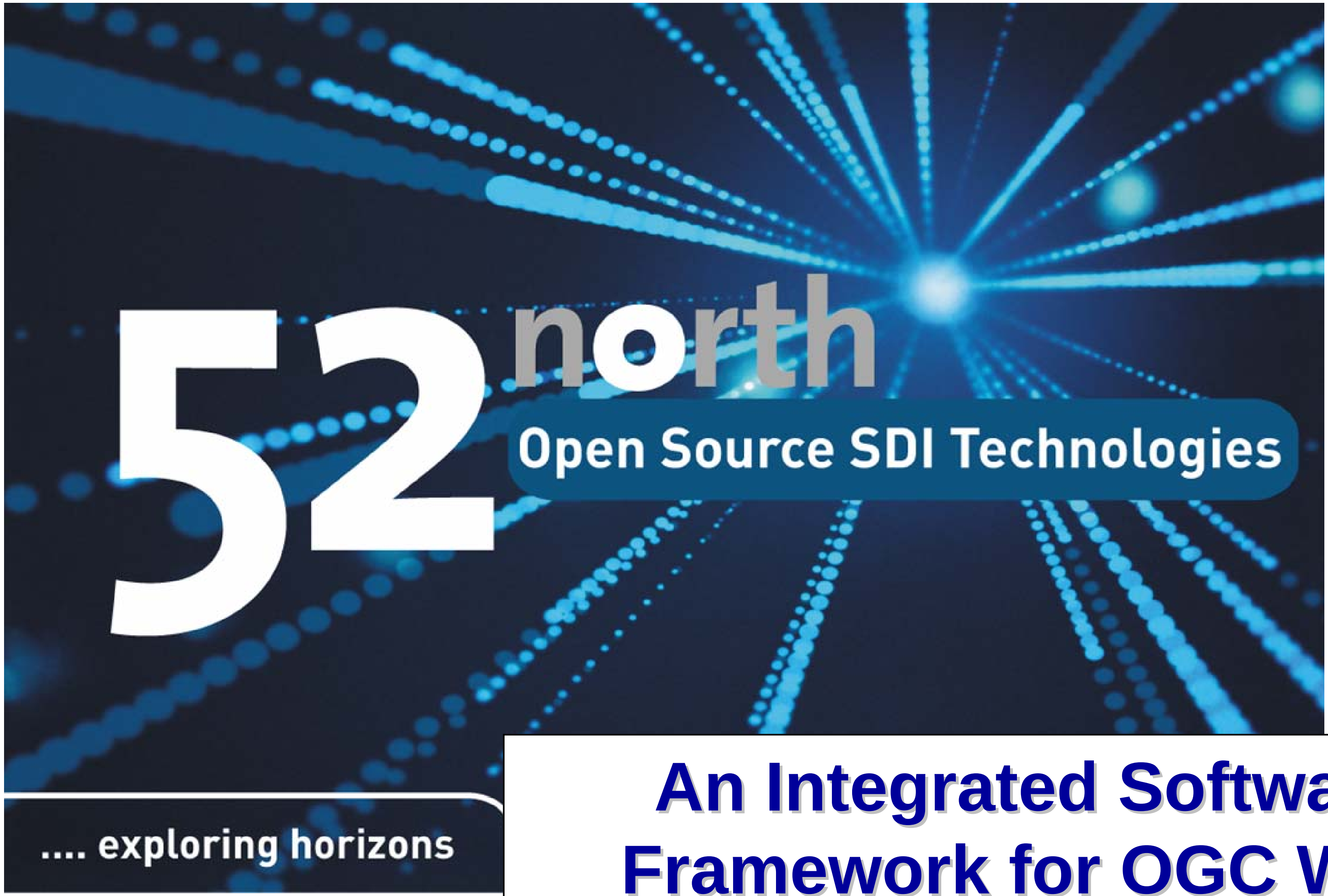


An Integrated Software Framework for OGC Web Services

Arne Bröring

FOSS4G 2006

- Various OGC Web Services providing different types of data:

WMS, WCS, WFS, SOS …

→ **aim:** integrate these data to carry out *reliable decisions*

→ develop a system enabling the *integration of arbitrary* OGC Web Service types

- Integrative …
  - …**Client** applications
    - Tillman & Garnett (2006): *OWS Integrated Client Architecture* (OGC Discussion Paper)
  - …**Service** applications
    - support sophisticated ***web processing*** and ***service chaining***
- *What is required?*: → an integrative approach for ***both*** environments

