

# RESTful SOA or Domain-Driven Design— A Compromise?

Vaughn Vernon  
vvernon@shiftmethod.com



Copyright © 2008-2010 ShiftMETHOD. All rights reserved.

# Overview



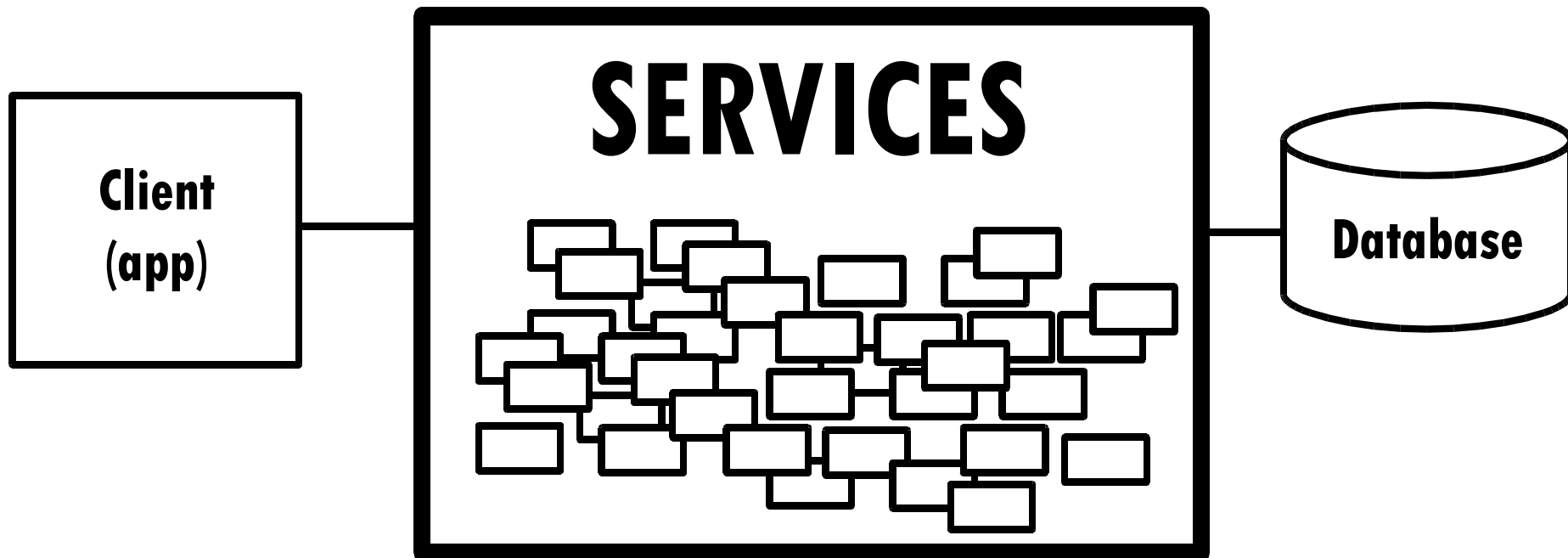
**Potential pitfalls of SOA-only thinking.**

**Strategic Modeling patterns: DDD's “other half” and RESTful SOA**

**Tactical Modeling patterns: DDD building blocks and integration**

**Integrating Bounded Contexts: Making models work together**

# Service-Oriented architecture



# SOa Pitfalls

Service concerns and responsibilities overload

- Services ▪ Transactions
- Security ▪ **Domain Logic**
- **Data Access** ▪ **Translation**
- Transformation ▪ Representation
- **Integration**

**TANGLE  
DISTORTION**

**Ask-Decision-Set**

**Tell-Don't-Ask**

# What about business?



The business focus is on its *domain*

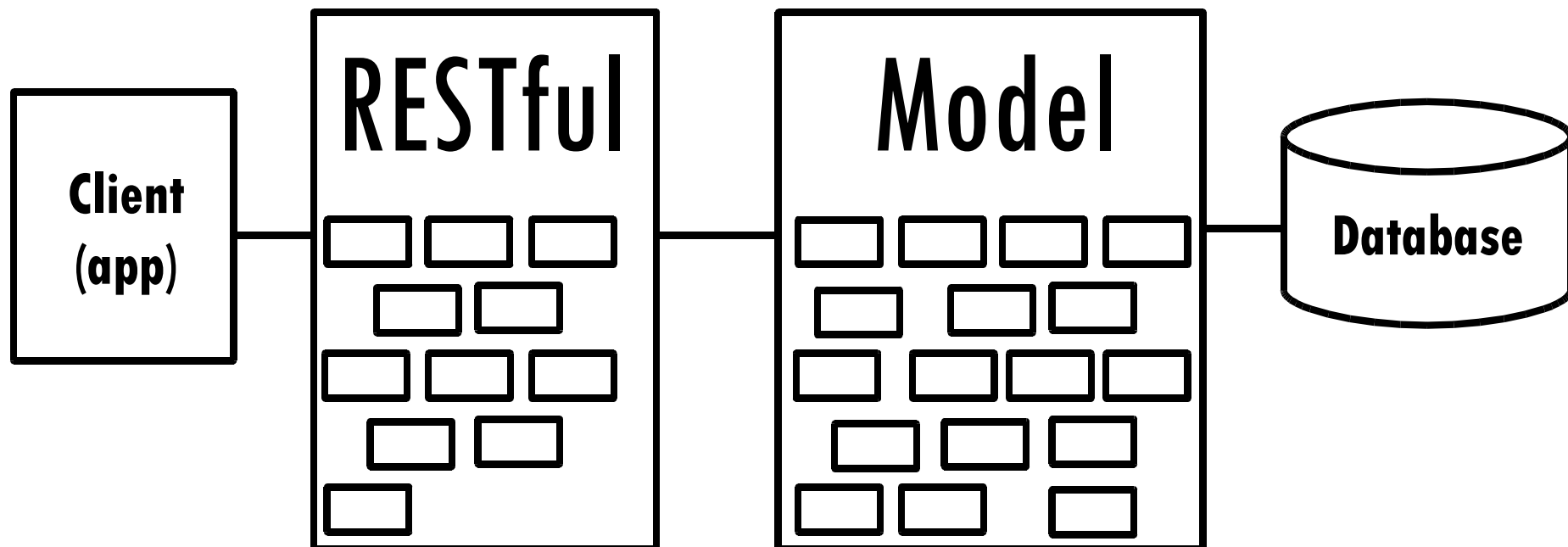
Is your business domain that of Service-Oriented Architecture?

We want to do *Domain-Driven Design*, not SOA-Driven Design

# RESTful SOA with DDD

Model based on the language of the business domain

Services based on resources that represent Model state/behavior



# DDD Strategic Design

“Say again!”

**SURPRISE!**



Strategic Modeling is not really the “other half” of DDD

Strategic Modeling is the most widely applicable part of DDD



# DDD Strategic Modeling

**Bounded  
Context**

**Context  
Map**

**Open  
Host  
Service**

**Published  
Language**

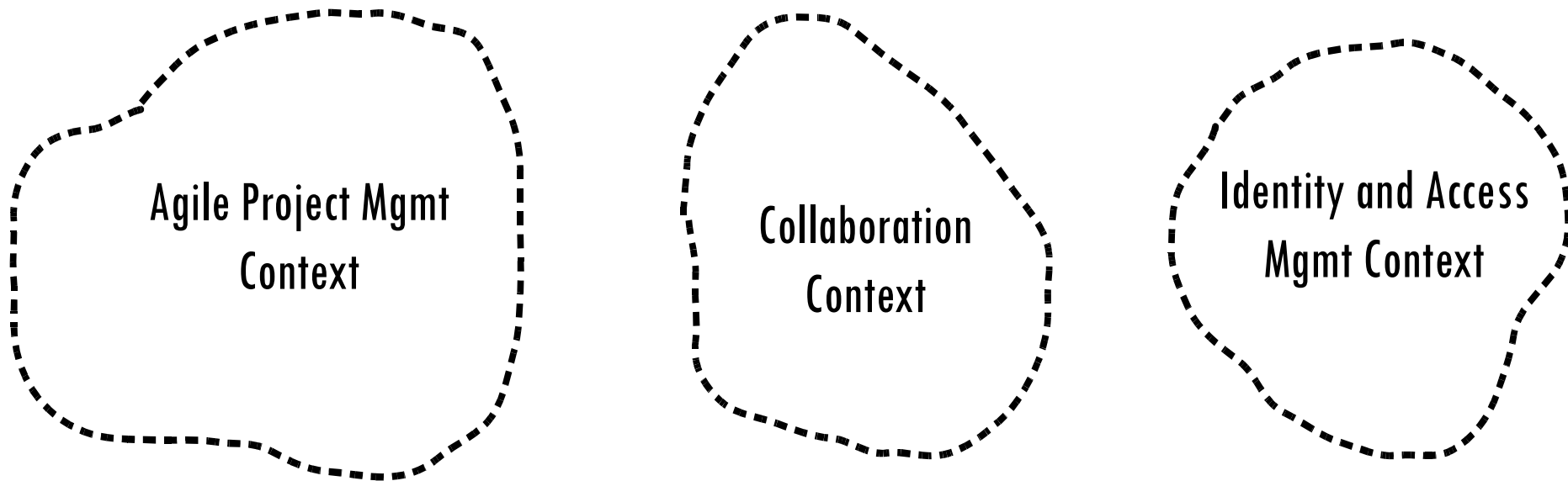
**Anti-  
corruption  
Layer**

Important patterns for basic DDD and integration (BC and CM)

DDD is not about Entities, Value Objects, Repositories, etc.

