Шаблоны рабочего процесса: о выразительной силе

(На основе сети Петри) Языки рабочего процесса

W.M.P. van der Aalst[1,2] и A.X.M. тер Хофстеде[2]

[1] Факультет технологического менеджмента, Технологический университет Эйндховена
Почтовый ящик 513, NL-5600 MB, Эйндховен, Нидерланды.
w.m.p.v.d.aalst@tm.tue.nl

[2] Квинслендский технологический университет, Школа информационных систем
Почтовый ящик 2434, Брисбен, Квинсленд, 4001, Австралия.
a.terhofstede@qut.edu.au

Аннотация. Современные системы управления рабочими процессами основаны на явных моделях процессов, т.е. Для реализации данного рабочего процесса требуется полностью конкретизированный дизайн рабочего процесса. Создание дизайна рабочего процесса - это сложный, отнимающий много времени процесс, который часто затрудняется ограничениями используемого языка рабочего процесса. Чтобы выявить различия между различными языками, мы собрали довольно полный набор шаблонов рабочего процесса. Основываясь на этих шаблонах, мы оценили 15 продуктов рабочего процесса и обнаружили значительные различия в выразительной силе. Языки, основанные на сетях Петри, работают лучше, когда дело доходит до шаблонов рабочего процесса, основанных на состоянии. Однако некоторые шаблоны (например, включающие несколько экземпляров, сложные синхронизации или нелокальные с- отрисовки) нелегко отобразить на (высокоуровневые) сети Петри. Эти шаблоны создают интересные проблемы моделирования и используются для разработки языка YAWL на основе сети Петри (еще одного языка рабочего процесса).

1 Введение

Технология документооборота продолжает совершенствоваться в своих традиционных областях применения - моделировании бизнес-процессов и координации бизнес-процессов, а теперь и в новых областях компонентных структур и взаимодействия между рабочими процессами, взаимодействия между предприятиями. Для решения этой широкой и довольно амбициозной задачи в продаже имеется большое количество продуктов для документооборота, в основном систем управления рабочими процессами (WFMS), в которых используется большое разнообразие языков и концепций, основанных на различных парадигмах (см., Например, [1, 4, 12, 19, 23, 28, 31, 30, 40, 47]).

В качестве текущего положения сравниваются и по мере появления новых концепций и языков ЭМ- рявкнул на, поражает, как мало, кроме глоссарии стандартов, доступен для Центральной справочник. Одной из причин, объясняемых отсутствием консенсуса относительно того, что составляет спецификацию рабочего процесса, является разнообразие способов, с помощью которых бизнес-процессы описываются иным образом. Утверждается, что отсутствие универсальной организационной "теории" и стандартных концепций моделирования бизнес-процессов объясняет и, в конечном счете, оправдывает основные различия в языках документооборота, способствуя различию "лошадей для курсов" в языках документооборота. Более того, сравнение различных продуктов workflow сводится в большей степени к распространению продуктов и в меньшей степени к критике

возможности языка рабочего процесса - выделяются различия в спецификациях рабочего процесса "в целом" , а также технологические проблемы, обычно зависящие от платформы.

Спецификации рабочего процесса могут быть поняты в широком смысле с различных точек зрения (см. [4, 23]). В *Поток управления* перспективная (или процессная) перспектива описывает действия и порядок их выполнения с помощью различных конструкторов, которые позволяют управлять потоком выполнения, например, последовательностью, выбором, параллелизмом и синхронизацией соединений . Деятельности в начальной форме единицы работы, и в составной форме модулировать выполнение заказа комплекта деятельности. В *зрения данных* выравнивает бизнес и обрабатывает данные с точки зрения управления. Деловые документы и другие предметы , которые текут между деятельностью и локальные переменные рабочего процесса, квалифицируются по сути пред - и пост-условий жизнедеятельности исполнения. В *перспектива ресурса* обеспечивает ор- организационные структуры привязки к рабочему процессу в виде человека и устройства, ролей , ответственных за выполнение деятельности. В *оперативный* перспектива описывает элементарные действия, выполняемые activities, где действия сопоставляются с базовыми приложениями. Как правило, (упоминания) деятельности и рабочего процесса данные передаются в и из АП- складки через деятельность-к-интерфейсов приложений, позволяющих обрабатывать данные в приложениях.

Очевидно, что перспектива потока управления дает существенное представление об эффективности спецификации рабочего процесса . Перспектива потока данных основывается на нем, в то время как организационные и операционные аспекты являются вспомогательными. ..... Если спецификации рабочего процесса должны быть расширены для удовлетворения новых требований к обработке, конструкторам потока управления требуется фундаментальное понимание и анализ. В настоящее время большинство языков документооборота поддерживают базовые так называемые конструкции последовательности, итерации, разделения (AND and XOR) и объединения (AND and XOR) - см. [4, 30]. Однако интерпретация даже этих базовых конструкций неоднородна и часто неясно, как можно было бы поддерживать более сложные требования. Действительно, поставщикам предоставляется возможность рекомендовать "хаки" уровня реализации, такие как триггеры базы данных и обработка событий приложения. Результатом является отсутствие ни текущих возможностей языков документооборота, ни понимания более сложных требований бизнес-процессов.

Мы указываем требования к языкам документооборота в разделе workflow *Шаблоны* [5-8, 48]. Как описано в [36], паттерн "это абстракция от конкретной формы, которая продолжает повторяться в конкретных неслучайных контекстах". Гамма и др. [17] впервые систематизировали около 23 шаблонов проектирования, которые описывают наименьшие повторяющиеся взаимодействия в объектно-ориентированных системах. Шаблоны проектирования, как таковые, обеспечивали независимость от технологии реализации и в то же время независимость от существенных требований предметной области, которые они пытались удовлетворить (см. Также, например, [15]).

Мы собрали набор из примерно 30 шаблонов рабочих процессов и использовали 20 из этих шаблонов для сравнения функциональности 15 систем управления рабочими процессами (COSA, Visual Workflow, Forte Conductor, Lotus Domino Workflow, Meteor, Mobile, MQSeries/Work- flow, Staffware, Verve Workflow, I-Flow, InConcert, Changengine, SAP R/3 Workflow, Eastman и FLOWer). Результат этой оценки показывает, что (1) выразительная мощь современных систем оставляет желать лучшего и (2) системы поддерживают различные шаблоны. Обратите внимание, что мы не используем термин "выразительность" в традиционном или формальном смысле. Если абстрагироваться от ограничений пропускной способности, любой язык рабочего процесса является полным по Тьюрингу. Следовательно, имеет смысл сравнить эти языки, используя for-

неправильные понятия выразительности. Вместо этого мы используем более интуитивное понятие выразительности, которое учитывает усилия по моделированию. Это более интуитивное понятие часто называют пригодностью. Обсуждение различия между формальной выразительностью и пригодностью см. в [27].

Наблюдение о том, что выразительная мощь доступных систем управления рабочими процессами оставляет желать лучшего, вызвало вопрос: *Как насчет высокоуровневых сетей Петри (т. е. Сетей Петри, расширенных цветом, временем и иерархией) в качестве языка рабочего процесса?*

Сети Петри существуют с шестидесятых годов [35] и были дополнены цветом [24, 25] и временем [32, 33] для улучшения выразительности. Высокоуровневые инструменты сетей Петри, такие как Design/CPN (Орхусский университет, http://www.daimi.au.dk/designCPN/) и ExSpect (EUT / D & T Bakl [1]:

1. Формальная семантика, несмотря на графическую природу. 2. Основанный на состоянии, а не (только) на событии. 3. Обилие методов анализа.

