Middleware Should Think Globally and Act Locally

Mark Hapner
Distinguished Engineer
Sun Microsystems, Inc.

11/27/2007

Middleware has been focused on data consistency ‘integrations’

Data Consistency

- ETL - extract/transform/load
  - Stovepipe Apps
  - Batch Infrastructure
- Data Pipes
  - Stovepipe Apps
  - MOM Infrastructure
- Middleware is local ‘data bus’
  - Clear line between ‘integration’ and ‘application’
  - Emphasis is on breadth of app ‘adaptors’
  - Alternative to shared data and shared function

Entering the ‘Service’ age

- Data and Function is shared via Services
  - Global computing delivers sharing
    - Unrestricted by locality
    - Unrestricted by proprietary architectures
    - Both synchronous and asynchronous
    - Data consistency pipes via services
  - Open standards
    - Internet
    - HTTP, Soap, AS2
  - Different partitioning of responsibilities
    - Intranet/Internet
    - Application ?
    - Middleware ?

We've All Heard the Service Mantra

- Reuse
- Agility
- Discovery
- Interface
- Interop

These are nice ...
But they aren't enough ...

Practically everything we will do in computing from now on requires collaboration via shared services.

To survive, orgs must
offer their value
and
leverage the value of others
via services.

Internal and external collaboration is interleaved.

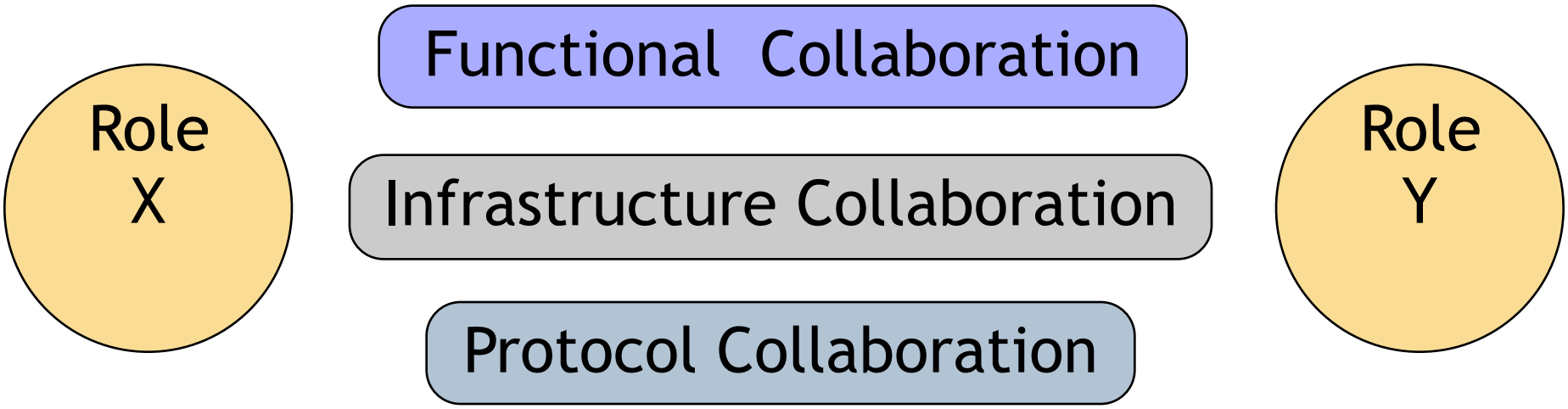
Different infrastructures
for internal vs external
collaboration
are no longer practical.

Service ‘architectures’ over-focus on ‘interfaces’ and ignore collaboration.

