

BC621

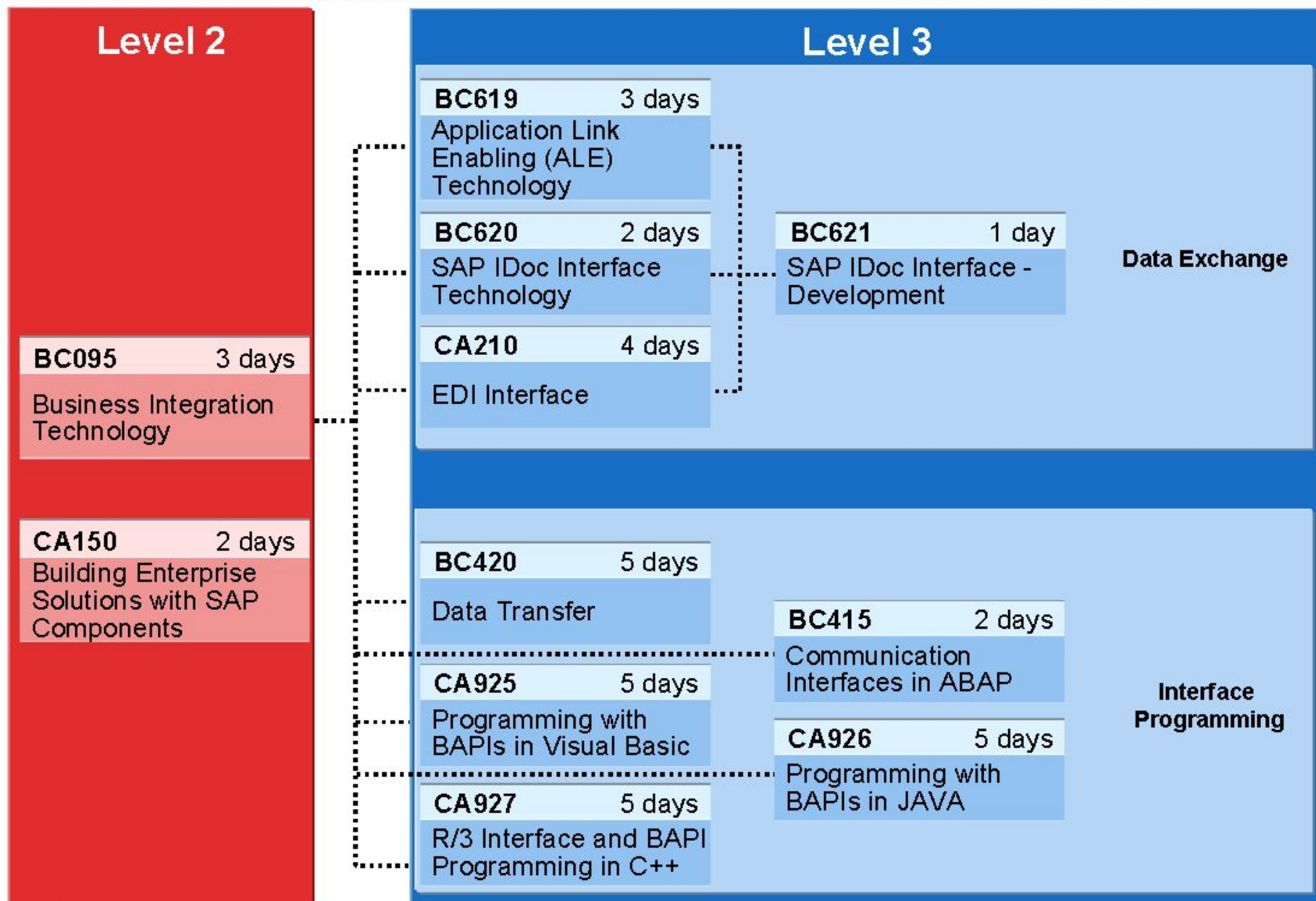


**SAP IDoc Interface
(Development)**

Copyright 2000 SAP AG. All rights reserved.

Neither this training manual nor any part thereof may be copied or reproduced in any form or by any means, or translated into another language, without the prior consent of SAP AG. The information contained in this document is subject to change and supplement without prior notice.

All rights reserved.

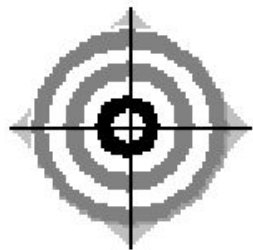


- **Recommended: Basis BC 400 - ABAP Workbench Basics**
- **Required: Basis BC620 - IDoc Interface (Standard)**

- **ABAP Developers**
- **Consultants**

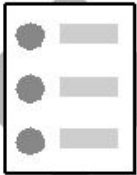


- **Course Goals**
- **Course Objective(s)**
- **Course Content**
- **Course Overview Diagram**
- **Main Business Scenario**



At the conclusion of this course, you will be able to:

- **Extend IDoc types**
- **Define new IDoc types**



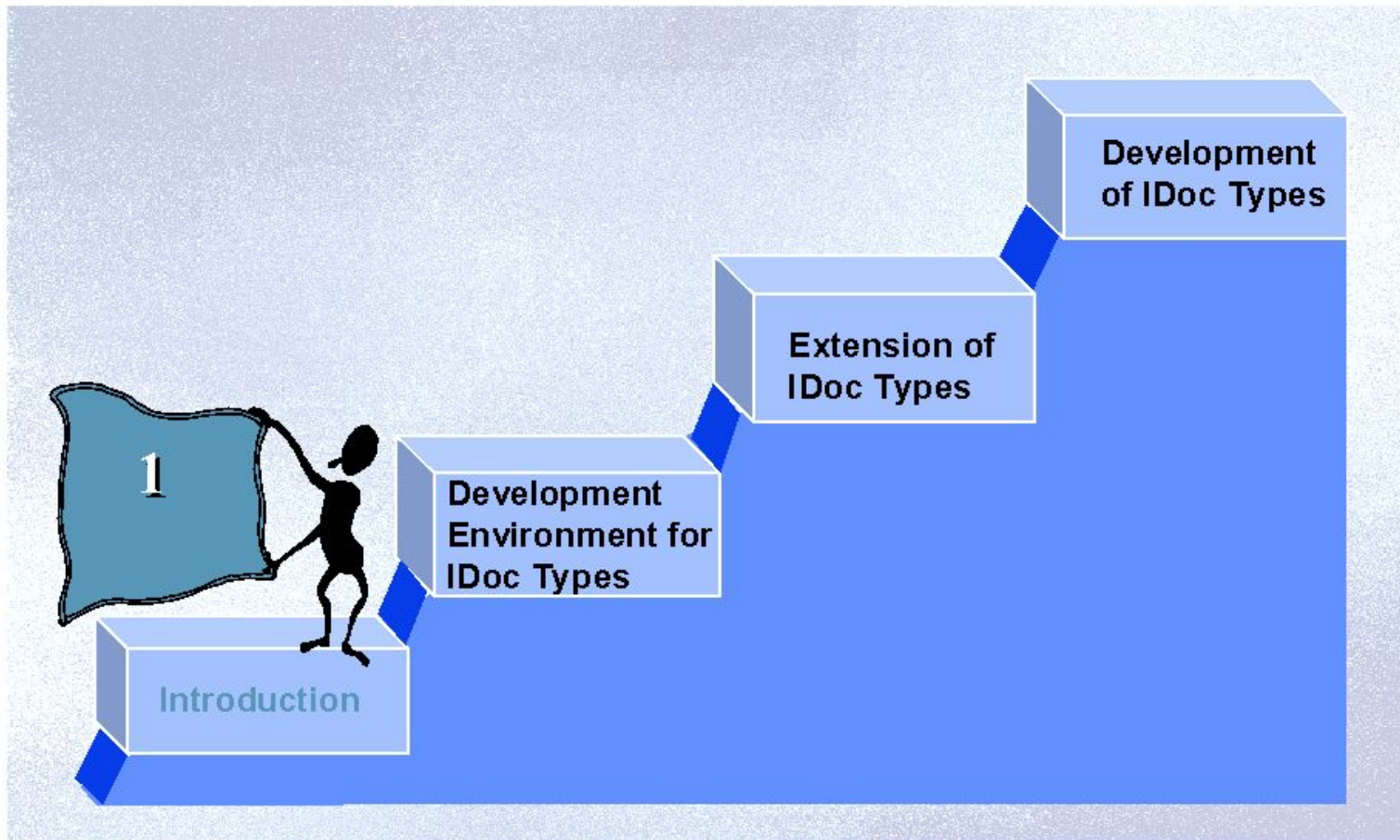
At the conclusion of this course, you will be able to:

- **Build and extend the data structure of IDoc types**
- **Use the IDoc type editor and segment editor**
- **Use customer exits to process IDoc types**

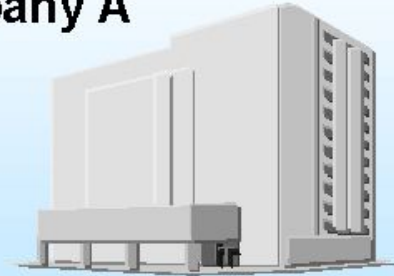
Preface

- | | |
|---------------|--|
| Unit 1 | Introduction |
| Unit 2 | Development Environment for IDoc Types |
| Unit 3 | Extension of IDoc Types |
| Unit 4 | Development of IDoc Types |
-

Appendix



Enterprise trading
company A

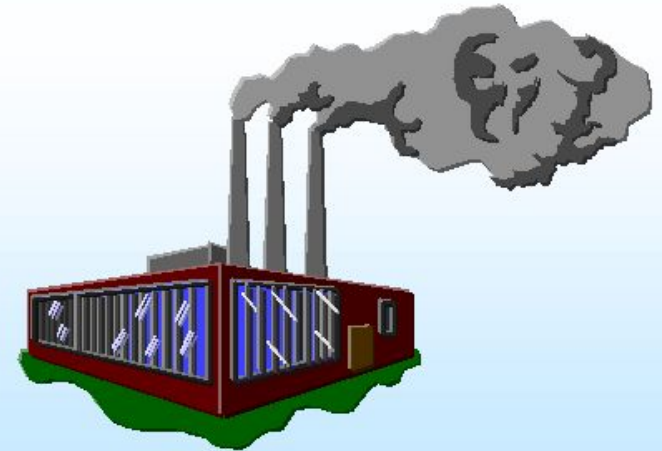


SAP R/3 System



EDI Subsystem

Vendor B



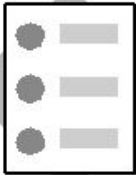
SAP R/3 System



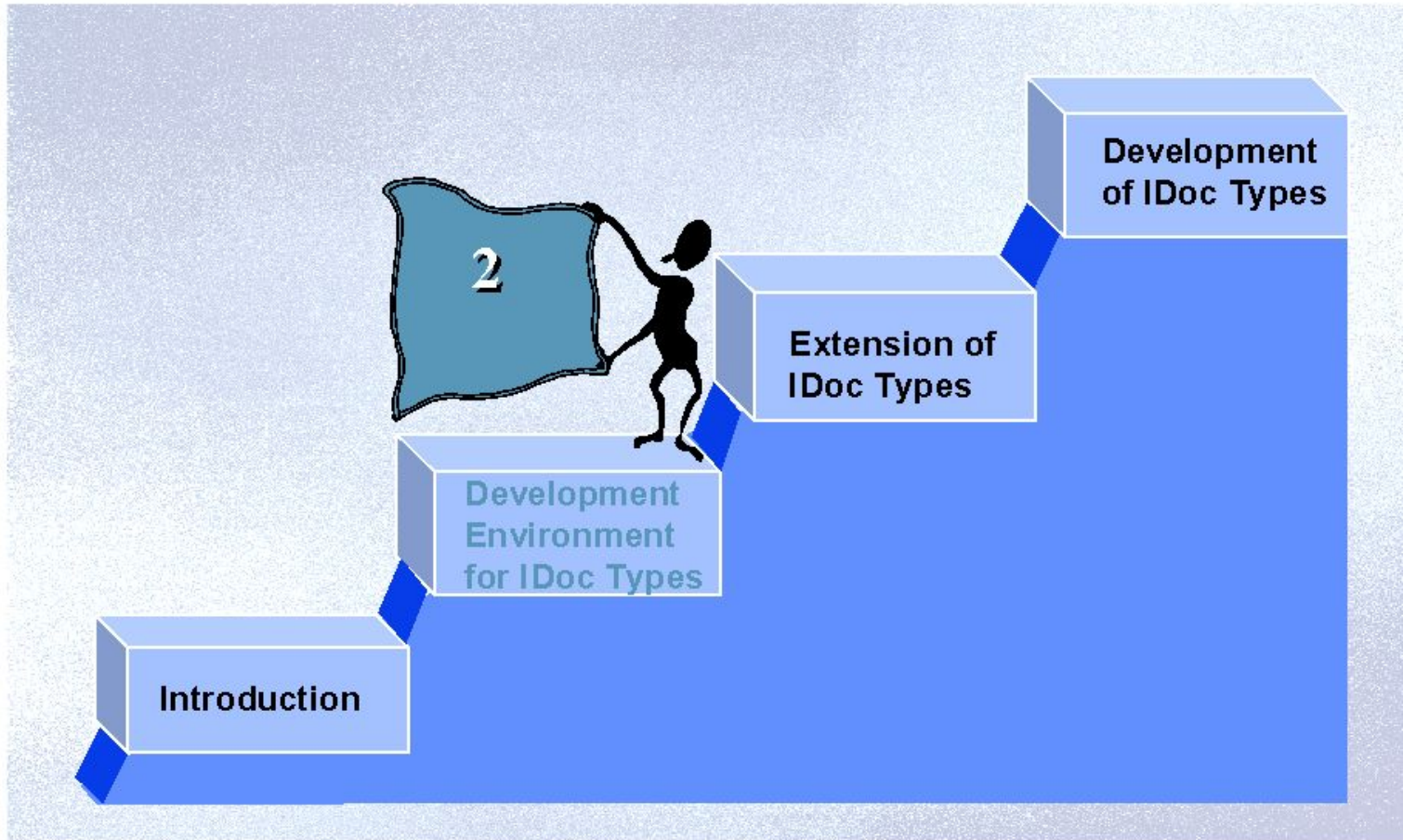
EDI Subsystem

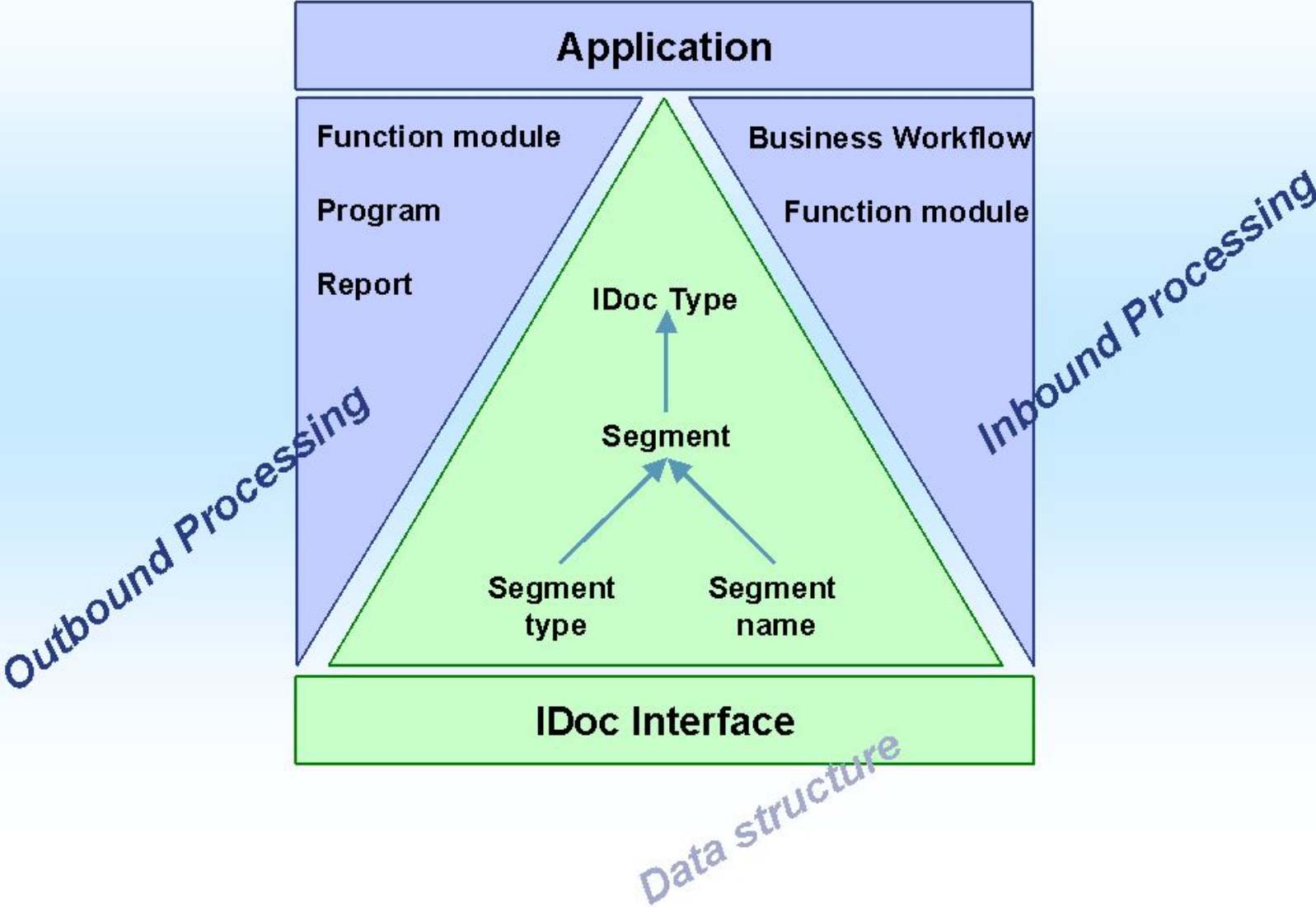


- **Data Structures:**
 - **IDoc Record Types**
 - **IDoc Types**
 - **IDoc Segments**
- **Development and Extension**



- **Understand the development environment for IDoc types**
- **Describe the functionality of the development environment for IDoc types**
- **Explain the difference between development and extensions**





