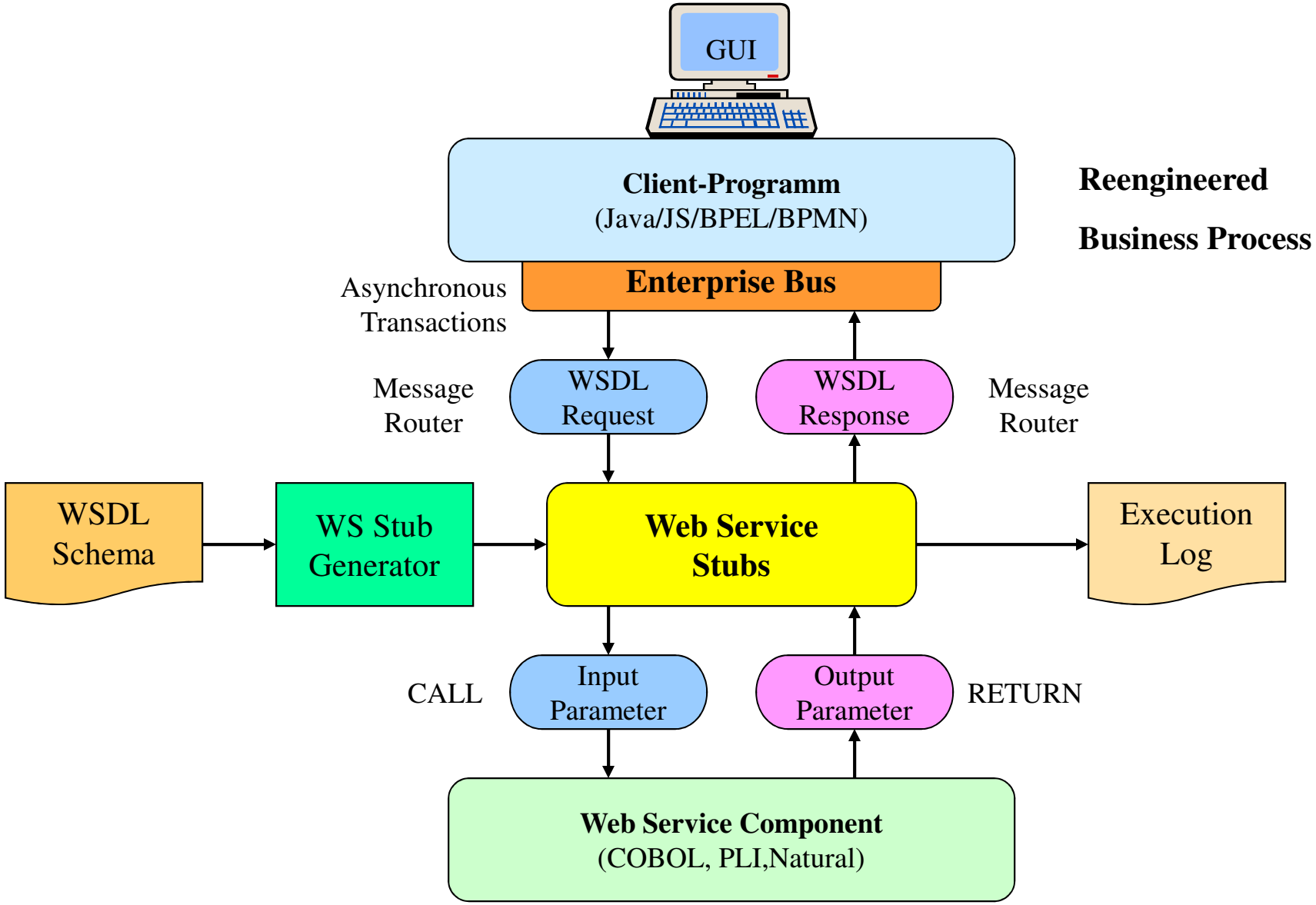
MESOA-2010
Temesvar, Sept. 2010

SOA Integration as an Alternative to Migration
(The role of data type conversion)