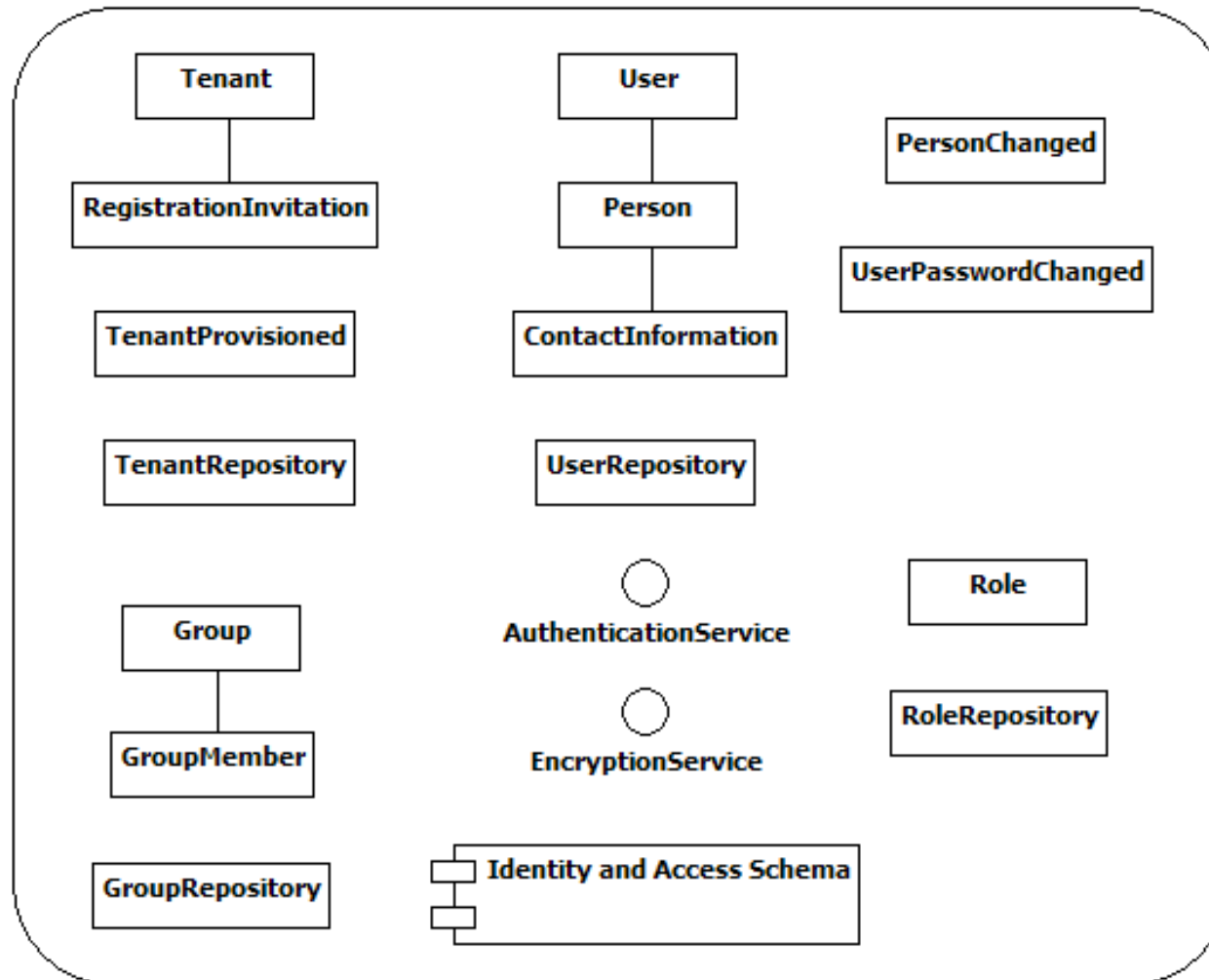
DDD is about modeling the business domain in its own language,

# Bounded Context

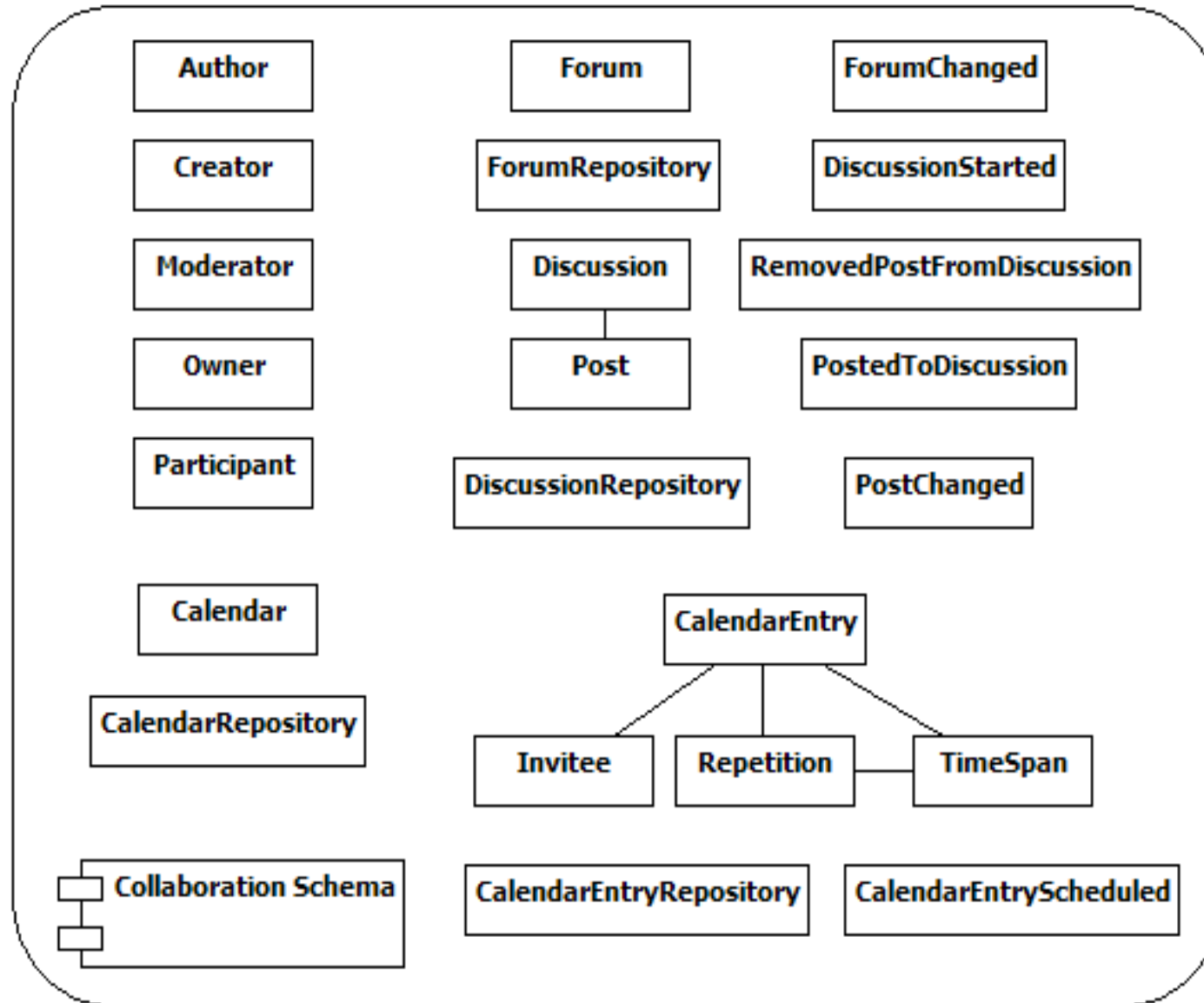


**The delimited applicability of a particular model. Gives team members a clear and shared understanding of what has to be consistent and what can develop independently.**

