Bård Lind and Totto - Objectware

Enterprise Domain Repository

Community Corner - JavaOne 2008

Enterprise Domain Repository

Suppliers

Customer

Internal Users

Partner

Customer Repository Service

CRM

Legacy

Billing

Inventory

Customer

Customer

Customer

Delivery-address

Billing-address

EDR is a "pattern" to reduce complexity when domain objects exist in several distinct back-end systems

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

EDR - Agenda

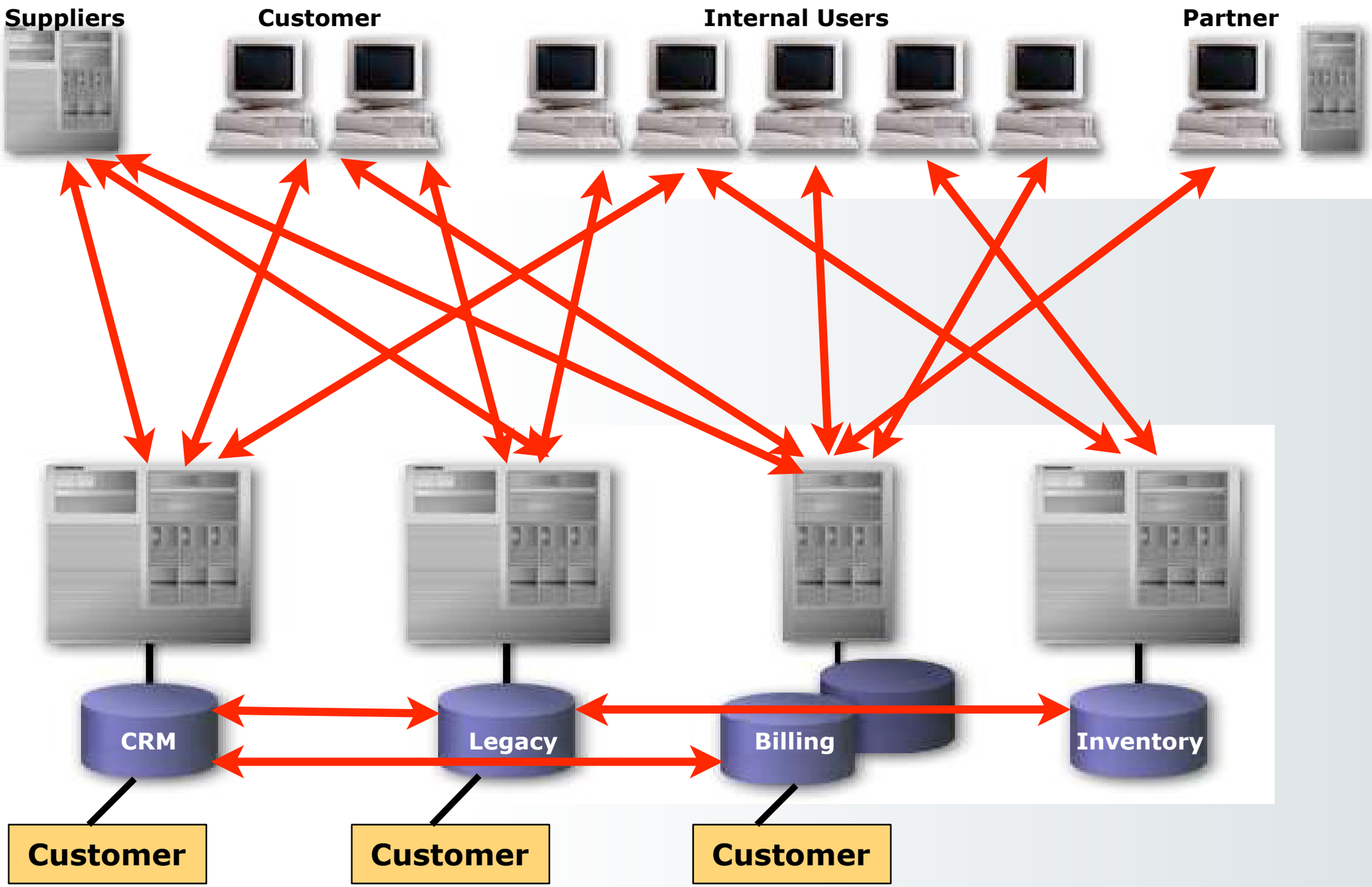
- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

Typical Integration Challenges

- Disjoint and overlapping data
- Maintaining data-quality

- ... performance...

Familiar with this ....



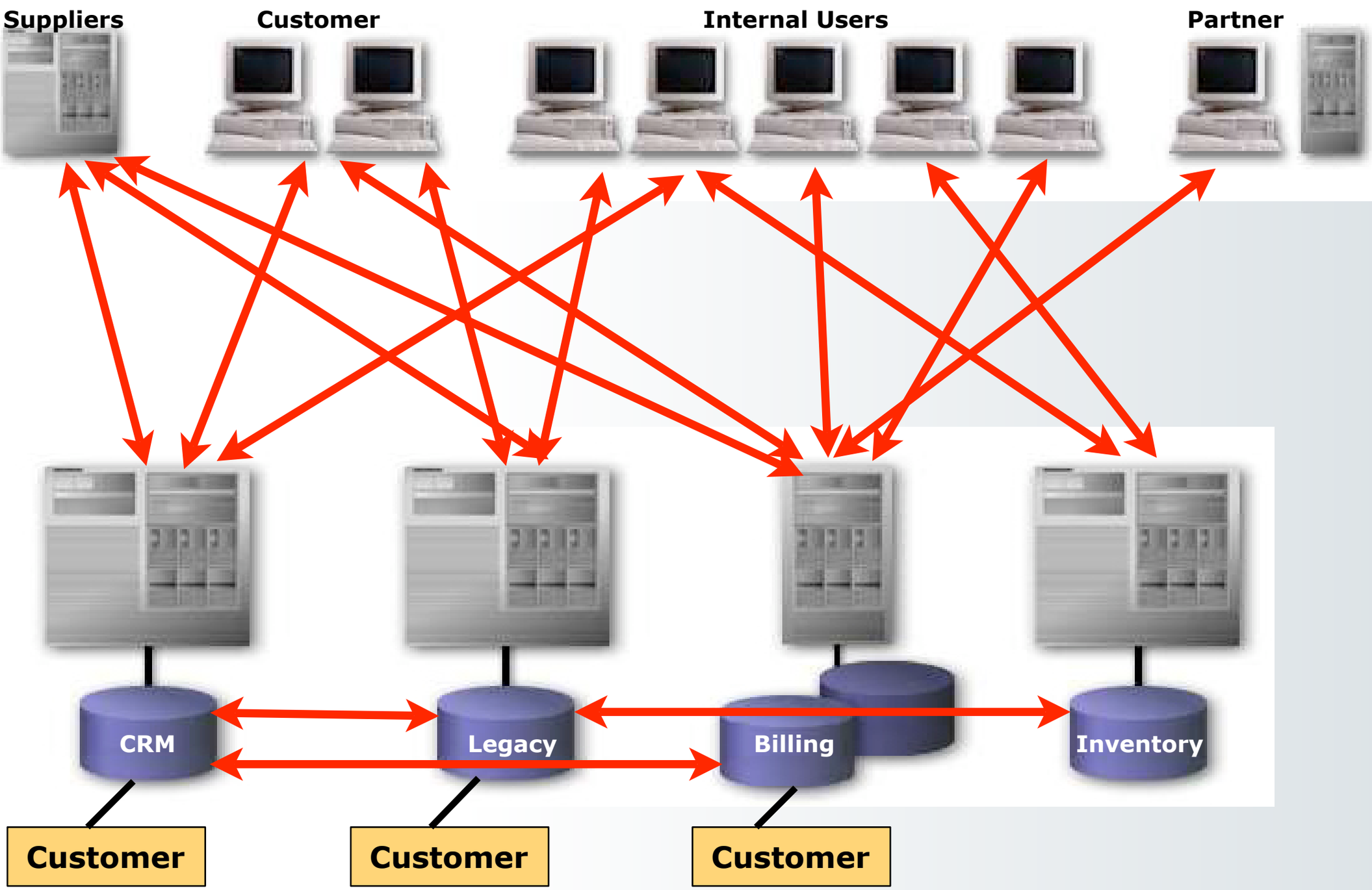
Disjoint and overlapping data

- What is the definition of Customer?
  - Legacy? CRM? Billing?
  - Combined Customer?
- Data quality suffers
  - What´s the right address?
- Same data exist in more than one back-end system
- The reason is ...