К сожалению, простое применение высокоуровневых сетей Петри не дает желаемого результата. По-видимому, существуют три проблемы, актуальные для моделирования процессов рабочего процесса. :

1. В высокоуровневой сети Петри возможно использовать цветные маркеры. Хотя это возможно использовать это для идентификации нескольких экземпляров подпроцесса, конкретной поддержки для *шаблоны, включающие несколько экземпляров* и бремя отслеживания, разделения и объединения ложится на разработчика. 2. Иногда требуется объединить два потока, хотя неясно, выполняется ли синхронизация.

необходим, т.е., если оба потока активны, требуется соединение AND, в противном случае соединение XOR- . Такие *расширенные шаблоны синхронизации* их трудно моделировать в терминах высокоуровневой сети Петри, потому что локальным правилом перехода является либо AND-join, либо XOR-join. 3. Запуск перехода всегда является локальным и зависит только от токенов во входных местах

и влияет только на выходные места. Однако некоторые события в рабочем процессе могут иметь эффект, который не является локальным, например, из-за ошибки токены необходимо удалить из разных мест, не зная, где они находятся. Все, кто смоделировал такой шаблон *отмены* (например, глобальный механизм тайм-аута) в терминах

из сетей Петри известно, что моделировать так называемый "пылесос", удаляющий токены из выбранных частей сети, громоздко. В этой статье мы обсуждаем проблемы при поддержке шаблонов рабочего процесса с помощью высокоуровневых сетей Петри. Мы также кратко представляем разрабатываемый язык документооборота: *YAWL (еще один язык документооборота)*. YAWL основан на сетях Петри, но расширен дополнительными функциями для облегчения моделирования сложных рабочих процессов.

2 Шаблоны рабочего процесса

С 1999 года мы работаем над сбором всеобъемлющего набора шаблонов рабочего процесса [5-8]. Результаты доступны в разделе "Шаблоны рабочего процесса".

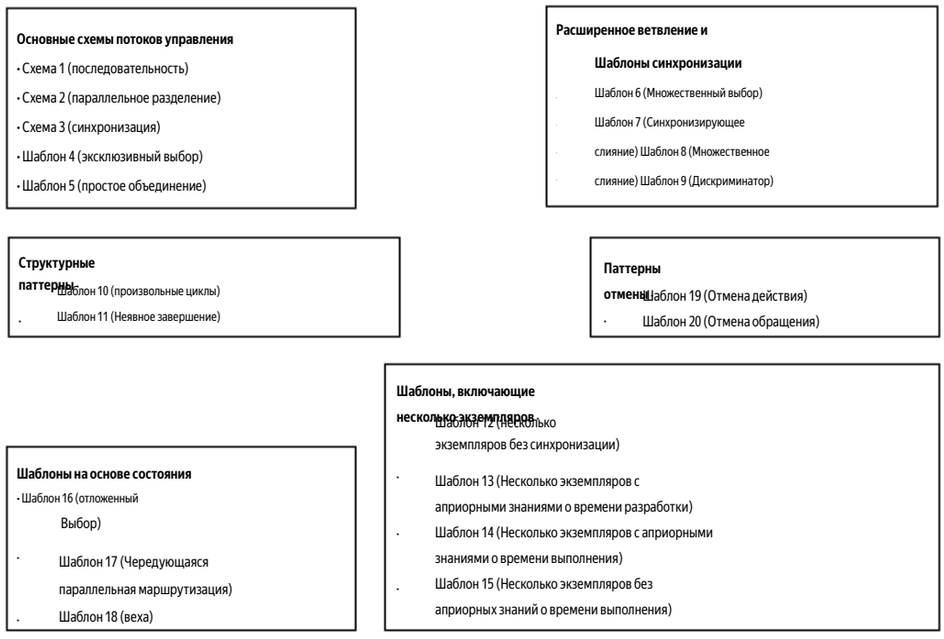


Рис. 1. Обзор 20 наиболее актуальных шаблонов.

Сайт WWW" [48]. Шаблоны варьируются от очень простых шаблонов, таких как последовательный маршрут (шаблон 1), до сложных шаблонов, включающих сложные синхронизации, такие как шаблон дискриминатора (шаблон 9). В этой статье мы ограничимся 20 наиболее релевантными шаблонами. Эти шаблоны можно разделить на шесть категорий:

1. *Базовые схемы потока управления.* Это базовые конструкции, присутствующие в большинстве рабочих процессов языки для моделирования последовательной, параллельной и условной маршрутизации.
2. *Расширенные шаблоны ветвления и синхронизации.* Эти шаблоны выходят за рамки базовых шаблонов, обеспечивая более продвинутые типы поведения разделения и объединения. Примером может служить синхронизирующее слияние (шаблон 7), которое ведет себя как соединение AND- или XOR-join в зависимости от контекста. 3. *Структурные шаблоны.* В языках программирования блочную структуру, в которой четко определяет точки входа и выхода вполне естественно. В графических языках, допуская параллелизма такое требование часто является слишком ограничительным. Следовательно, мы определили шаблоны, которые допускают менее жесткую структуру. 4. *Шаблоны, включающие несколько экземпляров.* В контексте одного случая (т.Е. Работы- flow instance) иногда части процесса необходимо создавать несколько раз, например, в контексте страхового требования необходимо обработать несколько свидетельских показаний. 5. *Шаблоны, основанные на состояниях.* Типичные системы документооборота фокусируются только на действиях и событиях , а не на состояниях. Это ограничивает выразительность языка рабочего процесса, потому что невозможно иметь шаблоны, зависящие от состояния, такие как шаблон Milestone (Этапный шаблон) (Шаблон 18).

6. *Шаблоны отмены.* Наступление события (например, клиент отменяет заказ) может привести к отмене мероприятий. В некоторых сценариях такие события могут даже привести к прекращению всего дела.

На рисунке 1 показан обзор 20 моделей, сгруппированных в шесть категорий. Подробное обсуждение этих закономерностей выходит за рамки данной статьи. Заинтересованному читателю отсылаем к [5-8, 48].

Мы использовали эти шаблоны для оценки 15 систем документооборота: COSA (Ley GmbH, [43]), Visual Workflow (Filenet, [13]), Forte Conductor (SUN, [14]), Lotus Domino Workflow (IBM/Lotus, [34]), Meteor (UGA/LSDIS, [41]), Mobile (UEN, [23]), MQ-Series/Workflow (IBM, [22]), Staffware (Staffware PLC, [44]), Verve Workflow (версия sata, [46]), I-Flow (Fujitsu, [16]), InConcert (TIBCO, [45]), Changengine (HP, [21]), SAP R/3 Workflow (SAP, [39]), Eastman (Истман, [42]) и FLOWer (Афина Паллада, [9]). В таблицах 1 и 2 обобщены результаты сравнения систем управления рабочими процессами с точки зрения выбранных шаблонов. Для каждой комбинации продукта и шаблона мы проверили, возможно ли реализовать шаблон рабочего процесса с помощью инструмента. Если продукт напрямую поддерживает шаблон через одну из своих конструкций, ему присваивается рейтинг +. Если шаблон не поддерживается *напрямую* , ему присваивается рейтинг +/-. Любое решение, результатом которого являются спагетти диаграммы или кодирование, рассматривается как не оказывающее прямой поддержки и оценивается как -.