Control Record

IDoc ID
Partner
IDoc Type and Logical Message
External Structure

Data Records

Control Part	Application Data
--------------	------------------

Status Records

IDoc ID
Status Information

Basic type

=

IDoc type

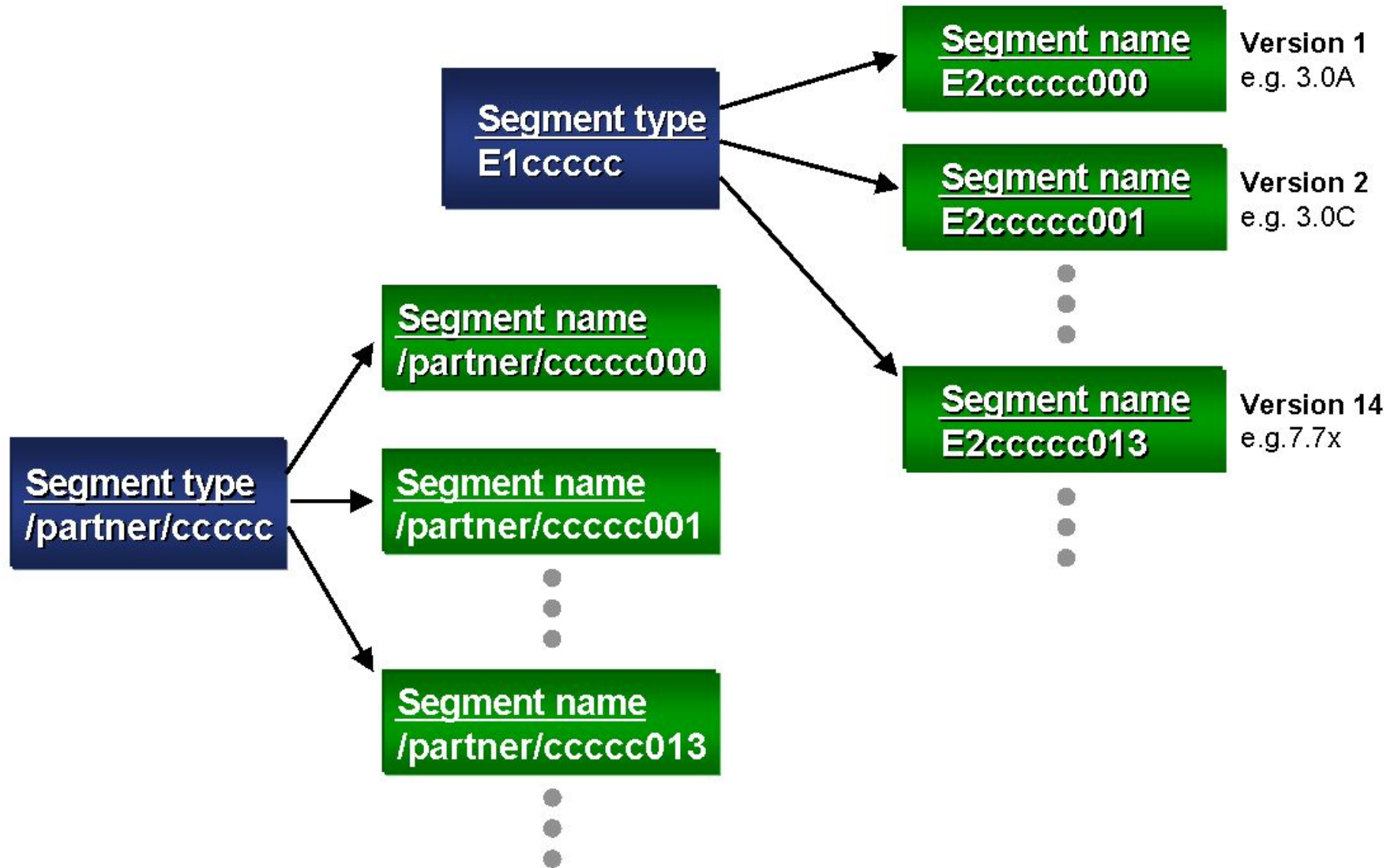
Basic type

Extension

+

=

IDoc type

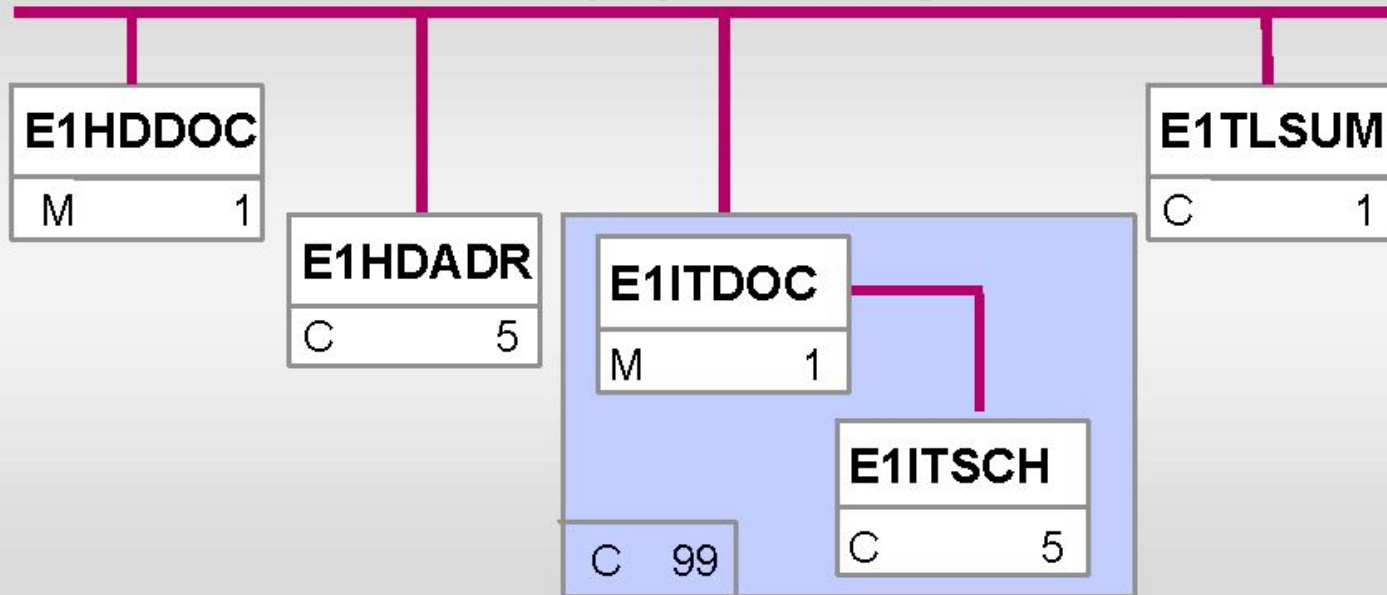


- **By releasing segments and IDoc types, the external interface is "frozen" and given unique names for these objects for an external partner system.**
- **There can only be one segment version for each SAP Release (for example 4.0B).**
- **The IDoc definition tools control the release. After each release, further development leads to new versions.**
- **Changes must be made in accordance with strict rules so that the interface remains compatible.**

Control record, EDIDC

"IDoc Type"

Data records, EDIDD, displayed as a segment tree

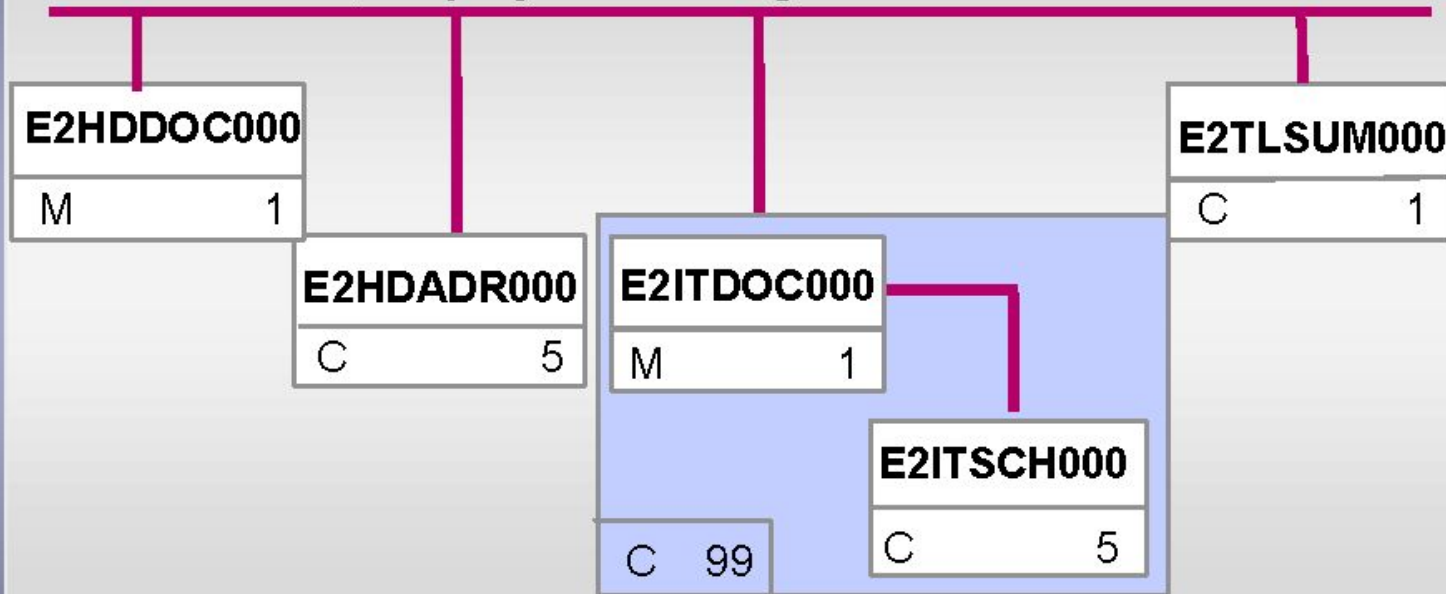


Status record, EDIDS & EDI_DS

Control record, EDI_DC40

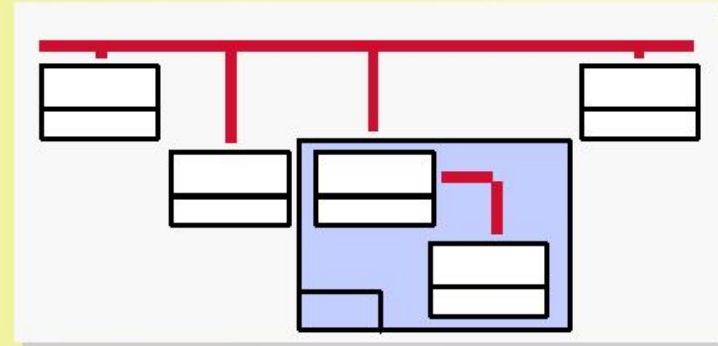
"IDoc Type"

Data records, displayed as a segment tree



Status record, EDI_DS40

IDoc Type Editor



WE30

Segment editor

E1HDDOC

WE31



The IDoc type required is available and meets all requirements:



No action needed!



