

# UML Profile for DCL

Julia Reznik, Marc Born

GMD Fokus

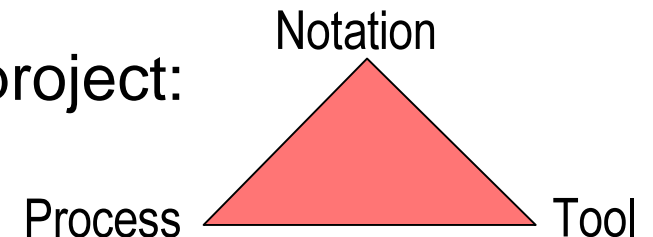
*{reznik,born}@fokus.gmd.de*

## Agenda

- Motivation
- Technology used for graphical support of DCL: UML profile
  - UML semantics
  - Definition of the UML profile
  - Extension mechanisms
- The profile for DCL
  - Stereotypes and notation
  - Automatic generation of XML descriptors with a UML tool
- Conclusion

## Motivation

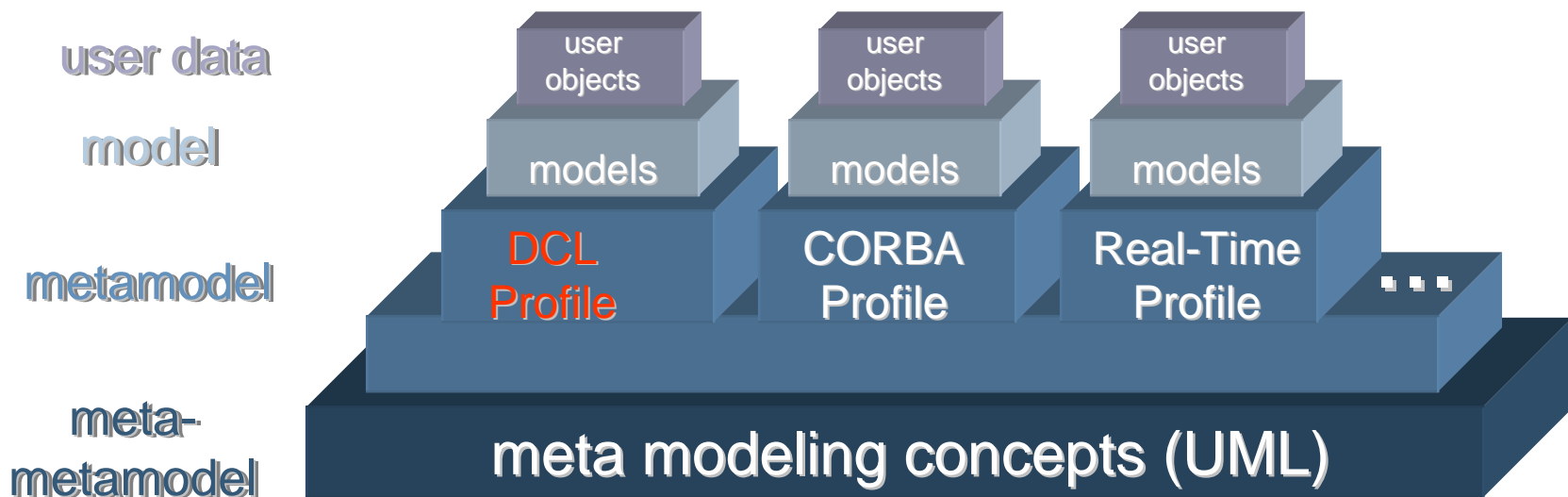
- The triangle for a successful software project:



- Why a graphical support for Deployment and Configuration Language (DCL)?
- Why Unified Modeling Language (UML)?
- Goal
  - Extend design phase to the distributed specification.
  - Connection between software development and deployment process.
  - One model contains all relevant information.

