An Open-EDI prototype based on UML, CORBA and Java

Lesson learned

Presented at ISO/IEC SC32 WG1 Ottawa

22. September 1998

Per Myrseth
per@nr.no
Norwegian Computer Center
www.nr.no

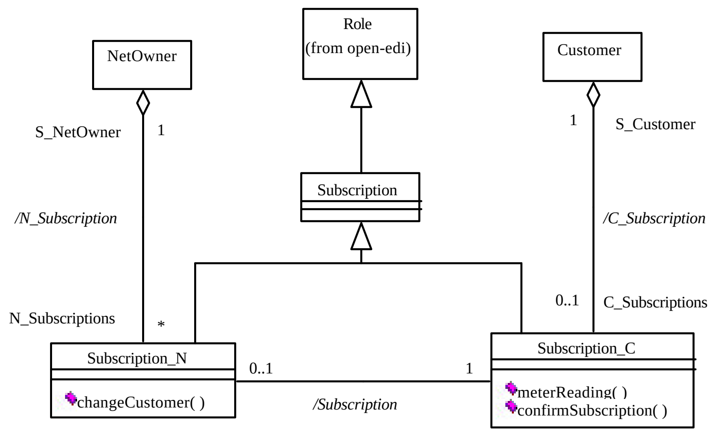
Target: Formal specification and automatic system generation

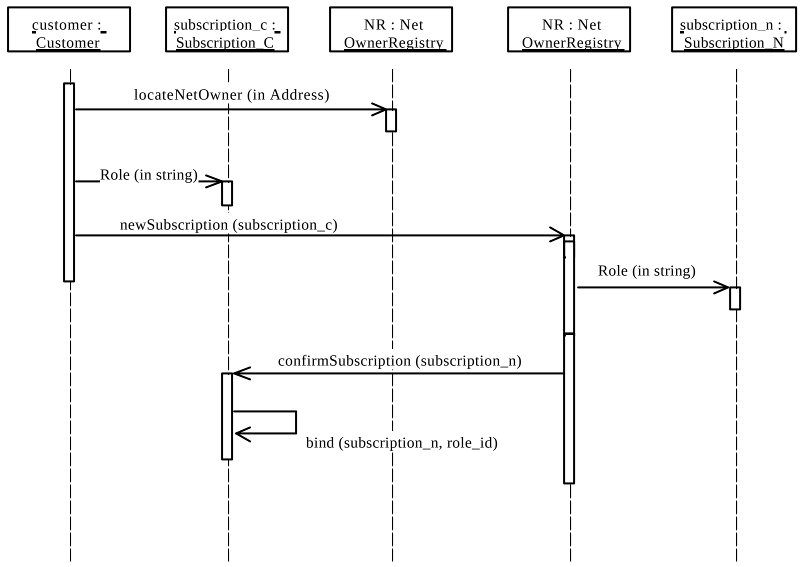
An old idea within system developement, but so far successful only under very limited circumstances.

How we tried to reach our target

- Used ”state-of-the-art” FDT toolkit. Rational Rose and UML notation
- Used ”state-of-the-art” technology, Corba and Java for implementation

Model in UML





customer : Customer
subscription_c : Subscription_C
NR : Net OwnerRegistry
NR : Net OwnerRegistry
subscription_n : Subscription_N
locateNetOwner (in Address)
Role (in string)
newSubscription (subscription_c)
Role (in string)
confirmSubscription (subscription_n)
bind (subscription_n, role_id)

Objectmodel

Message sequence diagram
(+State diagram)

