8th European Conference on Software Architecture, 27.8.2014, Vienna

ECSA 2014 Vienna, Austria

Modeling Principles

A Survey of Current Modeling Approaches in Industry and Where the Journey May Go

Dr. Horst Kargl

SPARX SYSTEMS

Who is Sparx Systems ?

- Sparx Systems is a modeling tool vendor.
  - Located in Australia (Cresswick next to Ballarat next to Melbourn)

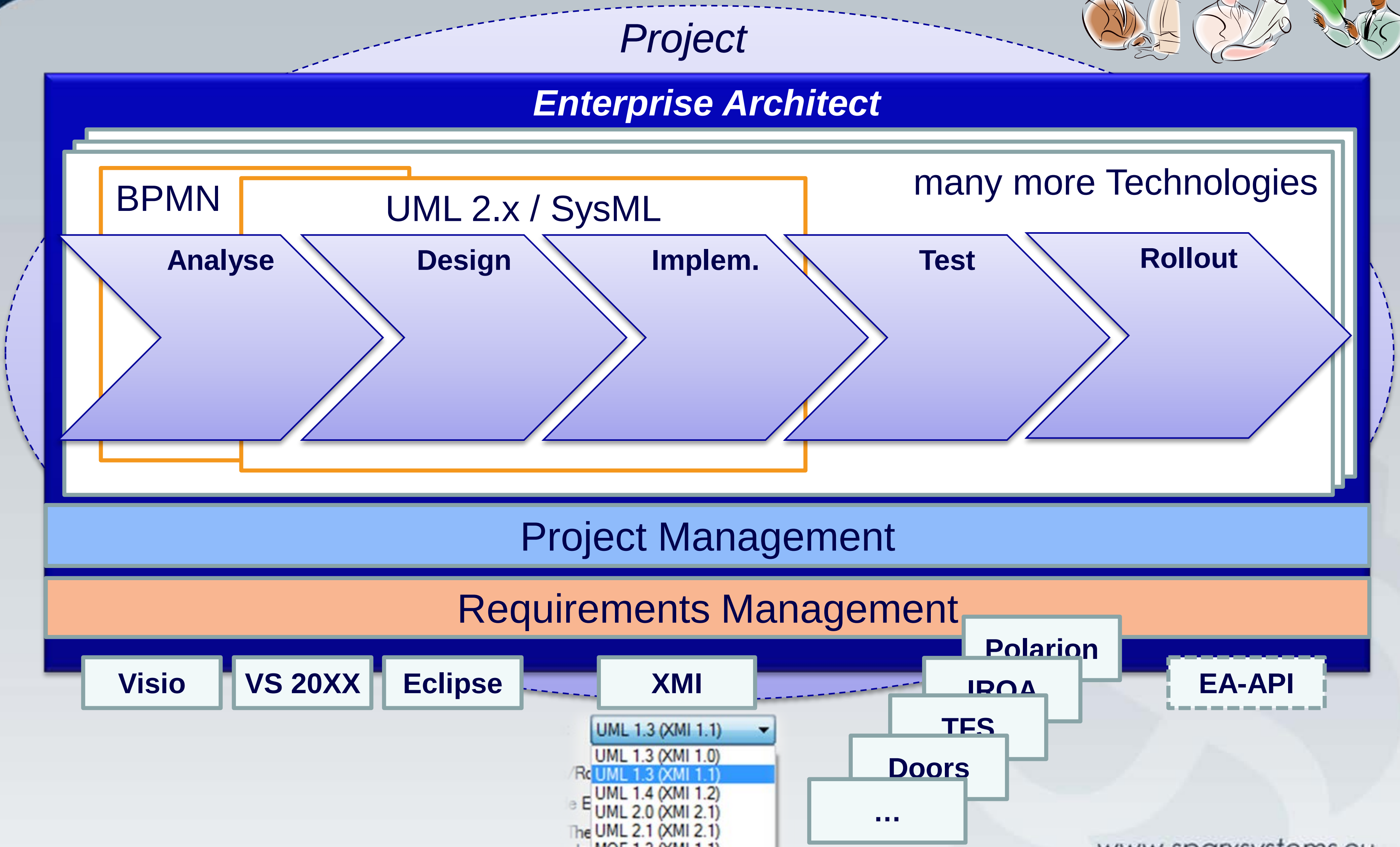
- Sparx Systems Central Europe is a sister company of Sparx Systems
  - Located in Austria (Vienna)
  - Training, License, Consulting, Customizations

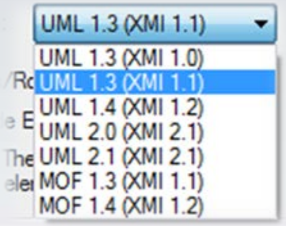
- 350.000 ++ licenses sold worldwide

- Used by companies of all sizes from large, well-known multi-national organizations to smaller independent companies and consultants in the domain of:
  Finance, Insurance, Embedded Systems, Automotive, Geo-Information Systems, IT, etc.

www.sparxsystems.eu

What is Enterprise Architect ?





UML 1.3 (XMI 1.1)
UML 1.3 (XMI 1.0)
UML 1.3 (XMI 1.1)
UML 1.4 (XMI 1.2)
UML 2.0 (XMI 2.1)
UML 2.1 (XMI 2.1)
MOF 1.3 (XMI 1.1)
MOF 1.4 (XMI 1.2)

www.sparxsystems.eu

Topics

- Current practice in modeling
- What is important when modeling in corporate practice
- Painting or Modeling?
- For each user group the right representation
- From single person project to worldwide teamwork
- Necessary Skills
  - Languages
  - Methods
  - Tools
  - Experience

Current Practice in Modeling

- Why?
  - Tools are available.
  - Everybody knows how to use them
- Is it good?
  - Read 10-50 pages of a document
  - Have a look at 1-3 Diagrams
- Alternatives?
  - Standardized modeling languages
  - Good tool support
  - Experienced user

www.sparxsystems.eu

What is important when modeling in corporate practice

Customer

It should be **easy** to build complex systems! Hence, a modeling language (or tool) should **hide everything complex**.

