innoQ Deutschland GmbH
D-40880 Ratingen
Tel +49 2102 77 1620

innoQ Schweiz GmbH
CH-6330 Cham
Tel +41 41 743 01 11

www.innoq.com

SOA and MDE

Stefan Tilkov, stefan.tilkov@innoq.com

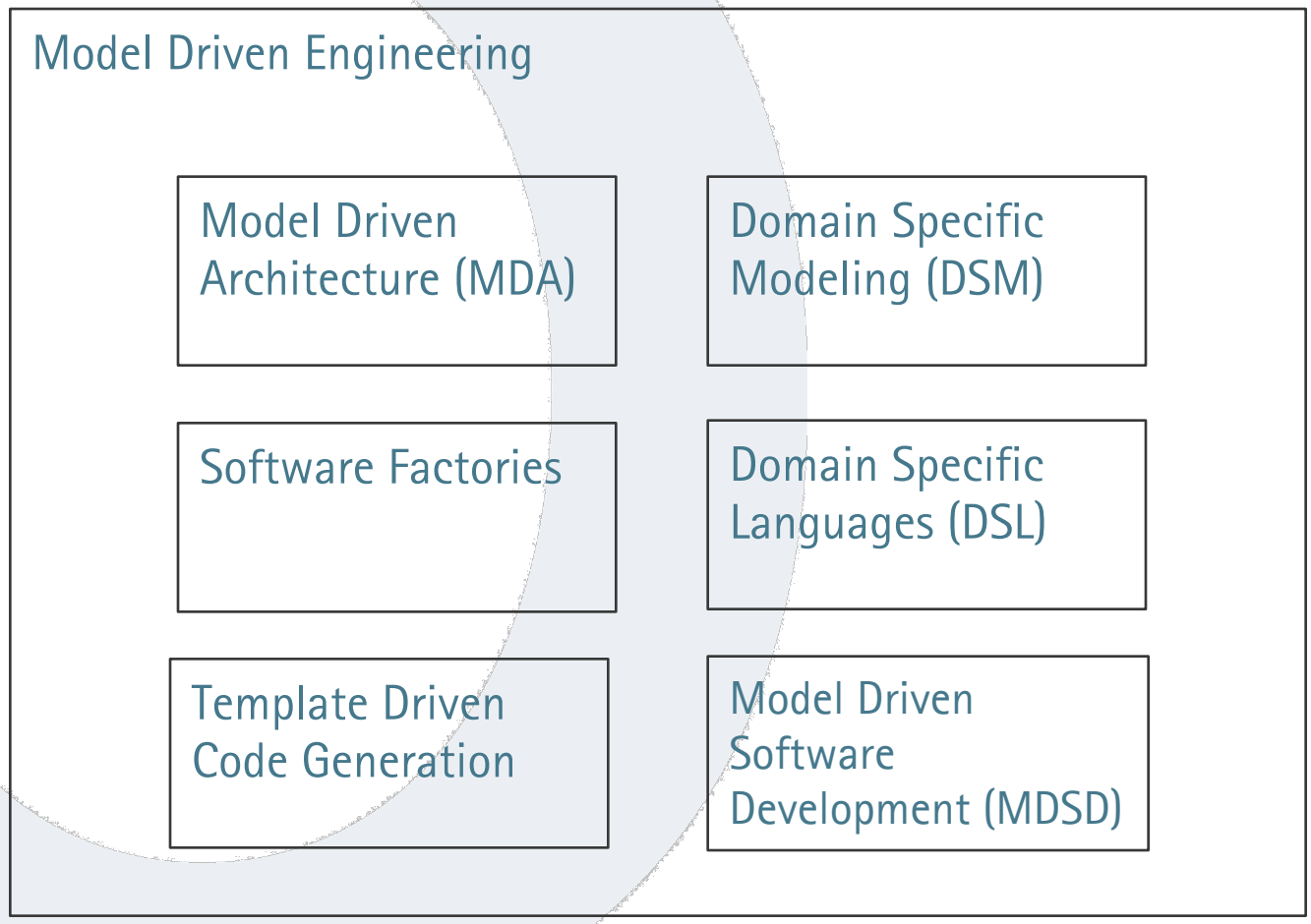
Goals

- Introduce MDE, MDA, MDD, MDSD, ...
- Define SOA
- Explore Relationships
- Show how MDx and SOA might be combined

innoQ

Part I: MDE

MDE Areas


Model Driven Engineering
Model Driven Architecture (MDA)
Domain Specific Modeling (DSM)
Software Factories
Domain Specific Languages (DSL)
Template Driven Code Generation
Model Driven Software Development (MDSD)

