Selecting Middleware Technologies

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Tricia Oberndorf, Tom Merendino,
Soumya Simanta
SSTC - April 2010

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

Outline

- **The Selection Problem**
- The Selection Process
- Example Use
- Results and Lessons Learned

The Customer Problem

Our customer has had an open, COTS-based architecture in place for over 10 years:

- Basic system function is to connect data providers with data consumers
- Middleware technology was used to achieve hardware independence
  - Hardware upgrades have been successfully achieved – the scheme works
- Current technology: CORBA

But 10 years is very long in technology time

- Should the program replace the current middleware technology?
  - Consider both a 5-year and a 10-year timeframe
- If so:
  - when?
  - what should the replacement be?
  - how should the program go about doing the replacement?

Outline

The Selection Problem

The Selection Process

Example Use

Results and Lessons Learned

Our Basic Evaluation Process

The SEI employs a basic evaluation process when addressing situations like this:

- **P**lan the evaluation
- **E**stablish criteria
- **C**ollect the data
- **A**nalyze the data

Process Used for this Study

Planning includes:

- Develop understanding of system and system context
- Develop understanding of system architecture

Establishing criteria includes:

- Conversion of key requirements to criteria
- Prioritization of those criteria with customer team

Collecting data includes:

- Market survey to determine the state of the standards and availability of products for the timeframes
  - Required “big picture” of the technology area (middleware)

Analyzing data includes:

- Includes Cost/Performance Benefit Analysis to determine whether the CORBA standard should be replaced and, if so, with what
- Also when and how

Outline

The Selection Problem

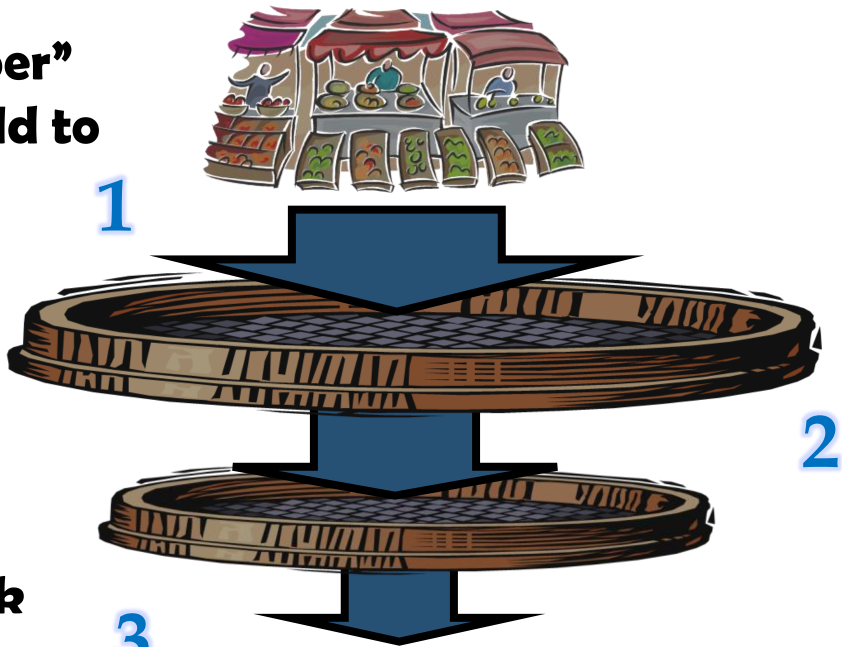
The Selection Process

Example Use

Results and Lessons Learned

Approach: Filter Followed by Deeper Evaluation

Middleware technologies for event-based, soft real-time, distributed systems



Initial "showstopper" criteria reduce field to a few for deeper evaluation

1

2

Broader, richer set of criteria provide basis for deeper evaluation

Cost/Benefit/Risk analysis of deeper evaluation results

3

Recommendations

System Attributes

Federated, loosely coupled subsystems

Data centric

- Based on common data groups
- Primary function is data distribution
- No transactions
- Data mostly transient

Soft real time

Uses both event-based (publish/subscribe) and data movement-based (client /server) interaction patterns

- Will continue to need support for both

Limited/no sharing of subsystem resources

Key quality attributes: performance, fault tolerance, security

Subsystem Attributes

Each subsystem:

- Internally uses its own middleware implementation(s)
- Has localized the CORBA code
  - In most cases, changes to middleware implementation will be localized to one place in the subsystem
- Is independent of the middleware implementation used (apart from localized CORBA-related code)
- May use non-CORBA interactions with other subsystems

Note: Subsystems are out of scope. However, this information can be helpful and may be used in final decisions.

System Context Characteristics

Controlled and closed development and run-time environments

Acknowledged system of systems (SoS) at development time

All decisions are made at design/development time

Environment is not dynamically changing

Prevailing attitude and approach of the program is very conducive to continued success.

Filter Criteria

The criteria we have used for this filter are mostly a subset of the full set we will use for the detailed evaluation:

- Whether the technology is platform-specific
- Whether the technology is language-specific
- Whether the technology is standards-based
- Whether the technology is vendor-specific
- Whether the technology has support for events
- The maturity of the technology
- The breadth of adoption of the technology

Adopted a color coding scheme for summary comparison.

