Опубликовано в Научно-техническом вестнике СПбГУ ИТМО. 2008. Выпуск 53. Автоматное программирование, с. 286-295.

УДК 004.4’233

ПРИМЕНЕНИЕ КОНЕЧНЫХ АВТОМАТОВ В ДОКУМЕНТООБОРОТЕ

В.О. Клебан, Ф.А. Новиков
(Санкт-Петербургский государственный университет информационных технологий, механики и оптики)

В работе рассматривается задача построения системы автоматизации документооборота. Для построения системы автоматизации документооборота применяются конечные автоматы, так как такой подход имеет ряд преимуществ по сравнению с традиционным. В качестве примера применения предлагаемого подхода рассматривается построение инструментального программного средства на базе *Microsoft Office*, которое предназначено для разработки и внедрения автоматизированной системы менеджмента качества в компаниях с *проектным* типом организации производства.
Ключевые слова: конечный автомат, документооборот

Введение

Проблема автоматизации бизнес-процессов на сегодняшний день достаточно остро стоит перед предприятиями. Многие крупные предприятия уже обзавелись системами электронного документооборота, тогда как малые и средние – практически нет и продолжают использовать разрозненные документы *Microsoft Office* (в редких случаях с использованием распределенного хранилища *Microsoft Groove*).

Традиционный способ автоматизации бизнес-процессов – разработка прикладного программного обеспечения – постепенно вынужден отходить на второй план ввиду того, что внесение даже небольших изменений в схему бизнес-процесса означает необходимость перепрограммирования и большие затраты времени и средств. В результате прикладные программы не успевают обновляться в том темпе, который диктуют изменяющиеся условия бизнеса и потребности самого предприятия.

Активно развивающийся бизнес, связанный с автоматизацией предприятий, требует большого количества обученных кадров из-за больших объемов работ. При этом количество квалифицированных специалистов в области автоматизации растет недостаточно быстро.

Таким образом, стоит задача создания простого в освоении, надежного средства автоматизации, имеющее в своем арсенале не только средства описания документооборота, но и его исполнения. Возможность исполнения является ключевым моментом, ведь чисто описательное средство интересно только с точки зрения анализа бизнес-процессов и может быть применено при реализации конкретной модели документооборота лишь как часть технического задания. Во всех известных авторам случаях «описательная» часть системы автоматизации документооборота оторвана от «исполнительной» части, что делает невозможным быстрый переход от спроектированной схемы к реализации.

Существующие технологии автоматизации бизнес-процессов

Business Process Execution Language (*BPEL*) – язык на основе *XML* для формального описания бизнес-процессов и протоколов их взаимодействия между собой [1]. *BPEL* расширяет модель взаимодействия веб-сервисов и включает в эту модель поддержку транзакций. Язык *BPEL* определяет два вида процессов – абстрактный и исполняемый. Абстрактный процесс определяет протокол обмена сообщениями между раз-

личными участниками, не раскрывая алгоритмы их внутреннего поведения. В отличие от абстрактного, исполняемый процесс содержит в себе алгоритмы, определяющие порядок выполнения *деятельностей* (activities), назначение исполнителей, обмен сообщениями, правила обработки исключений и т.д. Главной проблемой данного языка является организация человеко-машинного взаимодействия, так как данный язык ориентирован на полностью независимые от пользователя бизнес-процессы.

IDEF0 – методология и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является ее акцент на соподчиненность объектов. В IDEF0 рассматривается логические отношения между работами, а не их временная последовательность. Также на диаграмме отображаются все сигналы управления. Данная модель является одной из самых заслуженных моделей и используется при организации бизнес-проектов и проектов, основанных на моделировании всех процессов – как административных, так и организационных [2]. Основной проблемой данной методики является отсутствие операционной семантики, т.е. невозможность исполнить нарисованные диаграммы.

