

Eclipse Finance Day 2013

Eclipse technology in IFMS

Interface Management System

Marc Schlienger

A story today about Eclipse and IFMS

- SOA at Credit Suisse
- The construction of a System
- MDD in the large
- Leveraging assets for Modernization
- Outlook

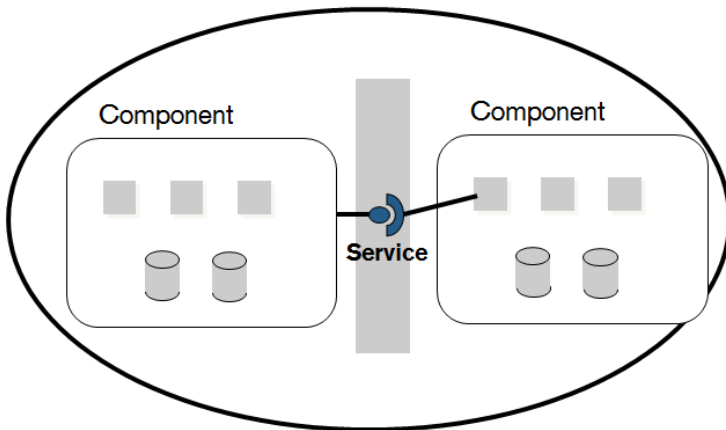
SOA at Credit Suisse

- Introduced for three major reasons
 - distributed computing (using CORBA technology)
 - standardize how services are documented and reviewed
 - centralize service documentation, foster re-use
- Overcome ongoing Challenges
 - People come and go, know-how gets lost
 - Application Landscape is aging
 - Technology diversifies
 - Manage complexity

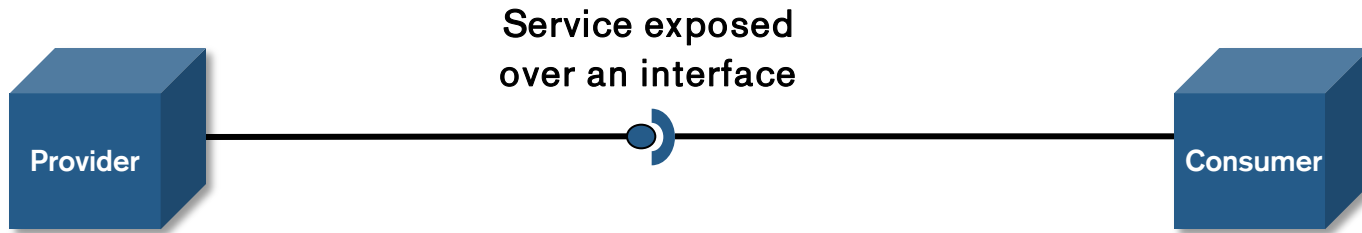


Decomposition into Components

- IT landscape decomposed into business domains
- These coarse-grained components are (de)coupled through services
- Services expose a business view



Services and Interfaces



IFMS makes SOA scale

- Interface Management System = central Service Repository
- Service and Data Type Catalog
- Service Contracts, Dependencies, Reviews
- Lifecycle Management
- Governance Enforcer
- Code Generator

