UNCLASSIFIED

OMG-DDS "Exploiting the Potential"

[Proven suitability in the Naval Combat System Domain]

OMG Real-time & Embedded Systems Workshop

July 2005

Hans van 't Hag

hans.vanthag@nl.thalesgroup.com

Defence

THALES

CONTENTS

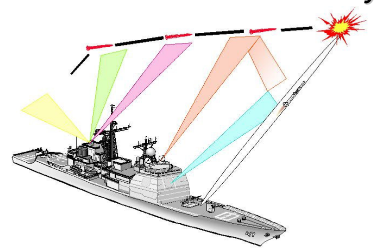
(1) ARCHITECTURE TRENDS

"Is there life after client-server …"

DARPA - Open Architecture Focus

Key Trend
- DoD system requirements are increasingly more dynamic, diverse, & demanding

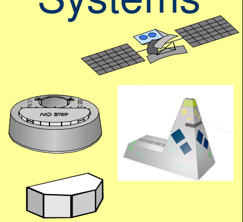
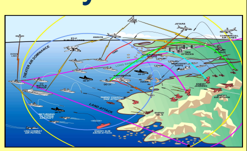
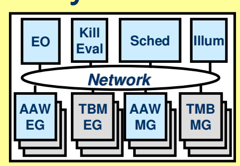
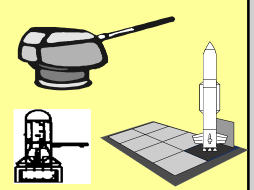
Problems
- Existing architectures - brittle & configured statically
- Existing COTS - too big, slow, buggy, incapable, & inflexible
- Existing multiple technology bases - proprietary & limit effectiveness by impeding
  - *Assurability* (of QoS),
  - *Adaptability* &
  - *Affordability*



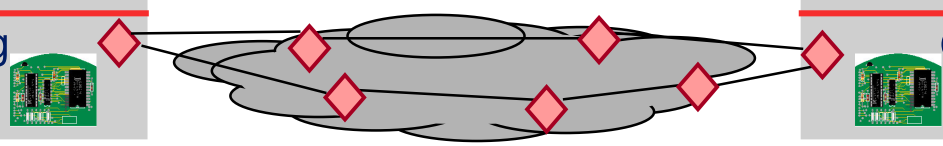
Today, each system brings its own:
networks
computers
displays
software
people

Applications

Applications

Sensor Systems	Command & Control System	Engagement System	Weapon Control Systems	Weapon Systems
				
Technology base: **Proprietary MW** **Mercury** **Link16/11/4**	**Technology base:** **DII-COE** **POSIX** **ATM/Ethernet**	**Technology base:** **Proprietary MW** **POSIX** **NTDS**	**Technology base:** **Proprietary MW** **VxWorks** **FDDI/LANS**	**Technology base:** **Proprietary MW** **POSIX** **VME/1553**

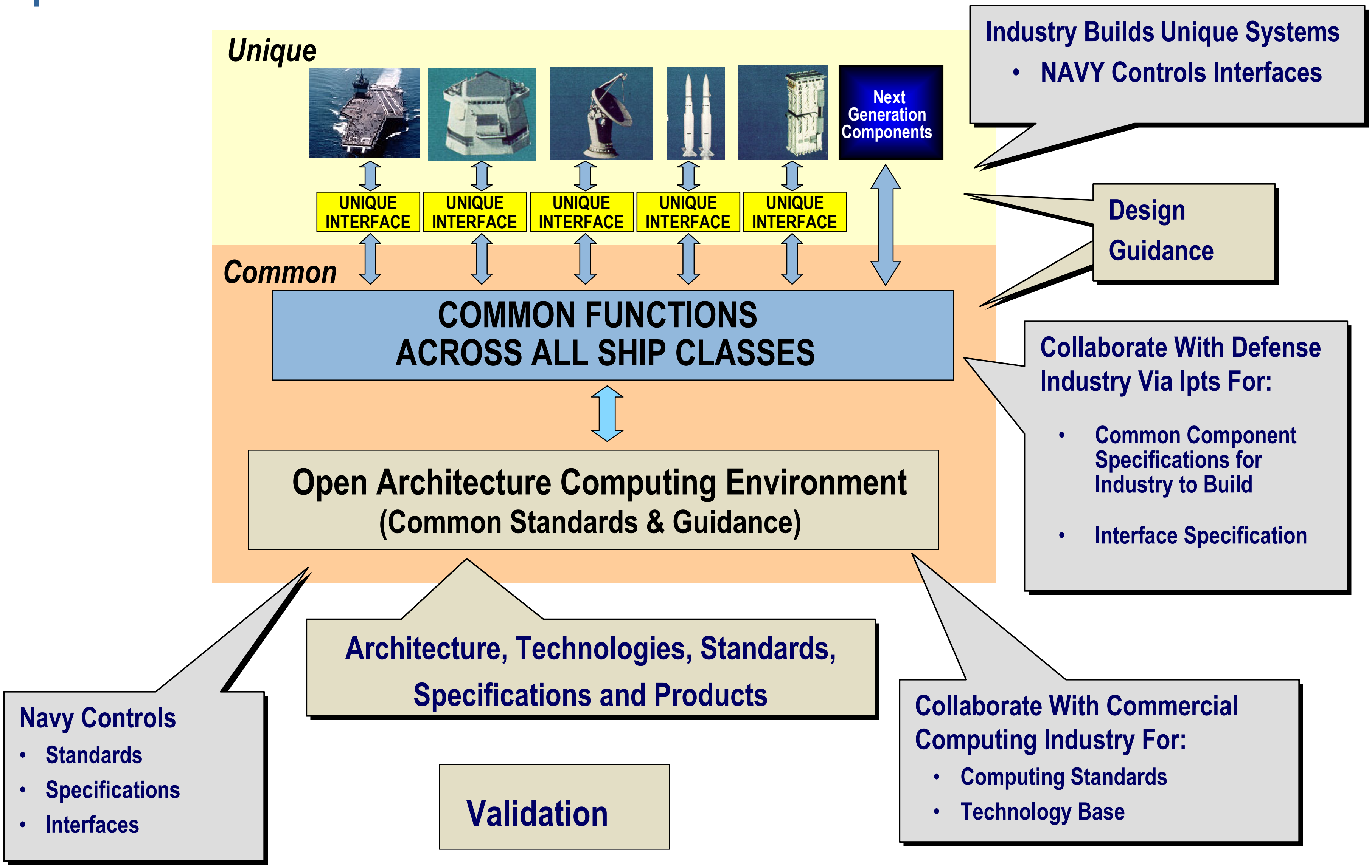
Operating System



Operating System

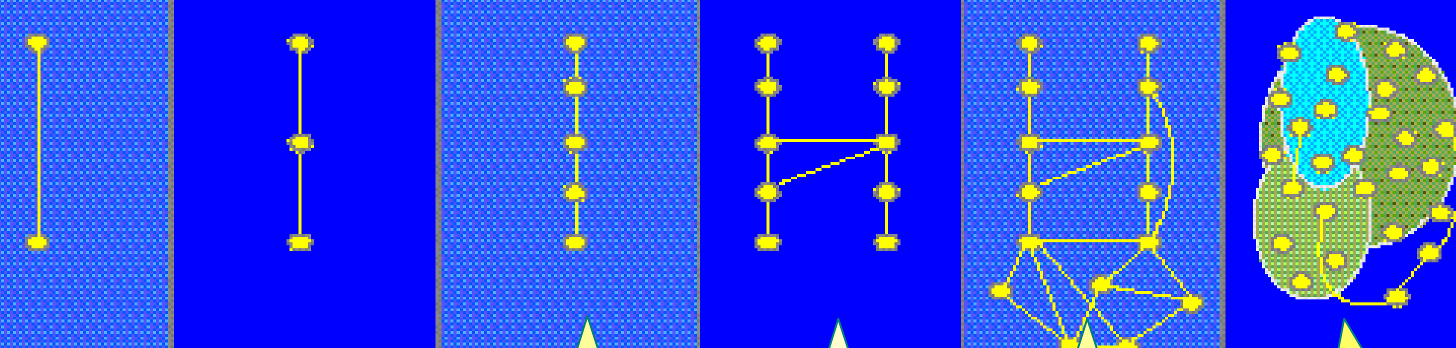
Consequences
- ***Hard to meet required performance levels***
- ***Hard to control distributed resources***
- ***High software lifecycle costs***
  - *e.g.,* many "accidental complexities" & low-level platform dependencies

Open Architecture Computing Environment (OACE)



TRENDS: © SUN

	Client-Server	3/N-Tier	Net Apps	Net Services	Next	After that
Catch Phrase	The Network Is the computer	Objects	Legacy to the Web	The Computer Is the Network	Network of embedded things	Network of Things
System						
Collections						
Components						
Scale	100s	1000s	1000000s	10000000s	100000000s	1000000000s
When/Peak	1984/1987	1990/1993	1996/1999	2001/2003	1998/2004	2004/2007
Leaf Protocol(s)	X	X	+HTTP (+JVM)	+XML		Unknown
Directory(s)	NIS, NIS+	+ CDS	+ LDAP (*)			+ ?
Session	RPC, XDR	+CORBA	+CORBA, RMI	+ SOAP, XML	+ RMI/Jini	+ ?
Schematic						



“OMG-DDS ??!!”

