

# FOSS4G 2010, Barcelona

## COMPARISON OF SOS-SERVER: 52°North, UMN and deegree

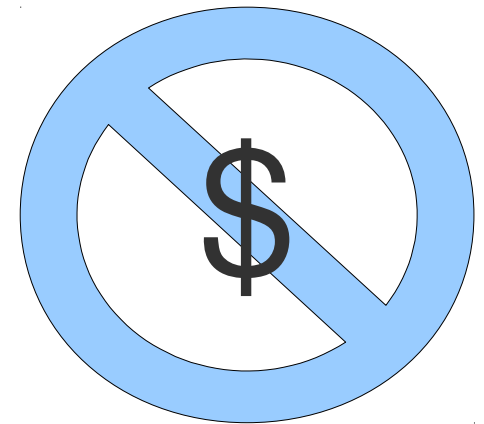
Christian Mayer, terrestris GmbH & Co. KG



# Motivation



WEB



# FOSS4G

# Sensor Observation Service

- Standard of the Open Geospatial Consortium (OGC)
- Providing an interface for
  - managing deployed sensors
  - retrieving sensor data, specifically „observation“ data
- Implementation Standard Version 1.0.0 [06-009r6]
- Released in 2007

# Sensor Observation Service

- Operations

- **GetCapabilities** - requesting a self-description of the service
- **GetObservation** - requesting the pure sensor data encoded in Observations & Measurements (O&M)
- **DescribeSensor** - requesting information about the sensor itself, encoded in Sensor Model Language (SensorML) instance document
- RegisterSensor - signing up new sensors
- InsertObservation - inserting new observations for registered sensors
- GetFeatureOfInterest - requesting the GML encoded representation of the feature that is the target of the observation
- GetResult - periodically polling of sensor data
- ...

# Sensor Observation Service

- Terms
  - Observation
    - An act of observing a property or phenomenon
  - Procedure
    - Method, algorithm or instrument observing
  - Offering
    - An offering is a logical grouping of observations offered by a service that are related in some way.
  - Feature Of Interest
    - Feature which is observed

# 52°North SOS

- Version 3.1.1
- Implements the latest SOS schema version 1.0.0
- Supported Operations
  - GetCapabilities
  - GetObservation
  - DescribeSensor
  - RegisterSensor
  - InsertObservation
  - GetFeatureOfInterest
  - GetResult
- Reference Implementation of the OGC



# UMN MapServer SOS

- Version 5.6.5
- Implements the latest schema version 1.0.0
- Supported Operations
  - GetCapabilities
  - GetObservation
  - DescribeSensor

# deegree3 SOS

- Version deegree3 [trunk]
- Implements the latest schema version 1.0.0
- Supported Operations
  - GetCapabilities
  - GetObservation
  - DescribeSensor
  - GetFeatureOfInterest



# Initial Situation

- Stations in Germany for weather observations ( $\sim 8000$ )
- Observation of temperature once a day for one week
- PostgreSQL/PostGIS Database
  - Table sensors incl. point geometry of location
  - Table measurements referring to sensor
- Random values ( $\sim 60.000$ )

# Initial Situation

id	the_geom
1	POINT(50 8)
2	POINT(51 9)
3	POINT(52 7)
...	...

- Num of Obs: 57890
- Max value: 39.99°
- Min value: 10.00°
- Average value: 25.03°

id	fk_sensor	time	value	phen
1	2	2010-08-1	23.8	temp
2	3	2010-08-1	20.9	temp
3	1	2010-08-1	17.3	temp



# Test Criteria

- Installation and configuration
- Performance
- Clients
  - Own clients
  - Integration in OpenLayers

- Installation and configuration
  - Deploy war-file in servlet container
  - Create PostgreSQL database with fixed schema
  - Configure database connection in a text file
  - Fill database
    - Direct SQL access
    - Feeder-Framework
    - Transactional operations of SOS