Нотация Architecture of Integrated Information Systems (ARIS) – методология и программный продукт компании IDS Scheer для моделирования бизнес-процессов [3]. Методология ARIS рассматривается с четырех точек зрения: организационная структура, функциональная структура, структура данных и структура процессов. Для описания бизнес-процессов предлагается использовать большое количество (около 120) типов моделей, каждая из которых описывает тот или иной аспект, кроме того, пользователь может сам определять новые типы моделей. Все это делает данную методологию гибкой, но трудно воспринимаемой, а большое количество типов моделей осложняет обучение новых специалистов.

Использование конечных автоматов

Наряду с описанием процессов в виде блок-схем (схем алгоритмов), существует другой подход – автоматный [4]. Данный подход заключается в представлении процесса в виде системы взаимодействующих автоматов. Автоматы могут взаимодействовать по вложенности (один автомат вложен в одно или несколько состояний другого автомата), по вызываемости (один автомат вызывается с определенным событием из выходного воздействия, формируемого при переходе другого автомата), по обмену сообщениями (один автомат получает сообщения от другого) и по номерам состояний (один автомат проверяет, в каком состоянии находится другой автомат). Вложенность может рассматриваться как вызываемость с любым событием. Ни число автоматов, вложенных в состояние, ни глубина вложенности не ограничены. Такое представление позволяет более компактно описывать поведение программы, модуля, а в нашем случае – жизненный цикл документа либо бизнес-процесса. Компактное представление, в свою очередь, улучшает наглядность.

Конечный автомат может быть представлен в виде простейшей языковой конструкции, состоящей из одного или нескольких операторов `SWITCH`. Такой подход делает возможным формальное и изоморфное автоматическое преобразование автомата в код программы. Повышается наблюдаемость программы за счет сокращения количества наблюдаемых переменных.

Автоматные программы могут быть эффективно верифицированы методом проверки на моделях (*Model Checking*) [5], так как в таких программах управляющие состояния явно выделены, а их количество обозримо. Это позволяет строить компактные модели Крипке даже для программ большой размерности.

Автоматные программы показали свою эффективность при построении «реактивных систем»[6]. Документооборот, по сути, является такой системой. Документы реагируют на действия пользователей, а бизнес-процессы на изменения в документах, то есть *система управляется событиями*.

В качестве примера использования данной технологии рассмотрим применение предлагаемого подхода к построению инструментального программного средства на базе *Microsoft Office*, которое предназначено для разработки и внедрения автоматизированной системы менеджмента качества (АСМК) в компаниях с *проектным* типом организации производства. В разработке также используется продукт *Microsoft Groove*, который позволяет организовать распределенное хранилище документов. При проектном типе организации производства *реализации* одного и того же бизнес-процесса могут отличаться в разных проектах в рамках одной организации. Автоматизированная система менеджмента качества – это частично или полностью программно реализованная система менеджмента качества определенная в стандарте ИСО 9001: 2000.

Принципы работы и средства описания

Система представляет собой инструментальное программное средство (конструктор), предназначенный для разработки и внедрения АСМК.

Определим следующие роли пользователей системы:

- разработчик – изготовитель инструментария;
- инженер – пользователь инструментария, осуществляющий автоматизацию предприятия;
- пользователь – сотрудник предприятия, использующий систему для выполнения повседневных обязанностей.

Инструментарий состоит из следующих основных средств, которые обеспечивают базовую функциональность:

- система времени выполнения (клиент и сервер);
- транслятор моделей жизненного цикла бизнес-процессов;
- транслятор моделей жизненного цикла документов.

Используя средства инструментария, инженер определяет в графическом виде модели бизнес-процессов и модели жизненных циклов документов, а также производит их стыковку с системой времени выполнения.

Моделью жизненного цикла *активного* объекта (документа или бизнес-процесса) является конечный автомат, в котором определены состояния объекта, условия перехода из одного состояния в другое и выходные воздействия (эффекты). Условие перехода является булевой формулой, в которой участвуют *входные воздействия* и *события.* Входное воздействие представляет собой функцию, как правило, оперирующую с записями о качестве (в терминах ИСО 9001) и возвращающую булево значение. Событие – это именованное сообщение, инициирующее переход автомата и поступающее на вход системы в произвольный момент времени. Событие может быть сгенерировано автоматом жизненного цикла документа, либо автоматом жизненного цикла бизнес-процесса, либо другим «поставщиком событий» в зависимости от задачи.