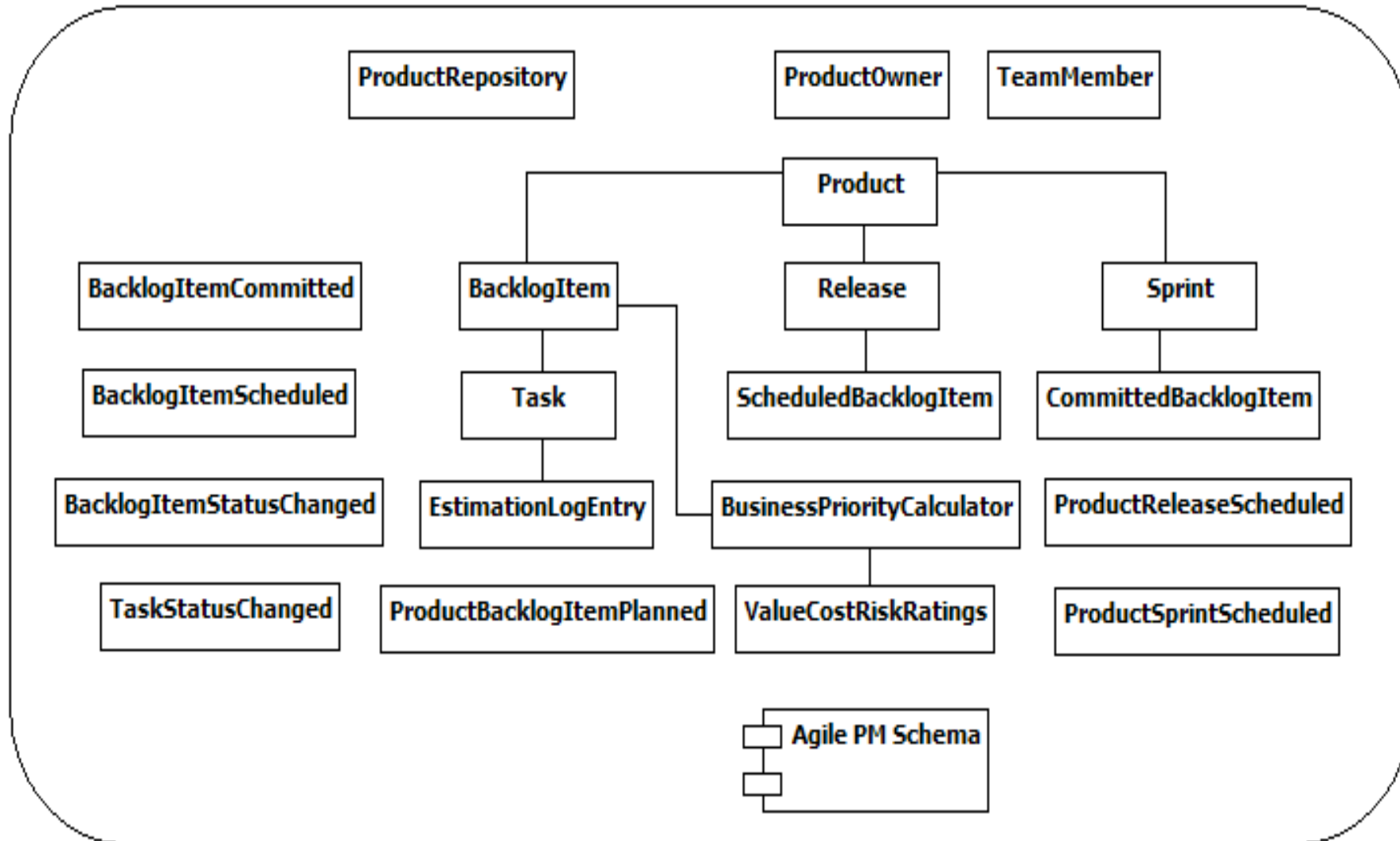
# Identity & Access Context



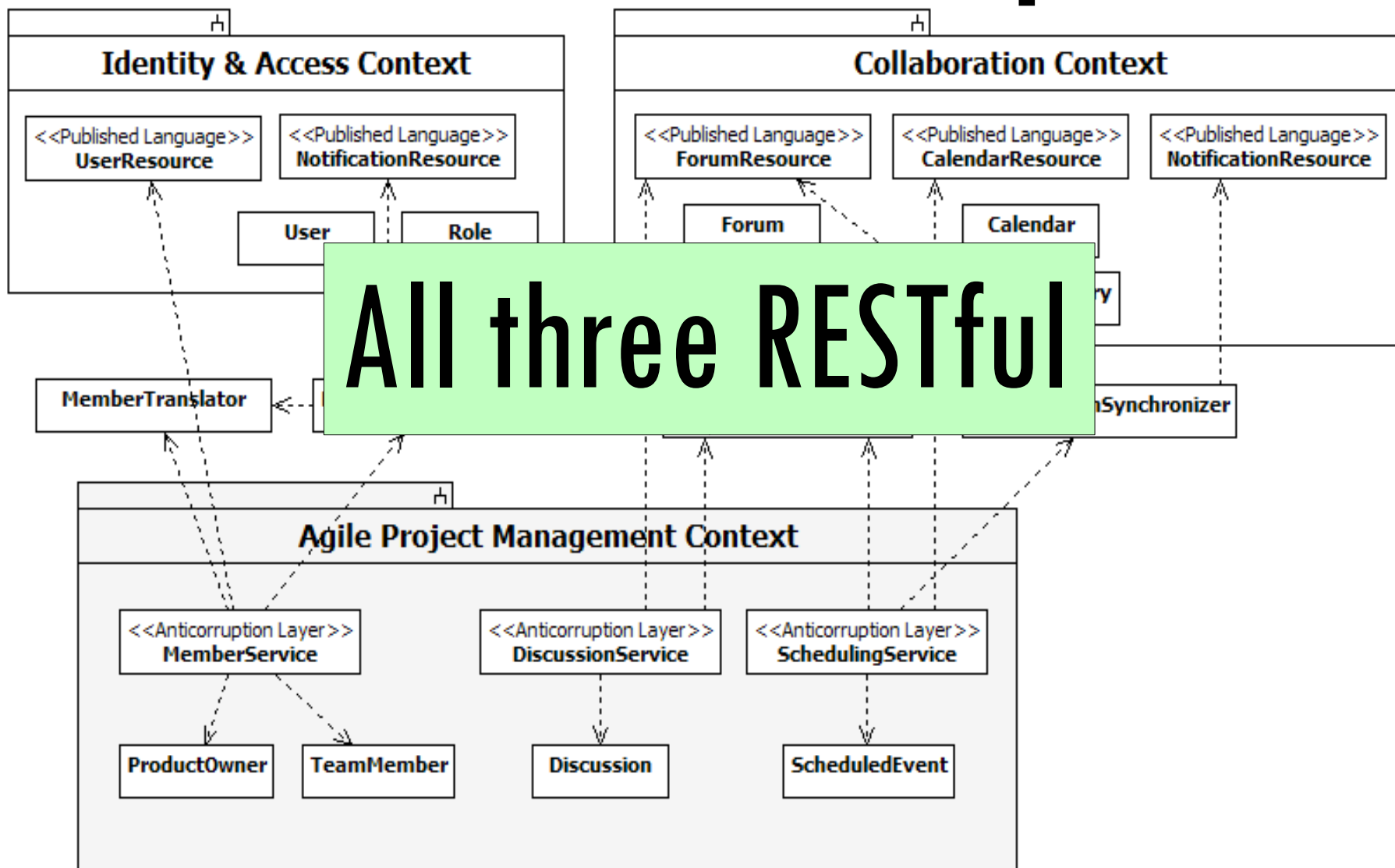
# Collaboration Context



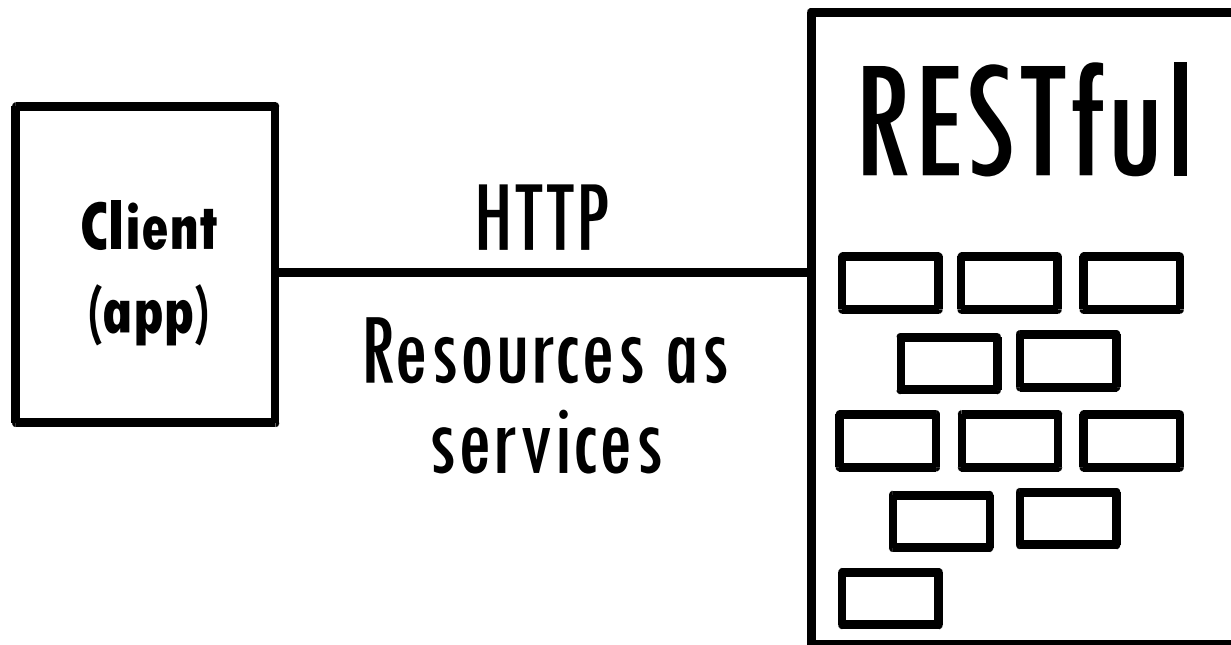
# Agile PM Context



# Context Map



# Open Host Service



Define a protocol that gives access to your subsystem as a set of services. Open the protocol so that all who need to integrate with you can use it.

# Published Language

VML VML Schema Vmath Atom ICON

<xs:schema xmlns="http://www.w3.org/2001/XMLSchema-Base" base="http://www.w3.org/2001/XMLSchema-Base" targetNamespace="http://www.w3.org/2001/XMLSchema-Base" >  
//element

**Model representation  
by choice... use case driven**

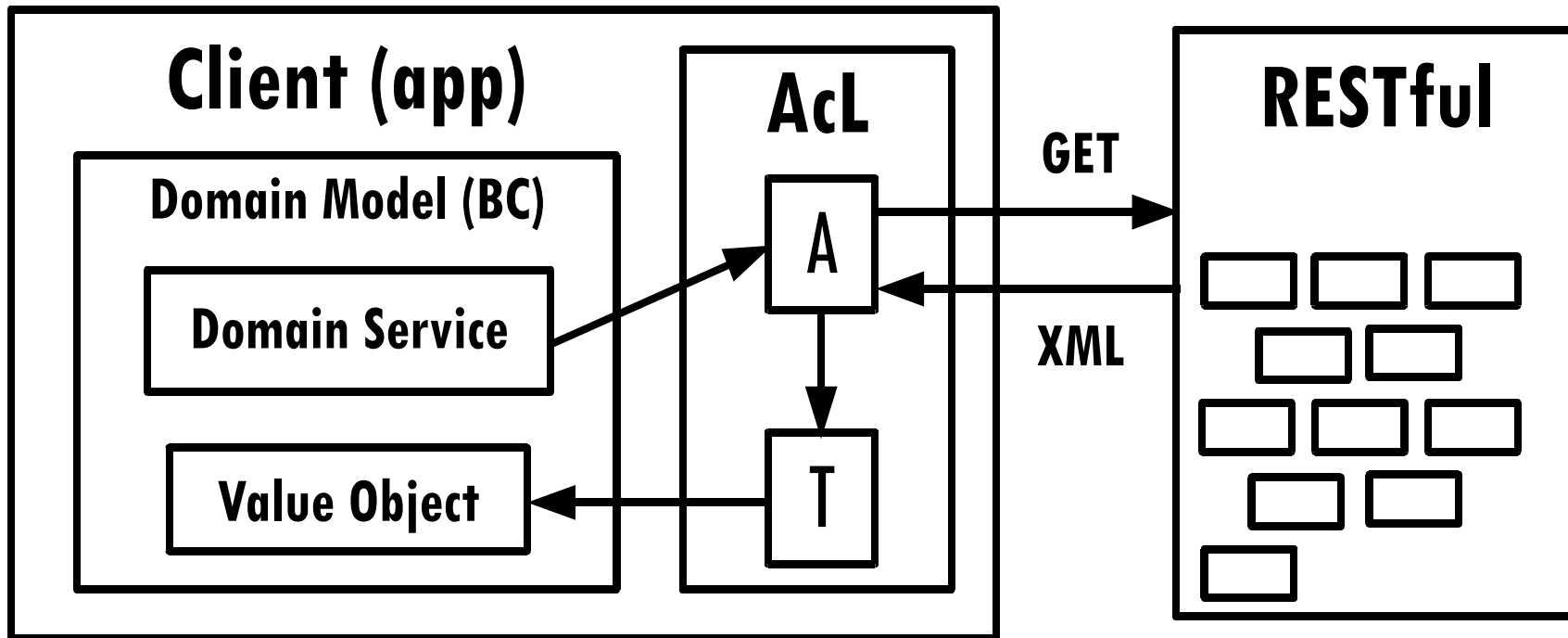
Access

@Produces({"application/xml", "application/json"})

Use a well-documented shared language that can express the necessary domain information as a common medium of communication, translating as necessary...



