**Background**

The Distributed Application Control System (DACS) is the backbone of the automatic processing of seismic, hydro-acoustic and infrasound (SHI) waveform data at the IDC. It drives the execution of processing applications by organizing data into time intervals, processing steps into pipelines, and using message queues for task scheduling.

**Objective**

Because licensed software hampers free distribution and use of IDC software and by National Data Centres, IDC aimed at eliminating the dependency on the proprietary Tuxedo middleware.

**Accomplishments**

We redesigned the existing system and implemented the new IDCDACS based on an open-source messaging solution in combination with existing in-house IDC libraries and a custom-developed application framework, which together replace Tuxedo in a robust, reliable and scalable way.

**Message Queue for Task Scheduling**

![Diagram of Message Queue for Task Scheduling]

Figure 2: Task Queue decouples Publishing and Consuming Clients. Yields a reliable and robust system. Permits parallel processing, for load balancing and scaling up.

**Software Application Framework**

![Diagram of Software Application Framework]

Figure 1: Building blocks for common functionality like main control loop, exceptions, resource management, transactional messaging, client-side failover for high-availability cluster.

**Processing Pipelines and Message Flow**

![Diagram of Processing Pipelines and Message Flow]

Figure 3: DACS processes (blue) run as daemons and listen on Message Queues (purple). Scheduler wakes up Data Monitor. Data Monitor checks conditions (data availability, status information, time, etc.) and creates Time Intervals. Tortoise executes Processing Application (aqua), reports status to Interval Table, and publishes to next Task Queue. Each Time Interval passes through all steps until reaching Done Queue. In case of an error it will be short-circuited into Failed Queue.

**Deployment and Distribution**

![Diagram of Deployment and Distribution]

Figure 4: 1 Database, 1 Message Broker, 1-N Processing Servers for DACS Server Programs and Processing Applications.

**Analyzable and Maintainable Code**

```c
/* Create the message processing state machine */
machine = StateMachine.new(10, "process", shell);
```

**Open Technologies**

Our solution utilizes the RabbitMQ high availability message broker and the Advanced Message Queuing Protocol (AMQP), an open industry standard and wire-level protocol mandating that senders and recipients can interoperate irrespective of their specific implementation.

**Development Approach**

The new IDCDACS was implemented using the Scrum agile development methodology, aligned with evolving requirements and priorities. The first major release is running on the IDC development LAN. Transition to Testbed and Operations LANs is in progress. Further development will improve usability, extend features, and lift current limitations.