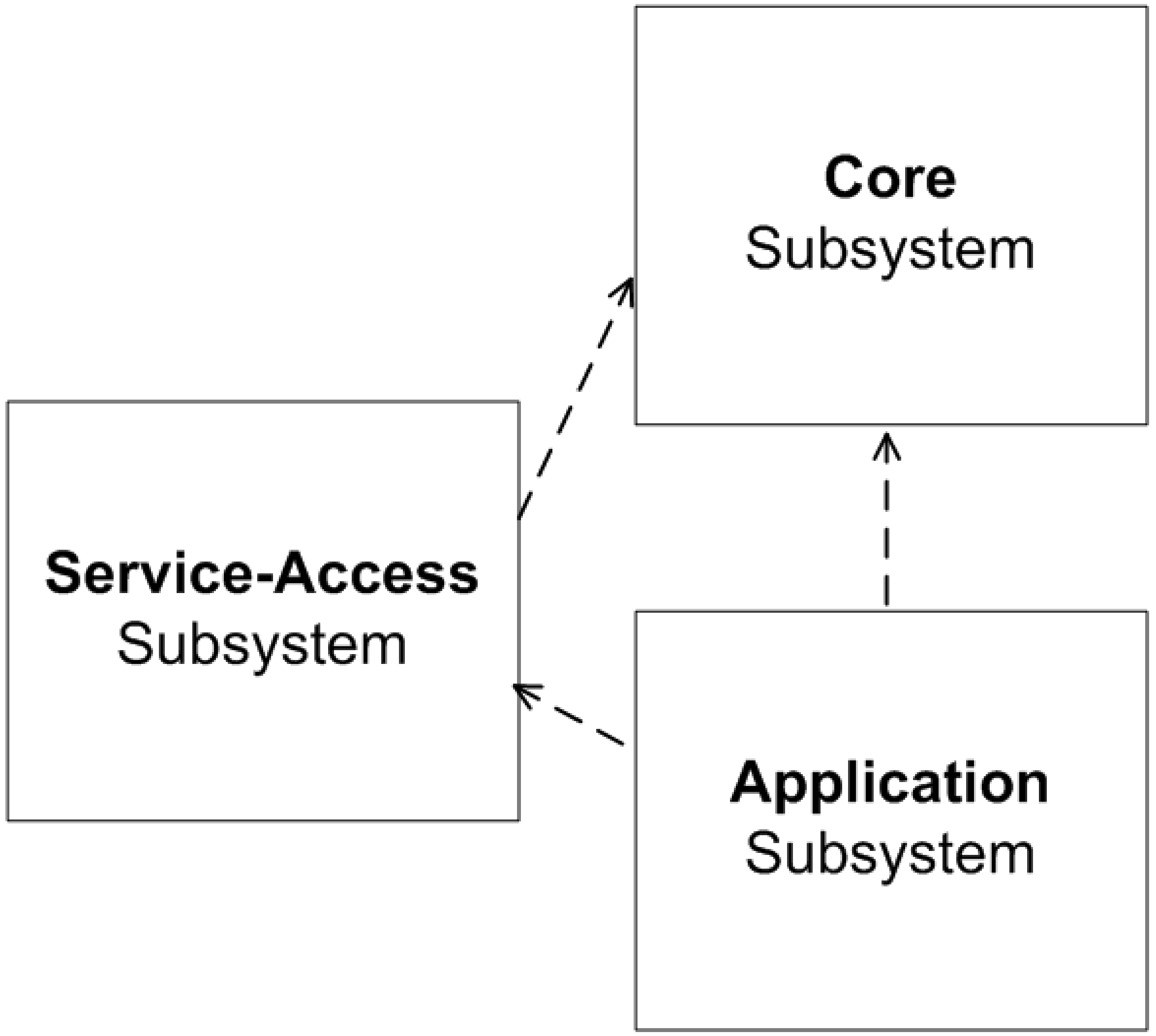
- **Generic solution**:

  OGC Web Service Access Framework (***OX-F****ramework*)

  - addresses **developers**
  - *customizable* and *extendable* system of **cooperating** classes
  - **reusable design**
  - applicable for *client* **and** *server* applications.

- Primary objective:
  - architecture has to be so much *flexible* and *extendable* that ***all*** kinds of OGC Web Services can be accessed
  - queried data can be *visualized* and *processed*

