PEACE-ful Web Event Extraction and Processing*

Tim Furche[1], Giovanni Grasso[1], Michael Huemer[2],
Christian Schallhart[1], and Michael Schrefl[2]

[1] Department of Computer Science, Oxford University,
Wolfson Building, Parks Road, Oxford OX1 3QD
firstname.lastname@cs.ox.ac.uk
[2] Department of Business Informatics – Data & Knowledge Engineering,
Johannes Kepler University, Altenberger Str. 69, Linz, Austria
lastname@dke.uni-linz.ac.at

Abstract. PEACE, our proposed tool, integrates complex event processing and web extraction into a unified framework to handle web event advertisements and to run a notification service atop. Its bitemporal schemata distinguish occurrence and detection time, enabling PEACE to deal with updates and delayed announcements, as often occurring on the web. To consolidate the arising event streams, PEACE combines simple events into complex ones. Depending on their occurrence and detection time, these complex events trigger actions to be executed.
We demonstrate PEACE's capabilities with a business trip scenario, involving as raw events business trips, flight bookings, scheduled flights, and flight arrivals and departures. These events are scrapped from the web and combined into complex events, triggering actions to be executed, such as updating facebook status messages. Our demonstrator records and reruns event sequences at different speeds to show the system dealing with complex scenarios spanning several days.

1 Introduction

If an event is published today, it is published on the web. And if so, the announcement is likely to change over time, as more precise and up-to-date information becomes available. Checking such continuously changing event streams is tedious and stressful, especially, if more than one event source is involved, requiring coordination among individual events. Thus, one would assume that event notification has been addressed already: On the one hand, there is a clear need for web event processing, as our life style is more dynamic and interactive than ever before. On the other hand, all necessary information is readily available, most people wear an Internet enabled mobile device at all times, and hence, one would suppose that events need only to be extracted, checked for some application specific criteria, and delivered to the user.

But until now, no integrated solution exists for this task, except for isolated solutions dedicated to specific application domains. We argue that a good solution for web

* The research leading to these results has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement DIADEM, no. 246858. Michael Huemer has been supported by a Marietta Blau Scholarship granted by the Austrian Federal Ministry of Science and Research (BMWF) for a research stay at Oxford University's Department of Computer Science.

```
doc("http://www.flightarrivals.com")
  //a#panel0/{click /}//form#qbaForm/descendant::field()[1]/{$airport }
  /following::field()[3]//option{select }/following::field()[1]/{click /}
   /(/descendant::a[string(.)='Next >'][1]/{click /})*
      //table#flifo//tr[position()>1]/self():<FlightArrival>
        [./td[1]:<fromLoc=string(.)>]        [./td[2]:<flightNo=string(.)>]
        [./td[3]/div:<flightDay=string(.)>]  [.:<toLoc=$airport>]
        [./td[3]/text()[1]:<occTime=toUnixTime(.)>]
```

Fig. 1: OXPATH Wrapper

event processing needs to integrate complex event processing and web extraction, dealing with frequent changes and retractions. We demonstrate PEACE *(Processing Event Ads into Complex Events)* as a possible solution. PEACE takes an event model to determine available event types and attributes, and based thereupon, **(1)** wrappers to extract events from multiple sources, **(2)** complex event queries to aggregate the raw events into complex events, and **(3)** action executors to perform the resulting actions.

Contrasting previous work, PEACE is fully *bitemporal,* distinguishing detection and occurrence time, i.e., the time when events are extracted and supposedly take place. PEACE allows for different reactions, depending on the detection and occurrence time in relation to the current time. While complex events [3] have been studied extensively, existing systems like [1, 6] deal only with some aspects of the bitemporality. Most systems, as [2, 4], drop this distinction in identifying occurrence and detection time.