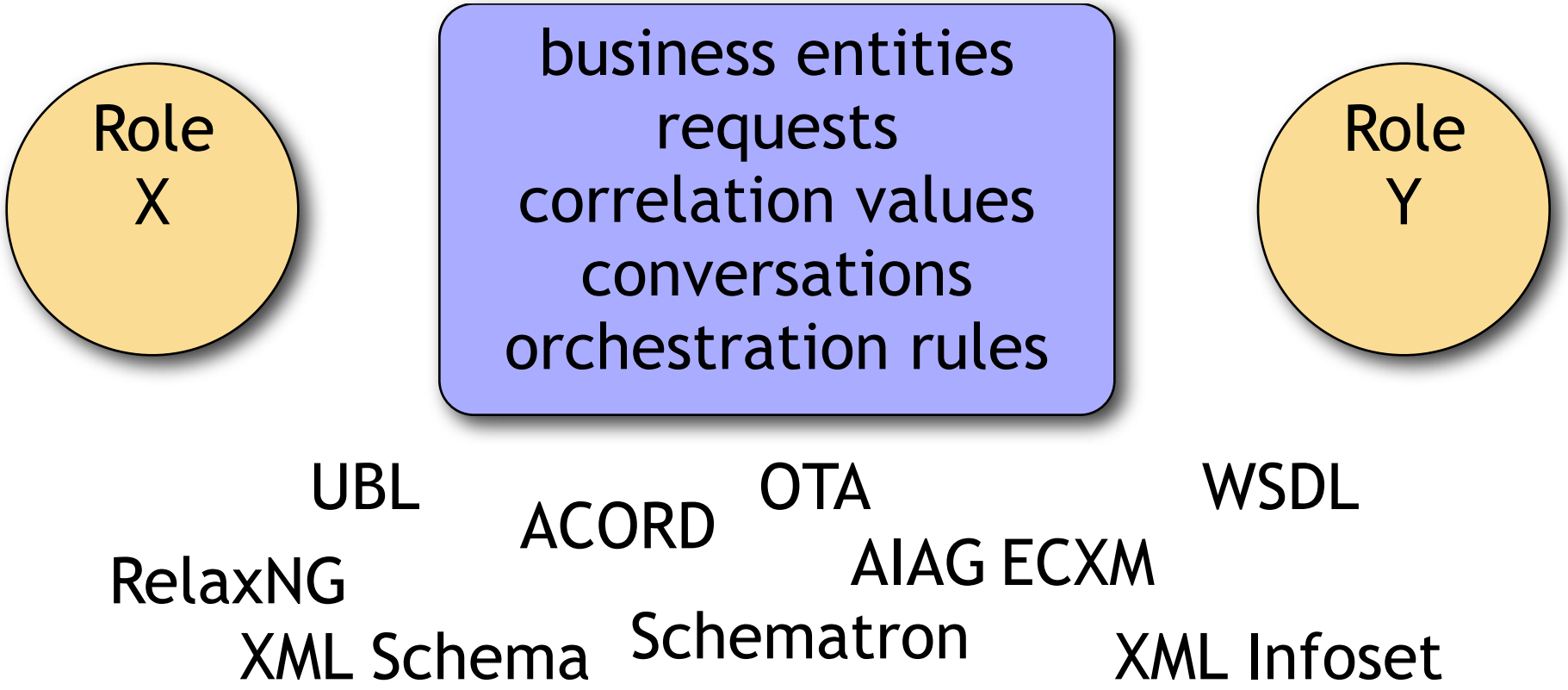
Creating and maintaining
a community of use
- a collaboration -
is the goal of a service.

The power of a collaboration is the capacity of its 'global' roles to hide the 'local' concerns of its parties.