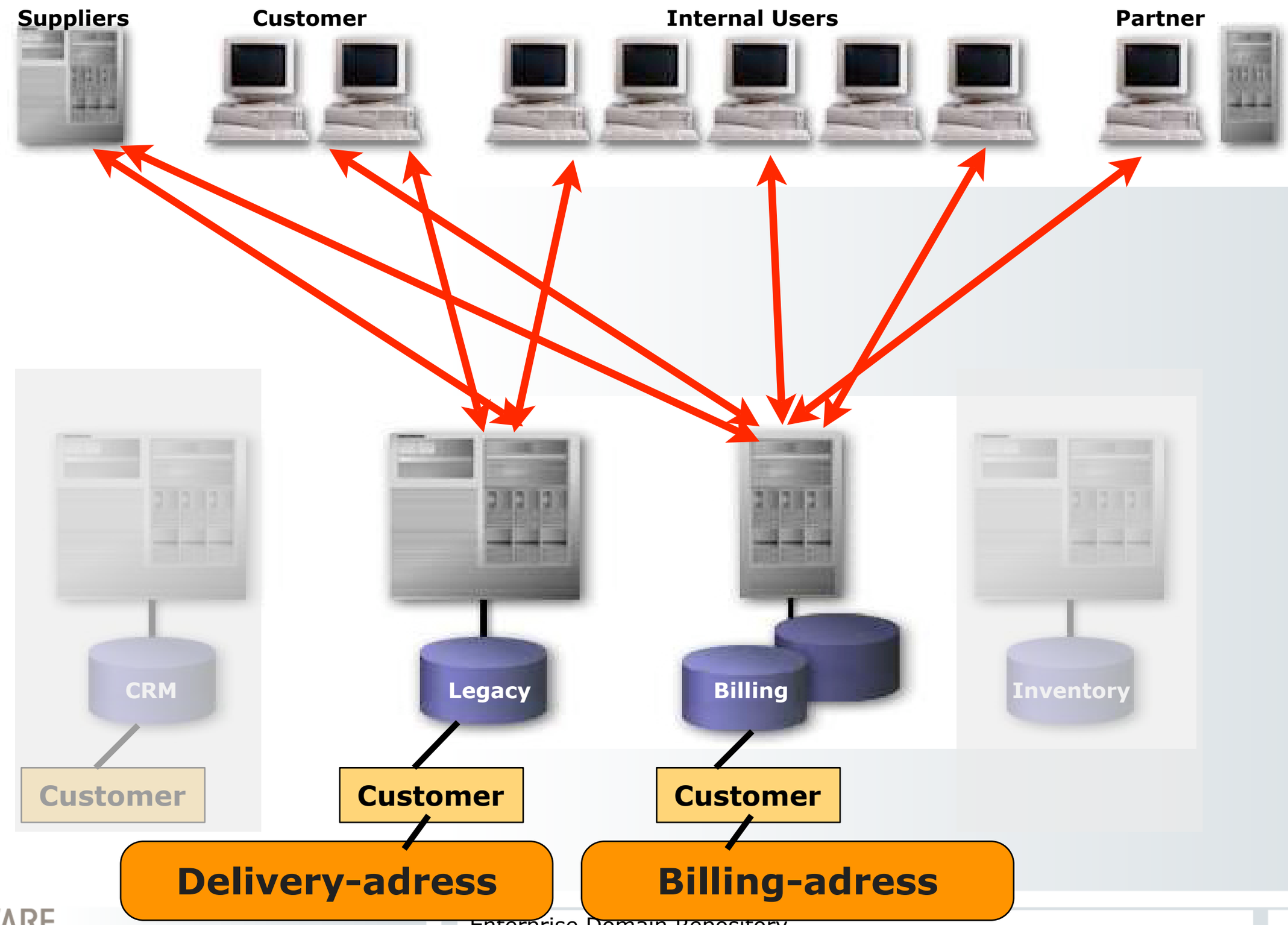
… we do need multiple back-ends

- One back-end system delivers great functionality for a single group of users
- These back-ends has a purpose - cost/value of their deliverables favors their existence
- Old back-end systems tend to survive - can´t get rid of those

Complex address example:



Complex address example:



Maintaining data-quality

- Typical Data Master problem
  - “Who has the right to update the data”
  - Trouble deciding how to merge “simultaneous” updates
  - Who is master for each data-element in an domain object?
- Data quality suffers because...
  - it is difficult to update data
  - data entered are in incorrect format

Performance

- Retrieval of data from slow back-end system
- Building of Domain Object might be lengthy

