Generazione di codice in Python

- Dal documento al codice C++ attraverso la modellizzazzione UML
  - ovvero
- Come usare python per fare un lavoro noioso e riuscire a guadagarsi la pagnotta
  - ovvero
- Come usare python anche se ti tocca scrivere codice in altri linguaggi meno divertenti

SSI
Sistemi Software Integrati
Una Società Finmeccanica

Il problema da risolvere

- L'applicazione C++ da sviluppare ha la necessità di comunicare sia con altri componenti dello stesso sistema software che con dispositivi controllati da computer
  - La comunicazione avviene scambiando pacchetti dati (messaggi) su canali UDP/IP
    - Più di 500 tipi di messaggi
    - Occorre usare necessariamente le API del middleware fornito dal cliente

Il problema da risolvere

- I campi di un messaggio possono essere:
  - Campi semplici (interi, floating point, booleani, stringhe, valori discreti)
    - Per ogni campo sono opzionalmente definiti attributi come minimi, massimi, LSB
  - Strutture
    - Fisse o varianti (union)
  - Array
    - A dimensione fissa o variabile

Il problema da risolvere

- Per ogni messaggio, è necessario scrivere il codice per:
  - Definire le strutture del messaggio
  - Definire enumerativi per i campi a valori discreti
  - Serializzazione del messaggio (packing)
  - Deserializzazione del messaggio (unpacking)
  - Verificare che i messaggi ricevuti siano validi:
    - Controlli di minimo e massimo se definiti
    - Controllare che i campi discreti abbiano un valore incluso tra i valori ammessi

