Дожидается поступления двух заявок в порты in 1 и in 2 (в произвольном порядке), а затем создает новую заявку и направляет ее на выходной порт.    SelectOutput			
Турной схемы. Для поступающей заяв-ки объект создает заданное число новых заявок (копий) и посылает их дальше.  Сотвіпе іп1 оцт ния двух заявок в порты іп1 и іп2 (в произвольном порядке), а затем создает новую заявку и направляет ее на выходной порт.  SelectOutput оцт поцт ния заданного условия.  SelectOutput5 объект направляет входящие заявки в один из двух выходных портов в зависимости от выполнения заданного условия.  Объект направляет входящие заявки в один из двух выходных портов в зависимости от выполнения заданных (детерминистических или заданных с помощью вероятностей) условий.  Хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Матсh Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd in out обозначает выход из области процесса, в которой может находиться ограниченное количество заявок.			
По оит поит поит поит поит поит поит поит			
ки объект создает заданное число новых заявок (копий) и посылает их дальше.  Сотвот оп		G 1'	- * *
Данное число новых заявок (копий) и посылает их дальше.  Сотмотром развет их дальше.  Дожидается поступления двух заявок в порты in1 и in2 (в произвольном порядке), а затем создает новую заявку и направляет ее на выходной порт.  SelectOutput оut1 заявки в один из двух выходных портов в зависимости от выполнения заданного условия.  SelectOutput5 оut2 ния заданных (детерминистических или заданных портов в зависимости от выполнения заданных (детерминистических или заданных (детерминистических или заданных портов в занисимости от выполнения заданных (детерминистических или заданных с помощью вероятностей) условий. Хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch Синхронизирует два потока заявок путем нахождення пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd in out Обозначает выход из области процесса, в которой может находиться только ограниченное количество заявок.			
заявок (копий) и посылает их дальше.  Сотвіпе по онто онто онто онто онто онто онто о		in out	ки объект создает за-
Сотвіпе піп оцт пі			данное число новых
Сотвіпе іп1 оит іп1 и іп2 (в произвольном порядке), а затем создает новую заявку и направляет ее на выходной порт.  SelectOutput оит заявки в один из двух выходных портов в зависимости от выполнения заданного условия.  SelectOutput5 оот от от от от от выполнения заданных портов в зависимости от выполнения заданных портов в зависимости от выполнения заданных с помощью вероятностей) условий.  Хранит заявки в один из пяти выходных портов в зависимости от выполнения заданных с помощью вероятностей) условий.  Хранит заявки в отределенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd іп оит области процесса, в которой может находиться ограниченное количество заянок.			заявок (копий) и посы-
Сотвіпе по оцт		outCopy	лает их дальше.
ип1 оит поит поит поит поит поит поит поит		The second	
ип1 оит поит поит поит поит поит поит поит		Combine	Поминается поступна
оит in 1 и in 2 (в произвольном порядке), а затем создает новую заявку и направляет ее на выходной порт.  SelectOutput оит заявки в один из двух выходных портов в зависимости от выполнения заданного условия.  SelectOutput5 Объект направляет входящие заявки в один из пяти выходных портов в зависимости от выполнения заданных с помощью вероятностей) условий.  Queue outPreempted outTim деленном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out в боласть процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd in out области процесса, в которой может находиться только ограниченное количество зая-			, ,
пп и пл и			2 .
In			
Создает новую заявку и направляет ее на выходной порт.  SelectOutput  outT  in  outF  SelectOutput5  in out1  out1  out1  out2  out2  out2  out2  out3  out0  outD  OutD  Queue  outPreempted outTim  out  in  out  Officer направляет входящие  заявки в один из пяти выходных портов в зависимости от выполнения заданных (детерминистических или заданных с помощью вероятностей) условий.  Хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd  in out  Обозначает выход из области процесса, в которой может находиться ограниченное количество заяниченное количество заяниченное количество заяниченное количество заяниченное количество обрасти процесса, в которой может находиться только ограниченное количество заяниченное количество за			ном порядке), а затем
ВеlectOutput оutТ заявки в один из двух выходных портов в зависимости от выполнения заданного условия.  SelectOutput5 оut1 оut1 оut1 оut2 оut2 оut2 оut2 оut0 оut0 оut0 оut0 оut0 оut0 оит заданных с помощью вероятностей) условий.  Queue оutPreempted outTim оut бобъектами, следующими заданным в потоковой диаграмме.  Маtch Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd обозначает выход из области процесса, в которой может находиться только ограниченное количество заявок.			создает новую заявку и
ВеlectOutput  In  OutF  SelectOutputS  In  OutO  OutO  OutO  OutO  OutO  OutD  Obsekt Hanpabner bxo  Jamuue 3ayakhu o Outh u3  Instru bыходных портов в 3abucuмости от выполнения заданных с помощью вероятностей) условий.  Xpанит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Обозначает вход  в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd  In out  Обозначает выход из области процесса, в которой может находиться ограниченное количество заявок.  Обозначает выход из области процесса, в которой может находиться только ограниченное количество зая-		in2	направляет ее на выход-