Corba

Web

Java

Information Grids

(2) Combat Management Systems

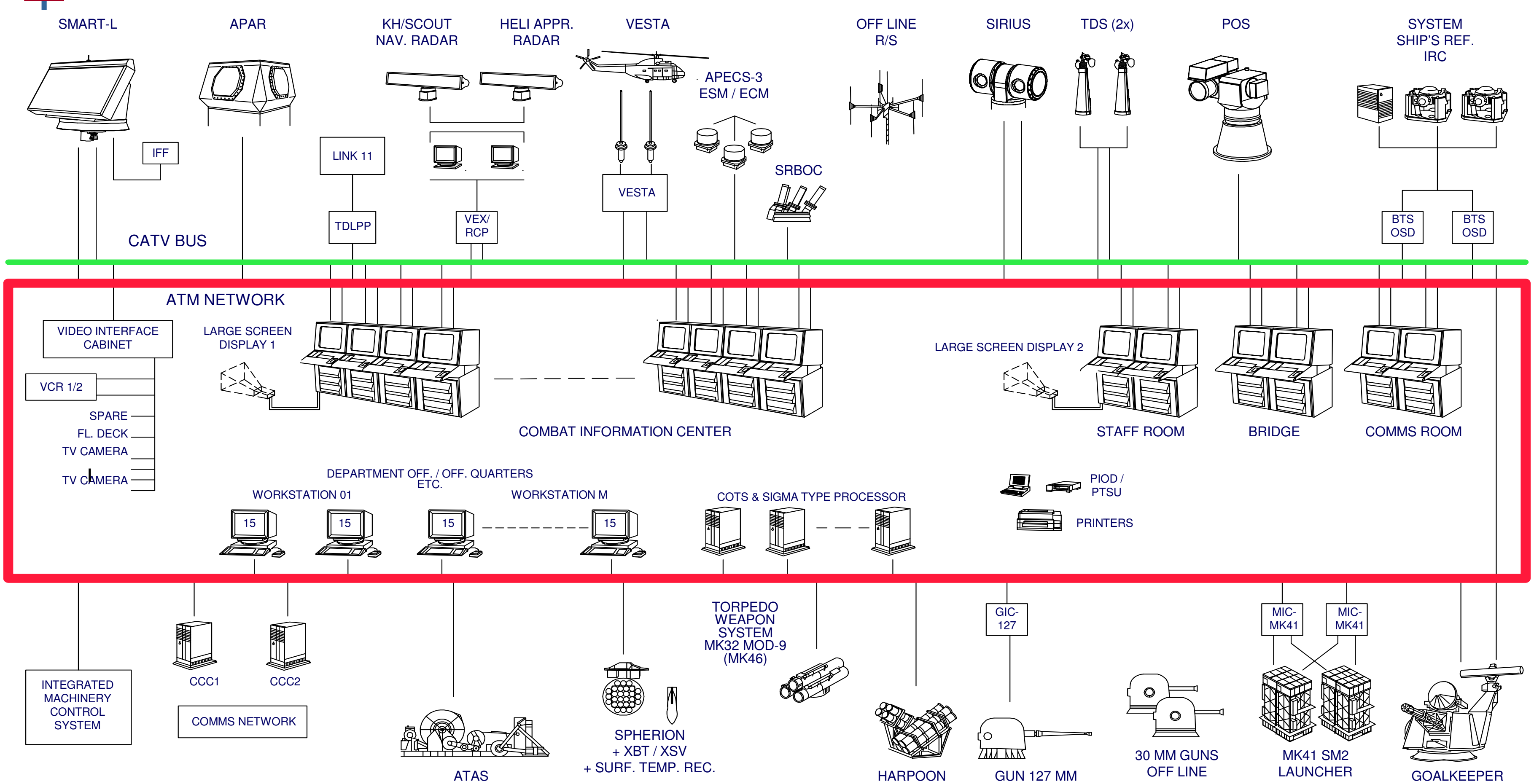
"If your live depends on it …"

Metrics: F124 Frigate

OPERATIONAL REQUIREMENTS

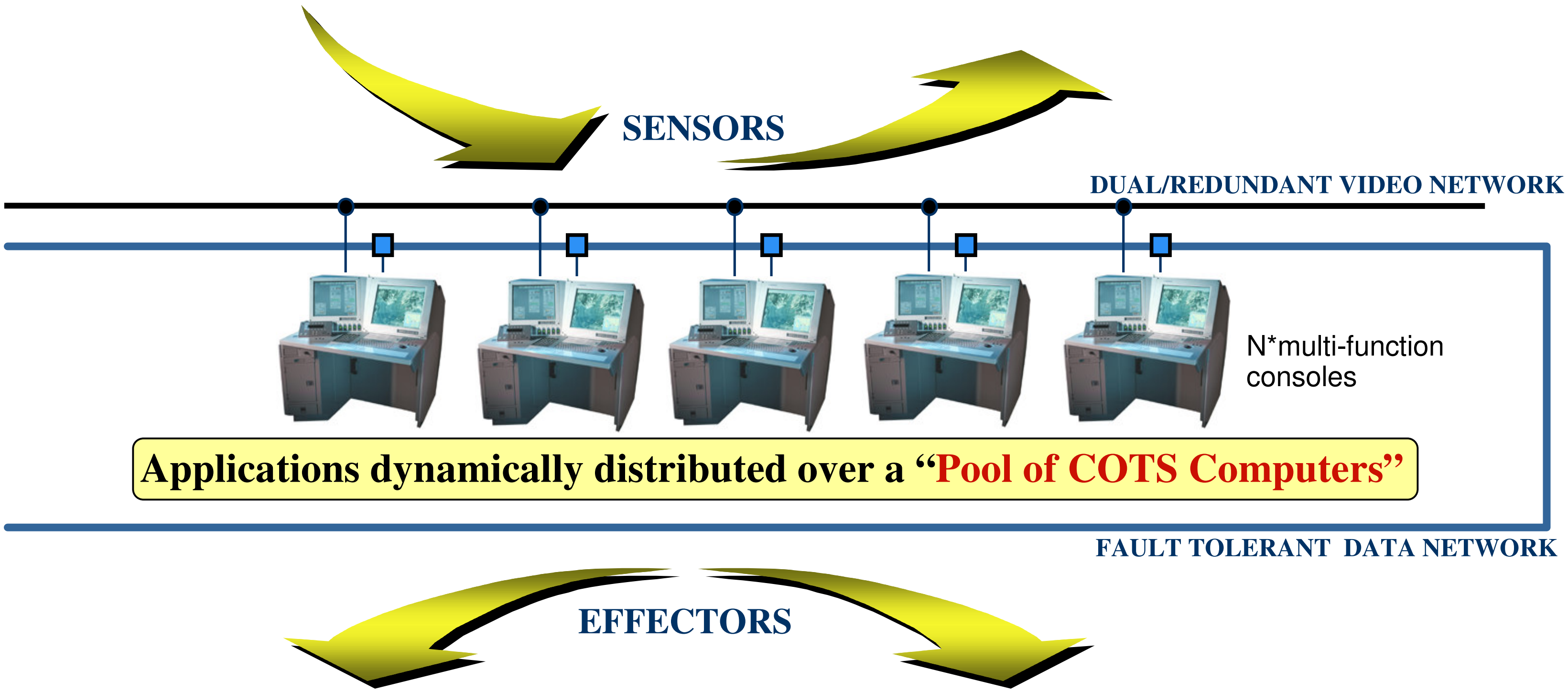
- Vast amounts of information
- Thousands of tracks
- High degree of automation
- Automatic multi- sensor data fusion
- Automatic threat evaluation
- Automatic sensor + weapon pairing
- Extensive Command Support
- Easy to Operate
- Easy to Maintain
- Flexible in Missions, and Manning
- On-line Simulation & Training
- Total System Integration

Metrics F124 Frigate: Overview



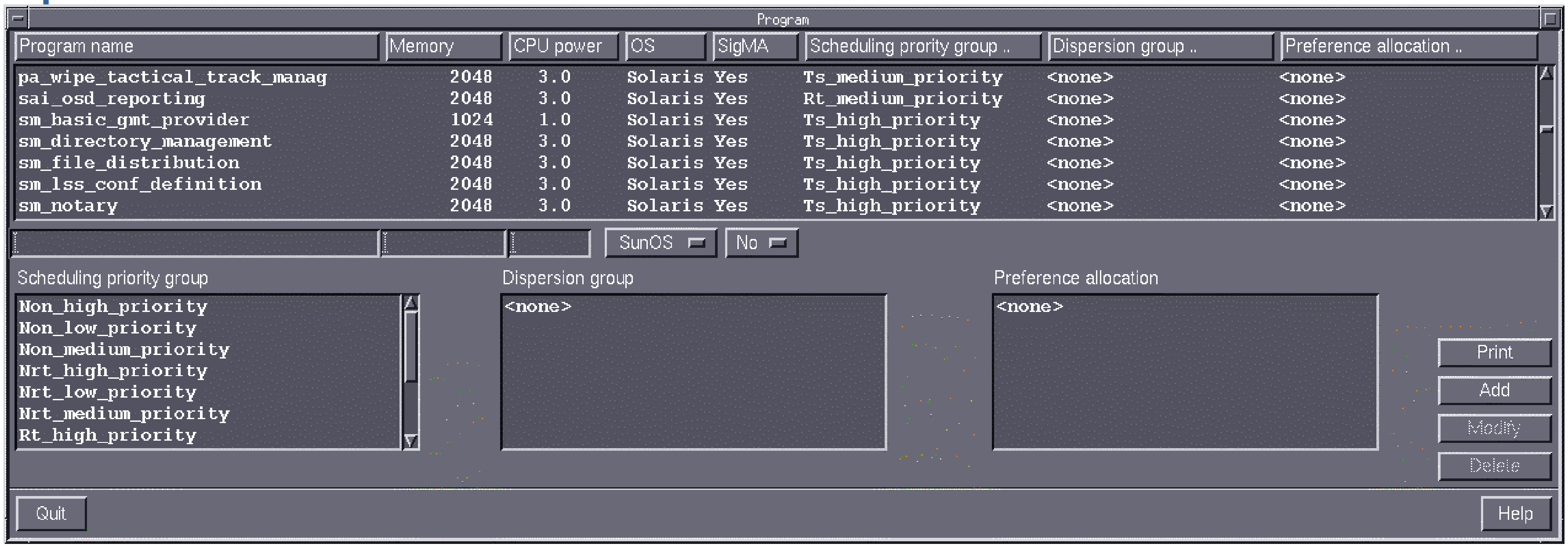
Data-traffic:	***4.000 publications per second over the system-data bus***
Programs:	***2.200 programs allocated over 150 processors***
Accuracy:	***100 us. time-alignment accuracy within the distributed system***

