A Classification Model for Reusable Software Components

Bernard Durin and Eric Rames

MATRA Espace, 31077 Toulouse Cedex, France

1. INTRODUCTION

This paper presents work which has been carried out in the ESF-ROSE project (referred to as ROSE in the remainder of this paper). Funded under the Eureka programme (Eureka is the famous interjection used by Archimedes in his bath and not an acronym), the ESF project (Eureka Software Factory) aims at providing a highly effective software manufacturing environment. The use of the word *factory* in the software context has very little connection to traditional assembly-line factories. Instead, a Software Factory is a factory in the modem sense providing Computer Integrated Software Manufacturing with emphasis on integration. A software factory covers the total software production process, including all technical and managerial tasks, with a high degree of automation and resource utilization.

The ROSE project is a collaborative effort involving MATRA Espace, the software house Sema Group (France), and the University of Dortmund (Federal Republic of Germany). Part of the work on classification is the Ph.D. research of author Eric Rames.

The main goals of the ROSE project (Reuse Of SoftwarE) are:

- to analyze and to define in a comprehensive way the concept of software reuse;
- to develop an environment for the reuse of software components within a factory.

A precondition for reuse in software development is the existence of libraries of reusable software components. In order to support reuse, the collection must contain not only the components themselves, but also be accessible by a system that provides descriptions of the components and retrieval mechanisms so that users may match their specific requirements against these descriptions. Indexing reusable software components according to a classification scheme allows reusers to have a better understanding and more efficient access to the libraries' contents. Therefore a classification scheme is built so that it represents selection criteria the reuser might have. These indexes would be searchable and keyed to retrievable software descriptions. Based on retrievals, users may then access the actual software.

Building such a collection is a domain analysis process [PRI90] that includes activities such as:

- Identification of software components that should be reusable and description in terms of reusable software components.
- Definition of a classification scheme appropriate for indexing and retrieving the reusable software components.

This paper focuses on the latter topic and how it is performed in the ROSE project. A case study carried out from the aerospace domain is then presented. A discussion of ongoing and future work will conclude this paper.

ISSN: 2324-9773

Durin, B., & Rames, E. (1990). A classification model for reusable software components. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 41-58. doi: 10.7152/acro.v1i1.12463

2. CLASSIFICATION MODEL

In the classification activity for building reusable software components libraries, we are using a classification scheme consisting of concepts that can be used for indexing and retrieving software components. The main feature of the classification model in the ROSE project for building such a scheme is its genericity in that it can be used for software systems written for various application areas (e.g., engineering, chemistry, aerospace, economics, medicine, etc.).

According to this classification model a classification scheme is composed of facets and classes as follows:

- **Facets** are used for arranging top-level classes of the classification scheme [PF87]. They represent discrete viewpoints of reusable software components. Taken together, these viewpoints serve to fully characterize a particular component. We also refer to these viewpoints as *classification criteria*. This is a useful expression as an alternative to since, in our application, the facets express well-known criteria for software selection. There is no relationship between facets except the fact that they compose one classification scheme.

  Examples of facets that may be defined are "system function" (the functional viewpoint of the system in which the reusable software component is embedded), "application area" (viewpoint of the use of the reusable software component), "function" (the requirement viewpoint of the elementary actions realized by the reusable software component), "object" (the design viewpoint on the entities manipulated in the reusable sofwtare component).

- **Class**: Each facet is composed of a set of classes that represent concepts of the domain defined by this facet. Examples of classes of the facet "object" are "telemetry data", "conversion tables", "communication message" ... Properties of a class are described by two types of attribute:

  — **Link Attribute**: Classes in a facet are connected by Link Attributes. A Link Attribute of a class refers to another class and is used first to describe a relationship between classes, second to control strategies of indexing and retrieving reusable components. Link Attributes such as "subclasses" or "superclass" define the generalization/specialization hierarchy of classes (from general classes to specific ones) whereas user-defined (user here is the library administrator) Link Attributes such as "is composed of" or "see also" define other semantic relationships between classes. There is no Link Attribute attached to a class that refers to a class belonging to a different facet (i.e., no inter-facet link).

  — **Component Attribute**: These are the common properties of a class that are used for verifying that a reusable software component belongs to that class (i.e., that the class should be assigned as an indexing term to this component). Component Attributes are inherited if the class is connected to another class with a specialization Attribute Link. Up to now Component Attributes values are simply terms (rather than other classes). Examples of Component Attributes of the class "communication message" are "message format", "communication protocol" ...

ISSN: 2324-9773

Durin, B., & Rames, E. (1990). A classification model for reusable software components. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 41-58. doi: 10.7152/acro.v1i1.12463

2.1 INDEXING REUSABLE COMPONENTS

At least one class from each facet is assigned as an indexing term to each reusable component. Pattern-matching algorithms have been defined in order to perform these assignments. We use Component Attributes values (terms) as patterns to determine membership of a given reusable software component to a class. Pattern matching succeeds if Component Attribute values are in a prior description of the reusable software component being indexed. A description would include names of files, packages, and modules associated with the software component (see next section for further discussion of this).

Link Attributes are used to control exploration of classes within a facet. For example, if pattern matching failed for the current class, classes linked by specialization Link Attributes are not explored but there is backtracking (towards more general classes) or matching with classes connected by other Link Attributes ("see-also", "is composed of" ...).

Indexing reusable software components is performed within the ROSE environment:

- interactively, i.e., the graphical representation of the classification scheme is display to users for selecting appropriate classes for indexing;
- automatically, i.e., from a user request the system proposes the best solutions found for indexing.

