УДК 519.712.1

ПРЕДИКАТНАЯ МОДЕЛЬ ГИБКОГО ПРОЦЕССА

Рудометкина М.Н.

ФГАОУ ВО «Национальный исследовательский Томский политехнический университет», Томск, e-mail: mn.rud@inbox.ru

В статье разработана предикатная модель гибкого (многовариантного) процесса на основе дерева процессов, которая обеспечивает возможность формирования процесса методами process mining из набора логов, а также адаптации процесса к предметной области. Адаптация процессной модели к предметной области достигается за счет применения известных операторов адаптации базовых элементов модели. Модель позволяет выстроить вертикальную иерархию «горизонтального» процесса, в виде иерархии предикатов. Каждый подпроцесс формализует алгоритм действий в заданной классификационными признаками ситуации. Для бизнес-процессов можно выделить подпроцесс, относящийся к заданному подразделению организации.

Ключевые слова: логическая сеть, алгебра предикатов, гибкий процесс, лог процесса, process mining

PREDICATE FLEXIBLE PROCESS MODEL

Rudometkina M.N.

National Research Tomsk Polytechnic University, Tomsk, e-mail: mn.rud@inbox.ru

The paper presents the predicate flexible process model based on a process tree. It provides the possibility of process formation using the process mining technique based on the process execution recorded in event logs and its adaptation to the data domain. The process model adaptation to the data domain is achieved by application of known adaptation operators to core elements of the model. Thus, it allows constructing a 'horizontal' process hierarchy in the form of a predicate hierarchy. Each subprocess formalizes a step-by-step procedure for a situation determined by classification criteria. Thus, in business processes, it is possible to highlight a subprocess relating to the given organizational subdivision. In software development process, it will be a subprocess relating to the development of the given software product subsystem.

Keywords: logical network, predicate algebra, flexible process, the log process, process mining

В настоящее время развивающейся областью научных исследований является область интеллектуального анализа процессов. В данной сфере разрабатываются методы построения моделей дискретных процессов на основе анализа последовательностей событий, отражающих их выполнение и представленных в виде файлов логов. При этом предполагается наличие лишь приблизительного, часто неформализованного (вербального) описания процесса, который уже реализуется.

Сложность построения адекватной модели процесса на основе анализа логов связана с необходимостью отобразить все возможные варианты поведения процесса на основе исследования следов его выполнения. Особенно актуальной указанная проблема адекватности модели является для гибких процессов, отражающих множество версий выполнения и адаптируемых к предметной области отсечением избыточной логики, что и определяет значимость данной работы.

1. Анализ литературы

Работа базируется на идее построения модели гибкого процесса путем использования методов process mining с применением алгебры конечных предикатов (АКП) и логических сетей [1]. Алгебра конечных предикатов предназначена для формализации процессов логической природы и представляет собой дискретный аппарат для описания произвольных конечных отношений. Решение уравнений АКП осуществляются посредством логической сети, для чего уравнения алгебры преобразуются в бинарную систему. Ключевое преимущество логических сетей заключается в определении полного набора свойств исследуемого объекта на основе ограниченного входного набора признаков, что обеспечивает возможность эффективного построения модели гибкого процесса на основе анализа логов событий. В данном случае набор логов содержит ограниченный набор признаков, а полученная модель формализует свойства гибкого многовариантного процесса [2].

В работе [2] был предложен способ построения и настройки модели гибкого процесса и детализированы особенности этапа адаптации модели. В соответствии с рассмотренным в работе [2] подходом при моделировании процессов как последовательности действий могут быть решены две основных задачи – построения модели процесса (1) и его адаптации (2), т.е. улучшения с учетом особенностей предметной области и внешней среды.