Kjartan

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

Kjartan

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

The 4 SOA Service categories

The 4 SOA Service categories

Human to Application services

Application to Application services

Aggregated Core services

Core services

The 4 SOA Service categories

CustomerInformation

Application to Application services

Aggregated Core services

Core services

The 4 SOA Service categories

CustomerInformation

CustomerCreditCheck

Aggregated Core services

Core services

The 4 SOA Service categories

CustomerInformation

CustomerCreditCheck

CustomerOverview

Core services

The 4 SOA Service categories

CustomerInformation

CustomerCreditCheck

CustomerOverview

Customer

PaymentHistory

Order

The 4 SOA Service categories

Cost of change

CustomerInformation

CustomerCreditCheck

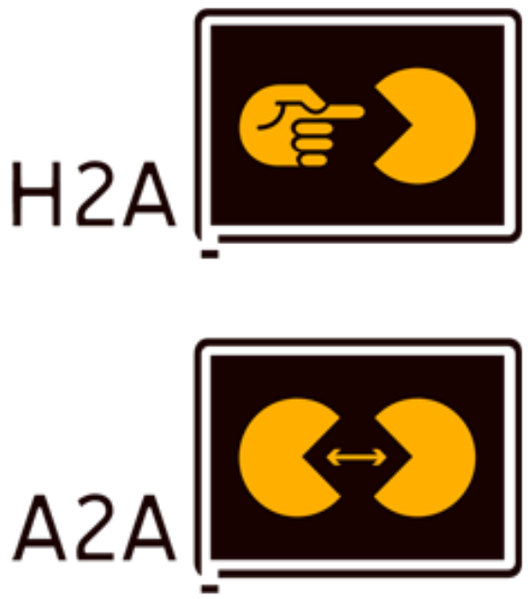
CustomerOverview

Customer

PaymentHistory

Order

The 4 SOA Service categories



H2A CustomerInformation

A2A CustomerCreditCheck

ACS CustomerOverview

CS Customer

CS PaymentHistory

CS Order

Cost of change

Need for agility

The 4 SOA Service categories

CustomerInformation

CustomerCreditCheck

CustomerOverview

Enterprise Domain Repository

Customer

PaymentHistory

Order

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

OBJECTWARE

Enterprise Domain Repository

Suppliers

Customer

Internal Users

Partner

Customer Repository Service

CRM

Legacy

Billing

Inventory

Customer

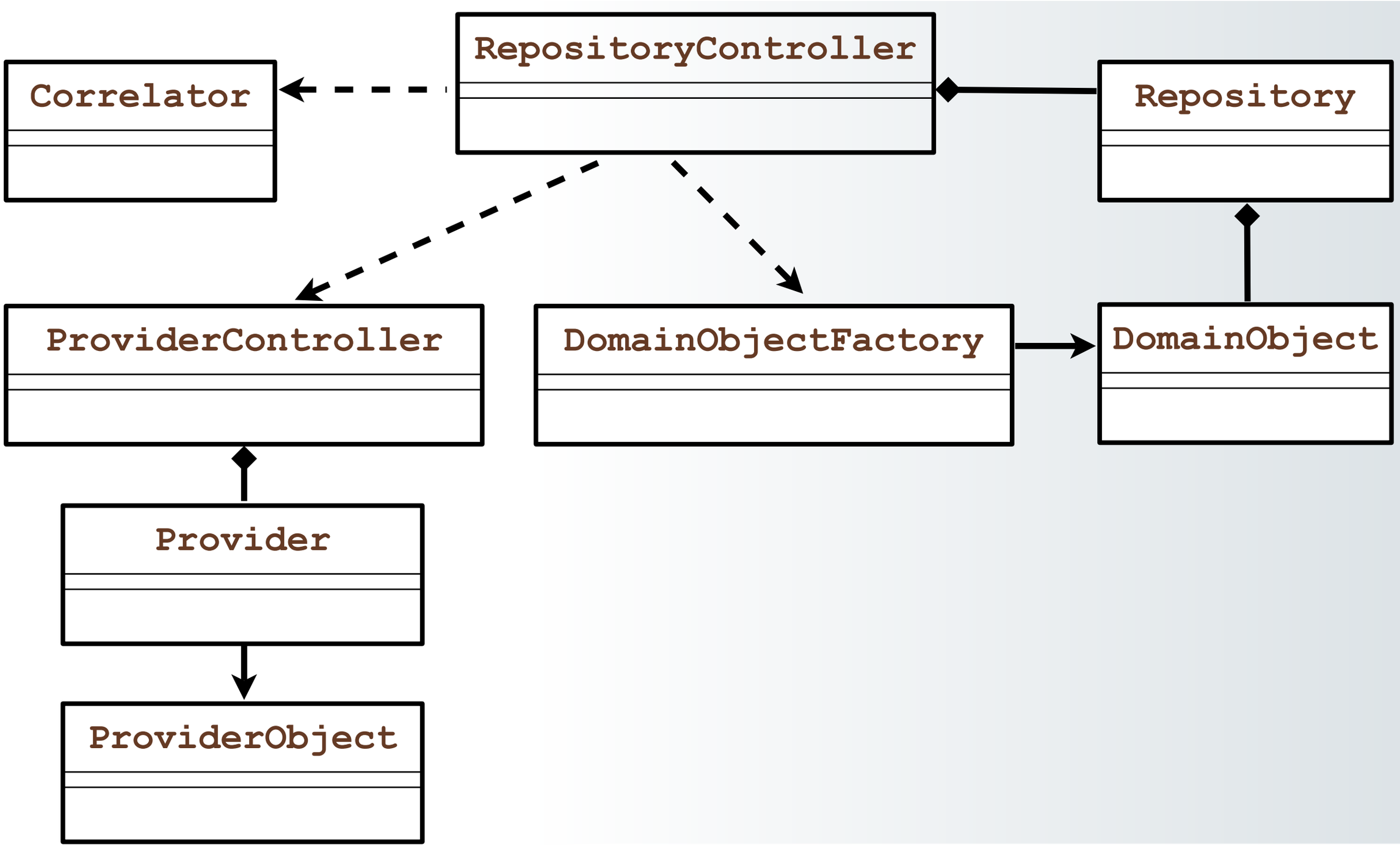
Customer

Customer

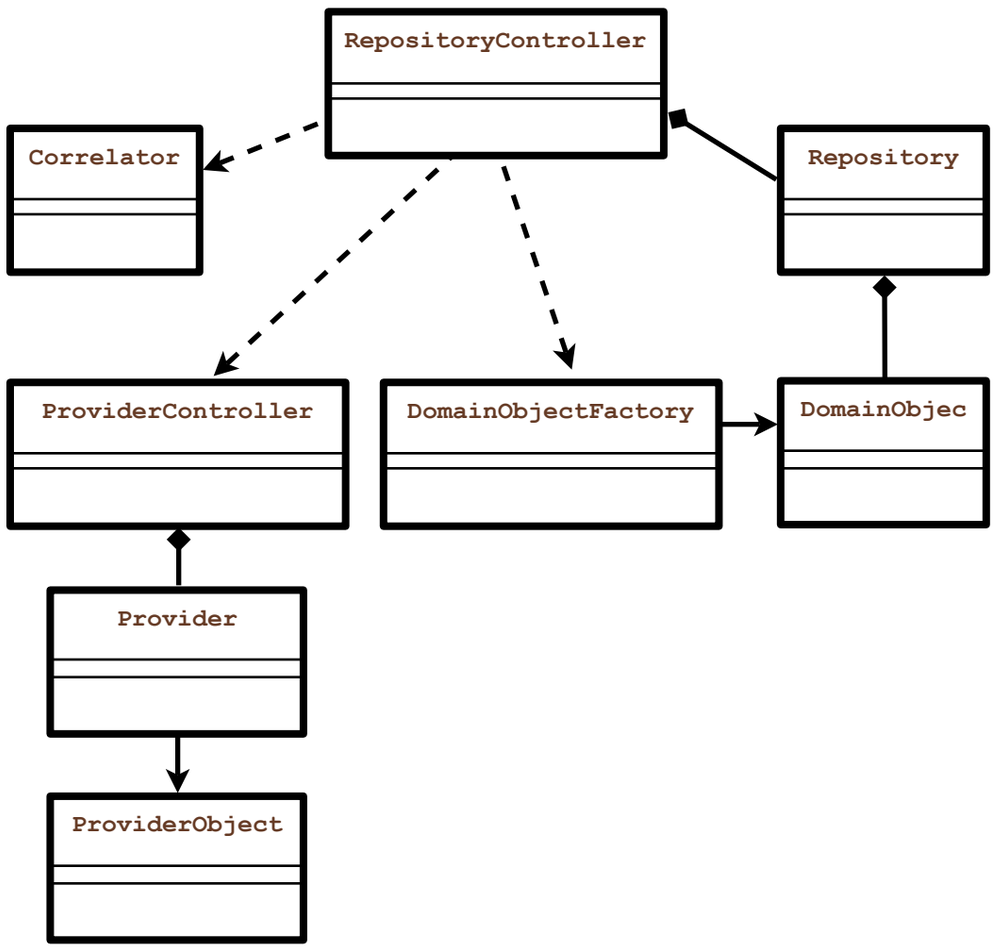
Delivery-address

Billing-address

Enterprise Domain Repository

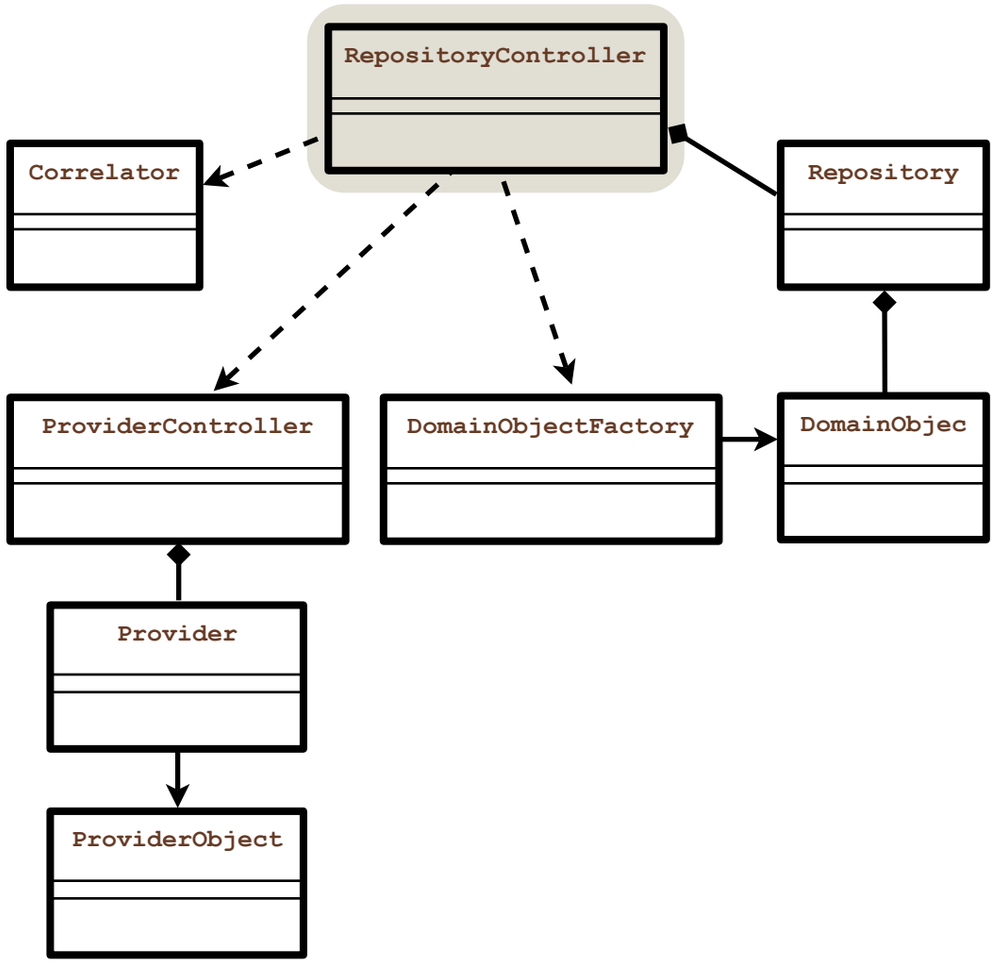


RepositoryController



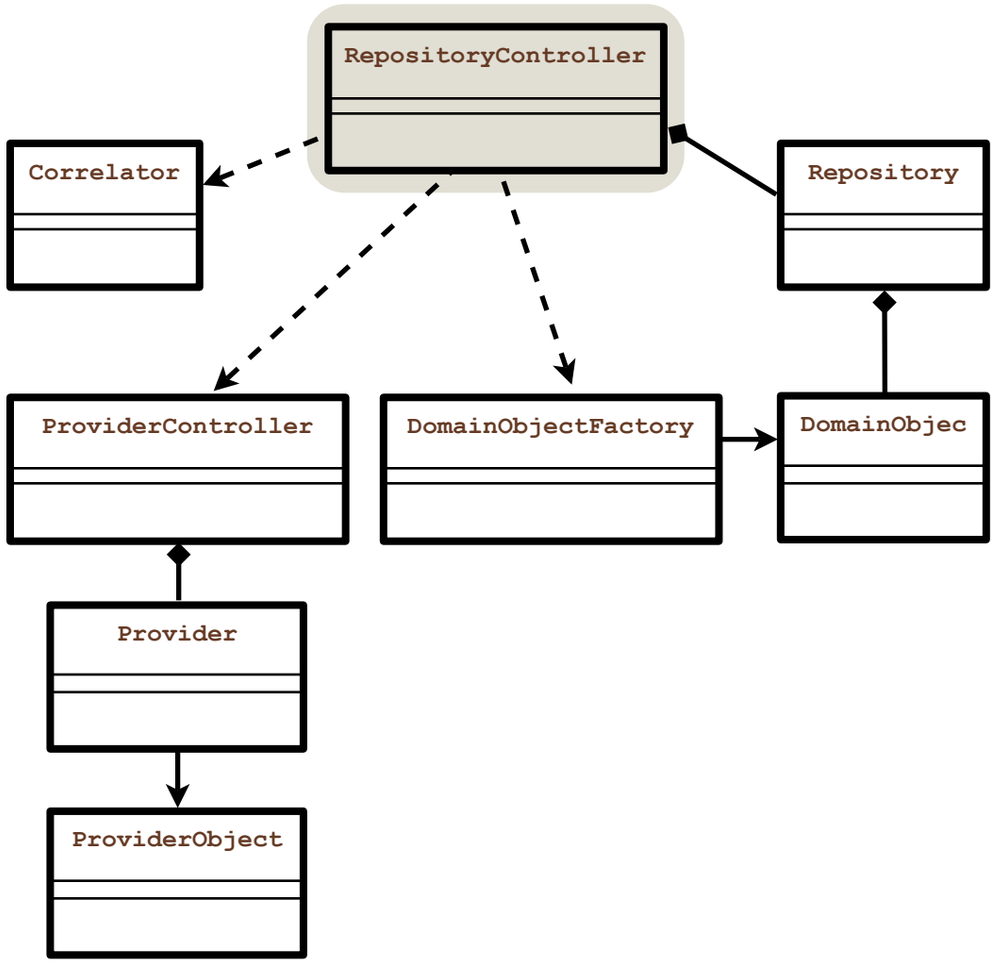
- The main public interface for the Enterprise Domain Repository
- Provides a clean repository analogy
- Retrieves and persist Domain Objects through its providers
- Manages the repository