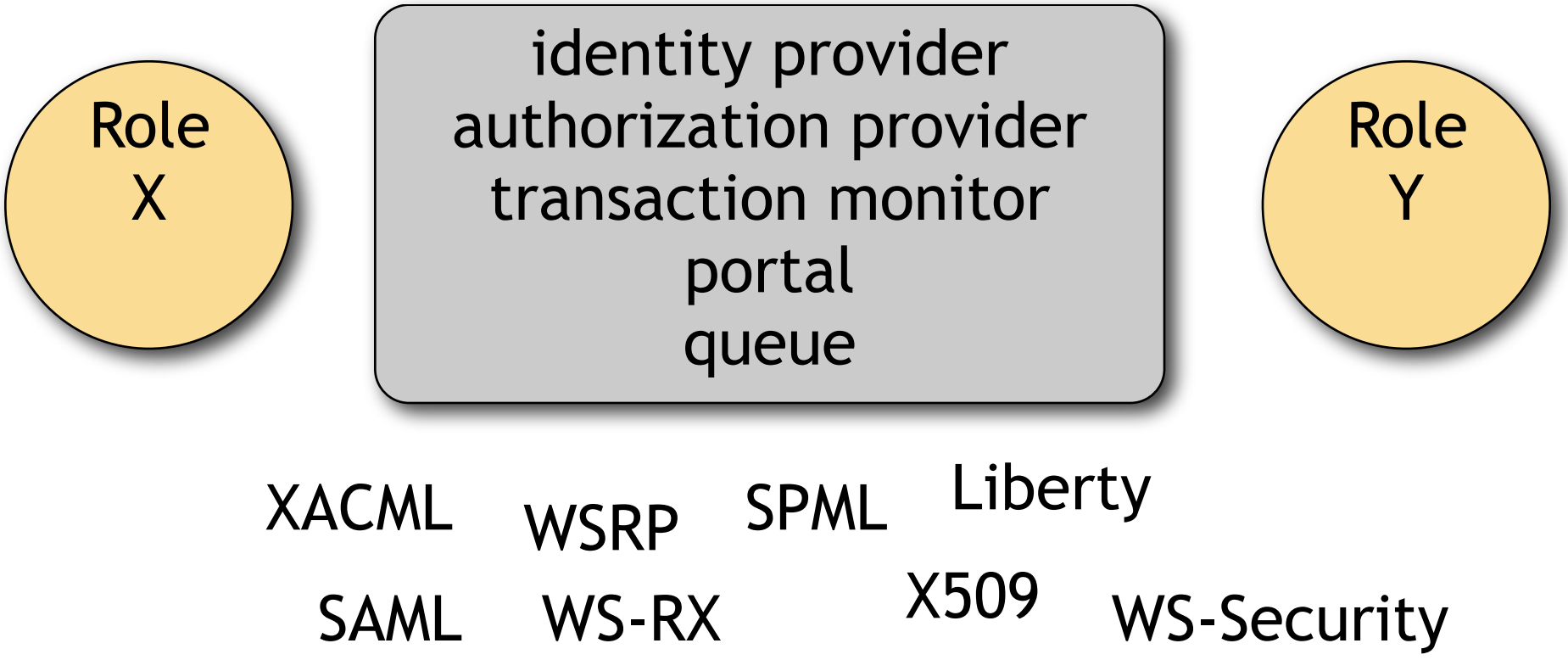
A ‘third party’ View of a Collaboration



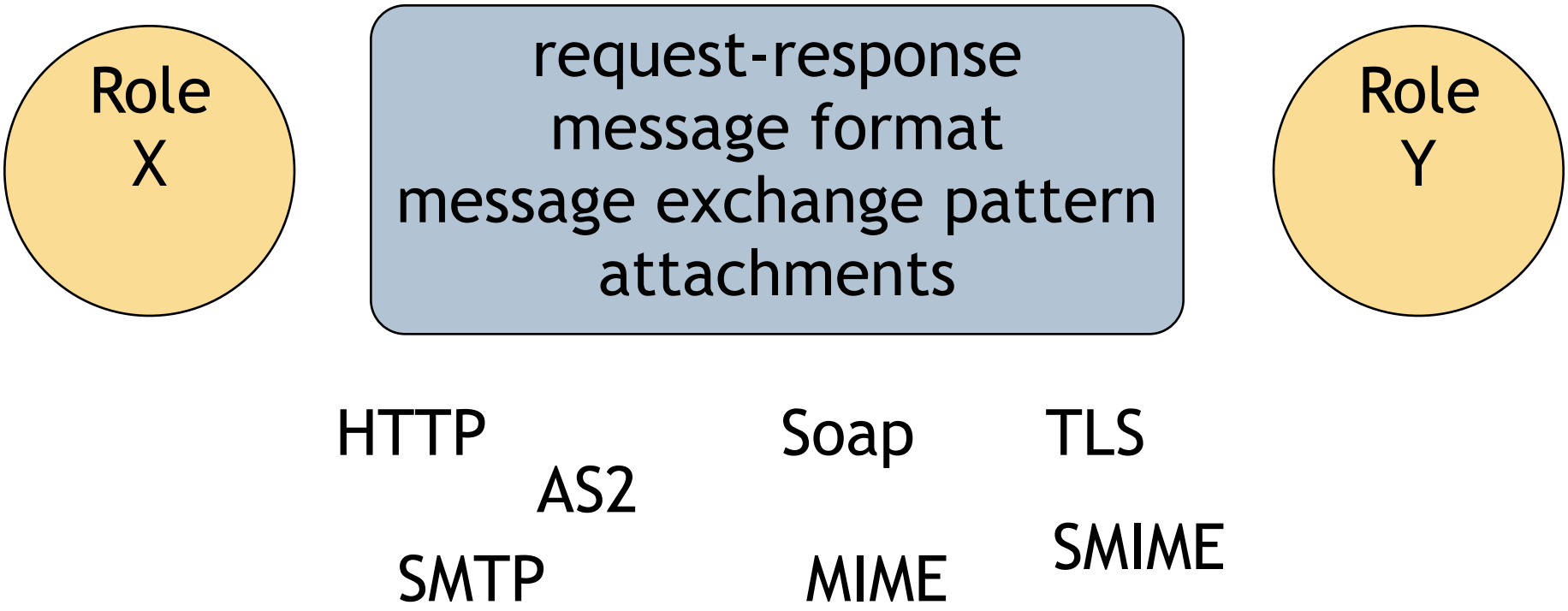
Functional Collaboration



Infrastructure Collaboration



Protocol Collaboration



Collaboration design should be formal and loosely coupled with service design.