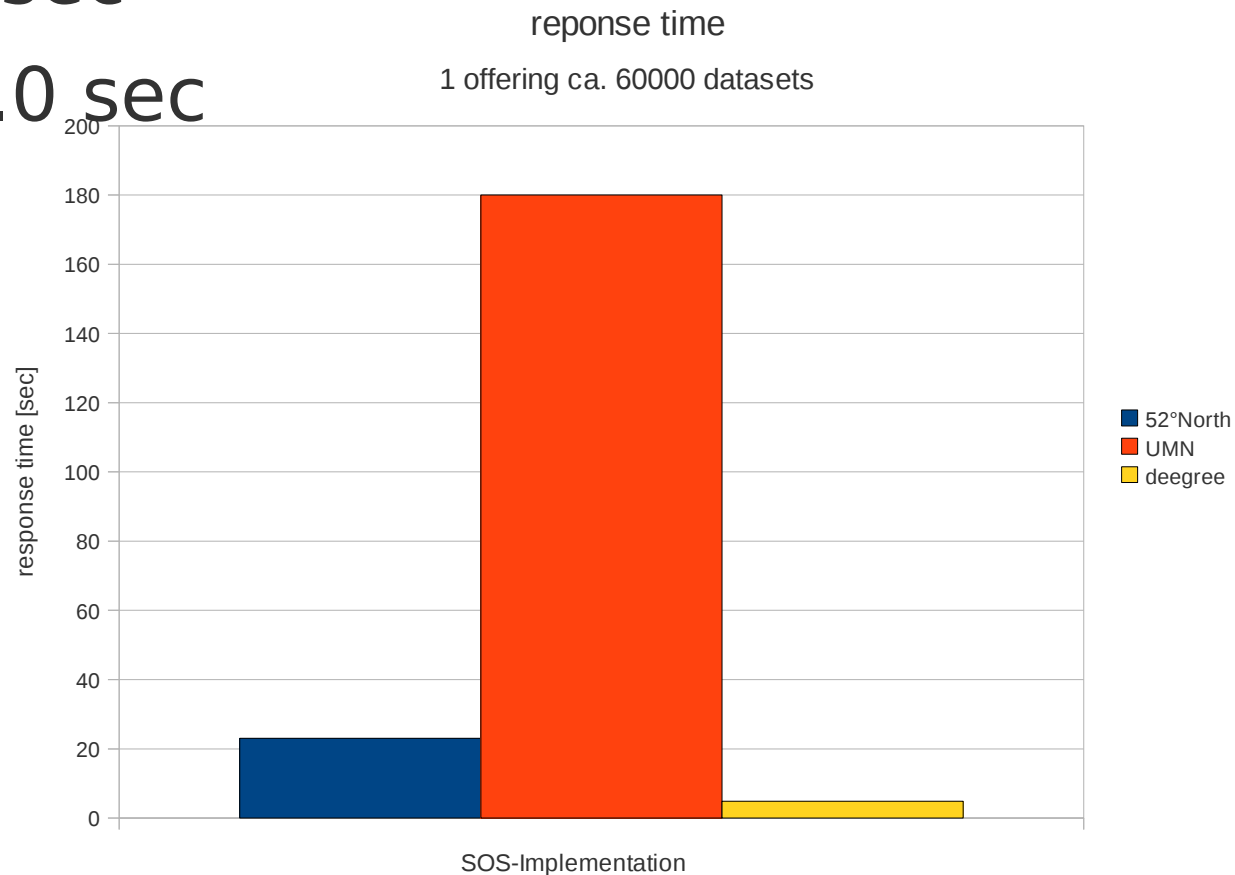
# UMN MapServer

- Installing and configuration
  - Install MapServer with 3 additional parameters
    - flag -DUSE\_SOS\_SVR is required
    - requires either -DUSE\_WMS\_SVR or -DUSE\_WFS\_SVR flags to be enabled
    - requires libxml2 and proj libraries
    - requires ICONV support (-DUSE\_ICONV) on Windows
  - Write a Mapfile for SOS
    - Define general SOS settings
    - Define offerings as layer
    - Configure DB-access and create SQL to query your observation data

- Installation and configuration
  - Deploy servlet with a maven script
    - Direct deployment to servlet container
    - Create war-file out of sources and put into servlet container
  - Configure SOS with several XML-files
    - „ServiceConfiguration“ defines offerings incl. sensors
    - „ObservationStore“ defines responsible DB-tables and columns
    - „PooledConnection“ defines database and access parameter (user / password)