CMS ARCHITECTURE: Starting points



Fault-tolerant: *High combat survivability & maintainability (no single-point-of-failure)*
Flexible: *Mission-based configuration, on-board training & simulation*
Evolvable: *Evolutionary upgrading based on COTS & Open Standards*
Scalable: *From patrol-boats up to destroyers*

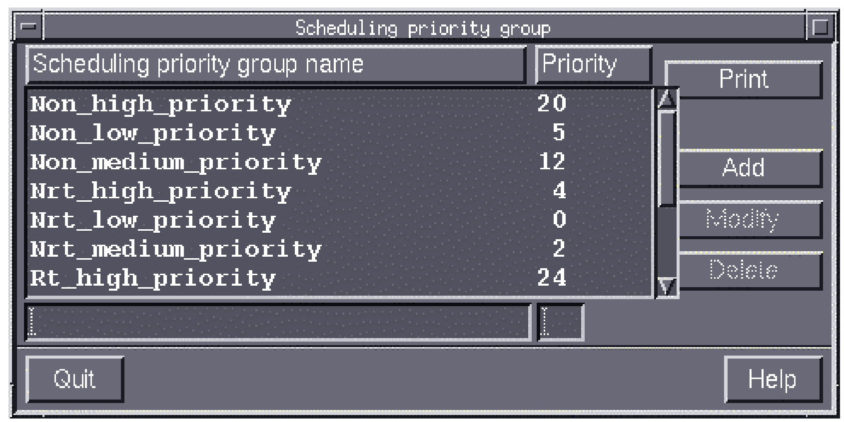
CMS: Dynamic Resource Management



Program name	Memory	CPU power	OS	SigMA	Scheduling prority group ..	Dispersion group ..	Preference allocation ..
pa_wipe_tactical_track_manag	2048	3.0	Solaris	Yes	Ts_medium_priority	<none>	<none>
sai_osd_reporting	2048	3.0	Solaris	Yes	Rt_medium_priority	<none>	<none>
sm_basic_gmt_provider	1024	1.0	Solaris	Yes	Ts_high_priority	<none>	<none>
sm_directory_management	2048	3.0	Solaris	Yes	Ts_high_priority	<none>	<none>
sm_file_distribution	2048	3.0	Solaris	Yes	Ts_high_priority	<none>	<none>
sm_lss_conf_definition	2048	3.0	Solaris	Yes	Ts_high_priority	<none>	<none>
sm_notary	2048	3.0	Solaris	Yes	Ts_high_priority	<none>	<none>

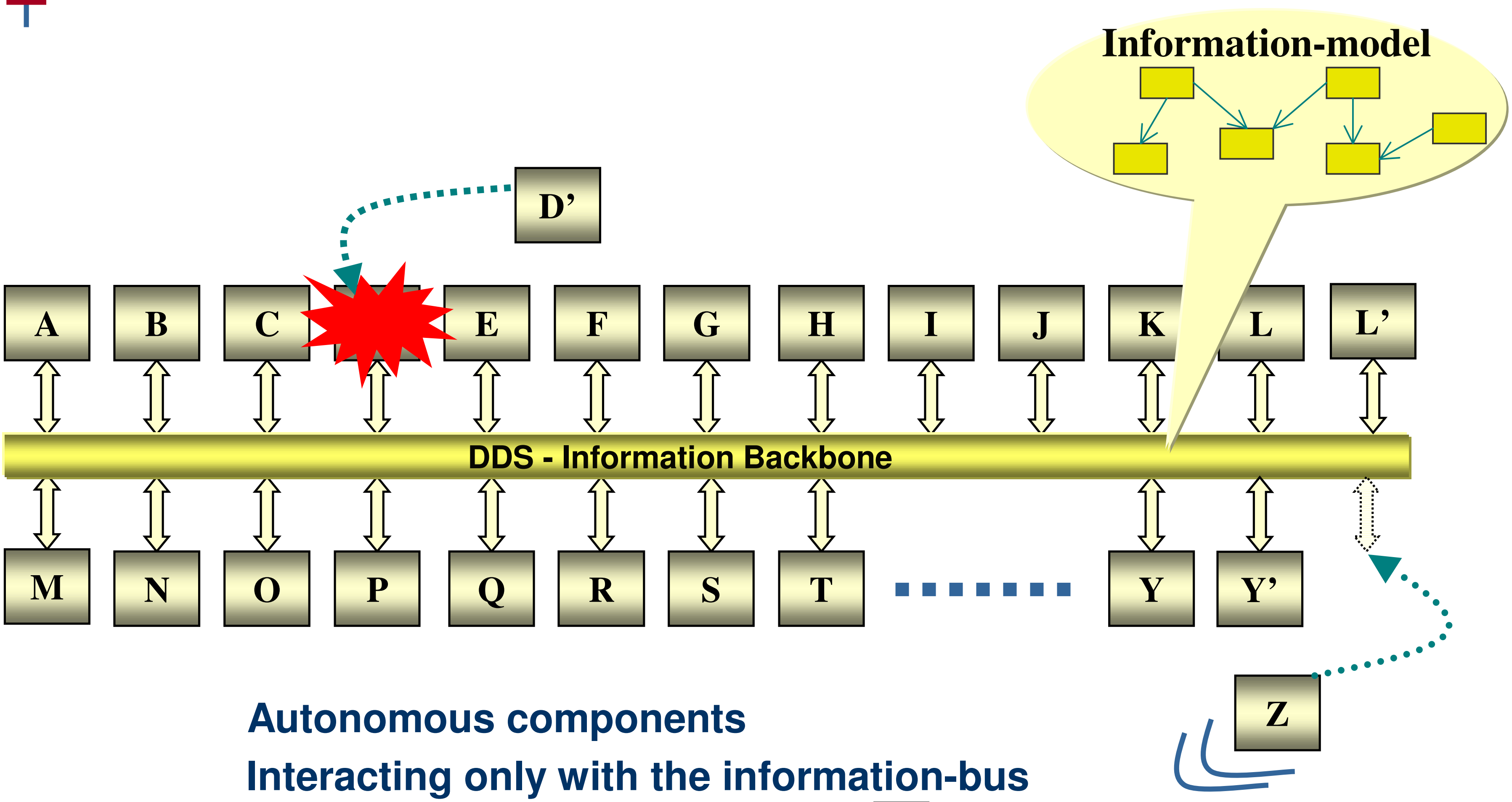
Scheduling priority group	Dispersion group	Preference allocation
Non_high_priority	<none>	<none>
Non_low_priority		
Non_medium_priority		
Nrt_high_priority		
Nrt_low_priority		
Nrt_medium_priority		
Rt_high_priority		

- **'pool-of-computers' utilization (Allocation schemes)**
  - **Allocation driven by resource-needs and availability (CPU-power, Memory)**
  - **'Dispersion groups' to force geographical separation (battle-damage resistance)**
  - **'Preference allocation groups' to advice co-location (efficiency)**
  - **Degradation driven by 'Functional Priorities' (importance)**