- > 3'000 services in IFMS
- > 7'000 operations in IFMS



3 Perspectives on IFMS

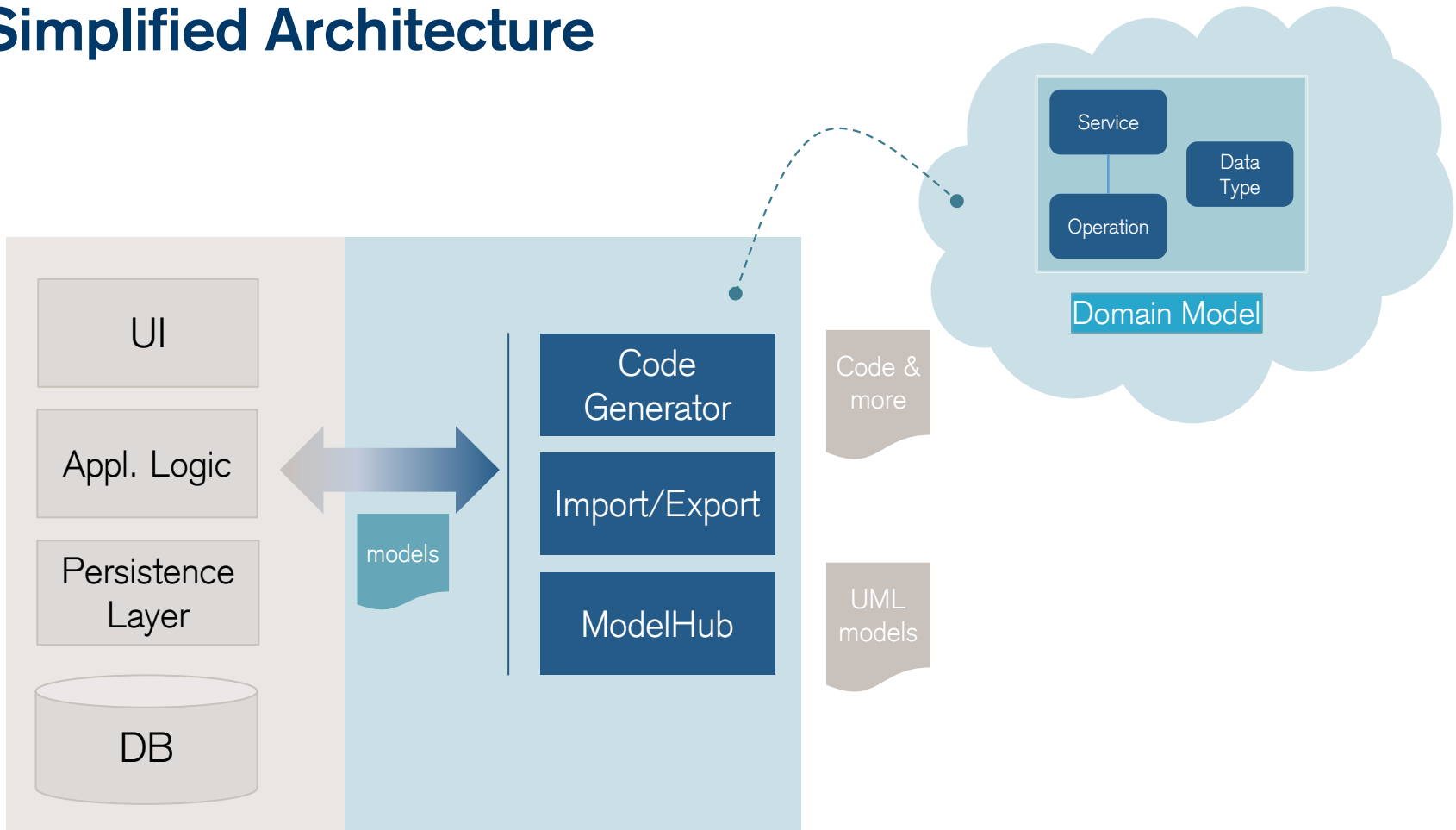
Construction

Scaling Factors

Modernization

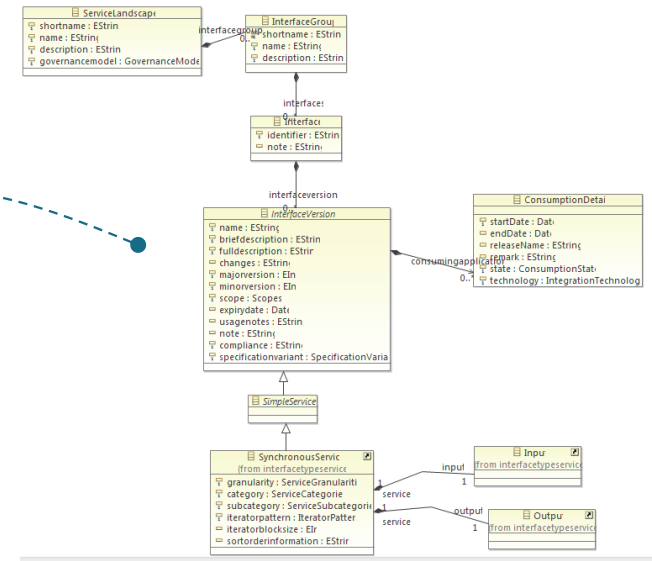
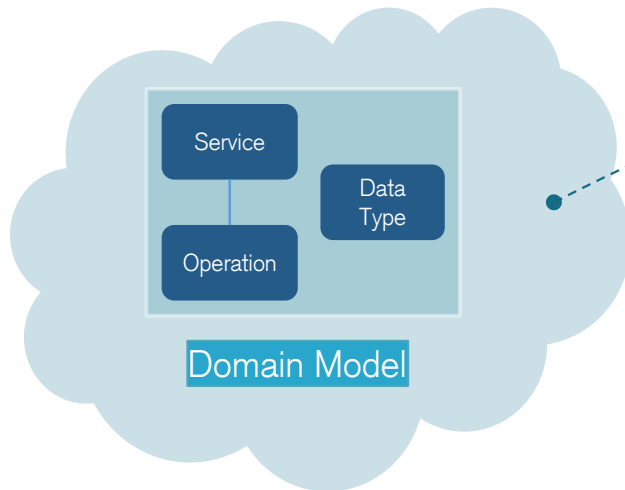