## UML Semantics

- All-purpose language
- Specialization and restrictions for DCL are needed !
- UML profile – domain specific library of UML „macros“
- Four layer meta-modeling architecture:



## UML Profile

- Is a predefined set of stereotypes, tagged values, constraints (extension mechanisms)
- Extension mechanisms collectively specialize and tailor UML for a specific domain or process
- Provides conventions for applying and specializing standard UML to a particular environment or domain
- Does not extend UML by adding any new basic concepts

## UML Extension mechanisms

- Stereotypes:
  - Sub-classification of an existing UML element
  - The new element has its own special properties (expressed as tagged values), semantics and notation
- Tagged Values:
  - New information about model elements and presentation elements (new properties)
- Constraints:
  - Conditions and restrictions, that apply to model elements

## UML Profile for DCL

- Provides conventions for applying and specializing standard UML to the graphical notation for DCL
- Specializes the UML metamodel for DCL
- Bridges the gap between design and deployment phase of distributed applications
- Realisation of this profile with existing UML tools is possible (e.g. Rational Rose)
- Interchange format between tools and programs: **XML** (deployment phase)

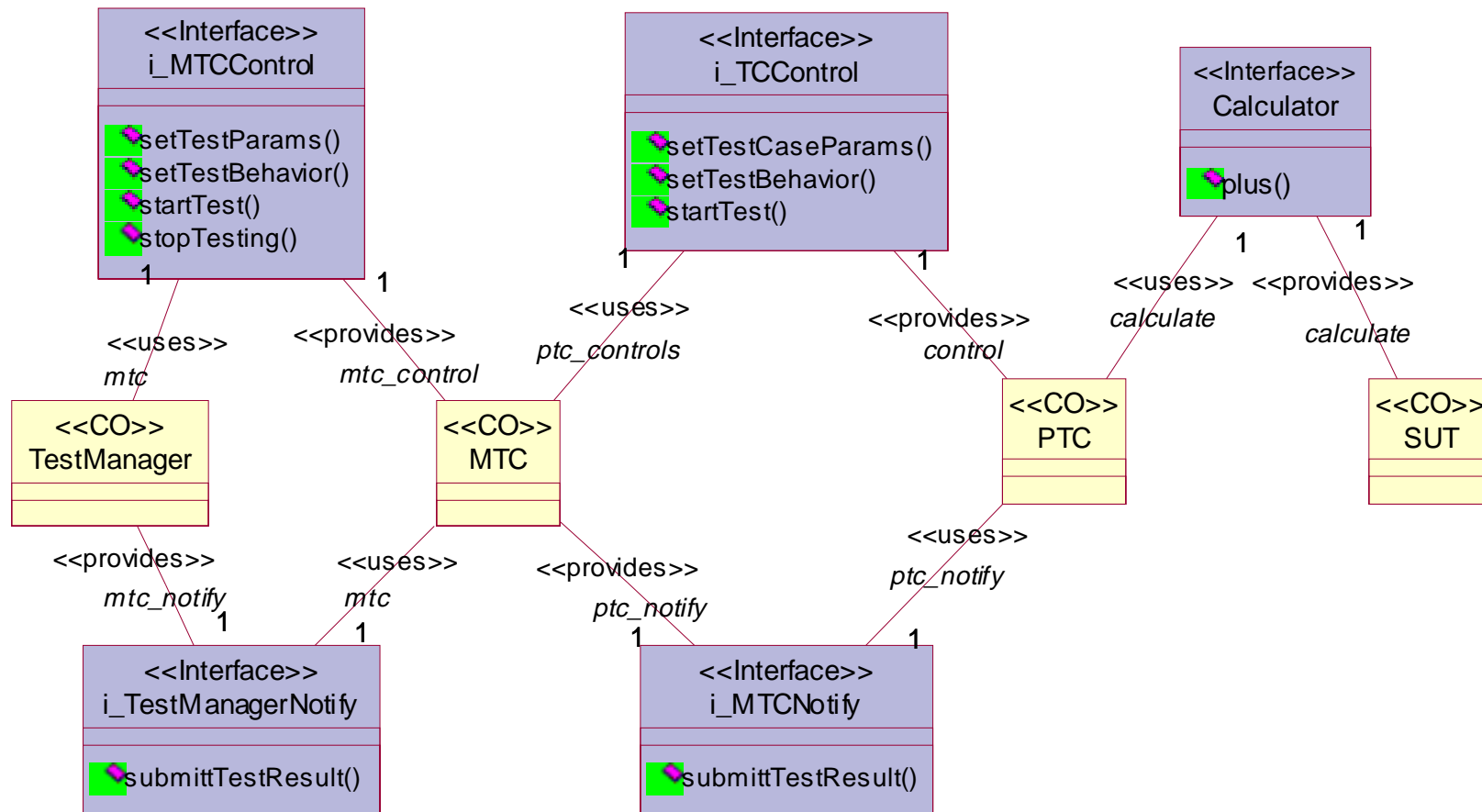
## Diagrams and Stereotypes

- Class diagram : Computational Object Types (CO Types) and their interfaces
- Collaboration diagram: Initial configuration of COs and their factories
- Component diagram: implementation components
- Stereotypes:

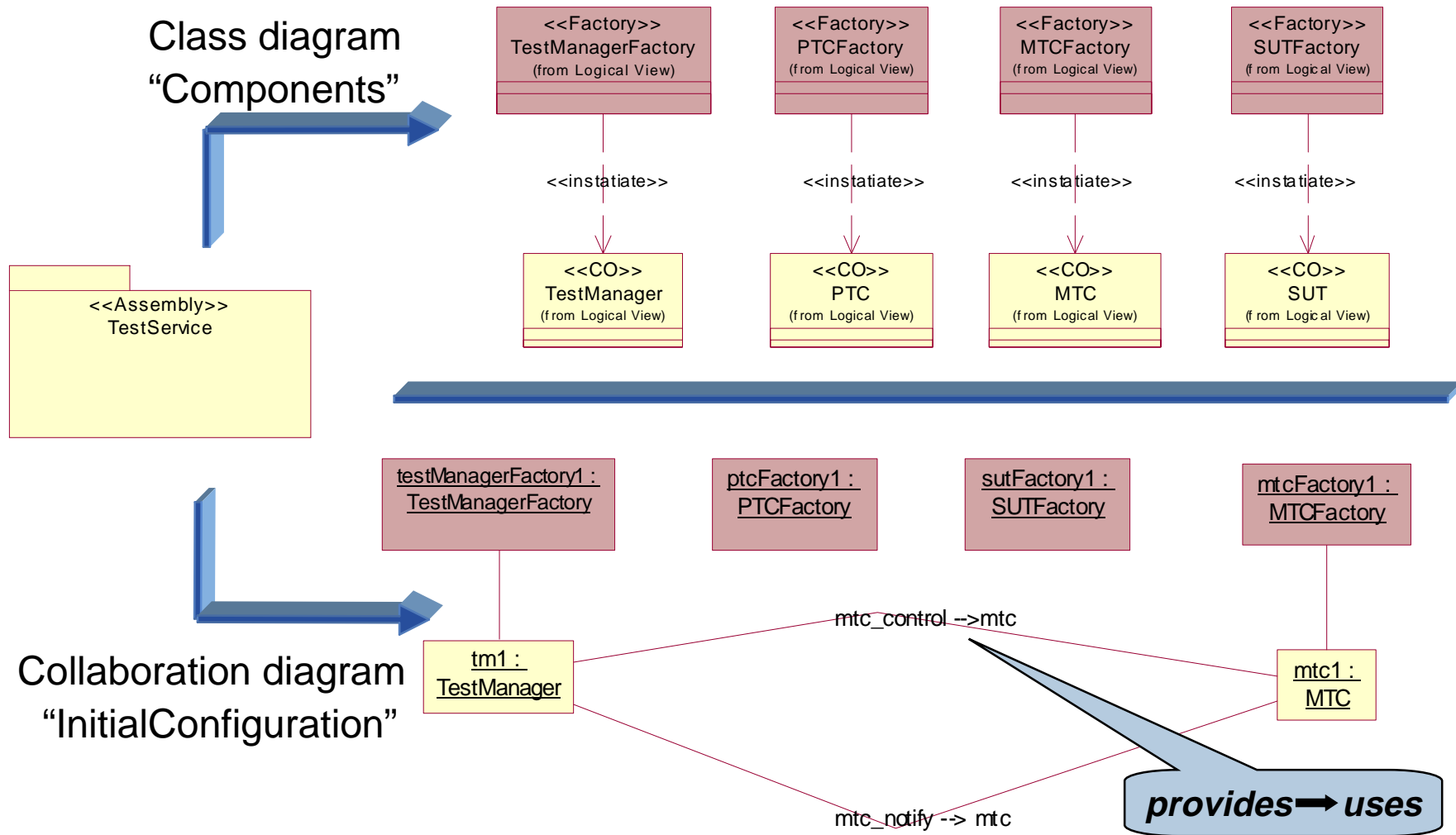
UML-Metamodel element	Stereotype
Class	<<CO>>
Class	<<Factory>>
Class	<<Interface>>
Component	<<implementation>>
Package	<<assembly>>
Association	<<uses>>
Association	<<provides>>
Dependency	<<instantiate>>