Рассмотрим в качестве примера процесс исправления ошибки в программном продукте, упрощенная модель которого представлена на рис. 1. На данной модели выделено два состояния: «Обнаружена ошибка» и «Ошибка исправлена». Используются два события, на которые реагирует модель: «Создан отчет об ошибке» и «Отчет об ошибке обновлен» и условия «Ошибка исправлена», «Ошибка не исправлена». Также

используется выходное воздействие «Сообщить об исправлении». Стрелками показано направление перехода.

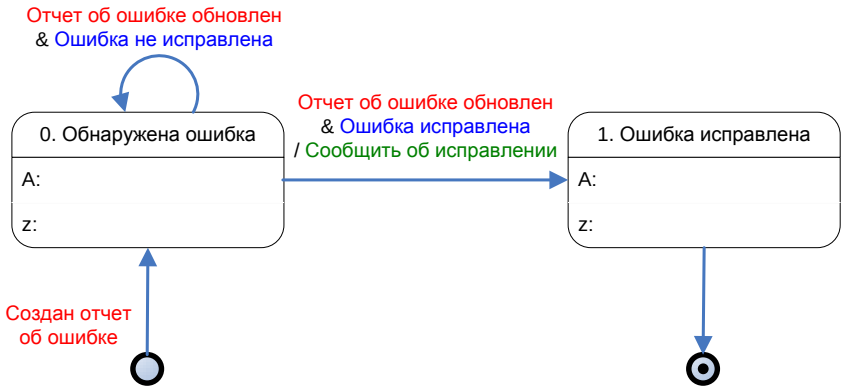

Рис. 1. Процесс исправления ошибки

Переход из состояния «Обнаружена ошибка» в состояние «Ошибка исправлена» следует читать так: при наступлении события «Отчет об ошибке обновлен» и истинности условия «Ошибка исправлена» совершить действие «Сообщить об исправлении» и перейти в состояние «Ошибка исправлена».

Как было указано ранее, активный объект системы способен реагировать на события от других объектов, а также сам порождать события (в эффектах). Например, документ при переходе из одного состояния в другое порождает событие, которое передается бизнес-процессу. Важно, что жизненный цикл документа исполняется на стороне клиента, а жизненный цикл бизнес-процесса на стороне сервера. Процесс передачи события проиллюстрирован на рис. 2.