Differences

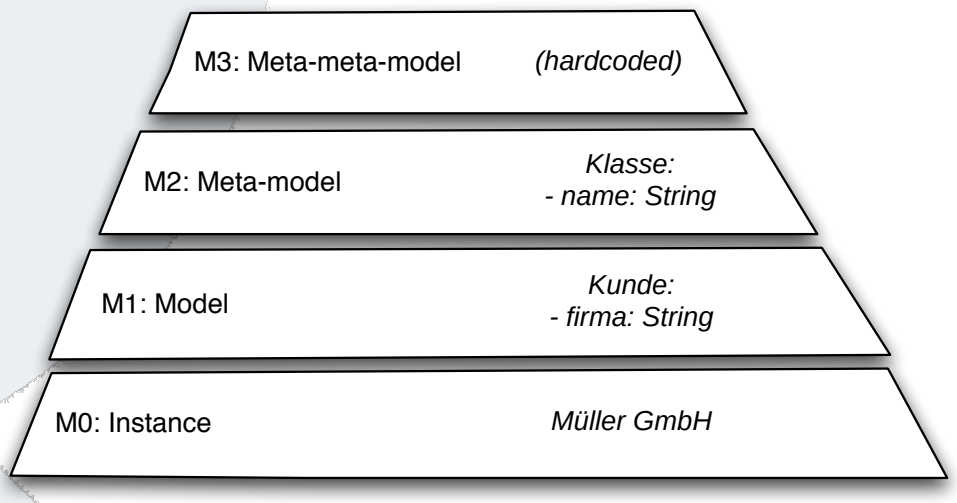
- Many of them more political than technical
- Microsoft vs. the rest of the world (OMG)
- OMG/MDA focuses on UML as a generally accepted language
- Others (Microsoft, Eclipse/EMF) highlight custom metamodels/ domain languages

Common Traits

- Models are more than documentation – they are formal, machine-readable
- Models follow (adhere to) a *metamodel*
- Models are transformed into other models – either automatically or manually
- The main goal is automation

Metamodel Layers

- The metamodel defines the language used to express models
- Meta-metamodel defines language to define metamodels
- Multiple metamodels means multiple languages