Harry M. Sneed
University of Regensburg
ANECON GmbH, Vienna

Integrating wrapped legacy components

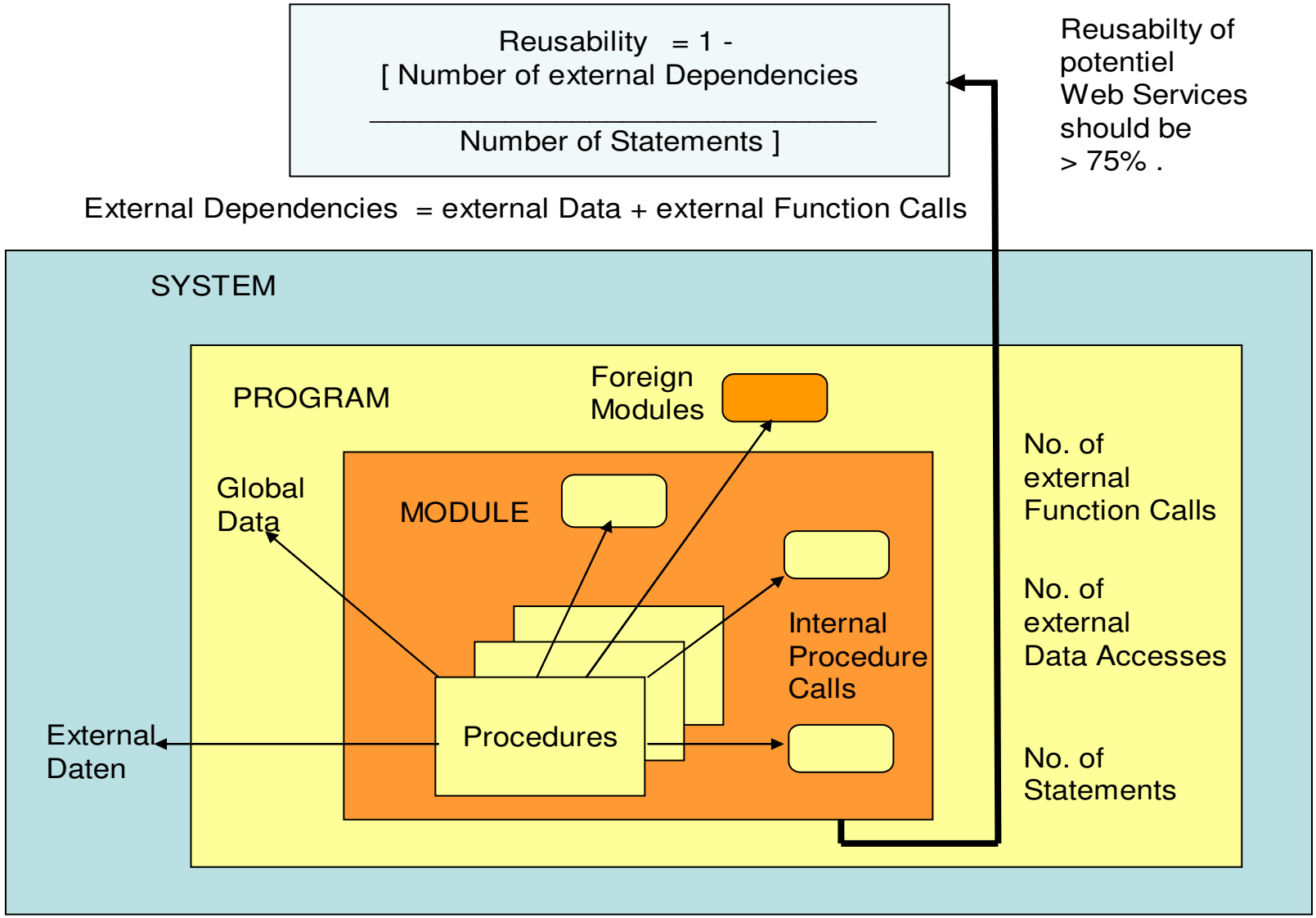


7 Steps to recovering Web Services from Legacy Systems

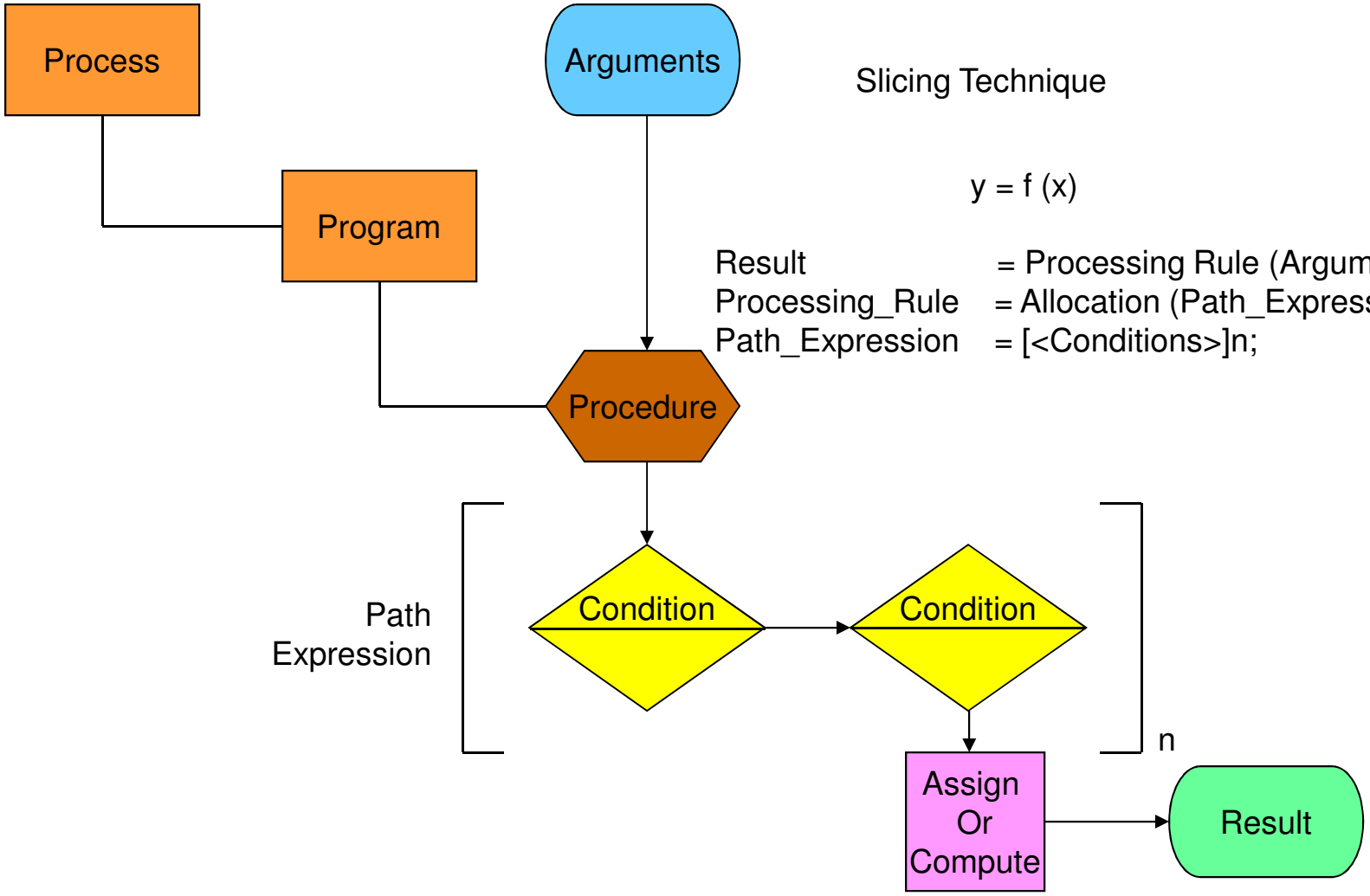
1. Discover potential services in the legacy code
2. Evaluate them for Reusability
3. Strip away the Code not needed
4. Extract the Code needed
5. Wrap the Code behind a WSDL interface
6. Generate wrapper code to convert the data types in the service request and transfer them to the parameters of the wrapped code.
7. Generating wrapper code to convert the results of the wrapped code and transfer them to the data types in the service response.