Distributed Systems – Levels of Abstraction -1

Vendor specificity increases

Abstraction decreases

Concept : Architectural Styles, Design Elements, Interaction Patterns

Central database – *data* centric
Cloud computing & Grid Computing – *resource* centric
Distributed Objects Architecture (DOA) – *object* centric
Event-Driven Architecture (EDA) – *event* centric
Message Oriented Middleware (MOM) – *message* centric
Remote Procedure Call (RPC) – *function* centric
Service Oriented Architecture (SOA) – *service* centric

Specification – Standards

examples

CORBA WS-* DDS DCOM RMI JMS EJB

Implementation – Types

Enterprise Service Bus (ESB) Application Server ORB Message bus

Implementation – Specific Software Products

Orbix ORB Websphere RTI DDS TIBCO RV JBoss Weblogic

Distributed Systems – Levels of Abstraction -2

Architectural concepts are not disjoint.

- EDA can be implemented using services and SOA can have events
- SOA uses messages and MOM can use services
- Both Grid and Cloud use web services

Each architectural pattern can be implemented using different technologies.

- Distributed Objects Architecture can be implemented using CORBA or EJB.

The architecture of the system is often a *hybrid.*

Filter Results

	Technology	Platform specific	Language specific	Standards based	Vendor specific	Support for Events	Maturity	Adoption
●	Java-RMI							
	JMS							
	Web Services							
●	EJB							
●	CORBA							
●	DCOM							
	DDS							
	Tibco RV							
●	Facebook Thrift							
	IBM MQSeries							
●	Protocol Buffers							
●	Cisco Etch							
●	ZeroC ICE							
	Elvin							
	AMQP							

● Technologies that are very similar to CORBA

Conclusions from Filter

Based on these results, we performed a deeper evaluation on:

- CORBA – as currently implemented and used in the system
- Web Services (WS-*) – taking into account the things that “most” implementations do or are found in the most popular implementations
- DDS – popular implementations (primarily RTI, PrismTech)

In the detailed evaluation, we did not reconsider these same criteria in the binary sense.

- In many cases, however, the detailed look examined the *degree* to which a particular technology satisfies the criterion and how it does it.

Final Criteria Considered -1

Alphabetical

Architectural Styles Supported	Support for key communication styles
	Support for events
Enterprise Architecture Alignment	Navy OA
	DoD and other enterprise
Environment	Support for heterogeneity
	Support for development-time federation
Future	Education
	Marketplace trend
	State of research on the technology
Implementations	Availability of open source implementations
	Strength in marketplace
	Complexity of implementation
	Quality of available implementations

Final Criteria Considered -2

Alphabetical

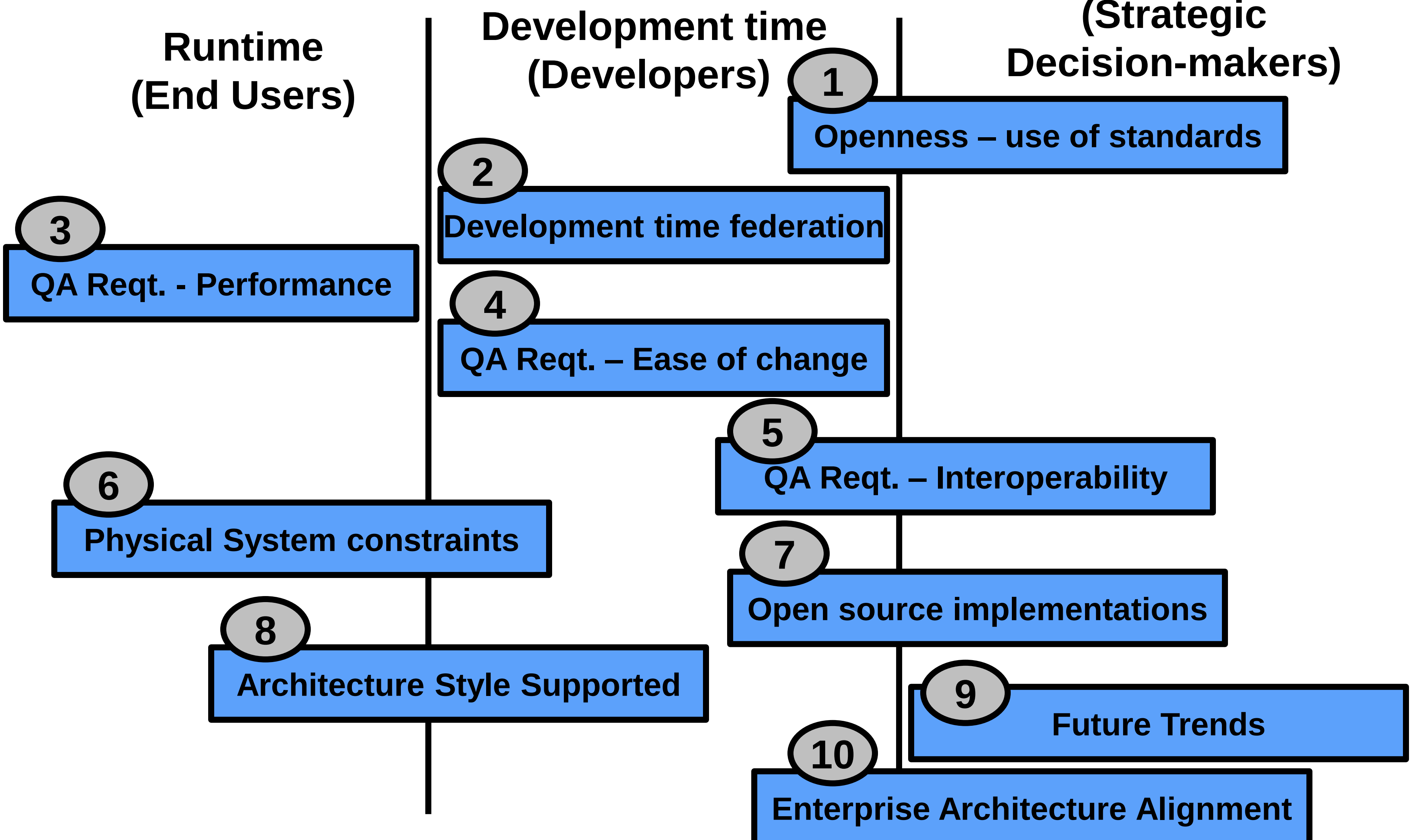
Openness	Use of standards
Quality Attribute Requirements	Availability
	Interoperability
	Performance
	Scalability
	Support for audit
	Support for data consistency
	Support for security
	Upgradability/maintainability
Reuse	Preservation of legacy infrastructure logic/concepts
	Future reuse (factoring)
System Constraints	Impact on physical system constraints