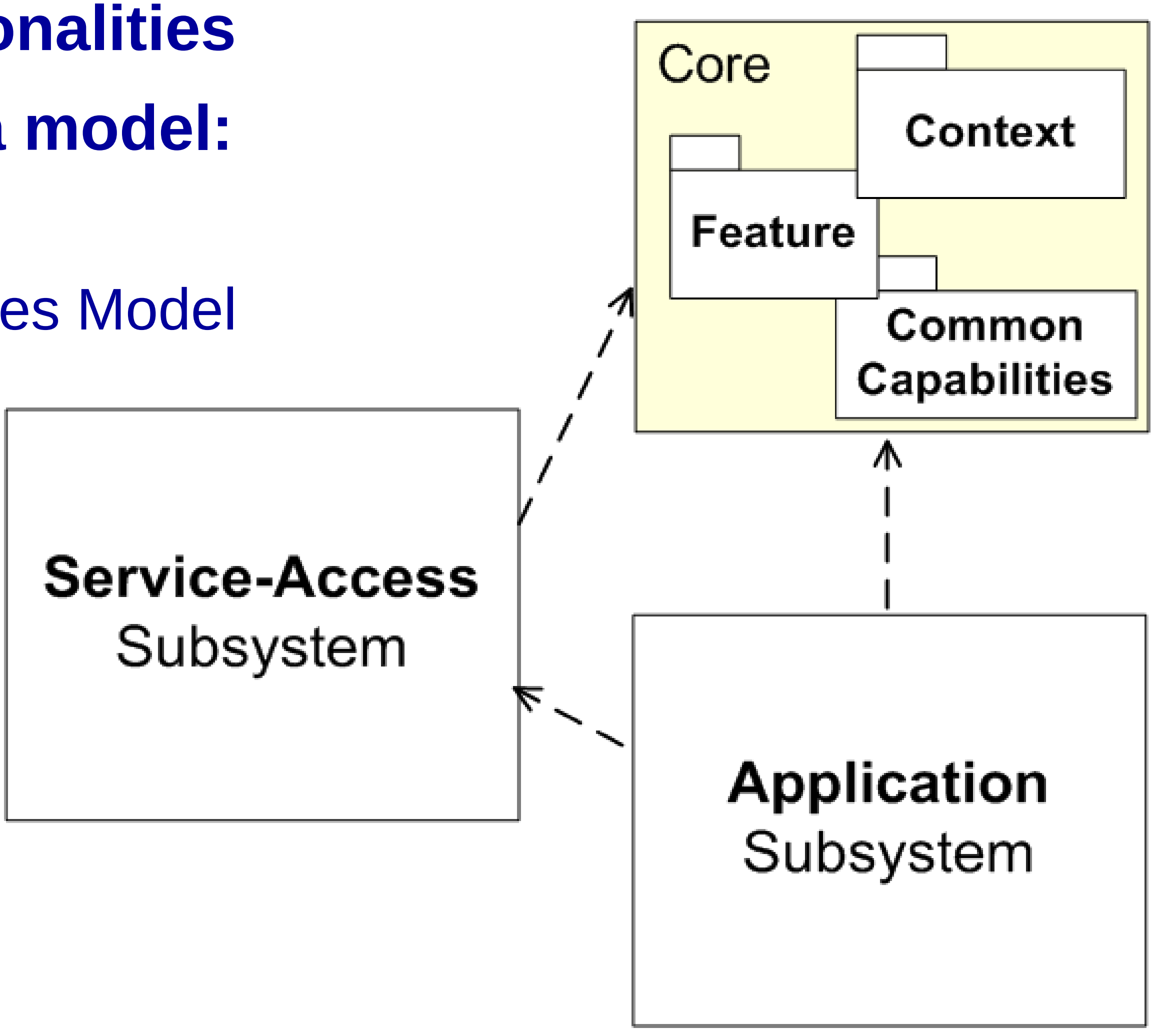
Architecture



Core Subsystem

- realizes **main functionalities**
- incorporates the **data model:**

1. Common Capabilities Model
2. Feature Model
3. Context Model



Core Subsystem

1. Common Capabilities Model

- uses the *OWS Common Specification*

  - „…specifies many of the aspects that are, or should be, common to ***all or multiple*** OWS interface Implementation Specifications”
  - already referenced by WMS, WCS, SOS …
  - defines main parts of *content* & *structure* of Capabilities document

➔ implemented here to marshal capabilities of various service types into java classes

Core Subsystem

2. Feature Model

- uses *OGC Abstract Topic 5* and *GML*

- basis for
  - ***Accessing***
  - ***Visualizing***
  - ***Processing***

  of feature geometry and attributes

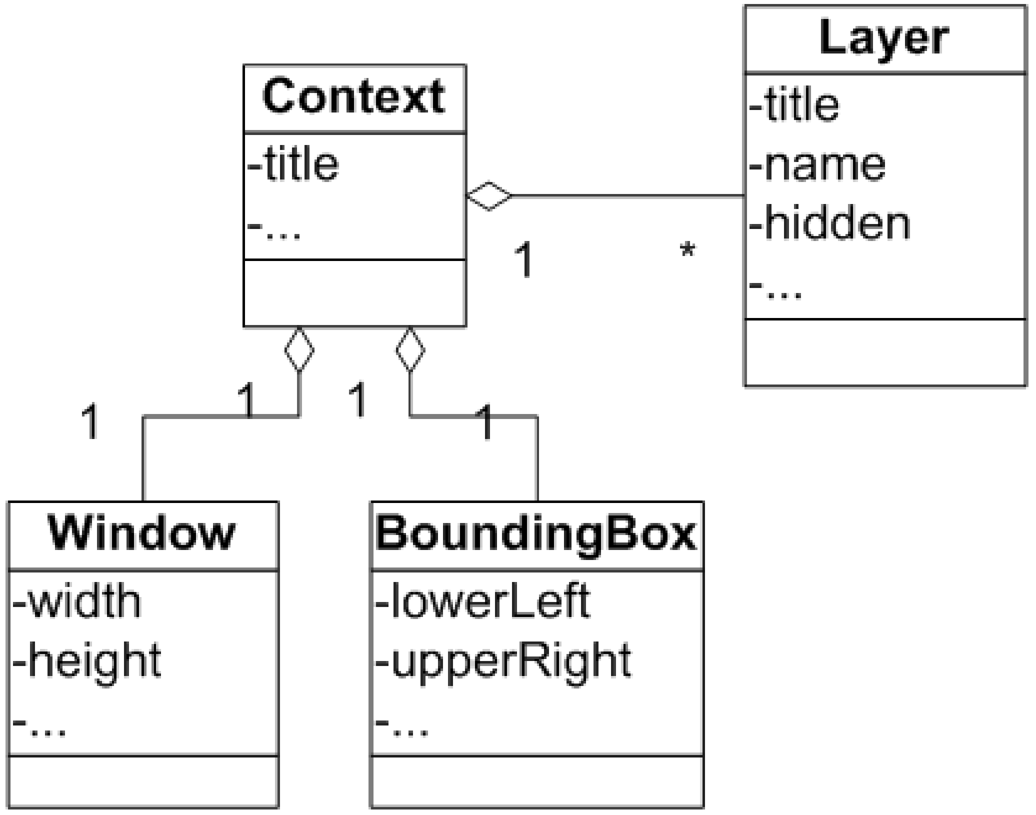
- allows feature schemas of arbitrary complexity

- received features can be marshalled into these classes

Core Subsystem

3. Context Model

- implements the *Web Map Context Specification* (WMC)
- manages the *current state* (→ “Context”) of an application