It should be **impossible to make a mistake**!

I will keep independent **versions** of the same model and **merge** them from time to time!

I will have the **flexibility** to do whatever I need, a tool should support **possibilities**.

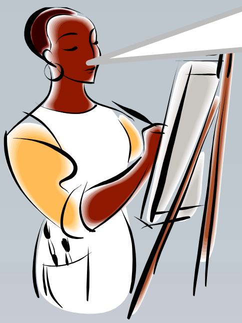
My **architecture** and the system **implementation** should always be **up to date** and **synchronized**!

Modeling languages!? I will only describe my system with the language, **I just need part of it**.

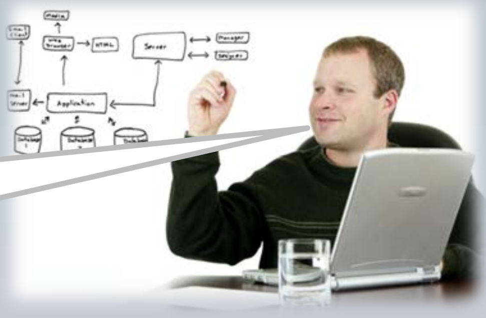
Modeling languages!? Well, what exists is not powerful enough for my systems. There is **still a lack of expressiveness**.

www.sparxsystems.eu

Painting or Modeling ?

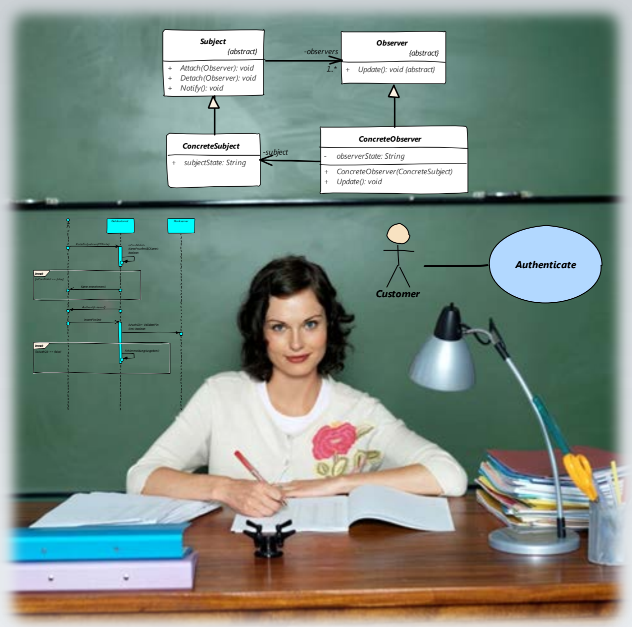


Modeling on a white board, flip chart or even electronic paper is just painting, even if you use standardised modeling languages like UML or other.



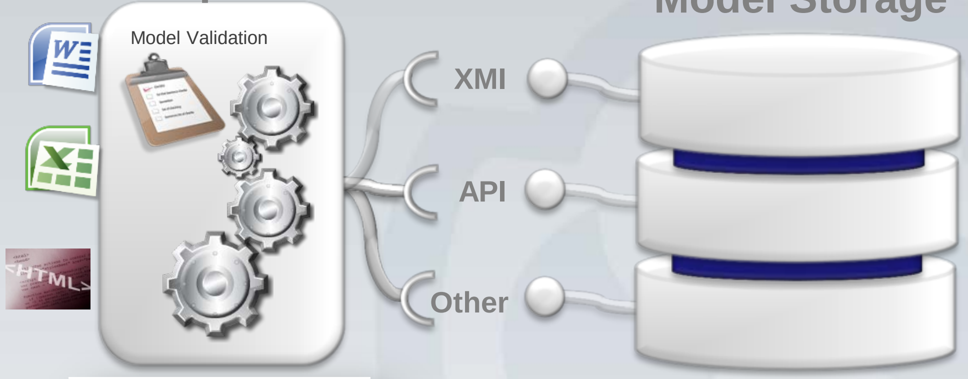
When I store my models in a model repository, I have the possibility to further process my models automatically. For instance, perform a model check, generate paper documentation, generate code etc.

Graphical Representation of a Model

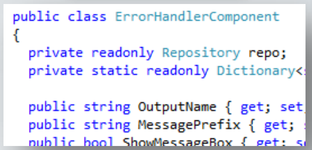


Model Processing Component

Model Storage



Code



```
public class ErrorHandlerComponent
{
  private readonly Repository repo;
  private static readonly Dictionary<

  public string OutputName { get; set
  public string MessagePrefix { get;
  public bool ShowMessageBox { get; s
```

www.sparxsystems.eu

For each user group the right representation

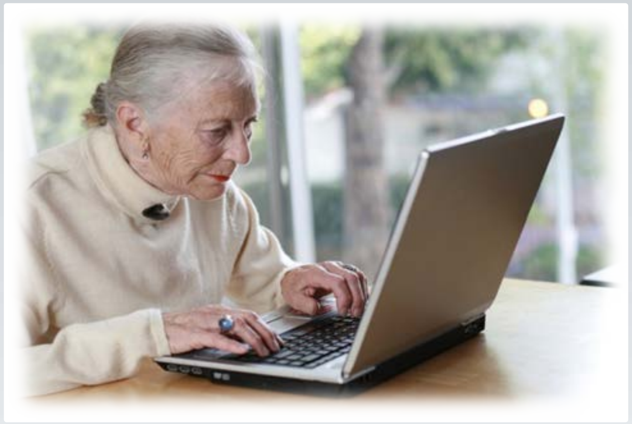
Customer

Requ.Eng.