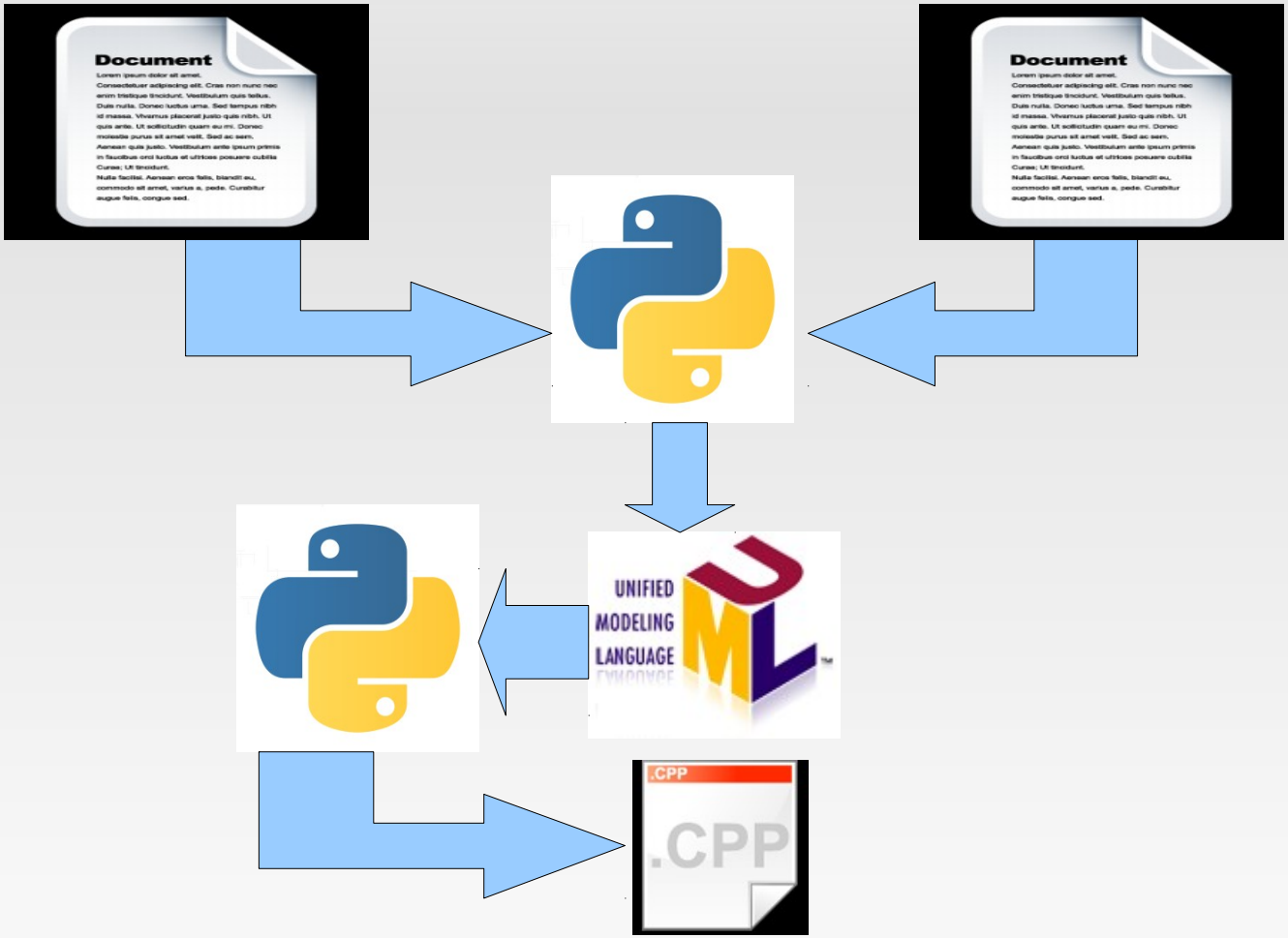
L'idea iniziale

Scrivere un programma in Python che – leggendo le specifiche dei messaggi – generi automaticamente il codice di packing, unpacking e verifica degli stessi

Ma quali specifiche ?

- Diversi tipi di formato
  - I messaggi per comunicare con i dispositivi esterni sono definiti in documenti generati automaticamente da un tool non noto e non disponibile
  - I messaggi per comunicare con componenti sviluppati internamente sono definiti come classi all'interno del CASE tool aziendale
  - I messaggi per comunicare con componenti software sviluppati esternamente sono specificati in documenti a loro volta generati dal CASE tool (ma a noi arrivano solo i documenti)

L'idea finale



Passo 1 : Dal documento al modello

- Un esempio di specifica della struttura

Nome Dato	Pos.	Size	Molt.
Header	0	16	1
Time of validity	16	8	1
Seconds	16	4	1
Microseconds	20	4	1
Ship position	24	36	1
Latitude	24	4	1
Longitude	28	4	1
Log speed	32	4	1
Course made good	36	4	1
Speed over ground	40	4	1
Set	44	4	1
Drift	48	4	1
Latitude Accuracy	52	4	1
Longitude Accuracy	56	4	1
Ship attitude	60	36	1
Heading	60	4	1
Relative roll	64	4	1
Absolute pitch	68	4	1
Heading rate	72	4	1
Relative Roll rate	76	4	1
Absolute Pitch rate	80	4	1
North velocity	84	4	1
East velocity	88	4	1
Vertical velocity	92	4	1
Ship heave	96	4	1
Water depth	100	4	1
Attitude and velocities validity	104	2	1
Heading validity	106	2	1
Course and Speed over ground validity	108	2	1

Passo 1 : Dal documento al modello

- Un esempio di specifica dei campi

Header
Descrizione Testata comune per tutti i messaggi.

Common Header

Time of validity
Size 8
Tipo CombatSystemTime

Struttura dati così composta

Seconds
Descrizione Numero di secondi trascorsi dalle ore 00:00 del 01/01/1970.

Number of second since h. 00:00 01 Jan 1970.
Size 4
Tipo LongUnsignedInteger
Intervallo 0, 2**31-1,
Unità di Misura sec

Microseconds
Descrizione Numero di microsecondi trascorsi a partire dal tempo riportato nel campo precedente "Seconds".

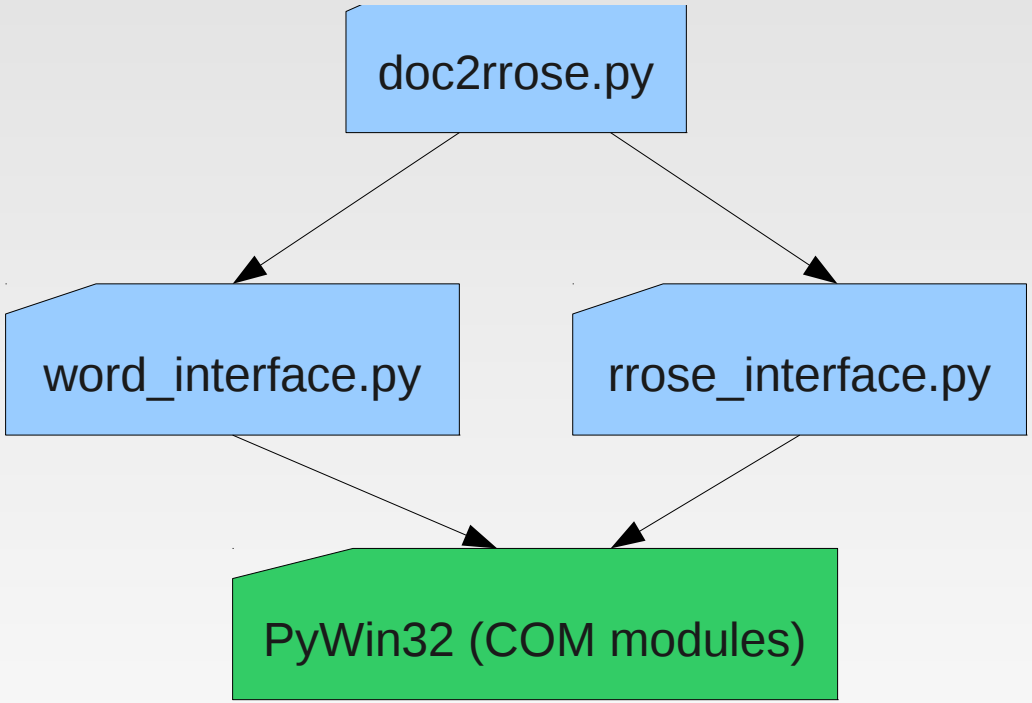
Number of microseconds since the time reported in the field "Seconds".
Size 4
Tipo LongUnsignedInteger
Intervallo 0, 10**6-1,