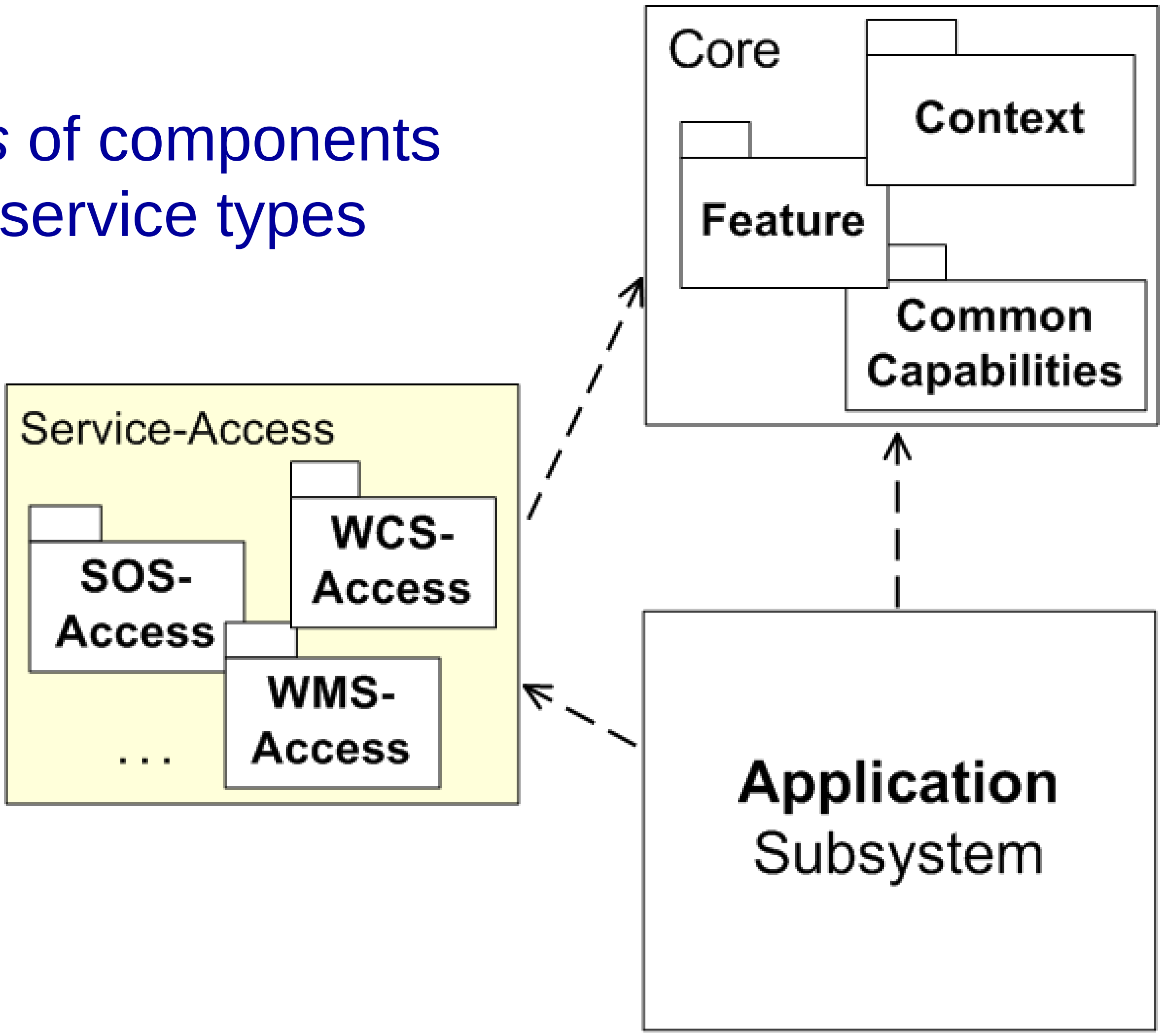
Core Subsystem

3. Context Model

- enables *persistence*- and *exchange*-functionality
- maintains *flat file representation of the state*
- encoding of serialization is also defined by the WMC Specification

Service-Access Subsystem

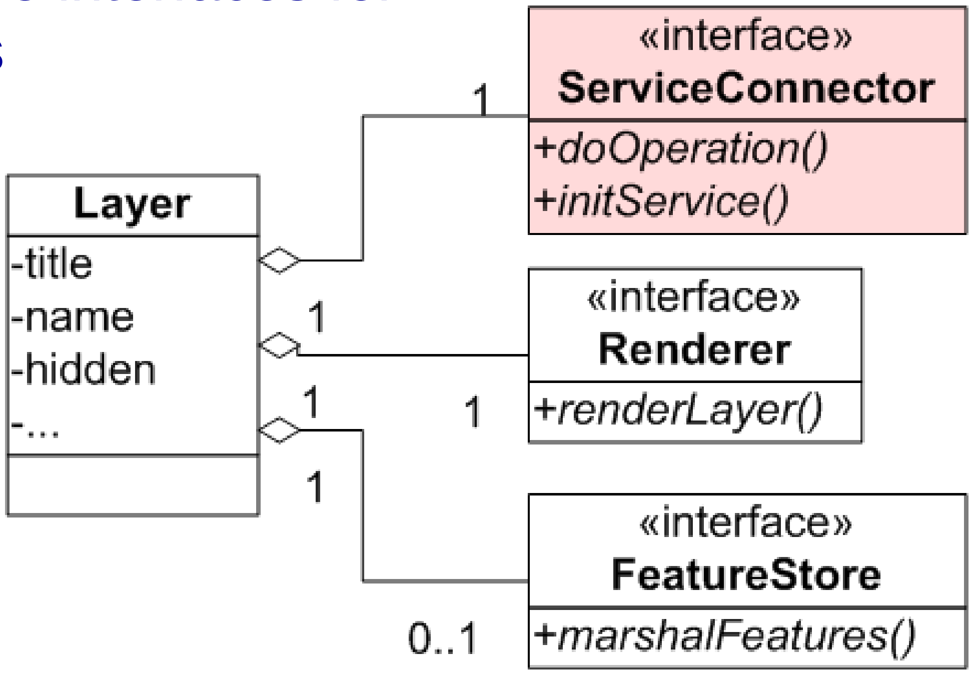
- contains *realizations* of components to access particular service types



