УДК 004.94

В.В. ЛИТВИНОВ[*], А.А. ЗАДОРОЖНИЙ[*], И.В. БОГДАН[*]

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ БЛОЧНОГО ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ АВТОМАТИЗИРОВАННЫХ СИСТЕМ В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ ВХОДНОЙ ИНФОРМАЦИИ

[*]Черниговський национальный технологический университет, г. Чернигов, Украина

Анотація. *У роботі розглядаються інформаційна технологія блочного імітаційного моделювання автоматизованих систем, її основні етапи і складові елементи, а також інструментальні засоби, які забезпечують можливість створення блочних імітаційних моделей з дворівневою архітектурою в рамках інформаційної технології. Запропонована архітектура дає можливість створювати як блоки рівня управління, призначені для управління модельним часом, так і функціональні блоки як у вигляді математичних виразів, так і з використанням індуктивних методів.*
Ключові слова: *інформаційна технологія, індуктивне моделювання, автоматизована система, мова моделювання.*

Аннотация. *В статье рассматриваются информационная технология блочного имитационного моделирования автоматизированных систем, ее основные этапы и составляющие элементы, а также инструментальные средства, которые обеспечивают возможность создания блочных имитационных моделей с двухуровневой архитектурой в рамках информационной технологии. Предложенная архитектура дает возможность создавать как блоки уровня управления, предназначенные для управления модельным временем, так и функциональные блоки как в виде математических выражений, так и с использованием индуктивных методов.*
Ключевые слова: *информационная технология, индуктивное моделирование, автоматизированная система, язык моделирования.*

Abstract. *Information technology of block simulation modeling of automated systems, its main stages and constituent elements and tools that enable the creation of block simulation models with two fold architecture within information technology are discussed in the paper. The proposed architecture gives opportunity to create both control-level blocks intended for controlling model time, and functional blocks, in the form of mathematical expressions and using inductive methods as well.*
Keywords: *information technology, inductive modeling, automated system, modeling language.*

1. Введение

Современные языки моделирования направлены на имитацию больших разветвленных систем, в которых процессы могут протекать как последовательно, так и параллельно, данные могут быть как дискретными, так и непрерывными, а процессы, протекающие в моделируемых системах, могут быть слабоизученными, количество данных, полученных при проведении экспериментов с реальной системой, может быть большим [1]. Создание моделей систем в таких условиях является сложной задачей, требующей определенных подходов к ее решению, для чего в данной работе и была предложена информационная технология, предоставляющая методы и инструментальные средства, позволяющие создавать имитационные модели в таких условиях. Целью данной статьи является описание этапов предложенной информационной технологии блочного имитационного моделирования автоматизированных систем, позволяющей создавать имитационные модели автоматизированных систем, а также описание инструментальных средств для поддержки каждого из этапов создания имитационных моделей.

2. Структура информационной технологии блочного имитационного моделирования автоматизированных систем

На рис. 1 представлена структура информационной технологии блочного имитационного моделирования автоматизированных систем в условиях неопределенности входной информации.

