Event Webs for Crisis Management

Caltech Infospheres Group
K. Mani Chandy,
Brian Aydemir, Elliott Karpilovsky,
Daniel M. Zimmerman
Background and Motivation

- Crises happen…
  - Natural disasters (earthquakes, fires, blizzards)
  - Unnatural disasters (terrorist attacks, train derailments)
  - Market events (large price fluctuations, corporate scandals)

- Task forces made up of people from multiple institutions:
  - Government agencies
  - Corporations
  - Other organizations (Red Cross, etc.)

- All these people need different information at different times to do their jobs.
Outline

- Event Web Architecture Overview
- Execution Model for Software Components
- Application to Crisis Management
Event Web

- Architecture to help task forces effectively manage crises by making information available when and where needed
- Network of reactive objects that process event streams
- Definitions:
  - Event - a message describing an aspect of a system’s state or history
  - Event Stream - a sequence of events describing the evolution of a particular aspect of a system’s state over time
Event Web Components

- **Event Processors** - persistent reactive objects that process event streams
- **Dissemination Network** - communication mechanism that distributes events among the event processors
- **Directory** - lookup service that describes both available event streams and requests for event streams
- **Service Layer** - modular system that provides various services to event processors
Event Processors

- Objects in the event web that may receive, process, and generate event streams
- Described by XML documents in a persistent store
- *Event Generator* - generates event streams
  - Monitors its environment and *publishes* events (as XML)
  - Could have physical sensors, passively receive information (email, SMS messages), or actively poll information sources
- *Event Consumer* - receives event streams
  - Performs actions based on the events it receives (executing applications, notifying specific people)
- Can be both at the same time
Dissemination Network

- Sends a copy of each event to each interested event consumer
- Consumers *subscribe* for events with particular characteristics by specifying criteria as predicates
  - Subscriptions specified using an XML query language
- Essentially a content-based publish/subscribe system
Dissemination Network - Formal Model

- Set of named input queues associated with each event consumer
- Set of subscriptions (predicates on contents of events) for each queue
- One event channel connects each event generator to each input queue
- Events take arbitrary finite time to arrive at destination queues
- Arriving events inserted in queue if they match a subscription, otherwise discarded
Directory

- Enables individuals and event processors to advertise characteristics of event streams they need or provide
- New event processors can combine event streams in the directory to generate new event streams
- Individuals can use the directory to determine where new sensors or event processors are needed

Therefore...

- Set of event streams in the event web becomes better suited to the needs of users over time
Service Layer

- Provides various services to event processors
- Currently, only two services available:
  - Creation of new event processors
  - Handling of timeout events
- Modular, so new services can be created and added as necessary
Event Processor Execution Model

- State of an event processor specified by values of its variables, including priorities assigned to its input queues and a timeout value
- Each event processor has:
  - state transition function
  - event generation function
  - service request function
Event Processor Execution Model

- Event processors interact only through events.
- Next event to be processed always comes from the highest priority input queue with an available event.
- Timeout value used by the service layer to send a timeout event if no other events arrive.
Event Processor Execution Model

- Events trigger transitions
- State transition, event generation, and service request functions determine new state, outgoing events, and outgoing service requests
- Transitions atomic with respect to other event processors
Event Processor Implementation

- Event processors live in persistent storage (as XML documents) unless actually executing a transition
- State transition, event generation, and service request functions implemented using XSLT or Java
- Persistent storage managed by special processes

- A single runtime system can support a much larger number of event processors than could fit in memory simultaneously
Application to Crisis Management

- Specify conditions on the history of a system and actions to be taken when those conditions hold.

- A condition and its corresponding action form a *when-then rule*:
  - *when* the reactor core temperature has increased more than 20°C in the last 15 minutes, *then* alert the chief engineer.
  - *when* the units of type O blood used in Los Angeles County over the last 7 days exceed 3/4 of the units available in Los Angeles County over the last 7 days, *then* alert health services.
Application to Crisis Management

- Event web actually implements an approximation to *when-then* rules
  - Taking an accurate global picture of the state and history of a distributed system is difficult
- We believe this to be adequate for most crisis management applications
Conclusions

- Event web architecture allows individuals in a crisis management task force to rapidly customize an information system to suit their needs.

- *When-then* rules allow task force members to easily specify conditions of interest and actions to take when they occur.

- Event web is made independent of specific hardware platforms and programming environments by using open standards such as XML and XSLT.
Questions?