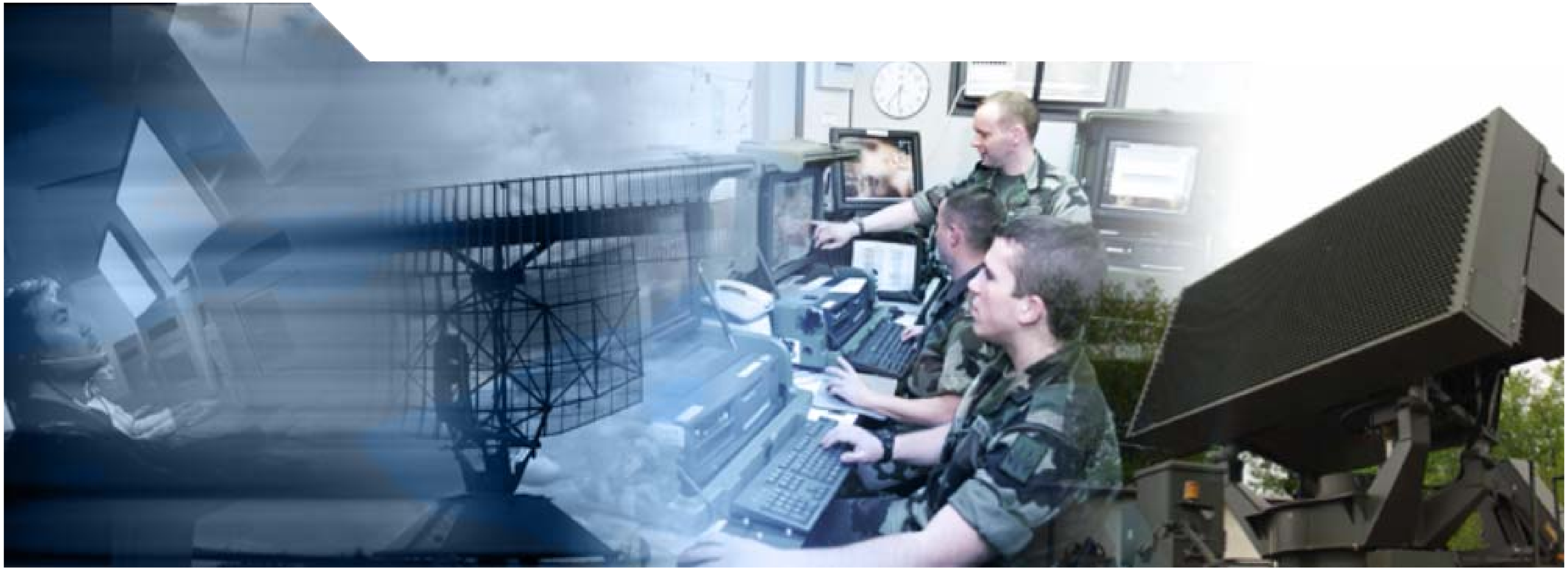