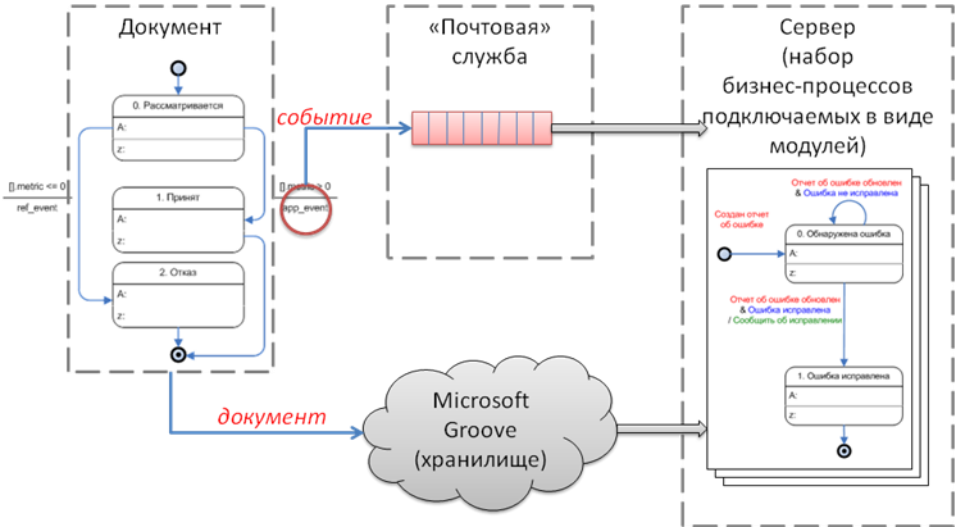

Рис. 2. Передача события из документа в бизнес-процесс

Документ при переходе из состояния «Рассматривается» в состояние «Принят» генерирует событие, которое попадает в «Почтовую службу». Она является временным хранилищем событий на стороне пользователя и сохраняет события (и порядок их следования), в том случае, если передача их на сервер затруднена или невозможна. Одновременно с отсылкой события на сервер происходит сохранение документа в распреде-

ленное хранилище *Microsoft Groove*. Сервер, после получения события и соответствующего ему документа, действует в соответствии с заложенными в него моделями жизненных циклов бизнес-процессов, например, извлекает из хранилища необходимый документ и анализирует, изменяет его содержимое.

Заметим, что для обеспечения функционирования АСМК в рамках организации необходимо соблюдение следующих *условий функционирования*:

- бизнес-процессы организации, подпадающие под действие АСМК, определены и документированы;
- управление бизнес-процессами отражено в электронных документах;
- выполнение бизнес-процессов основано на обработке событий;
- обрабатываемыми событиями являются события документов: создание, удаление, изменение;
- записи о качестве (метрики) хранятся в специальных полях документов.

Некоторые из данных условий справедливы не только в рамках внедрения АСМК, но и для документооборота в целом.

Структура документа

Документ делится на две составляющие: «модель» и «представление», объединенные в один файл в соответствии со стандартом *OpenXML*, который реализован в *Microsoft Office 2007*. К документу *Microsoft Office* прикреплено *XML*-хранилище, в котором хранятся служебные данные и значения записей (метрик). Подчеркнем, что данные (записи о качестве) хранятся непосредственно в документе.

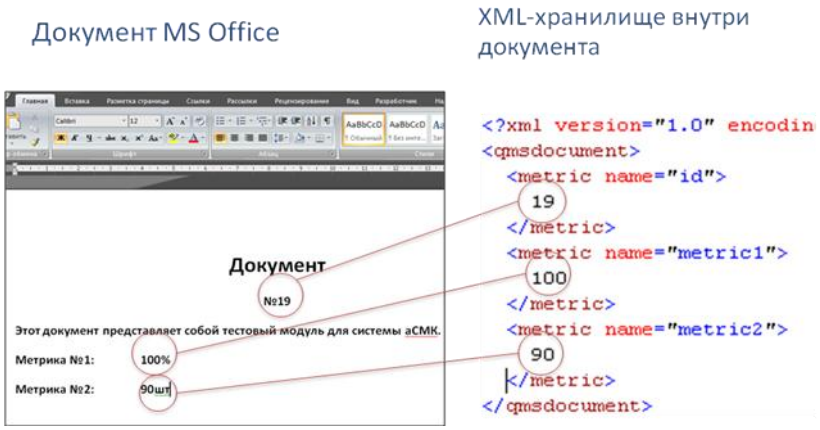

Рис. 3. Структура документа

Внутри документа также хранится его жизненный цикл, записанный в виде *XML*-представления (рис. 4).

Для возможности «отката» документа либо всей системы в предыдущее состояние в документе хранится также и история его изменения:

```
<!--История фиксирования состояний документа-->
<history>
  <state name="Review" date="22.02.2008 19:59:48" user="User">
    <docid>78ac5ada3142f3997306911d05c38dc2</docid>
    <processed>0</processed>
    <fromstate>Approved</fromstate>
    <metric name="metric1">01</metric>
    <metric name="metric2">02</metric>
    <metric name="metric3">01</metric>
  </state>
</history>
```

Также для активных объектов могут быть автоматически сгенерированы обратные автоматы при условии обратимости производимых операций [7].