Top Ten Priorities

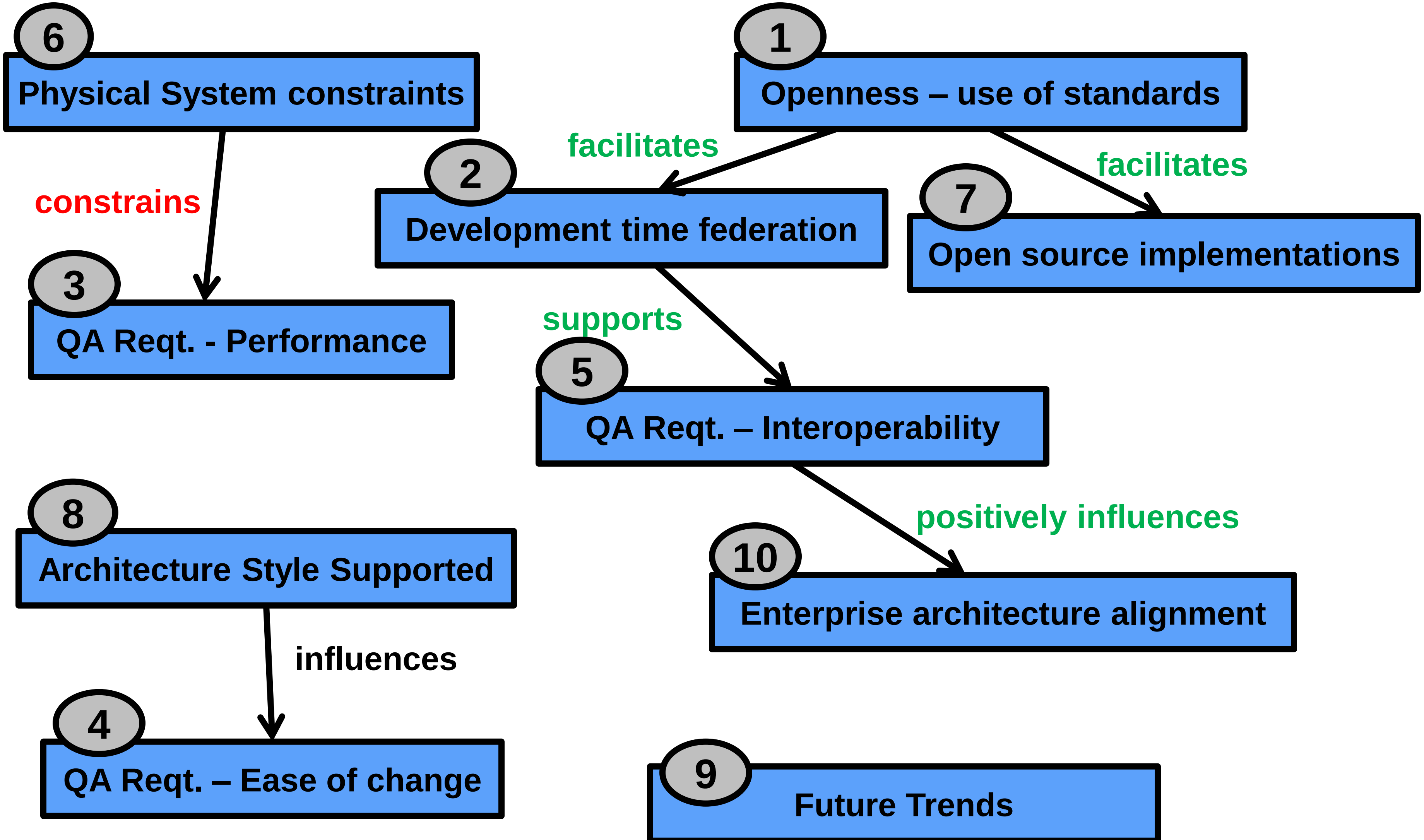
Ordered

Openness	Use of standards - Pedigree
Environment	Support for development-time federation
Quality Attribute Requirements	Performance
Quality Attribute Requirements	Upgradability/maintainability – ease of change
Quality Attribute Requirements	Interoperability
System Constraints	Impact on physical system constraints
Implementations	Availability of open source implementations
Architectural Styles Supported	
Future	Marketplace trend
Enterprise Architecture Alignment	Navy OA

Top Ten Criteria Grouped



Relationships Between Top Ten Criteria



DDS Overview

Design Principles and Key Features

- ❑ Data-centric
- ❑ Loose coupling in space and time
- ❑ Key quality attributes - Performance, reliability, and scalability
- ❑ Configurable QoS parameters

Architecture

- ❑ Decentralized peer-to-peer architecture
- ❑ Publish/subscribe many-to-many asynchronous communication model