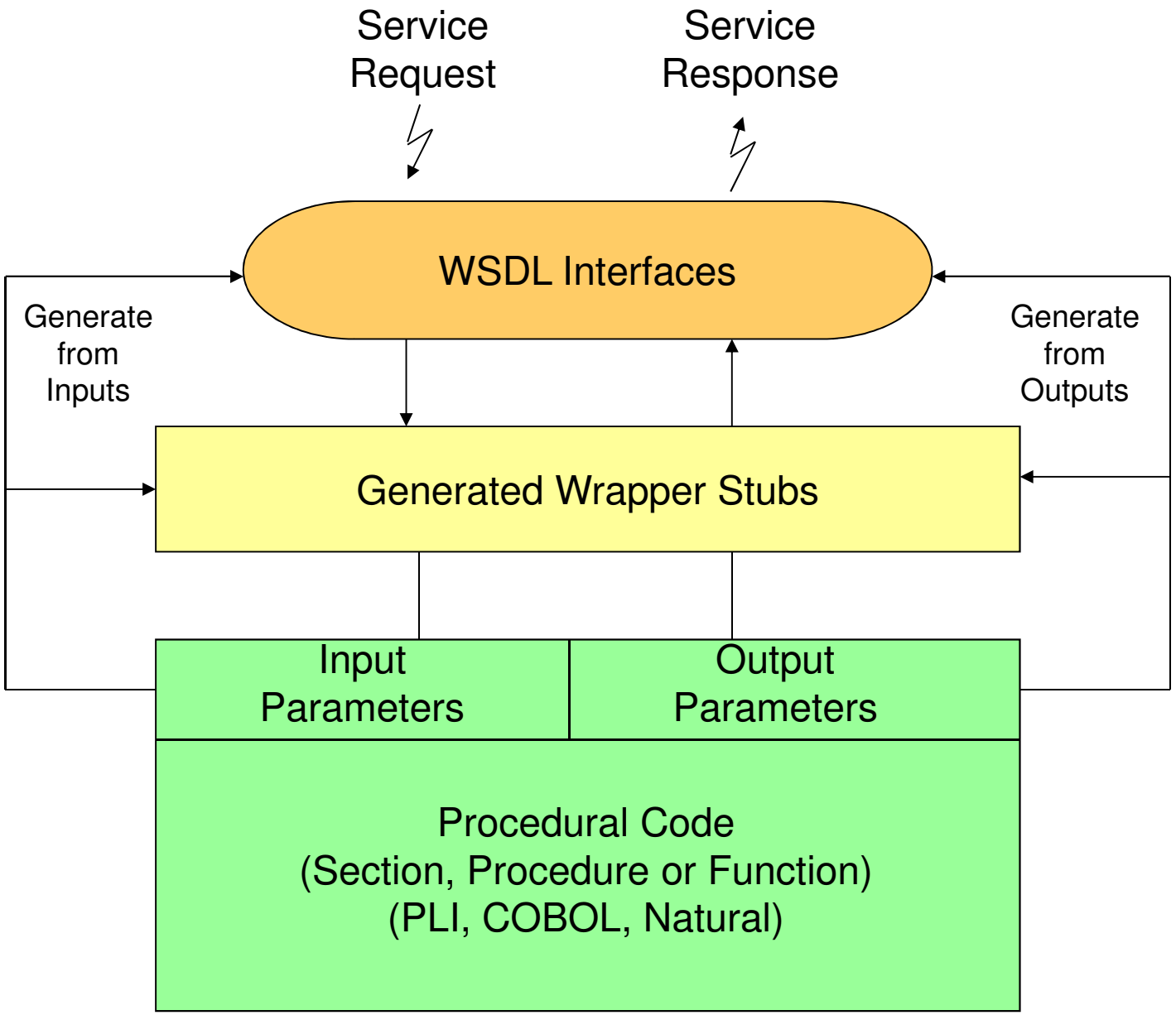
Evaluating the Reusability of a Software Component

Abb. 8.5



Identifying reusable Code Blocks based on their Results (Business Rule Identification)





Generating the Wrapper Code

The Rationale for Converting Data Types prior to Wrapping

- If the data types are not converted to ASCII character format, the data coming from the web service request must be converted every time a service request comes in.
- Converting WSDL data types to local data types not only takes time but is also error prone. Therefore, the data type definitions in the program should match those in the web service interface.
- When the databases are migrated all data should be converted to a common human readable format, namely ASCII character code.
- To avoid having to convert the data when data is read in from the database, the data type definitions in the program should match those in the database.
- Run time data casting should be avoided.
- Therefore all data should be in character format and only converted when they are used in an arithmetic operation.

Typical incompatible Data Types in Legacy Programs

Packed Decimal or COMP-3 in COBOL or Fixed Decimal in PL/I
Two digits are stored in each byte with the sign in the last byte