The IDoc type required is present but does not meet all requirements:



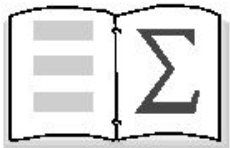
Action: Customer extension!



The IDoc type required is not available or it *is* present but does not meet all the requirements:

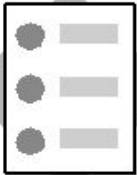


Action: Development!

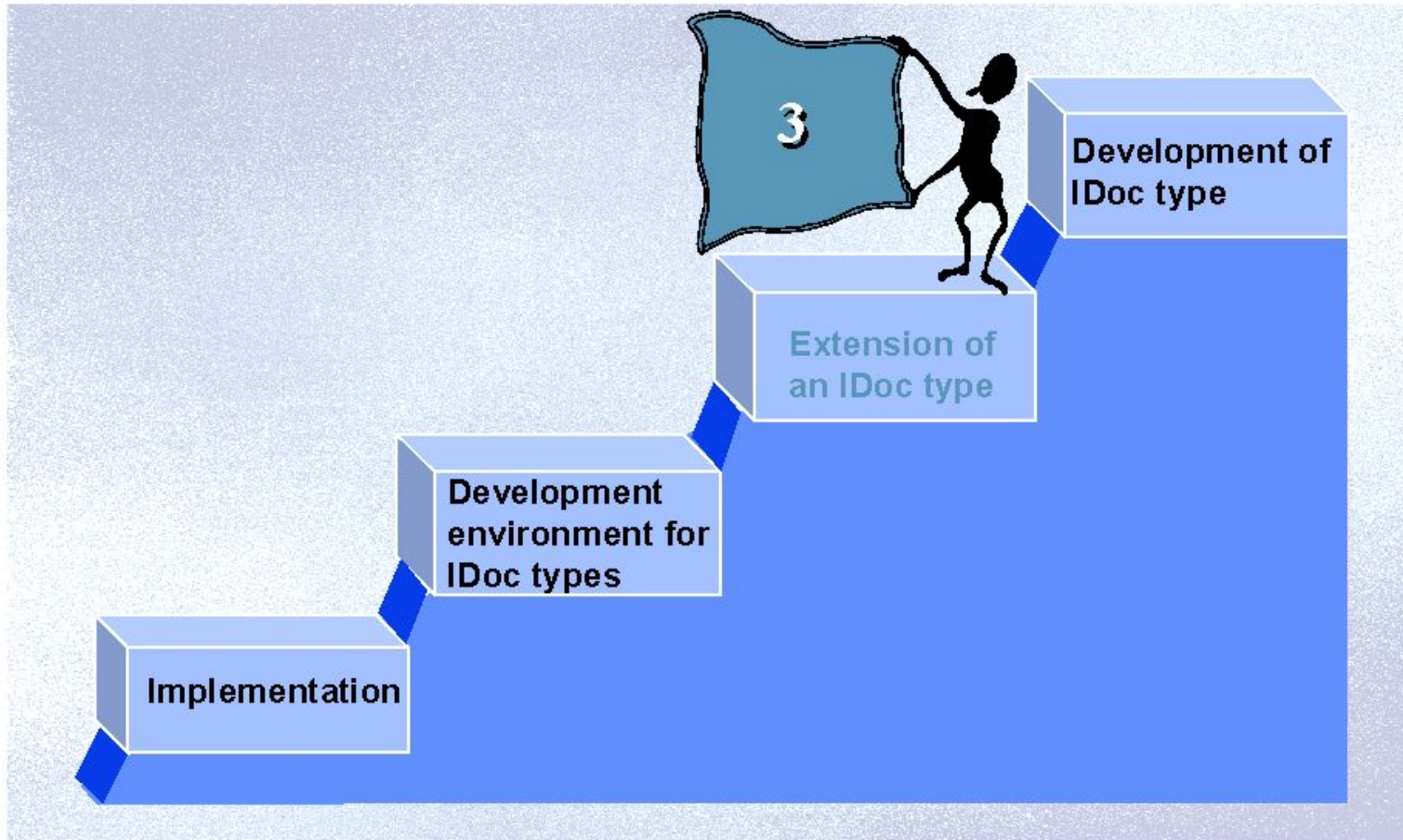


- **An IDoc type is a complex data structure composed of segments.**
- **Segment types are structures in the ABAP data repository.**
- **IDoc types and their processing can be extended in the appropriate positions by customers.**

- **Extension of the data structure**
- **Extension of Outbound Processing**
- **Extension of Inbound Processing**

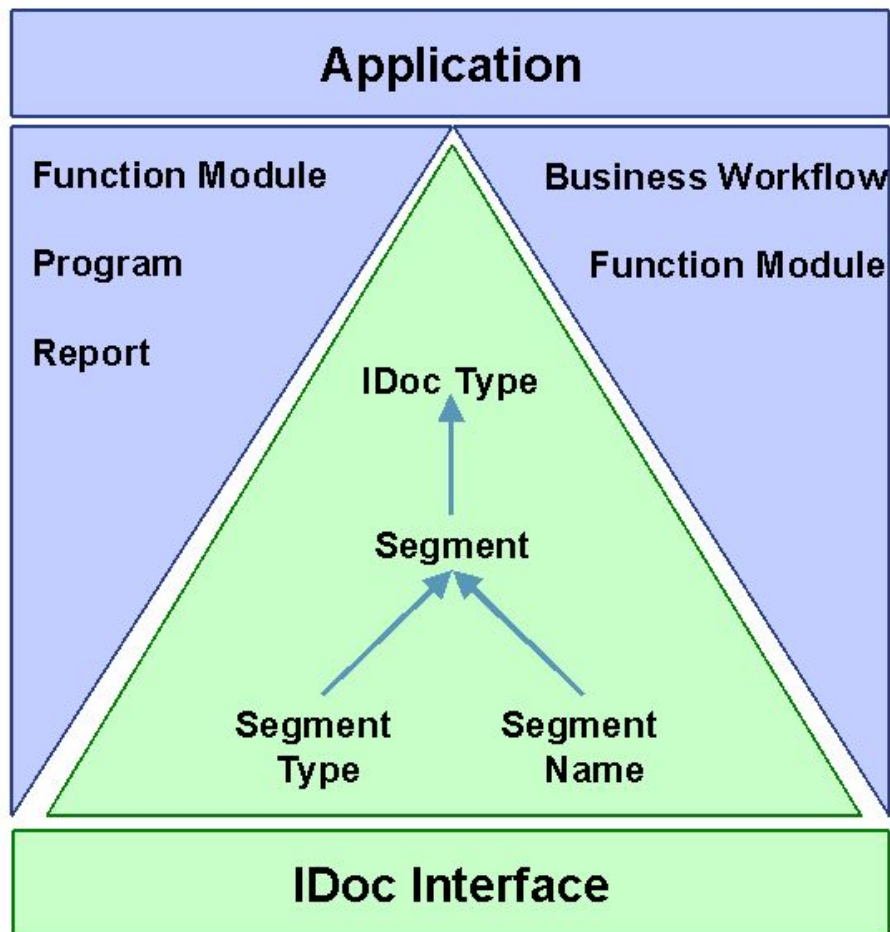


- **Create IDoc Segment**
- **Extend an IDoc Type**
- **Implement a customer exit in outbound processing**
- **Implement a customer exit in inbound processing**

