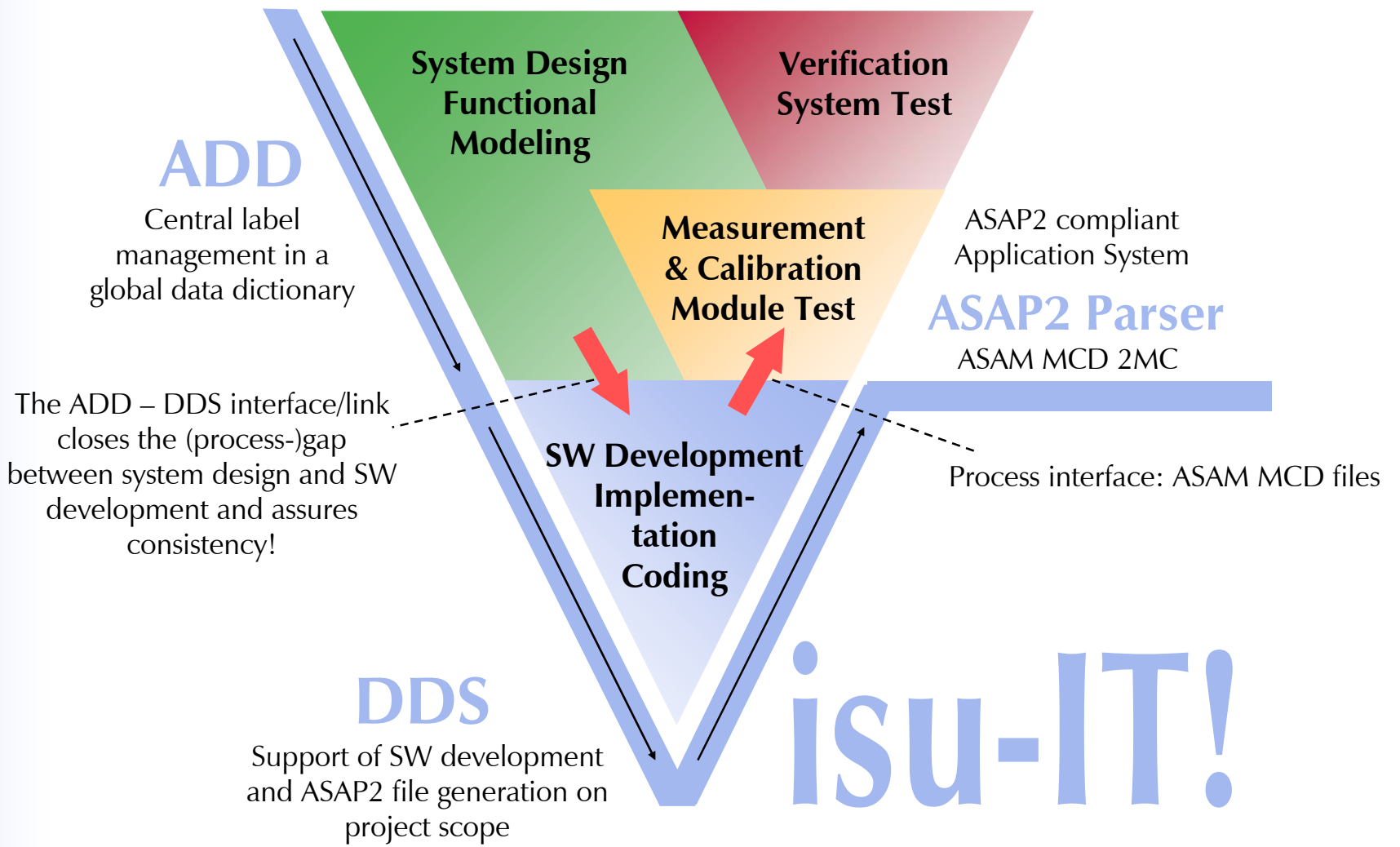