шаблон	продукт							
	Staffware COSA		Неубедительный	Eastman FLOWer	Domino		Meteor	Мобильный
1 (seq)	+	+	+	+	+	+	+ +	+
2 (par-spl)	+	+	+ +	+ +	+ +	+	+	+
3 (синхронизация)	+	+	+/-	+	+	+	+	+
4 (ex-ch)	+	+	+/-	+	+	+	+	+
5 (simple-m)	+	+	+/-	+/-	- -	+	+	+
6 (m-choice)	-	+/-	+	+	+/-	+/-	- +	-
7 (синхронизация-m)	-	-	-	+	+/-	- +	+/-	+
8 (мульти-m)	-	-	- -	+	- +	+	-	-
9 (диск)	-	+ -	+	+	- -	+	+	-
10 (arb-c)	+	+/-	-	+	+ +	+/-	-	-
11 (impl-t)	+	+	+	+	+	+	+ +	-
12 (mi-no-s)	-	+	-	+	+	+	-	+
13 (mi-dt)	+	- -	-	- -	+	-	-	-
14 (mi-rt) 15	-	+	-	-	+/-	-	-	-
(mi-no) 16	-	+	-	-	+/-	-	-	+
(def-c) 17 (int-par)	-	+	-	-	+/-	-	-	-
18 (самый	-	+	-	-	+/-	-	-	+
короткий)	-	+	-	-	+/-	-	-	-
19 (can-a)	+	+	-	-	+/-	+	-	-
20 (can-c)	-	-	-	-	+/-		-	-

Таблица 1. Основные результаты для Staffware, COSA, InConcert, Eastman, FLOWer, Lotus Domino Workflow, Meteor и Mobile.

шаблон	продукта						
	MQSeries	Сильная сторона	Воодушевление	Vis. WF	Изменение.	I-Flow	SAP/R3
1 (seq)	+	+	+	+	+	+	+
2 (par-spl)	+	+	+	+	+	+	+
3 (синхронизация)	+	+	+	+	+	+	+
4 (ex-ch)	+	+	+	+	+	+	+
5 (simple-m)	+	+	+	+	+	+	+
6 (m-choice)	+	+	+	+	+	+	+
7 (синхронизация-m)	+	-	-	+	-	-	-
8 (мульти-m)	-	+	+	- -	-	-	-
9 (диск)	-	+	+	- +/-	+	-	+
10 (arb-c)	-	+	+	-	+	+	-
11 (impl-t)	+	-	-	+	-	-	-
12 (mi-no-s)	-	+	+	+	-	+	-
13 (mi-dt)	+	+	+	-	+	+	+
14 (mi-rt) 15	-	-	-	-	-	-	+/-
(mi-no) 16	-	-	-	-	-	-	
(def-c) 17 (int-par)	-	-	-	-	-	-	- -
18 (самый	-	-	-	-	-	-	-
короткий)	-	-	-	-	-	-	-
19 (can-a)	-	-	-	-	-	-	+
20 (can-c)	-	+	+	-	+	-	+

Таблица 2. Основные результаты для MQSeries, Forte Conductor, Verve, Visual WorkFlo, Changengine, I-Flow и SAP / R3 Workflow.

Пожалуйста, с осторожностью применяйте результаты, обобщенные в таблицах 1 и 2. Прежде всего, организация, выбирающая систему управления документооборотом, должна сосредоточиться на шаблонах, наиболее подходящих для текущих процессов документооборота. Поскольку поддержка более продвинутых шаблонов ограничена, следует сосредоточиться на наиболее необходимых шаблонах. Во-вторых, тот факт, что шаблон напрямую не поддерживается продуктом, не означает, что вообще невозможно поддерживать конструкцию.

Из сравнения видно, что ни один инструмент не поддерживает все из 20 выбранных паттернов. Фактически, многие инструменты поддерживают только относительно небольшое подмножество более продвинутых шаблонов (т. е. Шаблонов с 6 по 20). В частности, ограниченная поддержка разоблачителя- криминалиста и его сообщение, -присоединиться, шаблоны, основанные на состоянии (только COSA), стоит отметить синхронизацию нескольких экземпляров (только FLOWer) и отмену действий /обращений .

Целью предоставления двух таблиц не является пропаганда использования конкретных инструментов. Однако они иллюстрируют, что существующие инструменты и языки действительно отличаются друг от друга и что большинство языков обеспечивают лишь частичную поддержку шаблонов, возникающих в реальных рабочих процессах. Эти наблюдения послужили нашей основной мотивацией для изучения выразительности высокоуровневых сетей Петри (раздел 3) и разработки нового языка (раздел 4).

3 Ограничения сетей Петри

Учитывая тот факт, что системы управления рабочими процессами сталкиваются с проблемами, связанными с рабочими схемами потоков, интересно посмотреть, могут ли устоявшиеся методы моделирования процессов, такие как сети Петри, справиться с этими схемами. Таблица, приведенная в приложении, показывает оценку высокоуровневых сетей Петри в отношении шаблонов. (Пока игнорируйте столбец в разделе YAWL.) Мы используем термин высокоуровневые сети Петри для обозначения сетей Петри , расширенных цветом (т.е. Данными), временем, и иерархия [4]. Примерами таких языков являются цветные сети Петри, описанные в [25], комбинация сетей Петри и спецификация Z, описанная в [20], и многие другие. Эти языки используются инструментами, такими как Design/ CPN (Орхусский университет, http://www.daimi.au.dk/designCPN/) и ExSpect (EUT / D & T Bakkenist, http://www.exspect.com/). Хотя эти языки и инструменты имеют различия, когда речь заходит, например, о языке для преобразования данных (например, дуговых надписей), существует четкий общий знаменатель. Когда мы ссылаемся на высокоуровневые сети Петри, мы ссылаемся на этот общий знаменатель. Чтобы избежать путаницы, мы максимально используем терминологию, определенную в [25]. Важно отметить, что для таблицы, приведенной в приложении, мы использовали те же критерии, что и в таблицах 1 и 2 для 15 систем документооборота (т. е. "+" ставится только при наличии прямой поддержки).

По сравнению с существующими языками высокоуровневые сети Петри довольно выразительны. Напомним, что мы используем термин "выразительность" не в формальном смысле. Высокоуровневые сети Петри являются полными по Тьюрингу и, следовательно, могут выполнять все, что мы можем определить в терминах алгоритма. Однако это не означает, что усилия по моделированию приемлемы. Сравнивая таблицу в приложении с таблицами 1 и 2, мы можем видеть, что сети высокого уровня, в отличие от многих языков документооборота, не имеют проблем с шаблонами, основанными на состоянии. Это прямое следствие того факта, что сети Петри используют places для явного представления состояний. Хотя высокоуровневые сети Петри превосходят большинство существующих языков, результат не является полностью удовлетворительным. Как указано во введении, мы видим серьезные ограничения, когда речь заходит о (1) шаблонах, включающих несколько экземпляров, (2) расширенных шаблонах синхронизации и (3) шаблонах отмены. В оставшейся части этого раздела мы обсудим эти ограничения более подробно.

3.1 Шаблоны, включающие несколько экземпляров

Предположим, что в контексте рабочего процесса обработки страховых требований существует подпроцесс обработки свидетельских показаний. Каждое страховое требование может включать ноль или более свидетельских показаний. Очевидно, что количество свидетельских показаний неизвестно во время разработки. Фактически, во время обработки свидетельских показаний могут появиться другие свидетели . Это означает, что в одном случае необходимо создать экземпляр части процесса переменное количество раз, и количество требуемых экземпляров известно только во время выполнения. Необходимым шаблоном для моделирования этой ситуации является шаблон 15 (несколько экземпляров без априорных знаний времени выполнения). Другим примером этого шаблона является процесс обработки отправлений журнала. Для обработки материалов журнала требуется несколько рецензий . Редактор журнала может принять решение пригласить различное количество рецензентов, в зависимости от характера статьи, например, если она вызывает споры, будет выбрано больше рецензентов . Во время рецензирования редактор может решить привлечь больше повторных зрителей. Например, если рецензенты не реагируют, имеют краткие или противоречивые отзывы.,

тогда редактор может назначить дополнительного рецензента. Другие примеры множественных случаев включают заказы, включающие несколько наименований товаров (например, клиент заказывает три книги в магазине электронных книг), процесс заключения субподряда с несколькими предложениями и т.д.