Architect

Developer

Tester


End User

Who reads the model?

What should be read?

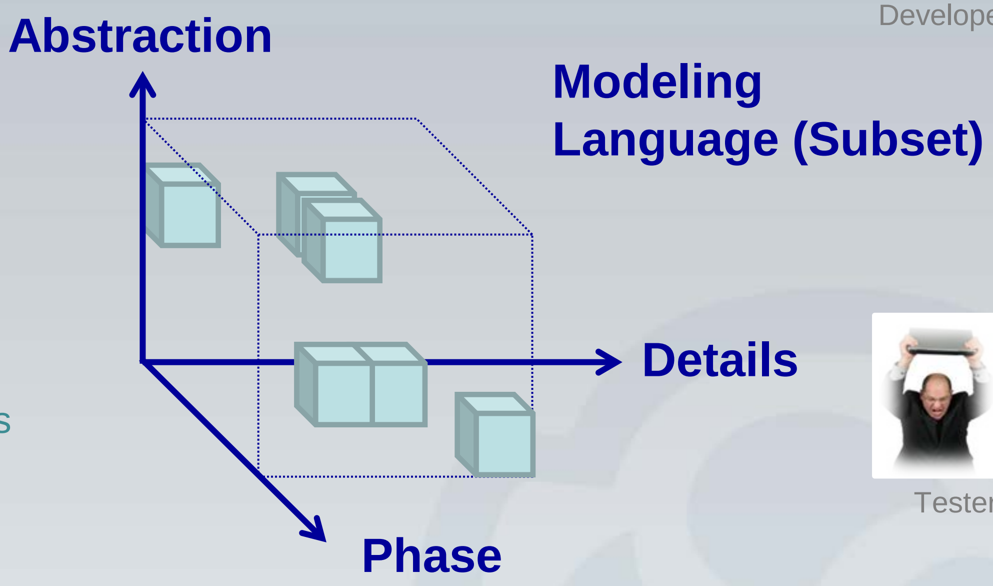
Support

www.sparxsystems.eu

For each user group the right representation

Select the proper representation for each user group!

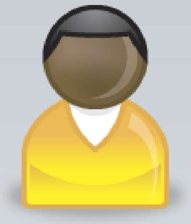
How to keep the models in synch!?



www.sparxsystems.eu

From single person project to worldwide teamwork

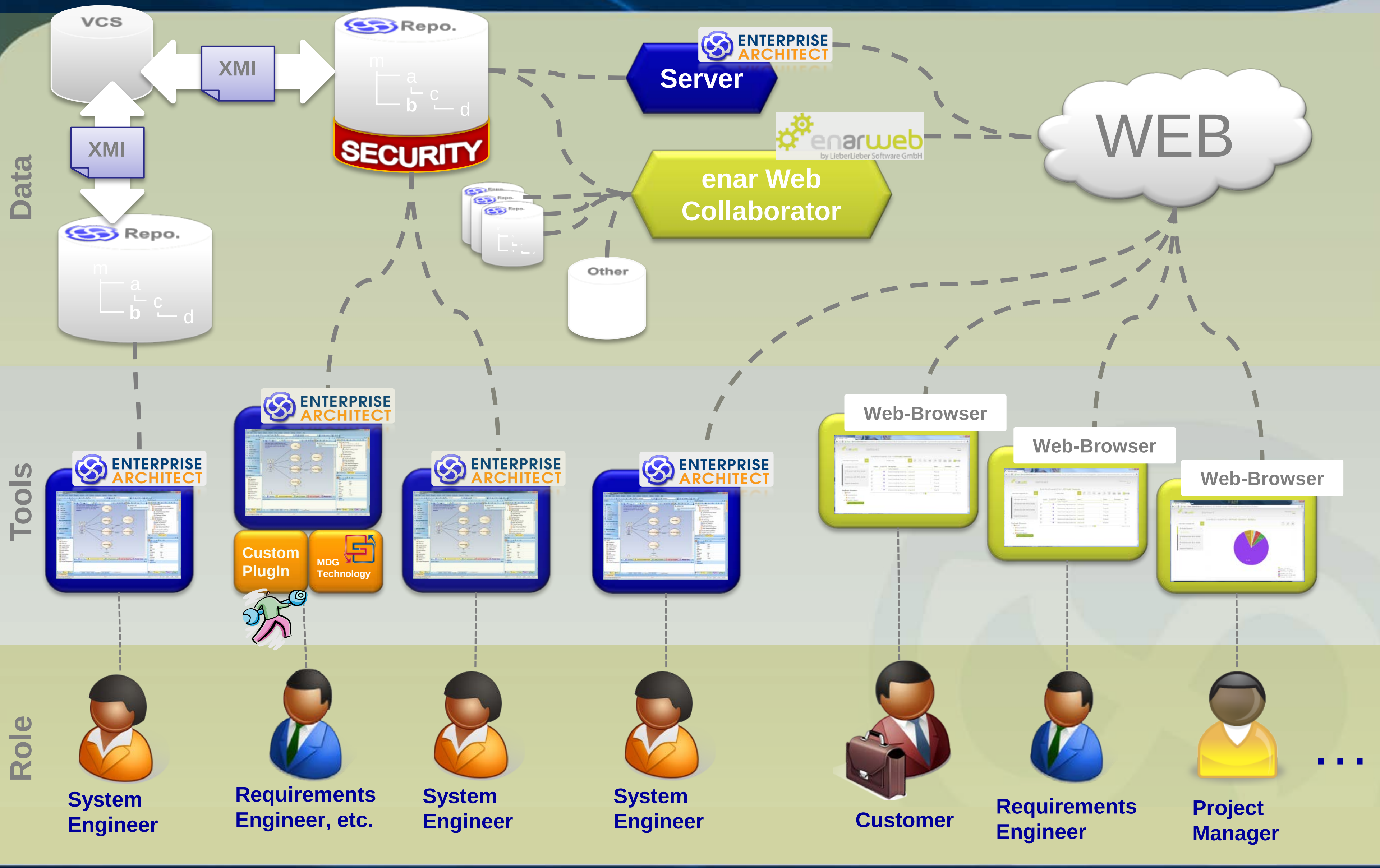
The main topics, when people start working together:



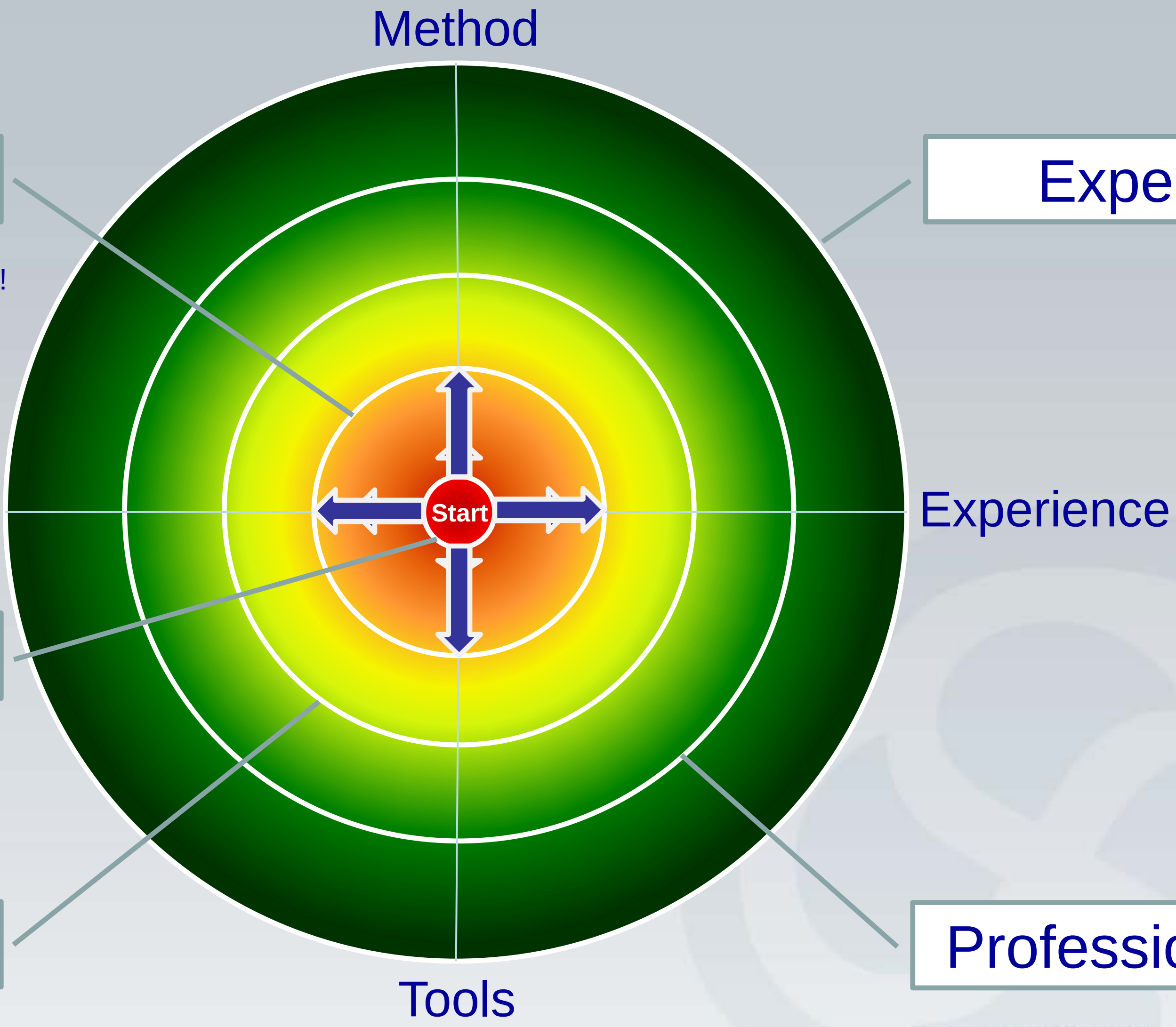
- How to exchanging information
- Versioning
- Variant management
- Secure Access
- Hide Information
- Trust
  - In the process
  - In the used tools

www.sparxsystems.eu

Exchanging Information



Skill Levels



Fundamental

... all dimensions should be
at least on fundamental level!

Expert

Novice

Advanced

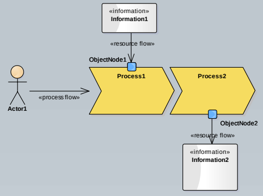
Professional

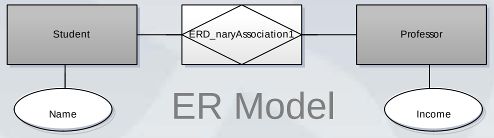
Languages

- Languages like UML, SysML etc. define the syntax and the semantics, but not how to use them.
- UML has many language concepts, but not all of them must be used.

- Depending on the experience level, select a subset of the modeling languages.
- Hide everything else.
- Do not allow other modeling elements.
- Provide suggestions.

Eriksson Penkner





ER Model

www.sparxsystems.eu

Method

- Methods like Waterfall, RUP, V-Model or Processes like Scrum, Kanban etc.
- How to use the modeling language (the selected subset) in your team with your preferred method?

- Prepare an example model (a reference model).
- Describe the rules. (modeling rules and guidelines)
- Configure and customize the used tools to provide a simple efficient working environment.

Tools

- The powers of modeling languages can only be unleashed when they are supported by tools.