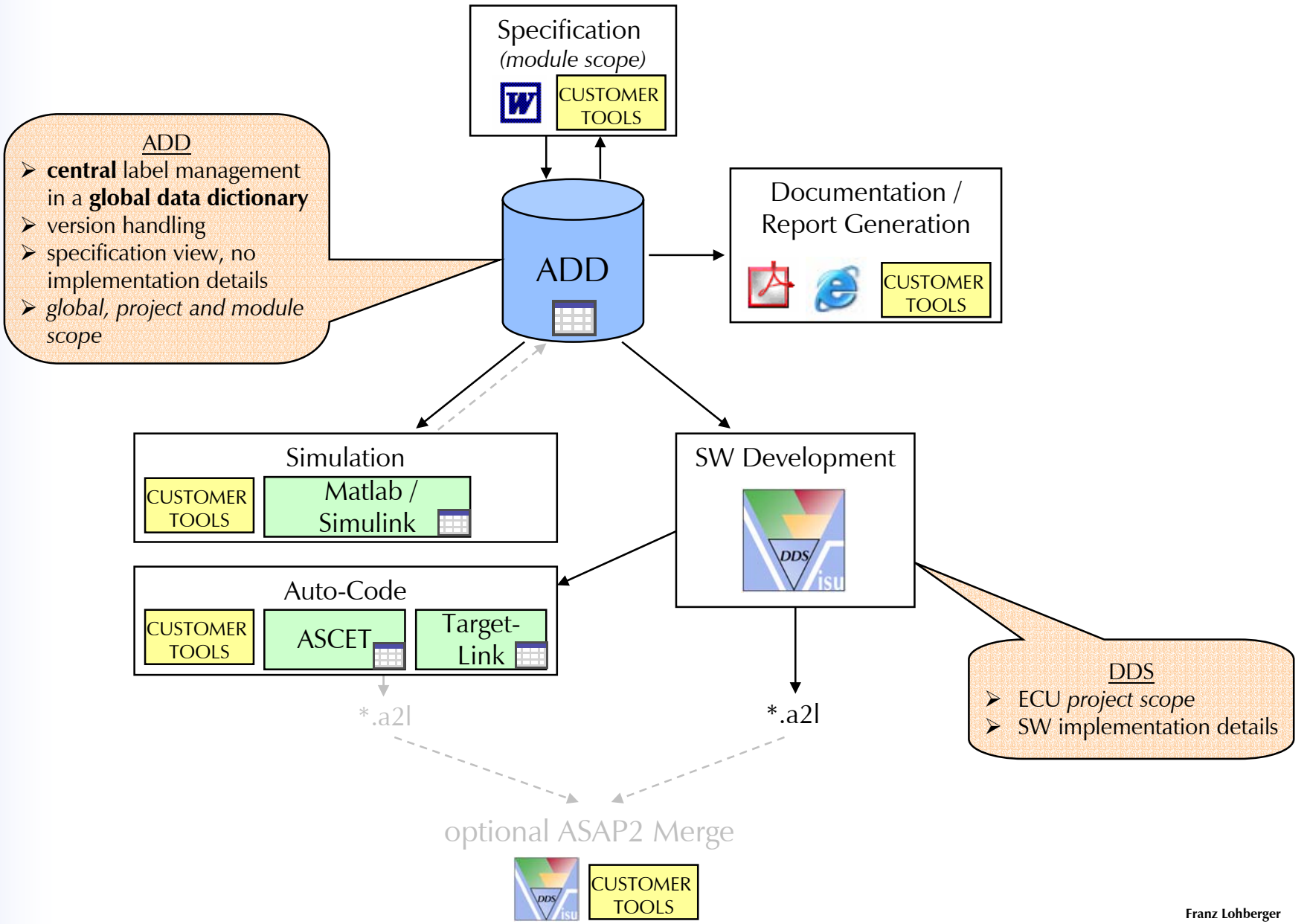
Visu-IT! - Tools in ECU Development