Модель гибкого процесса в общем виде представлена системой бинарных предикатов $M = \{R_j \mid j = \overline{1,n}\}$. Эти предикаты задают логику поведения процесса и допустимые взаимосвязи между действиями процесса. Под адаптацией модели под предметную область P понимается отбор только тех

предикатов, которые позволяют выполнять действия процесса с учетом взаимосвязей между объектами $P = \{\mathrm{P}_k \mid \mathrm{k} = \overline{1, K}\}$ предметной области. Тогда адаптированная модель процесса принимает вид:

$$M^P = \{R_j \mid \forall R_j \exists P_k, R_j \in \mathrm{M}, \mathrm{P}_k \in P, \mathrm{i}, \mathrm{j} = \overline{1, n}\}. \quad (1)$$

Существующие методы построения модели процесса основаны на следующих базовых подходах: объединение моделей различных вариантов реализации процесса, каждая из которых получена традиционными методами process mining; простое объединение логов для различных вариантов реализации и построение методами process mining единой модели гибкого процесса; дополнение отличающимися в отдельных реализациях возможностями базовой модели, полученной методами process mining и отражающей типовое, характерное для всех реализаций выполнение процесса [3, 4].

Однако указанные методы ориентированы в первую очередь на построения модели традиционного процесса с жестко заданной последовательностью действий. При формализации гибкого процесса адаптируемой структурой необходимо учитывать избыточность логики его поведения, что и определяет актуальность построения общей модели такого процесса.

2. Формулировка задачи

Задача построения модели гибкого процесса требует учета базовых элементов, отражающих его поведение, различных уровней детализации модели, а также многовариантности его реализации.

Исходными данными для реализации подхода являются логи событий, отражающие последовательность выполнения действий при реализации процесса. На основе слияния и анализа логов с помощью логических сетей выполняется построение полной многовариантной модели процесса [2].

В целом такая модель должна состоять из предикатов, задающих допустимые взаимосвязи между действиями процесса.

При адаптации модели из полной системы предикатов осуществляется отбор лишь тех предикатов, которые учитывают взаимосвязи в заданной предметной области.

Проведенная в работе [2] структуризация позволяет выделить дополнительное, но важное требование к гибкой процессной модели: необходимость совмещения в ее представлении как собственно модели процесса, так и возможностей ее адаптации.

Таким образом, при решении задачи построения модели необходимо исследовать структуру лога, особенности слияния логов как исходных данных для построения модели, выделить базовые элементы логики процесса и объединить их в единую обобщенную модель, обладающую возможностью адаптации путем отсечения избыточной логики процесса.

3. Структура лог-файла

В настоящее время многие информационные системы фиксируют свою деятельность с помощью так называемых файлов логов. В указанных файлах с привязкой ко времени фиксируются выполняемые задачи, изменения в данных, изменения статуса системы и ее отдельных компонентов и т.д.

Например, при доступе в интернет может быть зафиксирован IP-адрес пользователя, время доступа, имя пользователя, а также адрес ресурса, к которому пользователь обращался.

Файлы логов, как видно из приведенных примеров, содержат информацию о выполнении соответствующих процессов (доступа к информации, бухгалтерского учета и т.п.). Очевидно, что при построении модели процесса, отражающей алгоритм его действий, средствами process mining часть информации является избыточной и может не учитываться. Это, впрочем, не исключает использования «избыточных» данных при построении иных аспектов модели процесса – организационного, информационного, управляющего. Пример фрагмента лога, отражающего операции при регистрации документов приведен в табл. 1.

Таблица 1

Пример фрагмента лога

Код ситуации	Код события	Временная метка	Действие	Ресурс
1	354647	02.06.2014 10.01	Запрос на регистрацию документов	Иванов И.И.
1	354648	02.06.2014 10.06	Проверка данных	Петрова О.В.
1	354649	02.06.2014 11.12	Проверка исходных документов	Сенченко Р.И.
1	354650	01.05.2014 11.18	Принятие решения	Ивахненко С.И.
1	354651	02.06.2014 14.24	Выдача зарегистрированных документов	Петрова О.В.
2	354671	03.06.2014 11.08	Запрос на регистрацию	Смирнов И.И.

Основные элементы данной таблицы: код ситуации; код события; наименование действия; временная метка. Код ситуации позволяет сгруппировать последовательность событий процесса, соответствующих его однократному выполнению. Иными словами, последовательность действий с одним кодом ситуации представляет собой «след» или траекторию однократного выполнения процесса. Код события однозначно идентифицирует факт выполнения действия, название которого отражено в колонке «действие». Очевидно, что при многократном выполнении одного и того же действия коды соответствующих событий будут различаться. Временная метка фиксирует дату и время выполнения действия. В колонке «ресурс» в данном случае представлены данные исполнителя действия.