Passo 1 : Dal documento al modello

- Il risultato voluto:
  - Definire all'interno del modello una classe per ogni messaggio, i cui attributi sono i campi del messaggio
    - Includere le parti descrittive del messaggio come documentazione
    - Includere negli attributi dei campi dati relativi a dimensione, minimi e massimi, LSB
  - Definire classi addizionali per:
    - Strutture dati interne
    - Enumerativi

Passo 1 : Dal documento al modello

- Struttura del programma python



Passo 1 : Dal documento al modello

- word_interface.py : una interfaccia COM verso Microsoft Word ™
  - Può leggere/scrivere i seguenti elementi in un documento word:
    - Paragrafi (in uno stile predefinito)
    - Tabelle verticali semplici
      - La prima riga è l'intesazione
      - Solo testo all'interno di ogni casella

Passo 1 : Dal documento al modello

- Esempio di uso di word_interface.py

```
def test_1():
    w = WordInterface();
    w.set_visible(True)
    w.new_document()

    w.insert_paragraph( "ciao Moddo :-)",  stylename="Heading 1")
    w.insert_paragraph( "" )
    w.insert_paragraph("hello word :-/")
    w.insert_paragraph( "This is in bold and Italic", bold=True, italic=True)
    w.insert_paragraph( "This is underlined.", underline = True )
    w.insert_paragraph( "" );

    w.insert_vertical_table( "This is a table with default properties and predefined style",
                        ("First Column", "Second Column", "Third Column"),
                        style = "IDDTableStyle",
                rows=[("1", "2", "3"),
                     ("11", "22", "33"),
                     ("aa", "bb", "CCccCC")],
                )

    w.set_table_properties(header_font=Fontproperties(name=None, italic=True,
bold=True),
                        #header_color=33,
                        borders=False,
                        column_sizes = [0.2, 0.2, 0.6])
    w.insert_vertical_table( "This is a table with specific properties and the same style",
                         ("First Column", "Second Column", "Third Column"),
                [("1", "2", "3"),
                ("11", "22", "33"),
                ("aa", "bb", "CCccCC----------------------------------")],
                style = "IDDTableSTyle"
                )

   w.save_current()
   w.quit()
```

```
def test_tables_1():
   w = WordInterface()
   w.open_document(r'../../test/prova_word_interface.doc')
   for t in w.get_next_table():
     print t
   w.replace_table_element(table_idx=0, row_idx=1, col_idx=2, new_text='Paperino')
   w.set_visible(True)
```

```
def test_tables_2():
    w = WordInterface()
    w.open_document(r'../../test/prova_word_interface.doc')
    w.search_replace_rows(
              ( ( None, None, u'float', None, None, u'tbd', u'tbd' ), # pattern
                ( None, None, None,     None, None, u'N/A', u'N/A' ) # replacement
               ), # rule end
                ( ( None, None, u'boolean', None,   None,        u'tbd', u'tbd' ), # pattern
                   None, None, None,       u'N/A', u'True/False', u'N/A', u'N/A' ) # replacement
                ) # rule end
    )
    w.set_visible(True)
```

```
def test_read_paragraphs():
   w = WordInterface()
    w.open_document(r'../../test/prova_word_interface.doc')
   for style, paragraph in w.get_next_paragraph():
        print style
        print paragraph
```

Passo 1 : Dal documento al modello

- rrose_interface.py : una interfaccia COM verso il CASE tool RationalRose ™
  - Può leggere e scrivere all'interno di un modello esistente:
    - Categorie
      - Specifiche di classe
        - Specifiche di metodo
        - Specifiche di attributo
        - Dati associati a classi o attributi definiti dall'utente (tools)