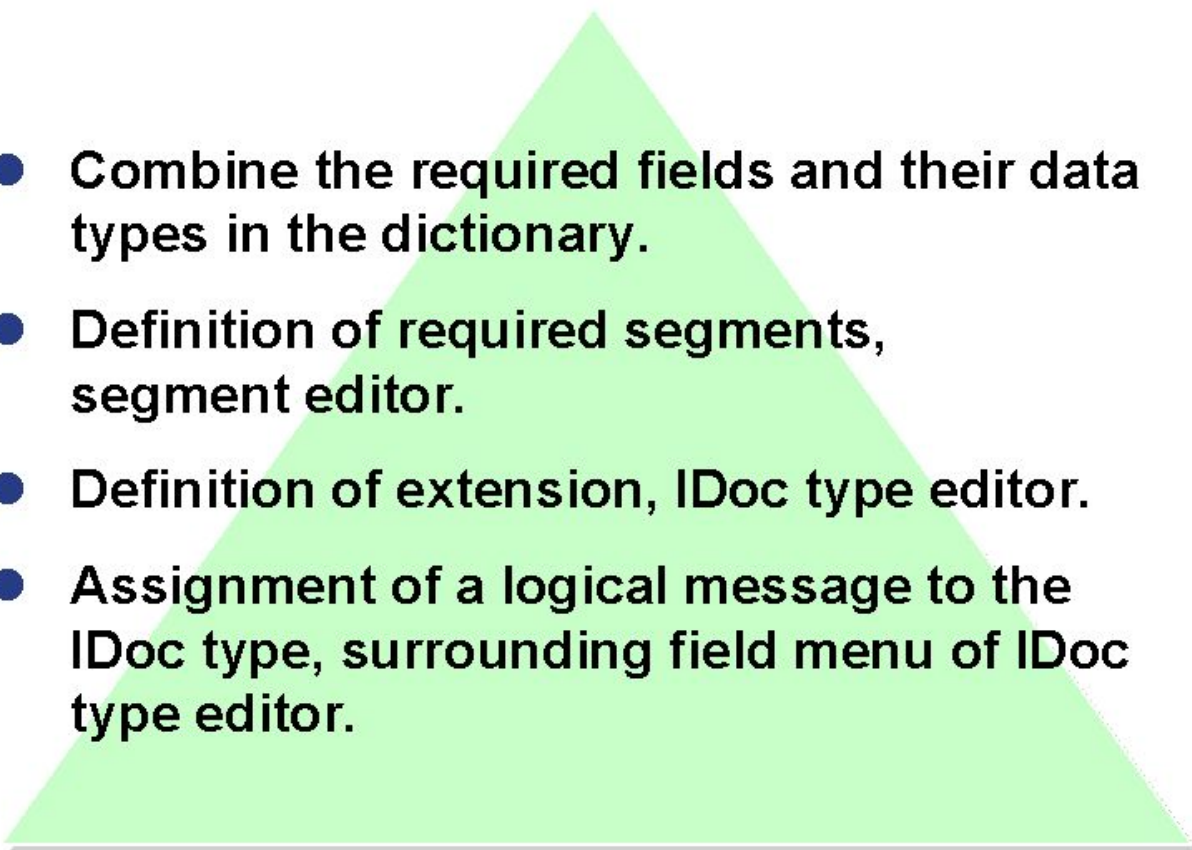
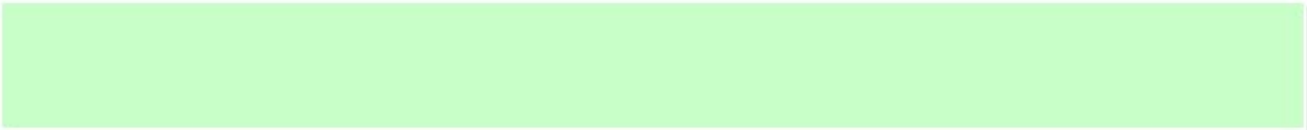


- **The coding provided in the standard system for processing is used.**
- **Developments and corrections of the coding delivered in the standard system are therefore automatically available.**
- **Extension is less time consuming than new development.**

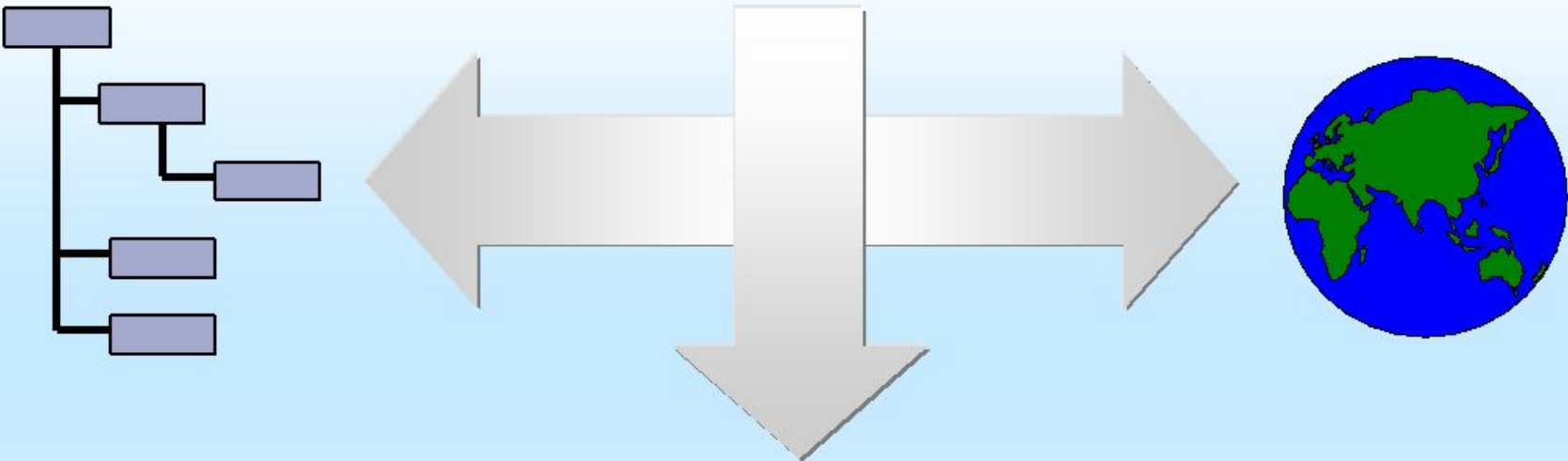
- **Additional customer fields are recorded in their own customer segments.**
- **Customer segments depend on SAP segments (successor or child relationships).**
- **The processing of customer segments is exclusively implemented in the customer exits of the coding provided in the standard system.**