Development Process



Visu-IT! – ECU Development Process

Process flow

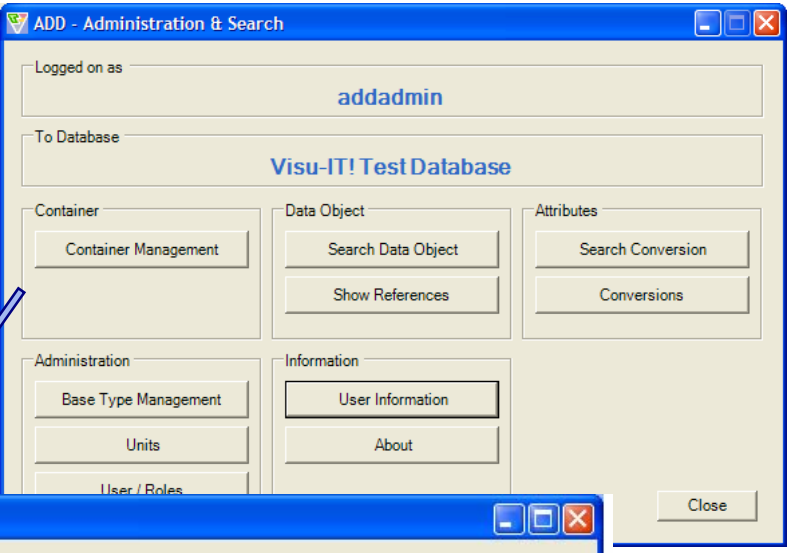


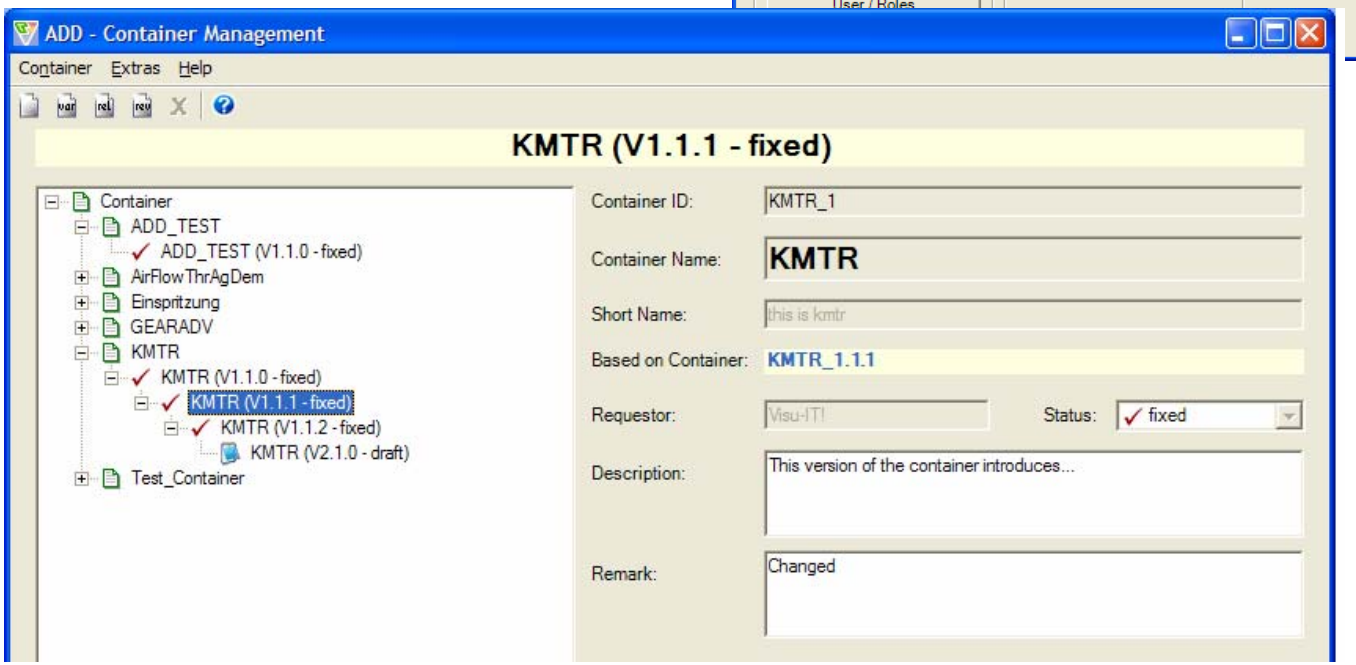
ADD – Getting Started

Starting ADD

1. Start ADD

2. Open Container-Management





ADD – Getting Started

Container Management

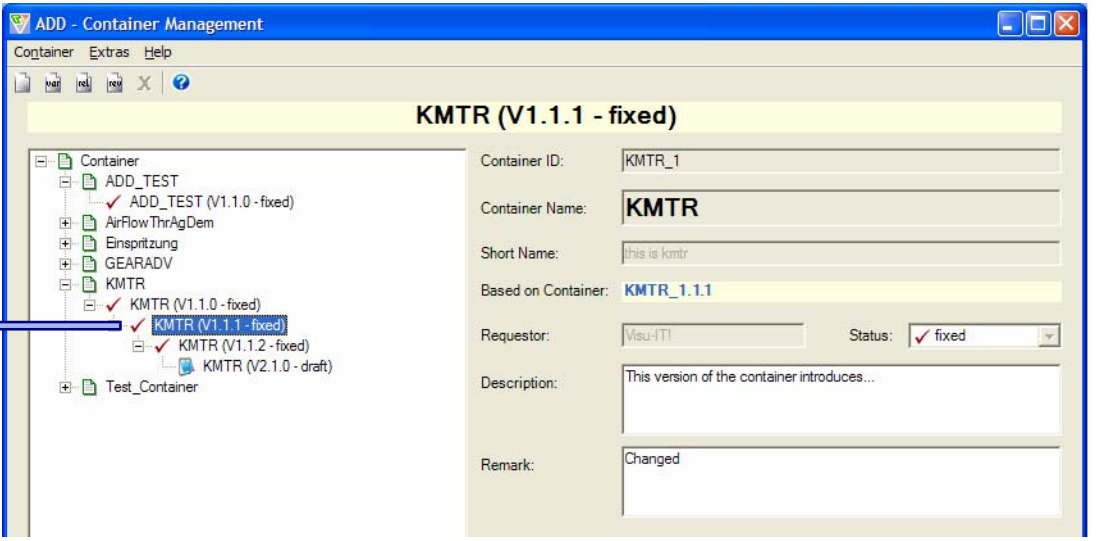
Container can be used to group a set of (calibration-)data.

In ADD a **SW specification/ SW module** can be modeled by using a container.

ADD allows to:

- create, modify, maintain and delete containers
- create new versions and revisions of existing containers (version management)
- define different roles and lifecycles for the container management

When opening a container the following form appears (see next slide)



ADD – Getting Started

Data Objects (1)

The **Interface** of the container is controlled via the tree view on the left side.

The (interface-)data can be “INPUT”, “OUPTUT” and “LOCAL” (optional).