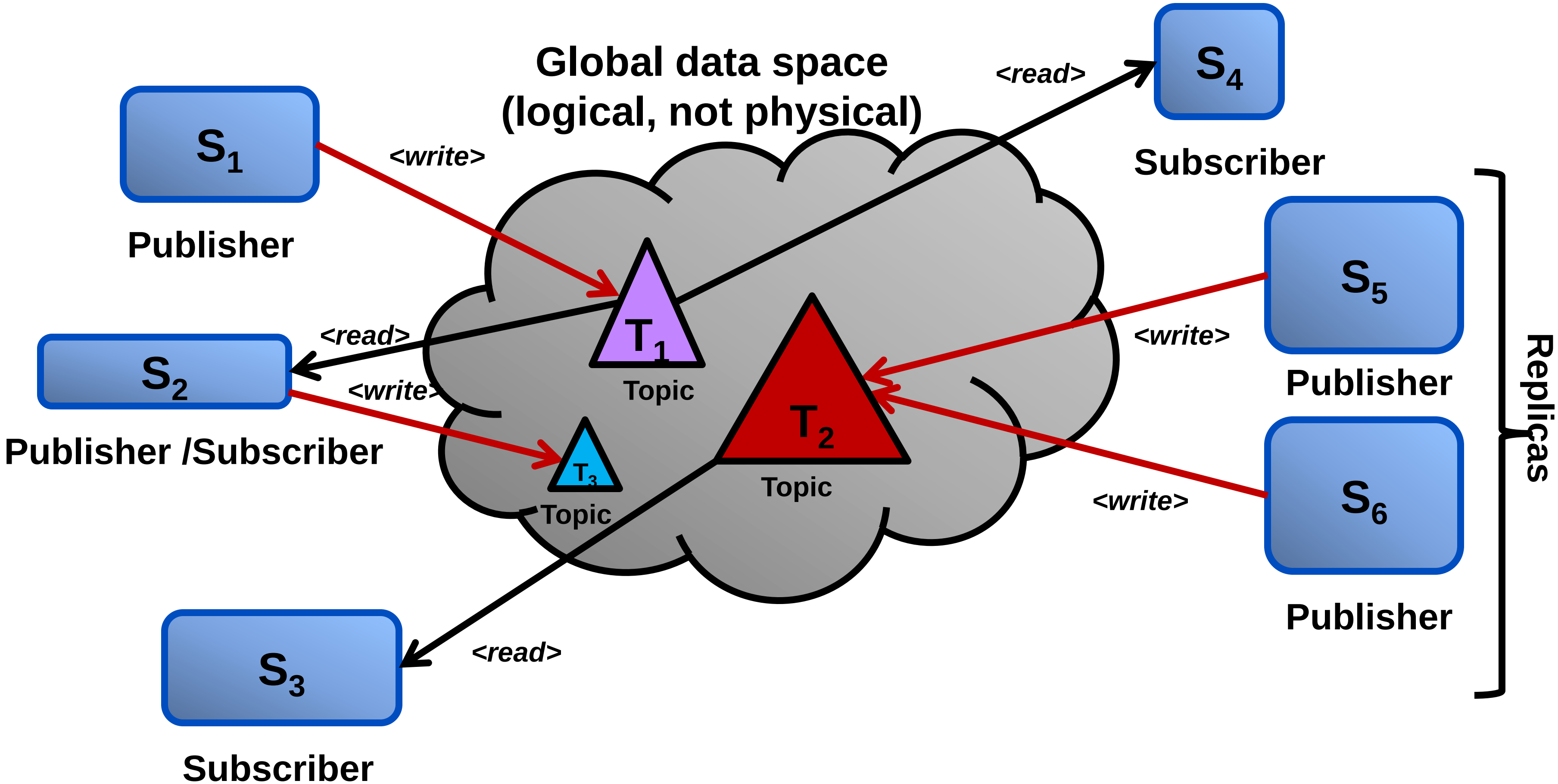
Target Systems and Environments

- ❑ Designed for distributed low latency and high reliability systems
  - ➢ Especially ideal for high-performance distribution of event-based messages
- ❑ Commercial and open source implementations available

Standards

- ❑ A set of standards by Object Management Group (OMG)
- ❑ Joint submission by RTI , THALES and OIS and Version 1 adopted in 2004

DDS – Core Concepts



Web Services Overview

Design Principles and Key Features

- ❑ XML-based message centric
- ❑ Loose coupling in space, time (optional)
- ❑ Autonomous
- ❑ Key quality attributes - interoperability, composability, reusability

Architecture

- ❑ Client/server and/or centralized bus architecture
- ❑ Synchronous and asynchronous communication model