- Data structure in the WEDI area menu
- Outbound and inbound processing: Typically through transaction "CMOD", otherwise through individual technique
- Documentation: In the WEDI area menu

- 
- **Combine the required fields and their data types in the dictionary.**
 - **Definition of required segments, segment editor.**
 - **Definition of extension, IDoc type editor.**
 - **Assignment of a logical message to the IDoc type, surrounding field menu of IDoc type editor.**
- 

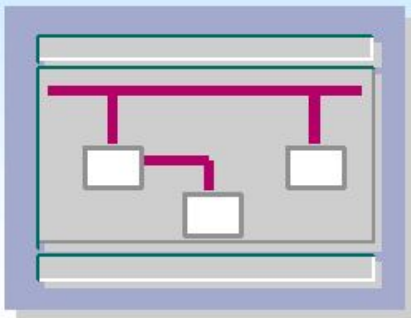
- **Definition of a project, project management, attribute**
- **Choosing the "correct" customer exits, project management, SAP extensions**
- **Implementation of the selected customer exits, project management, extension components**
 - **Outbound processing: reading of the SAP database and data in "IDoc format"**
 - **Inbound processing: writing data from the "IDoc format" into the SAP database**
- **Activate project in project management**

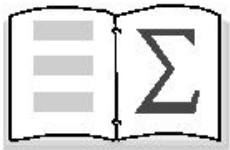


Begin
...
End

XML

```
typedef struct edi_dc  
{...}  
edi_dc
```





- **An extension can be limited to processing.**
- **Normally an extension encompasses the data structure as well as the processing.**
- **Extended data structures can be communicated to the subsystem by the documentation tools.**