# Anticorruption Layer



Create an isolating layer to provide clients with functionality in terms of their own domain model.

# DDD Tactical Design

**Layered  
Architecture**

**Aggregate  
(uses Entity)**

**Value  
Object**

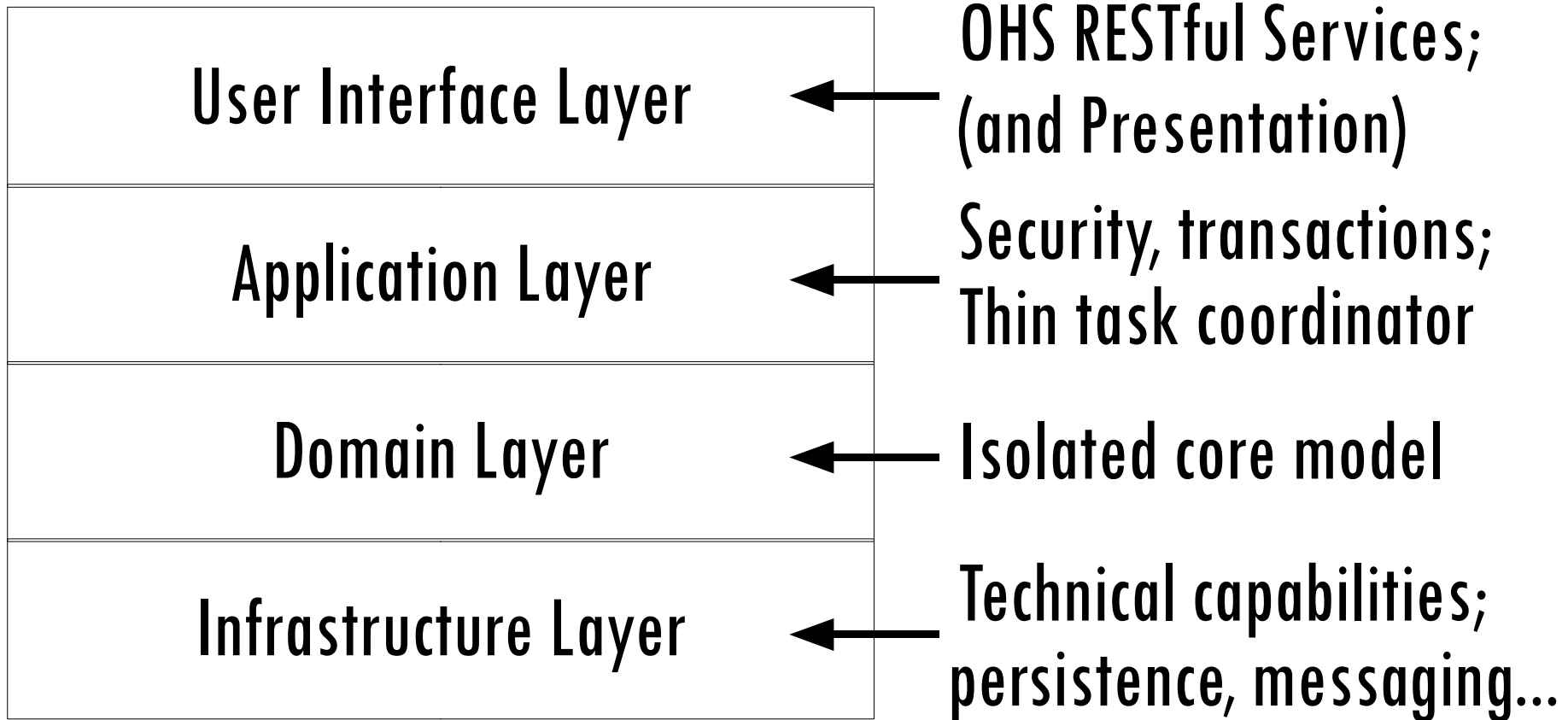
**Domain  
Event**

**Domain  
Service**

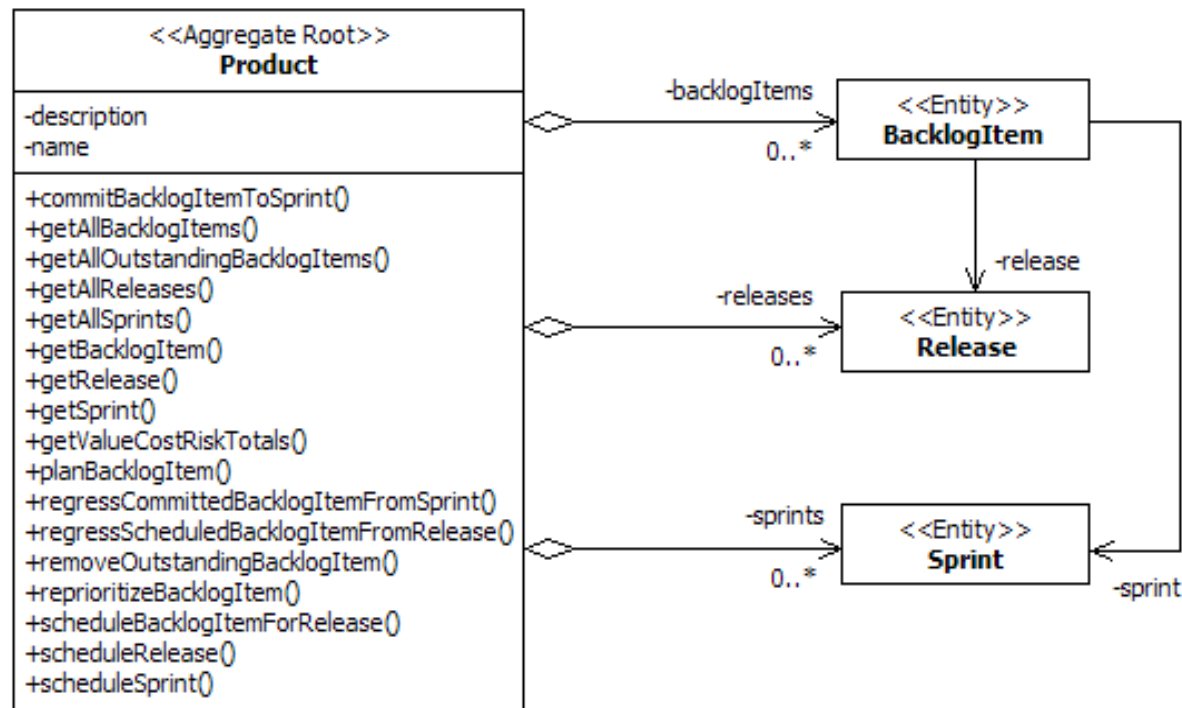
Building block patterns typically used in a DDD project