Можно смоделировать переменное количество экземпляров, выполняемых параллельно, используя высокоуровневую сеть Петри. Однако разработчик такой модели должен отслеживать две вещи: (1) идентификаторы обращений и (2) количество все еще запущенных экземпляров.

Одновременно обрабатывается несколько обращений. Предположим, и являются ли два активных случаи. Всякий раз, когда есть И-join, могут быть синхронизированы только токены, относящиеся к одному и тому же случаю. Если внутри создается экземпляр части процесса раз, тогда есть , то есть "дочерние случаи". Внутри части, которая создается несколько раз снова может быть параллелизм, и может быть несколько токенов, ссылающихся на один дочерний случай. Для обычного AND-join могут быть синхронизированы только токены, относящиеся к одному и тому же дочернему регистру . Однако в конце части, экземпляр которой создается несколько раз, все дочерние обращения, имеющие одного и того же родителя, должны быть синхронизированы, т.Е. case может продолжайте только в том случае, если для каждого дочернего элемента случае деталь была обработана. В этом дочерние случаи синхронизации и дочерние случаи должны быть четко обозначены разошлись. Чтобы усложнить ситуацию, конструкция из нескольких экземпляров может быть вложенной, в результате чего возникают дочерние случаи, такие как х.5.3, которые должны быть синхронизированы правильным способом. Очевидно, что хороший язык рабочего процесса не возлагает на разработчика рабочего процесса бремя отслеживания этих экземпляров и их синхронизации на нужном уровне.

Помимо отслеживания идентификационных данных и их синхронизации на нужном уровне, важно знать, сколько дочерних обращений необходимо синхронизировать. Это имеет особое значение, если количество примеров может меняться во время обработки (например, свидетель, указывающий на другого свидетеля, вызывает дополнительные свидетельские показания). В высокоуровневой сети Петри с этим можно справиться, введя счетчик, отслеживающий количество активных экземпляров. . Если активных экземпляров не осталось, дочерние обращения могут быть синхронизированы. Очевидно, что также неприемлемо возлагать бремя моделирования такого счетчика на разработчика рабочего процесса.

3.2 Расширенные схемы синхронизации

Рассмотрим рабочий процесс бронирования деловой поездки. Деловая поездка может включать в себя бронирование авиабилетов, отелей, прокат автомобилей и т.д. Предположим, что бронирование авиабилетов, отелей и автомобилей может происходить параллельно и что каждый из этих элементов является необязательным. Это означает, что одна поездка может включать только перелет, другая поездка может включать перелет и прокат автомобиля, и даже возможно размещение в отеле и прокат автомобиля (т. е. без перелета). Процесс бронирования каждого из этих элементов имеет отдельное описание , которое может быть довольно сложным. Где-то в процессе эти необязательные потоки должны быть синхронизированы, например, действия, связанные с оплатой, выполняются только после обработки всех элементов бронирования (т.е. рейса, отеля и автомобиля). Проблема в том, что не ясно, какие подпотоки необходимо синхронизировать. Для поездки, не связанной с перелетом, не следует ждать завершения бронирования рейса. Однако для деловой поездки, включающей все три элемента, все потоки должны быть синхронизированы. Ситуация, когда иногда нет синхронизации (XOR-join), иногда полная синхронизация (И-

объединение), а иногда требуется только частичная синхронизация (ИЛИ-объединение), называется шаблоном 7 (Синхронизирующее слияние).

Интересно отметить, что синхронизирующее слияние напрямую поддерживается InConcert, Eastman, Domino Workflow и MQSeries Workflow. В каждой из этих систем разработчику не нужно указывать тип соединения; это автоматически обрабатывается системой.

В высокоуровневой сети Петри каждая конструкция является либо соединением AND (переход), либо соединением XOR (место). Тем не менее, можно моделировать синхронизирующее слияние различными способами. Прежде всего, можно передавать информацию от узла разделения к узлу объединения. Например, если деловая поездка включает перелет и гостиницу, объединяющему узлу сообщается, что он должен синхронизировать потоки, соответствующие только этим двум элементам. Это можно сделать, поместив токен в место ввода перехода синхронизации, соответствующее элементу car rental. Во-вторых, можно активировать каждую ветвь, используя "логический" токен. Если значение токена равно true, выполняется все, что находится вдоль ветви. Если значение равно false, токен передается через ветку, но все действия над ним пропускаются. В-третьих, можно создать совершенно новый планировщик с точки зрения высокоуровневых сетей Петри. Этот планировщик интерпретирует рабочие процессы и использует следующее правило синхронизации: "Запускайте переход, если отмечено хотя бы одно из мест ввода " и исходя из текущей маркировки, невозможно разместить больше токенов ни в одном из других мест ввода " . В этом последнем решении проблема поднимается на другой уровень. Очевидно, что ни одно из трех решений не является удовлетворительным. Разработчик рабочего процесса должен добавить дополнительную логику в дизайн рабочего процесса (случай 1), должен расширить модель для размещения токенов true и false (случай 2) или должен смоделировать планировщик и поднять модель на другой уровень (случай 3).

Интересно посмотреть, как проблема синхронизирующего слияния решалась в существующих системах и литературе. В контексте рабочего процесса MQSeries используется технология "устранения тупиковых путей" [31, 22]..... [Это означает, что изначально каждая входная дуга находится в состоянии "без оценки". Пока одна из входных дуг находится в этом состоянии, действие не включено. Состояние входной дуги изменяется на true в момент выполнения предыдущего действия. Однако, чтобы избежать взаимоблокировок, для входной дуги устанавливается значение false в тот момент, когда становится ясно, что она не сработает. Распространяя эти ложные сигналы, взаимоблокировка невозможна, и результирующая семантика соответствует шаблону 7. Решение, используемое в рабочем процессе MQSeries , аналогично использованию токенов true и false (случай 2, описанный выше). Идея использования токенов true и false для решения сложных задач синхронизации уже поднималась в [18]. Однако схемы биполярной синхронизации, представленные в [18], в первую очередь направлены на то, чтобы избежать таких конструкций, как синхронизирующее слияние, т.е. Узлы являются чистыми И/XOR-разделяются / присоединяются, а частичная синхронизация не поддерживается и не исследуется стробируется. В контексте цепочек процессов, управляемых событиями (EPC, см. [26]), также возникает проблема решения проблемы синхронизирующего слияния. Модель EPC допускает это-

вызывается $\vee$ -соединителями (т.е. OR-соединениями, которые синхронизируют только активные потоки). Семантика этих $\vee$ -соединители часто обсуждались [3, 11, 29, 37, 38]. В [3] предлагается явное моделирование (пример 1). Денерт и Риттген [11] выступают за использование понятия слабой корректности (расслабленной обоснованности) и интеллектуального планировщика (пример 3). Ленгнер и др. [29] предлагают подход, основанный на булевых токенах (случай 2). Rump [38] предлагает интеллектуальный планировщик для принятия решения о том, следует ли синхронизировать an -connector

^{V} -соединитель: или нет (случай 3). В [37] для (1) предлагаются три разные семантики соединения. *подождите, пока придут все* (соответствует синхронизирующему слиянию, шаблон 7), (2) *подождите, пока придет первый, и игнорируйте остальные* (соответствует дискриминатору, шаблон 9), и (3) *никогда не ждите, выполняйте каждый раз* (соответствует множественному слиянию, шаблон 8). Обширная литература по проблемам синхронизации в EPC и системах документооборота иллюстрирует, что такие шаблоны, как синхронизирующее слияние, актуальны и далеки от тривиальных.

