FEU INSTITUTE OF TECHNOLOGY
COLLEGE OF ENGINEERING • COLLEGE OF COMPUTER STUDIES

ITSE333A
ABAP

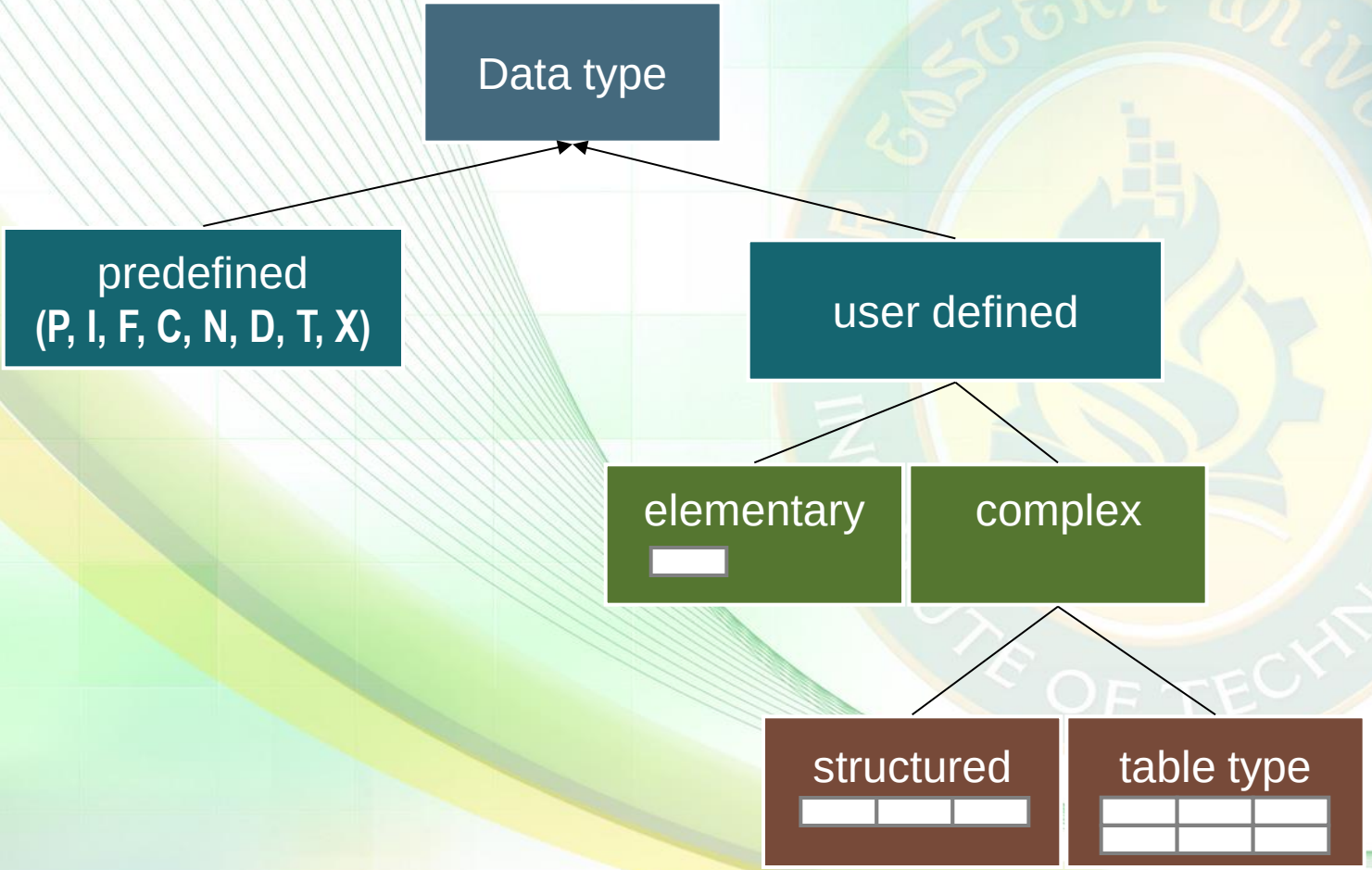
Lesson 3: Basic Concepts

Anthony D. Aquino

Agenda

1. Data types and data declaration
2. Important instructions
3. Local modularization
4. Background processing

Data types


Data type
predefined
(P, I, F, C, N, D, T, X)
user defined
elementary
complex
structured
table type