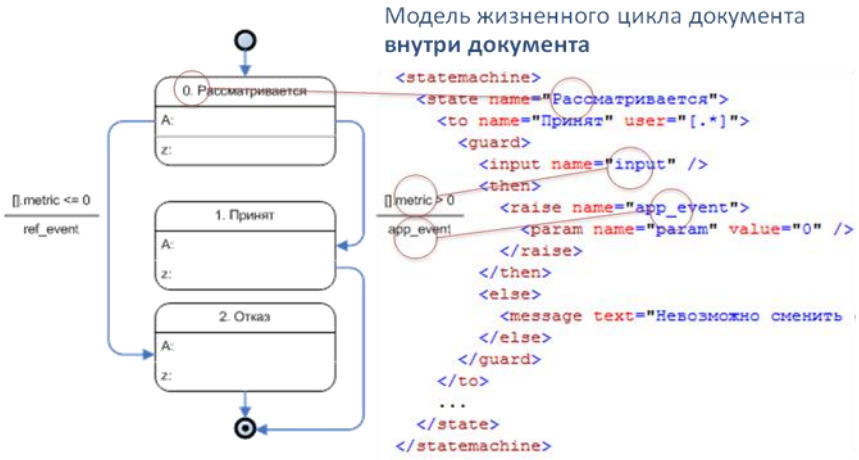

Рис. 4. Модель жизненного цикла и его представление

Структура модуля бизнес-процесса

Модуль бизнес-процесса представляет собой динамическую библиотеку (plug-in), которая подключается сервером АСМК.

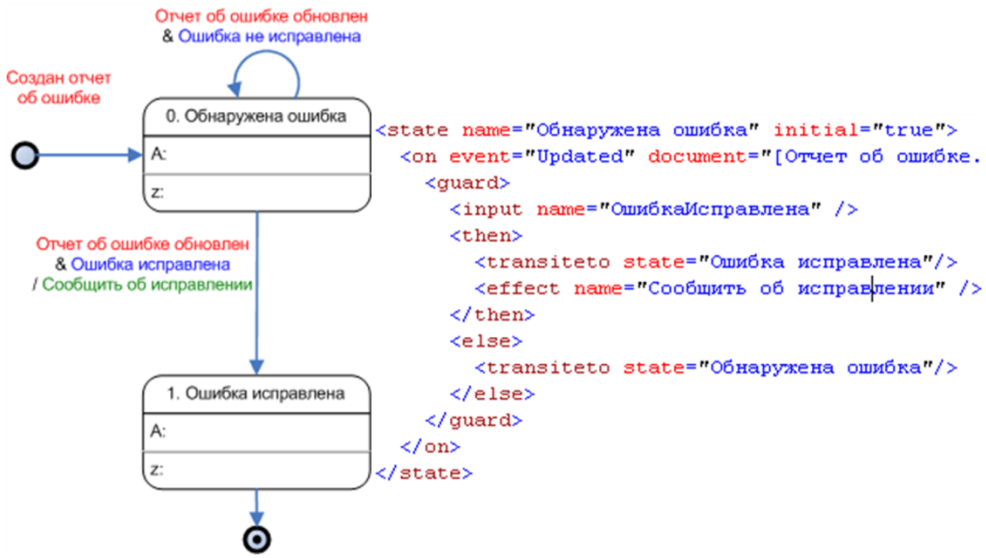

Рис. 5. Модель процесса исправления ошибки в графическом и XML представлениях

Устройство такой библиотеки аналогично устройству документа, с той разницей, что в документе модель жизненного цикла интерпретируется и хранится в виде XML-модели, а в модуле бизнес-процесса модель жизненного цикла скомпилирована. Модель жизненного цикла бизнес-процесса описывается аналогичным образом, в качестве примера приведем упрощенную модель процесса исправления ошибки и соответствующее *XML*-представление.

Описанные выше *XML*-модели, а в дальнейшем и исполняемый код, генерируются автоматически из графического представления конечных автоматов, созданных в редакторе *Microsoft Visio*, и текстовых описаний входных и выходных воздействий.