3.3 Шаблоны отмены

Большинство языков моделирования рабочих процессов, включая сети высокого уровня, имеют локальные правила, напрямую связывающие ввод действия с выводом. Для большинства ситуаций таких локальных правил достаточно. Однако для некоторых мероприятий местные правила могут быть довольно проблематичными. Рассмотрим, например, обработку таможенных деклараций. Пока таможенная декларация обрабатывается, лицо, подавшее декларацию, все еще может предоставить дополнительную информацию и уведомить таможню об изменениях (например, контейнер потерпел крушение, и, следовательно, будет меньше груза, указанного в первой декларации). Эти изменения могут привести к изъятию обращения из определенных частей процесса или даже из всего процесса в целом. Такие отмены не так просты, как кажутся, например, при использовании высокоуровневых сетей Петри. Причина в том, что изменение или дополнительное объявление может поступить в любое время (в течение заданного периода времени) и может повлиять на текущие и / или запланированные действия. Учитывая локальный характер переходов сети Петри, с такими изменениями трудно справиться. Если неизвестно, где в процессе находятся токены при повторном получении изменения или дополнительного объявления, удалить эти токены нетривиально. Дуги-ингибиторы позволяют проверять, содержит ли место маркер. Однако, для удаления to- kens из произвольного набора мест требуется довольно большая бухгалтерия. Рассмотрим, например, 10 параллельных ветвей по 10 мест в каждой. Чтобы удалить 10 токенов (по одному в каждой параллельной ветви), необходимо рассмотреть возможные состояния. Моделирование "пылесоса", то есть конструкции для удаления 10 токенов, возможно, но в результате получается диаграмма, похожая на спагетти. Поэтому трудно иметь дело с шаблонами отмены, такими как Отмена действия (шаблон 19) и Отмена обращения (шаблон 20) и всем, что находится между ними.

$1/2 1/4_{1/3 1/4}$

В этом разделе мы обсудили серьезные ограничения высокоуровневых сетей Петри, когда дело доходит до (1) шаблонов, включающих несколько экземпляров, (2) расширенных шаблонов синхронизации и (3) шаблонов отмены. Опять же, мы хотели бы подчеркнуть, что высокоуровневые сети Петри способны выражать такие схемы маршрутизации. Однако усилия по моделированию являются значительными, и хотя шаблоны требуются часто, бремя отслеживания всего возлагается на разработчика рабочего процесса.

4 YAWL: Еще один язык рабочего процесса

Совместными усилиями Технологического университета Эйндховена и Технологического университета Квинсленда в настоящее время мы работаем над новым языком документооборота, основанным на сетях Петри. Целью этих совместных усилий является преодоление ограничений, упомянутых в предыдущем разделе, путем добавления дополнительных конструкций. Подробное описание языка выходит за рамки данной статьи. Более того, язык все еще находится в стадии разработки.

Цель этого раздела - кратко описать особенности этого языка под названием *YAWL (Еще один язык рабочего процесса)*.

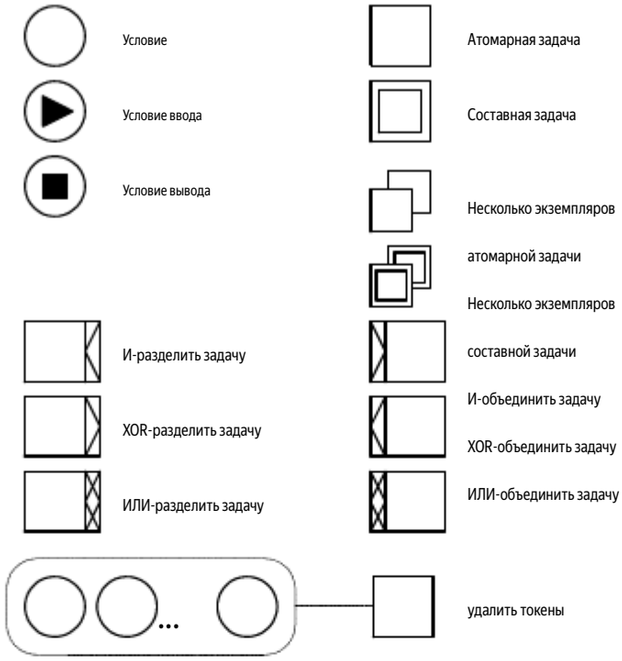


Рис. 2. Символы, используемые в YAWL.

На рисунке 2 показаны элементы моделирования YAWL. YAWL расширяет класс сетей рабочего потока, описанных в [2, 4], несколькими экземплярами, составными задачами, OR-объединениями, повторным перемещением токенов и напрямую связанными переходами. А *спецификация рабочего процесса* в YAWL - это набор *расширенных сетей рабочего процесса* (EWF-nets), которые образуют иерархию, т.Е. Существует древовидная структура. *Задачи* являются либо *(атомарными) задачами* или *составные задачи*.[1] Каждая составная задача относится к уникальной EWF-сети на более низком уровне иерархии. Атомарные задачи образуют листья древовидной структуры. Существует одна EWF-сеть без составной задачи, ссылающейся на нее. Эта EWF-сеть называется *рабочий процесс верхнего уровня* и формирует корень древовидной структуры.

Каждая EWF-сеть состоит из задач (составных или атомарных) и *условий* которые могут быть интерпретированы как места. Каждая EWF-сеть имеет одно уникальное *условие ввода* и одно уникальное *условие вывода* (см. Рисунок 2). В отличие от сетей Петри, можно соединять "объекты, подобные переходам", такие как составные и атомарные задачи, напрямую друг с другом без использования промежуточного "объекта, подобного месту" (т. Е. Условий). Для семантики эта конструкция может быть интерпретирована как скрытое условие, т. е. для каждого прямого соединения добавляется неявное условие.

[1] Обратите внимание, что в YAWL мы используем термин *задача* вместо того, чтобы *Активность* оставаться в соответствии с более ранней работой над сетями документооборота [2, 4].

Каждая задача (составная или атомарная) может иметь несколько экземпляров, как показано на рисунке 2. Можно указать нижнюю и верхнюю границы для количества экземпляров, созданных после запуска задачи. Более того, можно указать, что задача завершается в момент завершения определенного порогового значения экземпляров. В момент, когда этот порог достигнут, все запущенные экземпляры завершаются и задача завершается. Если пороговое значение не указано, задача завершается после завершения работы всех экземпляров. Наконец, есть четвертый параметр, указывающий, является ли количество экземпляров фиксированным после создания экземпляра. Значение параметра равно "фиксированному", если после создания никакие экземпляры не могут быть добавлены, и "переменному", если возможно добавить дополнительные экземпляры, пока есть все еще обрабатываемые экземпляры. Обратите внимание, что, расширяя сети Петри этой конструкцией с четырьмя параметрами (нижняя граница, верхняя граница, порог и фиксированный / var), мы напрямую поддерживаем все шаблоны, включающие несколько экземпляров (см. Раздел 3.1), и, кроме того, шаблон дискриминатора (шаблон 9) в предположении о нескольких экземплярах одной и той же задачи. На самом деле, мы также поддерживаем более общий подход -out-of- Ò

Мы принимаем обозначения, описанные в [2, 4] для AND/XOR-разбиений / объединений, как показано на рисунке 2. Более того, мы вводим OR-разбиения и OR-объединения, соответствующие соответственно шаблону 6 (множественный выбор) и шаблону 7 (Синхронизирующее слияние), см. Раздел 3.2.

присоединиться [6].