Predefined data types in ABAP

Data type	Sense	Initial value	Values range
d	Date	00000000	
t	Time	000000	
i	Integer	0	
f	Float	0.00	
String	String		
Xstring	Byte		
p	Packed number	0	
n	Numerical text	00 … 0	Max. 65536 figures
c	Character	<SPACE>	Max. 65536 characters
x	Byte (hex)	X'00'	

Data declaration

- Elemental field definition:

```
DATA f(len) TYPE <DATA TYPE>.
```

- Structured data object:

```
DATA: BEGIN OF struc,
…
END OF struc.
```

- Internal table:

```
DATA itab TYPE <TABLE TYPE>. or
DATA itab TYPE TABLE OF <STRUCTURE>.
```

- Constants:

```
CONSTANTS c VALUE <value> / is INITIAL.
```

- Parameters:

```
PARAMETERS TYPE <DATA TYPE>.
```

Data declaration

- Instead of defining every single data:

```
Data a type c.
Data b type i.
Data c type c.
Data d type i.
```

- Use:

```
Data: a type c, b type i, c type c, d type i.
```

Definition of own data types

- Definition of completely new data types
- New data types can derive from existing data types:

```
TYPES text10 TYPE c LENGTH 10.
```

- Definition of one's own data type:

```
TYPES: BEGIN OF str_student,
name(40) TYPE c,
family_name(40) TYPE c,
id TYPE i,
END OF str_student.
```

Definition of own data types

- Declaration of a new structure:

```
DATA student TYPE str_student.
```

- Access to the structure:

```
WRITE student-name.
```

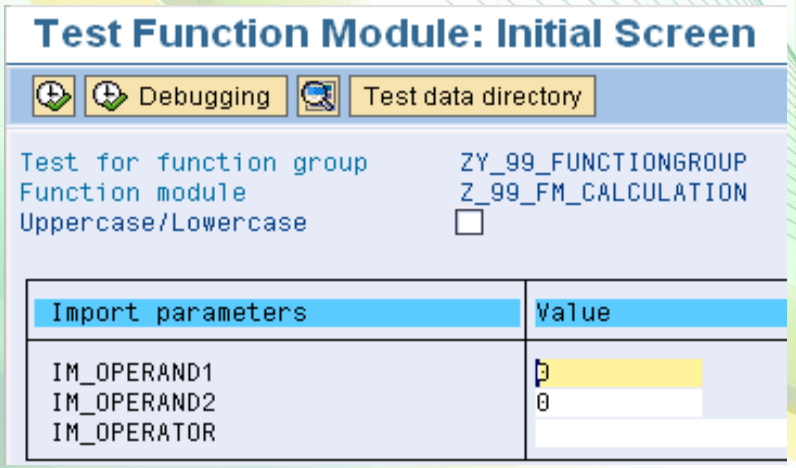
Structure SYST

- Structure SYST contains many system variables from the SAP system
- Structure can be viewed in Data Dictionary (SE11) via data type SYST

Field	Sense
Sy-subrc	Returncode of last instruction
Sy-date	Current date and time
Sy-uname	Username of the current user
Sy-host	Name of application server
Sy-langu	Current system language
Sy-dbsys	Name of database server
Sy-tcode	Current transaction code
Sy-index	Loop index
Sy-client	Current client number

Selection screens

- Selection screens simplify interaction with user
- Selection screens always have Dynpro number 1000
- Selection screens are generated automatically when keyword `Parameters` is used in source code
- `Parameters` is also used for variable declaration