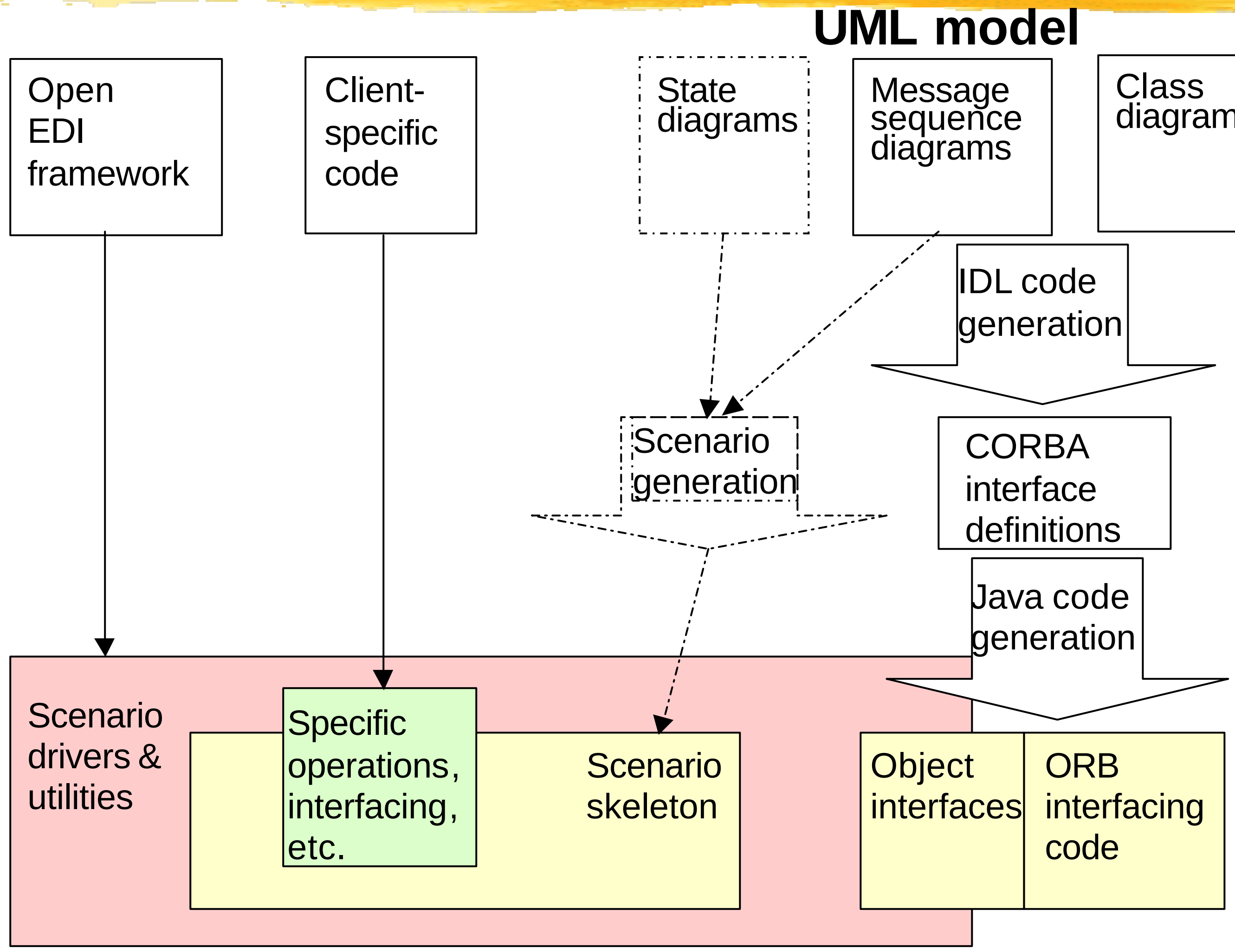
FSV

BOV

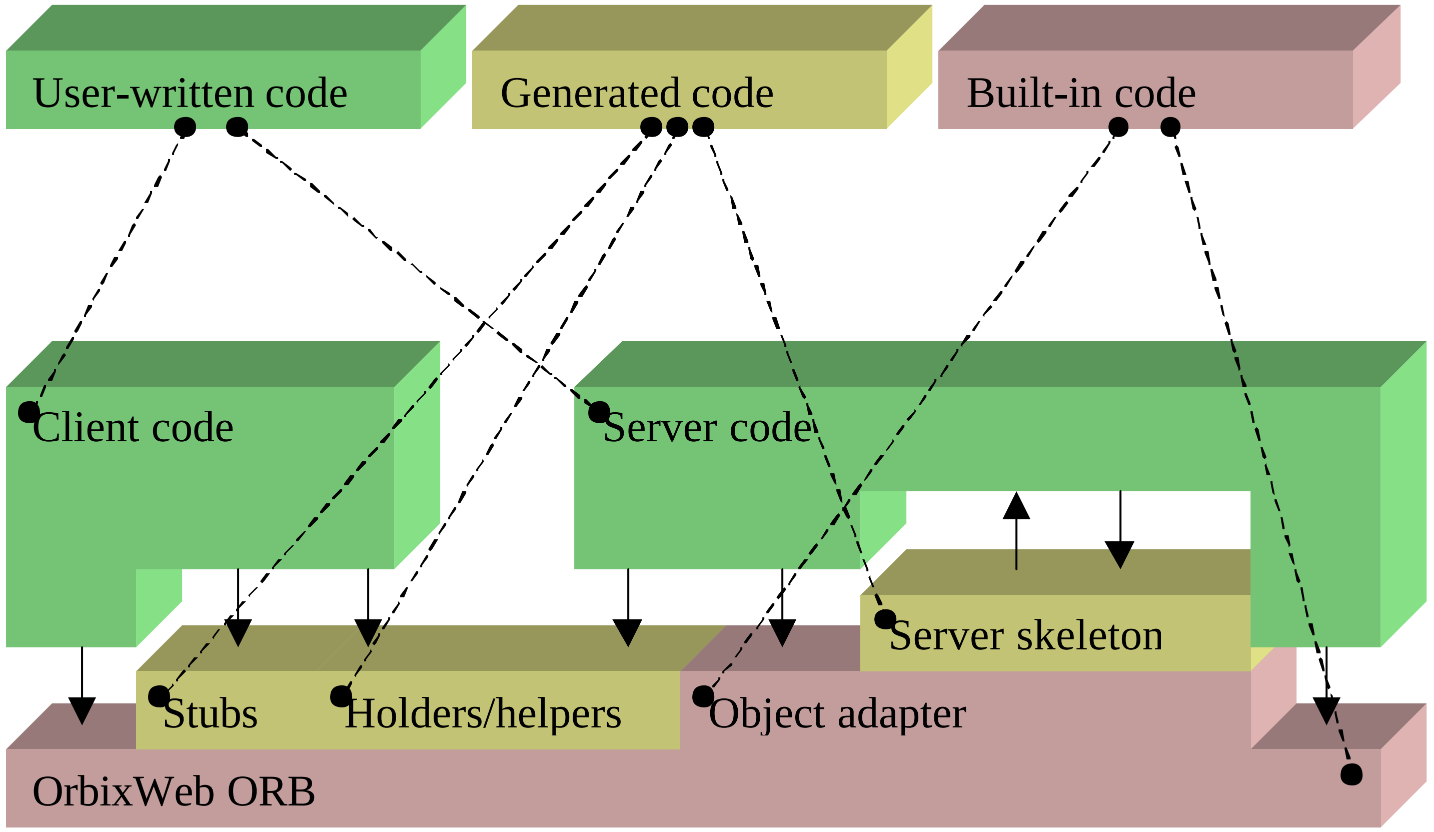
NR

Norsk Regnesentral
Norwegian Computing Center

From model to usable code



The structure in a CORBA/Java-application



Modeling Open-EDI scenarios with Rational Rose and using UML notation

Some findings:

⌘ Rational Rose is primarily designed for use in software development projects, not to model BOV.

⌘ Several objects could act in concert to implement a single role.

⌘ All actions must take place in concrete objects, where each is always executed on a specified host computer and under the control/responsibility, and on behalf of precisely one real-world actor.

FDT's for asyncronous versus syncronous system integration

Having in mind that an EDI message counterpart within distributed object systems is the method call.

At what time do we have to choose asyncronous versus syncronous integration?

- ☆ Reference model level (Open-EDI, eCo, Zackman)
- 🕐 FTD level
- 🕑 Design level
- 🕒 SW implementation tool

Message modeling versus state modeling

Difference between:

- Message modeling
- State modelling of agents/programs to act upon data in objects

- Type of error handling
- Security issues
  - Signature on messages versus callable objects
  - Encryption on messages versus restricted access to objects

”Information exchange” in a distributed object model, 1

Use reference to objects

- ⌘ Count on partner’s systems to be available at all time
- ⌘ Use version handling on objects

”Information exchange” in a distributed object model, 2

Copied or cloned

- What about the methods, they are not platform independent and can’t be copied?
- Could all involved actors use the same method libraries?
- What about deep versus shallow copying?
- Java: Coping objects by value, data, state and methods