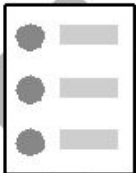
Erweiterung eines IDoc-Typs: Exercise

Data Used in the Exercises

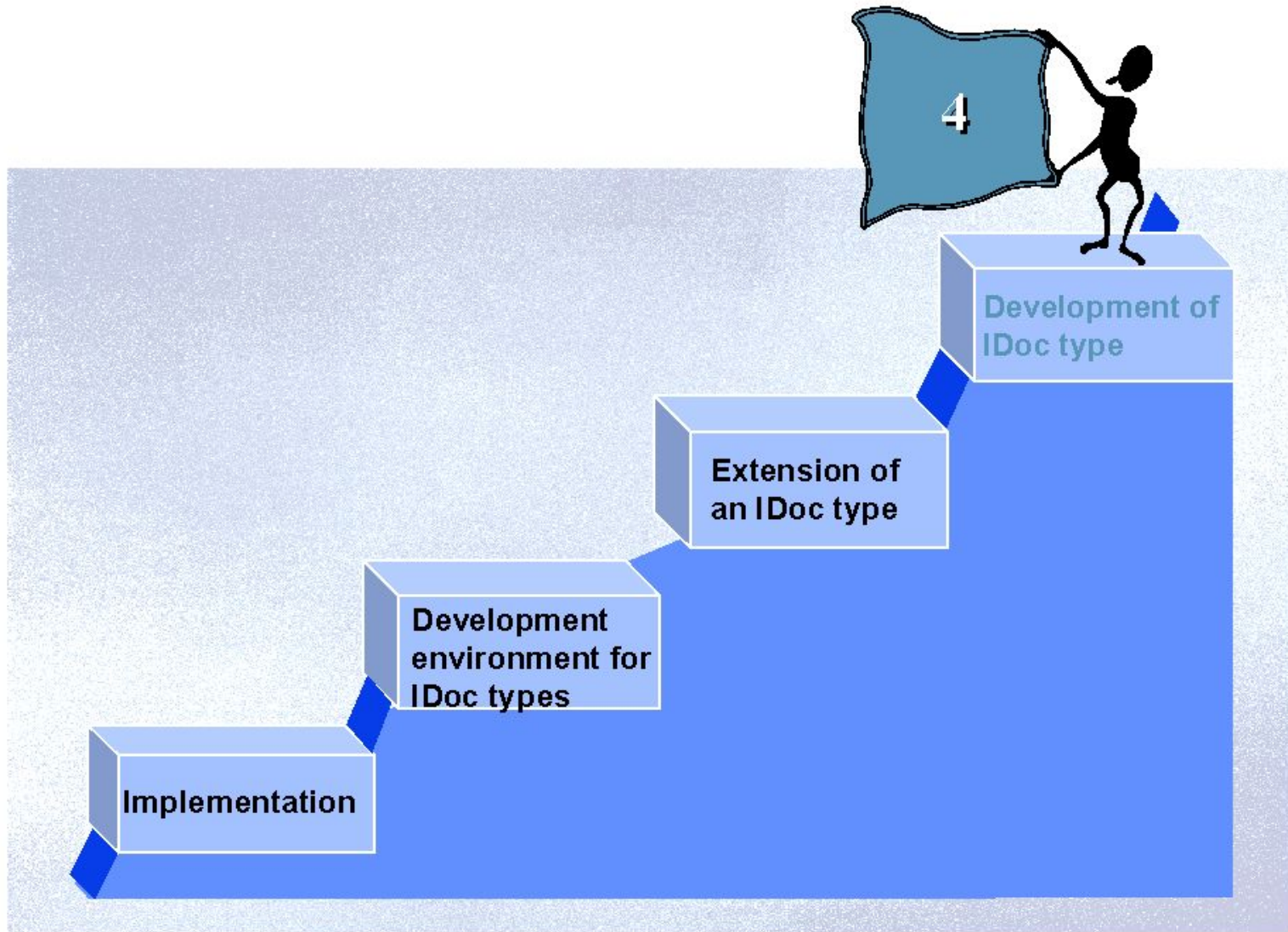
Extension of an IDoc Type: Solutions

Unit: Extension of an IDoc Type

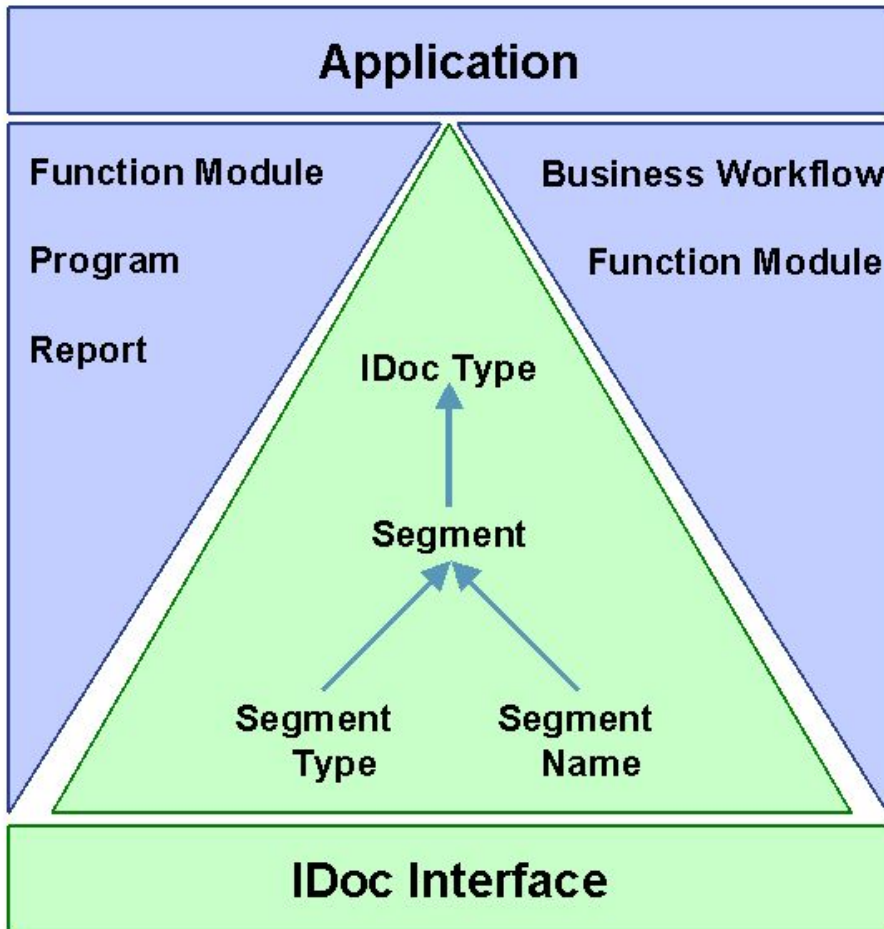
- **Development of the data structure**
- **Processing example**



- **Create IDoc segments**
- **Develop new IDoc types**
- **Template for IDoc outbound processing**
- **Template for IDoc inbound processing**



- **Segments are formed as logical units based on the (application) fields. They are reusable "modules" of IDoc development.**
- **Segment groups are formed as logical units from segments.**
- **IDoc types are derived from segments and segment groups.**
They are the data structure of an application document for the transmission.
- **During development IDoc types are created as basic IDoc types.**
- **The business process itself is not identified through an IDoc type, but rather through the logical message.**



- **Data structure:**
in the WEDI area menu
- **Outbound and inbound processing:** in the Object Browser "SE80"
- **Documentation:**
in the WEDI menu

- **Combine the required fields and their data types in the dictionary**
- **Definition of required segments, segment editor**
- **Definition of basic IDoc types, IDoc type editor**
- **Creation of a logical message, surrounding field menu of IDoc type editor**
- **Assignment of a logical message to the IDoc type, surrounding field menu of IDoc type editor**

- **Outbound processing using**
 - **Message Control (MC) ?**
 - **ALE Interface (FM MASTER_IDOC_DISTRIBUTE)?**

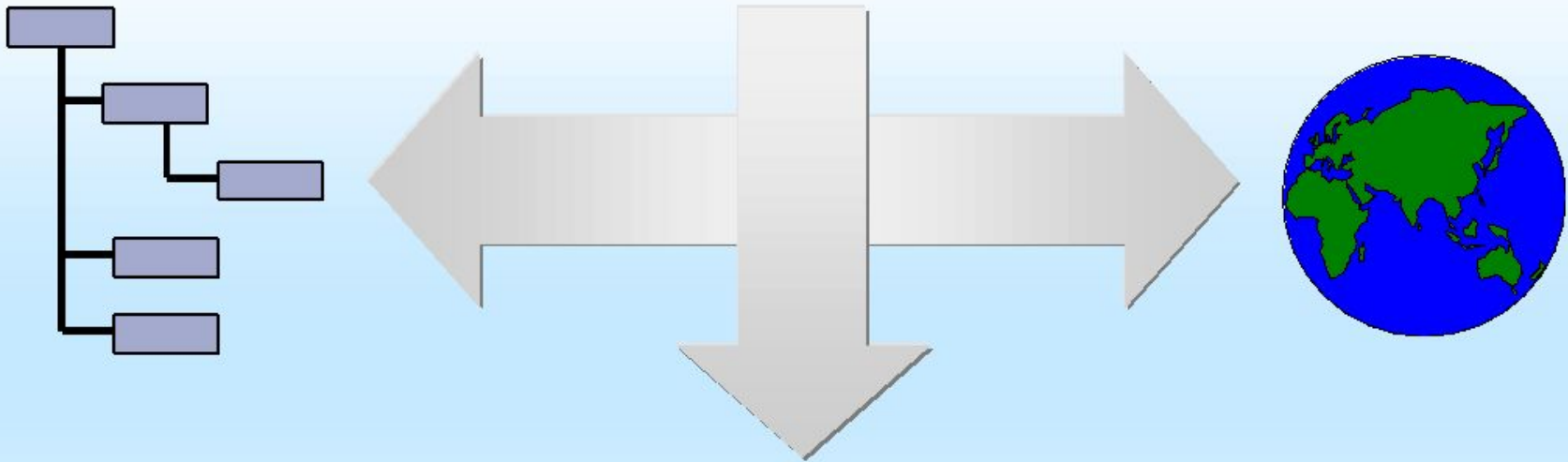
- **Inbound processing using**
 - **Business Workflow (single step or multistep task) ?**
 - **ALE Interface (function module) ?**