Collaborations are complex shared relationships.

They have an existence that transcends the local concerns of its parties.

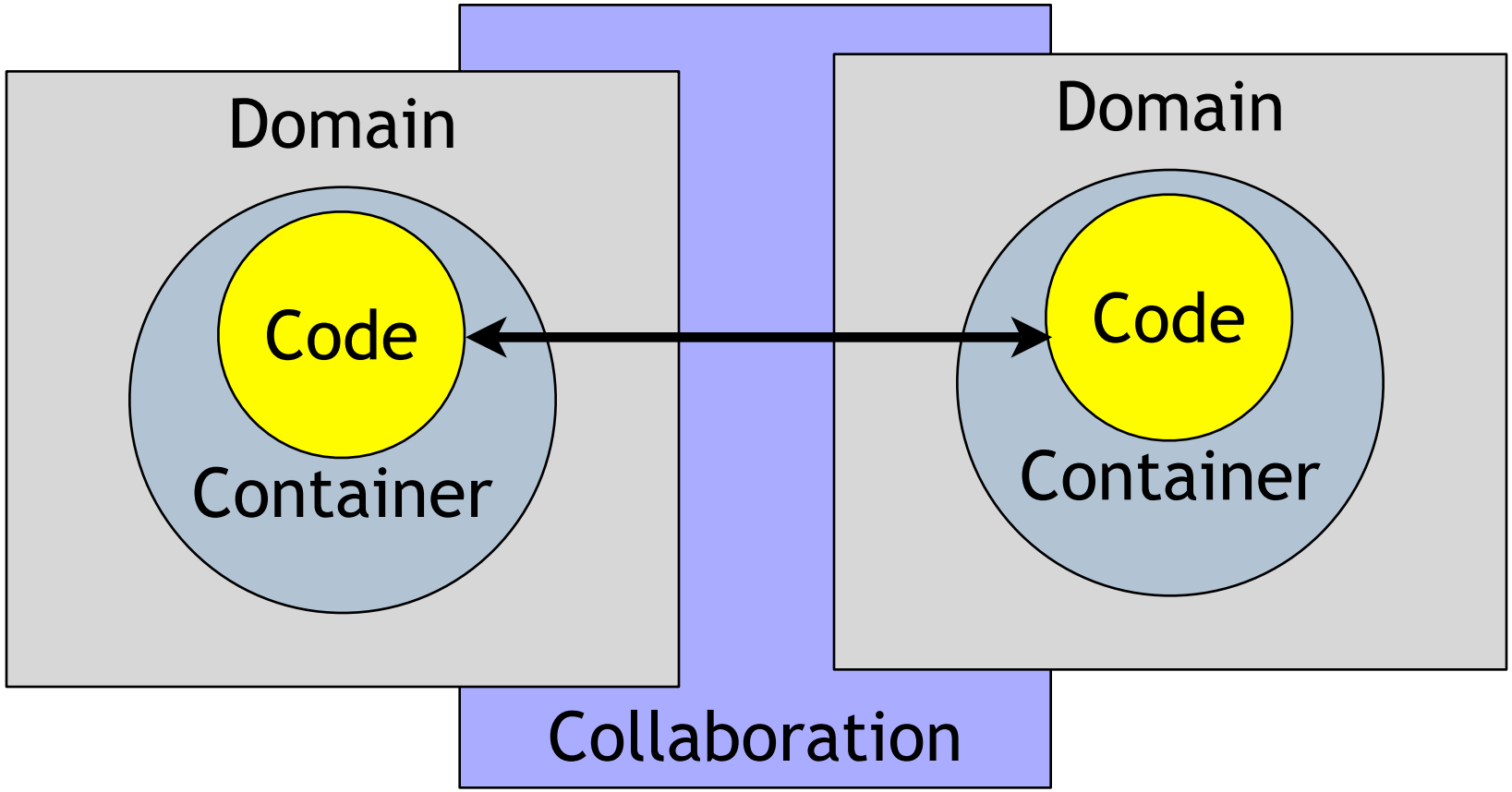
What local architecture best supports the implementation lifecycle and sharing that collaboration providers and consumers require?

What local architecture best separates the responsibilities of business logic developers from those of service domain administration?