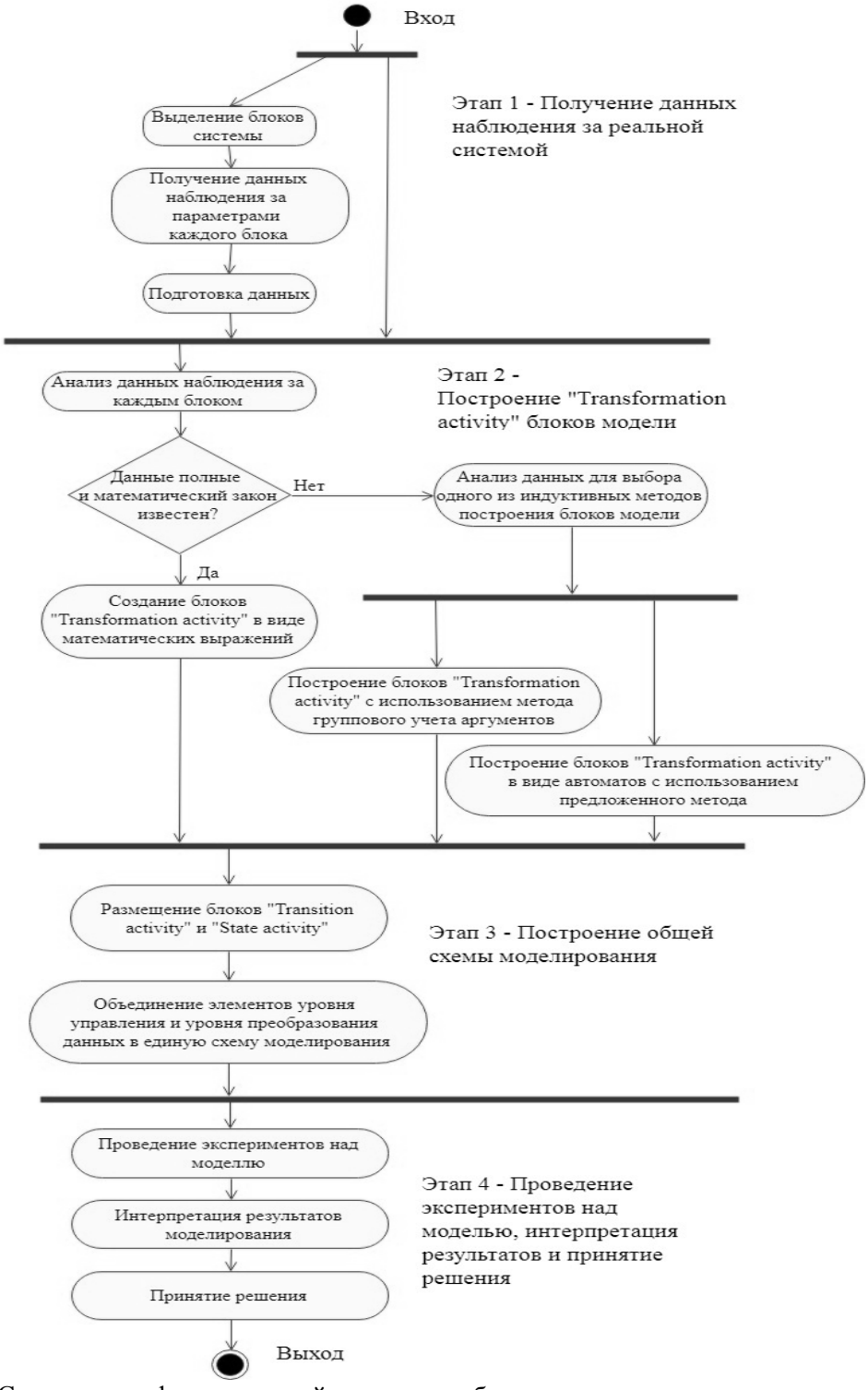

Рис. 1. Структура информационной технологии блочного имитационного моделирования автоматизированных систем в условиях неопределенности входной информации

Информационная технология включает такие основные этапы.

Первый этап. Получение данных наблюдения за реальной системой в виде временных рядов. На этом этапе система, для которой создается модель, разбивается на блоки и над каждым из блоков производятся наблюдения.

Второй этап. Построение с использованием данных наблюдения за каждым блоком элементов «Transformation activity». Если закон функционирования простой и известный, то блок создается в виде математического выражения, если же закон функционирования сложный, то блок создается с использованием индуктивного подхода.

Третий этап. Построение общей схемы моделирования. Элементы уровня преобразования и уровня управления объединяются в одну схему моделирования.

Четвертый этап. Проведение экспериментов, интерпретация результатов моделирования и принятие решения.

3. Архитектура инструментальных средств информационной технологии блочного имитационного моделирования систем в условиях неопределенности входной информации

На рис. 2 представлена архитектура информационной технологии блочного имитационного моделирования систем в условиях неопределенности входных данных. Инструментальные средства информационной технологии используют в качестве основы платформу Eclipse, которая содержит в себе функции для редактирования исходных кодов программ (Workspace), функции для организации командной работы над проектом (Team).

