

Contract-first Service Development Within the Venice Service Grid

Markus Hillenbrand, Joachim Götze, and Paul Müller

Dr. Markus Hillenbrand

University of Kaiserslautern, Germany
Integrated Communication Systems Lab, ICSY
e-mail: hillenbr@informatik.uni-kl.de

iiWAS 2008
24-26 November 2008 Linz, Austria



Outline

- Contract-first and code-first service development
- What is the Venice Service Grid?
 - Overview
 - Top Level View
 - Architecture
- What does the Venice Service Compiler?
 - Service life-cycle
 - Compiler Language EBNF
 - Example
 - Compiler Output
 - Automatically generated high-level functionality
- Conclusion and Outlook



Code-first vs. Contract-first

Code-first

1. Implement service
2. Generate data types and interface from code

Advantage:

- Easy to make (tool support)
- Fast service development

Disadvantage:

- Less interoperability

Contract-first

1. Define data types
2. Define interface
3. Generate code fragments
4. Implement service

Advantage:

- Interoperability
- Design independent of coding

Disadvantage:

- Much to do before development



Outline

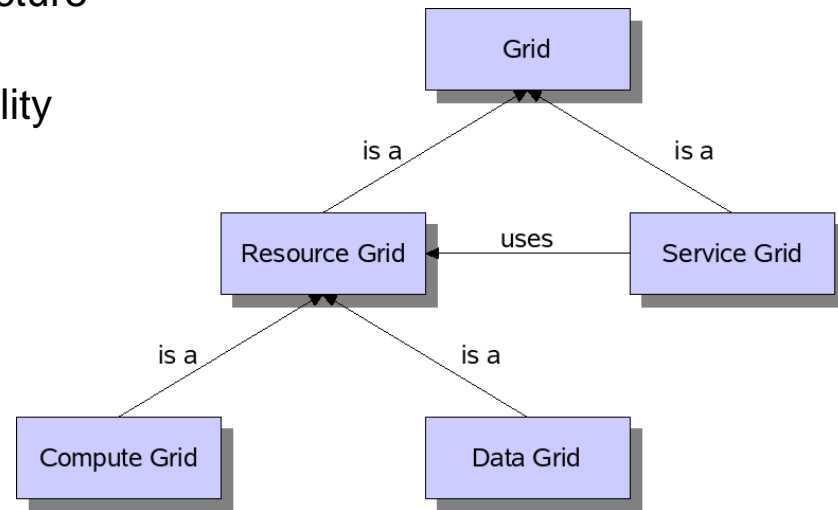
- Contract-first and code-first service development
- What is the Venice Service Grid?
 - Overview
 - Top Level View
 - Architecture
- What does the Venice Service Compiler?
 - Service life-cycle
 - Compiler Language EBNF
 - Example
 - Compiler Output
 - Automatically generated high-level functionality
- Conclusion and Outlook



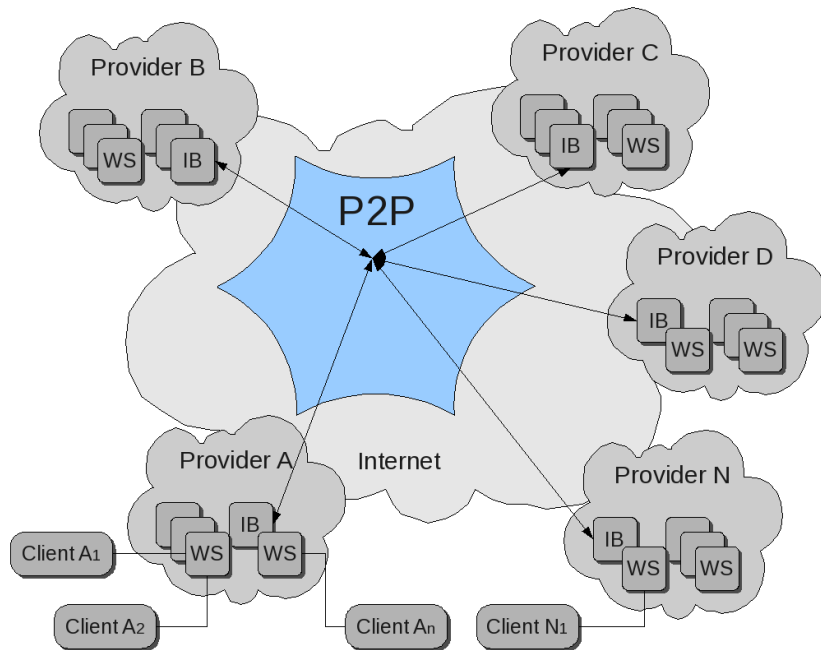
What is the Venice Service Grid?