Service-Access Subsystem

- contains implementations of these interfaces for specific OGC Web Service types

1. **Service-Connector**
   - initializes *Common Capabilities Model*
   - executes service operations

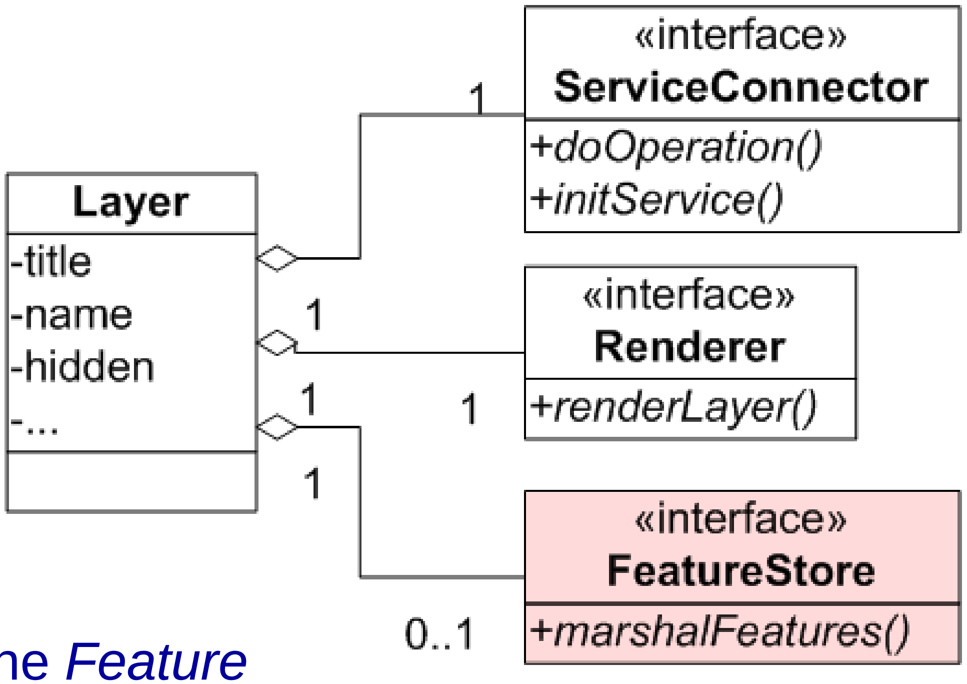


Service-Access Subsystem

- contains implementations of these interfaces for specific OGC Web Service types

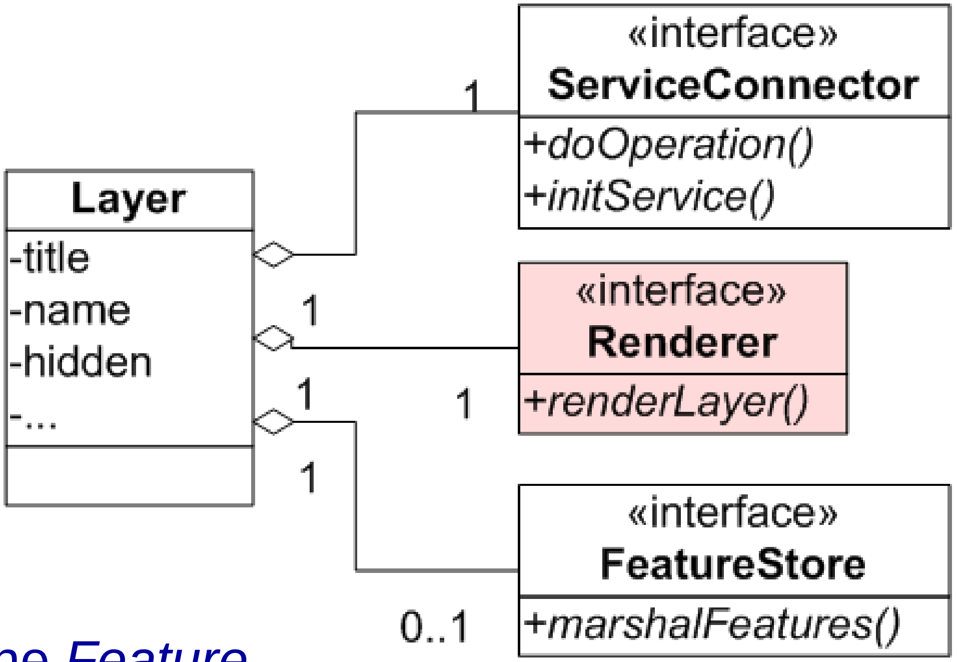
1. **Service-Connector**
   - initializes *Common Capabilities Model*
   - executes service operations

2. **Feature-Store**
   - marshals received features into the *Feature Model*



Service-Access Subsystem

- contains implementations of these interfaces for specific OGC Web Service types