RepositoryController



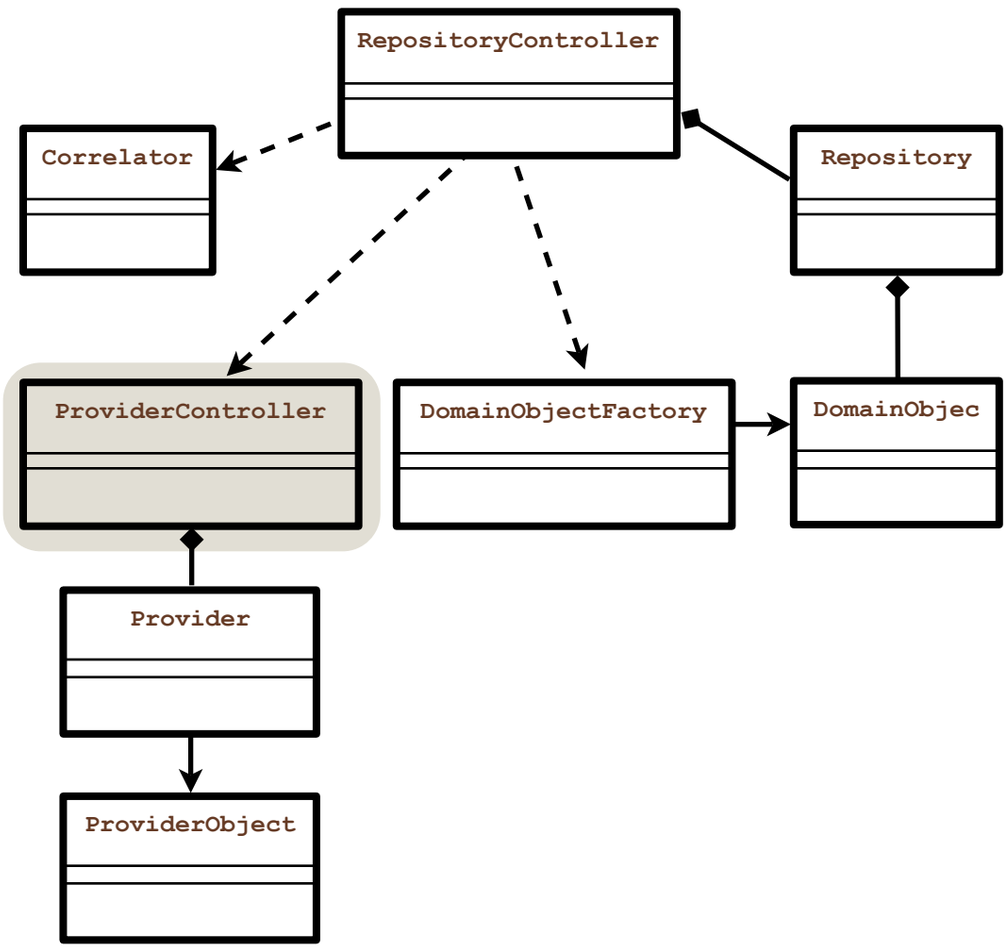
- The main public interface for the Enterprise Domain Repository
- Provides a clean repository analogy
- Retrieves and persist Domain Objects through its providers
- Manages the repository

ProviderController



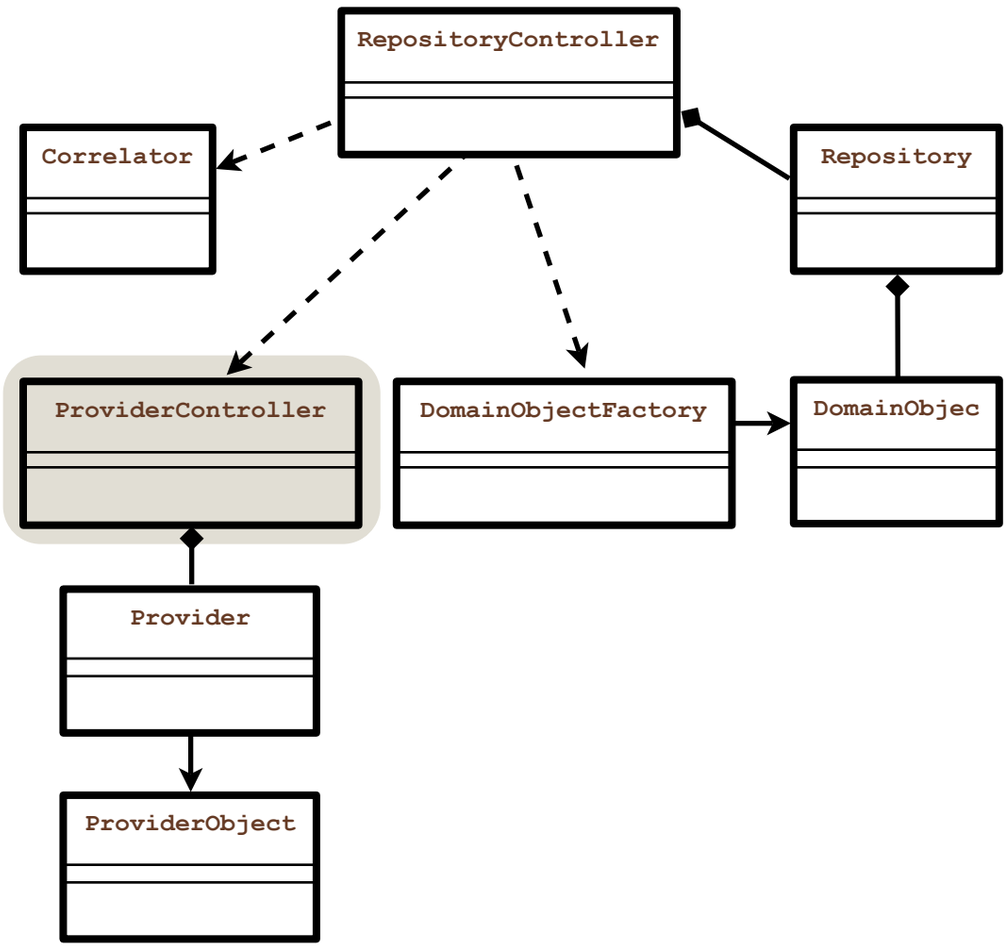
- Keep track of registered providers and execute cross-provider calls
- Supports the RepositoryController

ProviderController



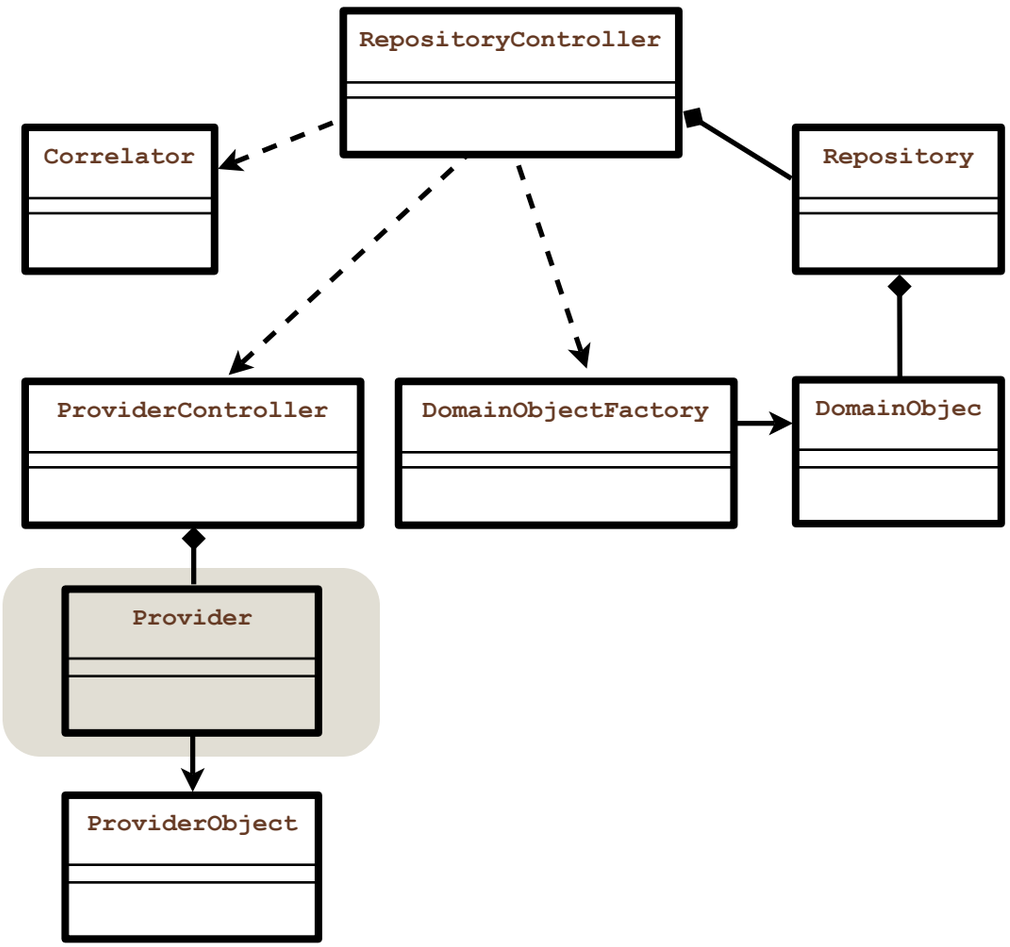
- Keep track of registered providers and execute cross-provider calls
- Supports the RepositoryController

