Совмещение программного кода и HTML-разметки «в одном флаконе» позволяет написать научную статью с традиционными описаниями, формулами, таблицами, рисунками, содержащую в себе готовую программу обработки данных, и опубликовать её в Интернете в виде одного текстового файла. Особенно привлекательна такая возможность, когда речь идет об учебных научных материалах.

Исследование выполнено при финансовой поддержке РФФИ в рамках научных проектов № 13-08-97063-р_поволжье_а, 13-08-97050-р_поволжье_а, 13-08-00359-а, 13-08-00504-а.

Литература

[1] *Реймонд, Э.С.*, Искусство программирования для Unix, М.: Издательский дом «Вильямс», 2005.– 544 с.

[2] *Занько, Ф.С.*, Комментарии Python в стиле HTML, `http://www.russianlutheran.org/python/literate/literate.html`

Миронов Андрей Михайлович, Михеев Андрей Геннадьевич, Пятецкий Валерий Ефимович
Москва, МГУ, Консалтинговая группа РУНА, НИТУ «МИСиС»

Проект: RunaWFE http://wf.runa.ru/rus

Реализация алгоритма проверки ограниченности количества точек управления в свободной системе управления бизнес-процессами и административными регламентами RunaWFE

Аннотация

В современных системах управления бизнес-процессами всвязи с ошибками проектирования бизнес-процессов могут возникать ситуации неограниченного возрастания количества точек управления в экземпляре бизнес-процесса, что приводит к неоправданно большой нагрузке . В докладе рассмотрена задача анализа схем бизнес-процессов, представлен алгоритм, проверяющий ограниченность количества точек управления, реализованный в системе RunaWFE, доказана теорема о корректности алгоритма

Проект RunaWFE

Проект RunaWFE развивает свободную, ориентированную на конечного пользователя систему управления бизнес-процессами и административными регламентами. Основная задача системы: раздавать задания исполнителям и контролировать их исполнение. Последовательность заданий определяется графом бизнес-процесса, который менеджер или бизнес-аналитик может быстро изменять при помощи редактора бизнес-процессов.

Проект RunaWFE предлагает использовать свободное ПО как средство кооперации генераторов идей и разработчиков промышленного ПО. В этом случае генераторы идей, передав свои идеи в проект, получают свободный инструмент, реализующий их идеи. Разработчики промышленного ПО, в свою очередь, получают идеи и теории, которые позволят разрабатываемому ПО получить качественные преимущества.

В настоящем докладе рассмотрен пример такого сотрудничества: решение сложной математической задачи и применение результатов в системе RunaWFE.

Проблема «взрывного» роста количества точек управления в экземпляре бизнес-процесса

В 4.0 версии RunaWFE в соответствии со стандартом BPMN 2.0 элемент «слияние» работает следующим образом: для каждого входящего перехода пришедшая в «слияние» точка управления ставится в очередь. Если для всех входящих в «слияние» переходов их очереди заполнены хотя бы одной точкой управления, то все точки управления, находящиеся на первой позиции очереди каждого перехода, уничтожаются и на выходе из «слияния» генерируется одна точка управления. Все остальные точки, находящиеся в очередях в «слияние», перемещаются на одну позицию вперед.

При отсутствии ограничений на комбинации элементов на схеме бизнес-процесса в случае такого определения поведения элемента «слияние» возможно появление экземпляров бизнес-процессов, в которых из-за ошибок проектирования количество точек управления бесконечно возрастает с течением времени. Такие процессы могут создать запредельную нагрузку на систему, что приведет к прекращению ее нормальной работы.

На рисунке 1 показан пример бизнес-процесса с бесконечно возрастающим количеством точек управления.

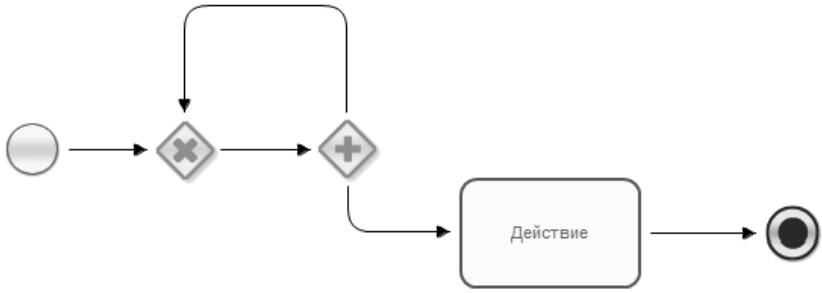

Рис. 1: Бизнес-процесс с бесконечно возрастающим количеством точек управления

Описание алгоритма

1. Нумерация узла «Начало» и всех переходов схемы

Посчитаем количество линий (стрелок) на схеме бизнес-процесса. Обозначим это количество — n. Поставим в соответствие узлу «Начало» номер 1. Всем переходам (стрелкам) схемы поставим в соответствие числа от 2 до n+1 в произвольном порядке.

2. Создание двух списков векторов и проверочного графа векторов

Создадим два списка векторов размерности n+1. Первая позиция каждого вектора соответствует узлу-началу, остальные позиции — переходу в соответствии с тем, как мы занумеровали переходы.

В первом списке (обозначим его V) будут содержаться вектора возможных изменений состояний. Все компоненты его могут принимать значения только из множества {-1, 0, 1}. Этот список будет один раз заполнен, и далее не будет изменяться.