ВеlectOutput  In  OutF  SelectOutputS  In  OutO  OutO  OutO  OutO  OutO  OutD  Obsekt Hanpabner bxo  Jamuue 3ayakhu o Outh u3  Instru bыходных портов в 3abucuмости от выполнения заданных с помощью вероятностей) условий.  Xpанит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Обозначает вход  в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd  In out  Обозначает выход из области процесса, в которой может находиться ограниченное количество заявок.  Обозначает выход из области процесса, в которой может находиться только ограниченное количество зая-			ной порт.
оиt выходных портов в зависимости от выполнения заданного условия.  SelectOutput5  in out1 out1 out2 out2 out2 munutruveckux или заданных с помощью вероятностей) условий.  Xранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Match  Cuнхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out  RestrictAreaEnd in out  Oбозначает выход из область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd области процесса, в которой может находиться ограниченное количество заяниченное количество заявок.		SelectOutput	
выходных портов в зависимости от выполнения заданного условия.  SelectOutput5  in out1 out1 out2 out2 hering sagakin в один из пяти выходных портов в зависимости от выполнения заданных (детерминистических или заданных с помощью вероятностей) условий.  Хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out  Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd in out  Обозначает выход из области процесса, в которой может находиться ограниченное количество заянок.			•
висимости от выполнения заданного условия.  SelectOutput5  in out1 out1 out2 out2 out3 out3 outD outD  Queue outPreempted outTim in out  Match  Match  RestrictedAreaStart in out  RestrictAreaEnd in out  RestrictAreaEnd in out  Bucuмости от выполнения заданных портов в зависимости от выполнения заданных (детерминистических или заданных с помощью вероятностей) условий.  Хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.  Обозначает выход из области процесса, в которой может находиться только ограниченное количество зая-			•
оиtf  SelectOutput5  in out1 out1 out2 out2 out3 out3 OutPreempted outTim in out in out  Match  RestrictedAreaStart in out  RestrictAreaEnd in out  Out1 Out1 Out1 Out1 Out2 Out2 Out2 Out2 Out3 Out3 Out3 Out3 Out4 Out9 Out9 Out9 Out9 Out9 Out9 Out9 Out9			•
SelectOutput5  in □ □ □ out0  out1  out2  out2  out3  outD outD оutD оutD оutD оutD оutD оutD оиtD оиtD оиtD оиtD оиtD оиtD оиtD ои		Out-E	
профонков оп от			ния заданного условия.
пяти выходных портов в зависимости от выполнения заданных (детерминистических или заданных с помощью вероятностей) условий.  Одене онфременто онфранка объектами, следующими за данным в потоковой диаграмме.  Маtch  Потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Потока заявок путем нахождения пар заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Потока заявок путем нахождения пар заявок путем нахождения пар заявок путем нахождения пар заявок.  Потока заявок путем нахождения пар заявок путем нахождения пар заявок путем нахождения пар заявок.  Потока заявок путем нахождения пар зам путем нахождения пар заявок путем нахождения пар заявок путем н		SelectOutput5	Объект направляет вхо-
оиt2 оиt3 оиt0 оиt0 Опити оит0 оит0 Опити о		in □-o-□ out0	дящие заявки в один из
оита оита оита оита вероятностей) условий.  Queue оитрееmpted оиттто оит объектами, следующими за данным в потоковой диаграмме.  Мател Оит Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Restricted Area Start in out вобласть процесса, в которой одновременно может находиться ограниченное количество заявок.  Restrict Area End область процесса, в которой может находиться ограниченное количество заявок.  Restrict Area End область процесса, в которой может находиться ограниченное количество заявок.			пяти выходных портов в
оиtD  Опеце оперативной вероятностей) условий. Оперативной вероятностей) условий. Оперативной вероятностей условий. Оперативной вероятностей условий. Оперативной порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме. Опетативной инферетивной инферетивной условиться ответствия. Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок. Обозначает выход из области процесса, в которой может находиться только ограниченное количество заяниченное кол			зависимости от выпол-
оиtD  оиtD  министических или заданных с помощью вероятностей) условий.   Хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out  Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd in out  Обозначает выход из области процесса, в которой может находиться ограниченное количество заяниченное кол			нения заданных (детер-
Queue outPreempted outTim   Дранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.   Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.   Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.   RestrictAreaEnd   Обозначает выход из области процесса, в которой может находиться ограниченное количество заяниченное количество заяничество заяничество заяничество заяничество заяничество заяничество заяничество заяничество з		⊢□³ out3	
Вероятностей) условий.  Хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd области процесса, в которой может находиться ограниченное количество заяниченное количе		L⊓□ outD	