Provider



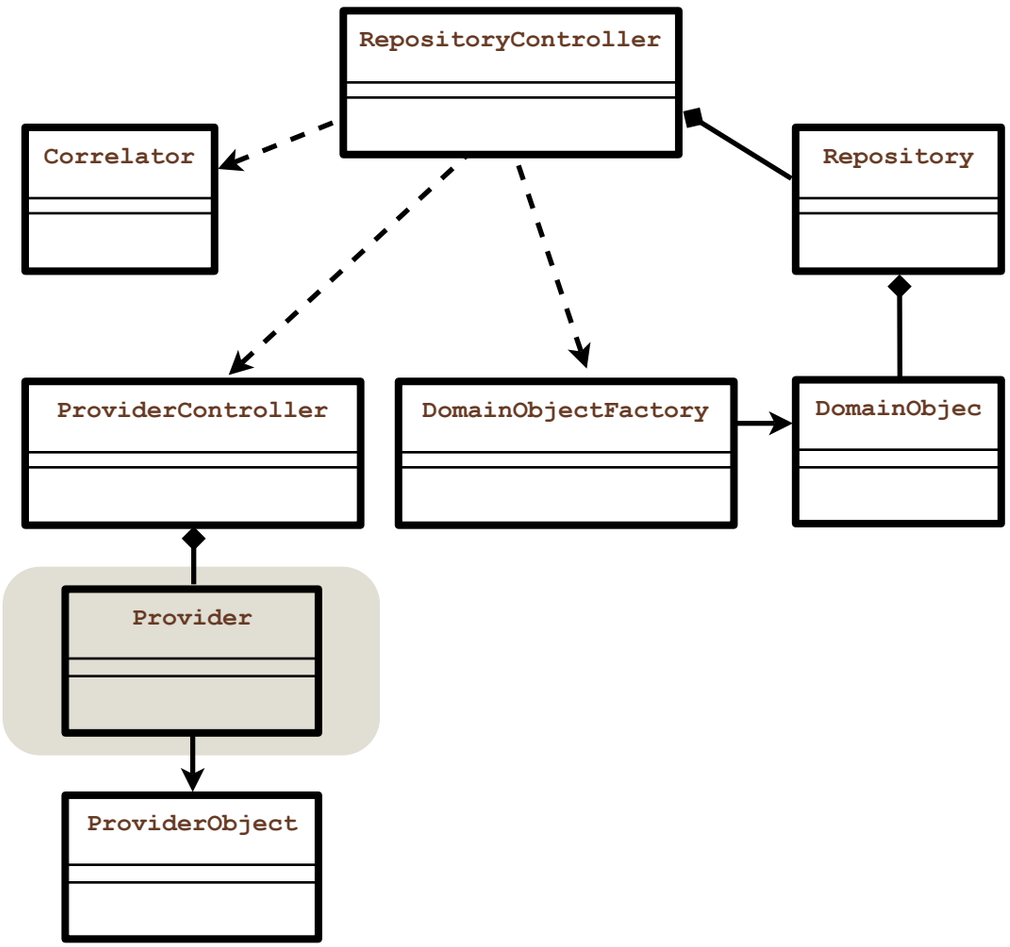
- The provider represents one back-end system and delivers Provider Objects to the ProviderController
- Uses Eric Evans Repository Pattern
  - *Encapsulates all complexity in calling the underlying back-end system*
- Its up to each provider to decide its cache strategy

Provider



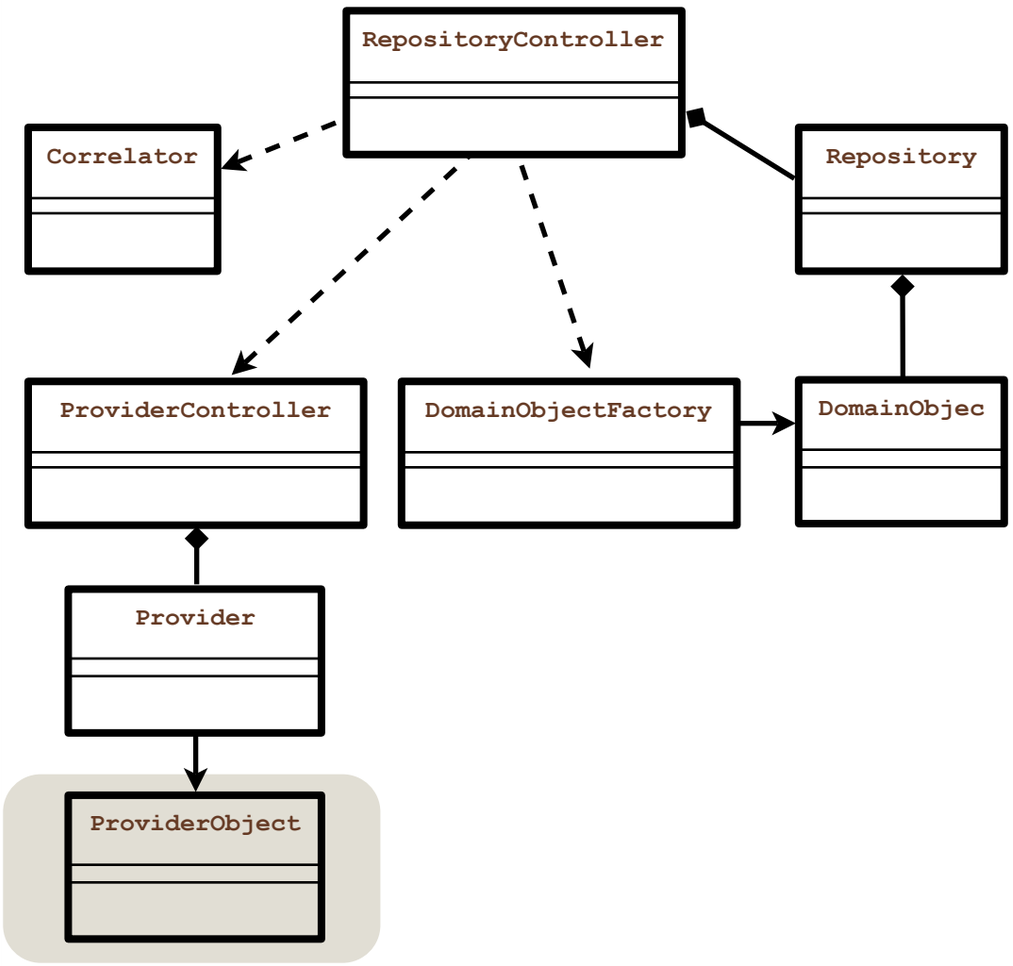
- The provider represents one back-end system and delivers Provider Objects to the ProviderController
- Uses Eric Evans Repository Pattern
  - *Encapsulates all complexity in calling the underlying back-end system*
- Its up to each provider to decide its cache strategy