Some Competing Architectures

- Direct Connect
- Service Fabric
- Service Bus
- Service Gateway

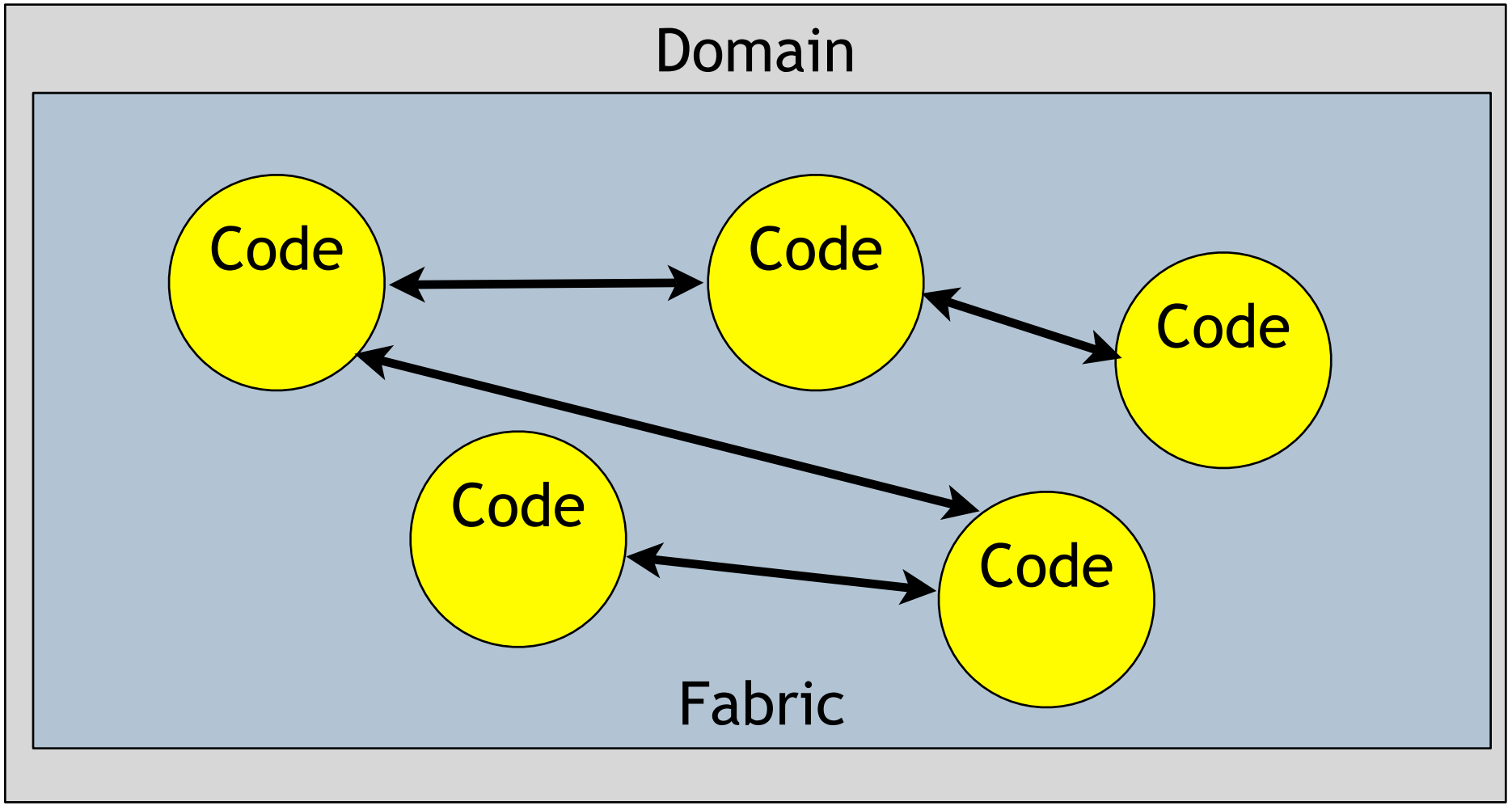
Direct Connect