ADD - Dataobject Definition

Data Object Container Print Tools Check Help

Container

KMTR (V1.1.1 - fixed)

- KMTR
  - OUTPUT
    - axis_array (1.0, fixed)
    - testCDefine (1.0, fixed)
    - map_array (1.0, fixed)
    - aNewOnline2 (1.1, fixed)
    - drehzahl (1.3, draft)
    - o1 (1.2, fixed)
    - online2 (1.0, fixed)
    - p1 (1.0, fixed)
  - INPUT
    - cDefine2 (1.0, fixed)
    - myOnline2 (1.1, fixed)
    - newInputOnline (1.0, fixed)
    - o2 (1.0, fixed)

map_array (1.0, fixed)

Data Definition | Data Object Information | Languages

Name: map_array

Display name: this is a map array

Type: Map

Status: fixed

Container specific Information

Classification: OUTPUT

Limits: -2,1474836E+09...2,1474836E+09

Array Size

Dim	Array Size
X	8
Y	-
Z	-

SIM Data Type

Base Type: SLONG Phys. Unit: [-]

Conv. Type: RationalFunction

Conv.: RATIONALFUNCTION_[0;1;0;0;0;1]

Formatstring: %6.3f

SW Data Type

Base Type: SLONG Phys. Unit: [-]