Provider Object



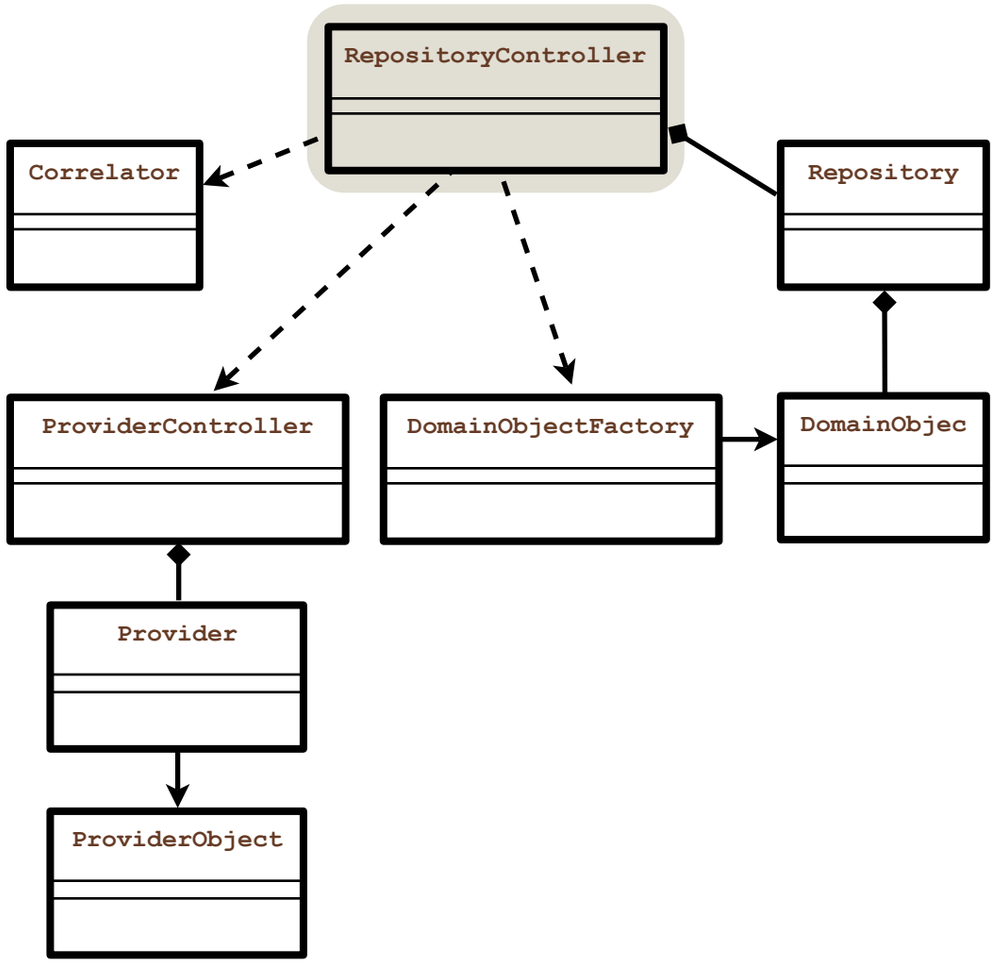
- A representation identical to, or tightly coupled to the semantics of the underlying system
- May be reasonable to implement the Provider Object using Eric Evans Aggregate Pattern in order to hide unnecessary complexity

Provider Object



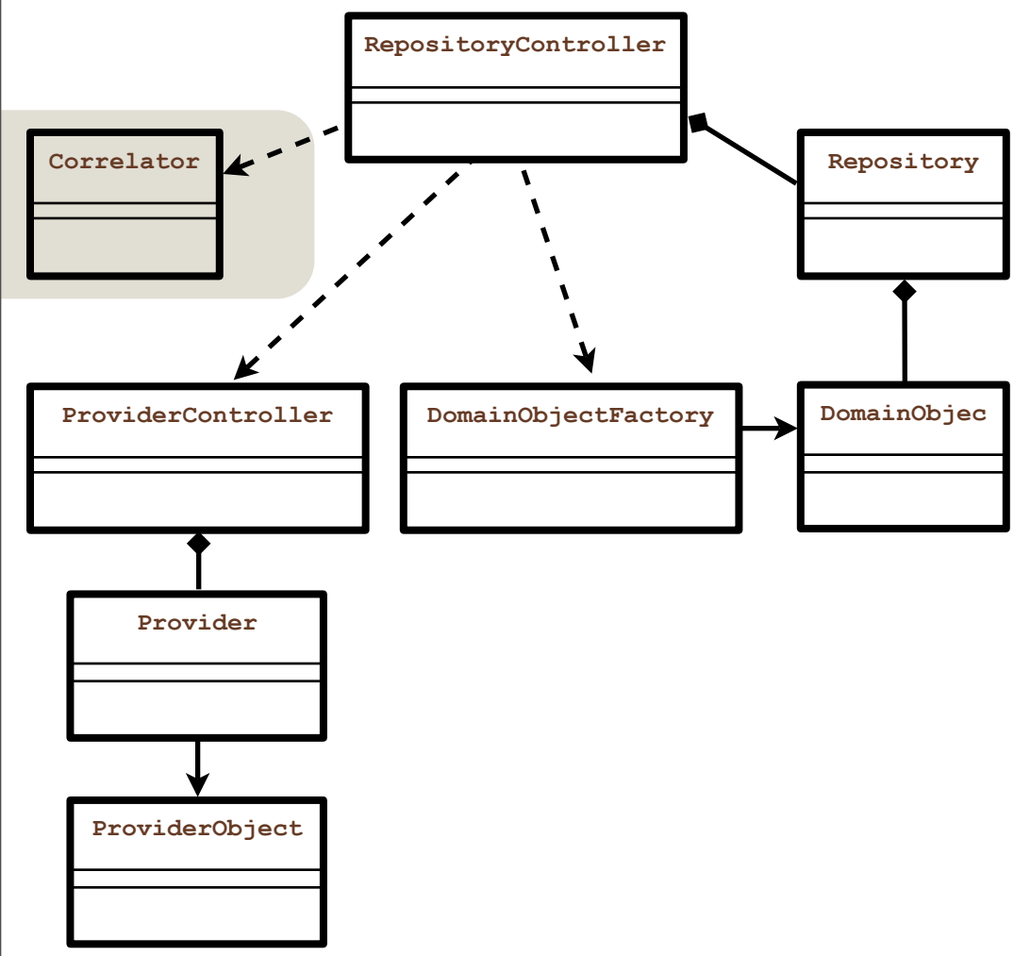
- A representation identical to, or tightly coupled to the semantics of the underlying system
- May be reasonable to implement the Provider Object using Eric Evans Aggregate Pattern in order to hide unnecessary complexity

Correlator



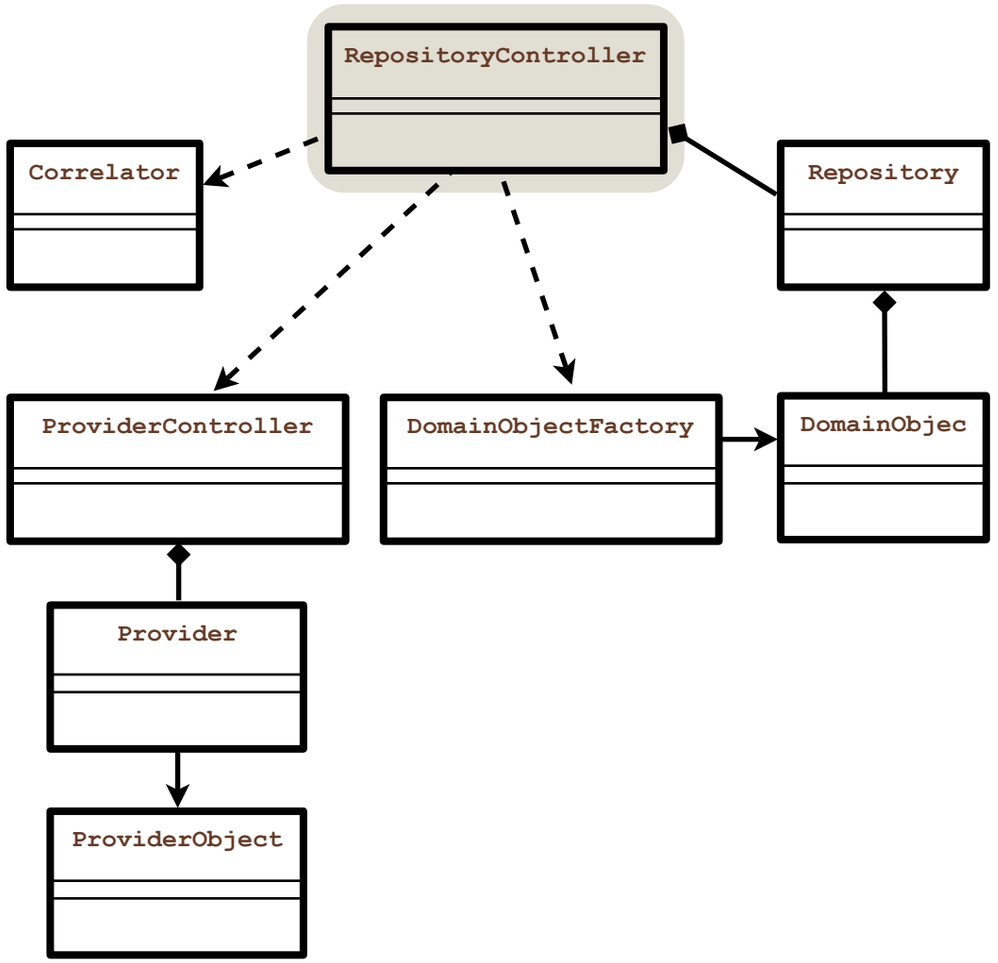
- Supplying id correlation services to the Repository Controller
- A vital part when creating the Domain Object
  - Correlate the Provider Objects and provide a logical id

