

A Simple Infrastructure for Aggregating and Visualizing Events in Distributed Cougaar Systems

Nathan Combs, Jeff Vagle
BBN Technologies

Key Phrases

- *Composing “service architecture” models from distributed workflow events*
- *Aggregating distributed events with a logging infrastructure*
- *Cougaar Agent Infrastructure*
- *Log4J (Apache)*
- *DASADA Runtime Infrastructure*

Objective

To explore this question: can we use an off-the-shelf mechanism for collecting publish/subscribe (and other) workflow events from a distributed infrastructure (Cougaar)? Can we then use these events to reconstruct an external model of the distributed workflow for visualization?

Introduction

Described here is one approach for collecting “Service and Contract” workflow events (including publish/subscribe) for visualization and analysis. Described is the use of a commercial hyperbolic visualization tool that was used. In 2002, distributed workflow events will be used to generate Architecture Description Language (ADL) representations for architectural analysis and verification.

Long-term we plan to use DASADA infrastructure (DARPA funded research) as the “event bus”, however, for the near term until alternatives mature, we choose to adapt a simple, Open Source distributed logging tool with surprising flexibilities.

Infrastructure Basis

We decided to piggy-back off of the existing logging infrastructure used by the BBN-DASADA Cougaar framework for simplicity. BBN-DASADA uses Apache’s Log4J to “log” workflow events and route a subset (filtered) to central logging server. For this experiment we performed a simple customization of this infrastructure to essentially create “event channels” to send relevant log events to a visualization tool. We started with an initial hypothesis.

Hypothesis

Can we co-opt Apache's Log4J to flexibly manage multiple log streams (filter and categorization capability) for our application event-routing purposes? Can we also co-opt Log4J's capability for collecting and aggregating logging data in a distributed environment?

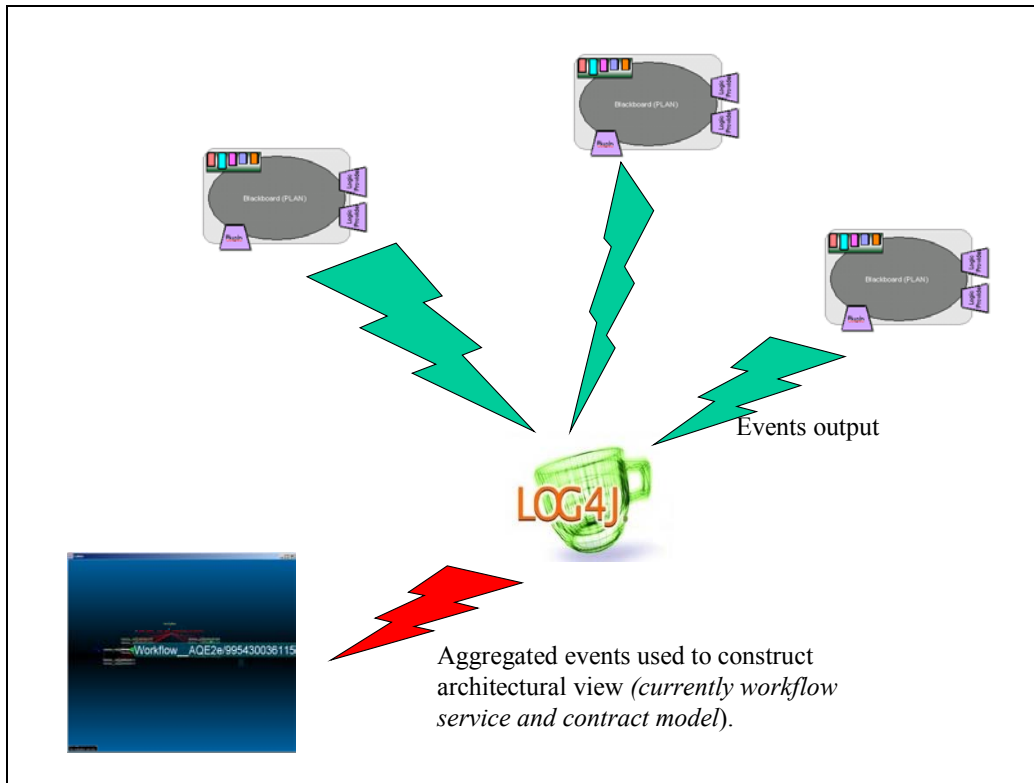


Figure 1 Current use-case: Service and Contract events logged, appropriate subset routed to Log4J server. Workflow representation of service architecture constructed and visualized.

Two Types of Events Logged to Server

Within the BBN-DASADA service architecture a significant number of logging events are captured for diagnostics and debugging. Some of these events are routed to a central logging server and saved to file. Tailoring Log4J to selectively route and aggregate information to the logging server vs. disposing information locally (e.g. file) is easily managed within Log4J.

For these experiments we wanted to derive a model of the service architecture from execution events that we could then visualize. We used factory events (creation of

Logical Data Model objects placed on Cougaar blackboard) from five (5) Cougaar nodes executing the BBN-DASADA June 2001 demonstration (“Abstract Query Engine”).