Queue outPreempted outTim in out         Хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.           Match         Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.           RestrictedAreaStart in out         Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.           RestrictAreaEnd in out         Обозначает выход из области процесса, в которой может находиться только ограниченное количество зая-		1111917	
оиtPreempted outTim деленном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out Вобласть процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd области процесса, в которой может находиться ограниченное количество заяниченное количеств		0	
делирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out  область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd области процесса, в которой может находиться ограниченное количество заявок.			
вок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.  Маtch  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd области процесса, в которой может находиться ограниченное количество заяном.		outreempted outrime	
объектами, следующими за данным в потоковой диаграмме.  Маtch  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  Restricted Area Start in out  область процесса, в которой одновременно может находиться ограниченное количество заявок.  Restrict Area End область процесса, в которой может находиться области процесса, в которой может находиться только ограниченное количество зая-			
ми за данным в потоковой диаграмме.  Маtch  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out  оит  Собозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd области процесса, в которой может находиться только ограниченное количество заяниченное количество за заяниченное количество заяниченное количество заяниченное количе		in out	
вой диаграмме.  Маtch  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих за- данному критерию со- ответствия.  RestrictedAreaStart in out  в область процесса, в которой одновременно может находиться огра- ниченное количество заявок.  RestrictAreaEnd in out  Обозначает выход из области процесса, в которой может нахо- диться только ограни- ченное количество зая-			объектами, следующи-
вой диаграмме.  Маtch  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих за- данному критерию со- ответствия.  RestrictedAreaStart in out  в область процесса, в которой одновременно может находиться огра- ниченное количество заявок.  RestrictAreaEnd in out  Обозначает выход из области процесса, в которой может нахо- диться только ограни- ченное количество зая-			ми за данным в потоко-
Маtch  Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих за- данному критерию со- ответствия.  Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd in out  Обозначает выход из области процесса, в которой может находиться ограниченное количество заяном.			
потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out В область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd Обозначает выход из области процесса, в которой может находиться только ограниченное количество заяниченное количество заяничество зая		Match	
нахождения пар заявок, удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out В область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd области процесса, в которой может находиться области процесса, в которой может находиться только ограниченное количество зая-			
удовлетворяющих заданному критерию соответствия.  RestrictedAreaStart in out в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd области процесса, в которой может находиться ограниченное количество заявок.			•
данному критерию соответствия.  RestrictedAreaStart in out В область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd области процесса, в которой может находиться ограниченное количество заявок.			
RestrictedAreaStart in out В область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd области процесса, в которой может находиться ограниченное количество заявок.			
RestrictedAreaStart in out  в область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd in out  области процесса, в которой может находиться области процесса, в которой может находиться только ограниченное количество зая-			
in out  В область процесса, в которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd области процесса, в которой может находиться ограниченное количество заяниченное количество заяничество з		D / 11 0: :	~ ~
которой одновременно может находиться ограниченное количество заявок.  RestrictAreaEnd области процесса, в которой может находиться только ограниченное количество зая-			
может находиться ограниченное количество заявок.  RestrictAreaEnd области процесса, в которой может находиться только ограниченное количество зая-		in out	
ниченное количество заявок.  RestrictAreaEnd Обозначает выход из области процесса, в которой может находиться только ограниченное количество зая-		_ <b>_</b>	которой одновременно
ниченное количество заявок.  RestrictAreaEnd Обозначает выход из области процесса, в которой может находиться только ограниченное количество зая-		-   -   -   -   -   -   -   -   -   -	
заявок.  RestrictAreaEnd  in out  области процесса, в которой может находиться только ограниченное количество зая-			•
RestrictAreaEnd in out  области процесса, в которой может находиться только ограниченное количество зая-			
in out области процесса, в которой может находиться только ограниченное количество зая-		Restrict Area End	
которой может находиться только ограниченное количество зая-			
диться только ограниченное количество зая-		III OUL	*
ченное количество зая-			-
			•
вок.			ченное количество зая-
		1	BOK