Sun microsystems

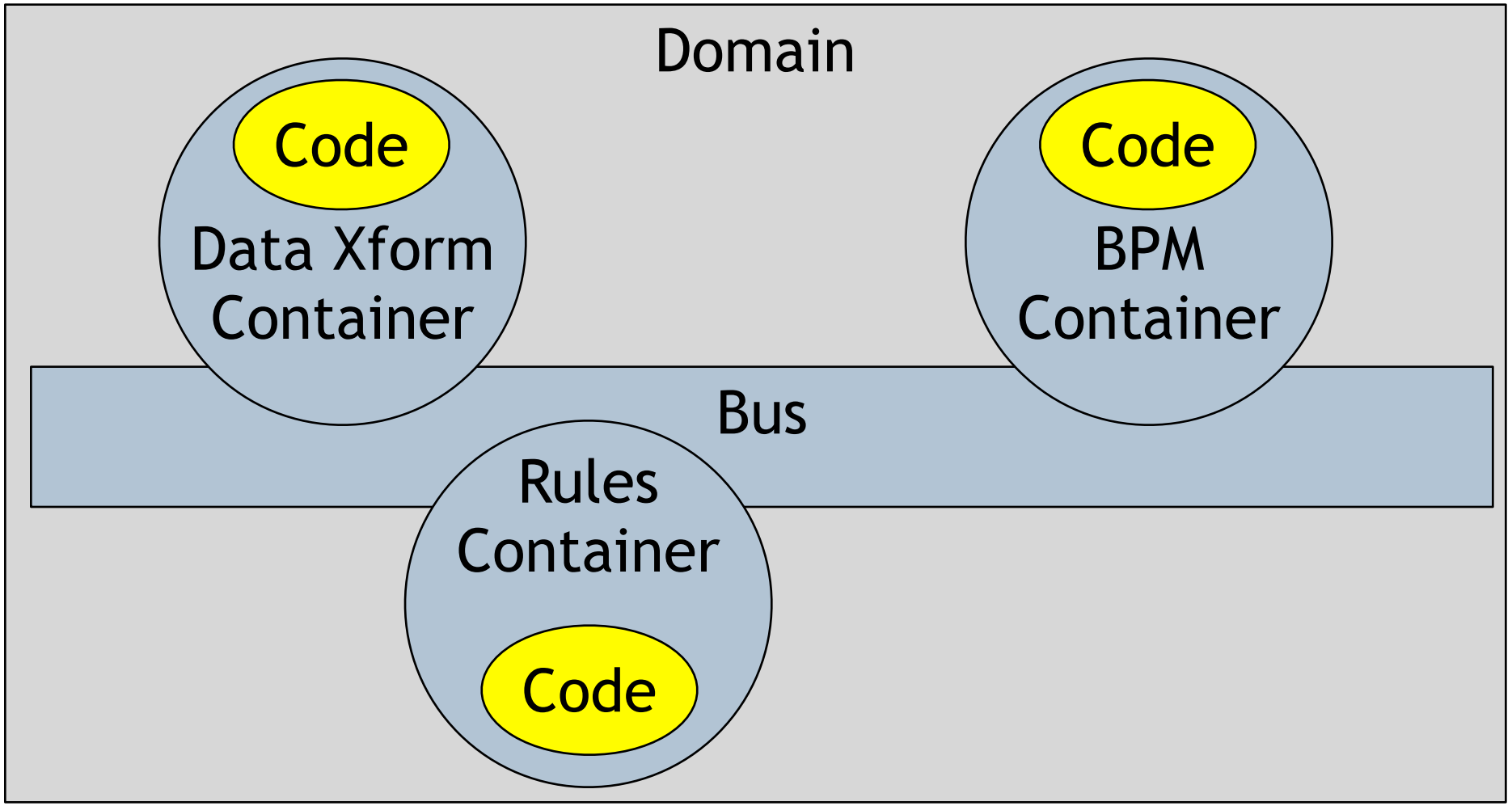
Containers are focused on ‘resourcing’ business logic rather than administering collaboration.