When a Service Provider creates a workflow element using the factory, sufficient information as to the “syntactic” wiring of the workflow can be obtained. Example, the factory interface enforces the following contract to create an object: ***“to create a BidAccepted Task requires that the caller must also provide a handle to the parent BidRequested Task..”***

In addition to factory object creation events, publish/subscribe events are also collected. Publish/subscribe events are not used in the experiments described in this paper. However, they will be used (FY 2002) to provide information to perform runtime validation of the workflow structure using service execution metrics from a distributed system. Cross event-stream integration/correlation is expected to provide useful measures of the extent to which external visualization and analysis models (e.g. ADL) are in fact “consistent” with distributed service execution.

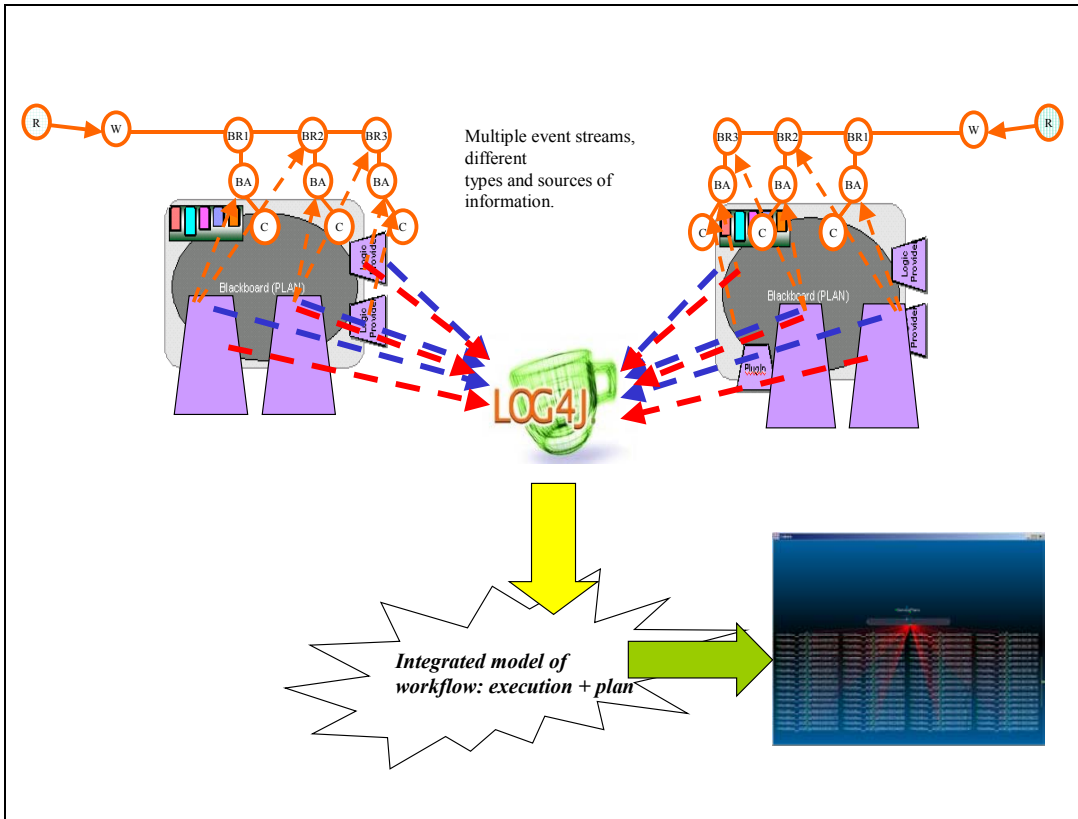


Figure 2 Integrating event streams to provide consistent view of distributed dynamic workflow.

Provided below are two sample logging messages. The first (1.) is an XML message that was used to test the throughput of the Log4J infrastructure (results given in later section). The XML style will be used by a number of DASADA event infrastructures that will replace Log4J in the BBN-DASADA system in the long run. The second logging message (2.) illustrates the format of Factory events used to identify workflow elements and their relationships. These events were used to generate the visualization model described later.

1. Illustrative XML logging message:

```
13850 INFO [main] dasada.examples.logging.test.TEST_EVENTS - <?xml
version="1.0"?><publishEvent xmlns="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"><type>Workflow</type><name>SC7</name></publishEvent></xml>
```

2. Factory event messages containing workflow element . relationship tuples:

```
13539 INFO [main] com.bbn.openzone.core.ldm.AAITFactory.FACTORY_EVENTS -
{CONCEPTCONTEXT,Concept__CONNECTIVITY_GAUGE,IsA,file:/C:/aai_openzone/install/dasada/sc
ripts/aeq.app.daml#CONNECTIVITY_GAUGE}
```

Reconstruction of Service Architecture from Events

Logical Data Model objects are created in the BBN-DASADA system using factories. BBN-DASADA “Service and Contract” factory creation events are routed to a logging server to reconstruct a workflow-based architecture of distributed services. For these experiments, the architecture model was viewed using a commercial hyperbolic information navigator. This particular visualization metaphor is experimental and to be evaluated. It is a promising navigational style, however both performance and “informational”/ “communicative” of the product for visualizing large workflow structures is to be determined.