Passo 1 : Dal documento al modello

- Esempio di uso di rrose_interface.py

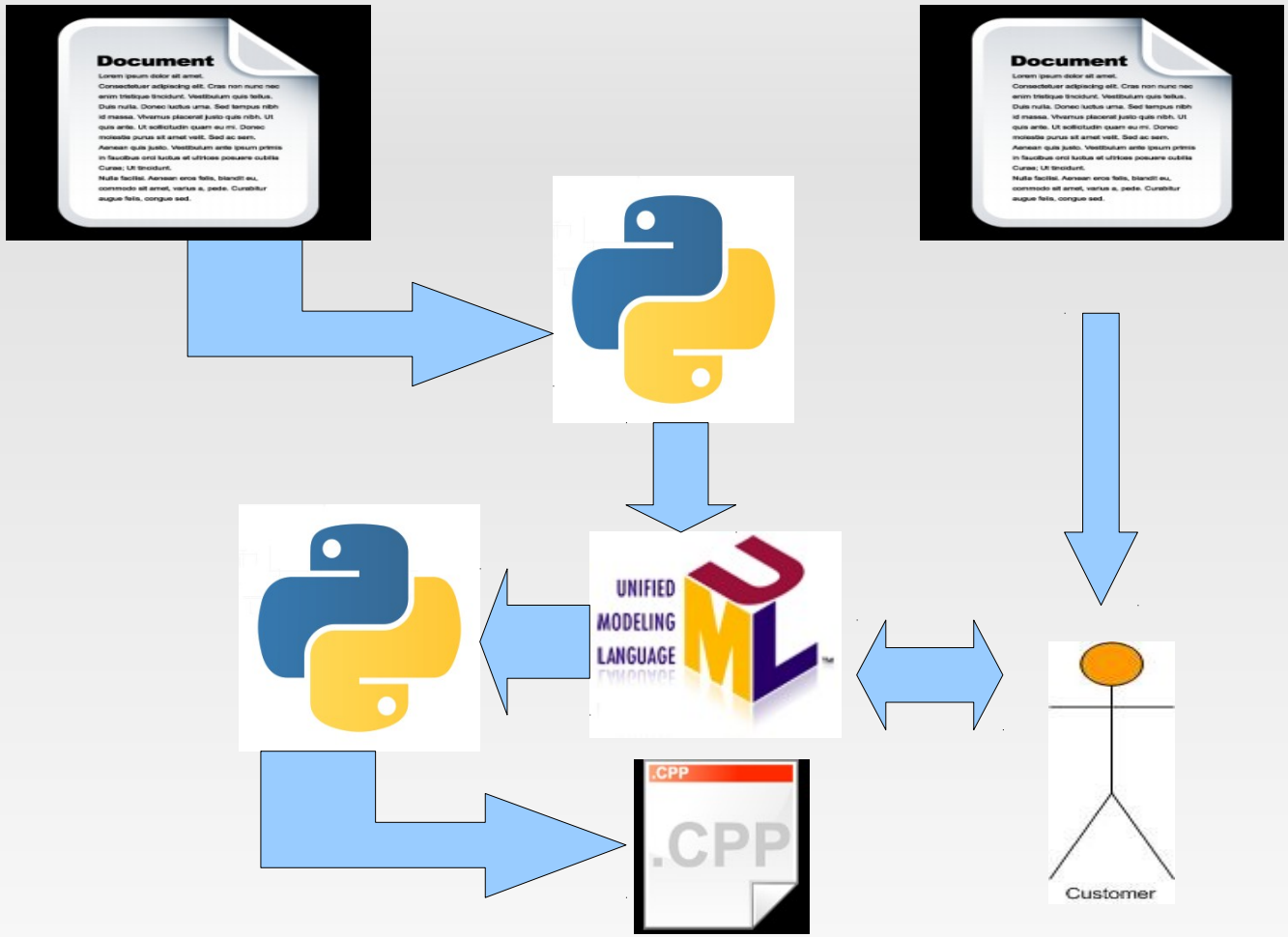
```
def test():
    TRACE.active = True
    rr_if = RRoseInterface()
    mname = rr_if.current_model.Name

    rr_if.load_model( os.path.join( PYUT_ROOT, "test", "prova.mdl" ) )
    mname = rr_if.current_model.Name

    rr_if.set_current_category( "Use Case View", "Capabilities" )
    rr_if.set_current_category( "Logical View", "Analysis Model", "Logging" )
   TRACE( 'Current category is now: %s ' % rr_if.current_category.Name )

    rr_if.set_current_category( "Logical View" )
    TRACE( 'Current category is now: %s ' % rr_if.current_category.Name )

    rr_if.add_category( "Prova" )
    TRACE( 'Current category is now: %s ' % rr_if.current_category.Name )

    rr_if.add_class("Attore1", stereotype='Actor', docum = 'Esempio di attore.' )
    rr_if.add_class("Attore1", stereotype='Actor', docum = 'Esempio di attore.' )
    rr_if.add_class("Boundary1", stereotype='boundary',
                        docum = 'Esempio di boundary class.',
                attributes = ( ('uno', 'integer', '12', 'attributo uno'),
                               ('due', 'boolean', 'False', 'attributo due'),
                               ),
                methods = ( ( 'metodo1', 'boolean', (), 'Metodo senza parametri' ),
                            ( 'metodo2',
                                'boolean',
                                ( ('a', 'int', '-10', 'Parameto a'), ),
                                'Metodo con parametri' )
                            )
                )
```

```
    TRACE( "*** ITERATE CATEGOTIES without a root" )
    for c, cpath in rr_if.iterate_categories():
            TRACE( 'Iterating categories:', cpath, c.Name  )
            TRACE( 'Current category is now: %s ' % rr_if.current_category.Name )

   TRACE( "*** ITERATE CATEGOTIES with a given root" )
    for c, cpath in rr_if.iterate_categories( "Logical View" ):
            TRACE( 'Iterating categories:', cpath, c.Name  )
            TRACE( 'Current category is now: %s ' % rr_if.current_category.Name )

    rr_if.set_current_category( "Prova" )
    classes = rr_if.get_current_category_classes(stereotypes=('boundary'))
    if classes:
            r_class = classes [0]
            rr_if.set_current_category( "Logical View" )
            TRACE( "Copying the class ", r_class.Name, "in Logical View" )
            rr_if.copy_class( r_class )
            rr_if.set_current_category( "Prova" )
            TRACE( "deleting the class", r_class.Name, "from Prova" )
            rr_if.delete_class( r_class )
    else:
        TRACE( "Cancel and recreate class test failed: not class found" )

    TRACE( "Per finire, la lista di tutte le classi nel modello sotto Logical View" )
    for c in rr_if.iterate_on_classes( ("Logical View",) ):
            print "\t", c.Name, c.Stereotype

    rr_if.save_current_model()
    rr_if.exit()
```