Simplified Architecture



Construction – the Data Layer

- Domain Modeling with EMF/ecore model
- Generate scaffolding for model-to-model transformation between Persistence Layer and EMF model
- XML serialization for transferring model data
 - Interface to Import/Export and Code Generator
 - Used for troubleshooting



Construction – Code Generator

- Code Generator part of Service Repository (centrally managed)
- Based on IFMS service models, generates:
 - PL/1
 - CORBA IDL
 - WSDL&XSDs
 - Java code
- Built on oaw (xtend, xpanse, check, mwe)
 - Express model validation concisely: check
 - M2M functional transformation language: xtend
 - M2T polymorphic template engine: xpanse
 - Reusing Abstract Syntax Tree and Java code serialization from Eclipse JDT

Construction – Import/Export and ModelHub

- Import/Export of model data expressed in terms of the domain model
 - Built using EMF Compare
- ModelHub for transforming from and to UML models
 - Xtend and ATL based transformations
 - Supports for RSM and Enterprise Architect XMI dialects

Scaling – Quality and Stability

Special needs for testing Code Generator

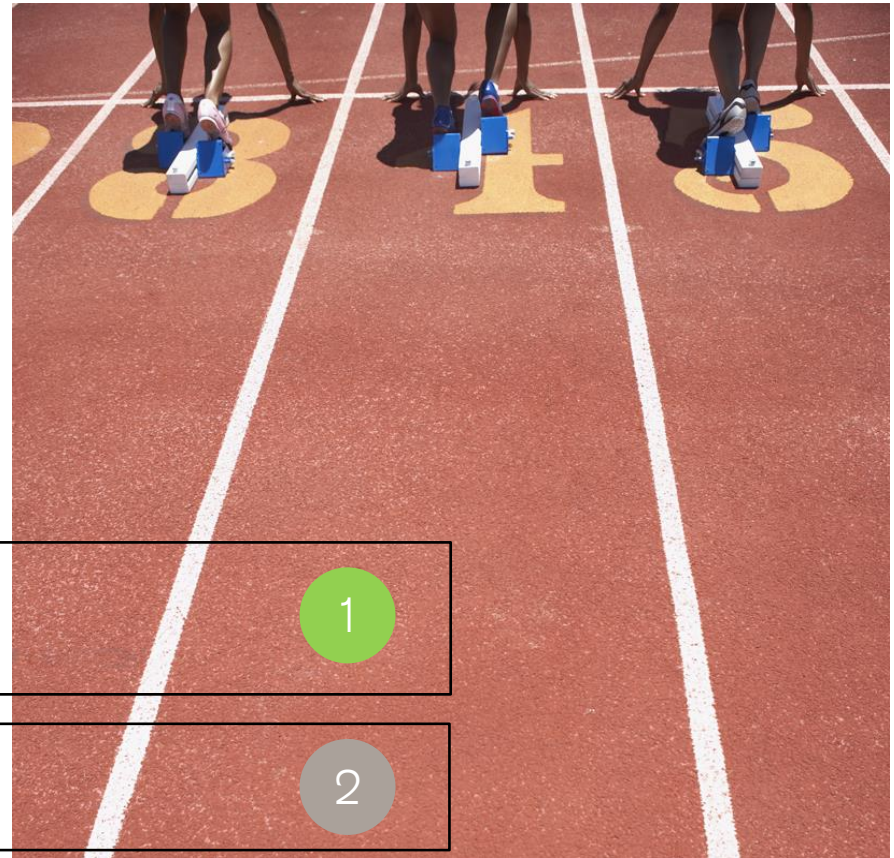
- Create test data (Builder Pattern on top of EMF model)
- Execute test
 - Normalize generated artifacts (remove differences due to moment of execution)
- Verify results
 - Normal JUnit asserts
 - File comparisons
 - Source code compilation
- Check model coverage
 - Annotations
 - Equivalence class matrix

```
Runs: 1503/1503 (5 skipped)  Errors: 1  Failures: 5
```