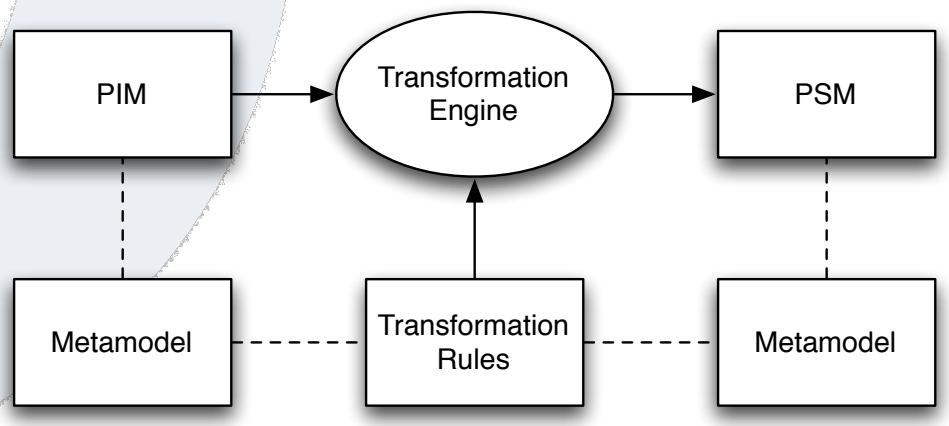


Example Modeling Languages

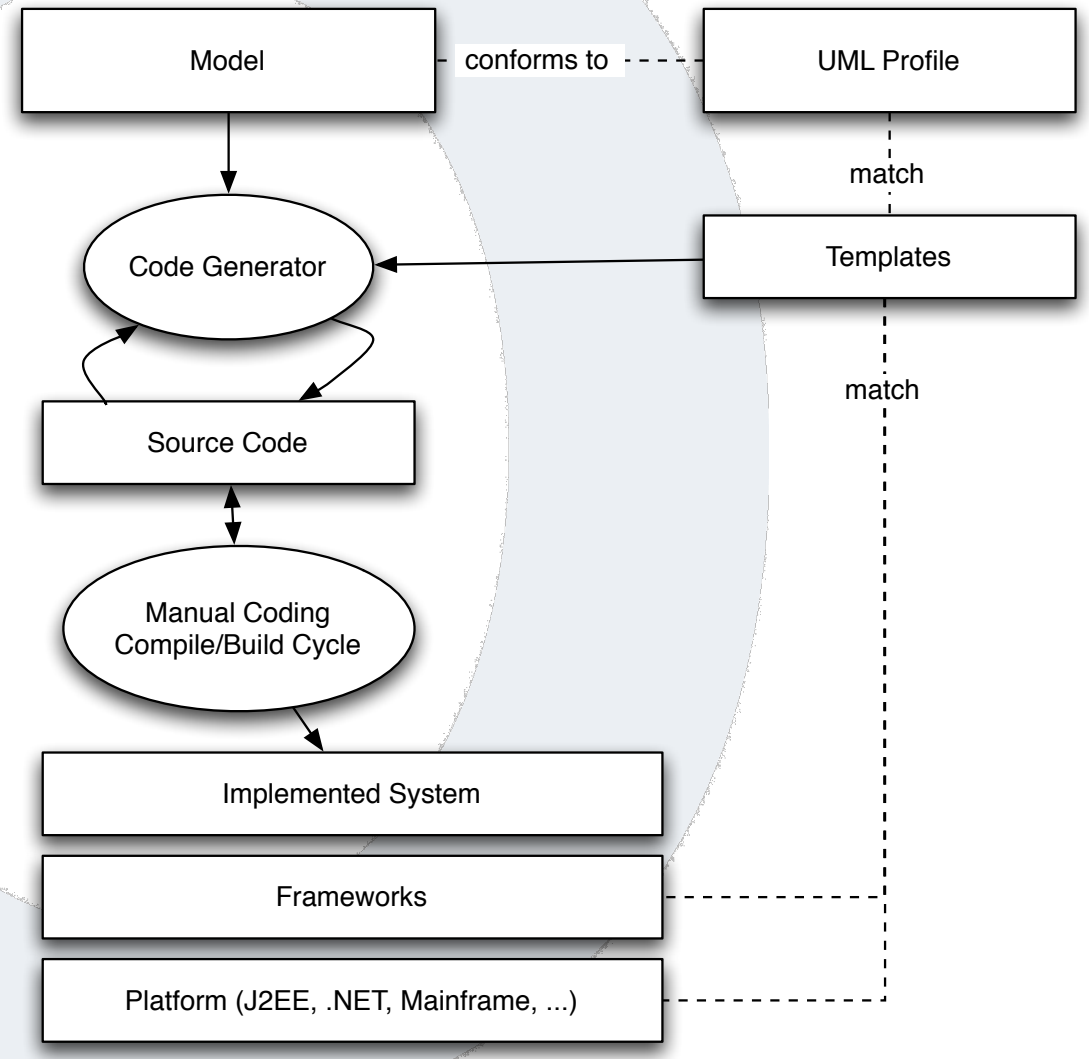
- UML
- MOF, EMF ECORE
- E/R
- eEPK
- SA/SD

Model Transformation

- Platform-Independent Model (PIM) is transformed into a platform-specific model (PSM)
- Optionally repeated multiple times – one guy's PSM is the other guy's PIM



MDA in Practice



MDE Today

- Many successful projects based on 1st generation MDA (UML/code-generation)
- Model-to-model transformation in early stages
- DSM/DSL (MS vs. OMG) debate
- Expect MDA Hype to turn into Anti-Hype

MDE Summary

- MDE and its many disciplines focus on models
- There will always be more than one model
- There will always be more than one metamodel, too

innoQ Deutschland GmbH
D-40880 Ratingen
Tel +49 2102 77 1620

innoQ Schweiz GmbH
CH-6330 Cham
Tel +41 41 743 01 11

www.innoq.com

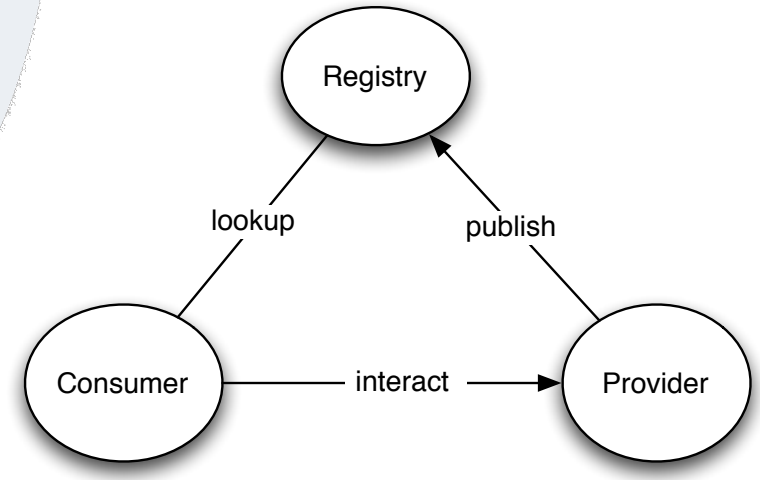
Part II: SOA

SOA Definition

An SOA is an enterprise-level IT architecture where the main construction principle is a *service*. Services encapsulate business functions that can be invoked remotely via document-oriented, standards-based interfaces which are discoverable at runtime and described by functional and non-functional metadata.