Passo 1 : Dal documento al modello

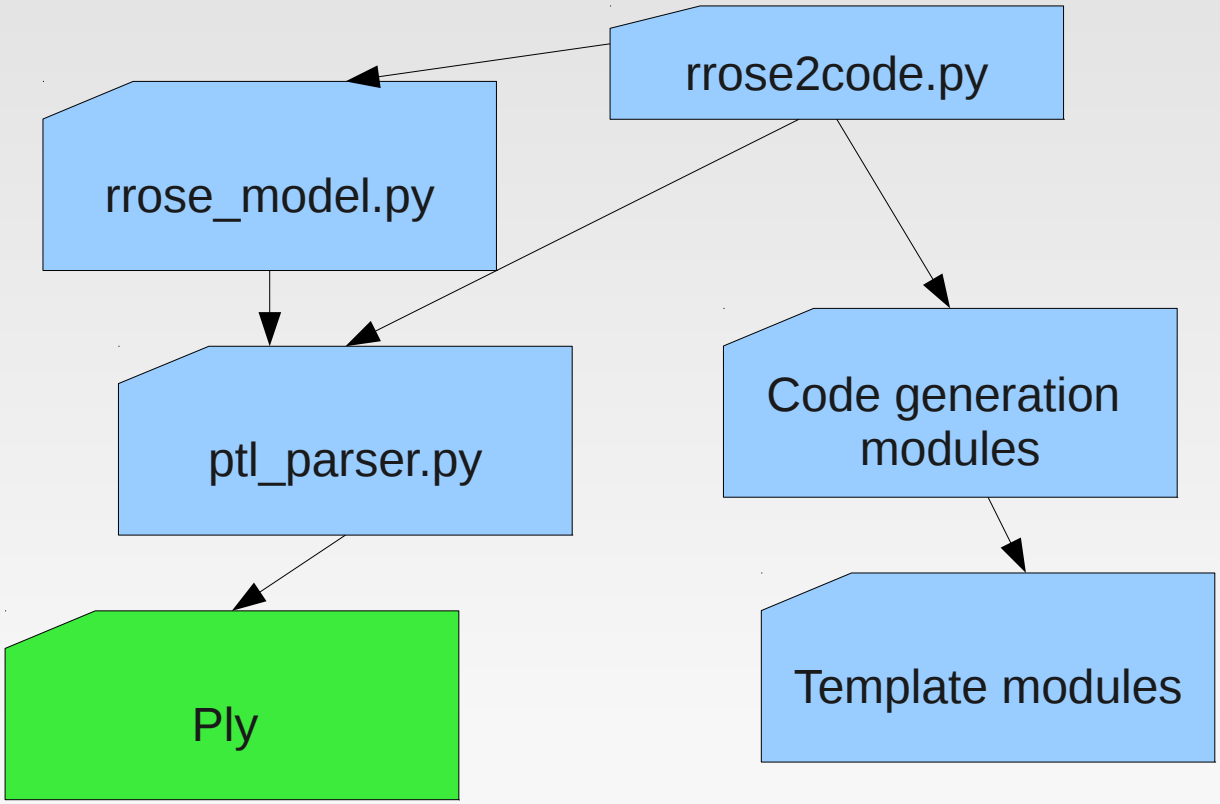
- Quello che ha funzionato
  - Il programma riesce a generare un modello con tutte le informazioni necessarie per la generazione di codice
- Quello che ha funzionato meno
  - I nomi dei tipi – derivati da quelli dei campi – a volte non sono molto signigficativi
    - Esempio: tipi *T_Track* e *T_Track1* per campi contenenti informazioni diverse ma chiamati entrabi *track*
- Cose non implementate:
  - Messaggi con struttura variabile (union)