- Find the proper tool, which supports all of your needs. Like:
  - Validation
  - Documentation
  - Code Generation
  - Model Transformation
  - Simulation
  - Customisable
  - etc.

Tools

www.sparxsystems.eu

Experience

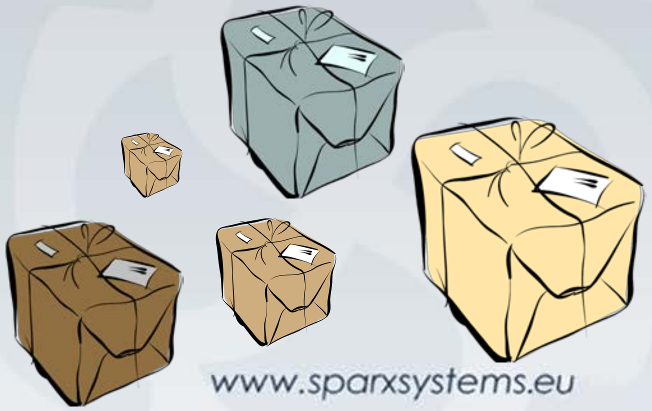
- Don’t talk about the path, walk it!
- The only way to find out what you really need is to do what you need and check if it’s enough.

- Do not hesitate to start.
- Always discuss with your colleagues about your approaches.
- Try to read other models.

www.sparxsystems.eu

Summary

- Modeling in practice needs the proper package:
  - A balance between *modeling languages*, *tools*, *methods* and *experience*
  - A team with a similar skill level (fundamental and above!)
  - Trust in the consistency and correctness of the model
  - Methods for different domains and user groups
  - Tools which support methods
  - Tools which can be adapted and customized to methods



