

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 моделированию 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 BWin-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 помощью 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 ± 6) - очередь неограниченной длины - обслуживающий аппарат (ОА) (равномерный закон 9 ± 7). Остановить моделирование после вывода из модели 200 транзактов. На рис.1 приведена структурная схема примера 1, сформированная в системе AnyLogic, с результатами моделирования.

Рис. 1 - Структурная схема примера 1 в системе AnyLogic

Таблица 1 - Объекты палитры основной библиотеки

Категория	Наименование	Графическое представление	Функция
Поток заявок	Source	Создает заявки	
Sink	Уничтожает поступающие заявки		
Enter	Вставляет уже существующие заявки в определенное место внутри процесса, заданного потоковой диаграммой.		
Exit	Извлекает		

поступающие в объект заявки из процесса, заданного потоковой диаграммой, позволяя пользователю самому решать, что следует сделать с этими заявками.

Hold Блокирует/разблокировывает поток заявок на определенном участке структурной схемы.

Split Для поступающей заявки объект создает заданное число новых заявок (копий) и посылает их дальше.

Combine Дождется поступления двух заявок в порты in1 и in2 (в произвольном 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 (детерминистических или заданных с помощью вероятностей) условий.

Queue 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 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.

Работа с содержимым заявки Batch Преобразует заданное количество поступающих в объект заявок в одну заявку-партию.

Unbatch Извлекает все заявки, содержащиеся в поступающей заявке-партии и пересылает их далее. Сама заявка-партия при этом уничтожается.

Pickup Удаляет заявки из заданного объекта Queue и добавляет их к содержимому поступающей заявки-контейнера.

Dropoff Удаляет избранные заявки из поступающей заявки-контейнера и пересылает их далее.

Assembler Осуществляет сборку одной новой заявки из определенного числа заявок, пришедших из различных источников (до 5).

Обработка Delay Задерживает заявки на заданный период времени.

Работа с ресурсами ResourcePool Задаёт набор ресурсов, которые могут захватываться и освобождаться заявками.

Seize Захватывает для заявки заданное количество ресурсов определенного типа.

Release Освобождает ранее захваченные заявкой ресурсы.

Service 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. /**
2.  * MyEntity
3.  */
4. public class MyEntity extends
com.xj.anylogic.libraries.enterprise.Entity implements java.io.Serializable {
5. public
double timeArrived = 0;
6. public double timeIn = 0;
7. /**
8.  * Конструктор по
умолчанию
9.  */
10. public MyEntity(){
11. }
12. /**
13.  * Конструктор,
инициализирующий поля
14.  */
15. public MyEntity(double timeArrived){
16. this.timeArrived = timeArrived;
17. }
18. @Override
19. public String toString() {
20. return
21. "timeArrived = " + timeArrived + " ";
22. }
23. /**
24.  * Это число
  
```

используется при сохранении состояния модели

25. * Его рекомендуется изменить в случае изменения класса 26. */ 27. private static final long 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 106 единиц времени. Блок delay задерживает заявки по равномерному закону 97 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: 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.

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 незначительно 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 приоритета ГТ (равномерный закон 10 ± 4) - очередь неограниченной длины - ОА (равномерный закон 8 ± 5). Для первого приоритета ГТ (равномерный закон 10 ± 4) - очередь неограниченной длины - ОА (экспоненциальный закон, среднее 25). Отказы транзактам с нулевым приоритетом 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. Рис. 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 104. Блок source1 генерирует транзакты по равномерному закону 15060. Блоки queue и queue1 выполняют роль буфера и собирают статистику о своем состоянии. Блок service задерживает транзакт по равномерному закону 85 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 Среднее время выполнения задачи: 8,872. а б Рис. 4 - а) 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 ± 3) -восемь устройств (равномерный закон 100 ± 50 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 MyEntity, приведён далее, с параметрами: время нахождения в системе, время вхождения,

времена начала 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. /**
 58. * Это число используется при сохранении состояния модели
 59. * Его рекомендуется изменить в случае изменения класса 60. */ 61.
 private static final long serialVersionUID = 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 принятой 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.