Important instructions and control structures

- Data manipulation
- Data object conversion
- Control structures
  - Loops
  - Branching conditionally

Data manipulation

- Assign: `MOVE f TO g` or `g = f`
- Numeric: `ADD n TO m` or `m = m + n`
- String: `CONCATENATE, SPLIT, SEARCH, REPLACE, CONDENSE, TRANSLATE …`
- Logical:
  - For all data types: `=, <>, <, >, <=, >=`
  - For character like types: `CO (contains only), CN (contains not only), CA (contains any) …`
  - For byte like types: `BYTE-CO, BYTE-CN, BYTE-CA …`
  - For bit patterns: `O (Ones), Z (Zeros), M (Mixed)`

String Operations

- **Concatenate** - statement allows two character strings to be joined so as to form a third string.

  Syntax:

```
concatenate  f1 f2 into d1.
```

- **Concatenate using SEPARATED BY** – string concatenated will be added by a space.

  Syntax:

```
CONCATENATE f1 f2 INTO d1 SEPARATED BY d2.
```

String Operations

- **SPLIT** – statement is used to separate its contents of a field into two or more fields.

  Syntax:

  SPLIT mystring AT separatestrings INTO a1 a2 [an].

- Example:

  mystring =‘1234**ABCD**7890

  will become 1234 ABCD 7890

String Operations

- **SEARCH** – searches for a specific character strings.

  Syntax:

  SEARCH field.[for]=value.

String Operations

- **REPLACE** - Replaces the sub string with another sub string specified, in the main string. If replaced successfully then sy-subrc is set to 0, else set to 4.

Syntax:

```
REPLACE s1 WITH s2 INTO field.
```

String Operations

- **CONDENSE** – removes the blank spaces between words in the variable, leaving only one character's spaces.

  Syntax:

  CONDENSE field.

Data object conversion

- If it is possible to migrate values from one data type to another, the SAP system does it automatically
- Static incompatible: between date and time
- Dynamic incompatible: between char ‘1234hello’ and integer
- Dynamic compatible: between char ‘1234’ and integer 1234
- Exceptions can be caught

```
CATCH SYSTEM-EXCEPTION conversation_errors = 4.
…
ENDCATCH.
```

Control structures: loops

- WHILE – ENDWHILE:

```
WHILE <logical expression>.
  <instructions>
ENDWHILE.
```

- DO – ENDDO

```
DO n TIMES.
  <instructions>
ENDDO.
```

- Sy-index: returns the current loop index and refers to the current loop (in case of nested loops)

Control structures: branching

- IF:

```
IF <logical expression>.
  <instruction 1>
[ELSEIF <logical expression>.
  [<instruction 2>
[ELSE.
  [<instruction 3>
ENDIF.
```

Control structures: branching

- CASE:

```
CASE <data object>.
  [WHEN <value 1>.
        [<instruction 1>.
  [WHEN <value 2>.
        [<instruction 2>.
  [WHEN OTHERS.
        [<instruction 3>.
ENDCASE.
```

Control Statements

- ABAP has 2 conditional logic statements:

  `IF/ENDIF` and `CASE/ENDCASE`

  2 loops: `DO/ENDDO`
  `WHILE/ENDWHILE`

  and others such as `CONTINUE`, `CHECK` & `EXIT`.

- ABAP does not have a `GOTO` statement.

- Control commands can be nested and/or joined with logical operators.

Logical Expressions

... <field> <operator> <literal> ...