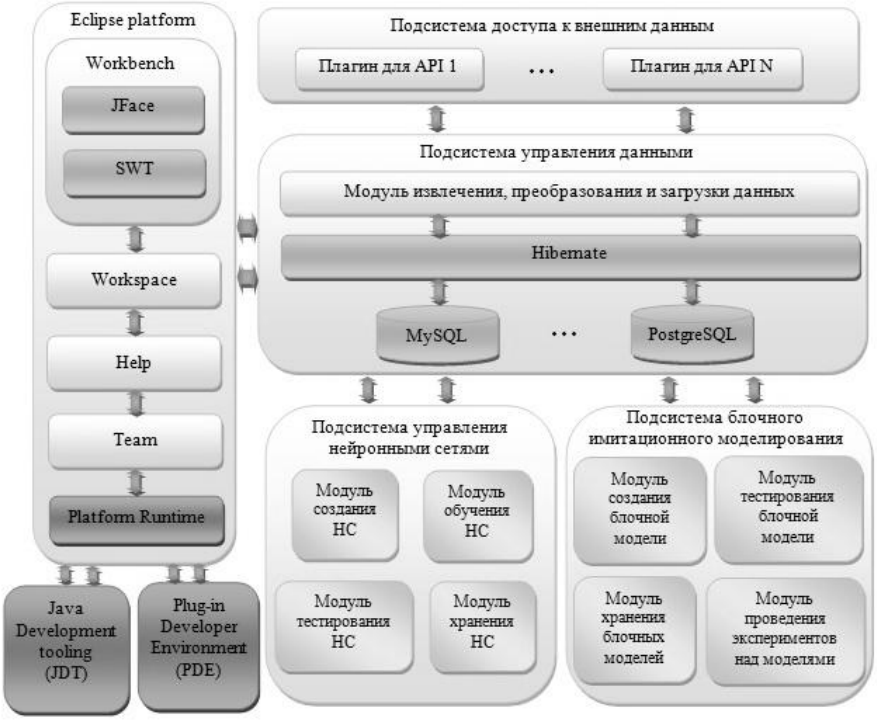

Рис. 2. Архитектура информационной технологии блочного моделирования систем в условиях неопределенности входной информации

Существует большое количество плагинов, которые могут значительно расширять возможности среды разработки. Например, с помощью плагинов можно добавить функции сохранения промежуточных результатов работы в систему контроля версий. Подсистема доступа к внешним данным предоставляет интерфейсы для добавления к существующему функционалу системы плагинов для доступа к сторонним источникам данных в виде файлов различных форматов.

Подсистема блочного моделирования систем состоит из модуля создания блочной модели, модуля тестирования блочной модели, модуля хранения блочной модели и модуля проведения экспериментов над моделями.

В рамках исследования возможности применения нейронных сетей для моделирования временных рядов было разработано распределенное инструментальное средство обучения и тестирования нейронных сетей. Использование распределенной архитектуры было обусловлено тем, что обучение нейронной сети занимает много времени и ресурсов.

Архитектура распределенного инструментального средства обучения представлена на рис. 3. В качестве протокола взаимодействия между менеджером и серверами обучения и тестирования нейронных сетей был выбран протокол RMI. Использование данного протокола позволило передавать сообщения между менеджером и серверами в виде сериализованных объектов. Модуль создания нейронных сетей позволяет создавать наборы нейронных сетей с различными параметрами, такими как размер входного вектора, количество скрытых слоев, количество нейронов в скрытых слоях. Выбрать вышеописанные параметры помогает модуль выбора параметров нейронной сети. Для выбора параметров нейронной сети используются методы статистического анализа.