<http://www.icsy.de>

- A software infrastructure:
 - Based on a service-oriented architecture (SOA)
 - With focus on openness, dependability and security
 - Service deployment on the Internet
- A set of services:
 - Service management at runtime
 - Service information and access
 - Service collaboration and communication
 - Services for building applications upon
- A runtime environment:
 - For service development and deployment
 - For client development and service access



Top Level View of Venice



- Service providers are
 - Independent
 - Autonomous
- Service requestors
 - Have a home domain
 - Can use services of other domains
- Venice is responsible for
 - Mutual authentication
 - Distributed authorization
 - Service brokering
 - Service access



Architectural Overview

Services of the Application Domains

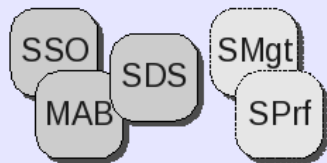
Application Domain X

Application Domain Y

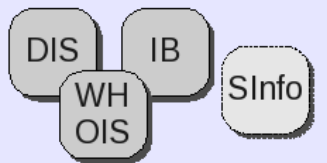
Application Domain Z

Service infrastructure

Management Services:



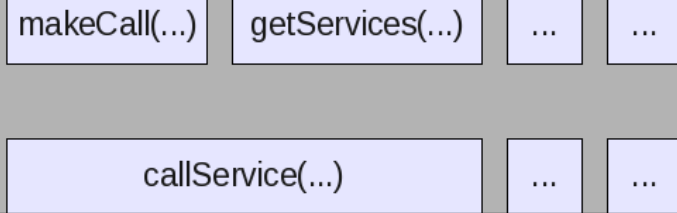
Information Services:



Common Application Services:



Abstraction Layer



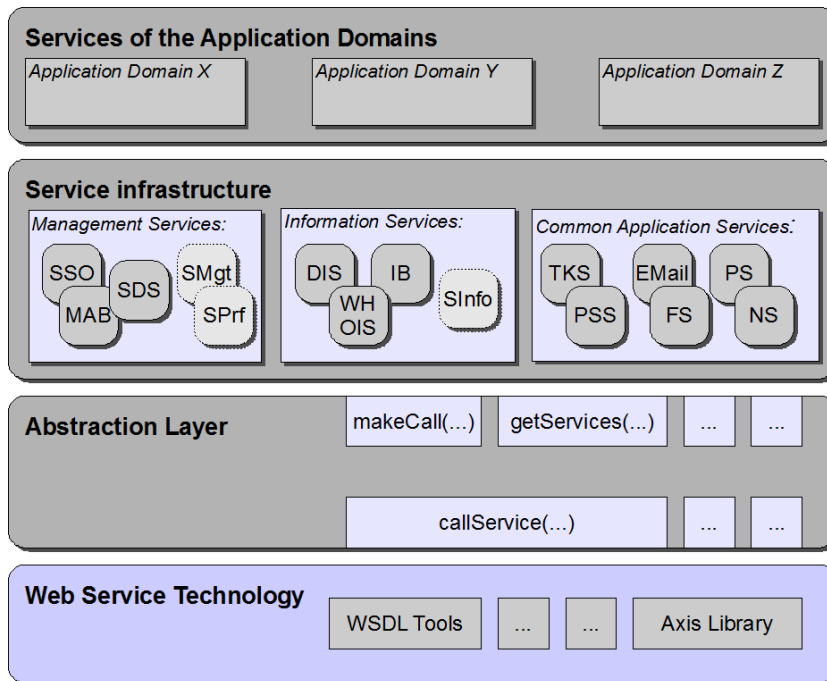
Web Service Technology



<http://www.icsy.de>



Web Service Technology



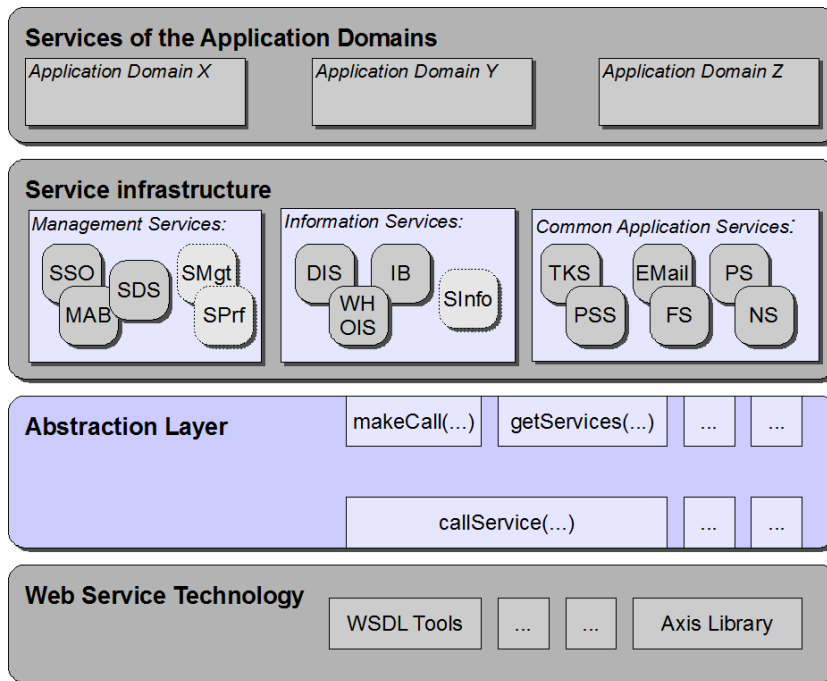
- Off-the-shelf products
 - Tomcat 5 and 6 (Service container)
 - Axis 1 (SOAP engine)
 - WSDL4Java
 - Additional tools as needed

- Tasks
 - WSDL handling
 - Data type conversion
 - HTTPS / certificates
 - Service deployment