www.sparxsystems.eu

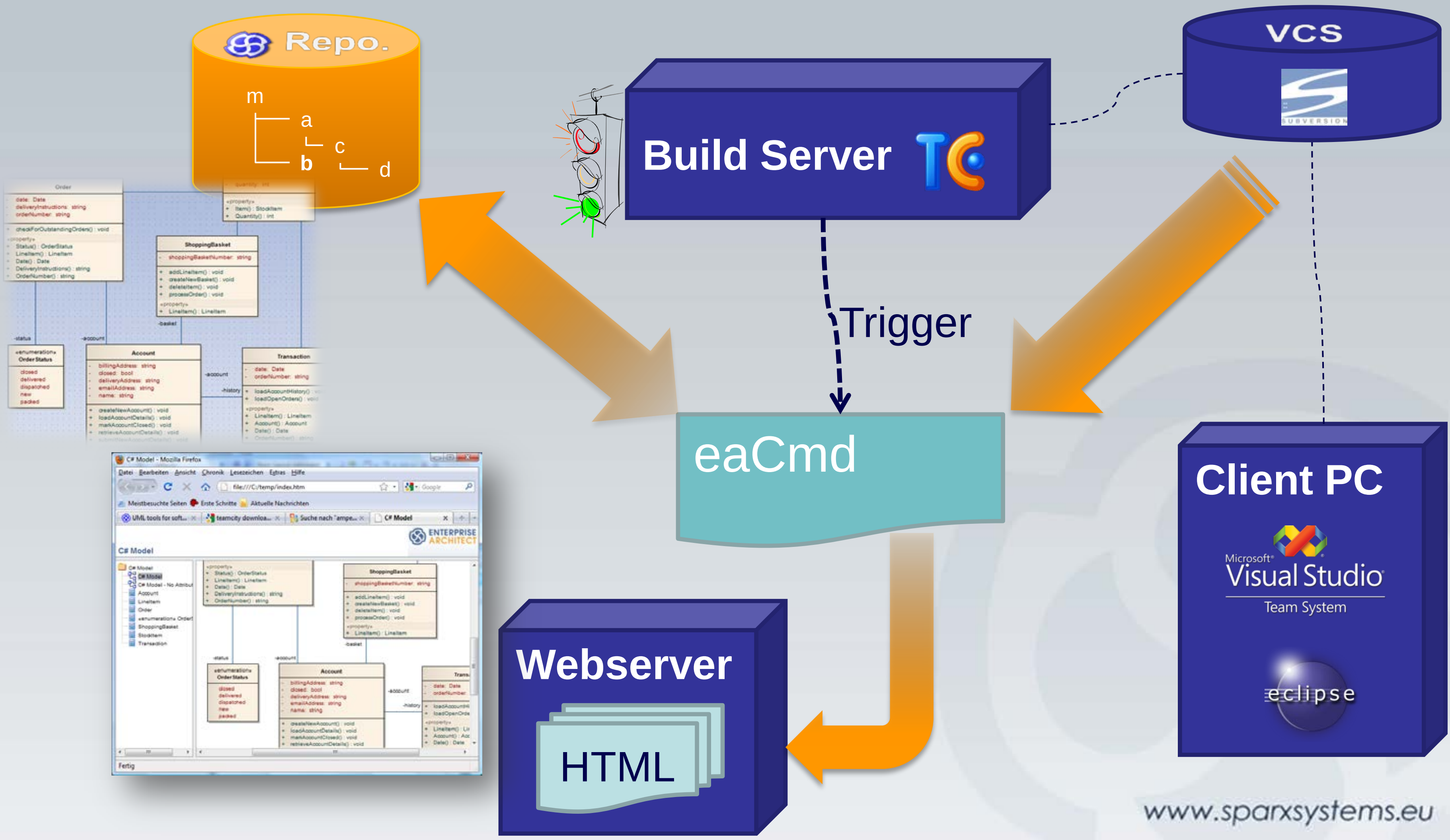
Summary

Backup

www.sparxsystems.eu

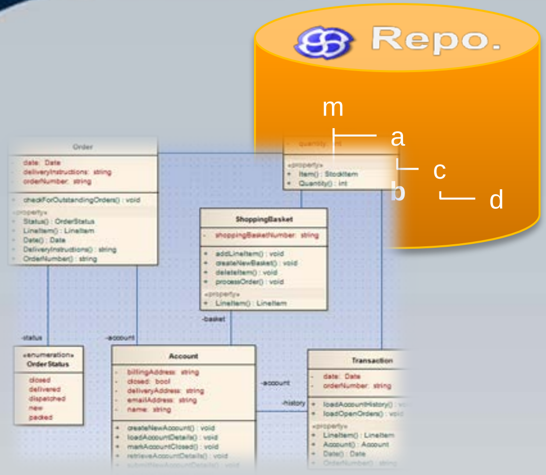
Models Without Modeling

- Code Reverse Engineering

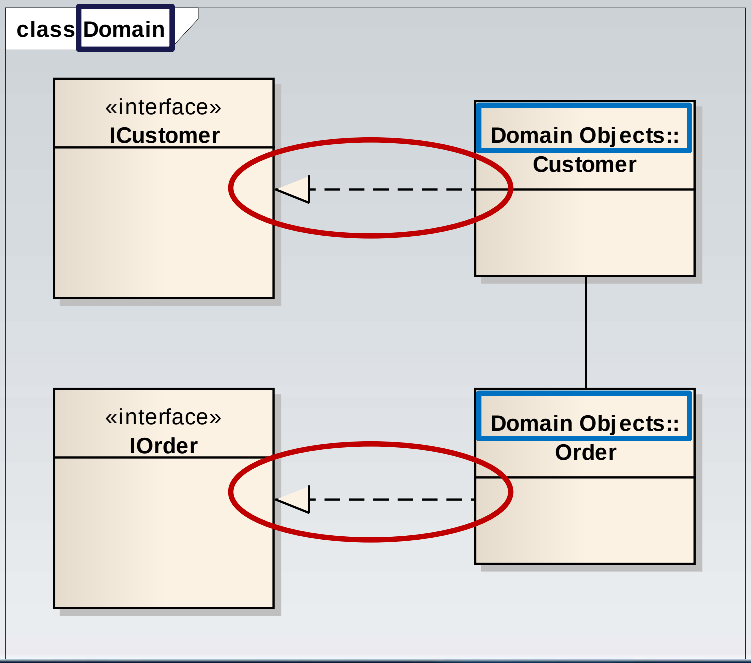


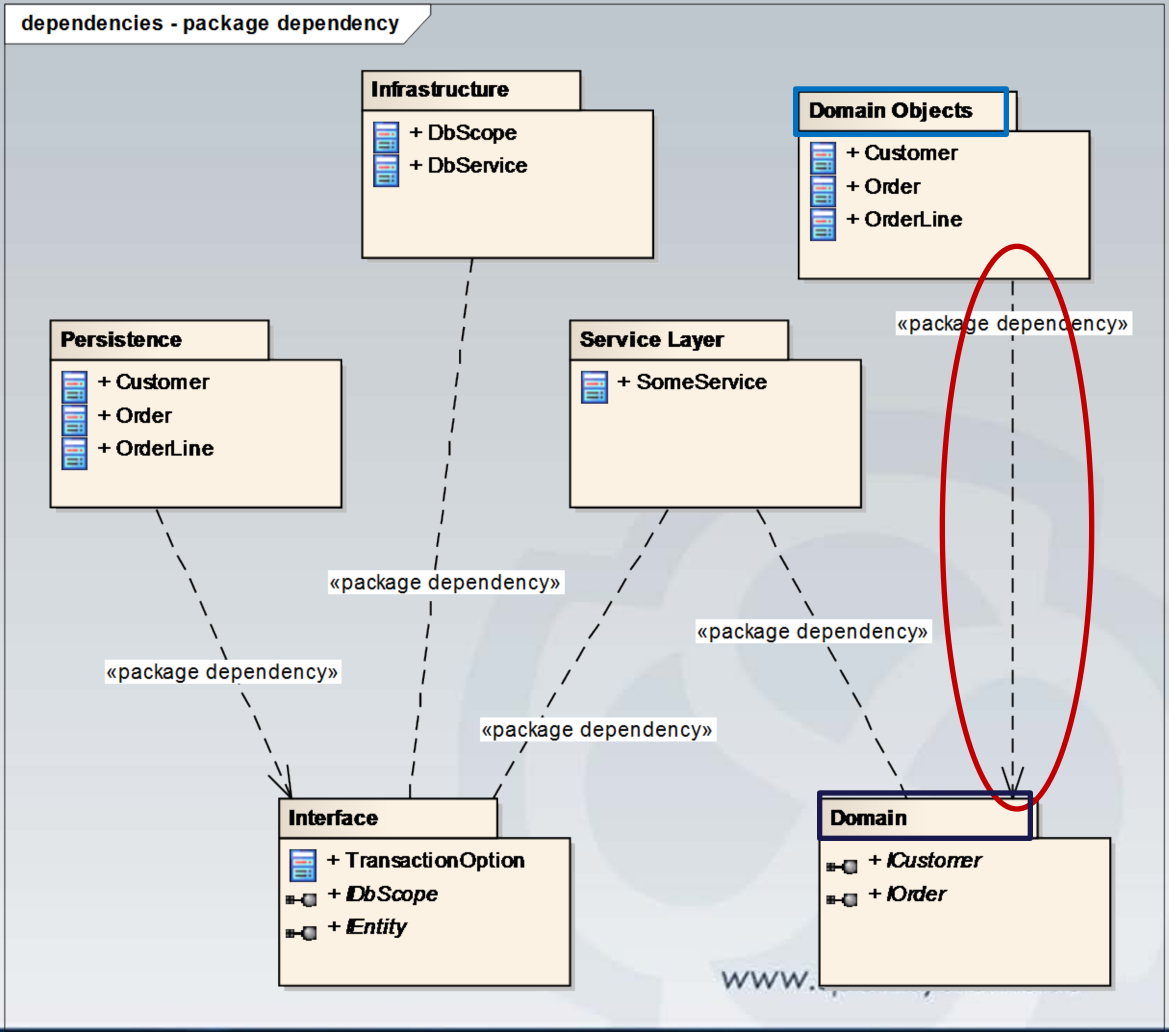
www.sparxsystems.eu

Models Without Modeling



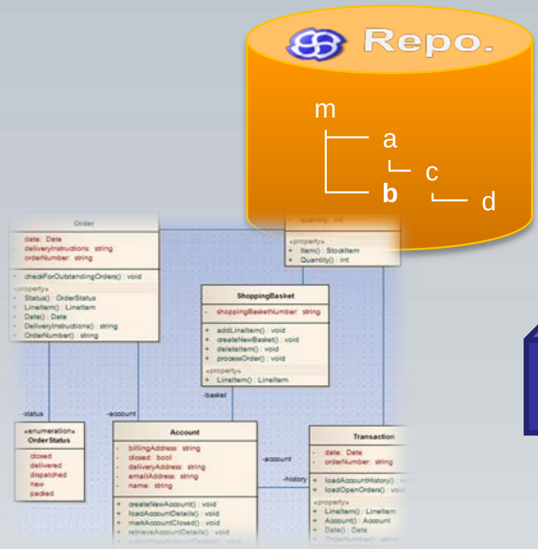
- Package Diagramm erstellen