Service Fabric



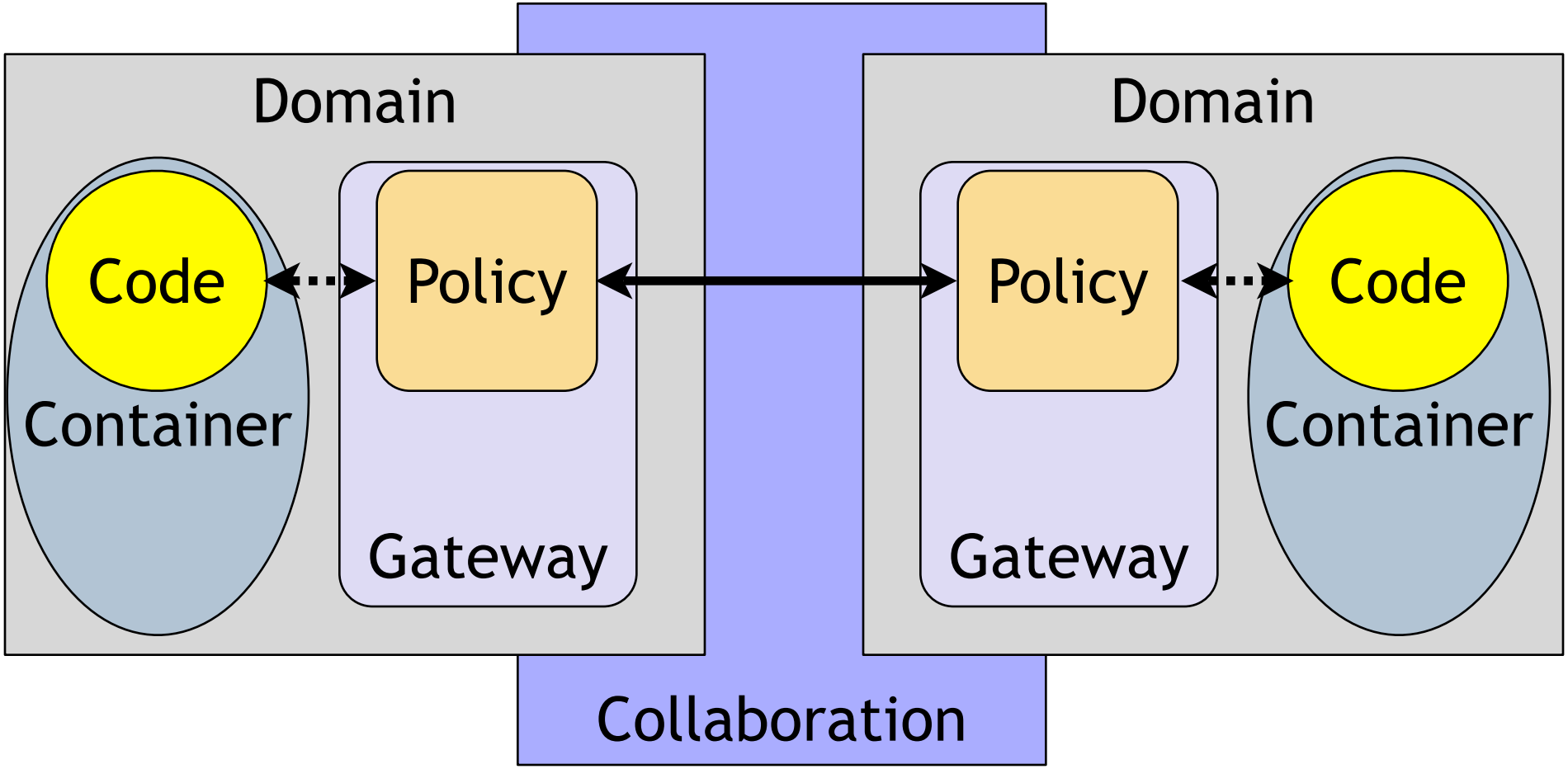
Fabrics simplify local composition.

Service Bus


Domain
Code
Data Xform Container
Code
BPM Container
Bus
Rules Container
Code

Service Bus's simplify composition within a service.

Service Gateway



In effect, the Gateway is the implementation of the service domain layer.

Gateway Role

- Protect the domain
  - From collaboration risks
  - From code risks
- Represent the domain
  - To the collaboration
  - To the code
- Enforce domain policy
  - On the collaboration
  - On the code
- Monitor and audit domain activity

A Gateway provides a Domain with a homogeneous layer of declarative policy to administer a heterogenous local service infrastructure.

Gateways ‘virtualize’ Services

- They map between collaboration policy and local policy
  - Security
  - Identity
  - Access control
- They validate schemas and enforce depth and complexity restrictions
- They provide content based routing for service levels, rolling upgrade, etc.
- They virtualize service metadata

Sun microsystems

Service consumption benefits from gateway policy as much as does service provision.

A Gateway administers
inter-domain
and
intra-domain
policy
with equal ease.

Service Gateways such as Layer 7, Reactivity (Cisco), DataPower (IBM), SOA Software and Forum Systems meet domain throughput and availability requirements.

Another form of domain policy infrastructure distributes some policy enforcement via container agents.

Amberpoint is one such product.

Gateways are also being layered.

Without some form of domain policy infrastructure it's difficult to create and maintain service collaborations.