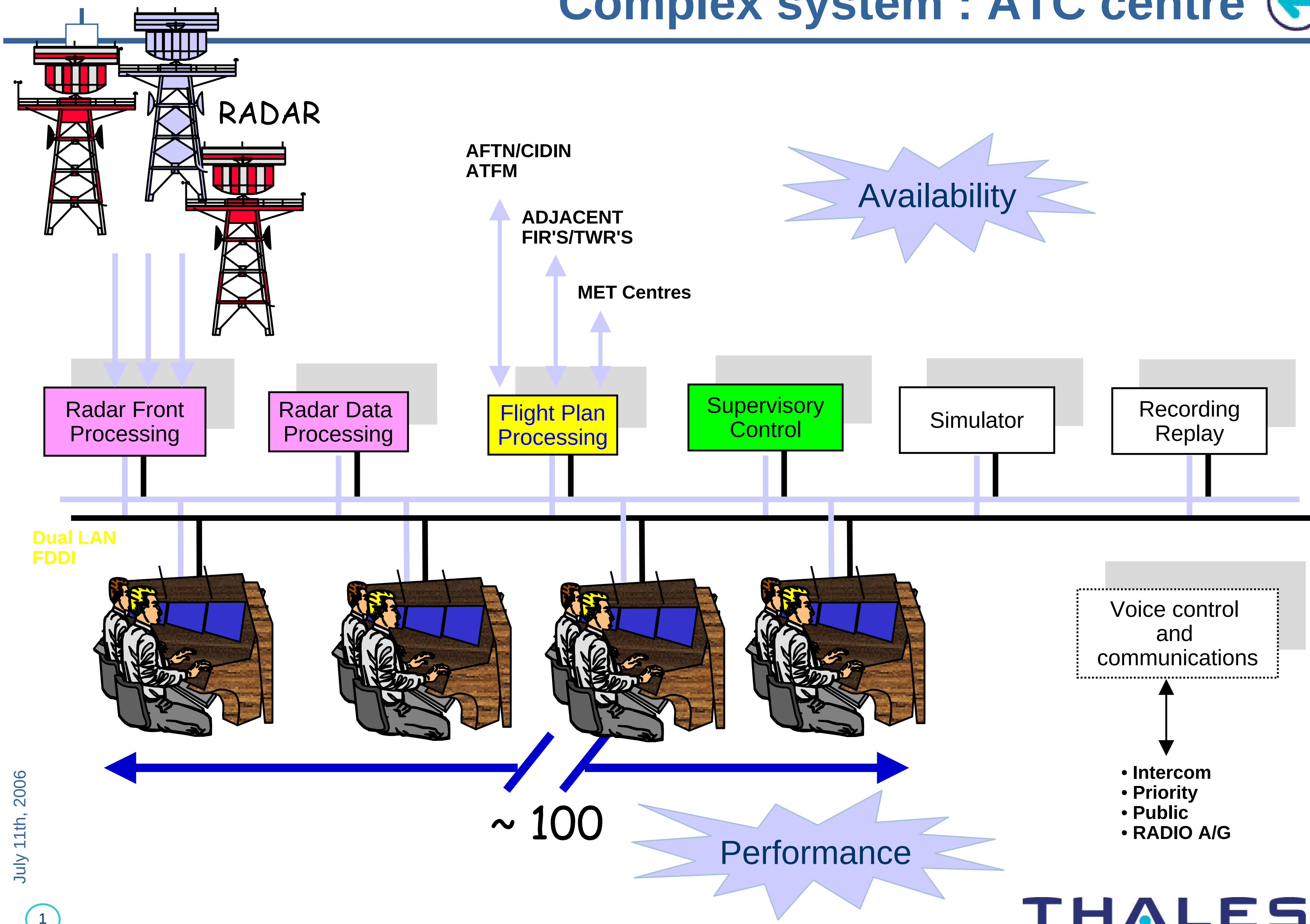