Катего рия	Наименование и графическое	Функция
Работа с содерж имым заявки	представление  Batch in out  Unbatch in out	Преобразует заданное количество поступающих в объект заявок в одну заявку-партию.  Извлекает все заявки, содержащиеся в поступающей заявке-партии и пересылает их далее.  Сама заявка-партия при этом уничтожается.
	Pickup in out in out inPickup	Удаляет заявки из заданного объекта Queue и добавляет их к содержимому поступающей заявки-контейнера.
	Dropoff in out Out OutDropoff	Удаляет избранные заявки из поступающей заявки-контейнера и пересылает их далее.
	Assembler  in1  out in2  in3  in4  in5  access	Осуществляет сборку одной новой заявки из определенного числа заявок, пришедших из различных источников (до 5).
Обрабо тка	Delay in out	Задерживает заявки на заданный период времени.
Работа с ресурса ми	ResourcePool	Задает набор ресурсов, которые могут захватываться и освобождаться заявками.
	Seize outPreempted outTimeout in	Захватывает для заявки заданное количество ресурсов определенного типа.
	Release in out access	Освобождает ранее захваченные заявкой ресурсы.
	Service outPreempted outTimeout in -1 -1 out access	Занимает для заявки заданное количество ресурсов, задерживает заявку, а затем освобождает занятые ею ресурсы.