Later (2002), we’ll be looking at providing alternative standardized architecture views (using ADL notation) from same or similar event streams.

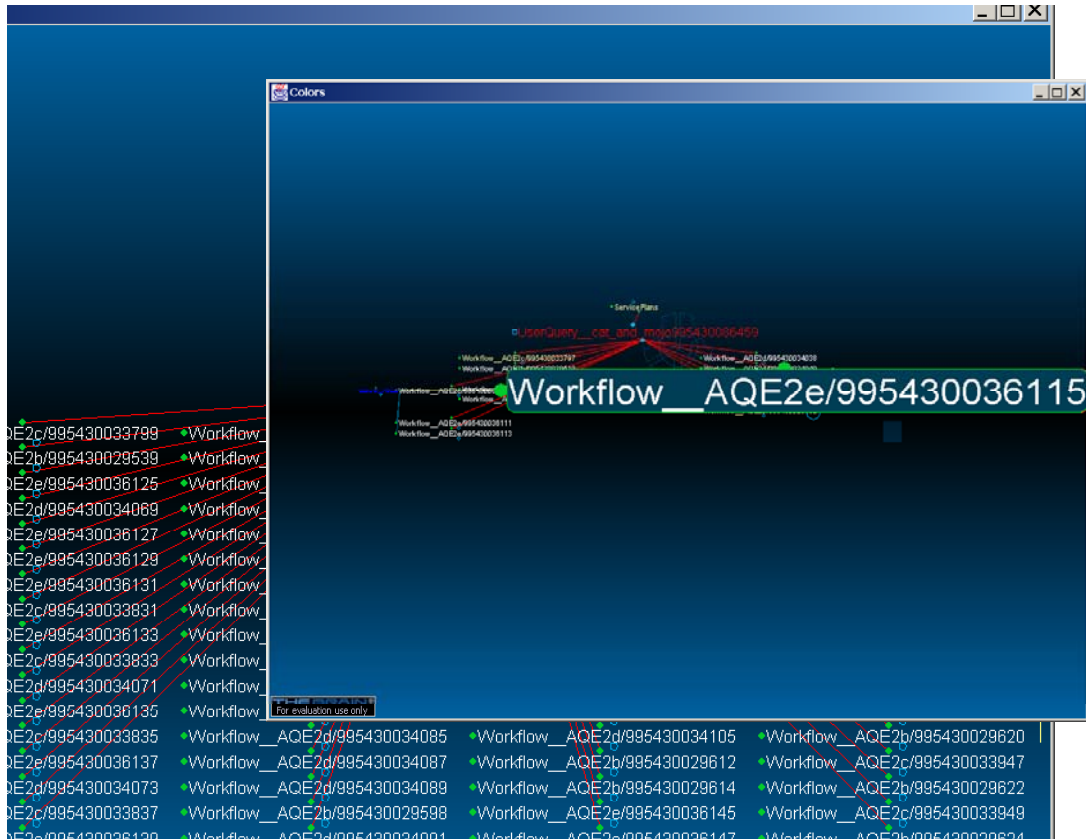


Figure 3 Navigating the workflow structure of a service architecture representation.

Performance Considerations of Log4J Infrastructure

The table below crudely illustrates the performance profile of the Log4J logging infrastructure with respect to 1 to 3 clients using a single Log4J logging server to log events. These experiments were conducted on a single desktop (P3 700). Individual test events were of the form:

```
13850 INFO [main] dasada.examples.logging.test.TEST_EVENTS - <?xml
version="1.0"?><publishEvent xmlns="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"><type>Workflow</type><name>SC7</name></publishEvent></xml>
```

Number of clients	Total event messages (# clients * 10K)	Average time for client to complete	Through-put of data to server.
1	10K	9.4 secs	2.1mb
2	20K	17.9 secs	4.3mb
3	30K	25.5 secs	6.5mb

Note. The “server bottleneck” is implied by average “client time” increase. Log4J can be configured to support multiple logging servers implying unexplored scalability options.

Conclusions

Described is an experimental/working solution for capturing and aggregating events from a distributed software system. Early results look promising. Virtues include considerable flexibility and ease-of-use.

Acknowledgements

This work is sponsored by the Defense Advanced Research Projects Agency (DARPA) under the DASADA (Dynamic Assembly for System Adaptability, Dependability, and Assurance) program.

Works Cited

1. “Service and Contract” distributed workflows: <http://aai.bbn.com/cougaar>
2. BBN-DASADA: <http://aai.bbn.com>
3. Dynamic Assembly for System Adaptability, Dependability, and Assurance (DASADA): <http://schafercorp-ballston.com/dasada/index2.html>
4. Cognitive Agent Architecture Open Source: www.cougaar.org
5. Log4J Open Source: <http://jakarta.apache.org/log4j/docs/index.html>