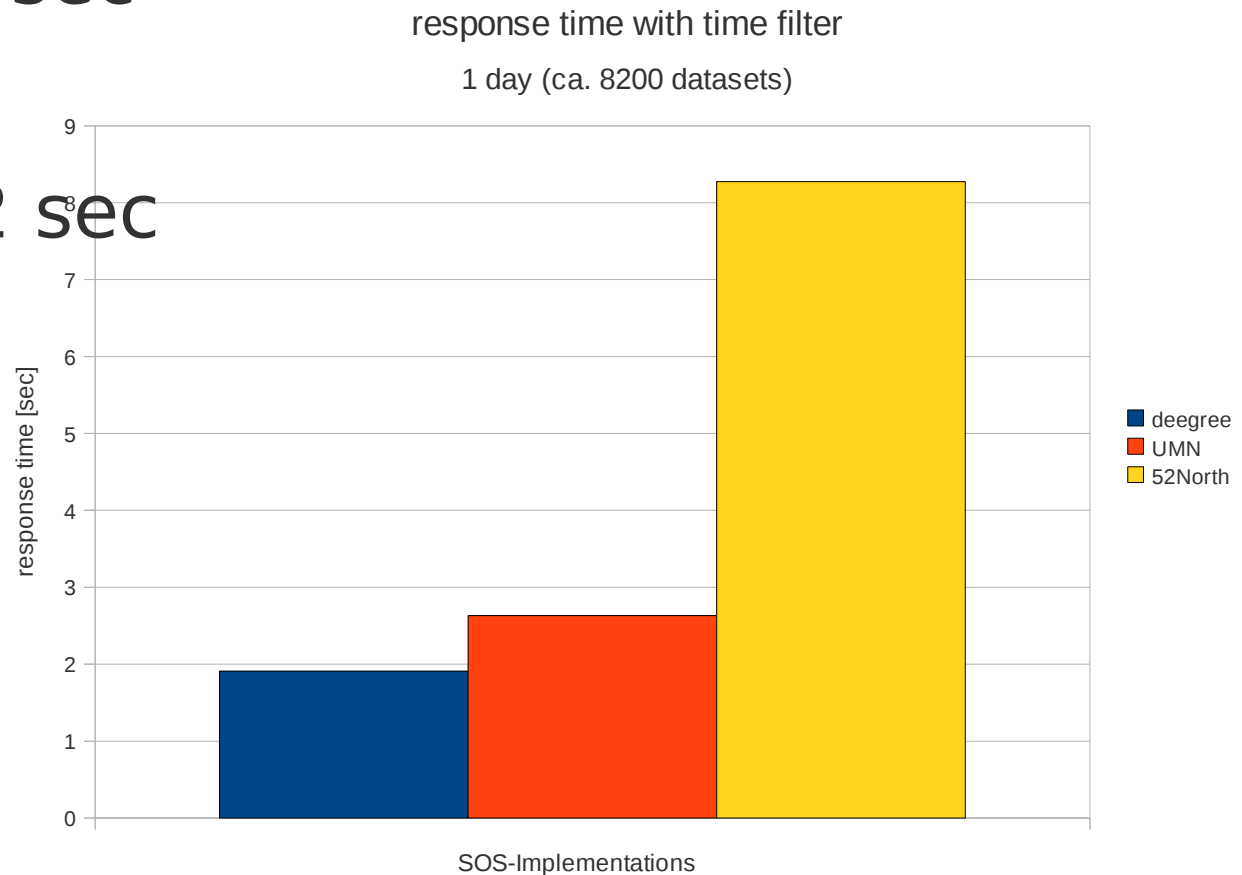
# Performance

- Requesting one offering with 60000 observations
  - deegree: ~4.8 sec
  - 52°North: ~23.0 sec
  - UMN: timeout