В этой и последующих моделях используется модифицированный класс транзакта MyEntity с параметрами: время нахождения в системе и время вхождения, приведённый ниже.

```
1.
     * MyEntity
2.
3.
   public class MyEntity extends
4.
    com.xj.anylogic.libraries.enterprise.Entity
    implements java.io.Serializable {
5.
       public double timeArrived = 0;
6.
       public double timeIn = 0;
7.
         * Конструктор по умолчанию
8.
9.
10.
        public MyEntity() {
11.
12.
         * Конструктор, инициализирующий поля
13.
14.
15.
        public MyEntity(double timeArrived) {
16.
               this.timeArrived = timeArrived;
17.
18.
       @Override
19.
       public String toString() {
20.
        return
        "timeArrived = " + timeArrived +" ";
21.
22.
23.
        * Это число используется при сохране-
24.
   нии состояния модели<br>
25.
        * Его рекомендуется изменить в случае
   изменения класса
26.
       private static final long
27.
    serialVersionUID = 1L;
28.
```

Кроме этого класса во всех моделях используются блоки для сбора статистики (statistics, workTimeOneTransact\_dataset, workTimeGist\_data) и блоки-файлы (workTimeGist, workTimeOneTransact, transactCount) для сохранения статистики в файле.

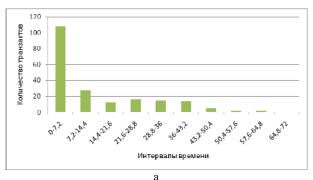
Для каждого элемента требуется ввести значения параметров во вкладке «Свойства», в соответствии с условиями задачи. как показано на рис.2 – рис.7. Блок source генерирует заявки по равномерному закону 10±6 единиц времени. Блок delay задерживает заявки по равномерному закону 9±7 единиц времени и записывает время задержки в statistics. Блок sink уничтожает заявки и записывает время их пребывания в системе. Событие event отслеживает количество заявок в системе и сохраняет его. Событие stopEvent заканчивает моделирование и регистрирует результаты моделирования. Описание программы ниже:

```
Действие:

workTimeGist.print(workTimeGist_data.toString());
queue_timeGist.print(queue_timeGist_data.toString());
for (int i=0; i<workTimeOneTransact_dataset.size(); i++)
    workTimeOneTransact.printf("%f\n",workTimeOneTransact_dataset.getY(i));
for (int i=0; i<transactCount_dataset.size(); i++)
    transactCount.printf("%f\n",transactCount_dataset.getY(i));
for (int i=0; i<queue_length_dataset.size(); i++)
    queue_length.printf("%f\n",queue_length_dataset.getY(i));

text.setText("Количество транзактов в системе: " +
    (source.count() - sink.count()));
text1.setText("Занятость delay: " +
    (delay_statistics.sum()/time()));
finishSimulation();
```

Количество сгенерированных транзактов: 201. Количество решенных задач: 200. Занятость устройства: 92%. Наиболее интересные результаты можно получить в графическом виде, как это показано на рис.2 а, б.



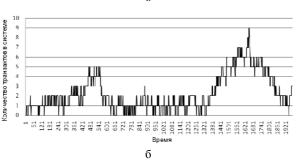


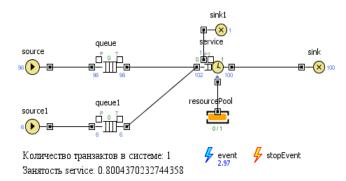
Рис. 2 — а) График распределения времени задержки заявки в системе, б) График изменения количества заявок в модели

В системе AnyLogic не формируются отчеты, привычные в GPSS World и необходимо самим вводить параметры, по которым необходимо получить результат.

По результатам моделирования сделаем заключение, что среднее время обслуживания устройством одного транзакта - 8.792 сравнительно ненамного отличается от заданного среднего значения -9.0; коэффициент использования устройства – 0.854 также не на много отличается от отношения среднего времени обслуживания к среднему времени между поступлением транзактов - 0.9. По статистическим данным по функционированию очереди отметим, что средняя длина очереди равна – 0.548, количество входов в очередь с нулевым временем ожидания – 78, среднее время ожидания в очереди – 5.6 и без учёта «нулевых» транзактов – 9.169. Таким образом, можно сделать заключение о том, что результаты моделирования не противоречат здравому смыслу.