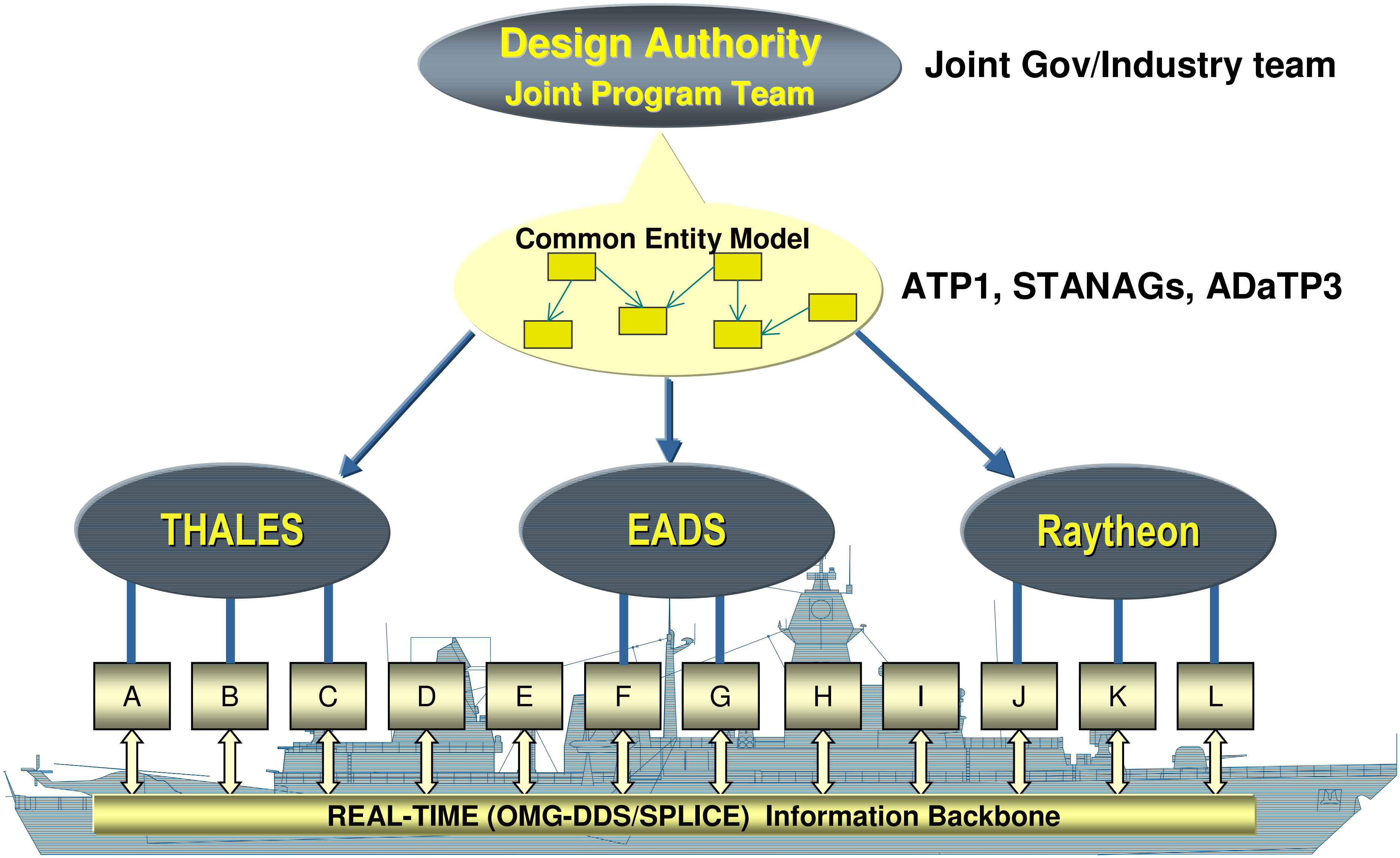
Scheduling priority group name	Priority
Non_high_priority	20
Non_low_priority	5
Non_medium_priority	12
Nrt_high_priority	4
Nrt_low_priority	0
Nrt_medium_priority	2
Rt_high_priority	24

INFORMATION-CENTRIC ARCHITECTURE

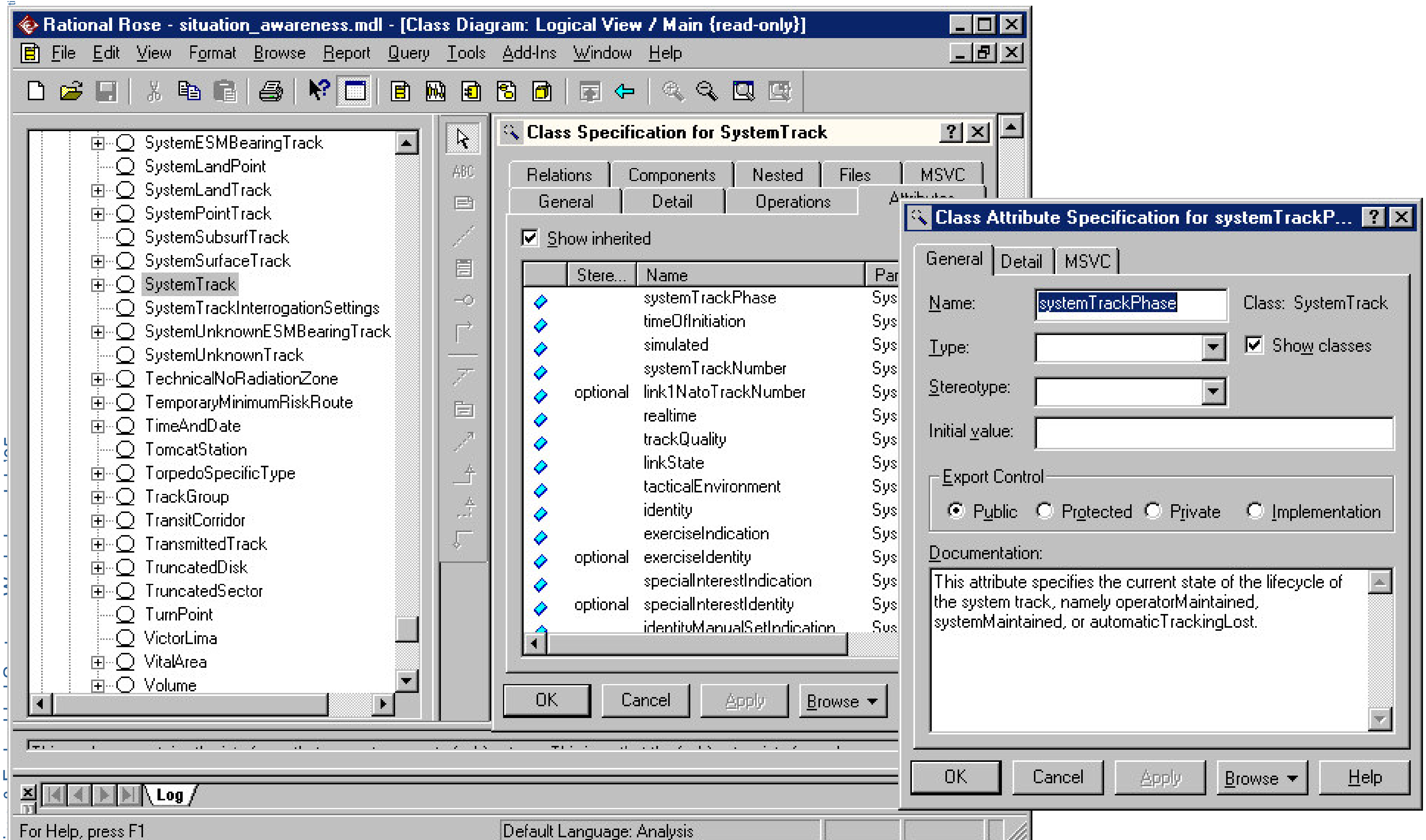


- Autonomous components
- Interacting only with the information-bus
- Spontaneous: Z, Self-healing: D’
- Redundant & Replicated: L’, Y’
- QOS-driven Data Distribution Service (reliability, persistency, latency): DDS

COMMON ENTITY MODEL : Deployment – F124



COMMON ENTITY MODEL : The “CMS Glossary”



THALES

Business Demands: *re-use, flexibility, scalability*

CHARACTERISTICS

Many different customers: *12 Navies world-wide use 1 CMS product-line*

Many different ships/missions: *20 Ships classes (from FPB's up to Destroyers)*

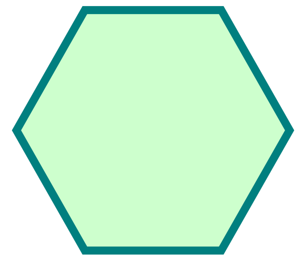
(3) ARCHITECTURE REQUIREMENTS

"Challenge your middleware …"

Summary: The Problem

Problem: *engineering (-cost) of distributed systems*

- too complex
- not reactive
- not future-proof
- not fault tolerant



Because ‘*multi-dimensional engineering’ is needed:*

What about *the current ‘state-of-the-art’?*

- architectures: message-passing, client/server, OO
- most efforts fall short in a number of dimensions:
- typically:
  - *limited performance* *(high-volume & low-latency balance)*
  - *exploding complexity* *(dependencies in many dimensions)*
  - *costly evolution* *(impact of changes & extensions)*

Architectural Requirements: Lifecycle focus

System design

- provide a stable basis to operate upon by applications
- enhance component autonomy
- allow transparent and global QoS assurance

System development

- reduce complexity and enhance re-usability
- provide shared/guaranteed properties
- small learning effort and flat learning curve

System integration

- support effortless component integration
- provide easy monitor & control
- shift ratio between design and integration effort

System deployment

- guaranty QoS for reliability, latency and persistency
- allow runtime migration of applications
- allow applications to join the system at any time

System maintenance & evolution

- allow runtime replacement and evolutionary upgrading
- support for logging & replay of information
- provide future-proof, re-usable, robust and scalable system

Proposal: An Information-Centric Approach

Towards a solution:

- make development effort more **simple**
- develop **less**
- develop solutions **only once**