- **Checking Message Control, that is, the connection for the message default:**
 - **Condition table**
 - **Access sequence**
 - **Condition types**
 - **Procedure**
 - **Application**
- **Implementation of a function module that makes application data available in "IDoc format"**
- **Definition of an outbound process code**

- Implementation of a program that calls the function module **MASTER_IDOC_DISTRIBUTE**. Interface
 - Import **MASTER_IDOC_CONTROL** like EDIDC
 - Tables **COMMUNICATION_IDOC_CONTROL** like EDIDC
MASTER_IDOC_DATA like EDIDD
- For this program you have to define how the control works, specifically, if the control is to take place via a time table, event or dialog step.

- **Implementation of a workflow to control inbound processing**
This includes:
 - **Definition of object and method(s) in BOR**
 - **Definition of workflow**
- **Definition of inbound process code**

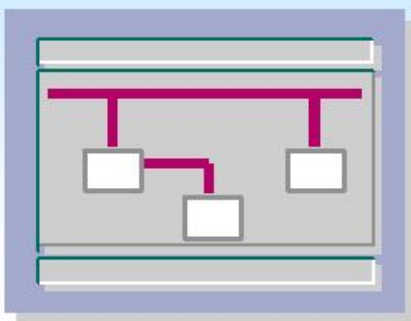
- Write function module that posts the IDoc as an application document
- Implementation of a workflow to send messages in case of error. This includes:
 - Definition of object and method(s) in BOR
 - Definition of workflow
- Maintenance of the characteristics for the function module for the ALE Call
- Allocation of the function module to a logical message and to an IDoc type for the ALE Call
- Definition of an inbound process code and establishing a connection with the ALE layer

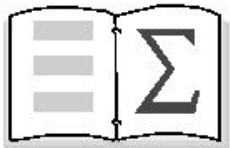


Begin
...
End

XML

```
typedef struct edi_dc  
{...}  
edi_dc
```





- **The data structure is defined as an IDoc type**
- **Depending on the processing course and direction, different programming templates can be used**
 - **Outbound uses 2 methods:
Message Control and ALE**
 - **Inbound uses 2 methods:
Workflow and ALE**

Appendix