# Class diagram: CO Types and interfaces



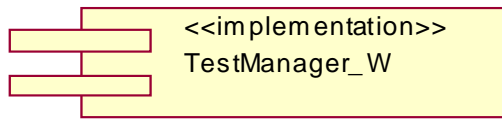
# Initial Configuration



## Tagged Values (1)

- DCL specific keyword pairs: property name and value
- For stereotypes <<CO>> and <<Implementation>> tagged values are defined in the profile
- Source: CCM Descriptors
  - Software Package
  - CORBA Component
  - Component Assembly

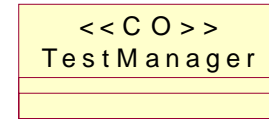
# Tagged Values (2)



Component Specification for TestManager\_W

Set: default

Name	Value
code_type	EXE
main_memory	128MB
compiler_name	Visual C++
compiler_version	2.1.0.0
dependency_action	
programming_language_name	C++
programming_language_version	6.0.0.0
human_language	English
os_name	Windows NT
os_version	4.0.0.0
processor	x86
storage_space_min	40MB
storage_space_max	100MB
property_file_link	ftp://x/y/



Class Specification for TestManager

Set: CO

Name	Value
version	3.0.0.0
component_kind	session
servant_lifetime	component
transaction_use	self-managed
security_rights_family	corba
threading_policy	serialize
configuration_complete_set	true
repository_ins	
repository_objref_string	
repository_link	
repository_type	CORBA Interface Repository
properties_file	