Наконец, мы вводим обозначение для удаления токенов из мест независимо от факта наличия и количества токенов. Как показано на рисунке 2, это обозначено пунктирными кругами / линиями. Включение задачи не зависит от токенов в пределах пунктирной области. Однако в момент выполнения задачи все токены в этой области удаляются. Очевидно, это расширение полезно для шаблонов отмены, см. Раздел 3.3. Независимо от этого, это расширение также было предложено в [10] с целью моделирования динамических рабочих процессов.

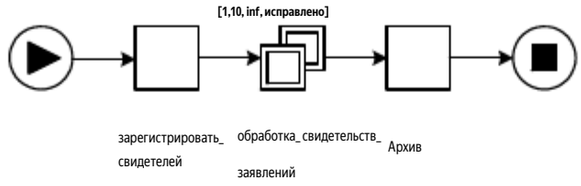
Как указывалось ранее, YAWL все еще находится в стадии разработки, и целью этого документа является не вводить язык в какие-либо подробности. Поэтому мы ограничиваемся простым применением YAWL к некоторым примерам, использованным в предыдущем разделе.

4.1 Пример: Шаблоны, включающие несколько экземпляров

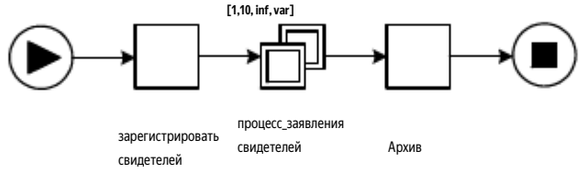
На рисунке 3 показаны три спецификации рабочего процесса, работающие с несколькими свидетельскими показаниями параллельно. Первая спецификация рабочего процесса (a) запускается между 1 и 10 экземплярами составной задачи *проверка процесса* после выполнения первоначальной задачи *зарегистрируйте ter witness*. Когда все экземпляры будут завершены, выполните задачу *Архив* выполняется. Вторая спецификация рабочего процесса, показанная на рисунке 3(b), запускает произвольное количество экземпляров составной задачи и даже допускает создание новых экземпляров. Третья спецификация рабочего процесса (c) запускается между 1 и 10 экземплярами составной задачи *заявление свидетеля процесса* но финиши должны быть завершены, если все они завершены или по крайней мере три завершены. Три примера иллюстрируют, что YAWL допускает прямую спецификацию шаблонов 14, 15 и 9.

Пример 4.2: Расширенные шаблоны синхронизации

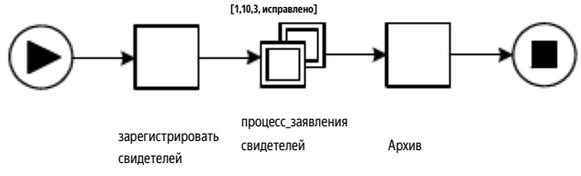
Как объясняется в разделе 3.2, OR-join можно интерпретировать по-разному. На рисунке 4 показаны три возможных толкования на примере бронирования деловой поездки. Первая спецификация рабочего процесса (a) начинается с ИЛИ-split *Зарегистрироваться* который позволяет выполнять задачи *полет*, *Гостиница* и/или *Автомобиль*. Задача *оплатить* выполняется каждый раз при выполнении одной из трех задач (т.е.,



(a) Рабочий процесс, обрабатывающий от 1 до 10 свидетельских показаний без возможности добавления свидетелей после регистрации (шаблон 14).

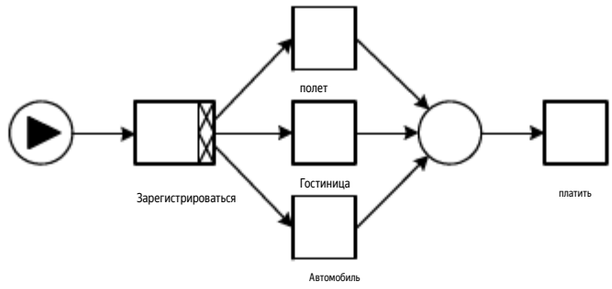


(b) Обработка рабочего процесса и произвольное количество свидетелей с возможностью добавления новых партий свидетелей (шаблон 15).

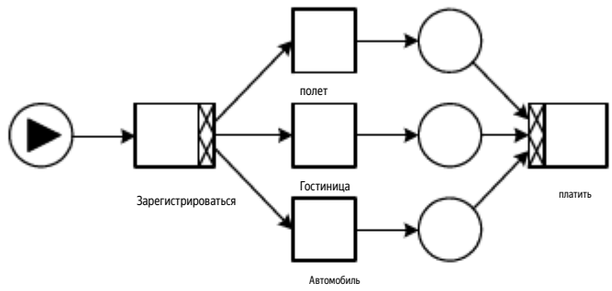


(c) Рабочий процесс, обрабатывающий от 1 до 10 свидетельских показаний с пороговым значением в 3 свидетеля (расширение шаблона 9).

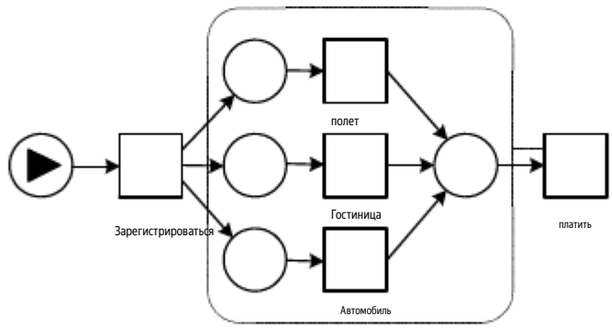
Рис. 3. Несколько примеров, иллюстрирующих, как YAWL работает с несколькими экземплярами.



(a) Оплата задания выполняется каждый раз, когда выполняется одно из трех предыдущих заданий (схема 8).



(b) Оплата задания выполняется только один раз, т.е. когда завершены все начатые задания (схема 7).



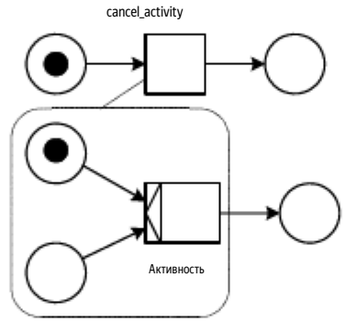
(c) Оплата задания выполняется только один раз, т.е. по завершении первого задания (шаблон 9).

Рис. 4. Несколько примеров, иллюстрирующих, как YAWL работает с расширенными шаблонами синхронизации.

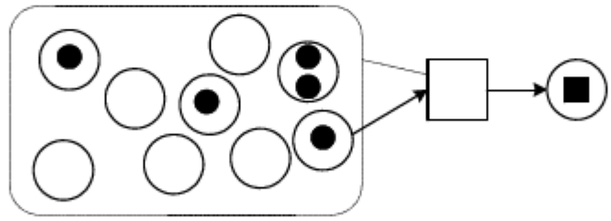
полет, *Гостиница*, и *Автомобиль*) завершается. Эта конструкция соответствует множественному слиянию (шаблон 8). Вторая спецификация рабочего процесса, показанная на рисунке 4 (b), аналогична, но объединяет отдельные платежи в один платеж. Следовательно, он ожидает, пока каждая из задач не будет включена с помощью *Зарегистрироваться* завершает. Обратите внимание, что если забронирован только рейс, синхронизация не выполняется. Однако, если поездка состоит из двух или даже трех элементов, задача *оплатить* откладывается до тех пор, пока все не будут завершены. Эта конструкция соответствует синхронизирующему слиянию (паттерн , страница 7). Третья спецификация рабочего процесса (c) позволяет выполнять все три задачи (т.е., *полет*, *Гостиница*, и *Автомобиль*), но оплачивается после завершения первой задачи. После оплаты все запущенные задачи отменяются. Хотя эта конструкция не имеет смысла в данном контексте, она была добавлена, чтобы проиллюстрировать, как можно поддерживать дискриминатор (шаблон 9), предполагая, что все запущенные потоки отменяются в момент завершения первого.