http://www.icsy.de



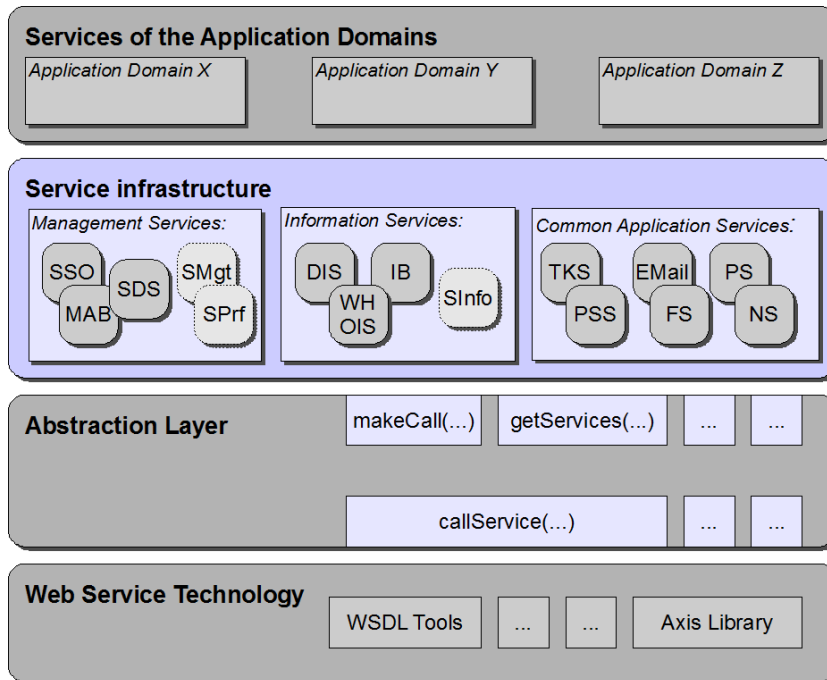
Abstraction Layer



- Hides Web service technology
- Solely uses URIs for
 - Identifying and
 - Accessing
 Web services
- Dynamic invocation (no stubs)
- Provides the *Venice Service Compiler*
- Functionality addressed
 - Security
 - URI to WSDL handling
 - Data collection
 - Service access



Service Infrastructure

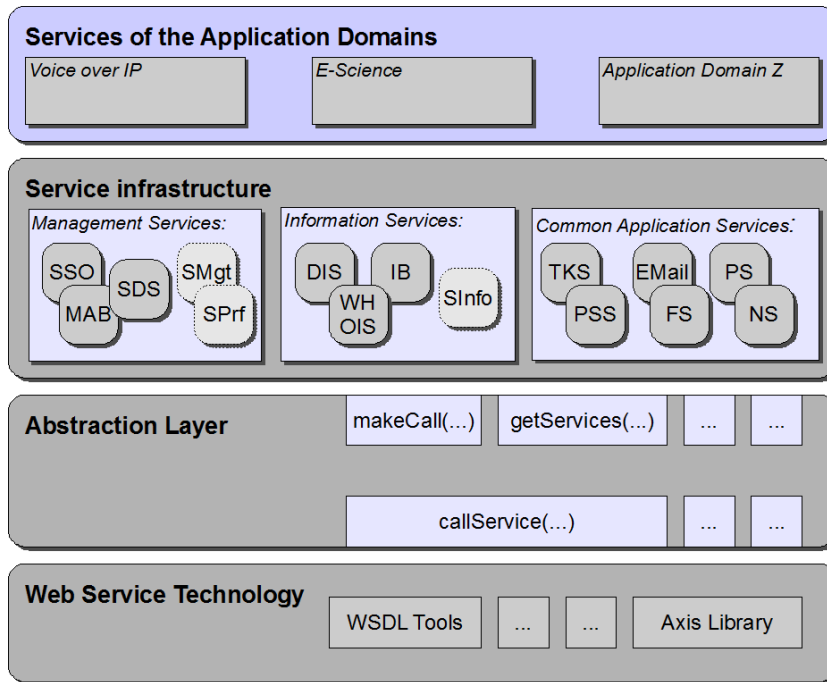


- Three service categories
 - Management services
 - Information services
 - Common application services

- Most important services:
 - Single Sign-on (SSO)
 - Domain Information
 - Information Broker
 - Notification framework
 - Presence framework



Application Domains



- Use underlying services
- Add specific functionality
- Voice over IP
 - Telephone services
 - SIP and H.323
 - Supplementary services
- E-Science
 - Call for Papers
 - GraphViz
 - TimeKeeper