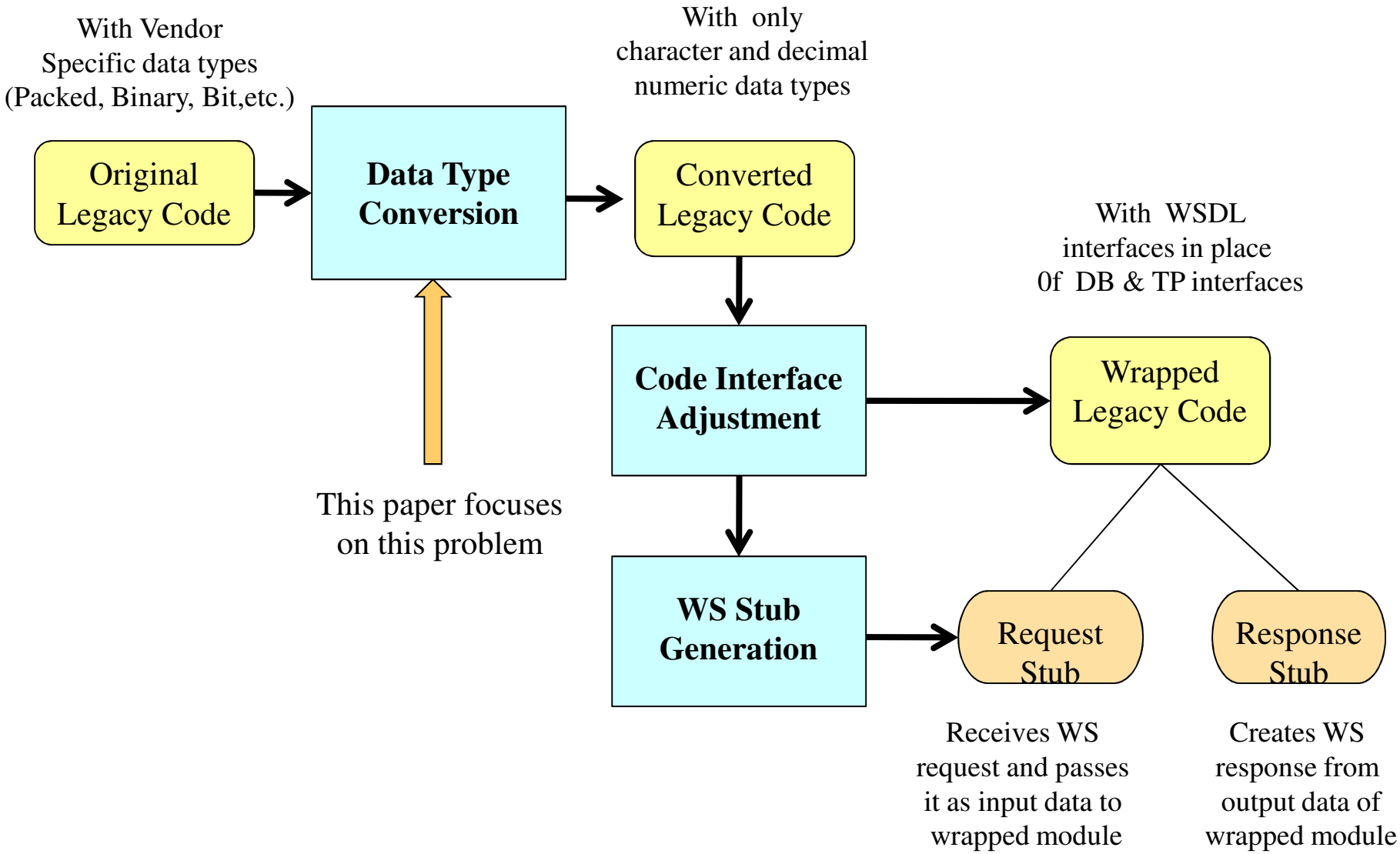
Binary Fullword or COMP 9(9) in COBOL or Fixed Binary (31) in PL/I
A number is stored in a four byte field.

Binary Halfword or COMP 9(5) in COBOL or Fixed Binary (15) in PL/I
A number is stored in a two byte field.

Floating Point or COMP-2 in COBOL or Float in PL/I
A real number with a variable fraction is stored in a 16 byte field.

Bit String or CHAR X‘Hex‘ or BIT(n) in PL/I
A string of one or more bits is stored in a one byte field.