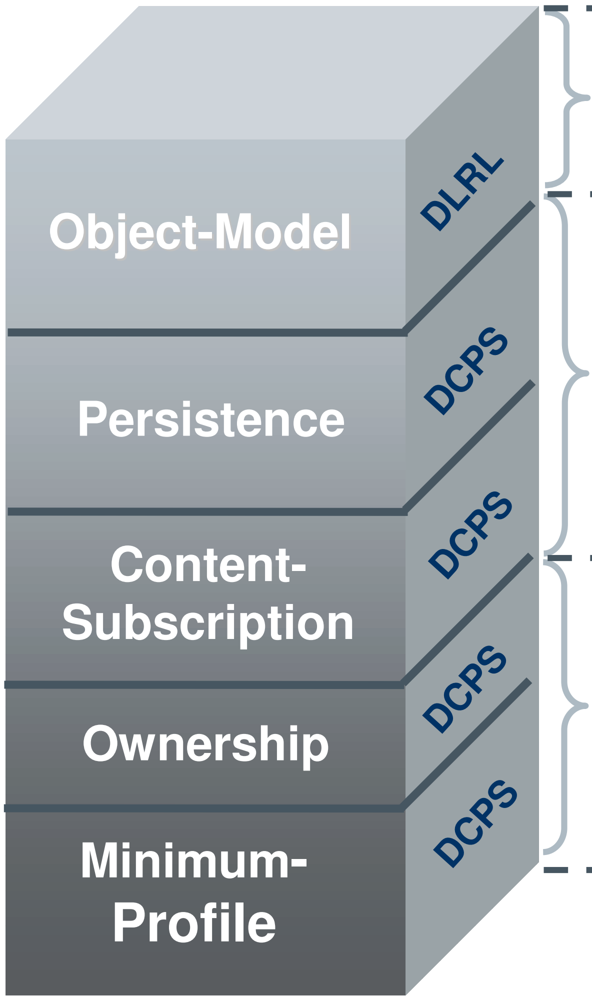
How:

- minimize component dependencies ***('simple')***
- maximize component autonomy ***('re-use')***
- normalize component interactions ***('only once')***

The clue:

- ***share*** the stable properties, ***localize*** the unstable ones
- ***information*** is what matters most, not how it is processed
- properly modeled data is ***stable***, processing often is not
- so ***focus on data*** first and then on the processing of it

OMG-DDS: A Data-Centric Solution



Object Orientated information view

- *Local object-model extending the distributed DCPS data-model*
- *Manages relationships and supports native language constructs*

Distributed QoS-driven information management

- *Fault tolerant and global persistence of selected data*
- *Guaranteed data availability supports application fault-tolerance*
- *Content-aware filtering and dynamic queries:*
  - *reducing application-complexity*
  - *improving system-performance*

Real-time pub/sub messaging:

- *Asynchronous 'one-to-many' real-time data communication*
- *Dynamic data-flow based on 'current-interest' (pub/sub)*
- *Platform independent data-model (IDL)*
- *Strong-typed interfaces for multiple languages*
- *Information Ownership management for replicated publishers*

(4) CMS USE-CASE

"DDS by example …"

INTRODUCING: THE EXAMPLE

SENSOR PROCESS

- Optical sensor
- Scans the environment
- Produces 'Tracks'
- Position of 'objects'
- Reports '***pointTrack***'

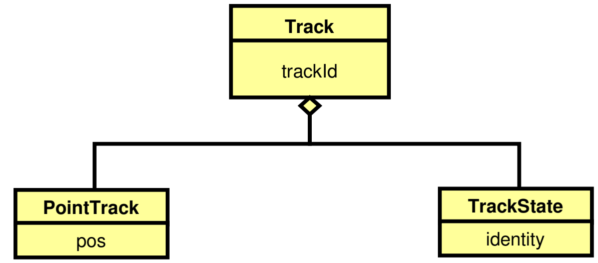
CLASSIFICATION PROCESS

- Classifies tracks
- Determines their identity
- Analyses the trajectories
- Determines hostility
- Reports '***trackState***'

DISPLAY PROCESS

- Displays track info
- Both position & identity
- Raises alerts
- Requires '***pointTrack***'
- Requires '***trackState***'

DATA DEFINITION



- Information modeled as "**TOPICS**"
  - Basic units of information
  - Individually produced and/or consumed
  - Support autonomous & decoupled programs
- Each **TOPIC** has an associated name and data type
  - Data-definition in IDL
  - 'Key' fields for unique identification
  - Relational Data Model (keys)
- Our example:

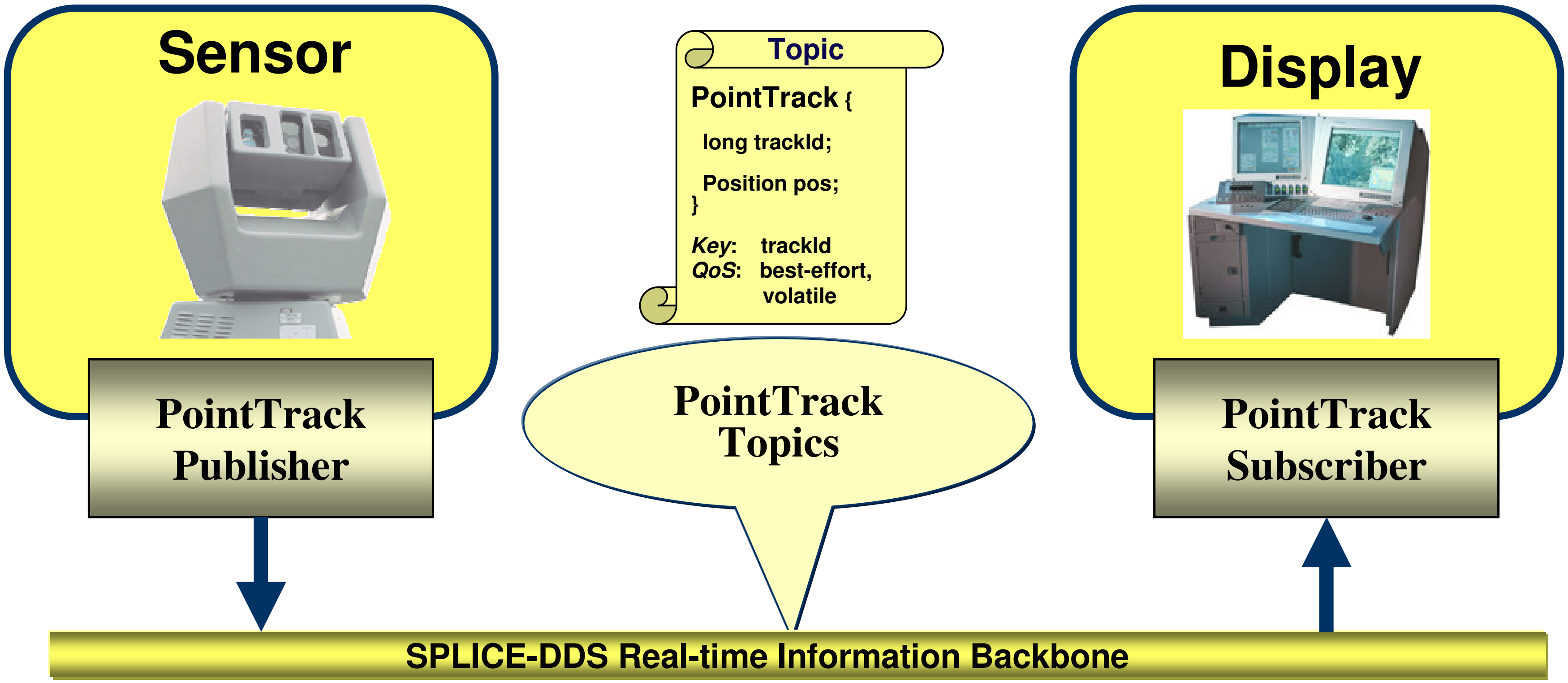
*Topic "**PointTrack**"*

```
Struct PointTrackType {
  long trackId;         // key

  Position pos;
}
```

*Topic "**TrackState**"*

```
Struct TrackStateType {
  long trackId;         // key

  Id identity;
}
```

