XXXII Неделя науки СПбГПУ. Материалы межвузовской научно-технической конференции. Ч.V : С.175-176

УДК 681.3

В.Р.Чурилов (6 курс, каф. РВиКС), А.Ю.Глебовский, к.ф.-м.н., доц.

РАЗРАБОТКА АЛГОРИТМОВ ОБНАРУЖЕНИЯ ЗАДАННЫХ ФРАГМЕНТОВ ПОВЕДЕНИЯ В СОБЫТИЙНЫХ СИСТЕМАХ.

Многие информационные системы можно рассматривать как событийные, то есть реагирующие на наступающие события и инициирующие новые события. Событийные системы можно подразделить на два класса: системы мониторинга и управляющие системы. При мониторинге осуществляется пассивное наблюдение за поведением некоторого внешнего объекта. Приложениями этого типа являются, например, мониторинг сетей и системы обнаружения вторжений (Intrusion Detection System – IDS). Управляющие системы не только отслеживают, но и формируют поведение объекта, например, на основе правил вида “событие–условие–реакция” (Event Condition Action – ECA). Примерами таких приложений являются продукционные системы, системы управления бизнес-процессами, системы дистанционного обучения. Программную систему, реализующую ту или иную событийную модель, будем называть системой обработки событий, сокращенно СОС (Event Processing System – EPS).

Целями данной работы являются разработка архитектуры СОС и отыскание алгоритмических решений задачи обнаружения фрагментов поведения СОС. Необходимость разработки обусловлена отсутствием известного адекватного метода решения поставленной задачи.

Алгоритмы, разработанные для ECA-систем, такие так RETE [Forgy82], TREAT [Miranker87], Gator [Hanson93], LEAP и другие не адекватны решаемой задаче, так как они оперируют более абстрактными, чем события понятиями (фактами), вследствие чего эти алгоритмы не поддерживают наиболее важные операторы корреляции событий (операторы следствия и независимости). В IDS, напротив, проблема обнаружения фрагментов поведения сводится к более узкой задаче обнаружения потенциально опасного поведения в сетевых системах. Поэтому применение IDS-решений ограничено спецификой этой области. Для поставленной задачи наиболее близкими представляются концепции, развиваемые в проекте Complex event processing [CEP].

Следуя CEP, будем под событием в СОС понимать структуру вида:

Событие:= (Вектор_Причинности, Тип_События, Экземпляр_События, Атрибуты)

Вектор причинности содержит указатели на экземпляры тех событий, которые являются непосредственной причиной возникновения данного события.

Паттерн – это выражение, построенное из элементарных событий посредством применения операторов корреляции. Каждый паттерн имеет тип и экземпляр. Тип паттерна – это абстрактный фрагмент поведения событийной модели системы, определяемый в терминах типов событий. Экземпляр паттерна – это фрагмент поведения системы, наблюдаемый в процессе ее функционирования и определяемый в терминах экземпляров событий.

В качестве операторов корреляции событий используются операторы, подобные применяемым в CEP: следствие (A—>B), независимость (A || B), дизъюнкция (A | B), конъюнкция (A & B), отрицание (!A), темпоральный (A>B, A<B). Предлагаемые операторы следствия и независимости имеют иную, более богатую семантику, нежели в CEP, и обладают рядом дополнительных параметров, в том числе временными параметрами, а также ограничениями на поток событий между событиями-операндами.

Для построения СОС предлагается модификация архитектуры CEP, содержащая ряд перечисленных ниже ключевых компонентов.

1. Набор адаптеров. Для каждой подсистемы необходим адаптер, который будет трансформировать внутренние события этой подсистемы в событийный поток (то есть в каноническую форму событий, понимаемую остальными компонентами СОС).
2. Событийная модель системы. Определение событийной модели состоит из сценариев возникновения событий в конкретном приложении. Сценарий констатирует объективные свойства СОС и задается посредством паттернов. Например, паттерн вида (*A*—3[*B*]—>*C*—[!*D*]—>*F*) констатирует, что после события *A* возможно возникновение события *C* не более чем 3 раза. В промежутке между событиями *A* и *C* допускается появление события *B*. Далее, после каждого события *C* возможно возникновение события *F*, но не *D*. События верхнего уровня определяются в терминах событий более низкого уровня при помощи тех же операторов корреляции.
3. Механизм обнаружения экземпляров паттернов. Он осуществляет функцию эффективного распознавания паттернов во время выполнения во входном событийном потоке.
4. Программные компоненты, обрабатывающие события. Компонент имеет порты приема и пересылки событий, а также внутреннюю логику поведения, описываемую правилами вида “паттерн — реакция”.
5. Спецификация, связывающая компоненты между собой через соответствующие порты.

В некоторых случаях требуются паттерны, составленные из вложенных операторов следствия. Такая ситуация возникает при разработке событийной модели иерархической системы, где будут созданы иерархии событий, то есть события более высокого уровня будут определяться в терминах событий более низкого уровня посредством паттернов.

Так, например, если события 1-го уровня определяются как

E11 = (A—>B—>C) & (D—>F), E12 = (K—>M) & (E—>G—>H),

а события 2-го уровня как E21 = E11 —> E12,

то, значит, E21 задается паттерном с вложенным оператором следствия

E21 = ((A—>B—>C) & (D—>F)) —> ((K—>M) & (E—>G—>H)).

Вложенный оператор следствия наиболее сложен в реализации, так как оперирует на частичных порядках, формирующихся динамически. Частично упорядоченный набор событий удовлетворяет (вложенному) оператору следствия, если выполняются два следующих условия: (1) каждое событие из правой части будет вызвано (непосредственно или транзитивно) хотя бы одним событием из левой части, (2) каждое событие из левой части вызовет (непосредственно или транзитивно) хотя бы одно событие из правой части.

Основными результатами работы на данном этапе являются: разработанный формализм, язык определения паттернов, модель СОС, а также эффективный алгоритм обнаружения сложных паттернов произвольного уровня вложенности во входном потоке событий, обладающий модулярностью, позволяющей строить распределенные СОС.

ЛИТЕРАТУРА:

1. C. Forgy, "RTE: A fast algorithm for the many patterns / many objects match problem", 1982.
2. D.P. Miranker, "TREAT: A Better Match Algorithm for AI Production Systems", 1987.
3. E.N. Hanson, "Gator: A discrimination network suitable for optimizing production rule matching", 1993.
4. Complex event processing (CEP) project. Program Analysis and Verification Group, Computer Science Department and Electrical Engineering Department of Stanford University.