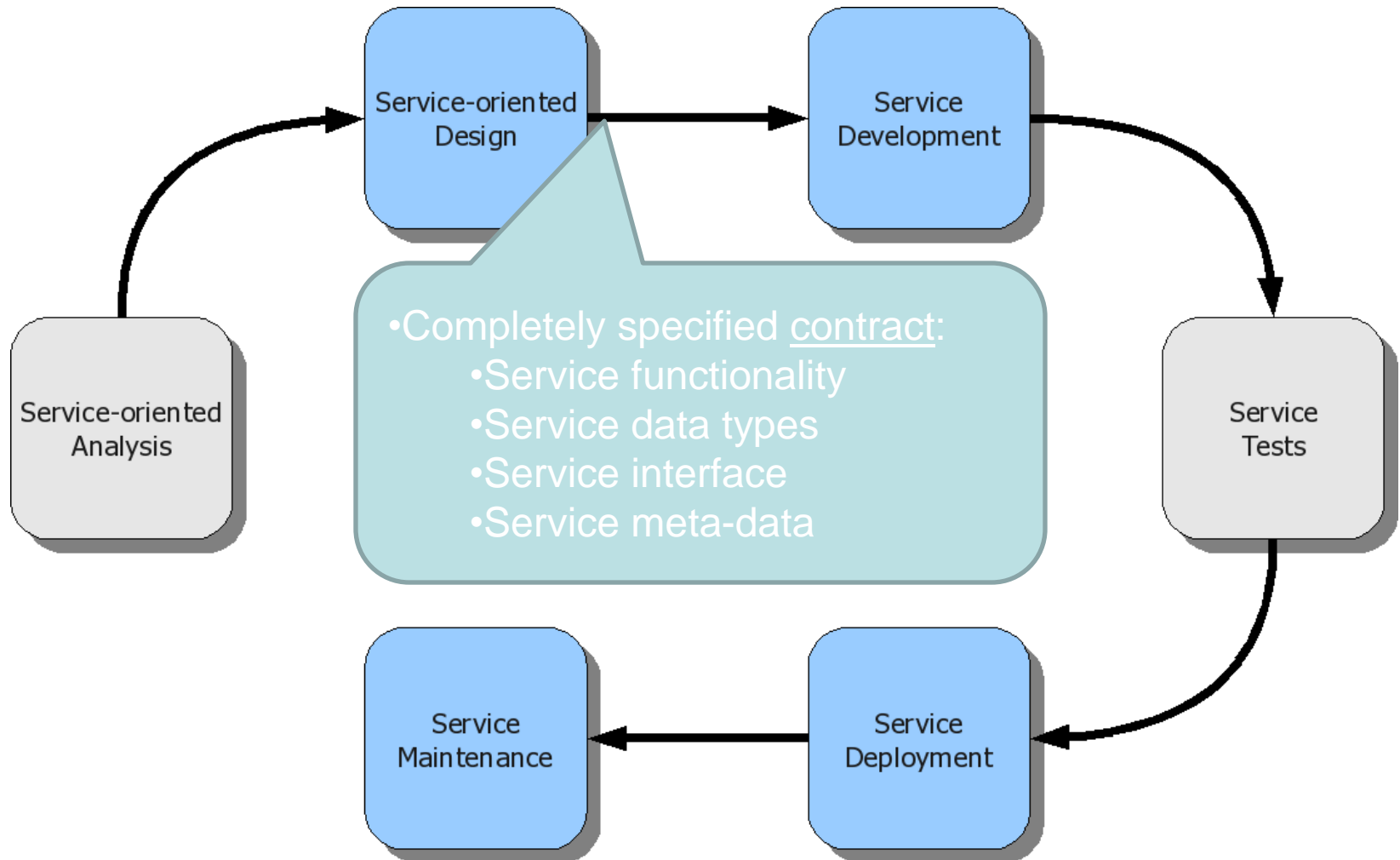


Outline

- Contract-first and code-first service development
- What is the Venice Service Grid?
 - Overview
 - Top Level View
 - Architecture
- What does the Venice Service Compiler?
 - Service life-cycle
 - Compiler Language EBNF
 - Example
 - Compiler Output
 - Automatically generated high-level functionality
- Conclusion and Outlook