L'idea rivisitata



Passo 2: Dal modello al codice

- Struttura del programma



Passo 2: Dal modello al codice

- I dati sulle classi sono estratte direttamente dal file del modello, che è un file ASCII con formato *petal*
  - Più veloce che usare l'interfaccia COM
  - Funziona anche sulla piattaforma target (Linux)
  - Non richiede istallazione o licenza del CASE tool

Passo 2 : Dal modello al codice

- Esempio di file com formato petal
  - In pratica un set di liste nidificate

```
(object Class "CMS_SCM_Alert"
    attributes (list Attribute_Set
       (object Attribute
           tool          "Additional Message Data"
           name          "IRS Requirement"
           value         "IRS-UWPSMF-DID-<number>"))
    quid          "496F6BE700D8"
    documentation
|Messaggio di notifica di un alert ad SCM.
|TBC : controllare il formato con IDD di SCM

    stereotype          "MESSAGE"
    superclasses        (list inheritance_relationship_list
      (object Inheritance_Relationship
           quid          "49EC376000AA"
           supplier      "Logical View::Design Model::UWP Common Software::UWP Common Utilities::Utilities::UWPMessage"
           quidu         "49E898BA03B6"))
    class_attributes          (list class_attribute_list
      (object ClassAttribute "header"
           attributes (list Attribute_Set
              (object Attribute
                   tool          "Level D Definition"
                   name          "Size (Bytes)"
                   value         (value Text "16")))
           quid          "49ABFFC50249"
           documentation "Header standard dei messaggi COM."
           type          "T_MessageHeader"
           quidu         "4051CAE502BF"
           exportControl "Public")
```

Passo 2 : Dal modello al codice

- PLY : una implementazione in python dei classici tool lex e yacc
  - Le regole lessicali sono *regular expressions*
  - Le regole sintattiche – in stile yacc – sono contenute nelle stringhe di documentazione delle funzioni di parsing corrispondenti
  - Per informazioni:
    - http://www.dabeaz.com/ply/

Passo 2: Dal modello al codice

- Esempi di funzioni di parsing:

```
def p_list_sequence(p):
    """
    list_sequence : list_sequence list
                  | list
    """
    if len(p) == 2:
        p[0] = [p[1],]
    else:
        p[0] = p[1]; p[0].append(p[2])


def p_list_as_object(p):
    """
    list : LPAREN OBJECT IDENT QUOTED REFERENCE field_list RPAREN
    """
    TRACE( 'p_list_as_object', p[3] )
    p[0] = (p[2], p[3], p[4], p[5], p[6])


def p_list_as_object_object_noreference(p):
    """
    list : LPAREN OBJECT IDENT QUOTED field_list RPAREN
    """
    TRACE( 'p_list_as_object_noreference', p[3] )
    p[0] = (p[2], p[3], p[4], None, p[5])

def p_list_as_object_noname(p):
    """
    list : LPAREN OBJECT IDENT REFERENCE field_list RPAREN
    """
    TRACE( 'p_list_as_object_noname', p[3] )
    p[0] = (p[2], p[3], None, p[4], p[5])

def p_list_as_object_noname_noreference(p):
    """
```

Passo 2: Dal modello al codice

- ptl_parser.py :
  - Analizza il file contenente il modello
  - Crea un *parse tree* che corrisponde ai dati estratti dal modello
- rrose_model.py :
  - Estrae dal *parse tree* le informazioni necessarie
  - Crea un grafo di istanze di classi corrisponenti ai dati estratti dal modello

Passo 2: Dal modello al codice

- Generazione di codice : le funzioni di formattazione della libreria standard di python sono piu che sufficienti:
  - I template sono stringhe di formattazione python, che includono le parti variabili come elementi di tipo %(varname).