- com.csg.cs.log.ifms.generator.pli.PLIFormatterTest [Runner: JUnit4]
- com.csg.cs.log.ifms.generator.pliws.PLIWSGetProvConsolBo
- com.csg.cs.log.ifms.generator.wsdl.WsdlElementaryTypeTes
- com.csg.cs.log.ifms.generator.async.AsyncStructureTest [Ru
- com.csg.cs.log.ifms.generator.corba.IDLUnionTest [Runner:
- com.csg.cs.log.ifms.generator.jsb.JSBElementaryTypeTest [R
- com.csg.cs.log.ifms.generator.jsb.JSBServiceTest [Runner: JUnit4]**
 - testTechnicalNameForJAXBNaming (19.948 s)
 - testExceptionsAndAttentions (18.976 s)
 - testOutputFieldNamedBusinessException (1.140 s)
 - testNoInputParameter (17.655 s)
 - testMissingPlatformSpecificBinding (0.784 s)
 - testAggregatedNoInputParameter (16.643 s)
 - testAggregatedNoOutputParameter (17.886 s)
 - testAggregatedWithExceptionsAndAttentions (17.718 s)
 - testWrongTechnicalName (0.000 s)
 - testAggregatedAllParameterTypes (24.075 s)
 - testWrongFileName (0.000 s)
 - testTechnicalName (14.337 s)
 - testOneServiceInTwoDifferentDUVs (0.000 s)
 - testOneDUWithTwoDVsWithOneServiceEach (18.290 s)
 - testFileName (16.482 s)
 - testNoOutputParameter (17.007 s)
 - testTwoMinorVersions (0.000 s)
 - testActiveServicesInDUHavingDifferentGenVersion (0.000 s)
 - testOneDUWithTwoDVsWithTwoServicesEach (0.000 s)
 - testAggregated (16.643 s)
 - testAllParameterTypes (20.424 s)