... <field1> <operator> <field2>

... <logical expression> AND <logical expression>
... <logical expression> OR <logical expression>
... NOT <logical expression> ...

```
DATA: START   TYPE D,
      SUM1    TYPE P,
      SUM2    TYPE P.
.
.
IF  SUM2  GE 1000.
IF  START IS INITIAL.
IF  SUM1  GT SUM2 AND
    SUM1 BETWEEN 0 AND 100.
IF  SUM1  = 1000 AND
    ( SUM2 LE 2000 OR
      START IS INITIAL ).
```

Operator	Meaning
EQ =	Equal
NE <> ><	Unequal
GT >	Greater than
GE >= =>	Greater than or equal
LT <	Less than
LE <= =<	Less than or equal
BETWEEN f1 and f2	Interval
IS INITIAL	Initial value

Local modularization

- Modularization in ABAP:
  - Includes
  - FORMs (Procedures)
  - Function Groups / Function Modules

Local modularization: Includes

- Outsource to external program
- The include-object is used in the main program to call the external program
- Instruction INCLUDE integrates external program into main program
- INCLUDE vs TOP INCLUDE:
  - TOP INCLUDE also contains data declaration, which must be available in all selection screens

Local modularization: FORMs

- Procedures in ABAP
- Declaration:

```
FORM <procedure name>
   USING value<input parameter> TYPE <type>
   USING <input parameter> TYPE <type>
   CHANGING <input/output parameter> TYPE <type>
   CHANGING value<input/output parameter> TYPE <type>.
ENDFORM.
```

- Parameter without value declaration means the variable points to the global variable
- Parameter with value declaration have their own values

Local modularization: FORMs

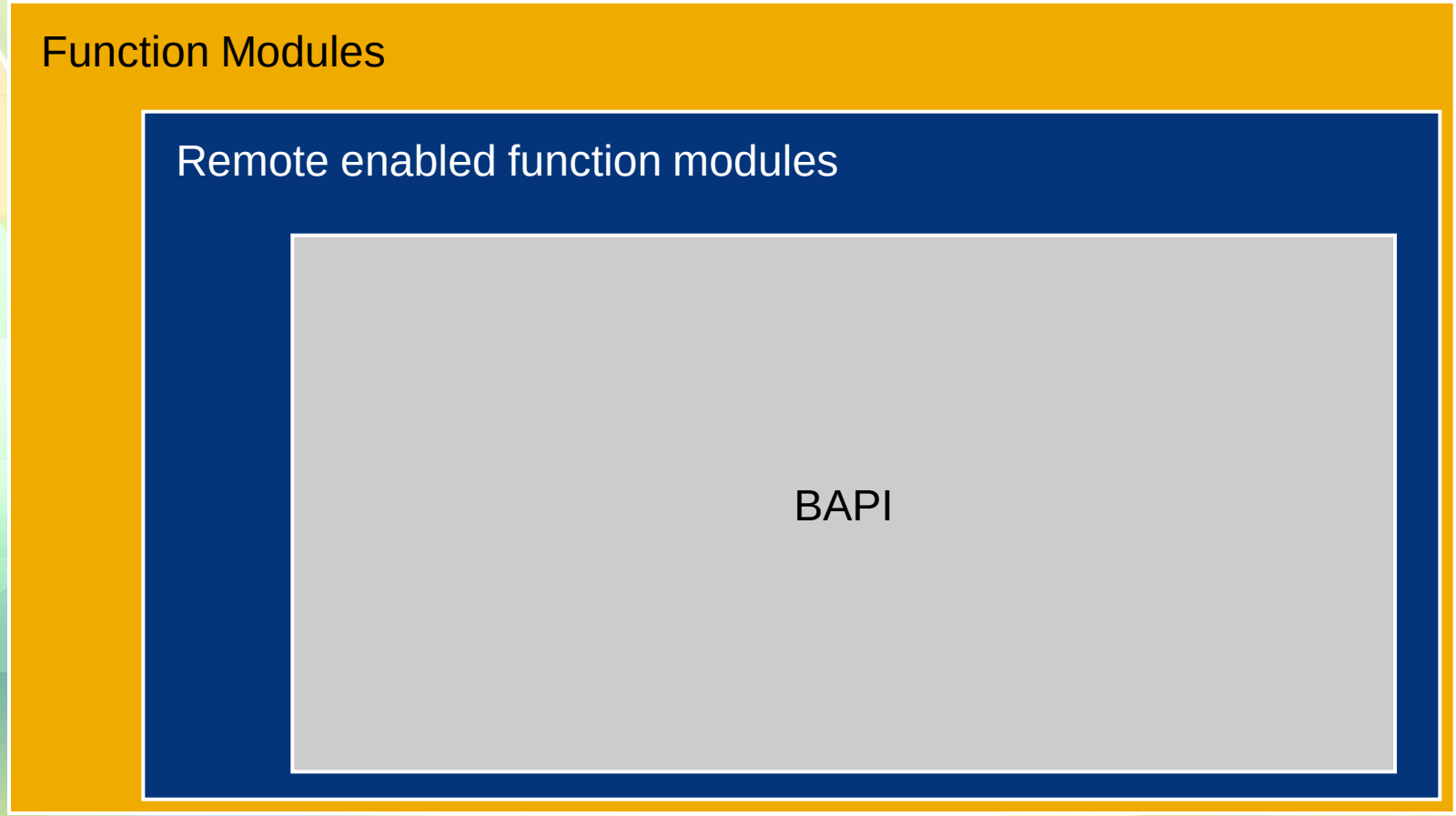
- Call:

```
PERFORM <procedure name>
   USING <input parameter>
   CHANGING <input/output parameter>.
```