Passo 2: Dal modello al codice

- Esempi di template

```
TEMPLATE_UNION_DECLARATION = """
/*
 %(documentation)s
*/
typedef struct {

  %(discriminant_declaration)s

  union  {
    %(field_list)s
 } u;

} %(union_name)s;
"""


TEMPLATE_BITMASK_DECLARATION = """
/*
 %(documentation)s
*/
typedef union {
   %(full_type)s full_value; /* the integer value of the bitmask */
   struct {
        %(field_list)s
   } bits;
}  %(bitmask_name)s;
"""
```

```
TEMPLATE_CLASS_HEADER = """


#ifndef _INCLUDED_%(capitalized_class_name)s
#define _INCLUDED_%(capitalized_class_name)s

%(include_statements)s

/*
 %(documentation)s
*/
class  %(class_name)s : public %(superclass_list)s
{
  public:
%(public_attributes)s
%(public_methods)s

  protected:
%(protected_attributes)s
%(protected_methods)s

  private:
%(private_attributes)s
%(private_methods)s

};


#endif
"""
```

Passo 2: Dal modello al codice

- Il risultato finale

```
unsigned  ClasseMessaggio::PackFields( unsigned char *buffer )
 {
      unsigned nbytes=0;
      nbytes +=COM::COMUtility::msgPutInt16 ( buffer+nbytes, field1 );
      nbytes +=COM::COMUtility::msgPutUInt16 ( buffer+nbytes, field_u1 );
      nbytes +=COM::COMUtility::msgPutFloat ( buffer+nbytes, field2 );

      T_UnsignedInteger16 temp_field3;
       temp_field3 = field3;
      nbytes +=COM::COMUtility::msgPutUInt16 ( buffer+nbytes, temp_field3 );


      nbytes +=  COM::COMUtility::msgPutString( buffer+nbytes, field_s, 12 );


      // Array packing loop
      for ( unsigned idx =0; idx < field1; idx ++ )
      {
        E_SomeValues field_a_elem;
        field_a_elem = this->field_a[idx] ;

            T_UnsignedInteger16 temp_field_a_elem;
             temp_field_a_elem = field_a_elem;
            nbytes +=COM::COMUtility::msgPutUInt16 ( buffer+nbytes, temp_field_a_elem );

      }

      nbytes +=COM::COMUtility::msgPutInt16 ( buffer+nbytes, field_struct.f1 );
            nbytes +=COM::COMUtility::msgPutFloat ( buffer+nbytes, field_struct.f2 );

      T_UnsignedInteger16 temp_field_struct_f3;
       temp_field_struct_f3 = field_struct.f3;
      nbytes +=COM::COMUtility::msgPutUInt16 ( buffer+nbytes, temp_field_struct_f3 );
```

Passo 2: Conlusioni

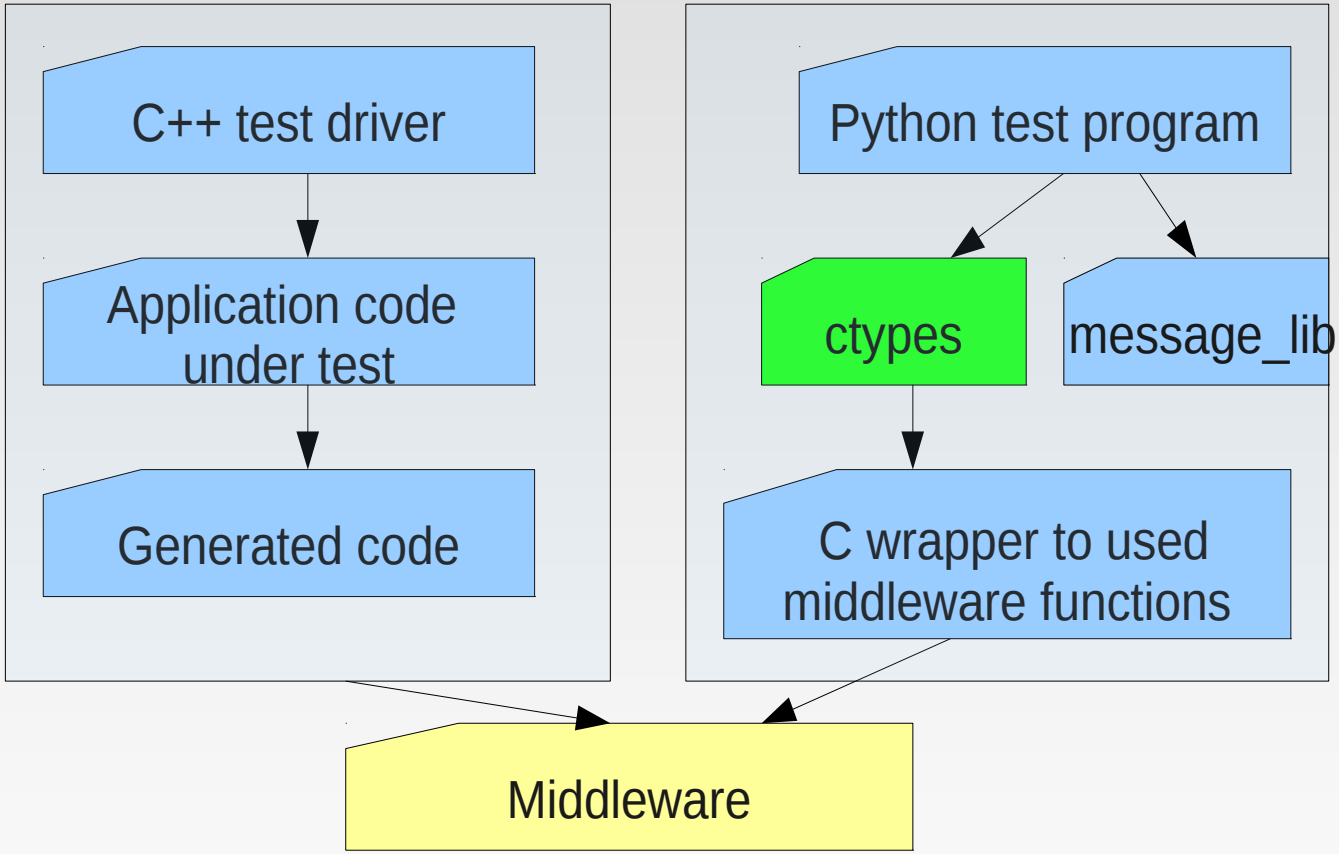
- Funziona :->
- Possibili miglioramenti:
  - Generazione di costanti, ad esempio per la dimensione degli array
    - Così che si possano usare senza duplicazioni in altre parti del codice
  - Supporto per tipi esterni:
    - I tipi dovrebbero essere usati per packing/unpacking ma non definiti all'interno del codice generato
  - Rendere il codice meno dipendende dalle specifiche del progetto