2 Extracting and Processing Events from the Web with PEACE

Once an event model with its event classes and attributes is fixed, PEACE takes wrappers, complex event specifications and action executors to extract, process and react upon events. **(1)** The wrappers for event extraction are given in OXPATH, an XPATH-based language for highly scalable web data extraction [5]. Aside extracting web events, we can also integrate other sources for events, such as a local database. **(2)** The complex events are specified in BICEPL, our SQL-based language to define complex events as `SELECT` statements, extended with constructs for expressing constraints over occurrence and detection time. BICEPL specifications also include publication events, issued when complex events are introduced, updated, or revoked. **(3)** Finally, these publication events trigger OXPATH action executors, e.g., notifying users or changing a booking.

In the following, we give an example for each those three component types. These examples are directly taken from the demonstration. **Event Extractors.** The wrapper in Figure 1 fetches its target page (Line 1), selects options to obtain all current flight arrivals at a parameterized airport, and submits the form (Lines 2-3). The wrapper deals with paginated results by repeatedly clicking on "next" (Line 4). On each result page, `FlightArrival` objects are extracted altogether with their attributes (Lines 5–7) for every entry on the page. Matching the event model for the scenario, each `FlightArrival` results in an event tuple. **Complex Event Specifications.** In Figure 2 we show the complex event specification for `ArrivedAtDestination` indicating when a business person ar-

```
1 CREATE COMPLEX EVENT CLASS ArrivedAtDestination (flightNo TEXT, flightDay
      TEXT, location TEXT) ID (flightNo, flightDay)
2   AS SELECT fa.flightNo, fa.flightDay, fa.toLoc
3      FROM BusinessTrip bt, FlightBooking fb, Flight f, FlightArrival fa
4      WHERE bt.tripTitle = fb.tripTitle AND fb.bookingId = f.bookingId AND
5            f.connFlNo = 'NULL' AND f.connFlDay = 'NULL' AND
6            f.flightNo = fa.flightNo AND f.flightDay = fa.flightDay
7      OCCURRING AT fa
8   PUBLISH ArrivedAtDestinationOnTime
9      CASE LATE (0s, 1h) ArrivedAtDestinationLate
10     CASE RETROACTIVECHANGE ArrivedAtDestinationChanged
11     CASE REVOKE ArrivedAtDestinationRevoked;
```

Fig. 2: Event Definition for Complex Event ArrivedAtDestination.

rives at the final trip destination. `ArrivedAtDestination` has explicit attributes `flightNo`, `flightDay` and `location`, along with implicit timing information, using `flightNo` and `flightDay` as key (Line 1). In BICEPL a complex event is defined with an extended SQL `SELECT` statement involving constituent events. In our example these constituent events are `BusinessTrip`, `FlightBooking`, `Flight` and `FlightArrival` (Line 3). These events are joined and filtered (Lines 4-6) and defined to occur when the constituent `FlightArrival` event occurs (Line 7). Further, complex event descriptions contain event publication statements to control the event published in case the complex event occurs on time, at maximum one hour late, has retroactively changed, or was revoked (Lines 8-11). **Action Executors.** For each publication event produced, PEACE runs action executors, such as a parameterized OXPATH expressions filling a form to send an SMS.

3 PEACE-ful Business Trip Management

Our demonstration scenario deals with business travelers and their arrangements. To keep up with tight schedules, business travelers observe and quickly react upon many events, e.g., upon learning about flight delays, they need to check for connecting flights, or change hotel reservations. This can be time consuming and stressful, but having PEACE, all this hassle can be dealt with automatically! Aside managing the daily disasters of business travel, PEACE can be even configured to keep friends and family posted about an ongoing trip by updating facebook status messages.