Support integration patterns of Strategic Design

# Layered Architecture



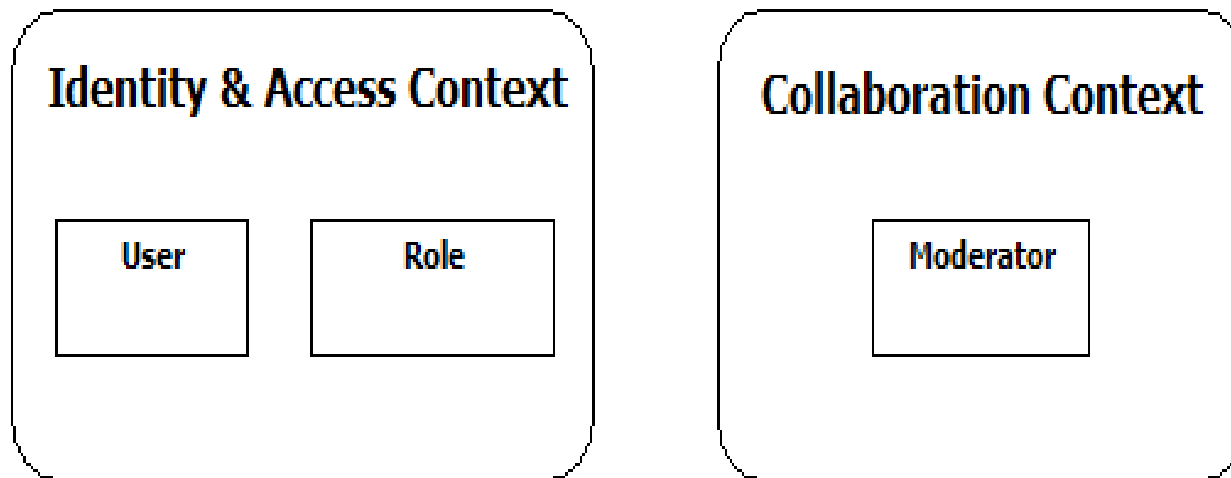
# Aggregate



Object composition cluster with consistency boundaries

Publishes Domain Events, which indicate significant occurrences

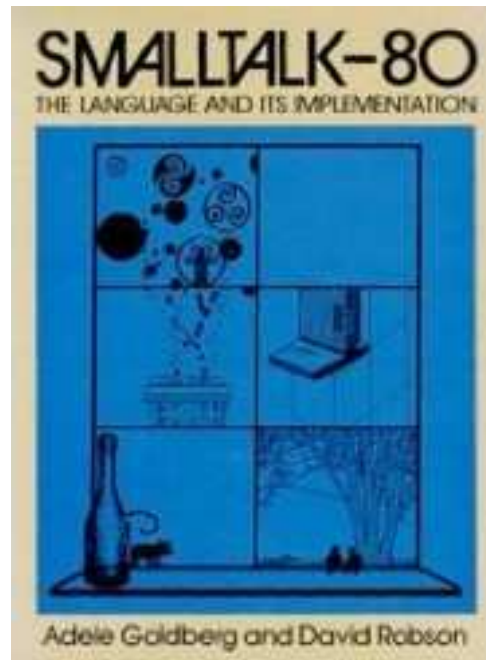
# Value Object



Describes something about an object in the domain model

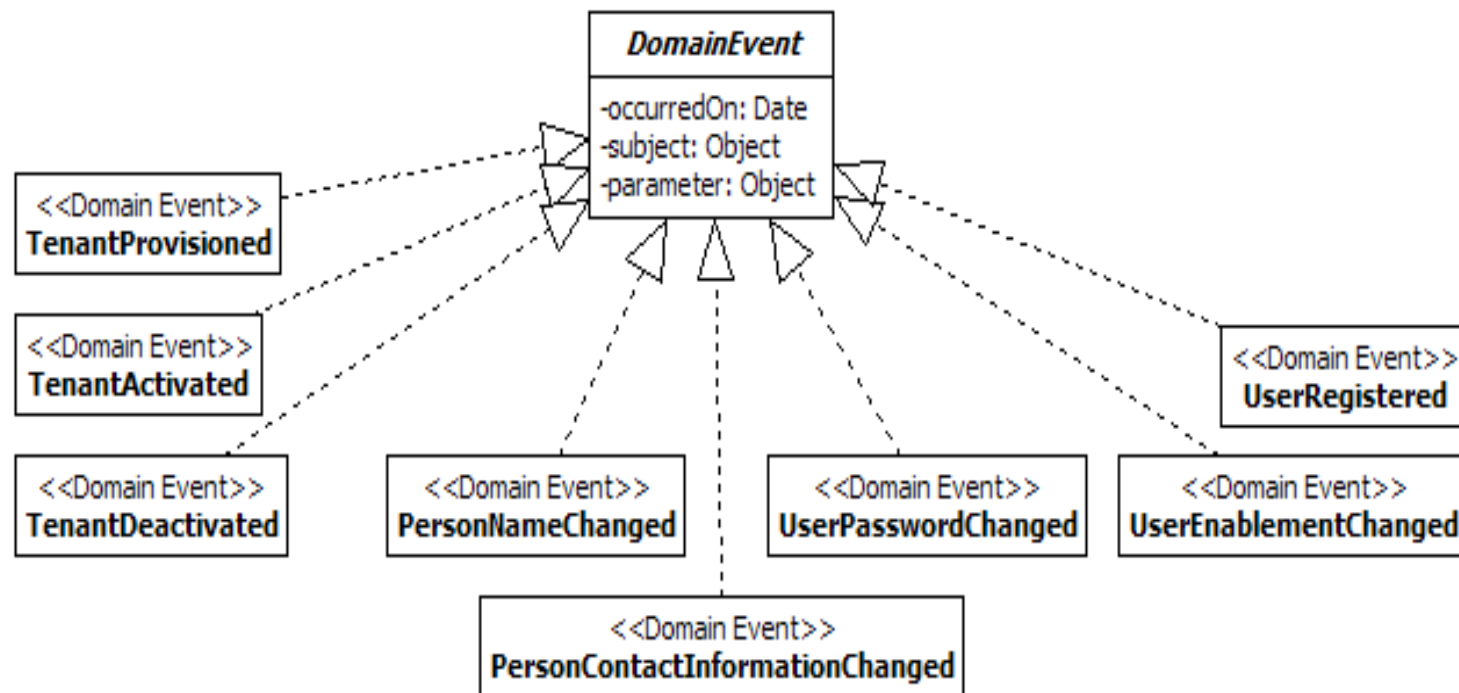
Supports minimalism; concepts translated from other contexts

# Domain Event History



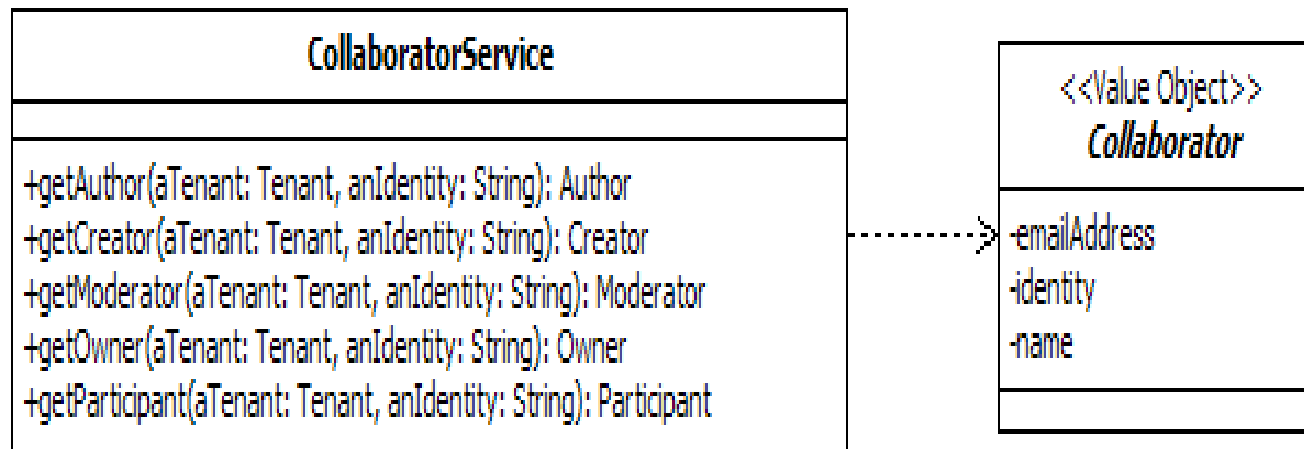
**Publish notifications to Views when updates occur in the Model  
Zero, one, or more updates may occur and View cannot assume**

# Domain Events Today



A significant occurrence in the model; defined in past tense  
Published largely by Aggregates as their states transition