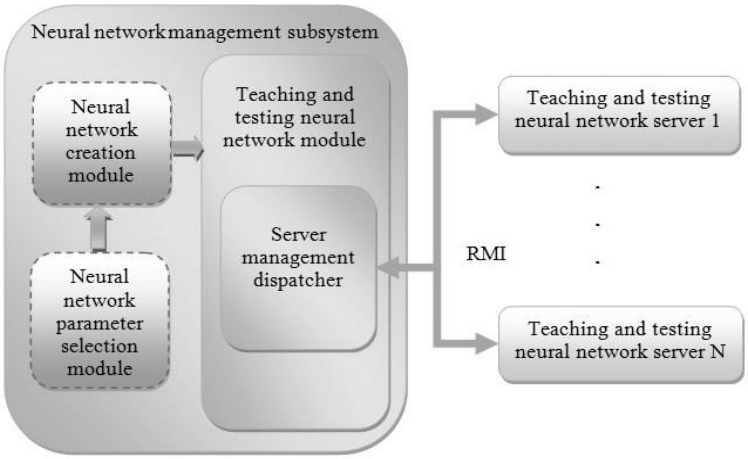

Рис. 3. Архитектура инструментального средства обучения нейронных сетей

Пользователь инструментального средства может создать с использованием модуля создания нейронных сетей набор необученных нейронных сетей. Серверы обучения и тестирования нейронных сетей устанавливают соединение с менеджером серверов тестирования и обучения, и менеджер добавляет их в список свободных серверов. Пользователь на основании списка необученных нейронных сетей и списка серверов обучения и тестирования может создать набор заданий для серверов обучения и тестирования.

Пользователь может выполнить список заданий по обучению нейронных сетей. Одному серверу может быть назначено одновременно несколько нейронных сетей. В таком случае сервер выполнит все задания последовательно. Все серверы выполняют задания одновременно и распределенно. По завершении выполнения задания сервер передает обученную нейронную сеть либо результаты тестирования сети по протоколу RMI диспетчеру серверов. Обученная нейронная сеть сохраняется в базе данных, что позволяет сохранять состояние обученной нейронной сети при выходе из инструментального средства.

Обучение нейронной сети происходит на сервере обучения и тестирования нейронной сети. Сервер обучения и тестирования может подключиться к менеджеру серверов, а менеджер добавляет сервер в список серверов обучения и тестирования.

Инструментальное средство позволяет проверить результаты обучения нейронной сети с использованием набора средств, размещенных на закладке Experiments management.

4. Новый язык блочного моделирования для создания блочных имитационных моделей систем в условиях неопределенности входной информации

Современный язык имитационного моделирования должен предоставлять возможность создания сложных имитационных моделей с разветвленной сетевой структурой. Если говорить о блочном языке имитационного моделирования, то он, как правило, включает в себя два аспекта: функциональные блоки и управляющие блоки. Функциональные блоки предназначены для осуществления различных функциональных преобразований. Функциональные блоки могут быть представлены преобразованиями в виде формул либо в виде автоматов. Таким образом, современный язык имитационного моделирования является двухуровневым [1] (рис. 4):

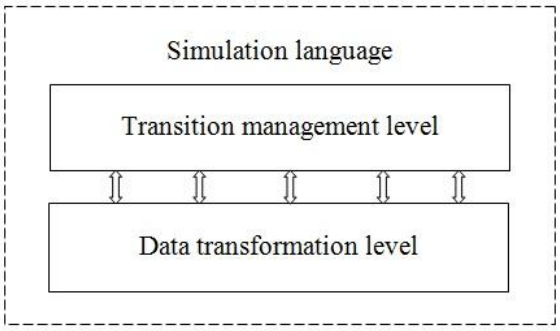

Рис. 4. Двухуровневая структура современного языка имитационного моделирования

Существует много разных подходов по представлению общей схемы моделирования, каждый из которых имеет свои достоинства и недостатки. Одним из способов представления общей схемы моделирования является сеть Петри [2]. Достоинством сетей Петри является то, что в ней хорошо представлен управляющий уровень языка в виде маркеров и переходов, которые срабатывают при поступлении необходимого количества маркеров.

Вторым способом представления общей схемы моделирования является механизм агрегатов [3]. Агрегаты позволяют представить на едином языке детерминированные и стохастические объекты, которые функционируют как непрерывно, так и дискретно. Понятие агрегата определяется на основании единого подхода к формализации процесса функционирования системы:

– состояние системы в данный момент времени определяется предыдущими состояниями и входными сигналами, поступившими в данный момент времени и ранее;

– выходной сигнал в данный момент времени определяется входными сигналами и состояниями системы, которые относятся к данному состоянию и предшествующим состояниям.

С позиций моделирования агрегат выступает как универсальный преобразователь информации: за конечный интервал времени он воспринимает конечное число входных сигналов и выдает конечное число выходных сигналов. Одним из видов входных сигналов являются управляющие сигналы.