Scaling – Performance

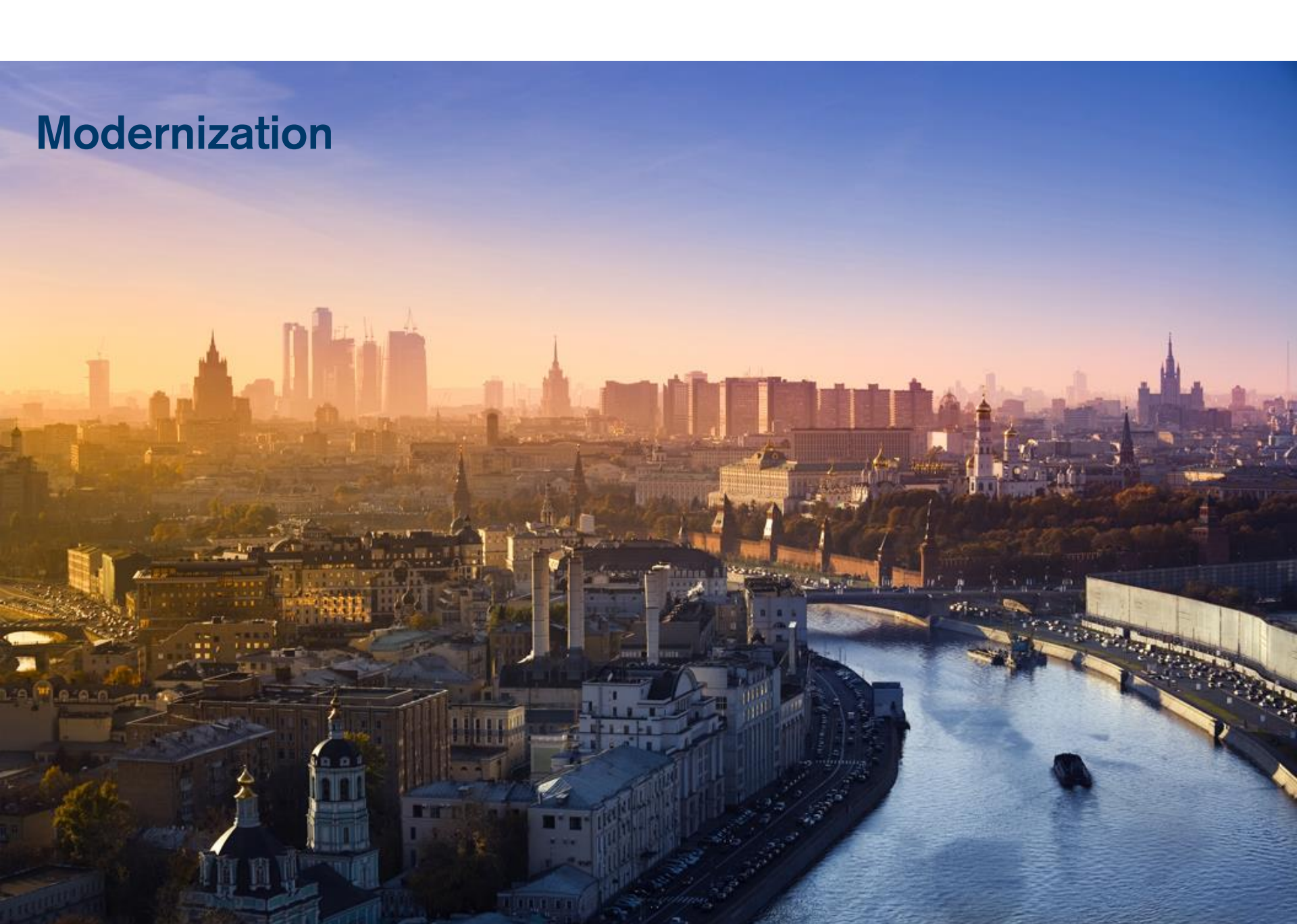
- Large user base (ca 400 in 2013)
- Generator started 2'600 times in 2013 (up to 150 per day)
- Limitations of oaw (xtend 1)
 - Slow, Java interpreted
 - Needs huge stack
- M2M vs M2T
 - Flexibility vs Readability
 - Fine vs Coarse granular objects



- Generator in separate Server/JVM
- Generator as a Service

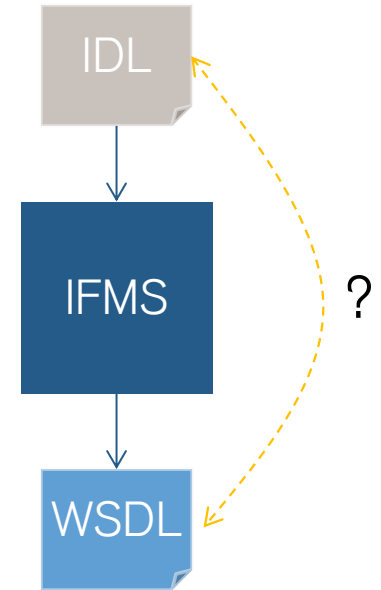
- Migrate to xtend 2

Modernization



Leveraging existing assets

- IFMS central in CORBA to WebService migration
- Import existing CORBA IDLs
- Generate diff models describing IDL vs WSDL
 - Leveraged for automatic testing
- Xtext based IDL Parser
 - Simplifies parser writing
 - EMF based models
- Groovy for intermediate transformations
 - Concise and elegant syntax
 - Mind the troubles when searching for errors



Outlook

There are many MDD styles (bold = IFMS style)

- Metamodel/Language: generic vs. **specific** (UML vs. DSL)
- Modeling Tool: trim existing case tool vs. **build specific one**
- Editor: graphical vs. textual vs. **forms-based** vs. **combination**
- Build overall system vs. build **specific parts** of a system
- Tool deployed **centrally** vs. available within the IDE
- Model transformations
- Store and manage models **centrally** vs. decentralized
- Physical model representation/store: **RDB**, XMI, Other

Thank you!

Marc Schlienger
marc.schlienger@credit-suisse.com