Correlator



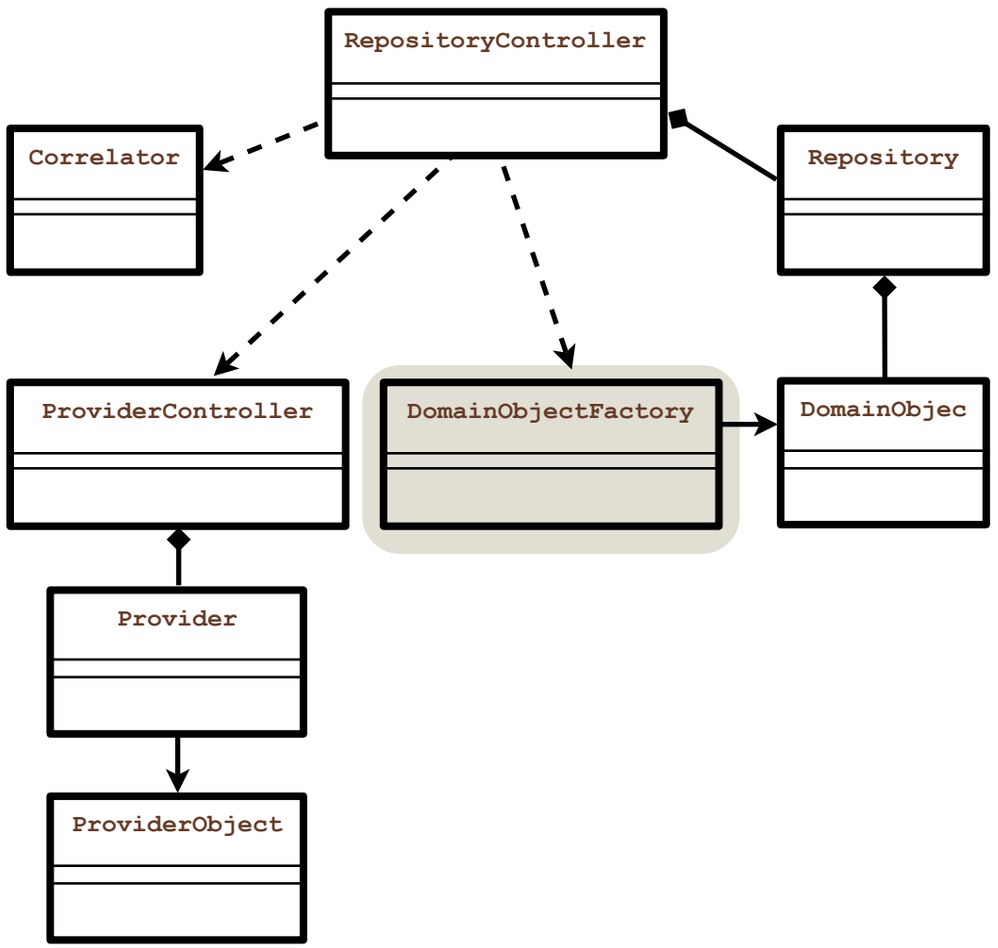
- Supplying id correlation services to the Repository Controller
- A vital part when creating the Domain Object
  - Correlate the Provider Objects and provide a logical id

Domain Object Factory



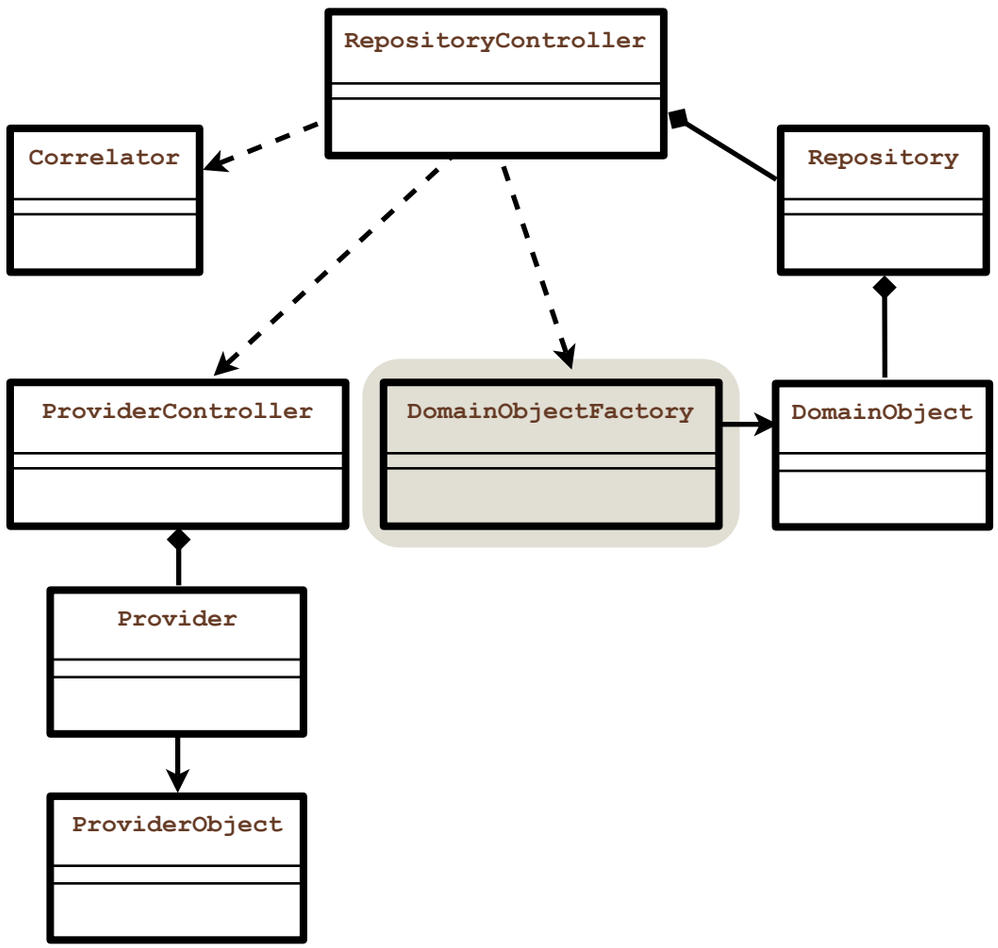
- Constructs domain object based on provider objects
- Responsible for data mapping between provider objects and domain objects
- There are several possible strategies to perform the data mapping
- The data mapping strategy is not part of EDR

Domain Object Factory



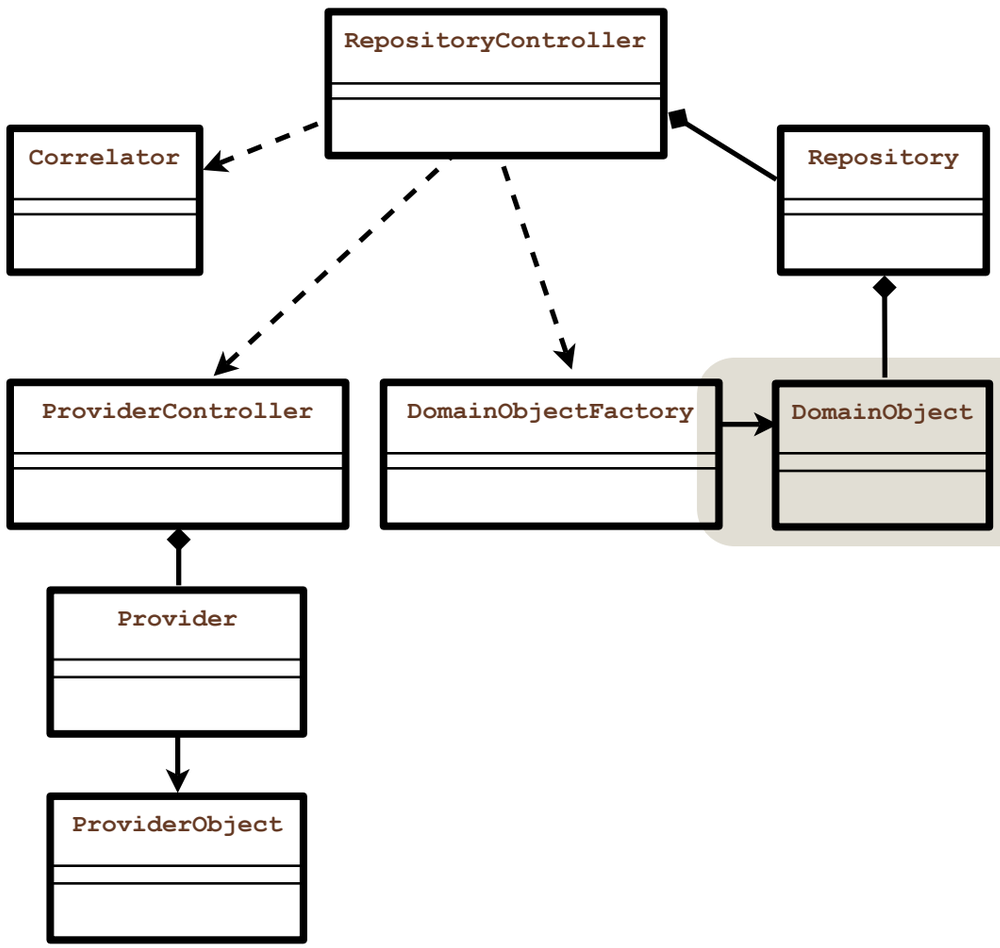
- Constructs domain object based on provider objects
- Responsible for data mapping between provider objects and domain objects
- There are several possible strategies to perform the data mapping
- The data mapping strategy is not part of EDR