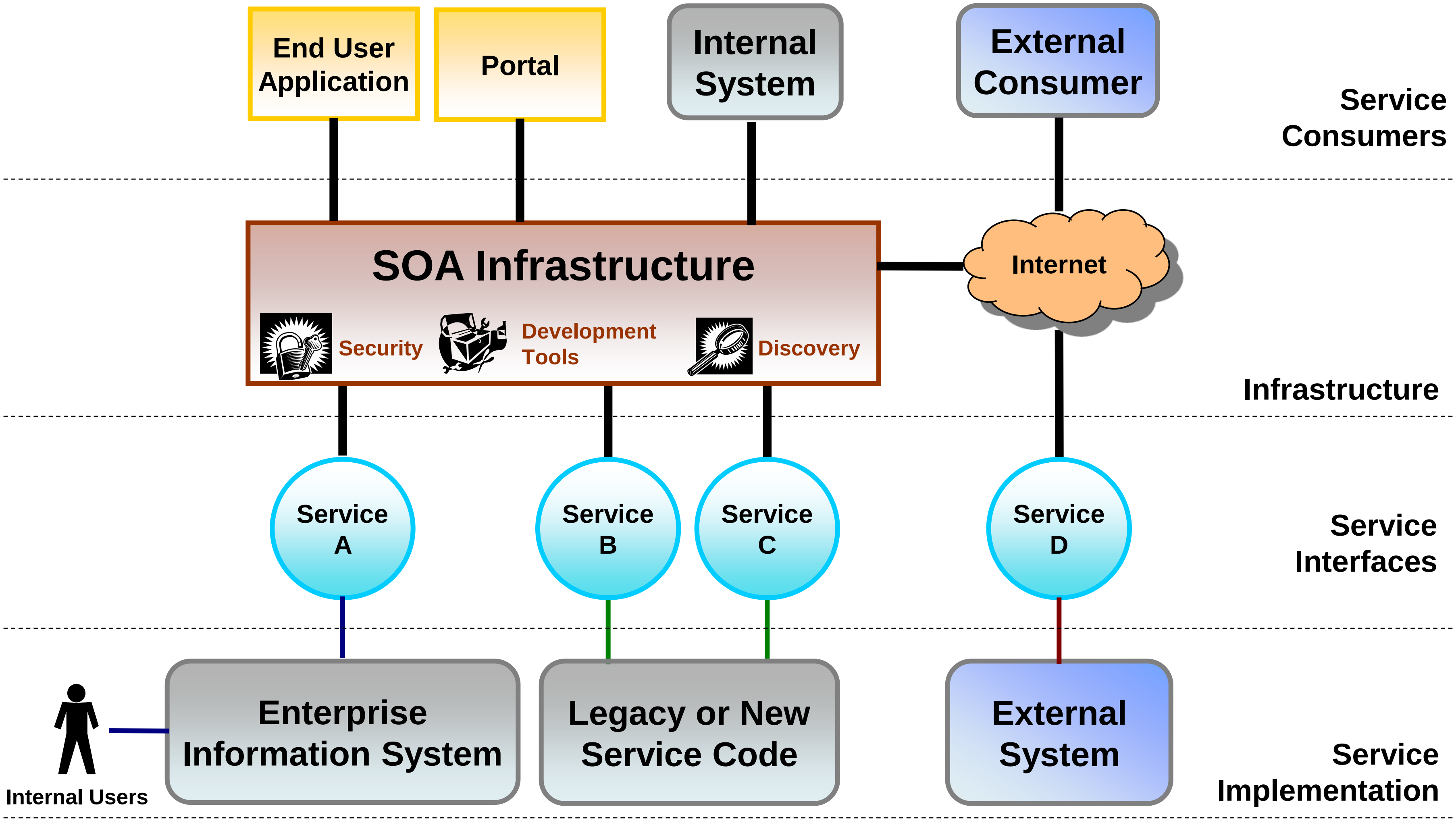
Target Systems and Environments

- ❑ Designed for integrating heterogeneous enterprise systems
- ❑ Widely adopted by web -based systems on the Internet
- ❑ Multiple implementations
- ❑ A strong and growing open source community

Standards

- ❑ Loosely coupled interoperable standards, mostly by W3C and OASIS
- ❑ Basic standards and optional standards addressing specific areas – security, transactions, reliability

Web Services – SOA Core Concepts



CORBA Overview

Design Principles and Key Features

- Distributed Object Middleware
- Distributed objects appear to be local
- Support for heterogeneous environments
- The standard provides definition of system services (e.g., Event Channel Service, Naming Service, etc.)

Architecture

- Client/Server pattern, supporting Remote Procedure Call-like interactions
- Event Channel w / Push-Pull interactions
- Object Request Brokers (ORBs) – in charge of component interactions
- Protocols for inter-ORB interaction
- Standardized Interface Definition Language (IDL) for interface publication

Target Systems and Environments

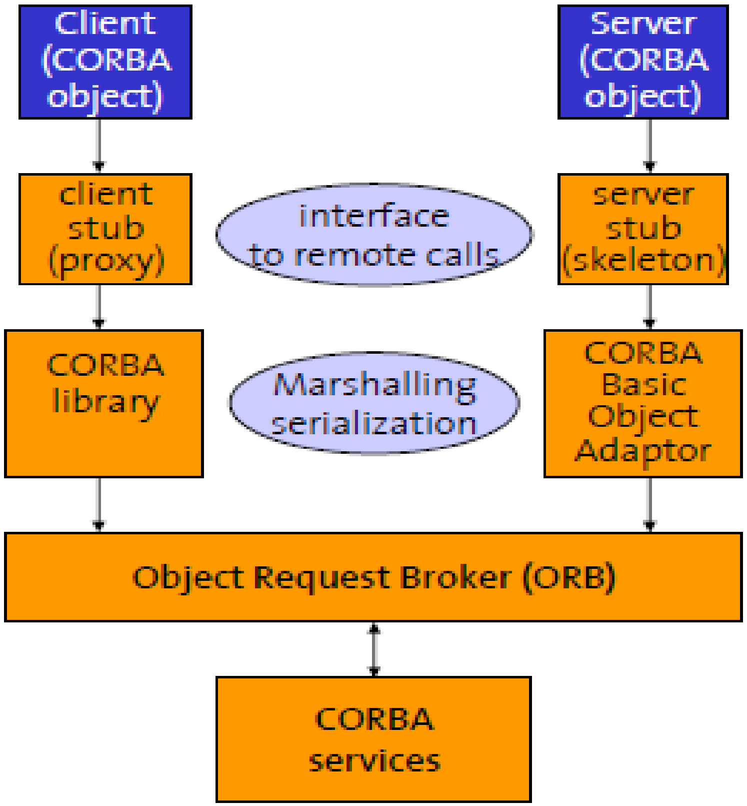
- Designed for integrating distributed systems across (particularly) heterogeneous platforms
- Used in both embedded and enterprise applications
- Commercial and open source implementations available