Агрегат имеет входные контакты, на которые в моменты времени t_j поступают входные сигналы. Входной сигнал x является элементом некоторого множества $X : x \in X$. Входной сигнал является вектором, размерность которого равна числу входных контактов. Входной сигнал может быть представлен конечным набором элементарных сигналов $x_1(t),...x_n(t), x_i \in X_i, i = 1, n$, одновременно возникающих на входе агрегата.

На другие особые контакты системы поступают управляющие сигналы в моменты времени τ_i. Управляющий сигнал g является элементом множества $G : g \in G$. За конечный интервал времени в агрегат поступает конечное число входных и управляющих сигналов. Совокупность входных сигналов, расположенных в порядке их поступления, называется входным сообщением, соответственно управляющих сигналов – управляющим сообщением.

Выходной сигнал агрегата y является элементом некоторого множества Y и определяется по состояниям агрегата $z(t)$ при помощи оператора G. За конечный интервал времени оператор выдает конечное число выходных сигналов. В общем случае оператор является случайным оператором. Совокупность выходных сигналов, упорядоченная относительно времени выдачи, называется выходным сообщением.

В каждый момент времени $t \in (0,T)$, в который функционирует система, агрегат находится в одном из возможных состояний. В общем случае множество T может быть непрерывным, дискретным или дискретно-непрерывным. Пример возможной схемы агрегатной системы приведен на рис. 5.

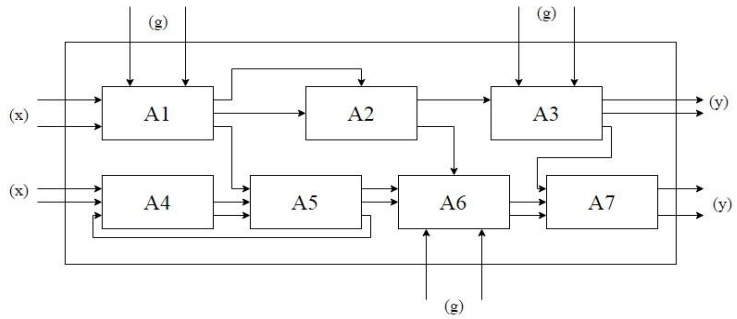

Рис. 5. Пример агрегатной схемы

Таким образом, агрегаты хорошо подходят для произведения последовательных и параллельных функциональных преобразований. Связи между агрегатами представлены с использованием оператора сопряжения, что является неудобным, а механизм управления переходами между агрегатами отсутствует.

Третьим способом представления общей схемы моделирования являются механизмы, предложенные в архитектуре WRIGHT [4]. Достоинством WRIGHT по сравнению с агрегатами является то, что в ней четко определено понятие канала данных (Connector), в отличие от агрегатов, в которых понятие канала данных отсутствует. Механизм управления переходами в архитектуре WRIGHT отсутствует. Как язык архитектурного описания язык WRIGHT построен на базе трех архитектурных абстракций: компонентов, разъемов и конфигураций. Язык WRIGHT предоставляет нотацию для каждого из этих элементов, формализуя понятие компонента как вычислителя и соединителя, как паттерна взаимодействия.

Компонент (component) представляет локальное независимое вычисление. В WRIGHT определение компонента состоит из двух важных частей – интерфейса и вычислителя. Интерфейс состоит из набора портов, которые предназначены для взаимодействия компонента с другими компонентами.

Соединитель (connector) представляет взаимодействие между набором компонентов. Например, соединитель Pipe представляет последовательный поток данных между фильтрами. Соединитель определяет требования, которым должны соответствовать компонент и информация, которая определяет, что компонент может ожидать от внешней системы. WRIGHT определяет соединитель как набор элементов Role и элемент Glue. Каждый элемент Role определяет поведение каждого участника взаимодействия.

Таким образом, для совмещения в общей схеме моделирования механизма функциональных преобразований, механизма передачи данных и механизма управления переходами необходимо применить комбинированный подход, который бы использовал механизм функциональных преобразований, использованный в агрегатном подходе, механизм передачи данных, использованный в архитектуре WRIGHT, и механизм управления, который применяется в сетях Петри.

5. Сравнение этапов жизненного цикла моделей и программ и жизненные циклы, пригодные для создания моделей

Рассмотрим, какие этапы жизненного цикла программ и моделей схожи между собой (рис. 6).