Conv. Type: RationalFunction

Conv.: RATIONALFUNCTION_[0;1;0;0;0;1]

Formatstring: No Formatstring

Axis

Dim	Axis	Version	Status	Axis size	SW Data Type
X	axis_array	V1.0	fixed	7	RATIONALFUNCTION_[0;1;0;0;...
Y	-	-	-	-	-
Z	-	-	-	-	-

ADD – Getting Started

Data Objects (2)

Data Objects (Calibration data):

- can be of the type: MEASUREMENT, AXIS, CHARACTERISTIC and System Constant (defines)
- can have both “SIM”- (simulation, e.g. in Matlab) und “SW”- (software, e.g. in C source code) attributes
- are controlled via a configurable version management (versions, revisions)
- are managed via a configurable lifecycle management (‘draft’, ‘simulation_fixed’, ‘fixed’ and ‘obsolete’)
- follow a configurable user and rights management

To add further data objects to a container you can either

1) search for existing data objects
   (see next slide)
2) add/create new data objects (or a new version/revision of a data object)
   (see slide after next)

ADD – Getting Started

Data Objects – Search for data objects

ADD - Dataobject Definition

Data Object Container Print Tools Check Help

KMTR (V2.1.1 draft)

ADD - Search for Data Objects

Data Object Name (Use Wildcards * And ? OR a Comma Separated List Of Keywords)

agmAck

Automatically With Like
Match Case
Key Word Search

Display name (Use Wildcards * And ?)

Automatically With Like
Match Case

Type

cDefine Parameter Online Axis Map

Data Type

Name	Phys. Unit	Base Type	Formats...