Таблица лога может содержать и дополнительные параметры, например информацию о документах, о стоимости операций процесса и т.п. Помимо табличной структуризации, а практике лог может быть представлен в стандартизованном формате OpenXES. Данный xml – подобный формат предполагает, что лог хранится в файле с расширением «xes».

Основные теги, обеспечивающие структуризацию лога:

– trace – траектория выполнения (следа) процесса;

– event – событие, отражающее выполнение действия процесса.

При описании каждого события согласно рассматриваемому стандарту указываются временные метки, наименование действия и ресурсы, как было показано при рассмотрении приведенного в табл. 1 примера.

В соответствии с поставленной в работе задачей основное внимание уделим последовательности (фактически – алгоритму) действий процесса. Мы будем рассматривать лог процесса как набор следов (траекторий однократного выполнения) процесса, причем в общем случае набор таких траекторий может быть пустым или неполным:

$$l_k = \{S_i\}, S_i \subseteq S, i = \overline{1,I} \qquad (2)$$

где S – полный набор всех возможных вариантов выполнения процесса, для которого получен лог l_i. В свою очередь, каждый след S_i объединяет упорядоченный набор событий $s_{i,j}$

$$S_i = < s_{i,1}, s_{i,2}, ..., s_{i,j}, ..., s_{i,J} > . \qquad (3)$$

Каждый элемент $s_{i,j}$ следа процесса отражает однократное выполнение одного из его действий.

При построении гибкого процесса мы объединяем логи, полученные при реализации одного и того же процесса в рамках различных информационных систем. Поэтому исходный для нашей задачи лог гибкого процесса объединяет k традиционных логов и имеет следующий вид:

$$L = \{l_k\} = \{\{S_i = < s_{i,1}, s_{i,2}, ..., s_{i,j}, ..., s_{i,J} >\}\}, k = \overline{1,K}, i = \overline{1,I} \qquad (4)$$

Данный подход позволяет единым образом объединять как «следы» процесса, входящие в один лог, так и различные логи с набором «следов» процесса.

Отметим, что необходимым условием слияния логов при предлагаемом подходе является соответствие меток (имен) операций в логах, отражающих различные варианты реализации процесса. Однако проверка такого соответствия, а также преобразование имен является инженерной задачей.

Объединяя все вышесказанное, будем рассматривать лог гибкого процесса как набор событий, отражающих все существующие «следы» реализации процесса и потому в идеальном случае отражающий все возможные траектории его выполнения. Объединение всех существующих «следов» процесса означает, что могут объединяться логи различных информационных систем. Форматы представления данных при этом могут быть различными. Однако задача преобразования форматов файлов логов к единому виду выходит за рамки данной работы.

4. Предикатная древовидная модель процесса

Разработанная предикатная модель основана на дереве процессов [3]. Древовидное представление в работе усовершенствовано с учетом предикатной формализации, что позволяет представить дерево процессов в виде иерархии предикатов и, в конечном итоге, в виде логической сети.

Дерево процессов представляет собой направленный ациклический граф G, содержащий вершины двух типов [5]:

$$G = (V, E), V = \{V^*, V^{**}\}, \qquad (5)$$

где E – дуги дерева процессов.

Обозначим переменными x_1, x_2,…, x_n события $v_i \in V, i = \overline{1,n}$. Эти переменные заданы на некотором конечном множестве возможных значений событий. Например,

$x_1 \in \{a, b, c\}$, где $x_1 = a$ означает «событие v_1 не выполнено», $x_1 = b$ – «событие v_1 приостановлено», $x_1 = c$ – «событие v_1 выполнено».

Дуги дерева процессов $e_i \in E, i = \overline{1,m}$, описывающие попарные связи (если они существуют) между событиями, будем обозначать предикатами $R_i, i = \overline{1,m}$.