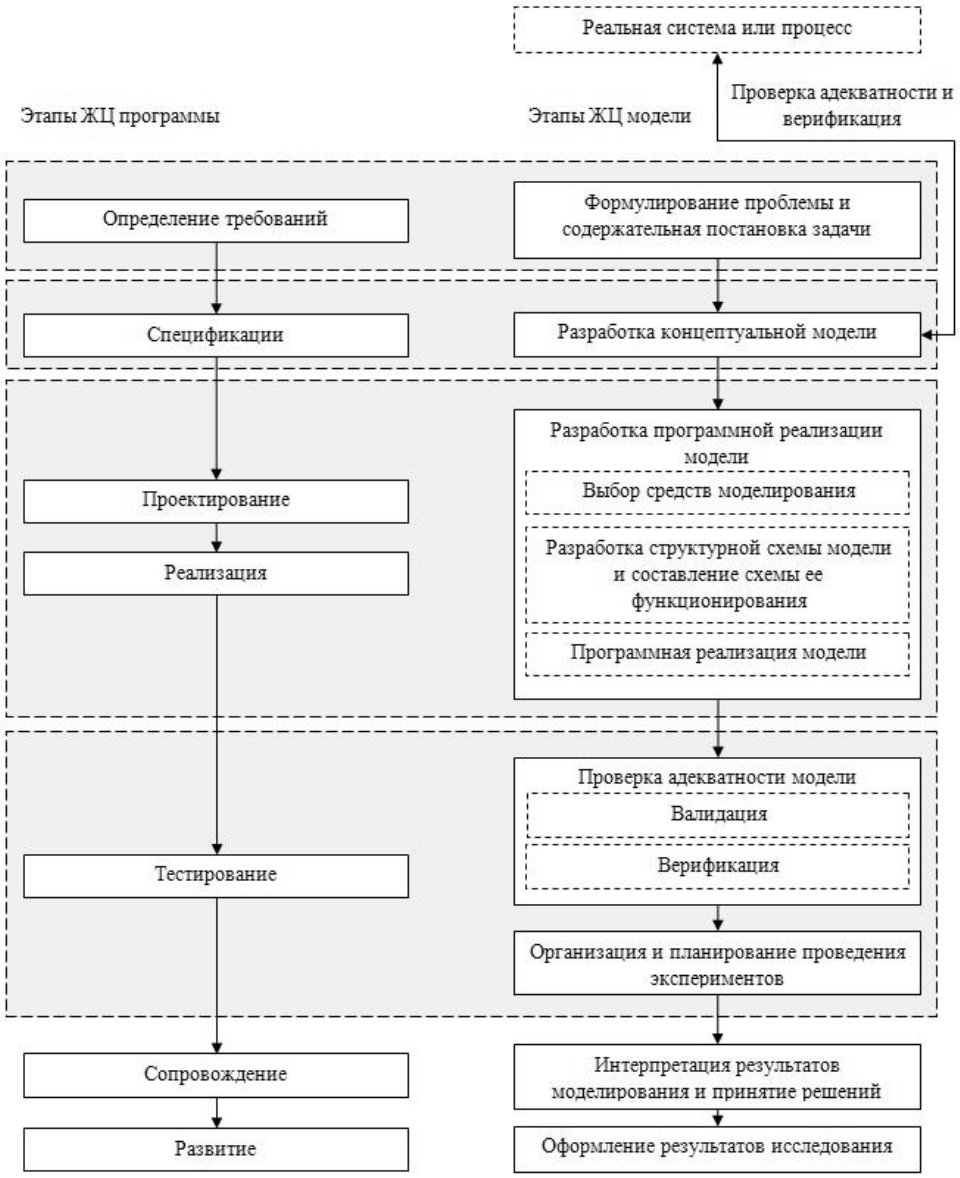

Рис. 6. Сравнение этапов жизненного цикла программы и модели

Если для моделирования сложной системы создается имитационная модель, то некоторые этапы жизненного цикла моделей очень схожи с этапами жизненного цикла программ.

Например, если говорить об этапе формулирования проблемы и содержательной постановке задачи, то одной из важных задач, которую необходимо решить на этом этапе при создании имитационной модели системы, является определение требования к модели и к моделируемой системе.