THE MINIMUM PROFILE



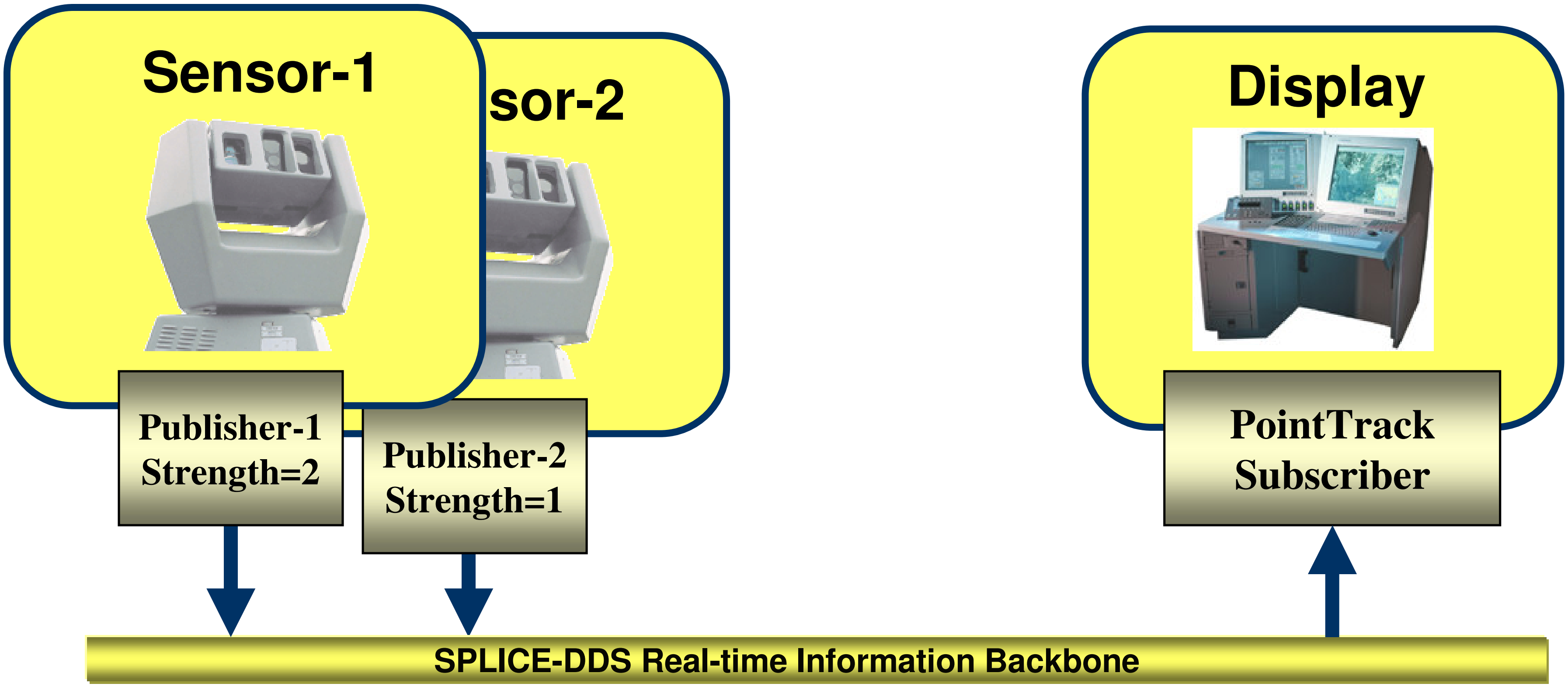
Characteristics

- Basic publish/subscribe data distribution
- Topics (types) specified in IDL
- QoS regarding: reliability, urgency & lifeliness

Features / Advantages

- Autonomous & loosely coupled applications
- Pub/Sub & QoS driven communication
- Strong-typed interfaces
- Smart networking based on priority & latency-budget

THE OWNERSHIP PROFILE



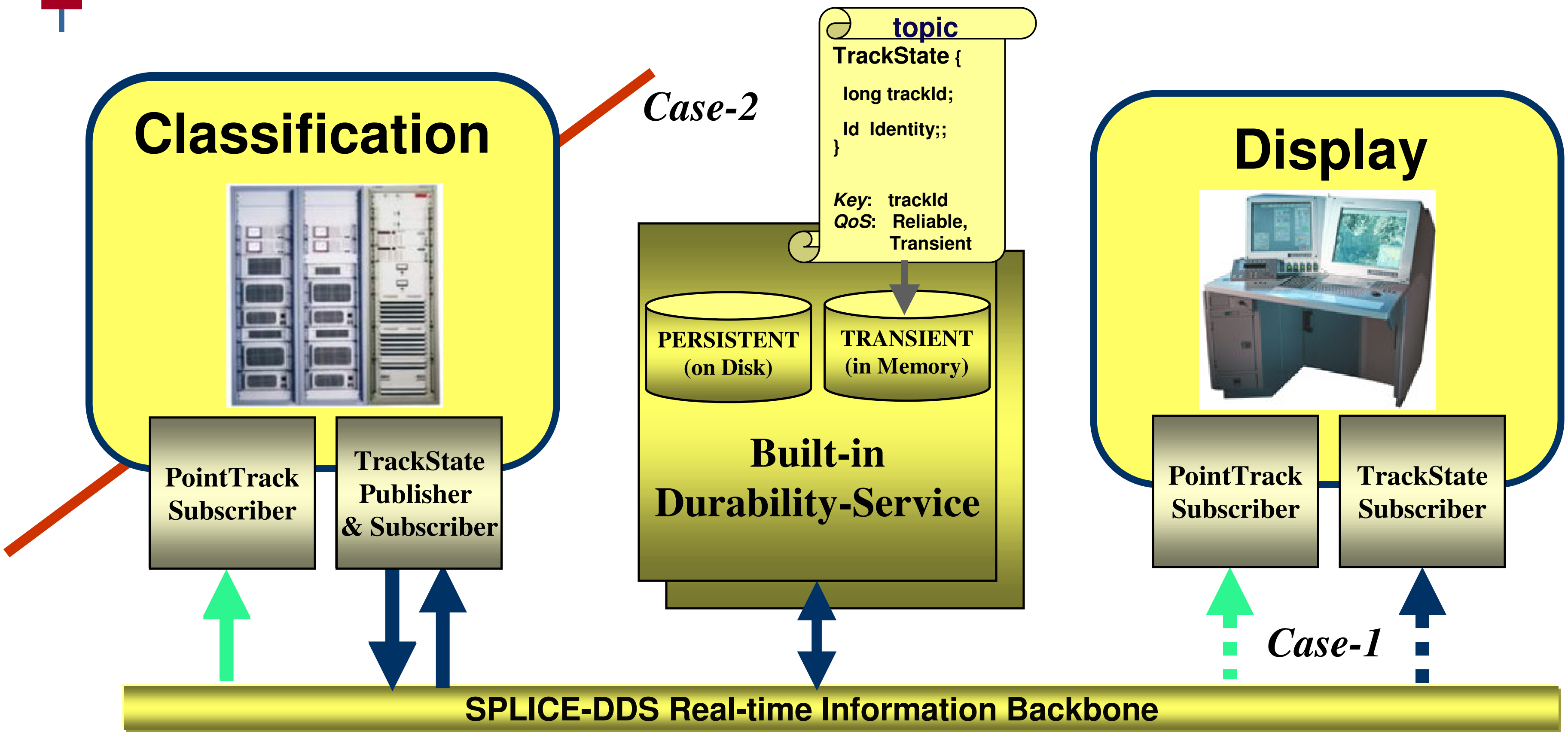
Characteristics

- Replicated publishers of data (with own 'strength)
- Only highest-strength will be received
- On failure, next highest-strength will 'take-over'

Features / Advantages

- Fault-tolerance by replication
- Notes:
  - Requires a lot of resources
  - Quality must be expressible as an 'integer'

THE PERSISTENCE PROFILE



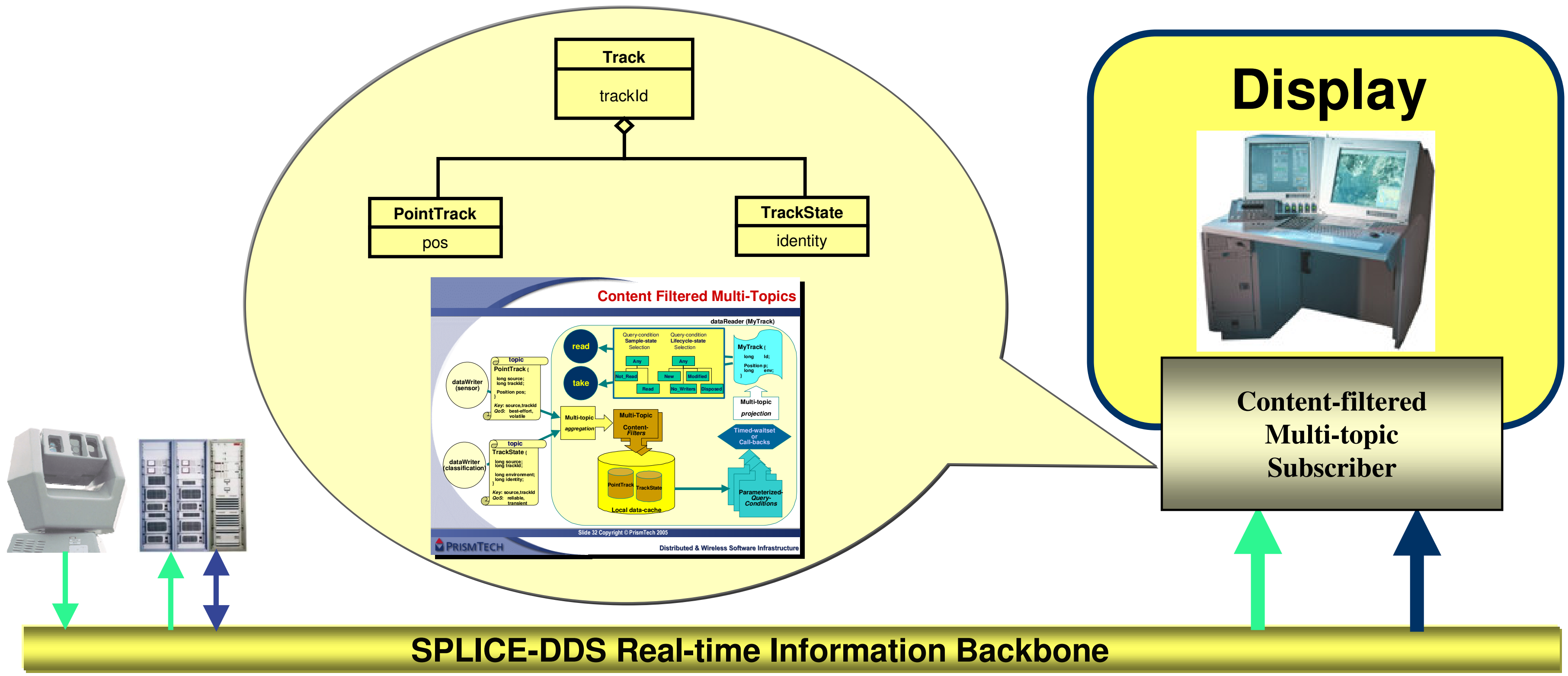
Characteristics

- Built-in persistence for non-volatile data
  - State preservation for transient publishers
  - Settings persistence surviving system downtime
- Replicated durability service for maximal fault-tolerance