Вершины первого типа $x_i \in V^*$ отражают конкретные действия (процедуры) процесса. С каждой вершиной связано определенное действие процесса с помощью бинарных предикатов R_k:

$$\forall x_i \in V^* \exists! x_j, R_k(x_i, x_j), i, j = \overline{1,n}, k = \overline{1,m}. \quad (6)$$

Данные функции реализуются с помощью системы бинарных предикатов (логической сети):

$$\begin{cases} R_1, \\ R_2, \\ ..., \\ R_k. \end{cases} \quad (7)$$

Соответственно, модель (1) дополнится следующим образом:

$$M^P = \{R_k \mid \forall R_k(x_i, x_j) \in M, x_i \in V^* \; i, j = \overline{1,n} \quad \exists P_k \in P, k = \overline{1,m}\} \quad (8)$$

В случае если из вершины x' графа G выходит две дуги, то такая вершина называется вершиной второго типа $x' \in V^{**}$. Для формализации вариантов порядка взаимодействия пар процедур для каждой вершины $x' \in V^{**}$ вводится одна из базовых предикатных операций (операций над предикатами):

– последовательное выполнение

$$\rightarrow (R_k(x', x_i), R_l(x', x_j), r), \quad (9)$$

где r – номер процедуры, которая выполняется первой;

– выбор

$$\mathrm{XOR}(R_k(x', x_i), R_l(x', x_j)) = R_k(x', x_i) \oplus R_l(x', x_j)$$

или

$$\mathrm{OR}(R_k(x', x_i), R_l(x', x_j)) = R_k(x', x_i) \vee R_l(x', x_j); \quad (10)$$

– параллельное выполнение

$$\mathrm{AND}(R_k(x', x_i), R_l(x', x_j)) = R_k(x', x_i) \wedge R_l(x', x_j); \quad (11)$$

– цикл

$$\Omega(R_k(x', x_i), R_l(x', x_j), t). \quad (12)$$

где t – число повторений в цикле или условие повторения.

Таблицы истинности введенных предикатов второго порядка представлены ниже.

Для удобства введем следующее обозначение. Предикатные операции типа (9)-(12) будем обозначать буквой S с числовым индексом, указывающим порядковый номер каждой операции в модели. Число предикатных операций будет определяться свойствами конкретной модели. Для модели, использующей операции (9)-(12), имеем:

$$S_i \in S, S = \{\rightarrow, \mathrm{XOR}, \mathrm{OR}, \mathrm{AND}, \Omega\}.$$

Таблица 2

Таблица истинности для предикатов XOR, OR, AND

	XOR	OR	AND
$R_k = 0, R_l = 0$	0	0	0
$R_k = 0, R_l = 1$	1	1	0
$R_k = 1, R_l = 0$	1	1	0
$R_k = 1, R_l = 1$	0	1	1

Таблица 3

Таблицы истинности для предикатов →, Ω

	→
$R_k = 0, R_l = 0, r = 1 \vee 2$	0
$R_k = 0, R_l = 0, r = 1$	0
$R_k = 0, R_l = 0, r = 2$	1
$R_k = 1, R_l = 0, r = 1$	1
$R_k = 1, R_l = 0, r = 2$	0
$R_k = 1, R_l = 1, r = 1 \vee 2$	1

	Ω
$R_k = 0, R_l = 0, r = t \in N$	0
$R_k = 0, R_l = 1, r = t \in N$	0
$R_k = 1, R_l = 0, r = t \in N$	0
$R_k = 1, R_l = 1, t = 0$	0
$R_k = 1, R_l = 1, t > 0$	1

Тогда модель процесса представляется в виде иерархии предикатных операций вида (9)-(12) на верхних уровнях, а также бинарных предикатов на нижнем уровне:

$$M = \langle S_k, R_i \rangle,$$

где

$$R_k(x_i, x_j) \in M, k = \overline{1,m}, x_i \in (V^* \cup V^{**}), i, j = \overline{1,n},$$
$$S_l(R_i, R_j) \in S, \; l = \overline{1,p}. \qquad (13)$$

Полученная предикатная модель процессного дерева обладает следующими преимуществами:

Во-первых, данная модель позволяет выстроить вертикальную иерархию «горизонтального» процесса, в виде иерархии предикатов, что позволяет выделять фрагменты процесса по заданным классификационным признакам и в дальнейшем рассматривать их как подпроцессы основного процесса. Каждый подпроцесс в данном случае формализует алгоритм действий в заданной классификационными признаками ситуации.