# Performance

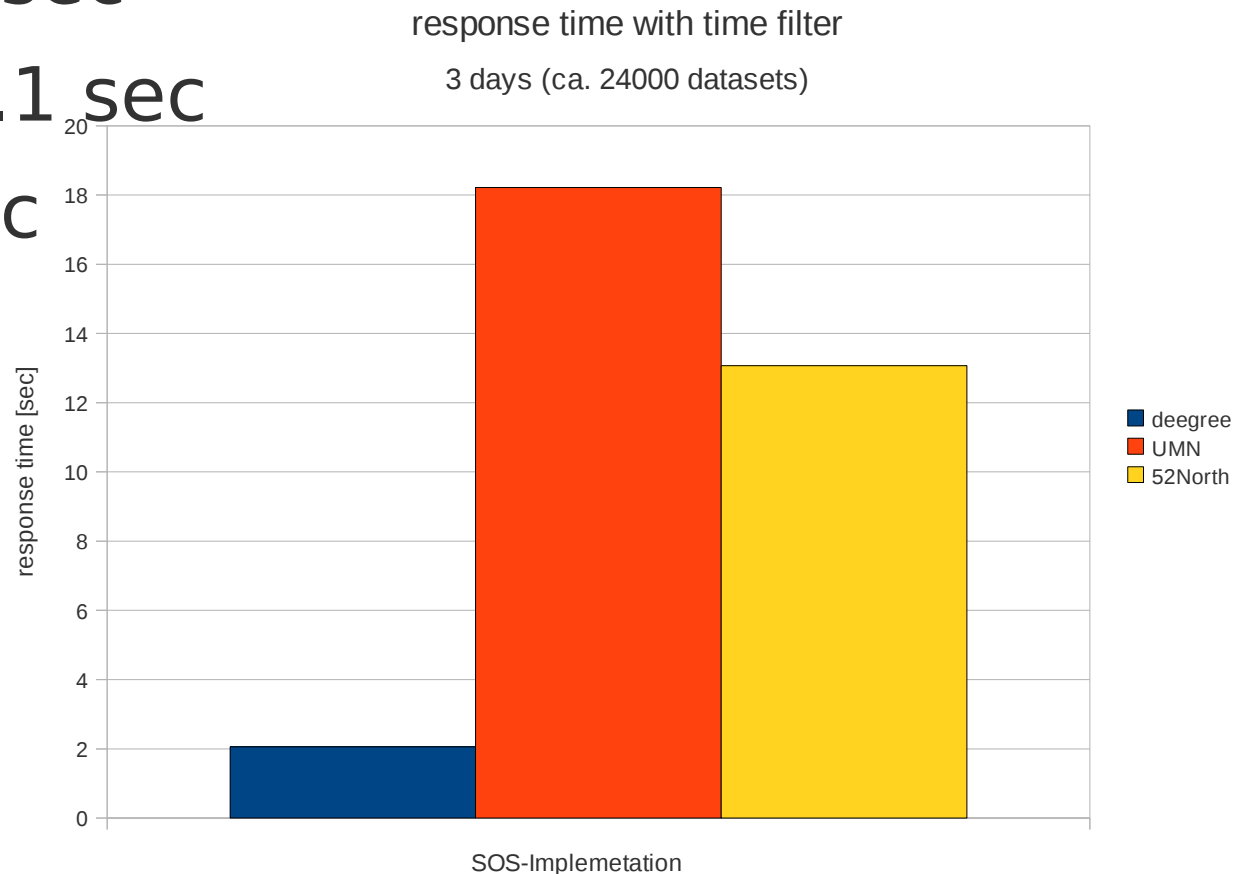
- Requesting data for one day ( $\sim 8200$  observations)
  - Deegree:  $\sim 1.9$  sec
  - UMN:  $\sim 2.6$  sec
  - 52°North:  $\sim 8.2$  sec





