

Panel - “Middleware for Real-Time Distributed Objects: Needs and Requirements from Different Application Domains”

Panelists:

Bran Selic - Rational - Canada - bselic@rational.com
Richard Turner - DoD - USA - Rich.Turner@osd.mil
Rob van den Berg - Philips - Netherlands - rob.van.den.berg@philips.com
Carlo Batini - AIPA - Italy - batini@Aipa.it
Greg Bolella - SUN - USA -
Luiz Bacellar - United Technologies Research Center - USA - bacelllf@utrc.utc.com
Sanjay Razdan - Tridium - USA - srazdan@mail.tridium.com

Panel Organizer:

Carlos E. Pereira - United Technologies Research Center - USA - pereirce@utrc.utc.com
(on leave from Federal University of Rio Grande do Sul - Brazil)

Motivation:

The main goal of this panel was to evaluate the needs and requirements that different application domains impose on middleware based on distributed real-time objects. Panelists were specialists from different application domains, with a large experience on the development of complex real-time distributed systems.

Richard Turner (software intensive military applications)

The US Department of Defense plans a large number of network-centric systems to implement warfighting, C4ISR, and support functions. These systems will need well-behaved, predictable, robust, and high-speed middleware to meet their mission critical requirements. While there is similarity in the requirements of military and commercial systems (particularly in the transportation and medical domains), military systems are generally longer-lived, more complex, require more varied interfaces, and must operate in a more hostile environment than commercial systems.

There is currently some concern about the reliability and performance of existing real-time middleware. Research in this area is a priority, particularly where middleware can support rapid reconfiguration, multi-level security, and support for widely varying legacy components.

Bran Selic (telecommunication)

Most telecommunications software can be grouped into one of two broad categories. In one case we have software that is involved in actual information transmission. This includes relatively simple functions such as setting up and tearing down of virtual circuits, routing of packets through protocol stacks, etc.

Transmission typically requires low latency and high bandwidth implying that software overheads must be minimal. The second category, often referred to as Operations, Administration, and Maintenance (OA&M), is used to control the actual telecommunications network and equipment. This software is usually much more complex, but generally requires less bandwidth and is much more delay tolerant than transmission software. COTS middleware—mostly CORBA based—is being applied with significant success in telecom systems but almost exclusively for OA&M. This is primarily due to the following reasons:

- Throughput and delay are still inadequate for most transmission functions (although the situation is improving steadily)
- Most COTS middleware assumes a heavyweight high-overhead threading model with unacceptable context switching times (most telecom systems implement very lightweight concurrency)
- The majority of telecom systems are based on proprietary hardware and software. This requires COTS middleware to be ported, which may be costly and difficult (thereby eliminating the primary benefits of using COTS software).
- Telecom systems in general have extreme availability and reliability requirements – something that is not supported by most COTS middleware
- It is necessary for middleware to respect and, if necessary, enforce end-to-end delay constraints; however, this requires mechanisms that are not found in most COTS middleware

Clearly, these shortcomings must be remedied before COTS middleware will be used in transmission software. A particularly useful feature for this purpose would be to provide reflective interfaces on middleware. The

application would use such interfaces to dynamically control the quality of service that it extracts from the middleware, by making resource management tradeoffs that suit its current usage profile. For example, depending on the traffic load, the middleware could be configured to support either fine-grained highly-dynamic concurrency (e.g., for datagram routing) or more heavyweight but resource-rich concurrency (e.g., for virtual circuits).

Rob Van den Berg (medical applications)

X-Ray & Middleware

Just as video camera's are replacing film camera's, X-ray devices have changed from film-based imaging techniques to digital detectors. As such the majority of development for modern X-Ray machine is software development. Typical requirements posed by the domain are:

- high variability: no one X-ray machine is the same, every customer has special requirements, every country has its own safety regulations
- They are used in a clinical environment. In operating theatres where dependable behavior is a must; Also image processing has to be real time, as it has to keep up with eye-hand coordination (if the surgeon moves something, he has to see the effect quasi immediately on the screen.
- Moreover, they can have a long life span. FDA regulations state that we must be able to service up to 12 years after end of production.

These aspects make that the following two requirements have to be addressed by middleware:

- *Configuration*: To be able to cope with high variability and the life the used middleware should be able to support a component-based approach: one should be able to devise parts that can be exchanged/upgraded independent from the rest of the systems. Moreover it is crucial that the middleware has an identification mechanism and versioning scheme, to guarantee that right parts are used.
- *Dynamic Behavior*: From an X-ray machine, guaranteed behavior is vital: no X-Ray should be left to waste. From a control point of view the requirements on middleware are not that difficult: there is very little or no emergent behavior during the hot-phase. Once the machine has been configured, a foot pedal is pressed and the patient is scanned. In all cases an error situation will just abort the process. However in this case, it has to show graceful degradation: the physician should still have an image on the screen! An aspect where requirements are quite strict is on the image

processing part: For certain endoscopic operations (e.g. in cardiology), images have to be processed in real time to make sure that the latency is below the latency for eye-hand coordination (~120 mSec). Also it is very data-intensive task: the data rates can be up to 60MB/s. Combining these two aspects means that middleware should make it possible to separate the control communication from the, low-latency, data communication. Also to be able to use a CPU efficiently (e.g. use of assembly language, c-extensions), it must support the use of low-level concepts.

Carlo Batini and Massimo Mecella (e-government)

In 1993, the need for better coordination of efforts and investments in government information systems pushed the Italian Parliament to create the Autorita' per l'Informatica nella Pubblica Amministrazione (Authority for Information Technology in the Public Administration). AIPA's general assessment called for it to promote technology innovation by defining criteria for planning, implementing, managing, and maintaining the Italian Public Administration's information systems. Of the various initiatives that AIPA has undertaken since its creation, the Unitary Network is the most important and challenging, as it represents the main step of the Italian approach to e-Government. The Unitary Network is a "secure Intranet" on top of which a Cooperative Architecture is being designed; such application architecture is a reference distributed computing model in which administrations are represented as Domains and exchange data and application services with the others through Cooperative Gateways.

In recent years, some projects have sought to validate the cooperative architecture. This talk outlines some lessons learned from this experience. Distributed object middleware has been used only in a subset of the projects, those one more targeted to the experimentation of innovative solutions in a small/medium set of administrations, whereas projects concerning more widespread applications and critical processes of the Italian PA have adopted more consolidated technologies. Distributed object middleware still represents a high technological risk.

Moreover the experience suggests that middleware alone is not enough, but other issues need to be addressed. The set-up of a cooperative development process and the management of common semantics of the business objects, to be carried out through meta-repositories, are crucial in developing effective cooperative information systems such as e-Government ones.