A Web Service Wrapping Process



Original COBOL Data Types

```
01  RPARA-ZE.
      02  RTABZEV.
        03  RZEV                    OCCURS 2500.
          04  RZEVV         PIC 9(01).
          04  RZELEV.
            05  RZEDATVX.
              06  RZEDATV PIC 9(07)   COMP-3.
            05  RZET2.
            06  RUNTNRV    PIC 9(03)   COMP-3.
            06  RBESV      PIC X(01).
            06  RDSTNRV    PIC 9(05)   COMP-3.
            06  RBEPEV     PIC 9(07)   COMP-3.
            06  RKATV      PIC 9(03)   COMP-3.
            06  RTAT1V     PIC 9(03)   COMP-3.
            06  RLVV       PIC X(01).
            06  RLAV       PIC X(01).
            06  RWIEDVX.
              07  RWIEDV  PIC 9(07)   COMP-3.
            06  RDIAGV     PIC 9(03)   COMP-3.
            06  RECV       PIC 9(03)   COMP-3.
            06  RD4V       PIC X(01).
            06  RBONV.
              07  RSTKZ   PIC X(01).
            06  FILLER     PIC X(06).
  *
  *   FREIER BEREICH FUER ZEITELEMENTE VERS.
  *     MUSS BLEIBEN WEGEN UEBERLAUF DER KOMBITABELLE
  *     2700 ZEITELEMENTE ZU 30 BYTE (2500 VERS. + 200 ANG.)
  *
        03  FILLER        PIC X(3600).
      02  FILLER   REDEFINES RTABZEV.
        03  FILLER       OCCURS 2500.
          04  RZED3V         PIC X(05).
          04  FILLER         PIC X(32).
```

Converted COBOL Data Types

```
01  RPARA-ZE.
      02  RTABZEV.
        03  RZEV                    OCCURS 2500.
          04  RZEVV         PIC 9(01).
          04  RZELEV.
            05  RZEDATVX.
REMOVE        06  RZEDATV PIC 9(07)  .
            05  RZET2.
REMOVE        06  RUNTNRV    PIC S9(03)  .
              06  RBESV      PIC X(01).
REMOVE        06  RDSTNRV    PIC S9(05)  .
REMOVE        06  RBEPEV     PIC S9(07)  .
REMOVE        06  RKATV      PIC S9(03)  .
REMOVE        06  RTAT1V     PIC S9(03)  .
              06  RLVV       PIC X(01).
              06  RLAV       PIC X(01).
              06  RWIEDVX.
REMOVE            07  RWIEDV  PIC  S9(07)  .
REMOVE        06  RDIAGV     PIC  S9(03)  .
REMOVE        06  RECV       PIC  S9(03)  .
              06  RD4V       PIC X(01).
              06  RBONV.
                  07  RSTKZ   PIC X(01).
              06  FILLER     PIC X(06).
  *
  *   FREIER BEREICH FUER ZEITELEMENTE VERS.
  *     MUSS BLEIBEN WEGEN UEBERLAUF DER KOMBITABELLE
  *     2700 ZEITELEMENTE ZU 30 BYTE (2500 VERS. + 200 ANG.)
  *
        03  FILLER        PIC X(3600).
      02  FILLER   REDEFINES RTABZEV.
        03  FILLER       OCCURS 2500.
          04  RZED3V         PIC X(05).
          04  FILLER         PIC X(32).
REMOVE 04  FILLER           PIC X(0110).
REMOVE*  Redefining Group Length does not match Redefined!
REMOVE*  Redefined  Group Length = 00360
REMOVE*  Redefining Group Length = 00250
REMOVE*  Difference is         = 0110
REMOVE*  Generated Filler must pad the shorter Group!
```