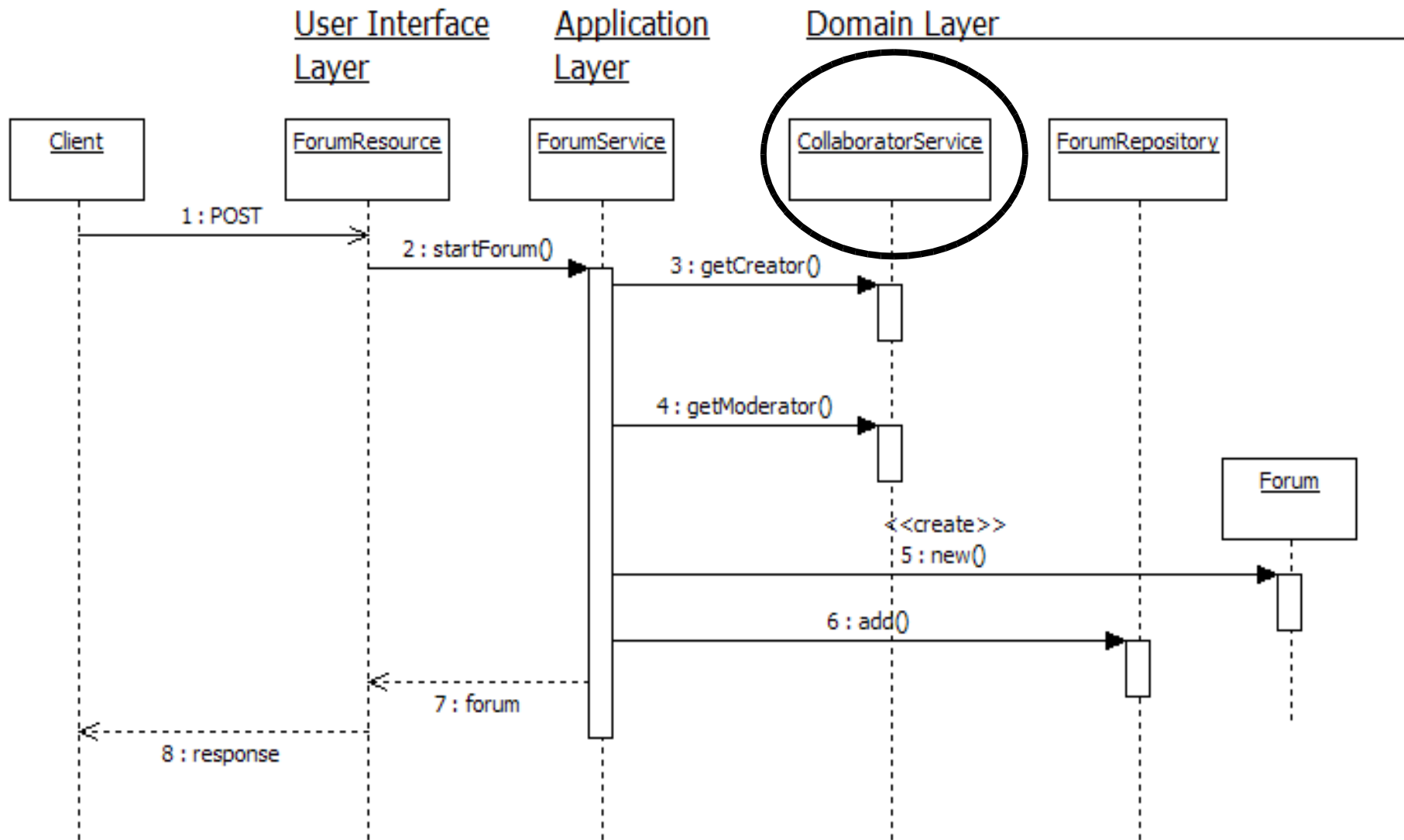
# Domain Service



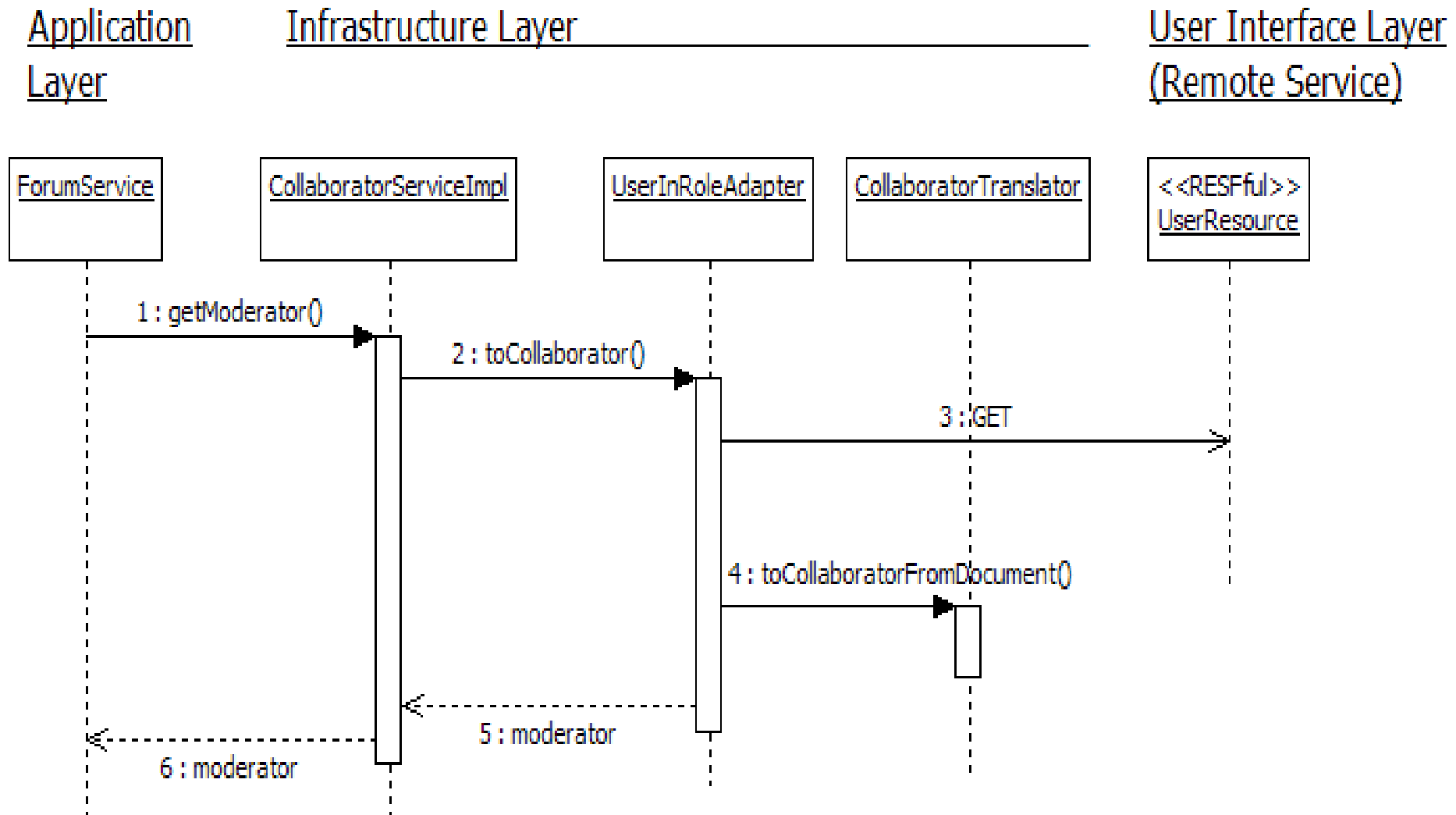
When a significant process or transformation in the domain is not a natural responsibility of an Entity or Value Object, add an operation to the model as a standalone interface declared as a Service.



# DDD + RESTful



# Collab AcL + RESTful



# UserInRoleAdapter

GET https://iam/tenants/{tenantId}/users/{username}/inRole/{role}

```
public Collaborator toCollaborator(...) {  
  
    ClientRequest request =  
        this.buildRequest(aTenant, anIdentity, aRoleName);  
  
    ClientResponse<String> response =  
        request.get(String.class);  
  
    if (response.getStatus() == 200) {  
        collaborator =  
            this.getTranslator().toCollaboratorFromDocument(  
                response.getEntity(),  
                aCollaboratorClass);  
    }  
    ...  
}
```

# GET .../inRole/{role}

```
@Path("/tenants/{tenantId}/users")
public class UserResource {

    @GET
    @Path("/{username}/inRole/{role}")
    @Produces({ "application/xml" })
    public Response getUserInRole(
        @PathParam("tenantId") String aTenantId,
        @PathParam("username") String aUsername,
        @PathParam("role") String aRoleName) {

        // use Application Layer to interact with Domain Layer...
        Response response = ...;
        return response;
    }
}
```

# HTTP GET User-In-Role

```
GET https://iam/tenants/{tenantId}/users/{username}/inRole/{role}
Accept: application/xml
```

---

```
HTTP/1.1 200 OK
```

```
Content-Type: application/xml
```

```
...
```

```
<userInRole>
```

```
  <tenantId>CCA701C2-6490-41B9-B4DA-DB785107C8C8</tenantId>
```

```
  <username>jdoe</username>
```

```
  <firstName>John</firstName>
```

```
  <lastName>Doe</lastName>
```

```
  <emailAddress>John.Doe@domainmethod.org</emailAddress>
```

```
  <role>Moderator</role>
```

```
</userInRole>
```

# CollaboratorTranslator

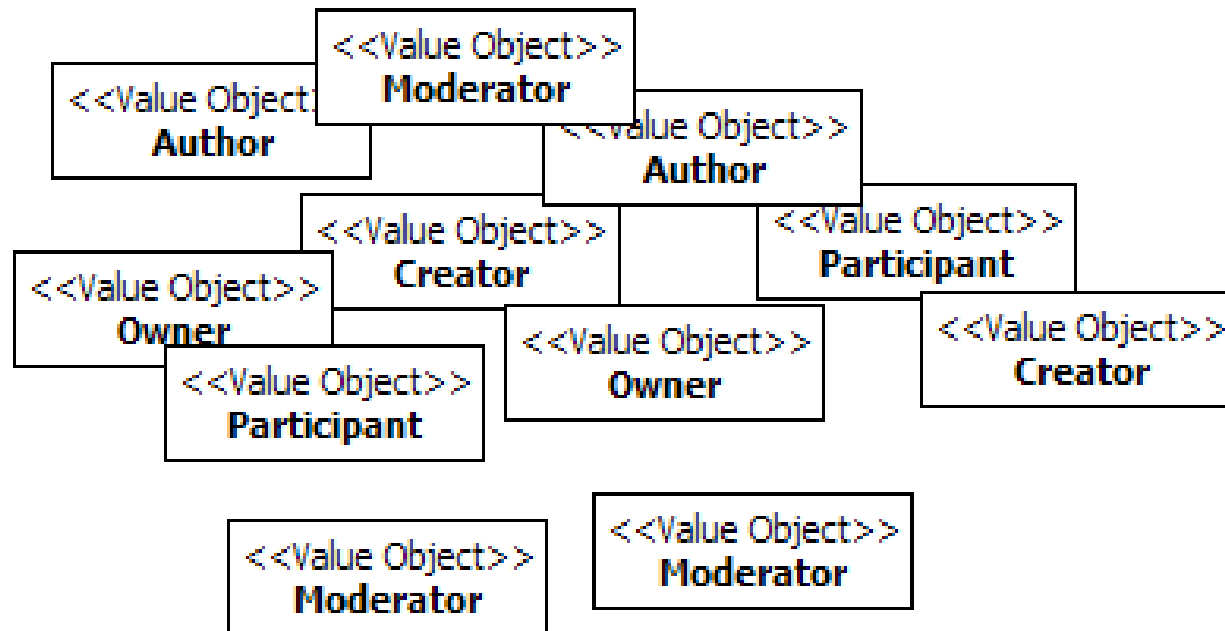
```
public Collaborator toCollaboratorFromDocument(...) {  
  
    Document doc = this.buildDocument(aUserInRoleDocument);  
  
    XPath xpath = XPathFactory.newInstance().newXPath();  
  
    String username = xpath.evaluate("//username", doc, ...);  
    String firstName = xpath.evaluate("//firstName", doc, ...);  
    String lastName = xpath.evaluate("//lastName", doc, ...);  
    String emailAddr = xpath.evaluate("//emailAddress", doc, ...);  
  
    Collaborator collaborator =  
        this.newCollaborator(  
            username, firstName, lastName, emailAddr, aCollabType);  
  
    return collaborator;  
}
```

# **“Simple” Event: Zoe Doe Gets Married**

**Service Capability: I&A self service allows Zoe to change name**

**Service Quality: Eventually names must reflect I&A Context**

# Problem?



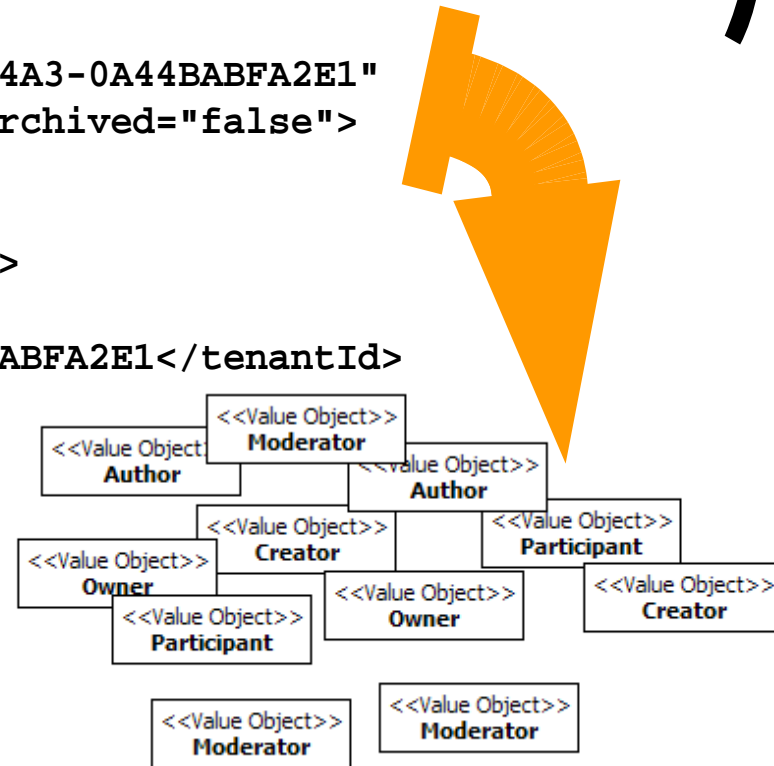
Collaborator Values are created new for each request (by choice)