Standards

- A set of standards by Object Management Group (OMG)
- CORBA 1.0 adopted in 1991, CORBA 2.0 adopted in 1996, CORBA 3.0 adopted in 2002

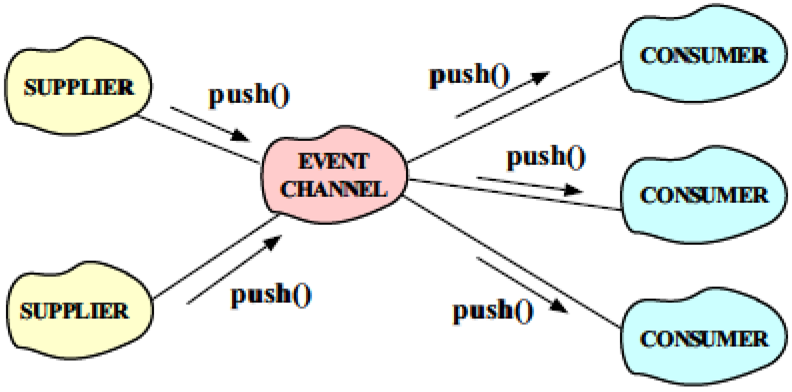
CORBA - Core Concepts

Object-Oriented Remote Procedure Call



Event Channel Service

(one of several CORBA system services)



Note: Pull model and push-pull model are also supported

Source: ETH Zurich (Swiss Federal Institute) Enterprise Application Integration (Middleware) Course
http://www.iks.inf.ethz.ch/education/ws05/eai/slides/lec5.pdf

Source: Doug Schmidt, Vanderbilt University, “An Overview of CORBA Event Channel Service”
http://www.cs.wustl.edu/~schmidt/PDF/coss4.pdf

Technologies Vs. the Top 10 Evaluation Criteria

Criterion	DDS	WS-*	CORBA
Openness – Use of standards – Pedigree	[green]	[green]	[green]
Environment – Support for development-time federation	[green]	[green]	[yellow]
Quality Attribute – Performance	[green]	[red]	[green]
Quality Attribute – Ease of change	[yellow]	[green]	[yellow]
Quality Attribute – Interoperability	[yellow]	[green]	[yellow]
System Constraints	[green]	[yellow]	[green]
Implementations – Availability of open source	[red]	[green]	[green]
Architectural Styles Supported	[green]	[yellow]	[green]
Future – Marketplace trend	[yellow]	[green]	[red]
Enterprise Architecture Alignment – Navy OA	[green]	[green]	[green]

DDS – Strengths and Weaknesses

Strengths

- Good fit for this class of system
  - Designed to optimize movement and sharing of data streams
  - Promotes the right qualities (performance, …), data-centric design
- It's open, and open source implementations are available.
- Alignment with Navy OA and other DoD and Navy programs
  - Navy surface community is using DDS
- Does not require dedicated hardware, due to peer-to-peer architecture

Weaknesses

- Small community (today), indicating small demand
- A niche specification
- Provides no direct support for client/server
- Open source implementations are limited and largely unsupported

DDS – Risks

- Critical mass
  - DDS adoption may not reach critical mass.
  - The small community means it could be difficult/expensive to obtain qualified resources.
- Implementations
  - Open source implementations may not yet be in use in production environments.
  - Unknown dependency on a vendor (in case of a non-open source solution)
  - Unknown cost and quality of support for the open source implementations
  - Implementations may not be of good quality.
    - E.g., there is reportedly a problem with reliable multicast
- Complexity
  - Uncertain complexity of the DDS programming model
  - Configuration of the QoS parameters is complex, with an accompanying steep learning curve.
  - Could be overkill for this system's requirements

Cost/Benefit Analysis: DDS

Costs:

- Potential high cost of training and personnel because it's a niche community with limited adoption and resources
- Significantly higher licensing cost for a commercial (as opposed to open source) implementation
  - Cost of commitment to support available open source products because of limited open source community support
- Cost of change: re-architecting/re-engineering, retraining, retooling, generating and implementing new governance, etc.
- Probably low/no cost for dedicated hardware

Benefits:

- Newer technology than CORBA
- Optimized for this kind of application IF invest in full exploitation of all the features that DDS offers
- Possibility of "government open source" within Navy?

WS-* – Strengths

- "Everybody's doing it."
  - Large, robust, active community, driven by the Web
  - Many vendors (including big ones), many people qualified and familiar
  - Many users/adopters using it for mission-critical systems
- Good support for interoperability, reuse, composability
- Many newer distributed computing paradigms (e.g., Grid and Cloud) are based on service-oriented concepts.
  - Service-orientation is the next logical progression in distributed computing.
  - The core idea of services is unlikely to go away.
- Open source implementations are available that are tested and widely used.
- Alignment with Navy OA and other DoD programs
  - Basic Web Services are sufficient for this system's purposes.
- It's open – lots of standards activity.

WS-* – Weaknesses

- Lots of standards activity, some overlapping and competing
- May not satisfy real-time performance needs
- Many pieces to coordinate
  - Multiple products (and vendors) will be required to cover all the requirements
  - Implies a need for more architecting, system engineering, and integrating
- Several important technical open questions – evolving body of research and knowledge
- Communication and governance overhead
  - The delta from current processes is unknown.