Service Life-cycle



<http://www.icsy.de>

Compiler Language EBNF

```

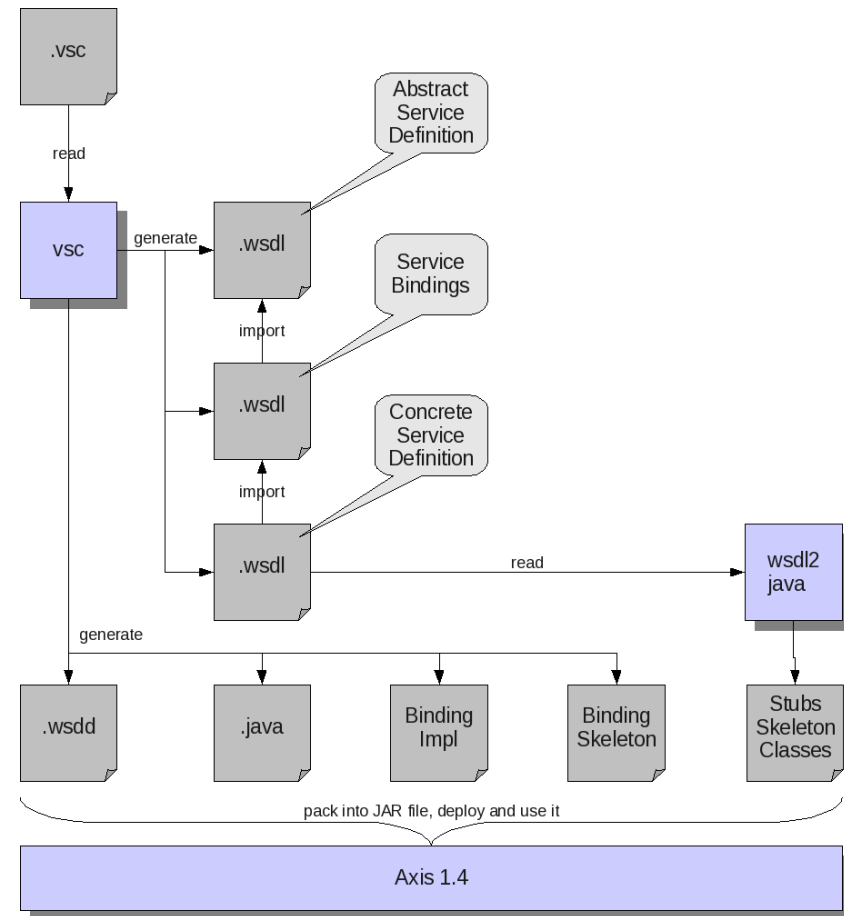
VSC                = ( ImportDeclaration )* ( TypeDeclaration )+ <EOF>
ImportDeclaration  = "import" <STRING_LITERAL> ";"
TypeDeclaration    = "interface" InterfaceDeclaration | "service" ServiceDeclaration
InterfaceDeclaration = "interface" <IDENTIFIER>
                    "{"
                    "namespace" <STRING_LITERAL> ";"
                    "location" <STRING_LITERAL> ";"
                    [ "implementation" <STRING_LITERAL> ";" ]
                    ( XSDTypesDeclaration )*
                    ( OperationDeclaration )*
                    "}"
XSDTypesDeclaration = "types" <IDENTIFIER> <STRING_LITERAL> ";"
OperationDeclaration = "operation" XSDType <IDENTIFIER>
                    "(" [ ParameterDeclaration ( "," ParameterDeclaration ) * "]"
                    [ "throws" ThrowsDeclaration ( "," ThrowsDeclaration ) * ] ";"
ParameterDeclaration = XSDType <IDENTIFIER> [ "nillable" ]
ThrowsDeclaration    = XSDType
XSDType              = ( <IDENTIFIER> ":" <IDENTIFIER> ) | "void"
ServiceDeclaration   = "service" <IDENTIFIER>
                    "implements" <IDENTIFIER> ( "," <IDENTIFIER> )*,
                    "{"
                    "namespace" <STRING_LITERAL> ";"
                    ( "location" <STRING_LITERAL> ";" )+
                    "package" <STRING_LITERAL> ";"
                    [ "baseclass" <STRING_LITERAL> ";" ]
                    ( "use" <STRING_LITERAL> "for" <STRING_LITERAL> ";" ) *
                    "}"
  