Таким образом, при создании модели системы можно выделить два вида требований: требования к моделируемой системе и требования к модели. Требования к системе можно получить из готовой системы или из разрабатываемой системы. Например, если мы создаем модель программной системы, то, скорее всего, требования к ней уже были выдвинуты на этапе анализа требований к системе. После того, как требования к системе и к модели системы были выдвинуты, их необходимо проверить, чтобы они были непротиворечивы.

И для этого можно использовать те же инструменты, которые используются в жизненном цикле программы на этапе анализа требований к системе. Также на этапе форму-

лирования проблемы решаются вопросы, связанные с определением целей создания программы либо модели, определением временных рамок разработки и ресурсов, которые можно затратить на разработку. Производится активное взаимодействие с заказчиком для более четкого формирования требований.

Этап спецификации жизненного цикла разработки программы соответствует этапу разработки концептуальной модели в жизненном цикле создания модели. Спецификацию программы можно считать ее концептуальной моделью, особенно, если спецификация представлена в виде сущности, описанной на некотором специализированном языке спецификаций. Задачей этапа спецификации является сбор всех требований и устранение противоречивости этих требований. Этап разработки концептуальной модели также в некоторой мере предназначен для устранения противоречивостей, таких как создание максимально точной модели при как можно большем ее упрощении по сравнению с реальным объектом, что позволит сэкономить ресурсы, затрачиваемые при создании модели, а также ресурсы, затрачиваемые на прогоны модели.

Наиболее схожими являются этапы проектирования и реализации в ЖЦ создания программ и этап разработки программной реализации модели. Схожесть проявляется особенно четко в том случае, если для создания был выбран прикладной язык программирования. Этап разработки программной реализации у модели тесно связан с разработкой концептуальной модели, хотя разработка имитационной модели очень сильно похожа на разработку обычной программы. Но разработка модели происходит на основании зафиксированной архитектуры, которая выбирается на основании разработки концептуальной модели.

Этап разработки концептуальной модели при создании имитационной модели системы состоит из трех шагов:

1. Выбор степени детализации описания объекта моделирования, на котором определяется, насколько детально должен быть описан объект моделирования. На этом шаге необходимо выбрать между стоимостью модели и погрешностями моделирования. Чем выше будет степень детализации объекта моделирования, тем дольше будет происходить процесс ее разработки и калибровки, и тем меньше будут отклонения от реальной системы полученных результатов.

2. Описание переменных модели, на котором определяются входные, выходные, внутренние параметры и их распределения.

3. Формализованное изображение концептуальной модели, которое является самым важным шагом на этапе разработки концептуальной модели и тесно связывает этот этап с этапом разработки программной реализации модели.

Необходимо рассмотреть, каким образом формализованное изображение концептуальной модели влияет на этап программной реализации модели, а также определить, что такое программа с фиксированной архитектурой. Фиксированная архитектура имитационной программы появляется в тот момент, когда происходит формальное изображение концептуальной модели. После того, как был выбран формализм для описания имитационной модели, у этого формализма, как правило, есть набор подходов, которые применяют, чтобы на их основании создать программную реализацию модели.

Этап тестирования ЖЦ программы соответствует этапам проверки адекватности модели, организации и планирования проведения экспериментов. Также можно говорить о том, что экспериментирование над программной реализацией модели можно проводить с использованием таких же инструментальных средств, как и при ее тестировании. Можно говорить о том, что процесс экспериментирования над моделью схожий с процессом ее тестировании, но без сравнения результатов эксперимента с заведомо известным результатом. Вместо того, чтобы сравнивать результат прогона модели с заведомо известным результатом, производятся их постобработка и вывод окончательного результата на экран.

Последовательные модели жизненного цикла (те, в которых движение происходит от одного этапа к другому без возвратов) являются идеализированными, так как только очень простые задачи проходят все этапы без каких-либо итераций. При программировании, например, может обнаружиться, что реализация некоторой функции очень громоздка, неэффективна и вступает в противоречие с требуемой от системы производительностью. В этом случае требуется перепроектирование, а, может быть, и изменение спецификаций. При разработке больших нетрадиционных систем необходимость в итерациях возникает регулярно на любом этапе жизненного цикла как из-за допущенных на предыдущих шагах ошибок и неточностей, так и из-за изменений внешних требований к условиям эксплуатации системы. Наиболее развитым в плане понятия итерации является объектно-ориентированный подход. Для традиционных подходов итерация – это исправление ошибок, то есть процесс, который с трудом поддается технологическим нормам и регламентам. При объектно-ориентированном подходе итерации никогда не отменяют результаты друг друга, а всегда только дополняют и развивают их.