Features / Advantages

- **Case-1**: late-joining of Display process
  - Previously produced TrackStates readily available
- **Case-2**: restart of failed Classification process
  - Internal state (already classified tracks) regained

THE CONTENT-SUBSCRIPTION PROFILE (1)



Characteristics

- Adds ‘content awareness’
  - Content-filtered Topics & query-conditions
- Supports ‘compound interest’
  - Multi-topics (combine/filter/re-arrange topics)

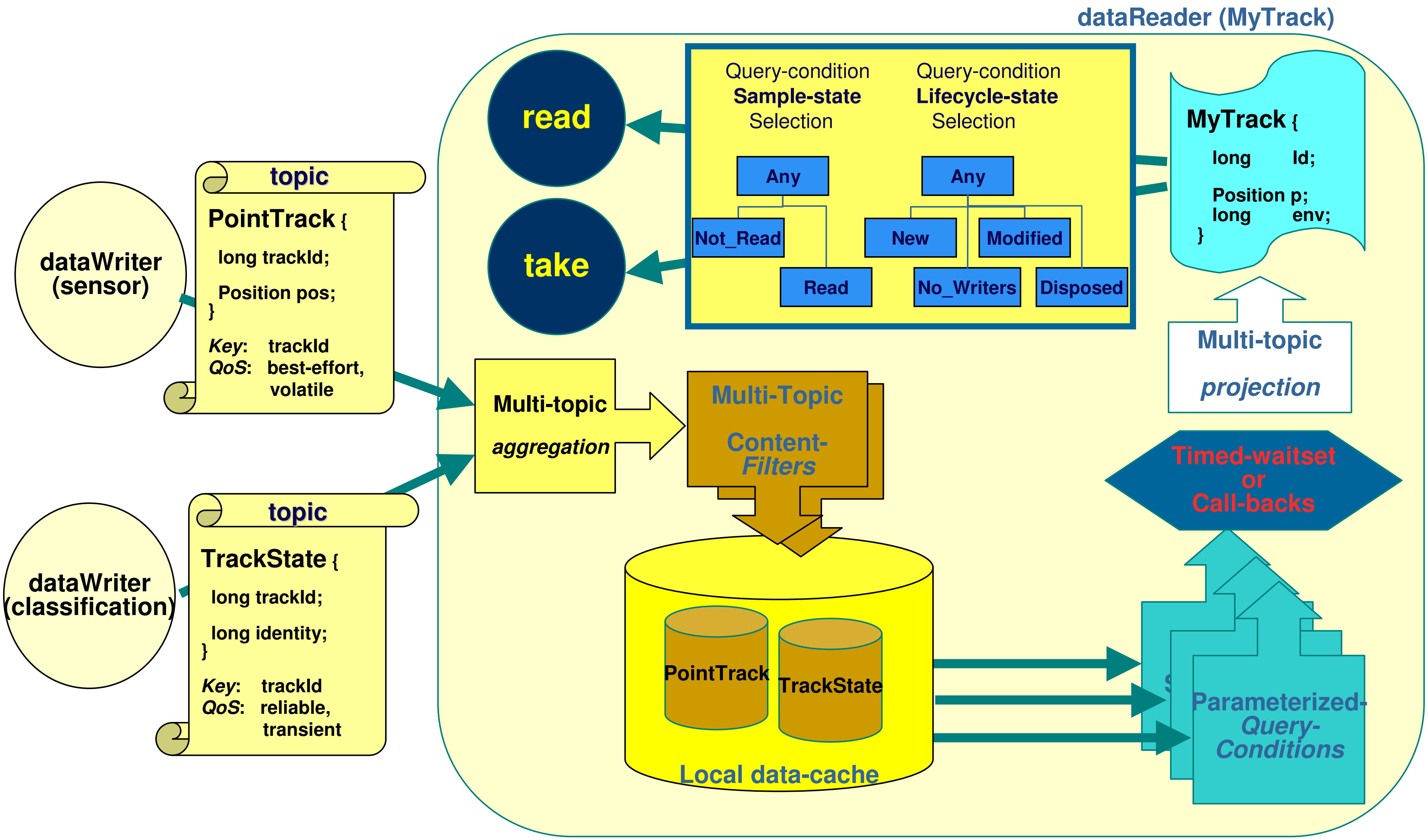
Features / Advantages

- Reduced application complexity
  - Automatic ‘reconstitution’ (join) of related data
- Improved system performance
  - Only receive/process what is of interest

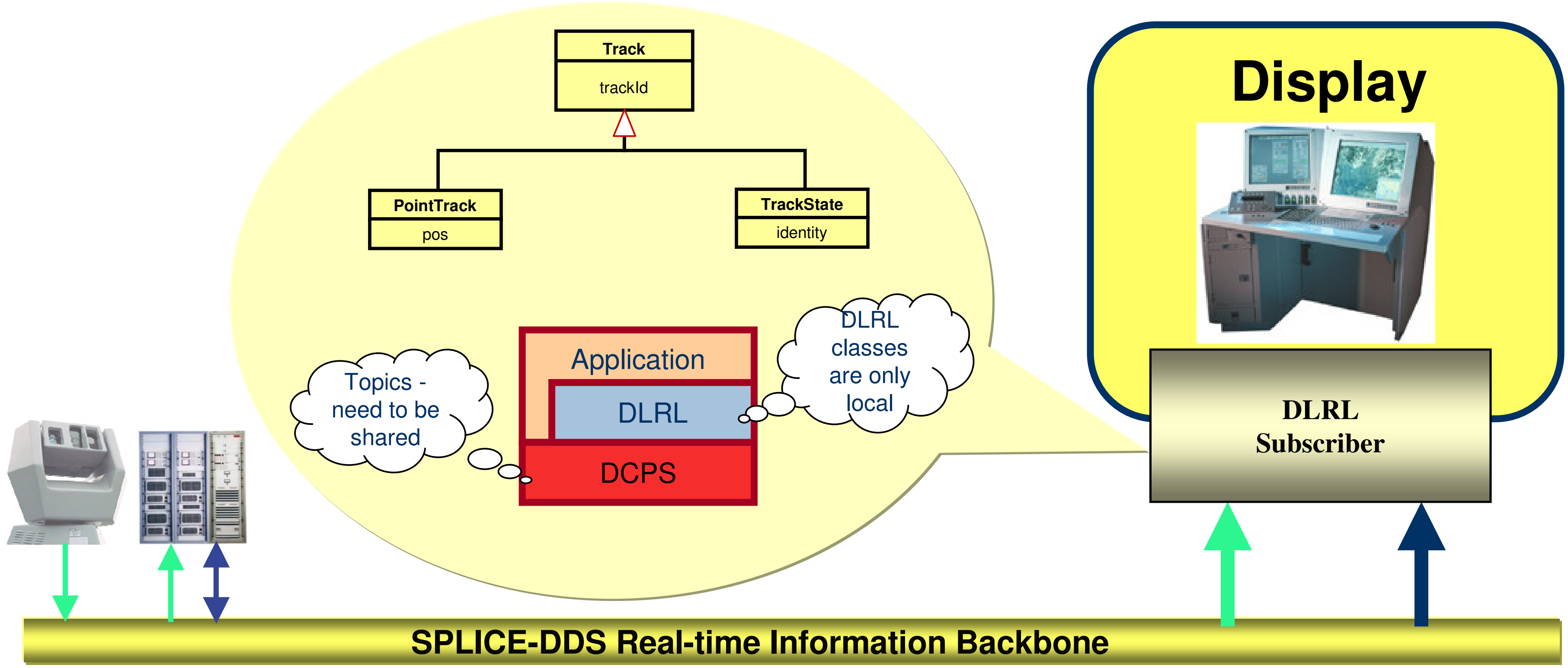
THALES

THE CONTENT-SUBSCRIPTION PROFILE (2)

Content Filtered Multi-Topics



THE OBJECT-MODEL PROFILE



Characteristics

- **Local** Object Oriented Data-Access Layer
- Supports 'OO' features:
  - Inheritance, aggregation, composition
- Uses DCPS to distribute **state** by 'mapped topics'

Features / Advantages

- Ease of Management of (related) data
  - Object oriented 'graphs of objects' (value-types)
- Supports 'native language constructs' (I.e. navigation)
  - Automatic 'change-management' of objects