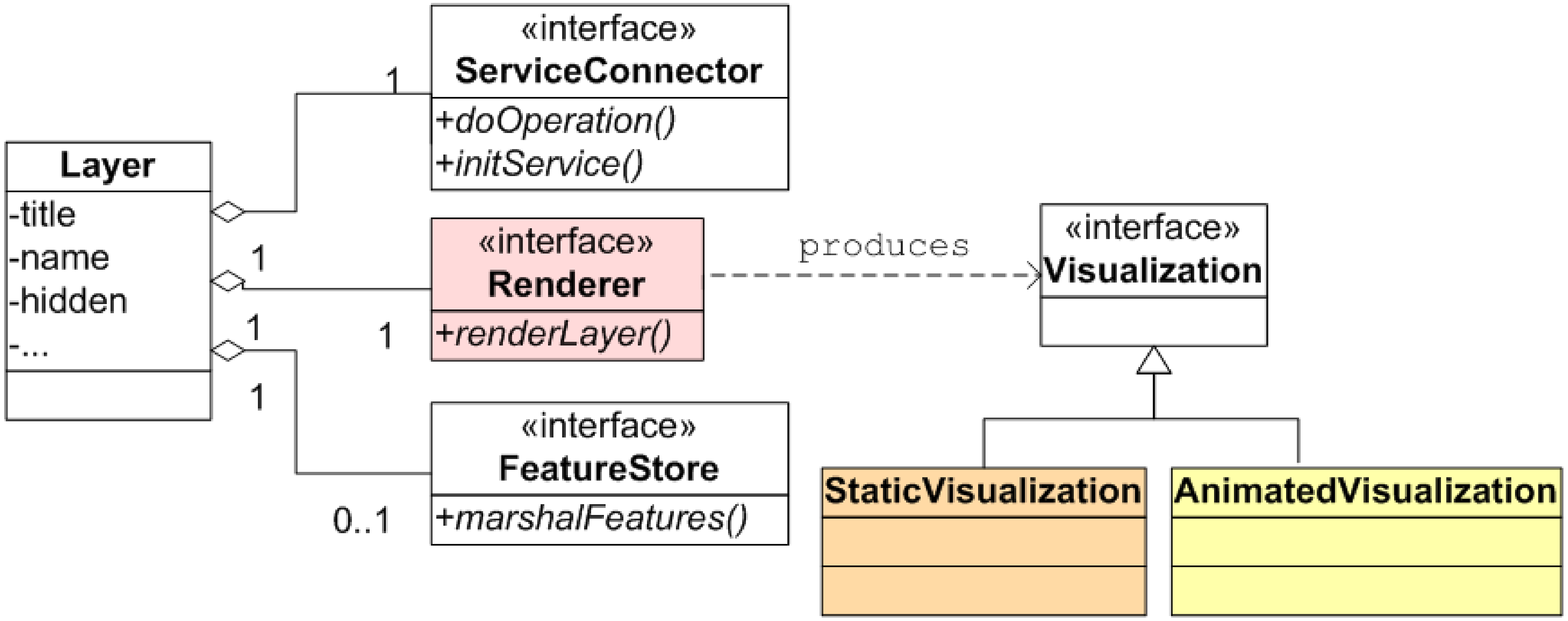
1. **Service-Connector**
   - initializes *Common Capabilities Model*
   - executes service operations

2. **Feature-Store**
   - marshals received features into the *Feature Model*

3. **Renderer**
   - converts data to graphical representation

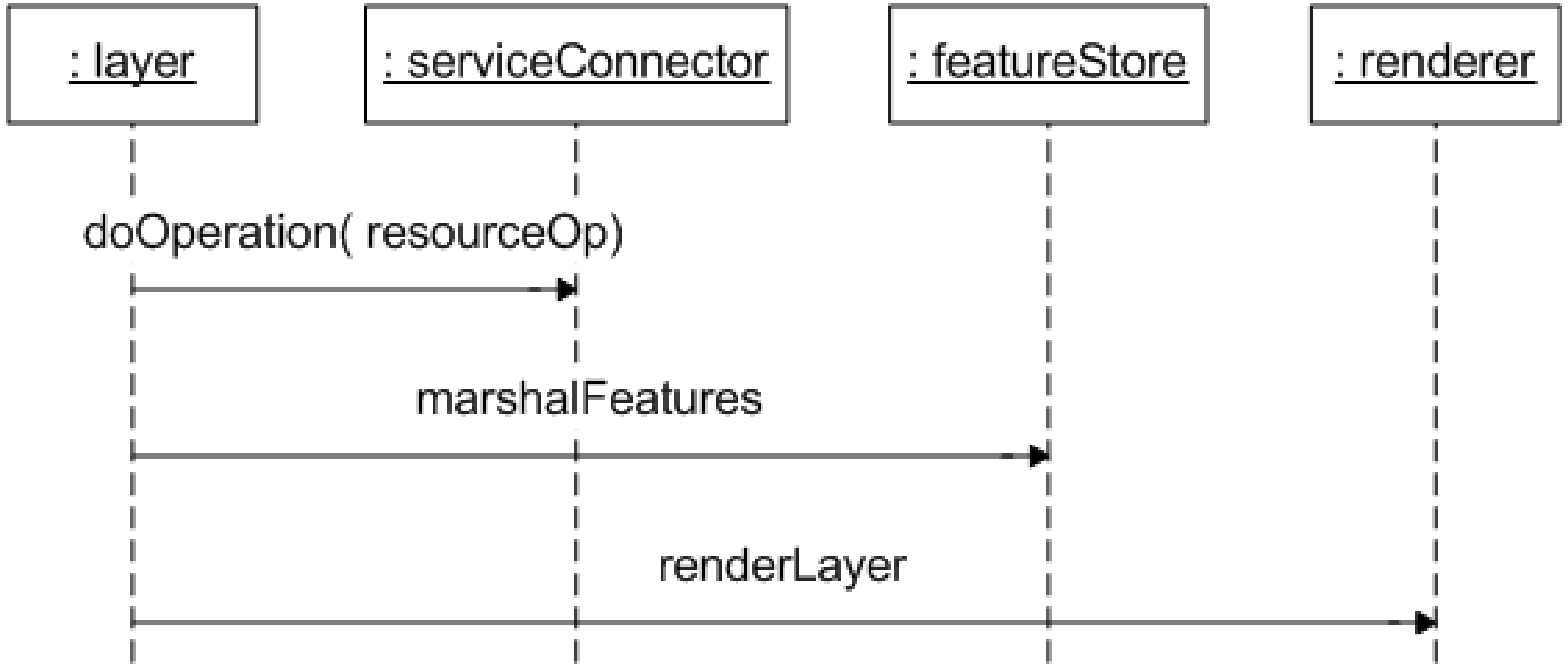
Service-Access Subsystem

- Renderer ➔ data visualization engine



Service-Access Subsystem

- Workflow:



Service-Access Subsystem

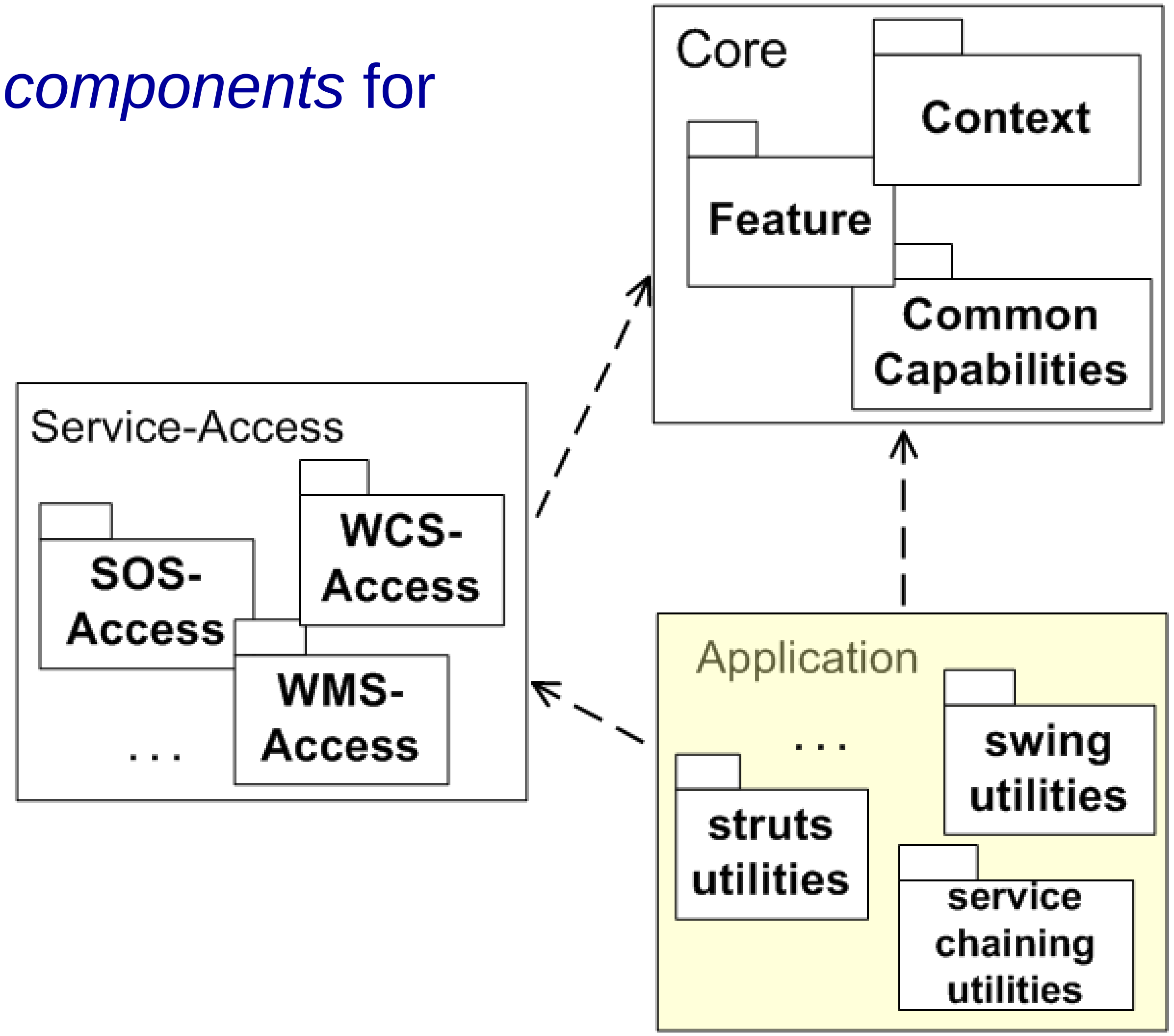
- …already implemented:

Service-Connectors	Feature-Stores	Renderers
• WMS • WCS • SOS	• SOS	• WMS • WCS • SOS - Charts - Charts in a Map - Interpolation