Модель 2. СМО с генераторами транзактов с нулевым и первым приоритетами. Для нулевого приоритеа ГТ (равномерный закон  $10\pm4$ ) — очередь неограниченной длины — ОА (равномерный закон  $8\pm5$ ). Для первого приоритеа ГТ (равномерный закон  $10\pm4$ ) — очередь неограниченной длины — ОА (экспоненциальный закон, среднее 25). Отказы транзактам с нулевым приоритетам при поступлении транзактов с первым приоритетом. Остановить моделирование после вывода из модели 100 транзактов.

На рис.3 приведена структурная схема модели 2, сформированная в системе AnyLogic, с результатами моделирования.



Puc. 3 — Структурная схема модели 2, сформированная в системе AnyLogic, с результатами моделирования

Листинг класса MyEntity такой же, как и в предыдущем примере. Далее для каждого элемента введем значения параметров во вкладке «Свойства», в соответствии с условиями задачи. Блок source генерирует транзакты по равномерному закону  $10\pm4$ . Блок *source1* генерирует транзакты по равномерному закону  $150\pm60$ . Блоки *queue* и *queue1* выполняют роль буфера и собирают статистику о своем состоянии. Блок service задерживает транзакт по равномерному закону 8±5 либо по экспоненциальному закону со средним 25 в зависимости от приоритета, выполняет выталкивание менее приоритетных задач и записывает время задержки. Блок sink уничтожает транзакты и записывает время их пребывания в системе. Блок sink1 уничтожает транзакты, обработка которых прервана по приоритету. Событие event следит за количеством транзактов в системе и сохраняет его. Событие stopEvent прерывает моделирование и сохраняет статистику.

Все блоки модели 2, кроме блока service, аналогичны блокам модели 1. Так как событие stopEvent значительно отличается от аналогичного события примера 1, то приведём его настройки:

Результаты моделирования можно привести в графическом виде, наиболее интересные из них приведены на рис.4.

В результате моделирования получили следующие результаты:

Количество сгенерированных транзактов: 102.

Количество решенных задач: 100.

Транзактов осталось в системе: 1

Занятость устройства service: 0.8 Средняя длина очереди queue: 0.5

Среднее время задержки одного транзакта в очереди

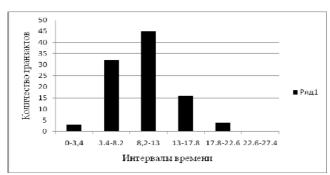
queue: 0

Средняя длина очереди queue1: 0

Среднее время задержки одного транзакта в очереди

queue1: 0

Средние время выполнения задачи: 8,872.



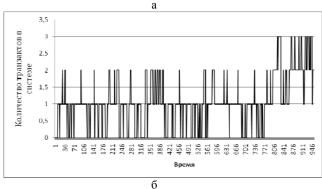


Рис. 4-a) График распределения времени пребывания транзакта в системе; б) График количества транзактов в системе по времени

По содержимому отчёта отметим, что в программе для неприоритетных транзактов для имитации очереди использована память с именем queue, а для приоритетных – очередь с именем queue1. Всего в модели обслужено 96 неприоритетных транзактов и 6 приоритетных. Обслуживание 1 неприоритетного транзакта прервано и он выведен из системы без обслуживания. Всего до завершения моделирования из системы выведено 100 транзактов.

Среднее время обслуживания устройством одного транзакта -8.353; коэффициент использования устройства -0.8. Среднее количество транзактов в памяти 1. Транзакты первого приоритета в очереди не задерживались.

**Модель 3.** СМО генератор транзактов (равномерный закон  $10\pm3$ ) —восемь устройств (равномерный закон  $100\pm50$  для каждого). Выбор устройства по правилу «первый свободный с наименьшим номером». Если все устройства заняты транзакт получает отказ. Остановить моделирование после вывода из модели 100 транзактов.

На рис.5 приведена структурная схема примера 2, сформированная в системе AnyLogic, с результатами моделирования.