Второй список называется «список векторов необработанных состояний». Все его компоненты могут содержать только целые неотрицательные значения. Значение в каждой компоненте обозначает коли-

чество точек управления на соответствующем переходе схемы. Этот список будет динамически изменяться во время работы алгоритма.

Проверочный граф представляет собой структуру из узлов, соответствующих векторам и направленных переходов (т.е. каждый переход направлен от одного узла-вектора к другому). Узлы являются векторами состояний (размерности n+1). Все компоненты каждого вектора могут содержать только целые неотрицательные значения. Значение в каждой компоненте обозначает количество точек управления на соответствующем переходе схемы бизнес-процесса. Проверочный граф тоже будет динамически изменяться во время работы алгоритма.

3. Заполнение списка V

А. Рассмотрим стартовый узел. Для каждого выходящего из стартового узла перехода добавим в список V вектор, в котором в позиции стартового узла (то есть, в первой позиции) находится «-1», в позиции выходящего перехода — «+1», а все остальные элементы — нули.

Б. Переберем в цикле все узлы «Параллельный шлюз» схемы бизнес-процесса. Для каждого элемента добавим в список V вектор, в котором для каждого входящего в узел перехода находится «-1», для каждого выходящего перехода — «+1», а все остальные элементы — нули.

В. Переберем в цикле все остальные узлы схемы бизнес-процесса. Для каждого узла, для каждой пары возможных комбинаций входящих и исходящих из этого узла переходов (входящий в узел переход, исходящий из узла переход) добавим в список V вектор, в котором в позиции входящего перехода находится «-1», в позиции выходящего перехода — «+1», а все остальные элементы — нули.

Далее вектора из списка **V** будем обозначать — **v(j)**.

4. Работа со списком векторов необработанных состояний и проверочным графом (основной этап работы алгоритма)

А. Поместим в проверочный граф вектор (1, 0, 0, ... , 0).

(В первой позиции, соответствующей узлу-Началу — одна точка управления, в остальных позициях — нули)

Б. Поместим в список «список векторов необработанных состояний» вектор (1, 0, 0, ... , 0).

В. Будем выполнять следующую итерационную процедуру до момента ее окончания:

Рассмотрим первый вектор из списка векторов необработанных состояний. Назовем этот вектор текущим вектором **u**.

Создадим для текущего вектора набор промежуточных векторов следующим образом:

Будем последовательно перебирать все вектора **v(j)** списка **V**:

К текущему вектору **u** покомпонентно прибавим вектор **v(j)**. Если ни в одной позиции полученного вектора не будет отрицательных чисел, то добавим этот вектор в набор промежуточных векторов.

Найдем в наборе промежуточных векторов все вектора, которые уже есть в проверочном графе. Для этих векторов создадим переходы из текущего вектора **u** в уже существующие вектора проверочного графа, совпадающие с промежуточными векторами, а эти промежуточные вектора удалим из списка промежуточных векторов.

Соединим текущий вектор **u** с оставшимися промежуточными векторами переходами и соответственно, удалим все оставшиеся промежуточные вектора из списка промежуточных векторов, поместим их в проверочный граф, а также добавим их в конец списка векторов необработанных состояний.

Удалим текущий вектор **u** из списка векторов необработанных состояний.

Рассмотрим список векторов необработанных состояний. Для каждого вектора из этого списка:

Построим множество векторов, из которых достижим данный узел (алгоритм построения этого множества описан ниже). Проверим, есть ли среди векторов, из которых достижим данный вектор, хотя бы один вектор, котором в каком-то компоненте число строго меньше числа из соответствующего компонента данного вектора, а в остальных — меньше, или равно.

Если хотя бы один такой вектор есть, то **алгоритм останавливается**, в редакторе процессов возникает сообщение — результат работы алгоритма: «**в бизнес-процессе может возникнуть ситуация с бесконечно возрастающим количеством точек управления**».

Далее итерационная процедура повторяется.

Если в какой-то момент времени список векторов необработанных состояний оказался пустым, то **алгоритм останавливается**, в редакторе процессов возникает сообщение — результат работы алгоритма: «**в бизнес-процессе не может возникнуть ситуация с бесконечно возрастающим количеством точек управления**».

5. Построение множества векторов, из которых достижим данный узел

Создадим два пустых список, которые назовем «список векторов» и «буфер»

Рассмотрим текущий вектор. Поместим в «буфер» все вектора, стрелки (переходы) из которых ведут в текущий вектор.

Далее проведем следующую итерационную процедуру:

Для всех векторов буфера ищем все возможные вектора, для которых существуют стрелки (переходы) из которых ведут хотя бы в один вектор из буфера. Удаляем из найденных векторов все вектора, содержащиеся в «списке векторов», или в буфере.

Переносим вектора из буфера в список векторов, а оставшиеся найденные вектора в буфер.

Если после этого буфер оказывается пустым, то прекращаем итерации

Повторяем итерацию.

Полученный «список векторов» и будет множеством векторов, из которых достижим данный узел.

В докладе будет представлено доказательство корректности работы данного алгоритма.

Результаты применения.

Для проверки бизнес-процессов на возможность бесконечного возрастания количества точек управления в экземпляре процесса, в редакторе процессов реализован алгоритм проверки.

В меню «Файл» добавляется команда «Проверить точки управления», которая активна, если в данный момент выбран бизнес-процесс в BPMN нотации, он сохранен и в нем нет ошибок (См. рисунок 2).