Полученная промежуточная *XML*-модель преобразуется в исходный код модуля на языке *C#* и компилируется соответствующим компилятором. Подробнее входные и выходные данные алгоритма генерации представлены на рис. 6.

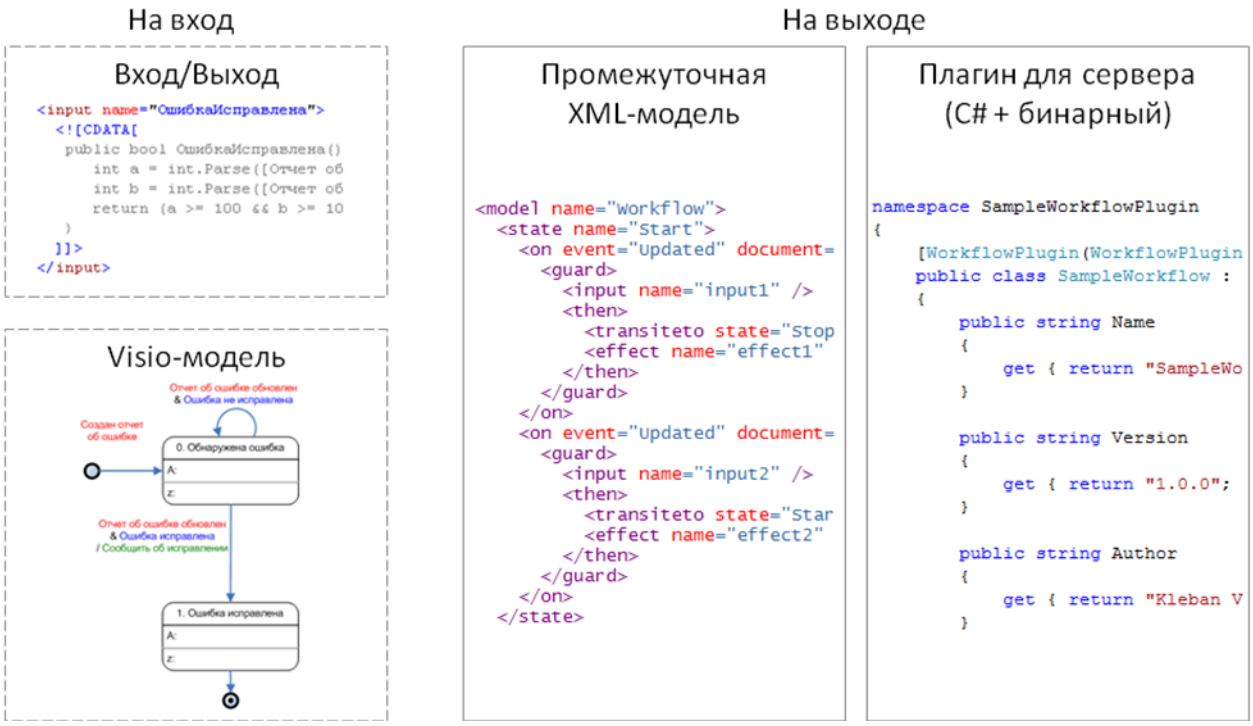

Рис. 6. Входные данные и результат работы

Язык представления модели

Модель жизненного цикла документа изначально представляется в графическом виде, но в документе хранится в виде *XML*-представления, формат которого приведен в таблице.

Таблица. Формат XML-представления

XML-тег	Комментарий
`<state name="Рассматривается">`	Имя состояния
`<to name="Принят" user="[.*]">`	Дуга из текущего состояния в состояние «Принят». Переход может быть инициирован любым пользователем. В случае указания квадратных скобок, имя пользователя сравнивается с регулярным выражением, в отсутствие скобок происходит прямое сравнение.