4.3 Пример: Шаблоны отмены

На рисунке 5 показано, как YAWL поддерживает два шаблона отмены (шаблоны 19 и 20). Первая спецификация рабочего процесса (a) показывает шаблон Отмены действия, который удаляет все токены из мест ввода задачи *Активность*. Во втором задании рабочего процесса-



(a) Отменить действие (шаблон 19).



(b) Отменить обращение (шаблон 20).

Рис. 5. Несколько примеров, иллюстрирующих, как YAWL работает с шаблонами отмены.

исправление (b) существует задача, удаляющая все токены и помещающая токен в условие вывода, таким образом реализуя шаблон отмены обращения. Примеры, приведенные в этом разделе, иллюстрируют, что YAWL решает многие проблемы, указанные в разделе 3. Таблица в приложении показывает, что YAWL поддерживает 19 из 20 шаблонов, используемых для оценки современных систем документооборота. Неявное завершение (т.е. Множественные условия вывода) не поддерживается, чтобы заставить разработчика думать о свойствах завершения рабочего процесса. Было бы довольно легко расширить YAWL с помощью этого шаблона (просто соедините все условия вывода с помощью OR-join, имеющего новое и уникальное условие вывода). Однако неявное завершение также скрывает ошибки проектирования, поскольку невозможно обнаружить взаимоблокировки. Следовательно, этот шаблон не поддерживается.

5 Заключение

Шаблоны документооборота, описанные в предыдущих публикациях [5-8, 48], обеспечивают функциональные требования к языкам документооборота. К сожалению, существующие языки документооборота только предлагают частичную поддержку этих шаблонов. По сравнению с языками рабочего процесса, используемыми коммерческими инструментами, сети Петри высокого уровня являются приемлемыми. Тем не менее, когда дело доходит до шаблонов, включающих несколько экземпляров, расширенную синхронизацию и отмену, высокоуровневые сети Петри предлагают небольшую поддержку. Поэтому мы работаем над созданием более выразительного языка на основе сети Петри, поддерживающего большинство шаблонов. Более того, мы надеемся, что проблемы моделирования, собранные в этой статье, подтолкнут других исследователей, работающих над сетями Петри высокого уровня, к разработке механизмов, инструментов и методов, обеспечивающих большую поддержку.

Ссылки

1. W.M.P. van der Aalst. Глава 10: Три веские причины для использования документооборота на основе сети Петри Система управления. Т. Вакаяма, С. Каннапан, К.М. Хун, С. Наватхе и Дж. Йейтс, редакторы, *Интеграция информации и процессов на предприятиях: переосмысление документов*, том 428 из *Международная серия Kluwer по инженерии и информатике*, страницы 161-182. Kluwer Academic Publishers, Бостон, Массачусетс, 1998. 2. В.М.П. ван дер Алст. Применение сетей Петри для управления рабочими процессами. *Журнал о схемах, системах и компьютерах.*, 8(1):21-66, 1998.
3. W.M.P. van der Aalst. Формализация и верификация технологических цепочек, управляемых событиями. *Информация-информационные и программные технологии*, 41(10):639-650, 1999.
4. У.М.П. ван дер Алст и К.М. ван Хи. *Управление рабочими процессами: модели, методы и Системы*. Издательство Массачусетского технологического института, Кембридж, Массачусетс, 2002.
5. В.М.П. ван дер Алст, А.Х.М. тер Хофстеде, Б. Кипушевски и А.П. Баррос. Дополнительно Шаблоны рабочего процесса. В книге О. Этциона и П. Шойерманна, редакторов, *7-я Международная конференция по совместным информационным системам (CoopIS 2000)*, том 1901 из *Конспектов лекций по информатике*, страницы 18-29. Springer-Verlag, Берлин, 2000.
6. В.М.П. ван дер Алст, А.Х.М. тер Хофстеде, Б. Кипушевски и А.П. Баррос. Рабочий процесс Шаблоны. Серия рабочих документов BETA, WP 47, Эйндховенский технологический университет, Эйндховен, 2000.

7. В.М.П. ван дер Алст, А.Х.М. тер Хофстеде, Б. Кипушевски и А.П. Баррос. Работа-

Схемы потоков. Технический отчет, Технологический университет Эйндховена,

Эйндховен, 2002. http://www.tm.tue.nl/it/research/patterns. 8. В.М.П.

ван дер Алст и Х. Рейерс. Adviseurs slaan bij workflow-systemen de plank regel-

matig mis. *Автоматизация Gids*, 36(15):15-15, 2002.

9. Афина Паллада. *Руководство пользователя Flower*. Pallas Athena BV, Апелдорн, Нидерланды, 2001. 10. П.

Хржастовски-Вахтель. Нисходящий подход на основе сети Петри к динамическому моделированию рабочего процесса-

разработка (продолжается). Университет Нового Южного Уэльса, Сидней, 2002.

11. Дж. Денерт и П. Ритген. Повышенная надежность бизнес-процессов. В книге К.Р. Диттриха,

А. Гепперт и М.К. Норри, редакторы, *Материалы 13-й Международной конференции по*

разработке передовых информационных систем (CAiSE'01), том 2068 из *Конспекты*

лекций по информатике, страницы 157-170. Springer-Verlag, Берлин, 2001. 12. К.А.

Эллис и Г.Дж. Натт. Моделирование и внедрение систем документооборота. В М. Аджмоне

Марсан, редактор, *Применение и теория сетей Петри 1993*, том 691 из *Конспекты*

лекций по информатике, страницы 1-16. Springer-Verlag, Berlin, 1993.

13. FileNet. *Руководство по визуальному дизайну WorkFlo*. FileNet Corporation,

Коста-Меса, Калифорния, США, 1997. 14. Forte. *Руководство по разработке процесса Forte Conductor*. Forte Software, Inc, Окленд, Калифорния,

США, 1998.

15. М. Фаулер. *Шаблоны анализа: объектные модели многократного использования*. Аддисон-Уэсли, Рединг, Массачусетс-

Сачусеттс, 1997.

16. Fujitsu. *Руководство для разработчиков i-Flow*. Fujitsu Software Corporation, Сан-Хосе, Калифорния, США, 1999.

17. Э. Гамма, Р. Хелм, Р. Джонсон и Дж. Влиссидес. *Шаблоны проектирования: элементы многократного использования*

Объектно-ориентированное программное обеспечение. Серия профессиональных

вычислений. Эддисон Уэсли, Рединг, Массачусетс, США, 1995. 18.

Х. Дж. Генрич и П. С. Тиагараджан. Теория схем биполярной синхронизации. *Theo-*

сетевая информатика, 30(3):241-318, 1984.

19. Д. Георгакопулос, М. Хорник и А. Шет. Обзор управления рабочими процессами.:

От моделирования процессов к инфраструктуре автоматизации документооборота. *Распределенные*

и параллельные базы данных, 3: 119-153, 1995. 20. К.М.

ван Хи. *Разработка информационных систем: формальный подход*. Кембриджский университет

Издательство, 1994.

21. HP. *Руководство по проектированию процессов HP Changengine*. Компания Hewlett-Packard, Пало-Альто, Калифорния,

США, 2000.

22. IBM. *Рабочий процесс IBM MQSeries - Начало работы со временем сборки*. IBM Deutschland En-

twicklung GmbH, Boeblingen, Germany, 1999.

23. С. Яблонски и К. Бусслер. *Управление рабочими процессами: концепции моделирования, архитектура и*

Реализация. International Thomson Computer Press, Лондон, Великобритания, 1996.