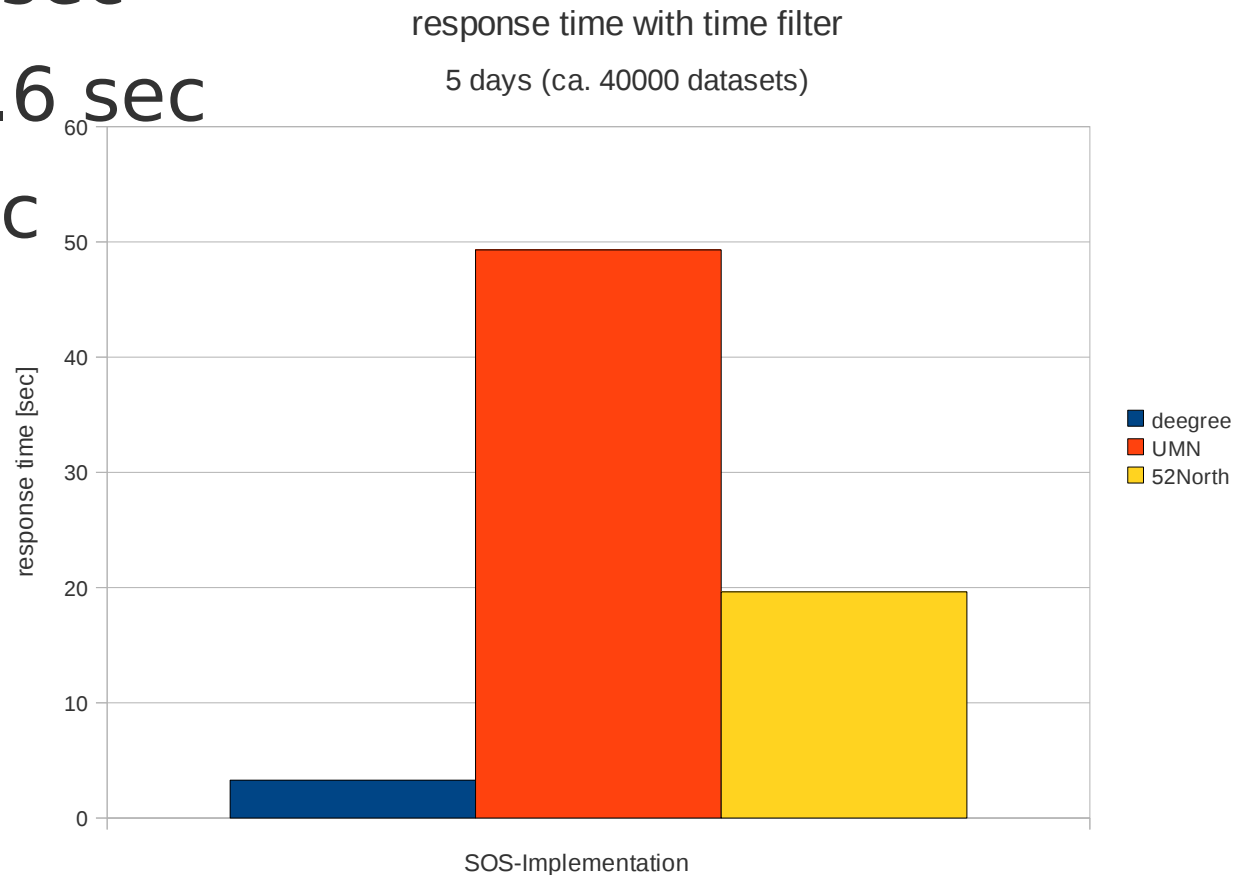
# Performance

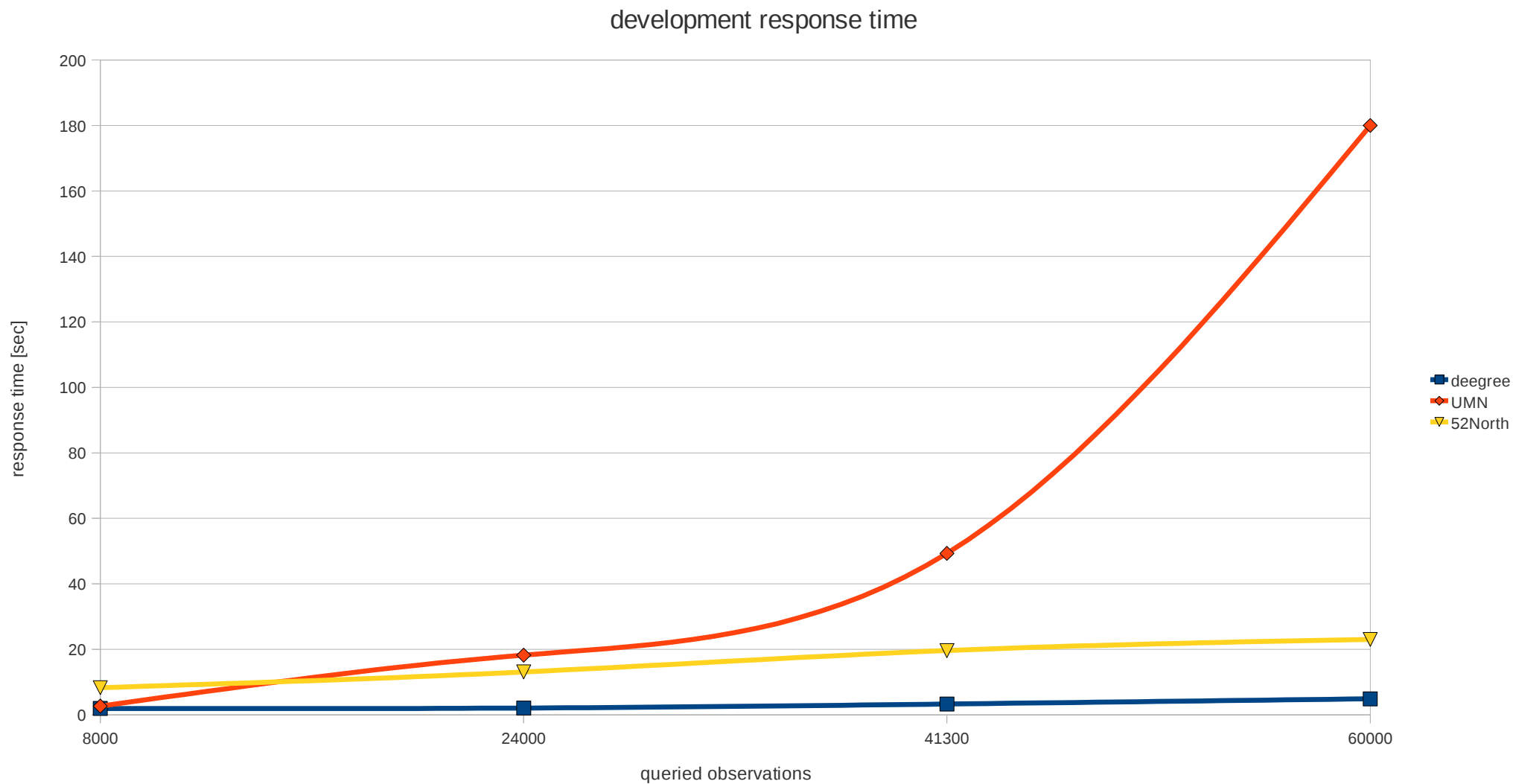
- Requesting data for three days (~24000 observations)
  - deegree: ~2.1 sec
  - 52°North: ~13.1 sec
  - UMN: ~18.2 sec



# Performance

- Requesting data for five days (~40000 observations)
  - deegree: ~3.2 sec
  - 52°North: ~19.6 sec
  - UMN: ~49.2 sec





# Clients

- 52°North
  - Test Client
  - Rich OX Client
  - [Thin SWE Client]
  - [ArcGIS SOS Extension]
- deegree
  - No client available

- UMN
  - No client available
- General
  - OpenLayers
  - uDig plugin
  - [gvSIG SOS extension]

	52°North	UMN	deegree
Browser HTTP GET			
52°North Testclient			
uDig			
OpenLayers			
Rich OX Client			

## SOS client example

Shows how to connect OpenLayers to a Sensor Observation Service (SOS)



This example uses a vector layer with a Protocol.SOS and a fixed Strategy.

When clicking on a point feature (the weather stations offered by the SOS), the latest values for all offerings are displayed in a popup.



This example uses a vector layer with a Protocol.SOS and a fixed Strategy.

When clicking on a point feature (the weather stations offered by the SOS), the latest values for all offerings are displayed in a popup.



# Summary 52°North

- Pro
  - Easy installation by deploying a war-file
  - Easy configuration by text file
  - Easy-to-use Testclient
  - FOIs in getObservation response → mapping easy
  - Supports most operations
- Contra
  - Only PostgreSQL is supported
  - Fixed database schema → fill database with SQL and transfer all observations into necessary tables
  - Performance is ok, but huge file size if many FOIs in DB

# Summary UMN

- Pro
  - Easy installation
  - Easy configuration, if user is used to handle Mapfiles
  - Extension for existing software
  - Free DB schema
  - Capabilities document is based upon Mapfile entries
- Contra
  - Handling of Mapfiles, if you are not used to
  - FOI as BBOX must be included in GetObservation request
  - Performance decreases with increasing observations (scaling problems)

# Summary deegree

- Pro

- Very easy installation with maven script
- Support of many databases (JDBC)
- Very fast
- Supports WFS as FOI store

- Contra

- Configuration with XML files
- FOIs have to be defined in XML, not in a DB
  - Script ?

# QUESTIONS ???

**mayer@terrestris.de**