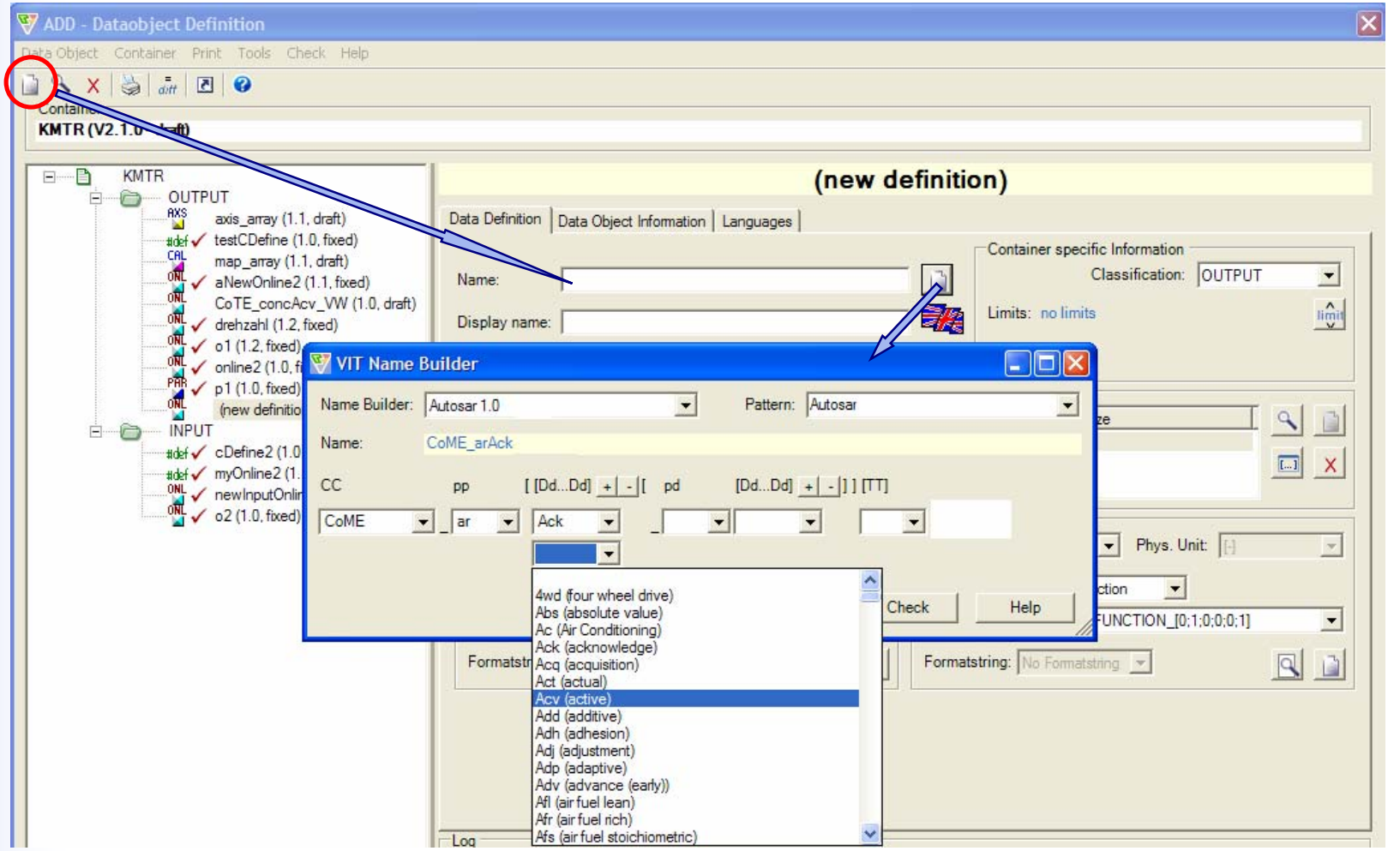
Start Search

Name	Version	State	SIM Data Type	Phys. Unit	Base Type	SV
CoVeh_agmAck_aAbs_VW	1.0	draft	RATIONALFUNCTION_[0;1...	[-]	FLOAT64...	RA
CoVeh_agmAck_VW	1.0	draft	RATIONALFUNCTION_[0;1...	[-]	FLOAT64...	
CoVeh_agmAckAcq_VW	1.0	draft	RATIONALFUNCTION_[0;1...	[-]	FLOAT64...	

Print Get Selected Close Help

ADD – Getting Started

Data Objects – Create new data objects



The optional “Name Builder/Checker” component helps applying naming conventions. In the example above, the component supports the **AUTOSAR** naming convention. The component is generic and freely configurable, thus it is also possible to support customer specific naming conventions.

ADD – Getting Started

Use case: Create a new function/module (1)

Steps in ADD:

1. Create a new container/module/specification
   Status: 'draft'

2. Define the interface of the container
   - define the **INPUT**s
     -> reference existing data objects
     e.g. select the correct ENGINE_SPEED from the global database for your function (e.g. 32bit, etc.)
   - specify the **OUTPUT**s and **LOCAL**s
     -> use existing data objects (from previous or related functions) and/or add new data objects
     The status of the new data objects is: 'draft'