24. К. Дженсен. Цветные сети Петри.: Язык высокого уровня для системного проектирования и анализа. В

Г. Розенберг, редактор, *Достижения в области сетей Петри, 1990*, том 483 из *Конспекты*

лекций по информатике Наука, страницы 342-416. Springer-Verlag, Berlin, 1990. 25. K. Jensen.

Цветные сетки Петри. Основные концепции, методы анализа и практическое использование.

Монографии EATCS по теоретической информатике. Springer-Verlag, Берлин, 1992.

26. Г. Келлер, М. Наттгенс и А.В. Шеер. Semantische Processmodellierung auf der

Grundlage Ereignisgesteuerter Processketten (EPK). Вероф fentlichungen des Instituts fur

Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbruck en, 1992.

27. В. Kiepuszewski. *Выразительность и пригодность языков для моделирования потоков управления*

в рабочих процессах. Докторская диссертация, Квинслендский технологический университет, Брисбен, Австралия,

2002. 28. Т.М. Кулопулос. *Императив рабочего процесса*. Ван Нострандт Рейнхольд, Нью-Йорк, 1995. 29.

П. Ленгнер, К. Шнайдер и Дж. Велер. Сертификация процессов, управляемых событиями, на основе сети Петри

Цепочки. В J. Desel и M. Silva, редакторы, *Применение и теория сетей Петри, 1998*, том

1420 из *Конспекты лекций по информатике*, страницы 286-305. Springer-Verlag, Berlin, 1998.

30. П. Лоуренс, редактор. *Справочник по рабочим процессам 1997, Коалиция по управлению рабочими процессами*. Джон Уайли и сыновья, Нью-Йорк, 1997.

31. Ф. Лейманн и Д. Роллер. *Производственный процесс: концепции и методы*. Прентис-Холл PTR, Аппер-Седл-Ривер, Нью-Джерси, США, 1999.

32. М. Аджмоне Марсан, Г. Бальбо и Г. Конте. Класс обобщенных стохастических сетей Петри для оценки производительности многопроцессорных систем. *Транзакции ACM в компьютерных системах*, 2(2): 93-122, май 1984. 33. М. Аджмоне Марсан, Г. Бальбо, Г. Конте и др. *Моделирование с помощью обобщенного стохастика Сети Петри*. Ряды Уайли в параллельных вычислениях. Уайли, Нью-Йорк, 1995.

34. С.П. Нильсен, К. Истхоуп, П. Госселинк, К. Гутце и Дж. Роэл. *Использование Lotus Domino Workflow 2.0, Redbook SG24-5963-00*. IBM, Покипси, США, 2000. 35. К.А. Петри. *Kommunikation mit Automaten* Докторская диссертация, Институт мека instrumentelle Mathematik, Bonn, 1962.

36. Д. Риле и Х. Зуллигхо преп. Понимание и использование шаблонов при разработке программного обеспечения. *Теория и практика объектных систем*, 2(1):3-13, 1996.

37. П. Ритген. Модифицированные EPC и их формальная семантика. Технический отчет 99/19, Университет Кобленц-Ландау, Кобленц, Германия, 1999.

38. Ф. Рамп. Erreichbarkeitsgraphbasierte Analyse ereignisgesteuerter Prozessketten. Технический-cher Bericht, Institut OFFIS, 04/97 (in German), University of Oldenburg, Oldenburg, 1997.

39. SAP. *Бизнес-процесс WF SAP*. SAP AG, Вальдорф, Германия, 1997. 40. Т. Шаль. *Управление рабочими процессами для организации процессов*, том 1096 из *Конспекты лекций по информатике*. Springer-Verlag, Berlin, 1996.

41. А. Шет, К. Кочут и Дж. Миллер. Крупномасштабные распределенные информационные системы (LSDIS) лаборатория, страница проекта METEOR. http://lsdis.cs.uga.edu/proj/meteor/meteor.html.

42. Программное обеспечение Eastman. *Руководство пользователя инструмента RouteBuilder*. Eastman Software, Inc, Биллерика, Массачусетс, США, 1998.

43. Набор программного обеспечения. *Руководство пользователя COSA 3.0*. Software-Ley GmbH, Pullheim, Germany, 1999.

44. Принадлежности для персонала. *Руководство пользователя Staffware 2000 / GWD*. Staffware plc, Беркшир, Великобритания, 2000.

45. Tibco. *Руководство пользователя TIB/InConcert Process Designer*. Tibco Software Inc., Пало-Альто, Калифорния, США, 2000.

46. Verve. *Концепции движка документооборота компонентов Verve*. Verve, Inc., Сан-Франциско, Калифорния, США, 2000.

47. WFMC. Терминология и глоссарий Коалиции по управлению рабочими процессами (WFMC-TC-1011). Технический отчет, Коалиция по управлению рабочими процессами, Брюссель, 1996.

48. Домашняя страница шаблонов рабочих процессов. http://www.tm.tue.nl/it/research/patterns .

A Сравнение высокоуровневых сетей Петри и YAWL с использованием шаблонов

Таблица, приведенная в этом приложении, указывает для каждого шаблона, обеспечивают ли высокоуровневые сети Петри / YAWL прямую поддержку (обозначается "+"), частичную прямую поддержку (обозначается "+/-") или никакой прямой поддержки (обозначается "-").

шаблон	сети Петри высокого уровня	YAWL
1 (seq)	+	+
2 (par-spl)	+	+
3 (synch)	+	+
4 (ex-ch) 5	+	+
(simple-m) 6	+	+
(m-choice)	+	+
7 (синхронизация-m)	-μ	+
8 (мульти-m)	+	+
9 (диск)	-μ	+
10 (arb-c)	+	+
11 (impl-t)	- μ	- μ
12 (mi-no-s)	+	+
13 (mi-dt)	+	+
14 (mi-rt) 15	-μ	+
(mi-no) 16	-) μ	+
(def-c) 17 (int-par)	+	+
18 (самый	+	+
короткий)	+	+
19 (can-a)	-• μ	+
20 (can-c)	-) μ	+

(i) Синхронизирующее слияние не поддерживается, потому что дизайнер должен отслеживать количества параллельных потоков и принять решение о слиянии или синхронизации потоков (см. Раздел 3.2).

(ii) Дискриминатор не поддерживается, поскольку разработчику необходимо отслеживать количество запущенных потоков и количество завершенных потоков и должно быть сброшено конструкция явно путем удаления всех токенов, соответствующих итерации (см. Раздел 3.2).

(iii) Неявное завершение не поддерживается, поскольку разработчик должен отслеживать запущенные потоки, чтобы решить, завершено ли обращение.

(iv) Неявное завершение не поддерживается, поскольку разработчик вынужден идентифицировать один уникальный конечный узел. Любая модель с несколькими конечными узлами может быть преобразована в сеть с уникальным конечным узлом (просто используйте синхронизирующее объединение). Это не было добавлено в YAWL, чтобы заставить разработчика задуматься об успешном завершении кейса. Это требование позволяет обнаруживать неудачное завершение (например, взаимоблокировки).

(v) Несколько экземпляров с синхронизацией не поддерживаются высокоуровневыми сетями Петри (см. Раздел 3.1).

(vi) Также не поддерживается, см. Раздел 3.1.

(vii) Действие отмены поддерживается лишь частично, поскольку токены можно удалить из место ввода перехода, но требуется дополнительная бухгалтерия, если имеется несколько мест ввода, и эти места могут быть пустыми (см. раздел 3.3).

(viii) Отменить действие не поддерживается, потому что нужно смоделировать очиститель вакуума, чтобы удалить токены, которые могут или не могут находиться в определенных местах (см. раздел 3.3).