Original PLI Data Types

```
DCL SUCHKEY         CHAR(04)        INIT('') ;
DCL LINECOUNTER     BIN FIXED(15,0) INIT('') ;
DCL COUNTER         DEC FIXED(15,0) INIT('') ;
DCL TAWERT_Z        PIC 'Z.ZZZ.ZZZ.ZZ9V,99' ;
DCL TAWERT_NEU      CHAR(16) ;
DCL BSTKNOM         FLOAT(16)   INIT(0) ;
DCL BNOMSFR         FLOAT(16)   INIT(0) ;
DCL H1              FLOAT(16)   INIT(0) ;
DCL H2              FLOAT(16)   INIT(0) ;
DCL H3              FLOAT(16)   INIT(0) ;
DCL STEUER_VALUE    PIC  '9'  INIT(0) ;
DCL LOOP_ERROR_COUNT      BIN FIXED (15,0) INIT(0) ;
DCL AKTUELLE_AKTSTUFE     CHAR(1) INIT('') ;
/* OPERATOR FUER DLI-ZUGRIFFE           */
DCL GLOBAL_OP       CHAR(2) INIT(' =') ;
/* ZAEHLERFELDER FUER LISTEN_OUTPUT     */
DCL VALOREN_I       DEC FIXED (5) INIT(0) ;
DCL VALOREN_O       DEC FIXED (5) INIT(0) ;
DCL BESTAND_I       DEC FIXED (5) INIT(0) ;
DCL BESTAND_O       DEC FIXED (5) INIT(0) ;
DCL TABELLEN_I      DEC FIXED (5) INIT(0) ;
DCL TABELLEN_O      DEC FIXED (5) INIT(0) ;
DCL MARCHZINSEN             DEC FIXED (11,2) ;
DCL MARCHZ_MELDUNG          BIT (1)         INIT('0'B) ;
DCL PARM_FAELL              CHAR    (10)    INIT('') ;
DCL PARM_SICHERST           PIC     '(8)9'  INIT('') ;
DCL PARM_LZPERB             CHAR    (10)    INIT('') ;
DCL PARM_LZPERV             CHAR    (10)    INIT('') ;
DCL PARM_EVERF              CHAR    (10)    INIT('') ;
DCL PARM_STKNOM             DEC     (13,2)          ;
DCL PARM_ZSATZ              DEC     (7,5)           ;
DCL PARM_WAEK               PIC     '999V9999'      ;
DCL PARM_WAEE               PIC     '999'           ;
DCL PARM_USANZ              CHAR    (1)     INIT('') ;
DCL INDEX_GEFUNDEN          BIT (1)         INIT('0'B) ;
DCL I_COUNTER               BIN FIXED (15)  INIT(0) ;
DCL AUSWAHL_PTR PTR ;
DCL CONVERSIONS_FELD_1      CHAR(11) INIT('') ;
DCL CONVERSIONS_FELD_2      CHAR(11) INIT('') ;
DCL CONVERSIONS_FELD_3      CHAR(12) INIT('') ;
DCL KEY_FELD                CHAR(33) INIT('') ;
```

Converted PLI Data Types


```
DCL SUCHKEY         CHAR(04)        INIT('') ;
 DCL LINECOUNTER    PIC '(04)9'     INIT('')   ;       /*CONVERTED*/
 DCL COUNTER        PIC  '(15)9'    INIT('') ;         /*CONVERTED*/
 DCL TAWERT_Z       PIC 'Z.ZZZ.ZZZ.ZZ9V,99' ;
 DCL TAWERT_NEU     CHAR(16) ;
 DCL BSTKNOM        PIC  '(16)9' INIT(0) ;             /*CONVERTED*/
 DCL BNOMSFRS       PIC '(16)9' INIT(0) ;              /*CONVERTED*/
 DCL HFLOAT_1       PIC  '(16)9' INIT(0) ;             /*CONVERTED*/
 DCL HFLOAT_2       PIC  '(16)9' INIT(0) ;             /*CONVERTED*/
 DCL HFLOAT_3       PIC  '(16)9' INIT(0) ;             /*CONVERTED*/
 DCL STEUER_VALUE   PIC '9'  INIT(0) ;
 DCL LOOP_ERROR_COUNT      PIC '(04)9'     INIT(0)   ;   /*CONVERTED*/
 DCL AKTUELLE_AKTSTUFE     CHAR(1) INIT('') ;
 /* OPERATOR FUER DLI-ZUGRIFFE          */
 DCL GLOBAL_OPERATOR CHAR(2) INIT(' =') ;
 /* ZAEHLERFELDER FUER LISTEN_OUTPUT    */
 DCL VALOREN_I      PIC  '(05)9'   INIT(0) ;           /*CONVERTED*/
 DCL VALOREN_O      PIC  '(05)9'   INIT(0) ;           /*CONVERTED*/
 DCL BESTAND_I      PIC  '(05)9'   INIT(0) ;           /*CONVERTED*/
 DCL BESTAND_O      PIC  '(05)9'   INIT(0) ;           /*CONVERTED*/
 DCL TABELLEN_I     PIC  '(05)9'   INIT(0) ;           /*CONVERTED*/
 DCL TABELLEN_O     PIC  '(05)9'   INIT(0) ;           /*CONVERTED*/
 DCL MARCHZINSEN            PIC '(11)9V.(2)9' ;        /*CONVERTED*/
 DCL MARCHZ_MELDUNG         CHAR(1)         INIT('0') ;;
 DCL PARM_FAELL             CHAR    (10)    INIT('') ;
 DCL PARM_SICHERST          PIC     '(8)9'  INIT('') ;
 DCL PARM_LZPERB            CHAR    (10)    INIT('') ;
 DCL PARM_LZPERV            CHAR    (10)    INIT('') ;
 DCL PARM_EVERF             CHAR    (10)    INIT('') ;
 DCL PARM_STKNOM            PIC  '(13)9V.(2)9'       ;  /*CONVERTED*/
 DCL PARM_ZSATZ             PIC  '(07)9V.(5)9'       ;  /*CONVERTED*/
 DCL PARM_WAEK              PIC     '999V9999'       ;
 DCL PARM_WAEE              PIC     '999'           ;
 DCL PARM_USANZ             CHAR    (1)     INIT('') ;
 DCL INDEX_GEFUNDEN         CHAR(1)         INIT('0') ;;
 DCL I_COUNTER              PIC  '(04)9'    INIT(0)   ; /*CONVERTED*/
 DCL AUSWAHL_PTR PTR ;
 DCL CONVERSIONS_FELD_1     CHAR(11) INIT('') ;
 DCL CONVERSIONS_FELD_2     CHAR(11) INIT('') ;
 DCL CONVERSIONS_FELD_3     CHAR(12) INIT('') ;
 DCL KEY_FELD               CHAR(33) INIT('') ;
```