WS-* – Risks

- Performance
  - May not meet all performance requirements
- Communication/governance overhead
  - Possible gap in governance implied by loss of (development-time recompiled) IDL as a foundation of collaboration and coordination IF ignored
- Flux
  - Some WS-* standards are still maturing and evolving
    - The high rate of change implies higher frequency of adapting to changes – and there are a lot of them – very chaotic, many ripple effects
- Integration
  - Possible need to integrate large number of products from different vendors
  - Fragmentation and churn in products
- Certification
  - Unknown implications of NSA treatment of services with regard to C&A requirements

Cost/Benefit Analysis: Web Services

Costs:

- Cost of engineering missing quality attributes, especially performance
- Keeping up with change/evolution generated by the WS-* community
- Initial cost of new governance and learning/implementation costs
- New tooling for development and maintenance
- Cost of change: profiling (WS-I profile), re-architecting/re-engineering, retraining, retooling, generating and implementing new governance, etc.

Benefits:

- Very popular, likely to continue to be so into foreseeable future
  - Large community, a lot of effort to leverage
  - Many viable options from which to choose, e.g., for vendors and products
  - Lower cost of finding qualified staff
- Open source implementations of good quality and mature

CORBA – Strengths and Weaknesses

Strengths

- It's working for you today
- It is open, and you are able to use an open source implementation

Weaknesses

- Need to recompile for changes
- Perceived complexity
- Shrinking community of developers and adopters (unlikely to be used for developing new systems)

CORBA – Risks

- Slowly but surely diminishing resources:
  - Diminishing support for open source implementation
  - Anticipated shortage of personnel with CORBA background
- Continually increasing investment in legacy assets
  - more to move if (when?) change at some point in the future

Cost/Benefit Analysis: CORBA

Costs:

- Cost of recompiles, which triggers new testing cycle
- Cost of training/mentoring new programmers, cost of attracting/retaining CORBA experts
- Cost of implementing new features (e.g., history) not provided in current standard or products
- No costs of change – at least in the short term

Benefits:

- Low/no initial investment – no cost of change (no prototyping, no training, no licensing)
- Status quo – everything good (and bad) remains

Cost/Benefit Summary

"No change" from CORBA is a viable choice, at least in the near term.

- The least-cost option initially
- At some point we expect the costs associated with this are likely to start to overtake the cost of other alternatives.

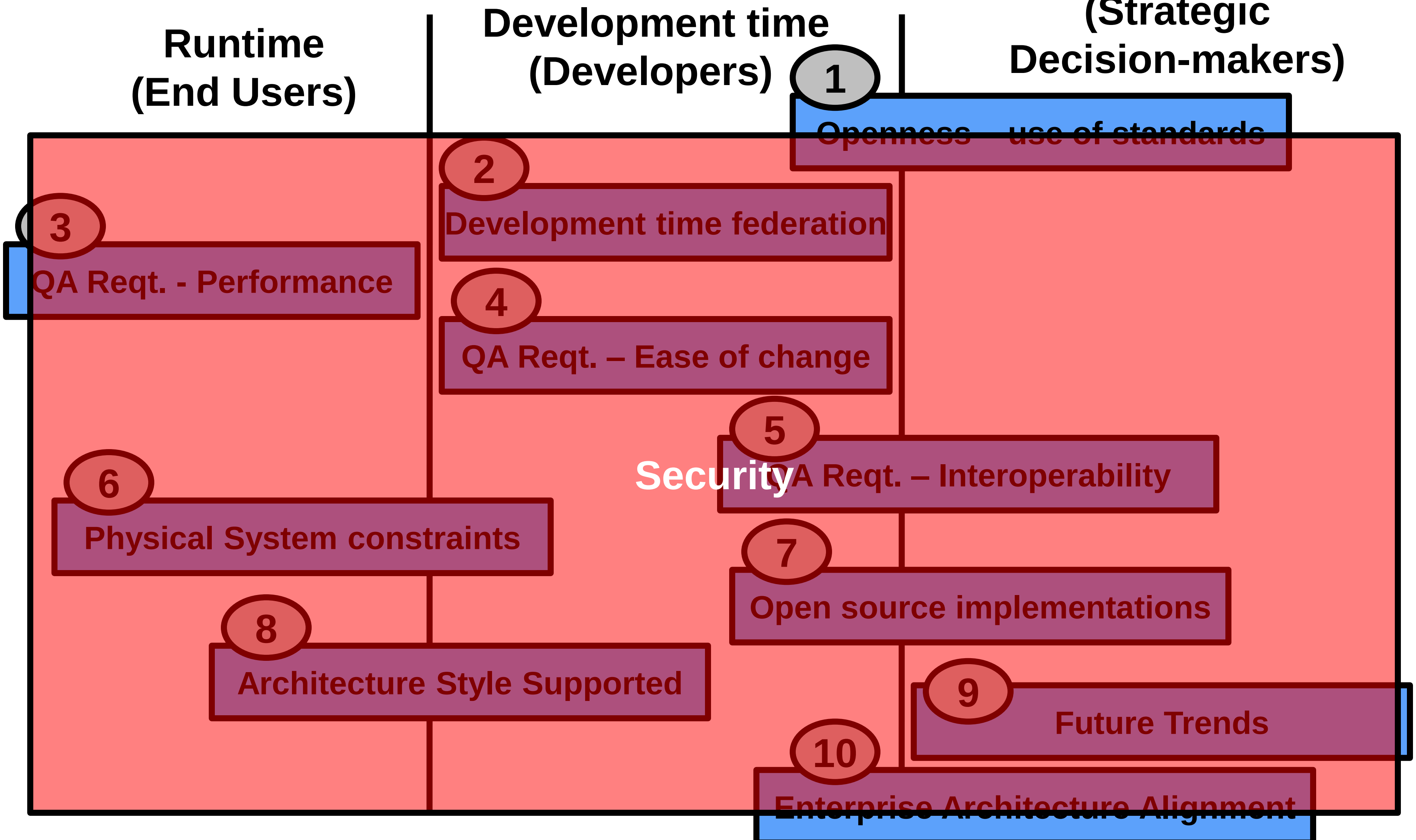
DDS could do the job, but we do not find a clear benefit within justifiable cost unless this system commits to utilizing the entire feature set.