Event Extractors. We implement our scenario with 5 event types from 3 different sources, partly extracted from the web via OXPATH wrappers. **(1)** `BusinessTrip` events are extracted from Google Calendar, providing a unique reference for each trip. **(2)** `FlightBooking` and `Flight` events are stored locally and describe a booking with all connecting flights of a one-way trip. **(3)** `FlightArrival` and `FlightDeparture` events for all relevant flights are extracted from `flightarrivals.com`. We show the wrapper for `FlightArrivals` in Figure 1. **Complex Event Specifications.** Atop these subscribed raw events, we specify 5 complex event types in BICEPL. **(A)** `E1DaysToBusinessTrip` indicates that a trip begins in 1 day; **(B)** `E3DaysToFlightDeparture` that a departure is going to take place in 3 days; **(C)** `MissedConnectingFlights` that a connection is unreachable

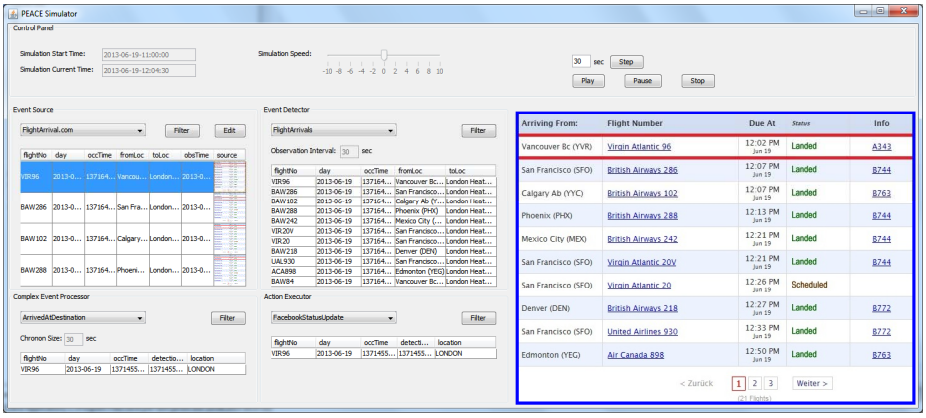

Fig. 3: PEACE Simulator.

due to a canceled or delayed flight. **(D)** `UncatchableConnection` captures connection flights which are missed. **(E)** `ArrivedAtDestination` indicates that a business traveler supposedly arrived at the destination of the trip. This specification is depicted in Figure 2. **Action Executors.** Complex events trigger various actions, for example upon an `E1DaysToBusinessTrip` event, the system notifies the business traveler via email, and upon an `ArrivedAtDestination` event, the system updates the traveler's facebook status. In case of `E3DaysToFlightDeparture`, the referred flight data is checked to be correct and still valid. Moreover, action executors *control* the event extractors currently deployed: For example, a `E3DaysToFlightDeparture` event triggers the setup of event extractors for `FlightArrivals` and `FlightDepartures` for the flight under scrutiny.

Simulation and Visualization. We showcase our scenario with the PEACE simulator and visualizer, illustrated in Figure 3. Presenting interesting cases, we not only run the system live but also on recorded event streams over a couple of days. The simulator runs the entire system at different speeds, skimming over uneventful phases and carefully analyzing more turbulent phases. With the simulator, we trace events, occurring on event sources, then being extracted, leading to complex events, and thus taken actions.

References

1. Adi, A., Etzion, O.: Amit - the situation manager. VLDB J. **13** (2004)
2. Cugola, G., Margara, A.: TESLA: a formally defined event specification language. In: DEBS. (2010)
3. Cugola, G., Margara, A.: Processing flows of information: From data stream to complex event processing. ACM Comput. Surv. **44** (2012)
4. Eckert, M., Bry, F.: Rule-based composite event queries: the language XChange^{EQ} and its semantics. Knowl. Inf. Syst. **25** (2010)
5. Furche, T., Gottlob, G., Grasso, G., Schallhart, C., Sellers, A.J.: OXPath: A language for scalable data extraction, automation, and crawling on the deep web. VLDB J. **22** (2013)
6. Luckham, D.: Event Processing for Business. John Wiley & Sons, Inc. (2012)