The SOA Trinity

- A *Provider* defines *Services* and publishes them to a *Registry*
- A *Consumer* can look up a service and interact with the provider
- The registry captures service metadata



1st Generation SOA

- Services expose interfaces, hide implementation
- Communication via interoperable standards (SOAP, XML, HTTP)
- Service description (WSDL) mainly for code generation
- *CORBA with angle brackets*

Advanced SOA

- Services expose document-oriented interfaces, with message formats described in XML Schema
- Messages are self-descriptive
- Non-functional aspects are described by Policies
- Services are accessed and combined based on service metadata

Consequences

- Concepts become more important than implementation technology
- Loose coupling in multiple dimensions
- Metadata as the connecting element

innoQ Deutschland GmbH
D-40880 Ratingen
Tel +49 2102 77 1620

innoQ Schweiz GmbH
CH-6330 Cham
Tel +41 41 743 01 11

www.innoq.com

innoQ

Part III: Model-driven SOA

Key Insights

- A single, unified, enterprise data or process model would be great – but is next to impossible to achieve
- Even a single standard for modeling is not going to be easy to define
- Even if one could unify everything, the next merger or acquisition would re-create the problems

Embrace what Exists

- Organizational models (roles, groups, users ...)
- Deployment models (systems, subsystems, applications, modules, components)
- UML application analysis and design models
- E/R logical/physical DB models
- BPM models
- Product and Pricing models
- System/network topology models
- ...

SOA Metamodel

- Vision: Create a unified SOA *meta*model
- Service as key abstraction
- Tailored to adapt to existing diversity
- Focus on process instead of metamodel wars
- Integrate and adapt to change

Registry vs. Repository

- SOA concepts include a registry
- Registry contains pointers – not actual data
- Repository as unified data store for model artifacts *of differing metamodels*
- Custom solutions as well as standard products emerging

Repository Runtime Use

- Service implementations can create metadata
- Service call statistics, usage patterns, dynamic relationships
- Feedback loop can influence code generation, deployment, policy

Harvesting

- The most valuable model information is in existing applications
- Usually not in the form of models, but as compiled code
- Model-driven reverse engineering enables access

MDRE Variants

- Extraction of Structural Information via static analysis
- Extraction of Behavioral Information via runtime analysis
- Possibly (meta)model-driven
- OMG Initiative: ADM (Architecture-Driven Modernization)

Drivers for MDSOA

- Complexity
- Re-Use
- Governance/Policy Enforcement
- Manageability
- Regulations (Sarbanes Oxley, Basel II, ...)

innoQ

Case Study

Model repository for service metadata in one of Switzerland's largest banks

Project Goals

- A single Repository for all service definitions, data types, structures and their relationships
- Simplify process, tooling
- Reduce redundancy
- Accelerate development time
- Reduce cost
- Improve quality