```

An Example Interface Definition

```
841 // This service type can be used to send emails.
842
843 interface EMail {
844
845     // Namespace and location
846     namespace      "http://www.icsy.de/wsd/basic/";
847     location       "http://www.icsy.de/~venice/wsd/basic/";
848
849     // Imported data types
850     types basic    "http://www.icsy.de/~venice/types/basic.xsd";
851     types email   "http://www.icsy.de/~venice/types/basic/EMail.xsd";
852     types domain  "http://www.icsy.de/~venice/types/domain.xsd";
853     types faults  "http://www.icsy.de/~venice/types/faults.xsd";
854
855     // Sets the email address
856     operation void
857         setEmailAddress(domain:SSOInformation sso, email:EmailAddress email)
858         throws faults:AuthorizationFault;
859
860     // Returns the stored email address
861     operation email:EmailAddress
862         getEmailAddress(domain:SSOInformation sso)
863         throws faults:AuthorizationFault;
864
865     // Sends an EMail
866     operation void
867         sendEmail(domain:SSOInformation sso, email:EMail email)
868         throws faults:AuthorizationFault;
869 }
```

Service Compiler Output

- Abstract service definition
 - wsdl:message, wsdl:portType
 - no internal data types
- Service bindings
 - Separate WSDL file
 - soap:style, soap:encoding
- Concrete service definition
 - Separate WSDL file
 - wsdl:service
- Implementation Code
 - Java source file
 - Code fragments
 - Higher-level functionality
 - Stubs and skeletons
 - Axis bindings and deployment





Higher-level Functionality

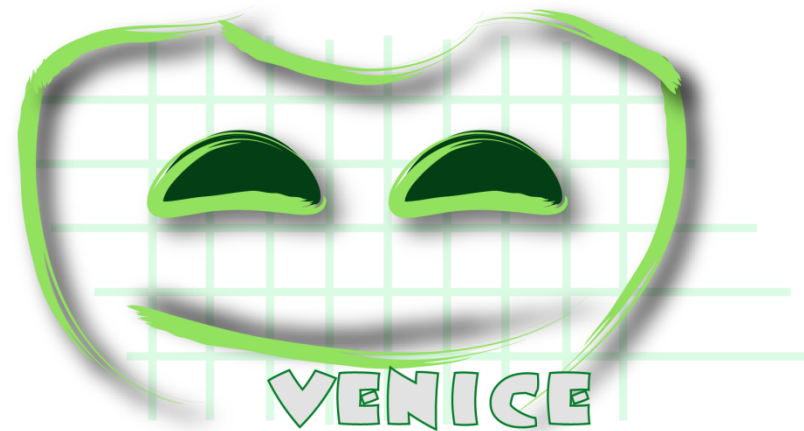
- WSDL compliance
 - Check WSDL compliance at run time
 - Helpful during development
 - Checks operations, faults, and data types
- Statistics
 - Count successful and unsuccessful service invocations
 - Count authentication and authorization faults
 - Identify bottlenecks, false configurations, or attacks
- State management
 - Notify users if a service is not ready to be used
 - Service downtime if backend is under maintenance
- Logging (via Log4J)
- Error handling
 - Well-defined error model
 - Convert all exceptions to faults
- Performance measurements
 - Insert code for sending performance records
 - Identify bottlenecks during development and test phase



Summary and Outlook

- The Venice Service Grid
 - Abstracts from Web services
 - Service infrastructure and runtime environment
- The Venice Service Compiler
 - Fragmented WSDL files
 - Java Implementation
 - WSDL compliance, statistics, state management
 - Error handling, logging, performance Measurement
 - Deployment
 - Documentation
- Outlook
 - Axis 2 and WSDL 2.0 support
 - Graphical frontend for the compiler
 - Manage service repository and data type repository

Thank you.
Questions?



<http://www.v-grid.info>

(still under construction)