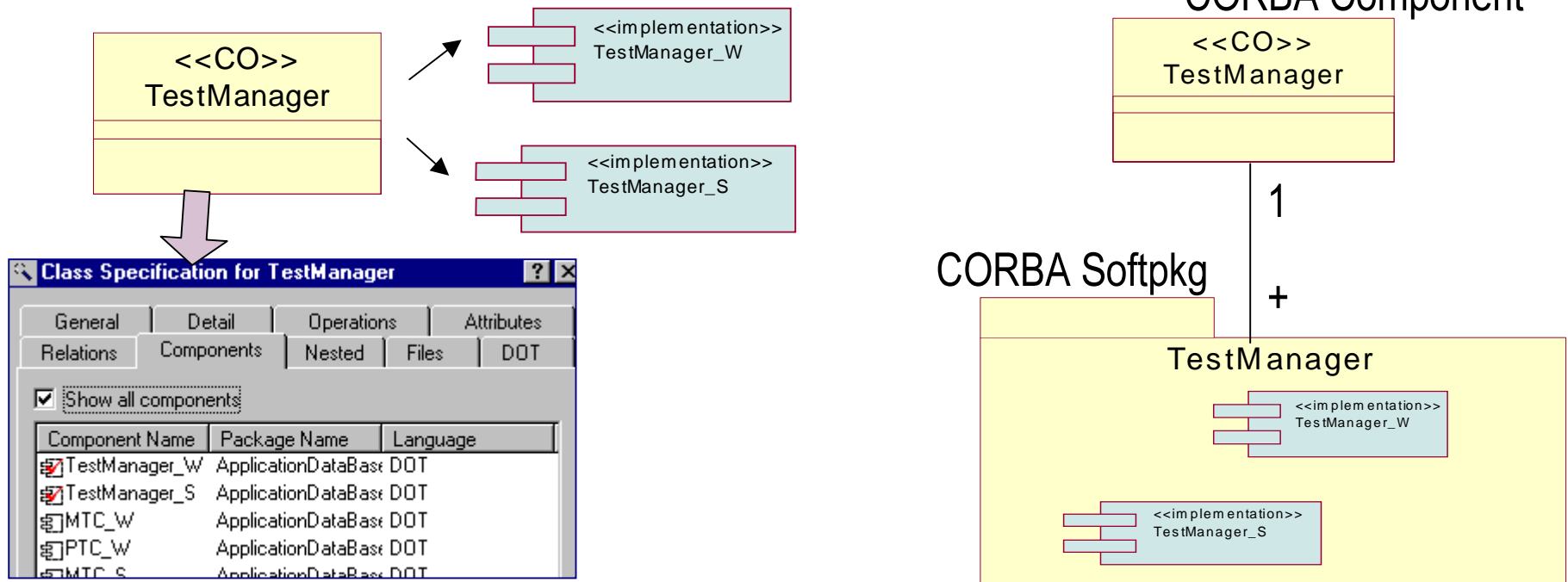
## Constraints

- <<assembly>> must have two diagrams: “Components” and “InitialConfiguration” diagram
- a class diagram of <<assembly>> package contains only classes with <<CO>> and <<Factory>>
- Generalisation: all elements of the same stereotype (e.g. <<CO>>)
- Valid association stereotype combinations:

From: \ To:	<<CO>>	<<Factory>>	<<Interface>>
<<CO>>			<<uses>> <<provides>>
<<Factory>>	<<instantiate>>		
<<Interface>>			

# Realization with UML tool Rational Rose

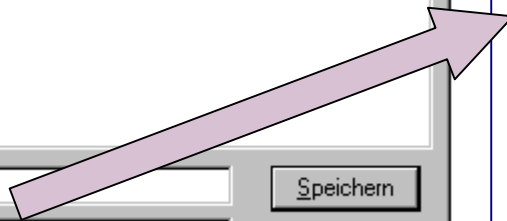
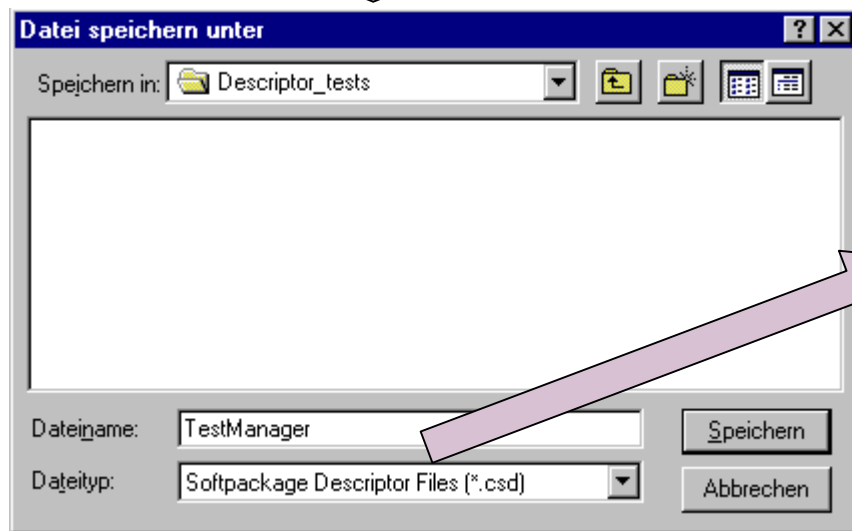
- Rational Rose provides
  - graphical support for UML
  - automatic generation of XML-Descriptors using Rose Extensibility Interface (REI)



# Generation of softpkg descriptor (1)

The image illustrates the process of generating a softpkg descriptor for a component. On the left, a context menu is shown over a component box labeled '<<CO>> TestManager'. The menu item 'Generate Softpkg Descriptor' is highlighted. An arrow points to the 'Softpkg descriptor generation' dialog box. This dialog is divided into two main sections. The top section, 'Autor', contains input fields for 'Name' (PLATIN), 'Company' (GMD FOKUS), 'Webpage' (http://www.fokus.gmd.de/), 'Title' (Test Manager for Windows and Solaris), 'Type' (CORBA Component), and 'License Server' (http://www.fokus.gmd.de/license/). The bottom section, 'Assignment Implementations to Softpkg', features two lists: 'Provided Implementations' (containing TestManager\_X) and 'Selected Implementations' (containing TestManager\_S and TestManager\_W). An 'Add' button is positioned between the lists, and a 'Remove' button is below it. The dialog concludes with 'OK' and 'Abbrechen' buttons at the bottom right.