Например, для бизнес-процессов можно выделить подпроцесс, относящийся к заданному подразделению организации. Для процессов разработки программного обеспечения – подпроцесс, относящийся к разработке заданной подсистемы программного продукта, для процессов общения в социальных сетях – подпроцессы взаимодействия заданной группы пользователей. Аналогично, при обработке языковых конструкций построение текста можно рассматривать как процесс, а отдельных предложений – как подпроцессы.

В перспективе предлагаемое предикатное представление дает возможность преодолеть противоречие между построением процесса как сквозным алгоритмом действий и поддерживающей его вертикальной структурой (организационной, технологической и т.п.).

Во-вторых, предлагаемая модель обладает всеми достоинствами традиционного процессного дерева и, следовательно, обеспечивает построение корректной в смысле достижения конечного состояния модели процесса, как было показано в работе [5].

В-третьих, процессное дерево включает в себя формализацию всех базовых элементов процесса [6].

В-четвертых, данное представление обеспечивает построение полной модели процесса и ее дальнейшую адаптацию, поскольку мы формализуем взаимосвязи между отдельными фрагментами, включая уровни подчиненности.

В-пятых, иерархическое представление дает возможность рассматривать процесс с различной степенью детализации. Такая возможность особенно важна при построении гибких процессов, поскольку позволяет выделить уровни (подпроцессы), требующие адаптации к предметной области.

Более того, можно связать иерархию реализуемых процессом функций с последовательностью его действий.

Заключение

В статье предложена предикатная модель гибкого процесса в виде иерархии предикатных операций на верхних уровнях, а также бинарных предикатов на нижнем уровне, которая позволяет выстроить вертикальную иерархию «горизонтального» процесса, обеспечивая различную

степень детализации, а также дает возможность построить полную модель процесса методами process mining на основе слияния логов.

Ключевая особенность модели – наличие двух типов вершин, задаваемых бинарными предикатами, определяющими последовательность действий процесса в предметной области, а также базовыми предикатными операциями, отражающими собственно действия процесса. Такая структура обеспечивает статическую и динамическую адаптацию модели процесса.

Список литературы

1. Бондаренко М.Ф., Шабанов-Кушнаренко Ю.П. Мозгоподобные структуры: Справочное пособие. Том первый. Под редакцией акад. НАН Украины И.В. Сергиенко. – К.: Наукова думка, 2011. – 460 с.

2. Рудометкина М.Н. Формирование и адаптация модели гибкого процесса на основе анализа логов. 2014. – № 4: Динамика сложных систем XXI-век, Москва. С. 90.

3. F. Gottschalk, W.M.P. van der Aalst, and M.H. Jansen-Vullers. Mining Reference Process Models and their Configurations. OTM 2008 Workshops, volume 5333 of Lecture Notes in Computer Science, p. 263–272, Berlin Heidelberg, 2008. Springer Verlag.

4. J.C.A.M. Buijs, M. La Rosa, H.A. Reijers, B.F. Dongen, and W.M.P. van der Aalst. Improving Business Process Models using Observed Behavior. In Proceedings of the Second Intern. Symposium on Data-Driven Process Discovery and Analysis, LNBIP. Springer, 2013.

5. W.M.P. van der Aalst, J. Buijs, and B.F. van Dongen. Towards Improving the Representational Bias of Process Mining. In K. Aberer, E. Damiani, and T. Dillon, editors, IFIP International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA 2011), volume 116 of Lecture Notes in Business Information Processing, p. 39–54. Springer-Verlag, Berlin, 2012.

6. J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst. On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery. In R. Meersman, S. Rinderle, P. Dadam, and X. Zhou, editors, OTM Federated Conferences, 20th International Conference on Cooperative Information Systems (CoopIS 2012), volume 7565 of Lecture Notes in Computer Science, p. 305–322. Springer-Verlag, Berlin, 2012.