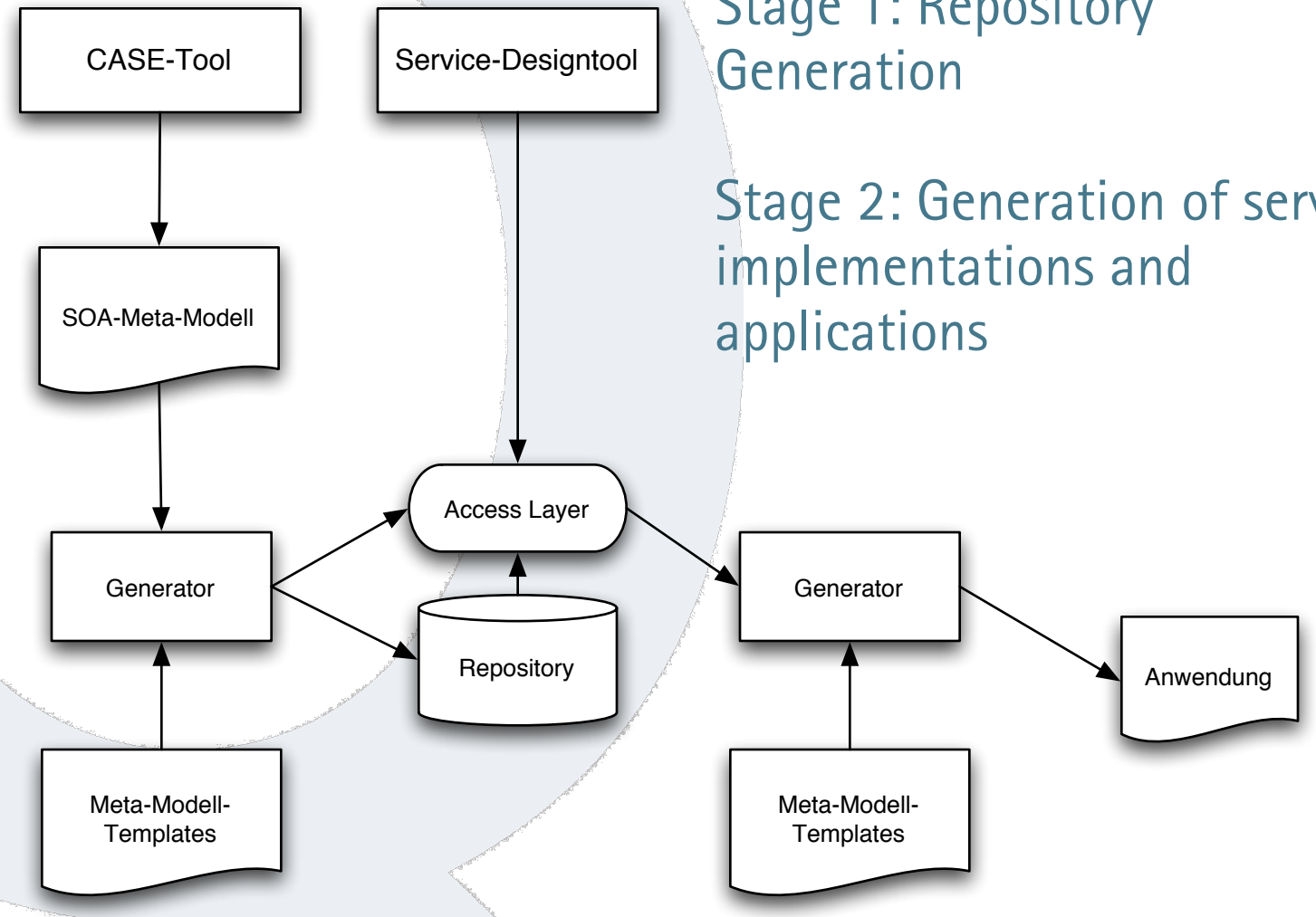
Project Goals (2)

- Improve Re-Use
- Simplify standards-conformant service implementation
- Establish feedback loop
- **Lay foundation for governance**

Repository Development

- Metamodeling using UML
- XMI Export
- Generation of database schema, access code, UI via templated generator (iQgen)
- Service design *without* CASE tool

Construction Processes



Stage 1: Repository Generation

Stage 2: Generation of service implementations and applications

Repository Components

- Service repository backend
- Service design tool based on Eclipse/ WSAD
- Browser-based client
- Code Generator
- Interfaces, API and importers for
  - Data types
  - Data structures
  - Service interfaces
  - Database schemas
  - Workflow management
  - SCCM
  - ...

Service Development

- Alternative strategies:
  - Import of Model or DTD
  - Copying of existing services
  - Data structure re-use („from scratch“)
- Optional: Add platform-specific transformation hints
- Generate service provider or consumer implementation
- Add custom business logic if necessary
- ...

Generated Artifacts

- Service interface description (DTD, XML)
- Human-readable service description (HTML)
- Cobol Copybooks
- Cobol Entity Service implementations
  - Read, ReadList, Insert, Update, Delete
- Cobol Process Service implementations
  - Composite services
- Web Presentation: JSPs, Servlets, XML-Descriptors, etc.
- J2EE: Business-Delegates, DTOs & Session-Facades
- All of the model information 100% in sync with current implementation

Summary: Model-driven SOA

- Knowledge is explicit
- Diversity is accepted
- Dependencies are managed
- Automation is key

innoQ

Questions?

Contact

innoQ Deutschland GmbH
Halskestraße 17
D-40880 Ratingen
Tel +49 2102 77 1620
Fax +49 2102 77 1601

innoQ Schweiz GmbH
Gewerbestrasse 11
CH-6330 Cham
Tel +41 41 743 01 11
Fax +41 41 743 01 19

Web Sites

innoQ www.innoq.com
iQgen www.innoq.com/iqgen