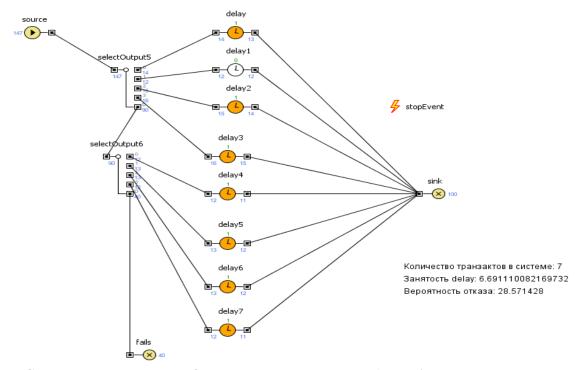


Рис. 5 – Структурная схема модели 3, сформированная в системе AnyLogic, с результатами моделирования

Модифицированный класс транзакта *МуЕптіту*, приведён далее, с параметрами: время нахождения в системе, время вхождения, времена начала и конца задержки.

```
чала и конца задержки.
29. /**
30. * MyEntity
31.
32. Public class MyEntity extends
    com.xj.anylogic.libraries.enterprise.Entity
    implements java.io.Serializable {
33.
34. Public double timeArrived = 0;
35. public double timeIn = 0;
36. public double delayIn = 0;
37. public double delayOut = 0;
38.
39.
        * Конструктор по умолчанию
40.
41. public MyEntity() {
42.
        }
43.
44.
45.
         * Конструктор, инициализирующий поля
46.
47. public MyEntity(double timeArrived){
48.
               this.timeArrived = timeArrived;
49.
50.
51.
       @Override
52.
       public String toString() {
53.
               return
54.
                       "timeArrived = " +
    timeArrived +" ";
55.
56.
57.
        * Это число используется при сохране-
58.
   нии состояния модели<br>
59.
        * Его рекомендуется изменить в случае
   изменения класса
60.
       privatestaticfinallong serialVersionUID
61.
   = 1L;
62.
63.
```

Кроме этого класса во всех моделях используются блоки для сбора статистики (statistics, workTimeOneTransact\_dataset, workTimeGist\_data, etc).

Блок *source* генерирует транзакты по равномерному закону.

Блок *selectOutput* распределяет транзакты между delay1 и delay n.

Блоки *queue* используются для формирования очередей транзактов.

Блок *delay* задерживает транзакты и записывает время их задержки в *statistics*.

Блок sink уничтожает транзакты и записывает время их пребывания в системе, обновляет графики «Распределение времени выполнения» и «Время выполнения одного транзакта в блоке delay».

Событие *event* отслеживает количество транзактов в системе и регистрирует его.

Событие *stopEvent* прерывает моделирование и регистрирует статистику.

Далее для каждого элемента вводятся значения параметров во вкладке «Свойства», в соответствии с условиями задачи.

По содержимому стандартного отчёта GPSS сделаем выводы, что устройства сравнительно сильно загружены, коэффициент использования первого устройства — 0,938; восьмого — 0,606. Разница в загрузке устройств значительная — 0,332. В систему для обслуживания поступило 107 транзактов, из них 87 обслужено; 7 находятся на обслуживании и 13 получили отказ из-за занятости устройств. Вероятность отказа 0,13. Время пребывания транзакта в системе 76.292, стандартное отклонение времени пребывания в системе 27.762.

Недостаток: сравнительно значительная разница в загрузке устройств, что объясняется при-

нятой дисциплиной выбора «первый свободный с наименьшим номером».

#### Заключение

По сравнению систем GPSS W с расширенным редактором, BPwin-Arena и AnyLogic можно отметить, что по наглядности представления структурных моделей первое место занимает AnyLogic, второе - BPwin-Arena и третье - GPSS W с расширенным редактором. По простоте освоения систем первое место занимает BPwin-Arena, второе - GPSS W с расширенным редактором и третье — AnyLogic. По простоте внесения в систему изменений с необходимостью введения новых программных моделей первое место занимает GPSS W с расширенным редактором, второе - BPwin-Arena и третье — AnyLogic.