Come testare il codice generato?

- Il generatore di codice genera anche dei semplici test:
  - test_pack : da un insieme casuale di valori dei campi al corrispondente array di byte
  - test_unpack : da un array casuale di byte ai corrispondenti valori dei campi
  - test_check : TBD
- Richiede una analisi manuale dei risultati

Come testare il codice generato?

- Test aggiuntivi sono stati fatti al momento di usare il codice generato:



Come testare il codice generato?

- message_lib : un modulo per serializzare/deserializzare i messaggi in python
  - Legge la descrizione dei messaggi da un file ASCII (formato .ini – like)
    - L'idea era di usare ConfigParser, ma non mantiene l'ordine dei campi
  - Dalla descrizione dei messaggi genera una stringa formato utilizzabile con *struct*
    - La stringa è generata dinamicamente per quei campi la cui struttura dipende dal valore di altri campi (union e array a dimensione variabile)

Come testare il codice generato?

- Esempio di uso di message_lib

```
def test_message_def( do_print=False):
    fname =  "test_message_def.txt"
    message_list = MessageDefinition.read_config( fname )
    if do_print:
        for m in message_list:
            print m
    return message_list

def test_pack_unpack():
    message_list = test_message_def()
    for msg in message_list:
        init_dict = dict()
        try:
            in_values = dict()
            for f in  msg.fields:
                in_values[f.name] =  f.get_random_value(100, in_values)
            bytes =  msg.pack( **in_values )
            out_values =  msg.unpack(bytes)
            for in_name, in_value in in_values.items():
                out_value = out_values[in_name]
                err_str = "Message %s: Value mismatch in field %s (\n\texpected:%s \n\tfound: %s)" % (
                        msg.name, in_name, str(in_value), str(out_value)
                    )
                assert almost_equal(in_value, out_values[in_name]),  err_str
        except:
            print "Errore in packing/unpacking la struttura %s" % msg.name
            err, det,  tb = sys.exc_info()
            print err, det
            traceback.print_tb(tb)
```

```
def test_mas_cs_full_status():
    message_list = test_message_def()
    message_dict = dict( (m.name, m) for m in message_list )
    message_def = message_dict['MAS_CS_full_status_INS']
    subheader_dict = dict( mas_state = 1,
                           operational_mode = 2,
                           function = 3,
                           training_type = 0,
                           message_counter = 10000 )
    bit_results_dict = dict(
        power_supply_test = 10,
        temperature = 20,
        fec_flooding = 30,
        communication = 40,
        tx_channels = [i%2 for i in xrange(24)],
        rx_channels = [(i+1)%2 for i in xrange(60) ]
        )
    bytes = message_def.pack(
        complementary_header_data = subheader_dict,
        bit_results  = bit_results_dict,
        spare_1        = 4 )
```

Domande ?

?