Operational Needs of Sprint's EAI Environment

OMG EAI Workshop
Orlando, FL
Feb 9th, 2000

Wing K. Lee
Business & Technical Architecture
Sprint
wing.lee@mail.sprint.com

Sprint.

A brief history

- **1990**
  - first foray into the partitioning of business logic and UI out of the mainframe environment
  - primarily C based monolithic apps
  - users love the productivity enhancements
  - immediately recognized dramatic increase in complexity
    - points of failure
    - development and testing
    - deployment and support
- **1993**
  - distributed computing and OO sanctioned as strategic directions
  - multi-platform; multi-lingual environment
  - formalization of infrastructure initiatives into a technical architecture organization

Today: distributed objects everywhere

- Component based two to n-tiered architectures built on CORBA and EJB
- Service based; process driven
- At the heart of our component infrastructure
  - point-to-point + multi-point distribution services
  - enterprise directory service
- Integration across technologies and domains
  - HTML - C++ - Java - Smalltalk - COBOL - Natural
  - Oracle - Versant - Adabas - VSAM - DB2
  - NT - UNIX - Tandem Guardian - MVS/CICS

And that comes with new challenges…

- We are finding ourselves re-creating the like of CICS in a heterogeneous environment
  - manageability
  - scalability
  - security
  - transactions
- Of all the attributes, manageability is most fundamental
- It is undoubtedly a key success factor that is often overlooked or become, at best, an afterthought

Business drivers

- Huge reliance on our distributed applications and the infrastructure they run on
- Time-to-market pressure calls for the streamlining of effort to deploy and manage applications
- Ongoing SLA management needed to gauge overall health of IT infrastructure
- The need to increase operational and cost efficiencies in managing heterogeneous:
  - applications
  - systems
  - networks
  - all under one management console

Technical Drivers

- Performance Management
- Configuration Management
- Fault Management
- Security Management
- System Management

Operational expectations (1/3)

- Performance Management
  - collection of performance metrics* from apps, BOA/POA and ORB (both pull/push semantics)
    - need standard API's
    - need standard formatting of log data
    - * take BOA/POA for example, some standard metrics could be:
      - number of outstanding requests
      - current thread pool count
      - number of requests per sec
      - average number of requests over a given time period
  - application response management (push from applications)
    - need standard API's
    - ARM from Tivoli/HP is a good example
  - ability to start/stop additional server objects depending on traffic
    - need standard scripting support

Operational expectations (2/3)

- Fault Management
  - detection
  - tracking
  - need standardized formatting and API's for
    - reporting and tracing of fault information
    - remote debugging
- Configuration Management
  - ability to dynamically change the configuration of BOA/POA, ORB & server objects, for example:
    - max thread for BOA/POA
    - thread idle time for BOA/POA
    - max incoming sockets
    - max outgoing sockets
    - threading policies

Operational expectations (3/3)

- Security Management
  - access control
  - audit logging
  - policy and user profile administration
  - excellent opportunity for CORBA security service integration
- System Management
  - correlate system data with application data
  - extremely useful for SLA measurement

Using COTS is our preferred path

- System Management Software
  - software distribution
  - enterprise console
- Application Management Software
  - multi-level application life cycle management
    - fine grain management at the server object level
    - logically grouping of server processes
    - definition of dependencies to facilitate fully automatic recovery
    - SDK for homegrown apps
    - non-intrusive support of 3rd party applications
  - performance monitoring
    - collection of pertinent data and the ability to log historical data
  - event management
    - ability to report to enterprise console
    - ability to store event in a repository for tracking purpose

Highlights of application management requirements

- Criteria:
  - feature richness as described in previous slides (PM/FM/CM…)
  - ability to provide full life cycle support
  - bi-directional integration with enterprise console
  - easy-to-use console interface
  - integration
    - completeness of SDK to instrument home grown apps
    - non-intrusive support of 3rd party apps
  - openness to database/repository connectivity
  - H/A friendly
  - firewall friendly
  - software quality

Thanks…