CORBA Fault Tolerance for Mission and Safety Critical Systems

Arlington, July 11th, 2006

H. Souami

Complex system : ATC centre

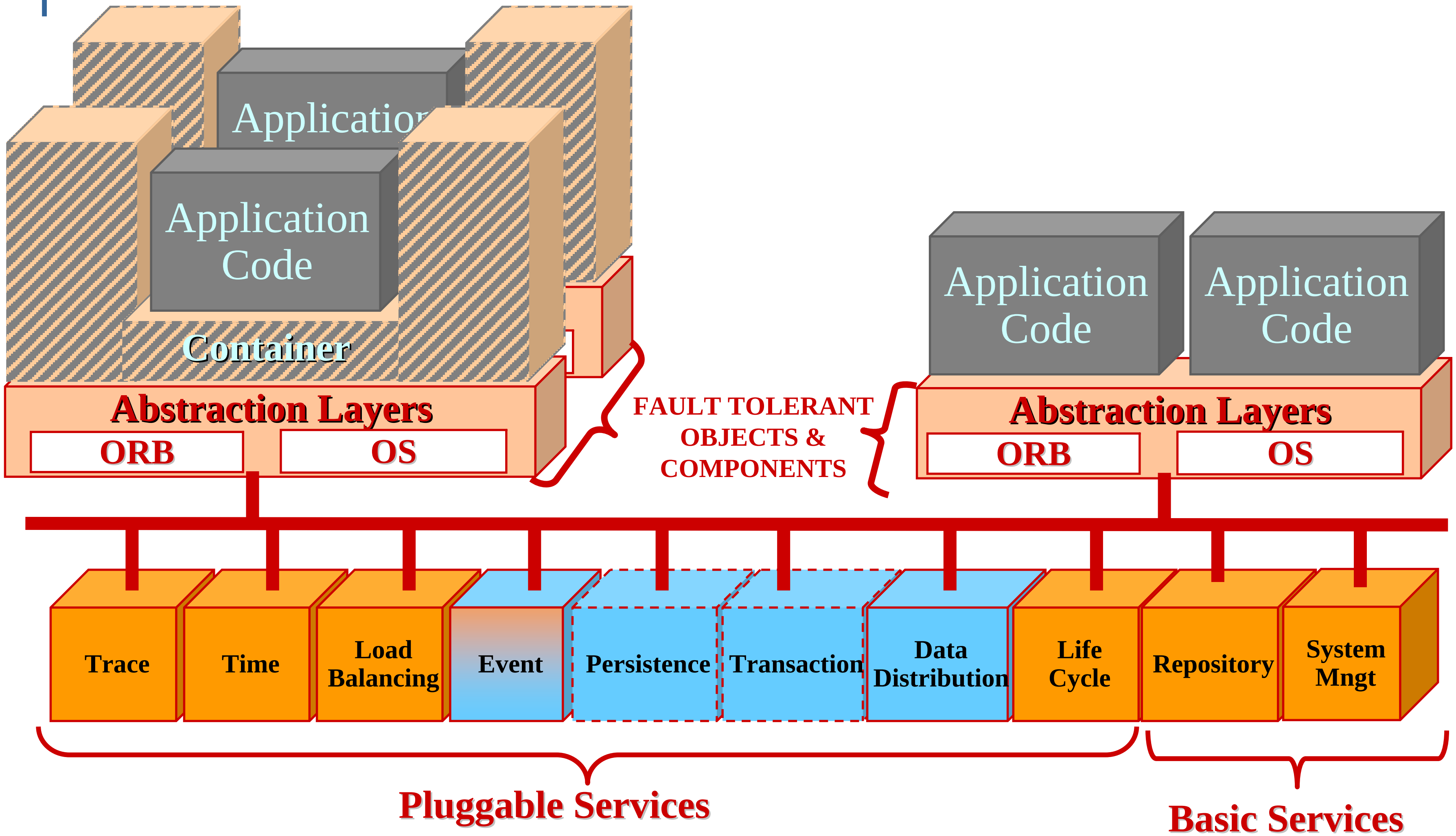


THALES

COFLIGHT (future French Italian Flight Data Processing System) architecture, demonstrated the need for:

- Fault Tolerance
  - Critical functions require strong data consistency
- System Management
  - Efficient control and monitoring of the applications and sub-system
- Data Distribution
  - Distribute Flight Plan data to ever growing number of Controller Working Positions (CWP)
- Component-Based Development
  - CCM eases deployment and configuration of system components

CARDAMOM General Architecture



Current Reference Platform:

- OS: PC/Linux
- ORBs: OpenFusion, JacORB

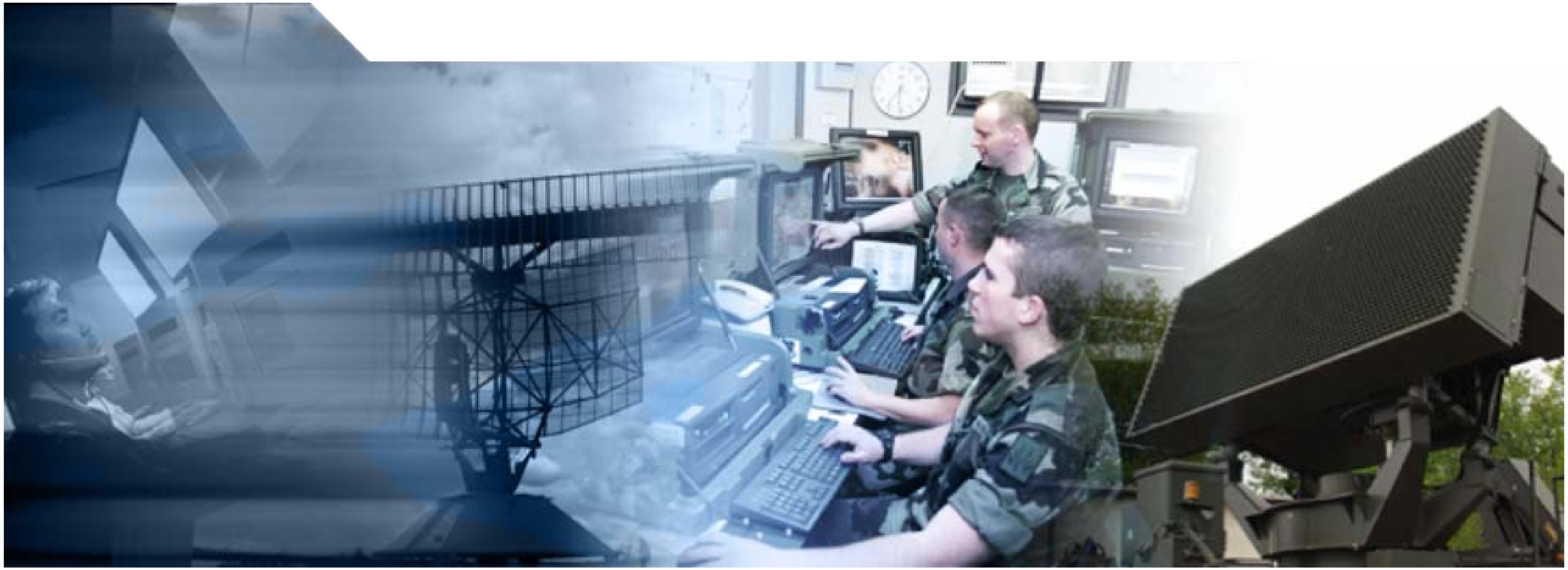
THALES

Common Development Organisation

Community Edition on http://cardamom.objectweb.org

Licence: LGPL

THALES



CARDAMOM Fault Tolerance

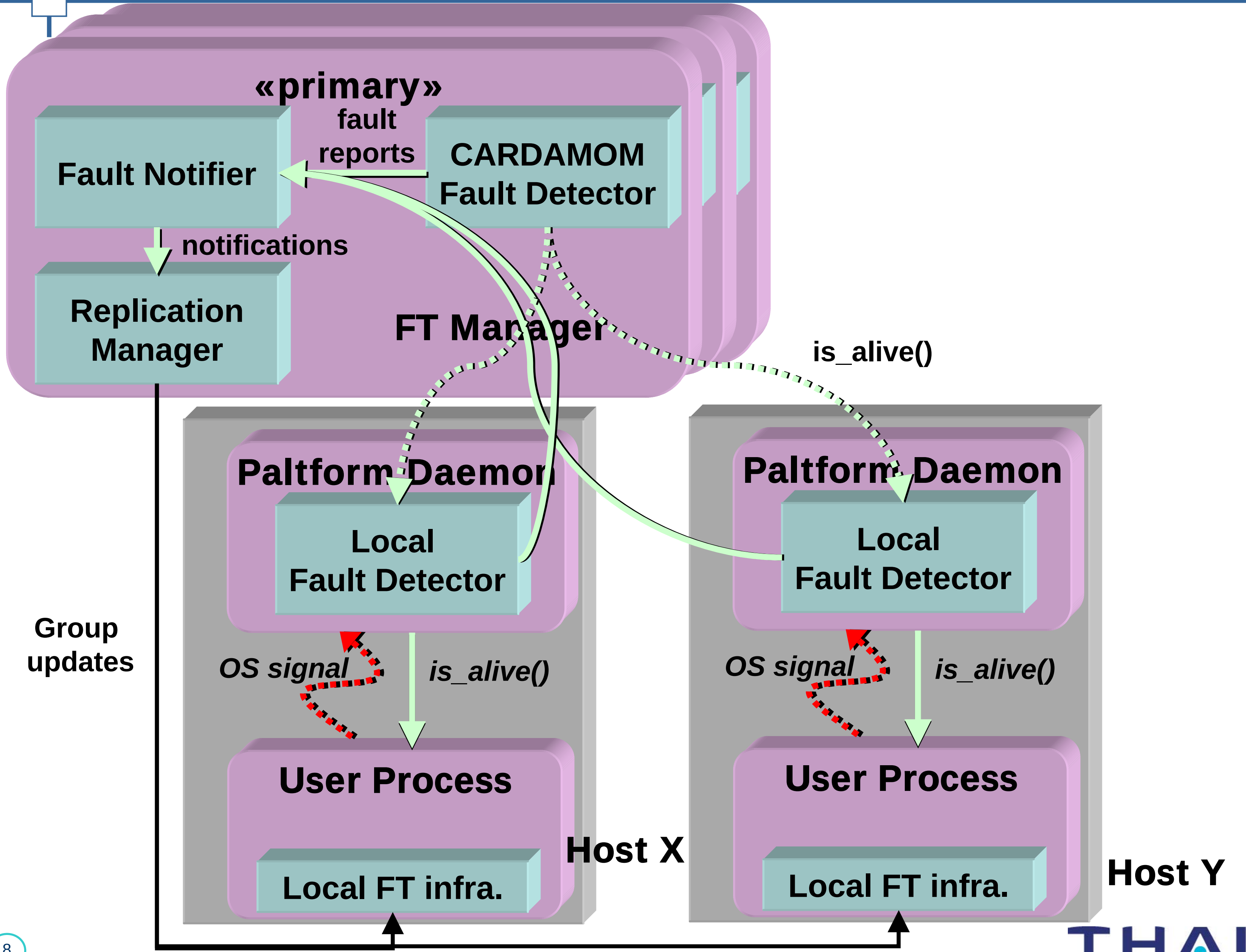
Air Systems Division

- No single point of failure
  - 99,99% availability for typical ATC system (~downtime 5 min /month)
- Hardware based solutions are hard to maintain
  - CCIS systems have a 10 to 20 years lifecycle
- Replication and Failure transparencies
  - Object Groups (IOGR and fail-over semantics) provide both replication and failure transparencies
- Warm Passive replication
  - ‘Field proven’ technique widely used in CCIS systems such as Air Traffic Control, Naval Combat Management and Airborne Command & Control systems
  - Can be implemented on non-RT OS, no special hardware…
- THALES experience in CORBA middleware
  - SCADA, OASIS, PERCO,COBRA, …

CARDAMOM V2 Capabilities

- Standard FT CORBA Replication Manager, Fault Notifier, and Fault Detector
  - Group membership, and
  - Failover semantics
- No single point of failure in FT Infrastructure
  - Fully distributed FT Manager
  - Use of IOGRs for FT Infrastructure objects (Replication Manager, …)
- Warm passive replication of objects and components
- Multicast-based monitoring (MIOP) of Platform Daemons (MIOP)
- Notification upon:
  - Fault detection, Violation of minimum number of replicas, Server reconfiguration
- CARDAMOM Add-on frameworks
  - Activation Framework
  - State Replication Framework with strong data consistency
- Enforcing '*at most one'* semantics of CORBA
  - Use request & reply logging

CARDAMOM Fault Tolerance Architecture