THALES

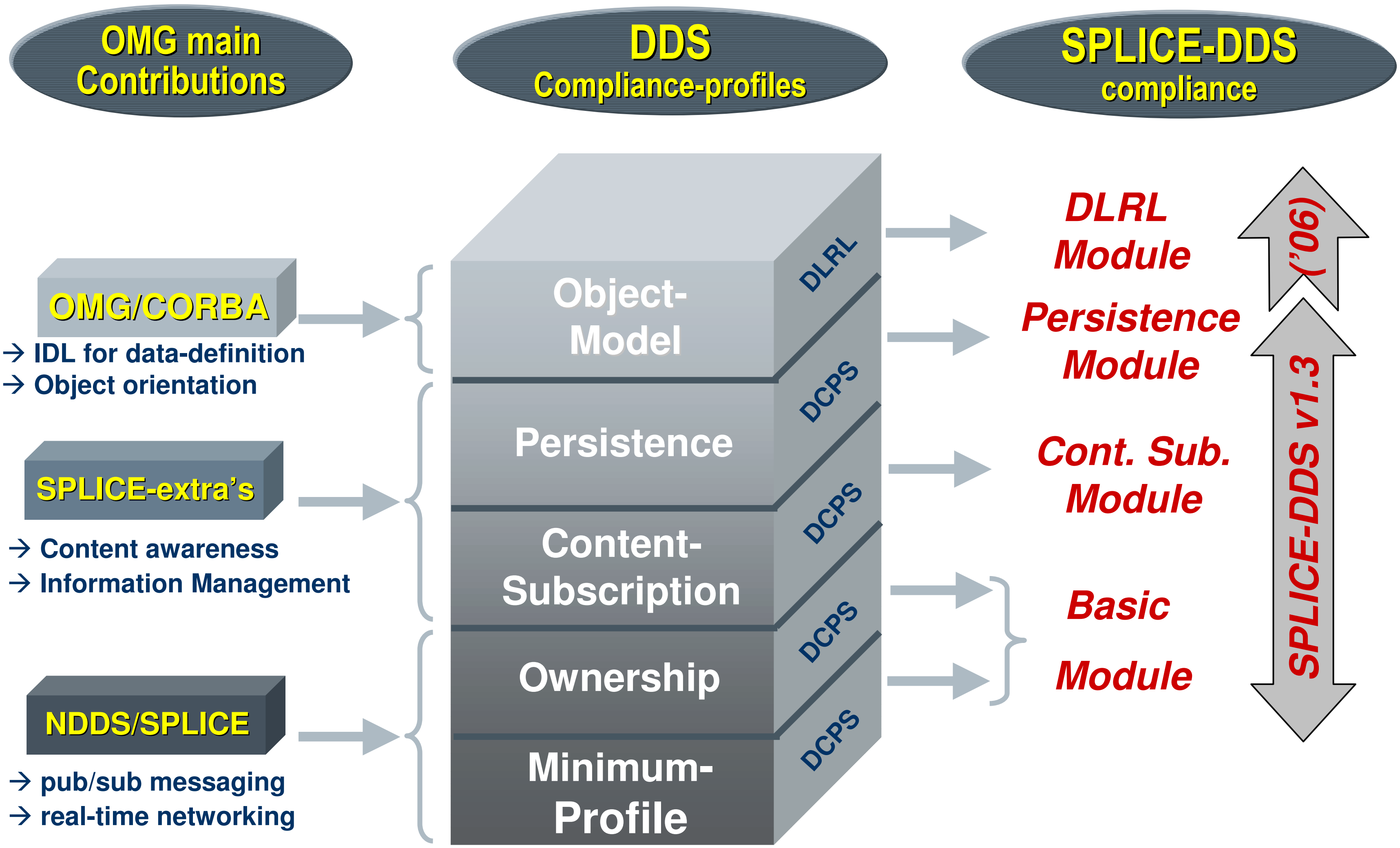
(5) CONCLUSION

"Proven suitability…"

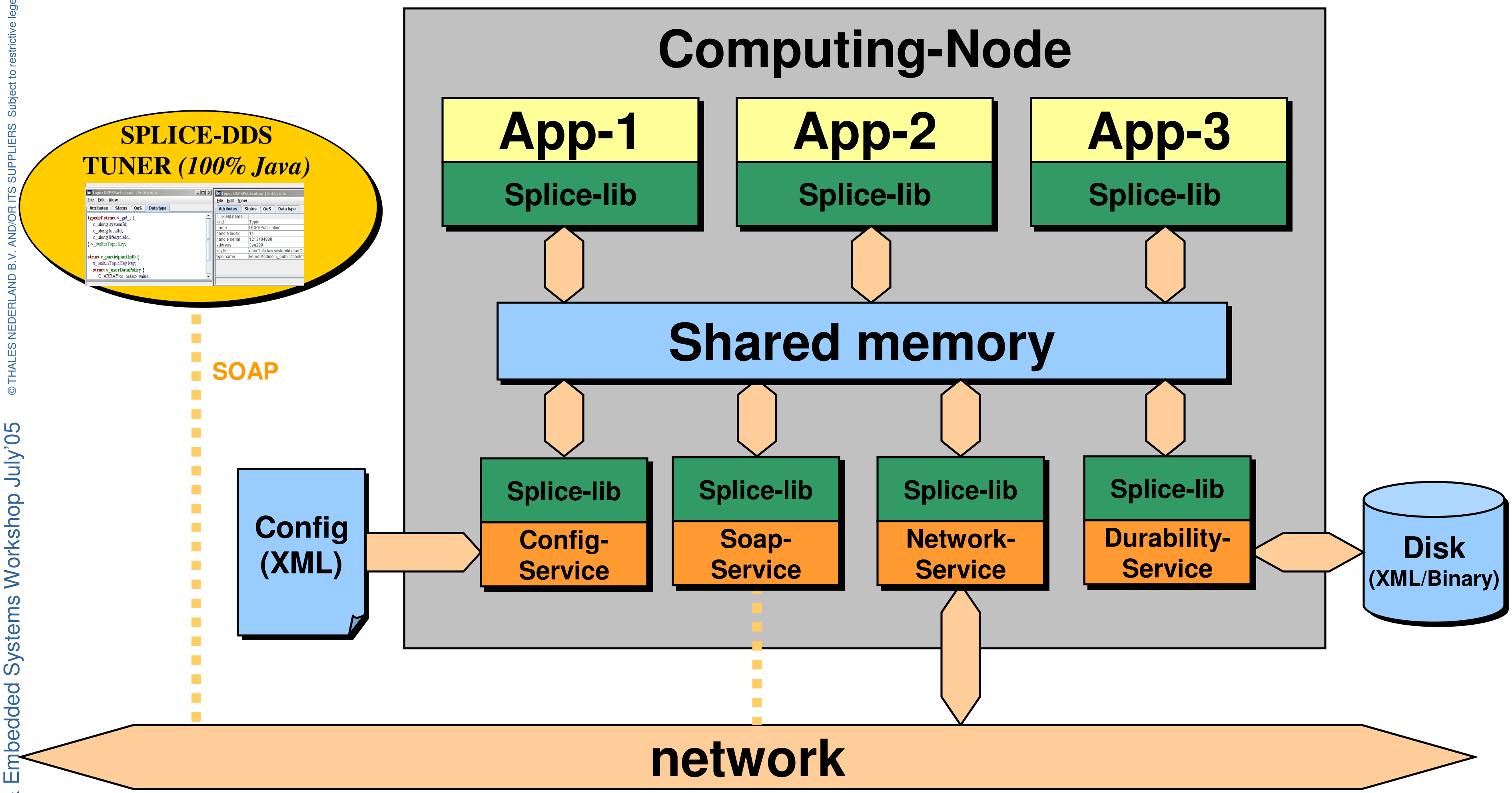
SUMMARY: Requirements & Realization

Requirement:	Realized by:
System design	
■ provide a stable basis to operate upon by applications	*- shared Information Model*
■ enhance component autonomy	*- state-based information-centric system*
■ allow transparent and global QoS assurance	*- Information classification (QoS topic-defaults)*
System development	
■ reduce complexity and enhance re-usability	*- minimized component dependencies*
■ provide shared/guaranteed properties	*- standardized (DDS-) interaction-environment*
■ small learning effort and flat learning curve	*- intuitive concept, simple/powerful features*
System integration	
■ support effortless component integration	*- maximized component autonomy*
■ provide easy monitor & control	*- globally accessible information (data+metadata)*
■ shift ratio between design and integration effort	*- focus on info-model & decoupled applications*
System deployment	
■ guaranty QoS for reliability, latency and persistency	*- realtime "DDS" information backbone*
■ allow runtime migration of applications	*- global & FT availability of transient state data*
■ allow applications to join the system at any time	*- dynamic discovery and data persistence*
System maintenance & evolution	
■ allow runtime replacement and evolutionary upgrading	*- de-coupled & autonomous components*
■ support for logging & replay of information	*- global availability of all (time-stamped) data*
■ provide future-proof, re-usable, robust and scalable system	*- highly adaptive associative data-model*

SPLICE-DDS *"Proven & Complete DDS implementation"*



SPLICE-DDS: *Pluggable Service Architecture*



SPLICE-DDS, a FULL OMG-DDS Implementation

- **SPLICE-DDS**, developed by **TNL**, Marketed & Supported by **PrismTech**
  - SPLICE-DDS v1.3 supports **ALL** DCPS profiles
  - Check http://www.prismtech.com for product-details and SPLICE-DDS webcast

- **PrismTech** brings its wide experience as a middleware ISV for the marketing and support SPLICE-DDS worldwide:
  - Product evaluation
  - Ports and customisations to customer's special environments
  - Support & training

- **Thales Netherlands (TNL)** is in charge of product evolutions, with a dedicated team of over 15 experienced engineers and a long term commitment to:
  - Continued contribution to OMG DDS specification process
  - Committed road map for full implementation of the DDS standard
  - Expert level consulting

THALES

“DDS SAILS THE SEVEN SEAS”

"The END"

QUESTIONS ?????