- Appears that the cost/benefit ratio on this is the least favorable

WS-* could do the job and affords many opportunities and choices, both near term and long term.

- Provided the solution can be engineered to meet the performance needs
- Provided the challenges of "herding the cats" can be kept under control

Security Changes Everything



Security Differences

- WS-*
  - WS-* has many security-related standards.
  - However, we have not looked at how usable they are for this system.
- DDS
  - Security was not a quality attribute of focus.
  - However, security has become an area of focus for DDS vendors.

A critical decision factor would be how security options provided by each technology work with the existing security infrastructure.

Outline

The Selection Problem

The Selection Process

Example Use

Results and Lessons Learned

Final Step

The last step in any such evaluation is to put together all the findings of executing the **PECA** process into a coherent recommendation.

Results

Should the program replace CORBA?

Yes

If so, when?

No sooner than 5 years – so have to start now

What should the replacement be?

IF *Web Services can be shown to provide needed performance, then it is the best choice as measured by the program's evaluation criteria.*

How should the program go about doing the replacement?

Incrementally – we have outlined a set of steps and activities to pursue in our recommendations.

Recommendations

Start to plan your change

- Use this study as a starting point
- Start prototyping and piloting
- Start looking at governance changes for each alternative technology
  - Identify changes required to existing processes
  - Identify new processes required
  - Try them out in pilots

Decide about making a change

Catch up and keep up with your architecture

- Take what we (and other studies?) have done and create the architecture documentation that will be required for evolution

Should You Change? Bottom Line

You will need to get off CORBA at some point, but you have the luxury of doing it carefully and through reasoned decisions.

No need to rush into anything

- No technical problem with CORBA today
- However, it is unlikely that any new development will use CORBA.

If So, When?

Very dependent on the timing of possible future events

- Mission requirements changes
- Other technology changes (e.g., multi-core)

Probably in the next 5 – 10 years

- Fewer ORBs and ORB vendors
- Historical technology trends from introduction to establishment tell us that it typically takes 13 – 17 years
- Several articles suggest that few/no new developments are going to be in CORBA
  - Further major investment makes little sense

Why Wait?

WS-*

- Moore's law
  - Processing power may be good enough to meet this system's performance needs.
- Second adopter's advantage
  - Better insights into limits of using WS-* to meet demands of embedded real-time systems
  - Lessons learned from DoD programs using a WS-* approach
- Publish/Subscribe
  - More open source implementations providing better support for publish/subscribe
  - Maturity in WS-* standards associated with publish/subscribe

DDS

- Adoption and market place
  - Better insights into DDS adoption
- Open source implementations
  - Insights into maturity of open source implementations

What About Further Out in the Future?

Some “truths”:

- Architectural styles won't change.
  - Any middleware technology for distributed systems needs to be built on existing concepts.
  - We don't see any paradigm shifts:
    - Next generation (WS-*) being built on previous (including CORBA)
  - Others on the horizon (e.g., cloud computing) don't seem to affect this situation.
    - Multi-core *might* IF someone discovers the software engineering approach to truly take advantage of massively parallel processing

We do not find any technologies right now that are likely to radically change your technology base in the foreseeable future.

- Barring any new disruptive technologies, you should experience evolution, not revolution.

If So, to What Should You Change?

A final answer to this question is not possible without more study.

If experiments and prototypes demonstrate the ability of Web Services to meet performance and security requirements, then Web Services should be the choice.

- Better ecosystem
  - Driven by the Web
  - More open source options
  - Substantial vendor commitment
  - DoD commitments
- Provides better adaptability and flexibility to support evolution of this system

How Should You Change?

Not “Big Bang” – migration needs to be incremental

We suggest 4 phases in making the change:

1. Technology study
   – Started with this activity
   – Needs to be followed with prototyping for enhanced decision-making
2. Re-architecting and re-engineering the software system and planning the migration
   – Including training, governance, system structure, acquisition of hardware/software, retooling, etc.
3. Incremental implementation of the migration
4. Maintenance

Further Studies and Investigations

This evaluation was the first step.

Proposed further investigations in 3 major categories:

- Existing System Analysis
- Market Research
- Prototypes and Experiments

Lessons Learned

PECA is an effective high-level evaluation process

- Created for product evaluation, but equally applicable to technology evaluation – tailorability provides needed flexibility

Keeping in mind that the half-life of evaluation information is about 6 months …

- Both DDS and WS* have a lot of potential for this class of systems
- CORBA is durable but probably no longer appropriate for new developments

Keys to this system's success:

- A forward-looking culture
- A basis in open systems
- A robust technology refresh process

Don't forget: architecture is the #1 system asset

- Never let it get away from you

QUESTIONS?

Contact Information

Tricia Oberndorf, Team Lead
ASP
Telephone: +1 412-268-6138
Email: po@sei.cmu.edu

Tom Merendino
ASP
+1 412-268-1154
Email: tjm@sei.cmu.edu

Soumya Simanta
RTSS
Telephone: +1 412-268-7602
Email: ssimanta@sei.cmu.edu

U.S. mail:
Software Engineering Institute
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Software Engineering Institute | Carnegie Mellon