Domain Object



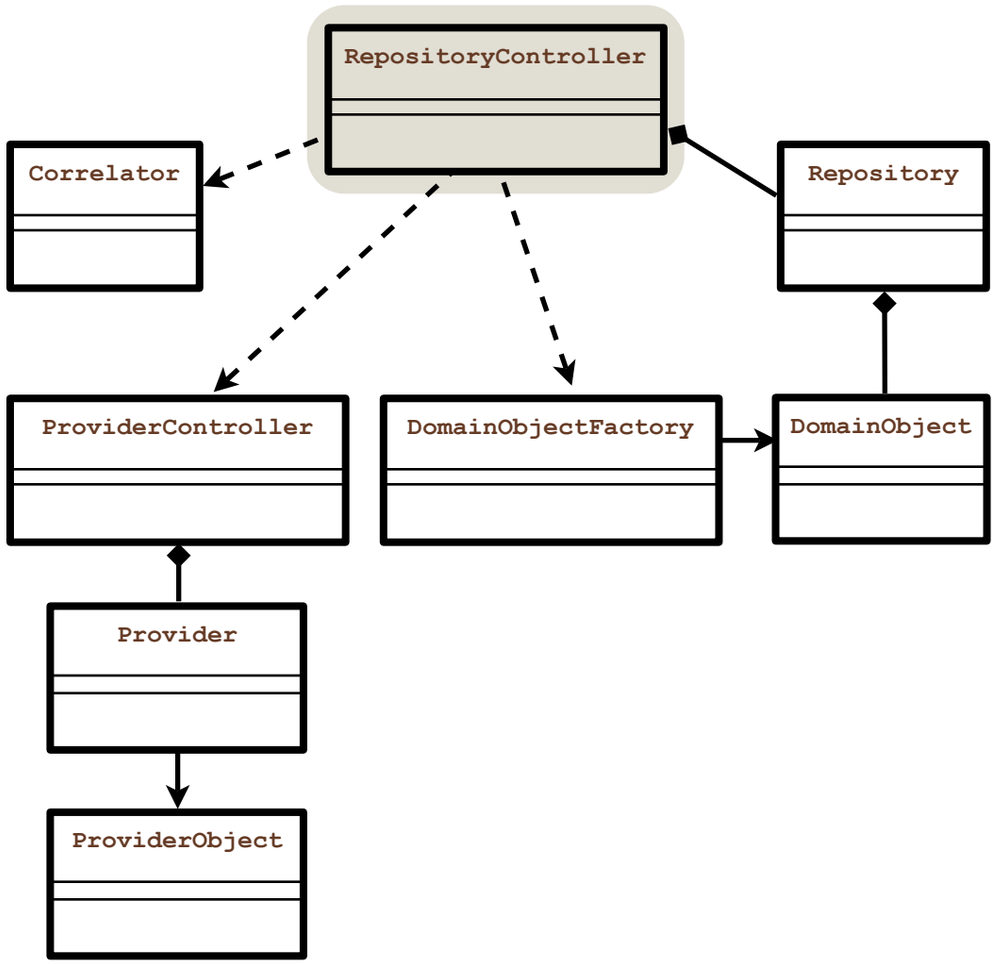
- Contains data from different back-end systems
- Provides business functionality
- Can be serialized and passed to the client
- Enables the client to perform operations on the actual domain object

Domain Object



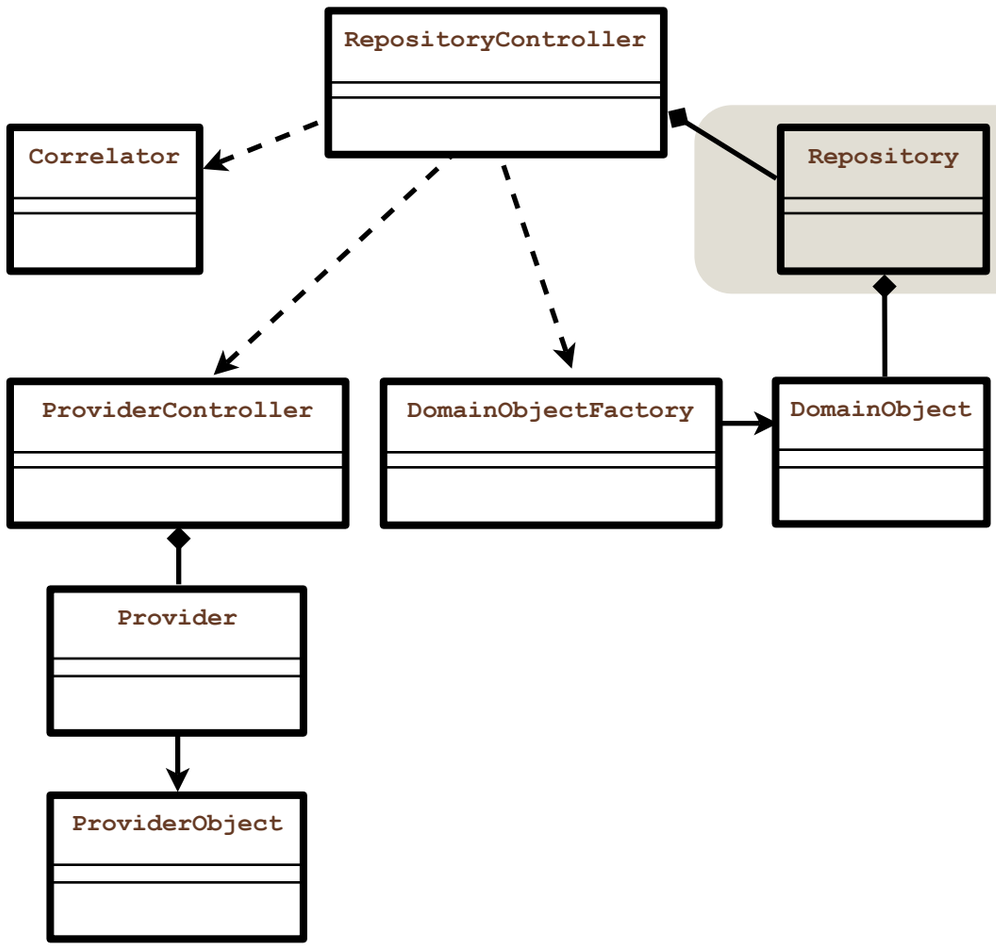
- Contains data from different back-end systems
- Provides business functionality
- Can be serialized and passed to the client
- Enables the client to perform operations on the actual domain object

Repository



- The Repository is responsible for storing the Domain Objects in a persistent or non-persistent store
- May be in-memory cache, file/xml-based persistence or a database
- May prove useful to have a persistent store for scaling and better performance

Repository



- The Repository is responsible for storing the Domain Objects in a persistent or non-persistent store
- May be in-memory cache, file/xml-based persistence or a database
- May prove useful to have a persistent store for scaling and better performance

Bård

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

Bård

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

What EDR solves

- Performance
- Evolvability of your system
- Predictability
- Control of your data
- Maintainability

Performance

- Slow back-ends
- Complex Domain Objects
- “Fire and forget” data update
  - Async update
- Faster data update
  - Update w/o need to directly calling provider

Evolvability of your system

- Loose coupling
  - Hide versioning behind provider
  - EDR can make working with older back-ends easier by separation of concern

Predictability

- Back-end down-time hurts less
  - Update even if a back-end system is down
  - Information is not lost
- Data-quality
  - Consistent data mapping

Control of Data

- Data Mapping
  - Data-mastering can be split into its own unit
- Record difference in data
  - Same data has inconsistent values in different back-ends
  - Automatic update
  - Moderate
  - Log

Maintainability

- SLA for each back-end
- Back-end update is less painful
  - Update provider only
  - Less downtime on your system
- Client is connected to less systems

Kjartan

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

Kjartan

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

Why use EDR?

- Business definition of a Domain Object requires data from disparate back-end systems
- Integrating multiple back-end systems cause high complexity
- Integrating directly from clients towards back-end systems reduces system evolvability

Why use EDR?

- Services exposed with higher granularity and larger responsibilities grow too complex over time
- Helps keeping your domain data consistent
- Its up to the service to decide which data to use and where to update - Not the client

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

Alternatives to EDR

- There are alternatives to parts of what EDR solves
- We have found no alternatives handling all the challenges EDR solves - Please enlighten us in the Q&A section

Bård

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

Bård

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

Our experience using EDR

- Used in both .Net and Java projects
- 4 Cases
- Use Correlation - even for read only operations
  - Because EDR can be addictive - the customer might want updates as well...
- Problem making the customer understand that the data is always a copy of the real data, and can be cached
- "Replication conflict" - resolve

References

- Eric Evans - “Domain Driven Design”

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

EDR - Agenda

- Purpose of this presentation
- Typical integration challenges
- The 4 SOA service categories
- Enterprise Domain Repository (EDR)
- What EDR solves
- Why use EDR
- Alternatives to EDR
- Our experience using EDR
- Q&A

Q&A

Enterprise Domain Repository

https://edr.dev.java.net

Totto(totto@java.no)
Bård Lind (bard.lind@objectware.no)

OBJECTWARE

Enterprise Domain Repository

Suppliers

Customer

Internal Users

Partner

Customer Repository Service

CRM

Legacy

Billing

Inventory

Customer

Customer

Customer

Delivery-address

Billing-address