Local modularization: Function modules

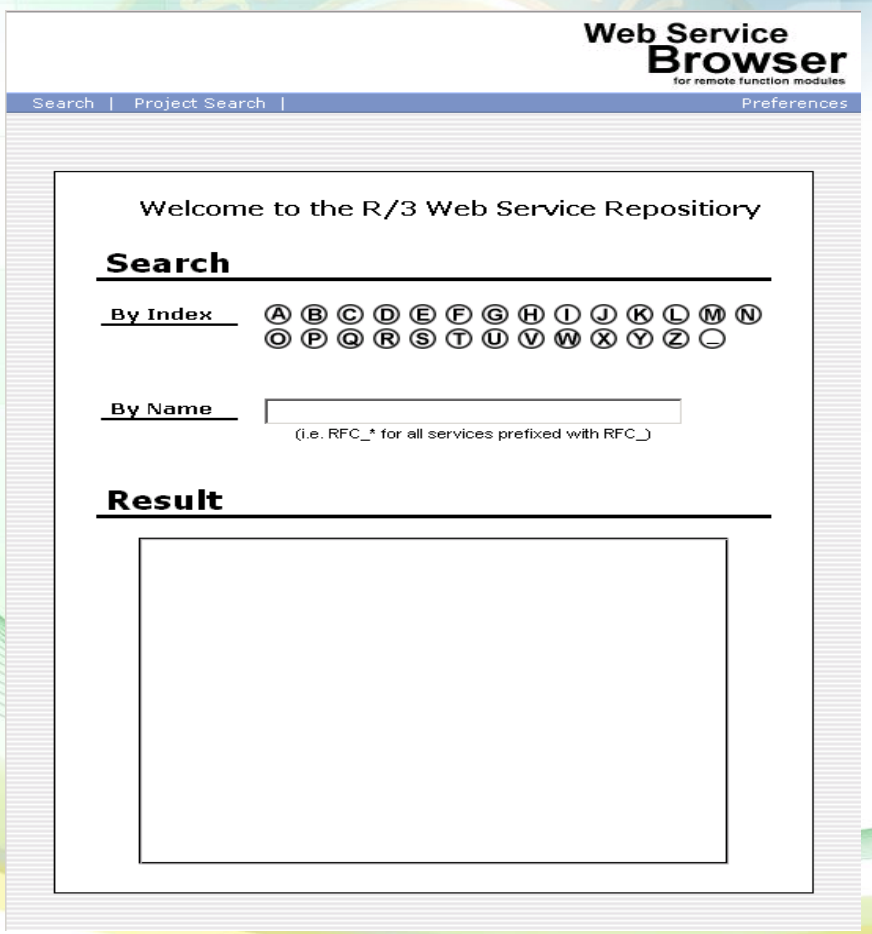
- Outsources functionality to external module
- Function modules are not allowed to access global variables → export variables when calling function module
- More than 100,000 function modules available
- Function modules can be organized in function groups
- Function modules can be remote accessible
- Function groups may have own TOP include