Pull-based fault detectors

- Invoke at regular intervals a specific operation on monitorable objects to detect their liveness
- Opportunity to perform some **sanity checks** (negative return of *is_alive())*

Push-based fault detectors

- Rely on OS signals to detect the death (crash) of a process and thus detects the crash of any object/component contained in it.

Fatal error reporting

- Applications may (voluntarily) report a *'fatal error'* condition to the CARDAMOM System Management.
  - The calling process is then immediately aborted/killed to avoid any side effects.

Problems solved during implementation:

- State Replication
  - Use of PSS, OO Database …
  - Backup insertion
- Bootstrap
  - Resolve ReplicationManager,
  - Naming object groups,
  - Narrowing empty groups
- Older IOGR versions
  - Ultimate fallbacks
- Replication level
  - FT properties at process-level
- FT_GROUP_VERSION Service Context
  - Efficient server-side interception with object group Id
- FT Components
  - Component Group Abstraction
  - Navigation/Introspection

CARDAMOM State Replication Framework

- Allow use of High Value Persistence (COTS)
  - Allow use of PSS, DDS, OO Database … instead of sequence<octet>
- CARDAMOM State Replication Framework
  - Strongly-typed ***'Data Stores'*** (C++ templates)
    - Map abstraction {OID, DATA}
  - Deployed on each replica's memory space
  - Strong consistency of caches
    - Multithreaded Transaction Engine
  - Configurable transport
    - Multicast (MIOP), or
    - Point to point (IIOP)
  - Non-blocking backup insertion
  - Concurrent insertions of backups

THALES

File, corbaloc URL, even Name Server may not be acceptable

- No NFS, no fix location … etc.