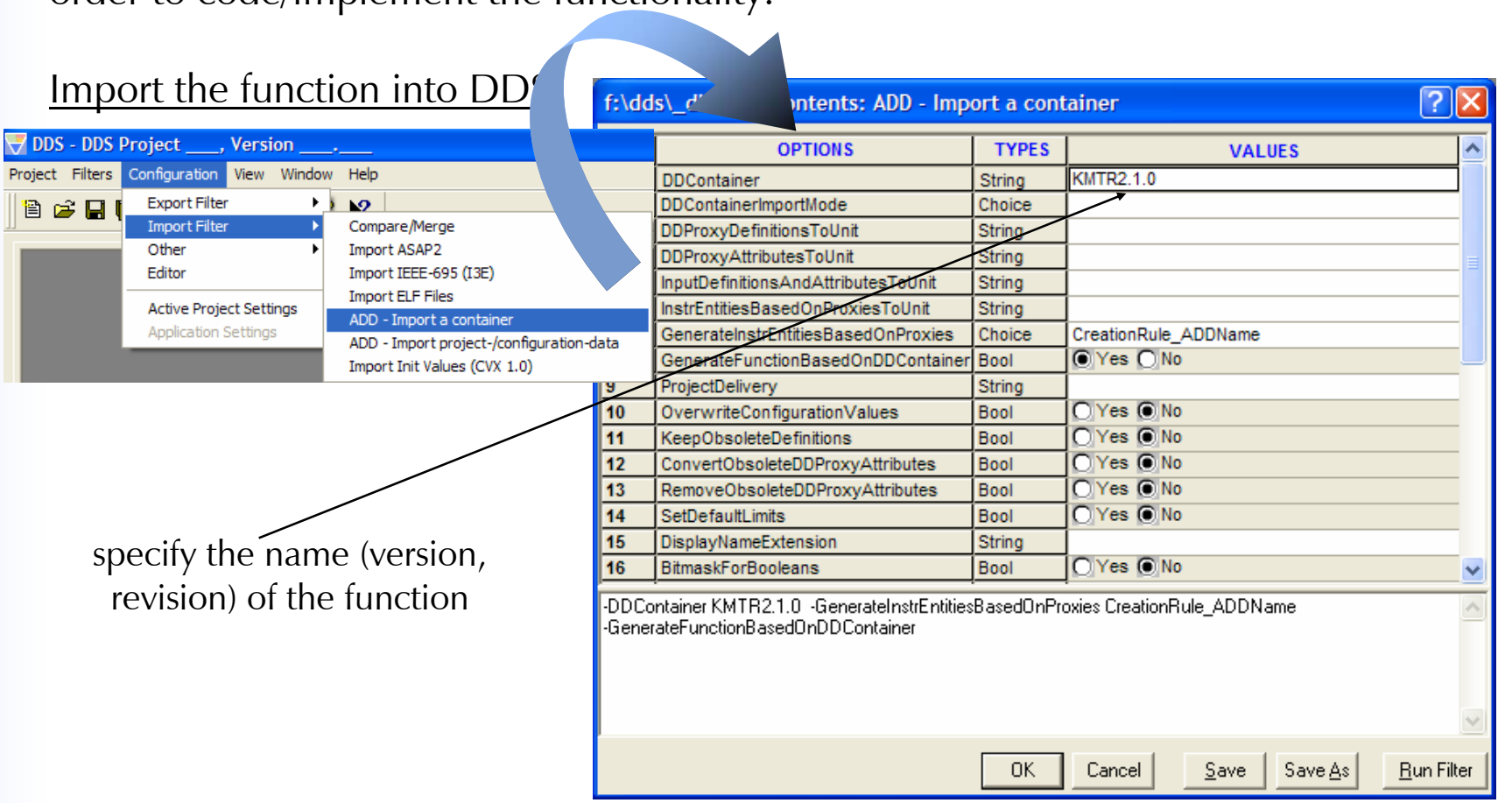
3. "Simulate" your function
   Before you can simulate your function, you have to set the simulation-related attributes (SIM-attributes). After that, you can set the status of the data objects to 'simulation fixed' and export the container and the data objects to Matlab in order to simulate the functionality.
   -> see http://www.visu-it.de/add, section "Interfaces & Links", section "Matlab"

ADD – Getting Started

Use case: Create a new function/module (2)

4. "Implement/code" your function
   Before you can create real C code, you have to set the software-related attributes (SW-attributes). After that, you can set the status of the data objects to 'fixed' and export the container and the data objects to DDS (and Ascet and TargetLink) in order to code/implement the functionality.