Changes to users in the I&A Context are not reflected in Collab



# Solution (HM not MOM)

```
<notifications tenantId="C888F4B2-71F0-48D2-B4A3-0A44BABFA2E1"
  id="55F03810-09FF-4925-B9FD-7935CC91F31D" archived="false">
  <notification type="personNameChanged"
    occurredOn="2010-08-19T16:38:06-06:00"
    id="A9D234C6-96EF-4B9A-BAC9-F720A2115B3B">
    <personNameChanged>
      <tenantId>C888F4B2-71F0-48D2-B4A3-0A44BABFA2E1</tenantId>
      <username>zoe</username>
      <firstName>Zoe</firstName>
      <lastName>Jones-Doe</lastName>
    </personNameChanged>
  </notification>
  ...
</notifications>
```



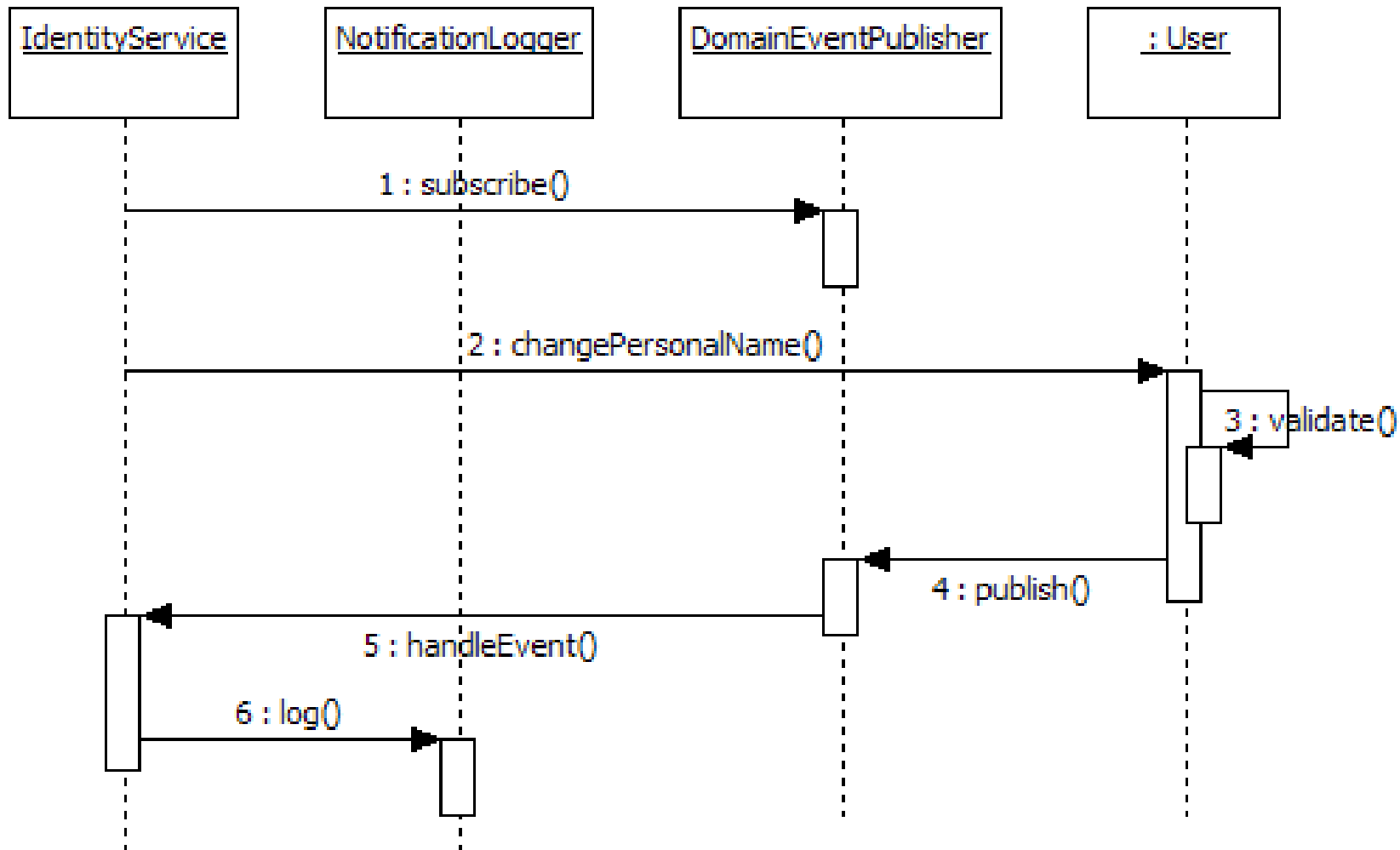
Use shared Value Objects that are eventually consistent