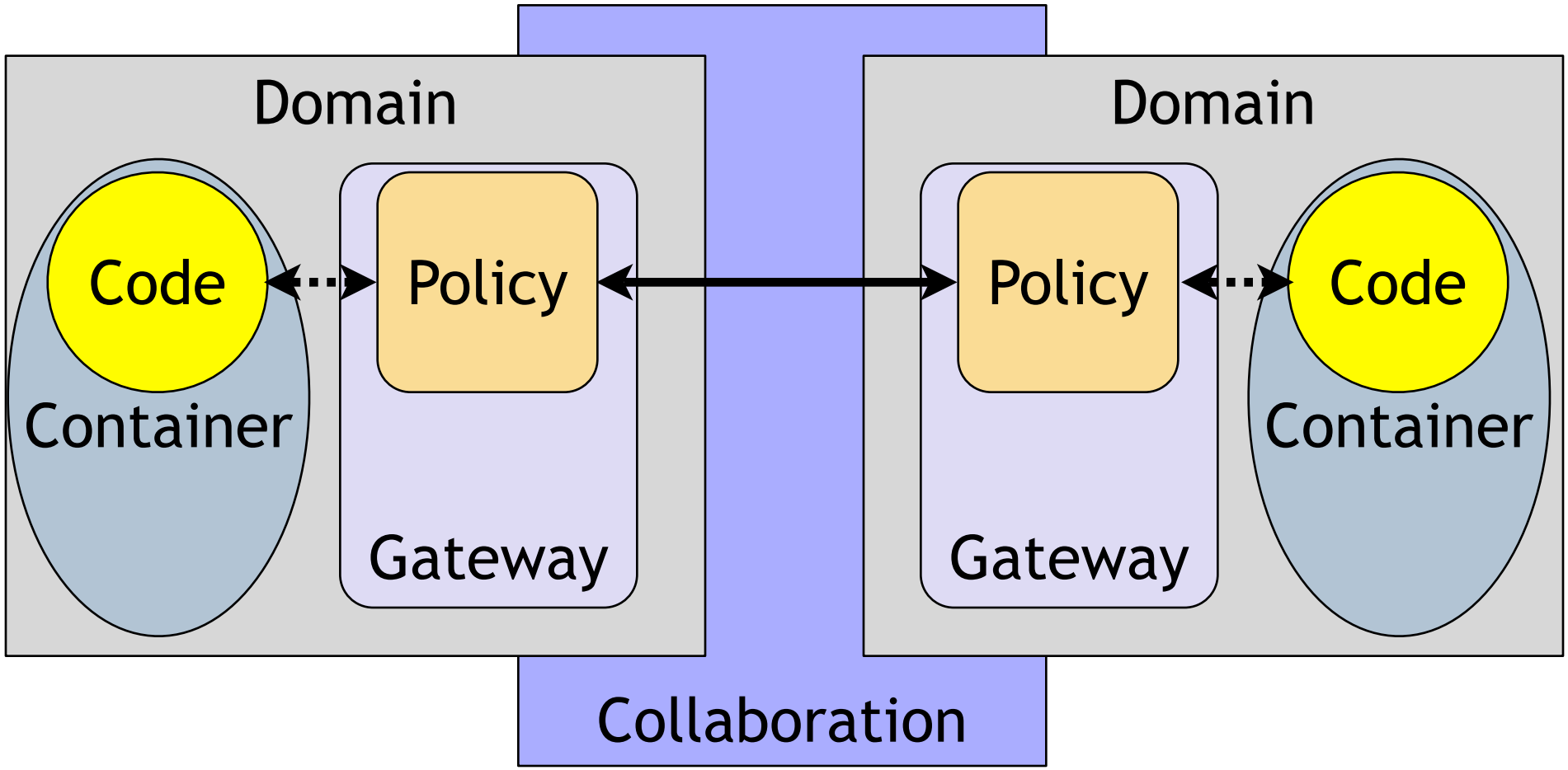
Container Role Expands

- Collaboration also requires more complex business logic
- Containers are evolving to support multiple forms of business logic
- The combination of data transform, BPM, rules, EJBs, scripting, etc. that earlier required a Service Bus can often be done within a single ‘composite’ container

Conclusions

- The objective of Middleware is evolving from ‘integration’ to ‘collaboration’
- Local and global collaboration are becoming interleaved - a single architecture for both is required
- Middleware is dividing into two loosely coupled layers
  - Composite application container tooling/infrastructure
  - Domain policy enforcement tools/infrastructure
- Their combination provides the local architecture for global collaboration

Inter-Domain Collaboration



Intra-Domain Collaboration