# Generation of softpkg descriptor (2)



```

<?xml version="1.0" ?>
<!DOCTYPE softpkg SYSTEM "softpkg.dtd">

<softpkg name="TestManager">
  <idl id="IDL:M1/TestManager:1.0"/>
  <author>
    <company>GMD FOKUS</company>
    <name>PLATIN</name>
    <webpage href="http://www.fokus.gmd.de"/>
  </author>
  <license href="http://www.fokus.gmd.de/license"/>
  <title>Test Manager for Windows and Solaris</title>
  <pkgtype>CORBA Component</pkgtype>
  <implementation id="DCE:TestManager_W">
    <description>This is an implementation for Windows operation system </description>
    <descriptor>
      <fileinarchive> TestManager.ccd </fileinarchive>
    </descriptor>
    <mainmemory size="128MB"/>
    <compiler name="Visual C++" version="2.1.0.0"/>
    <programminglanguage name="C++" version="6.0.0.0"/>
    <os name="Windows NT" version="4.0.0.0"/>
    <processor name="x86"/>
    <storage>
      <space min="40MB" max="100MB"/>
    </storage>
    <propertyfile>
      <fileinarchive name="TestManager_W.cpf"/>
      <link href="ftp://xy"/>
    </propertyfile>
    <code type="EXE">
      <fileinarchive name="TestManager_W.EXE"/>
    </code>
  </implementation>
  <implementation id="DCE:TestManager_S">
    <description>This is an implementation for UNIX operation system </description>
    <descriptor>
      <fileinarchive> TestManager.ccd </fileinarchive>
    </descriptor>
    <code type="DLL">
      <fileinarchive name="TestManager_S.DLL"/>
    </code>
  </implementation>
</softpkg>
  
```



# Generation of CORBA component descriptor

**<<CO>>**  
**TestManager**

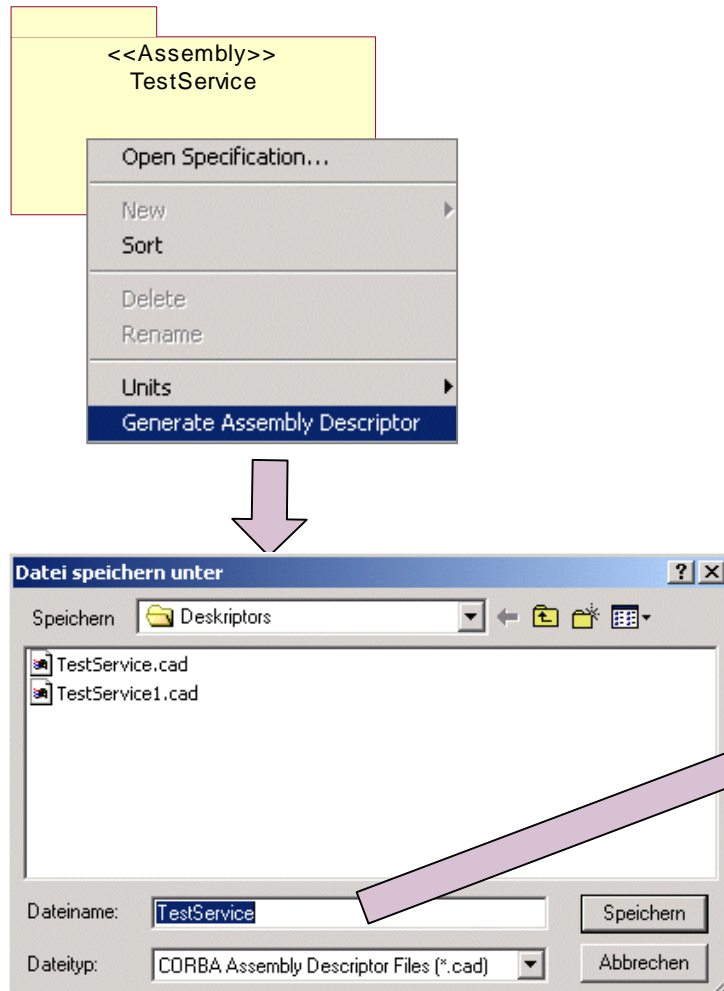
- Open Specification...
- Open Standard Specification...
- Sub Diagrams
- New Attribute
- New Operation
- Select In Browser
- Relocate
- Options
- Format
- Generate Softpkg Descriptor
- Generate Component Descriptor**

```

<?xml version="1.0" ?>
<!DOCTYPE corbacomponent SYSTEM "corbacomponent.dtd">
<corbacomponent>
  <componentrepid repid="IDL:TestManager:1.0">
    <corbaversion>3.0</corbaversion>
    <componentkind>
      <session>
        <servant lifetime="component"/>
      </session>
    </componentkind>
    <transaction use="self-managed"/>
    <security rightsfamily="corba"/>
    <threading policy="serialize"/>
    <repository type="CORBA Interface Repository"/>
    <componentfeatures name="TestManager" repid="IDL:TestManager">
      <ports>
        <provides>
          providesname="mtc_notify"
          repid="IDL:i_TestManagerNotify:1.0">
            <operationpolicies>
              <operation name="submitTestResult">
                <transaction use="required"/>
                <requiredrights>
                  <right name="submit"/>
                </requiredrights>
              </operation>
            </operationpolicies>
          </provides>
          <uses>
            usesname="mtc"
            repid="IDL:i_MTCControl:1.0"/>
          </uses>
        </ports>
      </componentfeatures>
      <interface name="i_TestManagerNotify" repid="IDL:i_TestManagerNotify:1.0">
        <operationpolicies>
          <operation name="submitTestResult">
            <transaction use="required"/>
            <requiredrights>
              <right name="submit"/>
            </requiredrights>
          </operation>
        </operationpolicies>
      </interface>
    </componentfeatures>
  </component>
</corbacomponent>