Publish event-based notifications out of each Bounded Context

# I&A Domain Events

Application Layer

Domain Layer



# I&A Event Types

**GroupGroupAdded**

**GroupGroupRemoved**

**GroupUserAdded**

**GroupUserRemoved**

**PersonContactInformationChanged**

**PersonNameChanged**

**TenantActivated**

**TenantDeactivated**

**TenantProvisioned**

**UserEnablementChanged**

**UserPasswordChanged**

**UserRegistered**

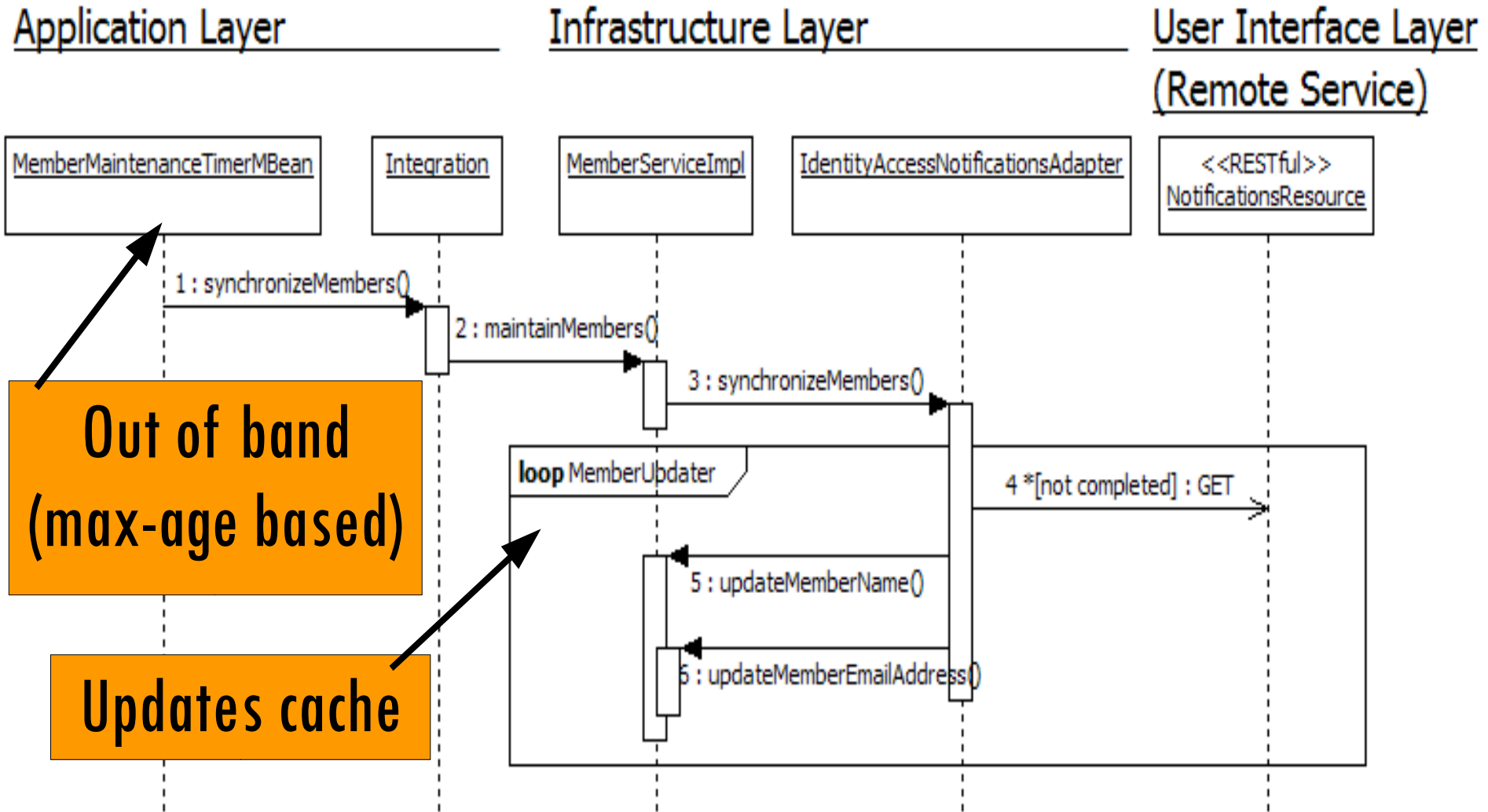
**RoleGroupAssigned**

**RoleGroupUnassigned**

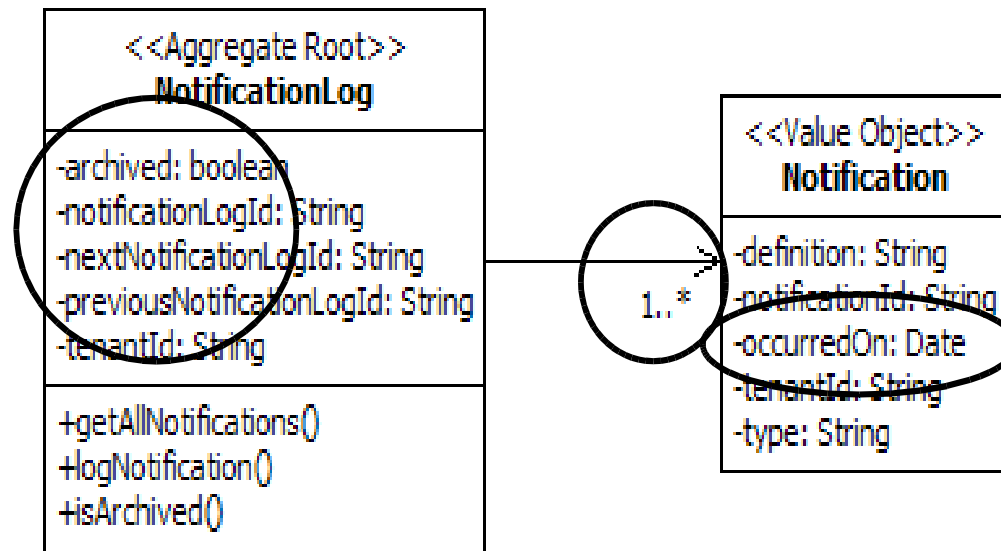
**RoleUserAssigned**

**RoleUserUnassigned**

# Agile PM Synchronization



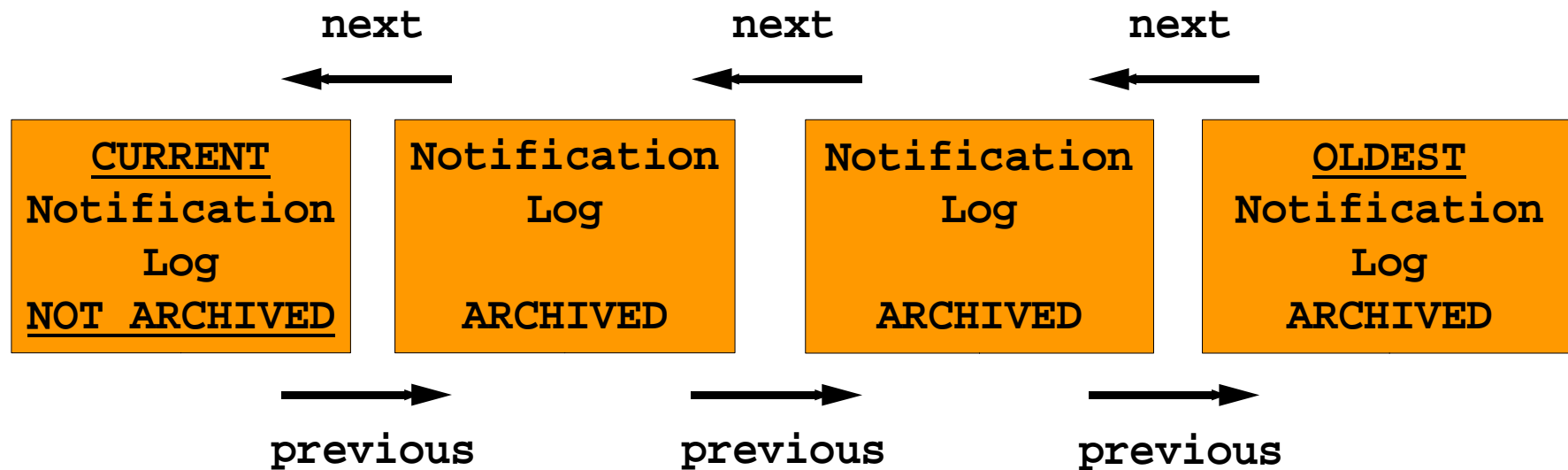
# Notification Logs



Series of logs from the beginning of time; NOT in Core Domain

Current working log and any number of archived logs

# Notification Referencing



Navigate through previous logs to find latest applied notification

Apply all newer notifications, navigating to current using next

# Feeds



`application/atom+xml`

```
<notifications
  tenantId="..."
  id="..."
  archived="false">
  ...
</notifications>
```

`application/xml`

**Can produce Atom-based notifications from NotificationLogs**


**Can produce custom notifications from NotificationLogs**

# NotificationResource (1)

```
@Path("/tenants/{tenantId}/notifications")
public class NotificationResource {
    @GET
    @Produces({ "application/xml" })
    public Response getCurrentNotificationLog(
        @PathParam("tenantId") String aTenantId,
        @Context UriInfo aUriInfo) {

        NotificationLog currentNotificationLog =
            this.getNotificationService()
                .getCurrentNotificationLog(new TenantId(aTenantId));

        // ...
        return tempResponse;
    }
}
```



RESTful notification service as a resource (OHS producing PL)



# NotificationResource (2)

```
@Path("/tenants/{tenantId}/notifications")
public class NotificationResource {
    @GET
    @Path("{notificationId}")
    @Produces({ "application/xml" })
    public Response getNotificationLog(
        @PathParam("tenantId") String aTenantId,
        @PathParam("notificationId") String aNotificationId,
        @Context UriInfo aUriInfo) {

        NotificationLog notificationLog =
            this.getNotificationService()
                .getNotificationLog(
                    new TenantId(aTenantId), aNotificationId);

        // ...
        return tempResponse;
    }
}
```

**Specific log, next/previous**

**May filter using parameter**

# HTTP GET Notifications

```
GET https://iam/tenants/{tenantId}/notifications
Accept: application/xml
```

---

```
HTTP/1.1 200 OK
Content-Type: application/xml
Link: <https://iam/tenants/C888.../notifications>;
      rel=self
Link: <https://iam/tenants/C888.../notifications/55F03810-...>;
      rel=previous
...
<notifications ...>
  <notification ...>
  </notification>
  ...
</notifications>
```

# HTTP GETting It All

HTTP/1.1 200 OK

Content-Type: application/xml

Link: <https://iam/tenants/C888.../notifications/72G31419-...>;  
rel=self

Link: <https://iam/tenants/C888.../notifications/46A0C283-...>;  
rel=next

Link: <https://iam/tenants/C888.../notifications/55F03810-...>;  
rel=previous

Cache-Control: max-age=3600

...

<notifications ...>

<notification id="A9D234C6-96EF-4B9A-BAC9-F720A2115B3B">

</notification>

...

</notifications>




Track latest locally

# Incoming Notification Tracker

```
private IncomingNotificationTracker tracker;
//...
private void applyNotification(... aNotification) {
    // ...
    this.getTracker()
        .recordMostRecent(
            aNotification.getId(),
            aNotification.getType(),
            aNotification.getOccurredOn());
}
```

# Notification Element

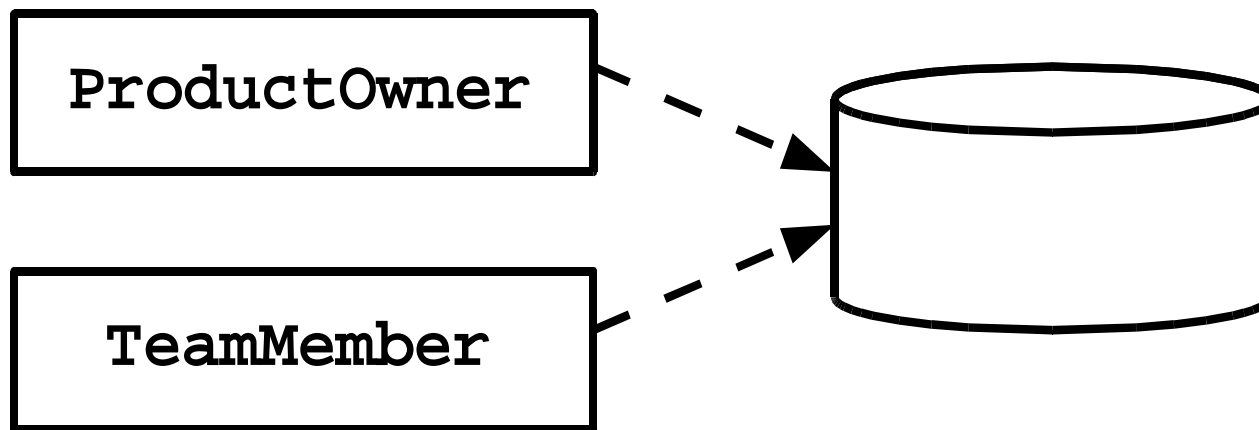
```
<notification type="personNameChanged"
  occurredOn="2010-08-19T16:38:06-06:00"
  id="A9D234C6-96EF-4B9A-BAC9-F720A2115B3B">
  <personNameChanged>
    <tenantId>C888F4B2-71F0-48D2-B4A3-0A44BABFA2E1</tenantId>
    <username>zoe</username>
    <firstName>Zoe</firstName>
    <lastName>Jones-Doe</lastName>
  </personNameChanged>
</notification>
```



Outer notification element contains common information

Child reflects Domain Event from originating Bounded Context

# Agile PM Shared Values



MemberService (Domain) caches single instance in database

Anticorruption Layer (notifications adapter) updates

# Zoe's New Name

```
ProductOwner productOwner =
    DomainRegistry
        .getMemberService()
        .getProductOwner(tenant, "zoe");

assertNotNull(productOwner);

assertEquals(productOwner.getName(),
    "Zoe Jones-Doe");

assertEquals(productOwner.getEmailAddress(),
    "zoe@shiftmethod.com");
```

# Autonomous Services



**Notifications from Domain Events of all three Bounded Contexts**

**Notifications allow seeding Model from “beginning of time”**

**Notifications allow recovery from any service down-time**



# Summary



**For Business: combine RESTful SOA with Domain-Driven Design**

**Strategic Modeling patterns: DDD's “better half” for RESTful SOA**

**Tactical Modeling patterns: Domain Events key building block**

**Integrating Bounded Contexts: RESTful + Hypermedia**

# Contact

Vaughn Vernon  
vvernon@shiftmethod.com

## Questions?

Please fill out evaluation forms!



Copyright © 2008-2010 ShiftMETHOD. All rights reserved.