При создании моделей так же, как и при разработке программного обеспечения, применяется командный подход, когда над проектом работают несколько разработчиков одновременно. Командный подход требует набора специальных инструментальных средств, которые обеспечивают эффективную работу над проектом. Важными инструментами при командной работе над проектом являются система контроля версий, сервер непрерывной интеграции и фреймворки модульного и функционального тестирования. Связь инструментальных средств командной разработки программ и инструментальных средств командной разработки моделей представлены на рис. 7 и 8. Данные инструментальные средства помогают сохранять текущее состояние файлов, связанных с проектом (систем контроля версий VCS), а также производить прогон тестов (сервер непрерывной интеграции CI). Модульные и функциональные тесты используются для того, чтобы определить, не наступил ли регресс в программном коде, а также для определения возможности завершения текущей итерации.

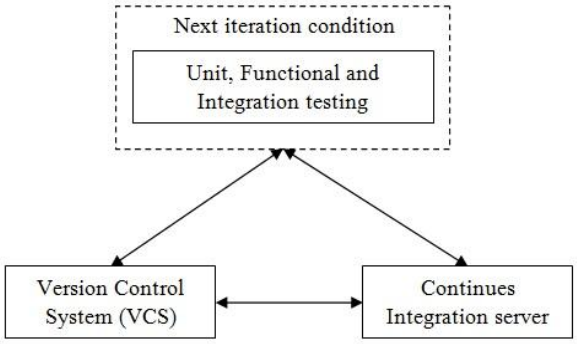

Рис. 7. Связь инструментальных средств командной разработки программ

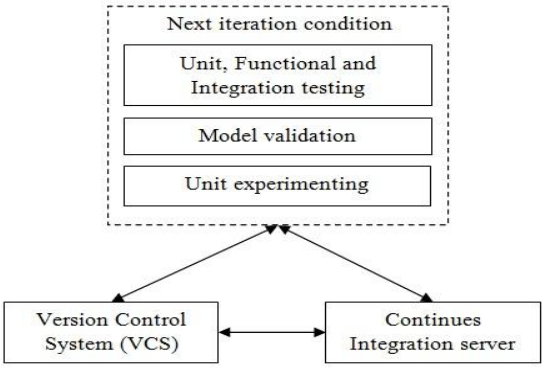

Рис. 8. Связь инструментальных средств командной разработки моделей

Учитывая то, что для определения завершения итерации при создании модели необходимо не только произвести тестирование программной реализации модели, а также произвести ее верификацию, необходимо расширить набор инструментальных средств командной разработки инструментальными средствами автоматической верификации моделей. Это даст возможность производить верификацию и тестирование моделей в автоматическом режиме в процессе создания модели, а также по завершении итерации модели жизненного цикла.

6. Выводы

В данной статье предложена новая информационная технология блочного имитационного моделирования автоматизированных систем в условиях неопределенности входной информации, а также рассмотрены ее основные этапы, составляющие элементы и инструментальные средства, которые обеспечивают возможность создания блочных имитационных моделей с двухуровневой архитектурой. Предложенная архитектура позволяет создавать как блоки уровня управления, предназначенные для управления модельным временем, так и функциональные блоки как в виде математических выражений, так и с использованием индуктивных методов.

СПИСОК ИСТОЧНИКОВ

1. Киндлер Е. Языки моделирования / Киндлер Е. – М.: Энергоатомиздат, 1985. – 288 с.
2. Питерсон Дж. Теория сетей Петри и моделирование систем / Питерсон Дж. – М.: Мир, 1984. – 256 с.
3. Полляк Ю.Г. Вероятностное моделирование на электронных вычислительных машинах / Полляк Ю.Г. – М.: Советское радио, 1971. – 400 с.
4. Allen R.J. A Formal Approach to Software Architecture / Allen R.J. – Pittsburgh: Carnegie Mellon University, 1997. – 236 p.

Стаття надійшла до редакції 20.02.2018