- **Plugin-Mechanism** → include and replace components at runtime

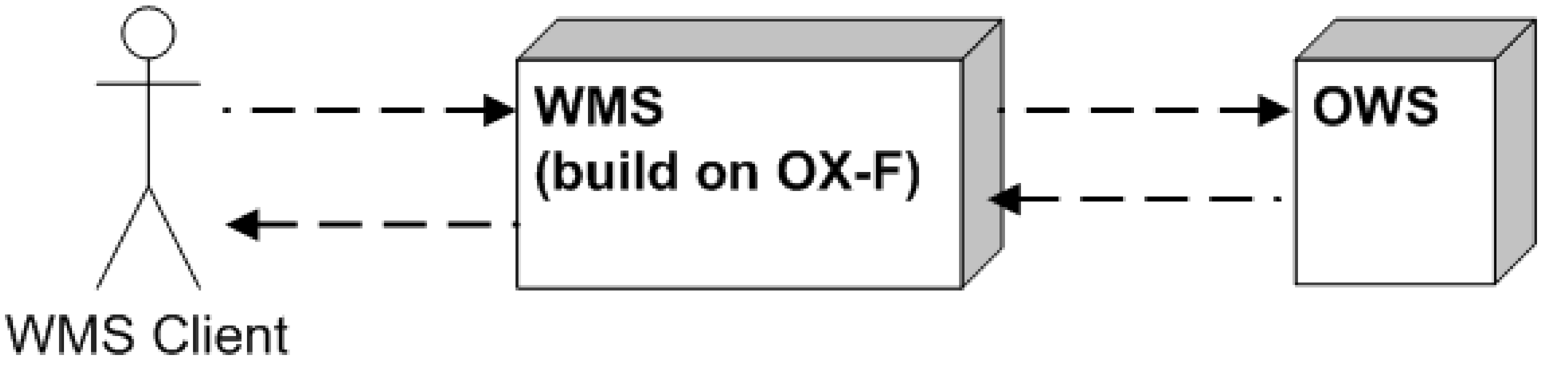
Application Subsystem

- contains *classes & components* for *specific frontends*



Application Subsystem

- already implemented:
  - → *Swing-Frontend*
    - ➢*Focus: sensor data visualization*
  - → *WMS-Frontend*



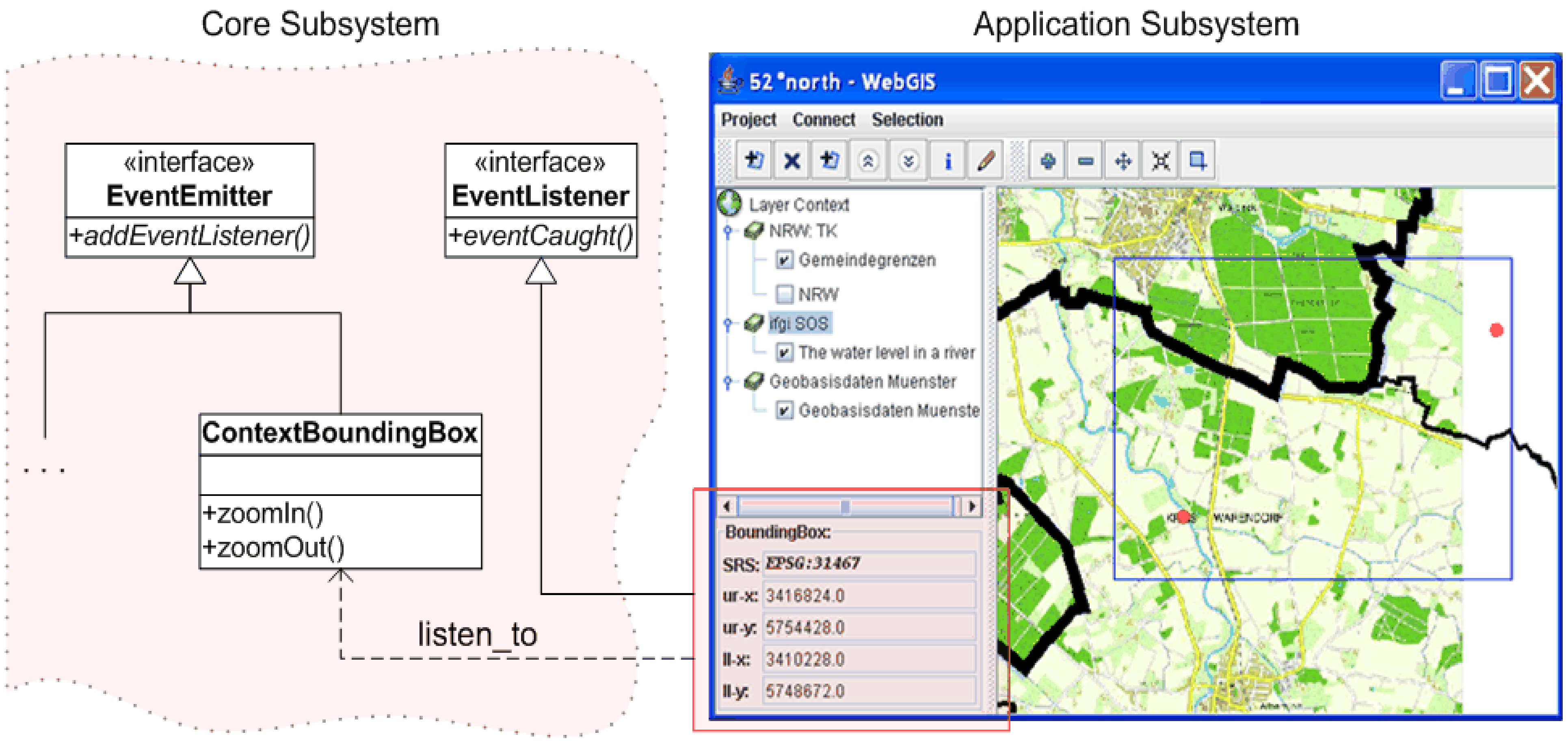
➔ emerging benefit:

- ***reuse*** of already implemented *Renderers*, *FeatureStores* and *ServiceConnectors*

Listener-Concept

- aka “Observer”- or “Pub/Sub”-pattern
- affords *extensibility* and *transparency*
- endows developer with *absolute control* over the framework

Listener-Concept



BoundingBox:

SRS:	EPSG:31467
ur-x:	3416824.0
ur-y:	5754428.0
ll-x:	3410228.0
ll-y:	5748672.0

→ Client-Video

→ WMS-Video

ifgi
Institute for Geoinformatics
University of Münster

Thank you
for your attention!

Arne Bröring

Robert-Koch-Str. 26-28
D-48149 Münster
Tel: (49)-251-83-31965
Fax: (49)-251-83-39763
arneb@uni-muenster.de
http://ifgi.uni-muenster.de

http://www.52north.org