```

# Generation of CORBA assembly descriptor



```

<!DOCTYPE componentassembly SYSTEM "componentassembly.dtd">
<componentassembly id="TestService">
  <description>Example TestManager</description>
  <componentfiles>
    <componentfile id="TestManager">
      <fileinarchive name="TestManager.ccd" />
    </componentfile>
    <componentfile id="PTC">
      <fileinarchive name="PTC.ccd" />
    </componentfile>
    <componentfile id="MTC">
      <fileinarchive name="MTC.ccd" />
    </componentfile>
    <componentfile id="SUT">
      <fileinarchive name="SUT.ccd" />
    </componentfile>
  </componentfiles>
  <partitioning>
    <homeplacement id="testManagerFactory1" type="TestManagerFactory">
      <componentfileref idref="TestManager" />
      <componentinstantiation id="tm1" />
      <componentinstantiation id="tm2" />
    </homeplacement>
    <homeplacement id="ptcFactory1" type="PTCFactory">
      <componentfileref idref="PTC" />
    </homeplacement>
    <homeplacement id="sutFactory1" type="SUTFactory">
      <componentfileref idref="SUT" />
    </homeplacement>
    <homeplacement id="mtcFactory1" type="MTCFactory">
      <componentfileref idref="MTC" />
      <componentinstantiation id="mtc1" />
    </homeplacement>
  </partitioning>
  <connections>
    <connectinterface id="i_TestManagerNotify">
      <usesport>
        <usesidentifier>mtc</usesidentifier>
        <componentinstantiationref idref="mtc1" />
      </usesport>
      <providesport>
        <providesidentifier>mtc_notify</providesidentifier>
        <componentinstantiationref idref="tm1" />
      </providesport>
    </connectinterface>
    <connectinterface id="i_MTCControl">
      <usesport>
        <usesidentifier>mtc</usesidentifier>
        <componentinstantiationref idref="tm1" />
      </usesport>
      <providesport>
        <providesidentifier>mtc_control</providesidentifier>
        <componentinstantiationref idref="mtc1" />
      </providesport>
    </connectinterface>
  </connections>
</componentassembly>
  
```

## Conclusion

- Graphical support for DCL
  - Specification of all CO Types, their interfaces and properties
  - Specification of Initial Configuration
  - Automatic generation of XML-Descriptors for the Assignment Tool
- In process:
  - Constraints for Run-time Configuration
  - Design of run-time configuration for a hardware system  
(which component on which node?)
- Demo follows ...

**Thank you!**