Local modularization: Function modules



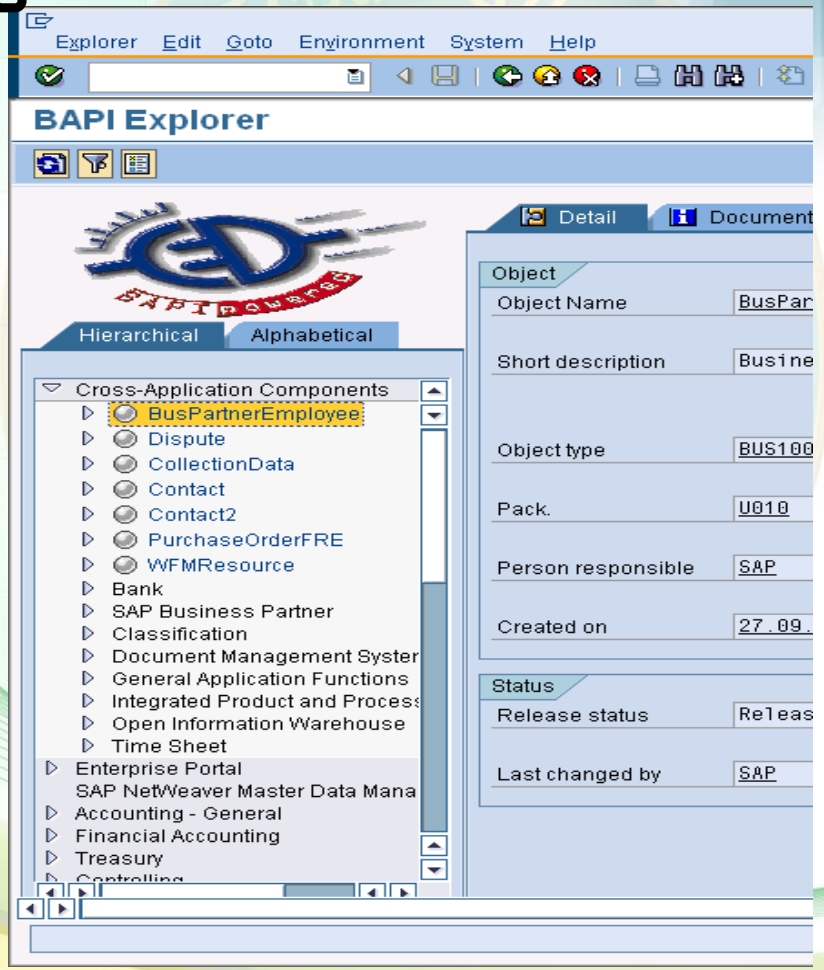
Local modularization: Function modules

- Since WebAS 6.20 web services are available
- Web service browser available under: http://<host>:<ABAPport>/sap/bc/bsp/sap/webservicebrowser/search.html
- <host> and <ABAPport> can be obtained from TA SM51



Local modularization: Function modules

- BAPI = Business Application Programming Interface
- RFC enabled function modules
- Overview about all BAPI can be obtained from BAPI explorer (TA BAPI)



Local modularization: Function modules

- Usage of BAPI's:
  - BAPI give you the functionality of a SAP transaction → be sure to be familiar with the SAP transaction
  - Search for the appropriate BAPI and read the documentation carefully
  - Test the BAPI using the Function Builder
  - Use the BAPI
- Possible problems:
  - Pay attention to the data types and mandatory data

Background processing

- Usual programs use dialog work processes
- Long running programs should always run in the background
- All ABAP programs can be scheduled as background jobs in TA SM36
- For ABAP programs with a user interface you can predefine the user input by using variants