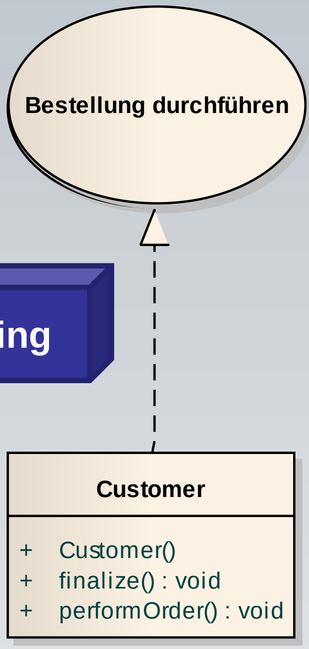


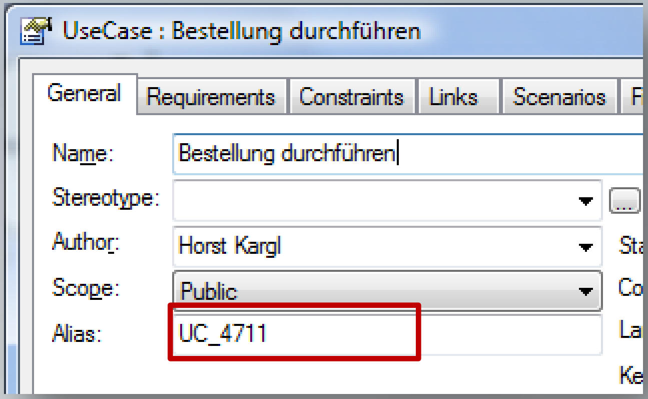
Models Without Modeling

- Requirements mit Implement. verbinden



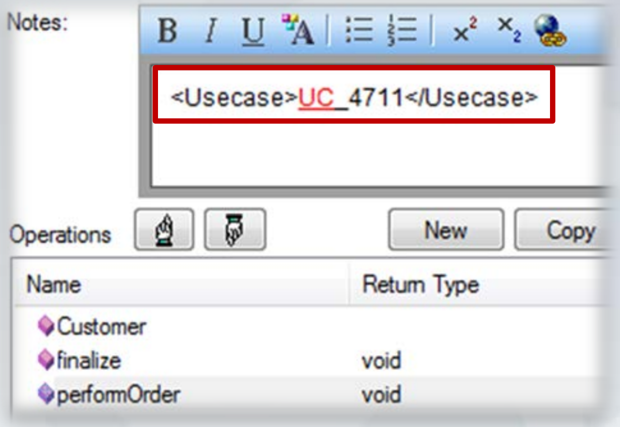
Post Processing





UseCase : Bestellung durchführen

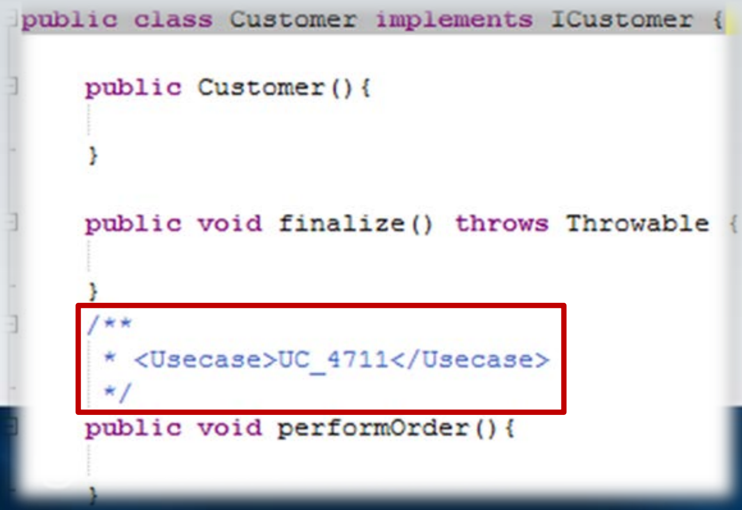
Name:	Bestellung durchführen
Stereotype:	
Author:	Horst Kargl
Scope:	Public
Alias:	UC_4711



Notes:

<Usecase>UC_4711</Usecase>

Name	Return Type
Customer	
finalize	void
performOrder	void

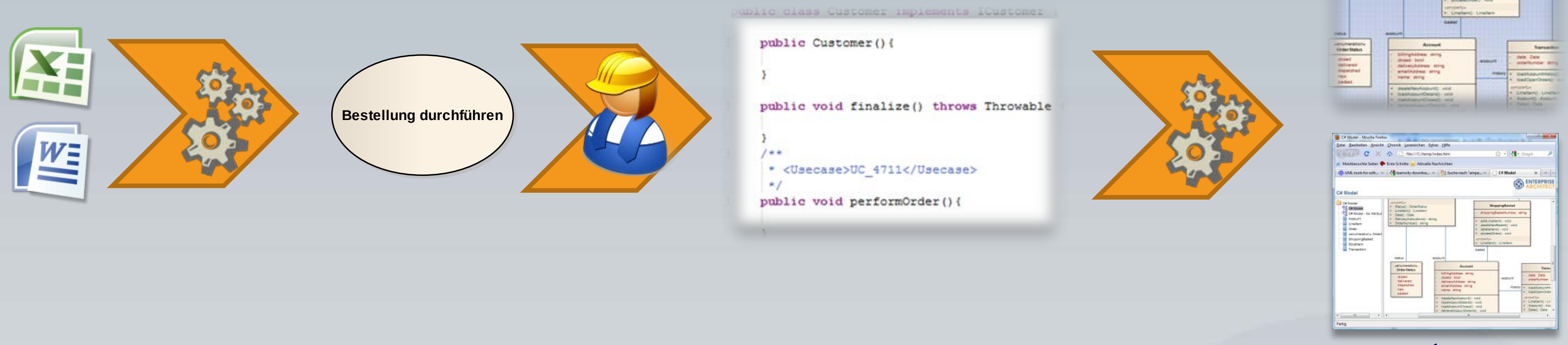


```
public class Customer implements ICustomer {

    public Customer(){

    }

    public void finalize() throws Throwable {

    }
    /**
     * <Usecase>UC_4711</Usecase>
     */
    public void performOrder(){

    }
```

www.sparxsystems.eu

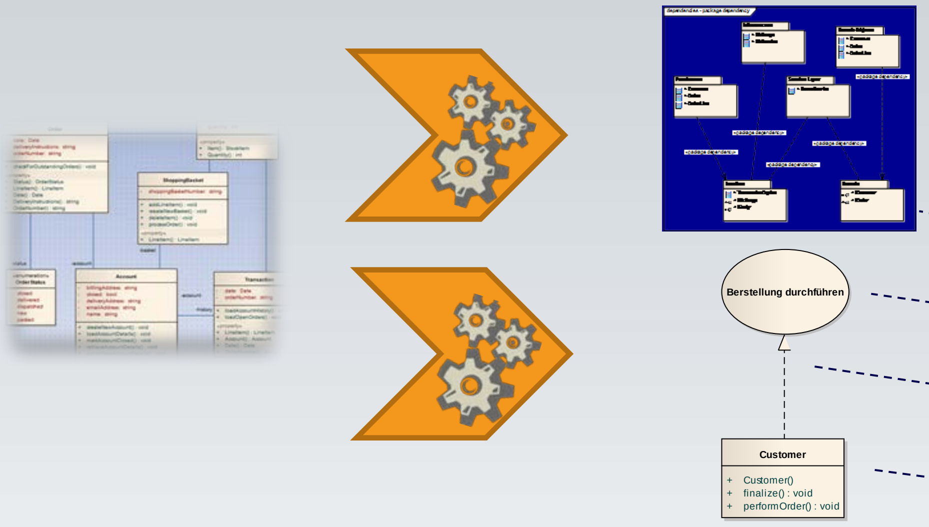
Models Without Modeling

- Zusammenfassung



```
ublic class Customer implements ICustomer {
public Customer(){

}

public void finalize() throws Throwable {

}
/**
 * <Usecase>UC_4711</Usecase>
 */
public void performOrder(){
```

Automatisches Erstellen von:



HTML

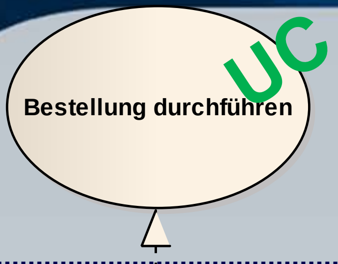
Paket Abhängigkeiten

Use Cases/Requirements

Traces

Klassen Diagramm

Abstraction from Implementation



Analyse + Design

• Analyse + Design Model can be connected with implementation models.

Abstraction of the implementation:

• Better project understanding

• Easy maintenance of Code/Modell

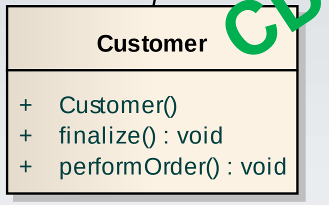
AD

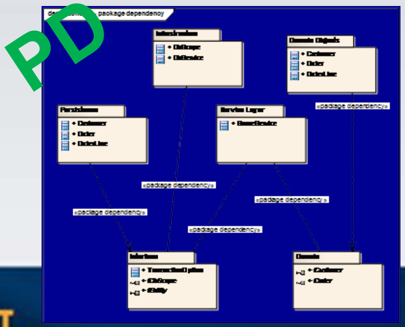
CD

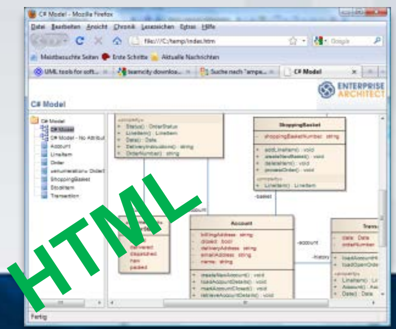
StD SD

CoD

Abstraktion







Implementierungs-modelle