AnyLogic – мощный инструмент имитационного моделирования, который позволяет моделировать любые системы с помощью основных методов моделирования или комбинации двух и более методов в одной модели для достижения лучшего результата. Кроме удобного справочного материала, находящегося непосредственно в самом программном продукте, быстроте начала работы также способствует продуманный и интуитивно понятный интерфейс.

Исходя из этого, можно порекомендовать моделирование в системе BPwin-Arena экономистам, тем более что в этой системе есть возможность моделирования бизнес-процессов по рабочим календарям. Система AnyLogic наиболее хорошо понятна научно- техническому персоналу и имеет явно выраженный инженерный стиль. Систему GPSS W с расширенным редактором можно рекомендовать к применению специалистам информационного профиля (программистам) в условиях, когда требуется

«дописывать» значительные объёмы новых программ.

Отметим, что при формировании заключения сказался имеющийся опыт некоторых авторов, в частности, первый автор более 30 лет ведёт занятия по моделированию на языке GPSS и только последние два года в системах GPSS W с расширенным редактором, BPwin-Arena и AnyLogic.

## Литература

- 1. Якимов И.М., Кирпичников А.П., Мокшин В.В., Костюхина Г.В., Шигаева Т.А. Комплексный подход к моделированию сложных систем в системе BPwin-Arena // Вестник Казанского технологического университета. 2014. Т. 17. № 6. С. 287-292.
- 2. Якимов И.М., Кирпичников А.П., Мокшин В.В. Моделирование сложных систем в среде имитационного моделирования GPSS W с расширенным редактором // Вестник Казанского технологического университета. 2014. Т. 17. № 4. С. 298-303.
- 3. Карпов Ю. Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic. СПб: БХВ-Петербург, 2006. 400 с.
- 4. Мокшин В.В., Якимов И.М. Метод формирования модели анализа сложной системы / Информационные технологии, №5. М.: Изд-во Новые технологии, 2011. С. 46-51.
- 5. Мокшин В.В., Кирпичников А.П., Шарнин Л.М. Отслеживание объектов в видео потоке по значимым признакам на основе фильтрации частиц // Вестник Казанского технологического университета. Казань: КНИ-ТУ, 2013. Т. 16. № 18. С. 297-303.
- 6. Степанова М.А., Сытник А.С., Кирпичников А.П., Мокшин В.В. Оптимизация процесса ремонта грузоподъемных машин по математической модели // Вестник Казанского технологического университета. 2013. Т. 16. № 20. С. 309-314.
- 7. Мокшин В.В., Якимов И.М., Юльметьев Р.М., Мокшин А.В. Рекурсивно-регрессионная самоорганизация моделей анализа и контроля сложных систем // Нелинейный мир. М: 2009. №1. С. 48-63.

<sup>©</sup> И. М. Якимов - канд. техн. наук, проф. каф. автоматизированных систем обработки информации и управления КНИТУ-КАИ им А.Н.Туполева; А. П. Кирпичников - д-р физ.-мат. наук, зав. каф. интеллектуальных систем и управления информационными ресурсами КНИТУ, kirpichnikov@kstu.ru; В. В. Мокшин - канд. техн. наук, доц. каф. автоматизированных систем обработки информации и управления КНИТУ-КАИ им А.Н.Туполева, vladimir.mokshin@mail.ru.

<sup>©</sup> I. M. Yakimov- PhD, Professor of the Department of Automated Information Processing Systems & Control, KNRTU-KAI; A. P. Kirpichnikov - Dr. Sci, Head of the Department of Intelligent Systems & Information Systems Control KNRTU, kirpichnikov@kstu.ru; V. V. Mokshin - PhD, Associate Professor of the Department of Automated Information Processing Systems & Control, KNRTU-KAI, vladimir.mokshin@mail.ru.