<guard> <input name="input" />	Переход из текущего состояния в состояние «принят» будет осуществлен, только если будет соблюдено условие с именем input
<then> <raise name="app_event"> <param name="param" value="0" /> </raise> </then>	Если условие соблюдено, то сгенерировать событие с именем app_event и параметром param=0
<else> <message text="Невозможно сменить состояние" /> </else> </guard> </to>	Если условие не соблюдено – вывести диалоговое окно, с текстом «Невозможно сменить состояние»
<to name="Refused" user="[.*]"> <raise name="MyExternalEvent"> <param name="MyParam" value="123" /> </raise> <message text="Document refused" /> </to> </state>	Пример перехода без условия (отсутствует guard)

Входные и выходные воздействия описываются с помощью языка C# в следующем виде:

```
<input name="ОшибкаИсправлена">
  <![CDATA[
   public bool ОшибкаИсправлена(){
      int a = int.Parse([Отчет об ошибке.docx].metric1);
      int b = int.Parse([Отчет об ошибке.docx].metric2);
      return (a >= 100 && b >= 100);
   }
  ]]>
</input>
```

Для удобства инженера в язык введена операция доступа к документам `[]`. Чтобы считать запись с именем «`metric1`» из документа «`Отчет об ошибке.docx`», необходимо использовать следующую операцию:

```
[Отчет об ошибке.docx].metric1
```

Заключение

Применение конечных автоматов в документообороте уже находит поддержку: например, в *Microsoft Workflow*, помимо представления документооборота в традиционной форме (в виде блок-схем), введено представление в виде конечных автоматов.

Использование конечных автоматов при автоматизации документооборота позволяет решить несколько важных задач:

- отделение в автомате логики работы от входных и выходных действий позволит в ряде случаев возложить обязанности по описанию бизнес-процессов не на инженера, а на руководящий состав;
- простота автоматной модели существенно снижает требования к разработчикам, которые занимаются автоматизацией предприятий, что очень важно при наблюдающемся дефиците специалистов;
- изменение модели жизненного цикла активного объекта (документа или процесса) не приводит к необходимости перепрограммирования, достаточно лишь изменить

схемы, так как большая часть программного кода генерируется автоматически по моделям;

- наблюдаемость автомата и его обратимость позволяют совершать «откат» системы в предыдущие состояния, не прибегая к сложным алгоритмам;
- возможность верификации автоматной модели позволяет строить системы правил, которые будут отвечать за ключевые моменты документооборота, что, в свою очередь, позволит проверять построенные модели еще до внедрения;
- классический способ наблюдения за показателями бизнеса – по метрикам (количество продукции и т.д.) – в случае использования автоматной модели может быть дополнен показателями, основанными на поведении моделей, а это означает, что повышается вероятность обнаружения «патологии» в работе предприятия.

Литература

1. Business Process Execution Language for Web Services version 1.1. – Режим доступа: http://www.ibm.com/developerworks/library/specification/ws-bpel/
2. Шеер А.-В. Моделирование бизнес-процессов. – М.: Весть-Метатехнология, 2000
3. Марка Д., МакГоуэн К. Методология структурного анализа и проектирования. – Режим доступа: http://www.marathon.ru/~fedor/doc/IDEF/ooad.asf.ru/standarts/idef/sadt/index.shtml
4. Шалыто А.А. Switch-технология. Алгоритмизация и программирование задач логического управления. –СПб.: Наука, 1998. http://is.ifmo.ru/books/switch/1
5. Автоматное программирование. Раздел «Верификация». http://is.ifmo.ru/verification/
6. Шалыто А.А., Туккель Н.И. SWITCH-технология — автоматный подход к созданию программного обеспечения «реактивных» систем // Промышленные АСУ и контроллеры. – 2000. – № 10. – Режим доступа: http://is.ifmo.ru/works/switch/1
7. Корнеев Г. А. Автоматизация построения визуализаторов алгоритмов дискретной математики на основе автоматного подхода. – Диссертация … канд. техн. наук /СПбГУ ИТМО. – 2006. – Режим доступа: http://is.ifmo.ru/disser/korn_disser.pdf