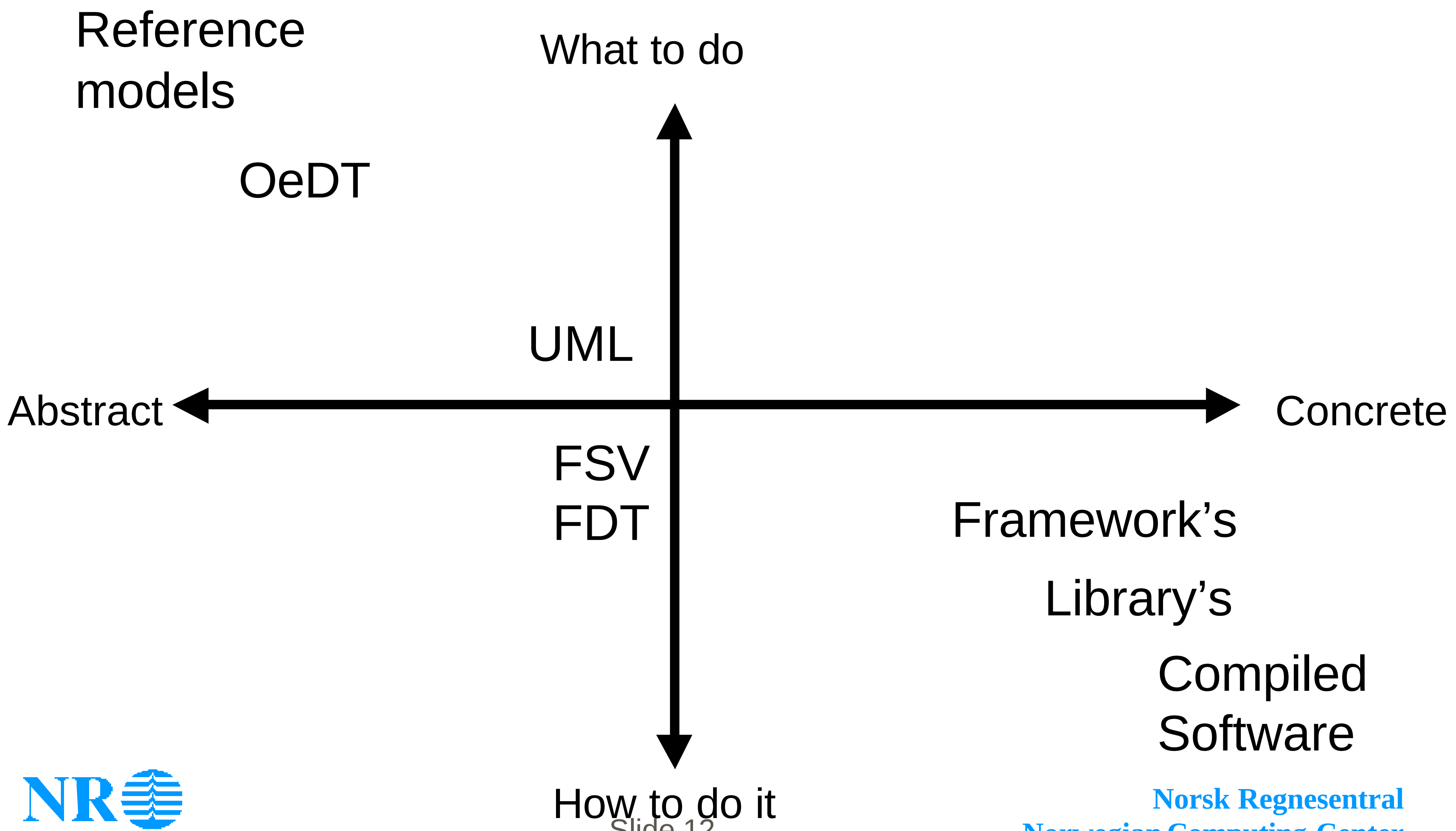
The idea of an object is that it should be viewed as a whole in the context it belongs. Copy by value is not in line with this idea.

Points of interest for evaluating FDT's and variations of implementations

In a ”live” business transaction:

- **Where/what is the scenario state?**
- **What/where are the scenario attributes?**
- **How is a scenario initiated?**
- **When does a role die?**
  - **End of scenario**
  - **End of business relation with partner**
- **Error handling? Where is it modeled?**
- **Security issues? Where is it modeled?**
  - **Digital signatures, encryption, access controll**

What to do versus how to do it



Comments

To have automatic code generation I believe that:

- A FDT’s expression power has to be equal to the expression power to the target abstration level
- Open-edi reference model serves as a good vocabulary for a very complex field, made up by simple parts
- UML descriptions is interchangable by using XMI, XML Metadata Interchange.