Storing object group Ids not suitable for ‘cold’ restarts, testing and prototyping … etc

- CARDAMOM Service Locator (discovery protocol) for FT Infrastructure objects
- CARDAMOM Replication Manager provides *get_object_group_ref_from_name()* operation (supports "*com.thalesgroup.cdmw.ft.Name*“ property).

Narrow on empty groups (no members) required at start-up .

- Fallback in FT Manager implements *is_a*() operation.

IOGRs at the client side are only updated when used. Its profiles may not be valid anymore!

- Use an IOR from primary FT Manager as a fallback profile in IOGR still suffers from failure of the primary FT Manager.
- Add an ultimate fallback to a Service Locator on client nodes.

IOGRs for objects, components, facets … etc. are great!

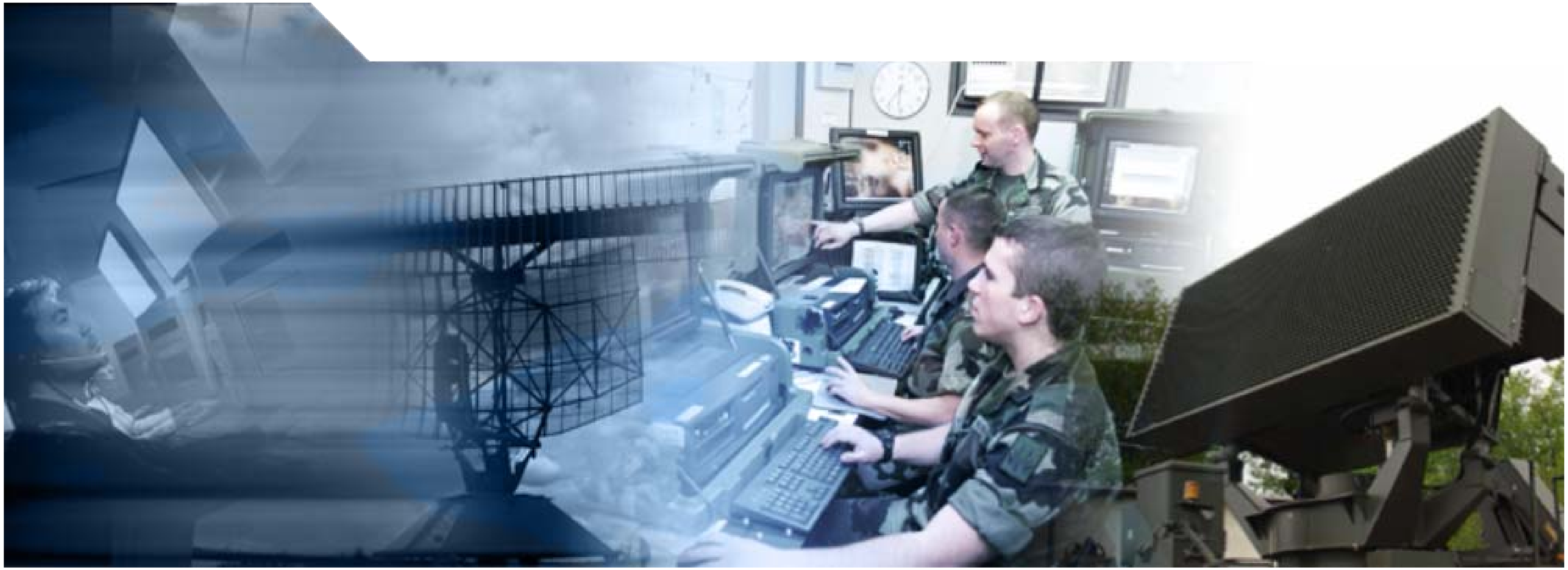
- All members within the same process/component server share resources.
- Should be all primaries, or all backups.

- Need for object groups and components groups abstractions but FT properties shall be enforced at process level.
- Supervisors are only interested in ***Process Groups*** (when OS provides processes).
- Process/Component Server state includes Container state (event subscriptions … etc.)

THALES

Extension of Object Groups to Component Groups (CCM)

- A monolithic container = 1 IOGR
- A segmented container = many IOGRs
- Components::Navigation and Object::get_component() operations FT aware:
  - Return IOGRs when called via an IOGR
  - Return IOR when called via an IOR
- Update of Assembly and Deployment tool for assembling FT components (call of add_member() … etc.)



Thank you for your attention

Hakim Souami: hakim.souami@thalesatm.com