Problems that can come up

- There may be some problem in synchronizing the converted data types in the program with the data types in the data base.
- In the case of data redefinitions, it is necessary to match the length of the redefining data structure to the length of the redefined data structure.
- There may be some loss of performance due to the conversion of the decimal numbers to internal numeric values by the compiler.
- The first two problems can be easily overcome.
- The third problem is not significant.

8 Steps to recovering Web Services from Legacy Systems

1. Discover potential services in the legacy code
2. Evaluate them for Reusability
3. Strip away the Code not needed
4. Extract the Code needed
5. **Convert the data types in the code to be wrapped**
6. Wrap the Code behind a WSDL interface
7. Generate wrapper code to transfer the data types in the service request to parameters of the wrapped code.
8. Generate wrapper code to transfer the results of the wrapped code to data types in the service response.

Generated Web Service Interface Definition

```
<XSDCOB:complexType type = "#params" name = "PSAB001V-PARAMS"
            content = "eltOnly" model = "closed" level = "02"
            occurs = "ONEORMORE" minOccurs = "0001" maxOccurs = "unbounded">
     <XSDCOB:complexType type = "#group" name = "WBER"
             content = "eltOnly" model = "closed" level = "02"
             occurs = "ONEORMORE" minOccurs = "0001" maxOccurs = "0001">
       <XSDCOB:element type = "#char" name = "WRATE"
                content = "TextOnly" model = "closed" level = "03"
                occurs = "ONEORMORE" minOccurs = "0001" maxOccurs = "0001"
                pos = "0007" lng = "0004"
                pic = "X(4)" usage = "DISPLAY"/>
       <XSDCOB:element type = "#char" name = "WADATE"
                content = "TextOnly" model = "closed" level = "03"
                occurs = "ONEORMORE" minOccurs = "0001" maxOccurs = "0001"
                pos = "0011" lng = "0008"
                pic = "X(8)" usage = "DISPLAY"/>
       <XSDCOB:element type = "#dec" name = "WLFNR"
                content = "TextOnly" model = "closed" level = "03"
                occurs = "ONEORMORE" minOccurs = "0001" maxOccurs = "0001"
                pos = "0019" lng = "0006
                pic = "X(6)" usage = "DISPLAY"/>
       <XSDCOB:element type = "#dec" name = "WHTNR"
                content = "TextOnly" model = "closed" level = "03"
                occurs = "ONEORMORE" minOccurs = "0001" maxOccurs = "0001"
                pos = "0025" lng = "0007"
                pic = "X(7)" usage = "DISPLAY"/>
     </XSDCOB:complexType>

 <message name="getPSAB001VResponse">
     <part name="getPSAB001VResponse" name = "PSAB001V-PARAMS"
           type="types:PSAB001VResponseTypes" />
 </message>

 <portType name="RSFServiceTemplatePort">
     <operation name=„PSAB001V">
           <input message = "PSAB001VRequest"/>
           <output message= „PSAB001VResponse"/>
     </operation>
```

Open Questions and Challenges

- There can be no real technical objections to this approach. It cannot be challenged.
- The first open question is “Do users really want to move to SOA?”
- The second open question is “If users move to SOA, do they want to take their legacy code with them?”.
- If these two questions can be answered positively, then there is a need for this approach.