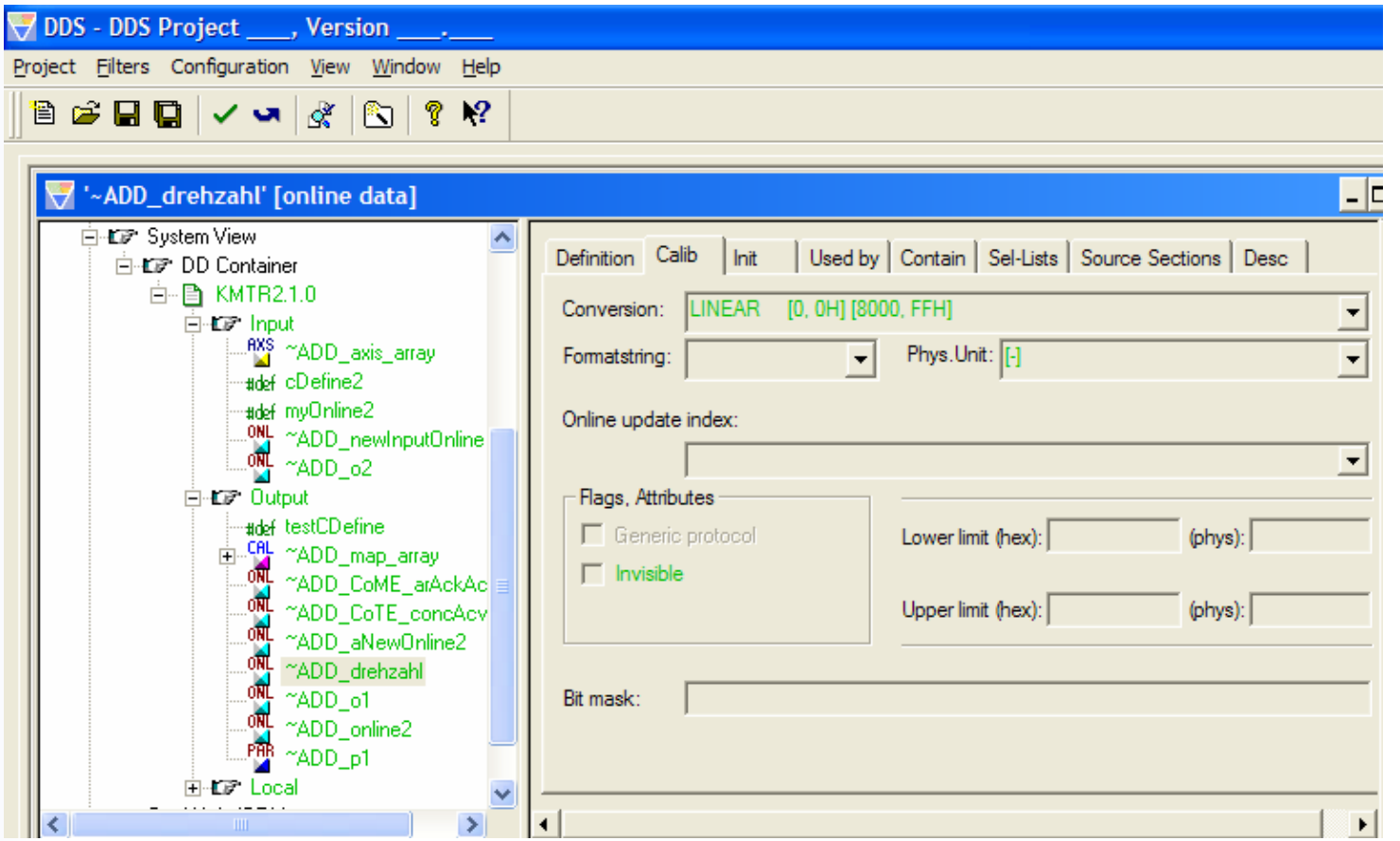
   Import the function into DD



specify the name (version, revision) of the function

ADD – Getting Started

Use case: Create a new function/module (3)



Note: In DDS, all data objects and attributes which are defined and specified in ADD are shown in green color and are readOnly! ADD is the 'master' for these attributes.

5. Container lifecycle
   Similar to data objects, the container is/can be also set to 'simulation fixed' when the simulation has been successfully done. The container can be set to 'fixed' when all data objects of the container are also set to 'fixed'.

ADD – Getting Started

Use case: Create a new version/revision of a function

Steps in ADD:

1. Create a new version/revision of an existing container
   Status: 'draft'

2. Define the (new) interface of the container, add new data objects, remove obsolete data objects, create new versions/revisions of existing data objects etc.
   Note: When only a new revision of the container is created, some modifications are not allowed.

3. "Simulate" your function
   …identical to the use case "Create a new function/module"

4. "Implement/code" your function
   …identical to the use case "Create a new function/module"
   Note that DDS provides a smart "Upgrade" mechanism!

5. Container lifecycle
   …identical to the use case "Create a new function/module"