2.2 RETRIEVING REUSABLE COMPONENTS

Reusable software components are retrieved thanks to the classification scheme. Basic mechanisms (pattern-matching) are the same for retrieving as for indexing. A user's request is analyzed to determine if some Component Attribute values of the current class (terms) are present in the request. All the facets are scanned. The intersection of classes for which pattern-matching succeeded contains the reusable software components required by the user. In case the resulting set of reusable software components is too large, additional filtering operations would be performed.

3. DOMAIN ANALYSIS

As stated earlier, building a classification scheme is a domain analysis process as defined in [PRI90], i.e., "a process where information used in developing reusable software systems is identified, captured, structured and organized for future reuse".

Domain refers to the specific application area within which the software was written. We would expect software for a specific application to have characteristic structures and functionalities. For instance, in aerospace software systems we found:

- common structures such as "telecommand , telemetry source, parameters ..."
- common functionalities such as "enable the generation of telecommand, disable the generation of telecommand, add parameters ..."

ISSN: 2324-9773

Durin, B., & Rames, E. (1990). A classification model for reusable software components. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 41-58. doi: 10.7152/acro.v1i1.12463

The domain analysis process enables us to identify and to describe not only potentially reusable software components, but also to build a classification scheme. Sources of information (input of the process) consist of knowledge of application area specialists, software systems produced, associated documentation (specification documents, test documents ...), methods and tools in use, standards ...

Reusable software component descriptions (mentioned in the preceding section in connection with indexing) are identified as part of the domain analysis process. These descriptions are produced according to a reusable software component model that provides an appropriate framework for describing the structure of reusable software components in a way that enable their reuse. Although these descriptions are used in the domain analysis process, building this model and constructing descriptions accordingly are outside our definition of the domain analysis process. Nevertheless, we give a rough description of what could be such a description:

```
Reusable software component TELEMETRY GENERATION is
View INFORMAL SPECIFICATION: <list of text files>,
View FORMAL SPECIFICATION: <list of SADT-like files>,
View DESIGN: <list of HOOD-like files>,
View INTERFACE: <list of ADA-like interface packages>,
View BODY: <list of program modules>,
end TELEMETRY GENERATION
```

Assuming that a set of reusable software components has been identified, making a classification scheme consists of the following steps:

- Define the scope of the application area i.e. domain (boundaries).
- Define a priori a set of classification criteria (facets).
- Determine possible classes for each facet by identifying common structures and functionalities across software systems in the domain.
- Identify hierarchical and/or other semantic relationships between classes.

This scheme now yields a standard vocabulary for users to look up reusable software components in the collection and for a library administrator to index new components.

4. CASE STUDY

Our initial objective was to verify the classification model proposed within the ROSE project by building a concrete classification system. For our study, we chose software systems for an application domain that is part of the aerospace domain: ground support software systems for unmanned spacecraft (satellite tests benches). There were three systems. A system ranged in size from 50,000 to 320,000 lines of Fortran source code. Systems took from 15 to 100 man-months to develop over a period of 1 to 4 years. There were about 2000 modules in each system. Modules include subroutines, functions, main programs, and blocks of data in these systems.

Two of these software systems coming from different satellite families were selected and analyzed. The domain analysis was performed using the following information sources:

ISSN: 2324-9773

specification and design documents accompanying the software systems, notes gathered from conversations with domain specialists, etc. Comparison of these systems enabled us to emphasize several common structures and functionalities with a high reusability level. Examples of structures and functionalities for this domain were given in the explanation of domain analysis in the preceding section.

This domain analysis was conducted manually. The resulting classification scheme was made of four facets: "function", "entity", "functional area" and "application domain". These facets ranged in size from 20 to 300 classes. 450 reusable components have been identified and then indexed using the scheme. Indexing a component consisted of scanning manually the informal specification view of reusable software component descriptions (see this view in the sample description in the preceding section), and selecting a combination of classes, one from each classification criterion (facet), that best represents the component's characteristics. See the appendix to this paper for an example of this scheme.

Tools have been developed and used to edit the resulting classification scheme, to check its consistency, and to index the software components. As new components enter the collection, new facets may be defined and new classes added to the scheme (evolutive notion). One of these tools is a mock-up written in Le-Lisp and Aida (Le-Lisp and Aida are trademarks of Ilog) and is running under X11 windows. Another tool has been built upon a commercial expert system called Classic (Classic is a trademark of Ilog). These experiences have demonstrated that a faceted classification is easily tailorable to a particular application domain for a software system, taking into account specific terminology and perspectives of an application domain.

5. ONGOING WORK

We are currently working on indexing and retrieving tools to be integrated into the ROSE environment. Once the domain analysis has been performed, and a classification scheme established, these tools would provide automatic assistance in indexing and retrieving a reusable component by searching the scheme for the best class. The strategy is based on analysis of prior component descriptions (for indexing) or user queries (for retrieving) to produce class templates, and on the evaluation of the degree of likeness between these templates and class patterns in the scheme. In our experiments, we applied word-matching procedures using the informal specification view in the reusable software component description.

A major drawback in classification is the overhead cost in constructing and maintaining a scheme. As future work, we are investigating techniques from the area of knowledge acquisition in order to develop a tool which should assist in building a classification scheme for the purpose of reuse.

6. REFERENCES

[PF87] R. Prieto-Diaz and P. Freeman. Classifying software for reusability. *IEEE Software*, vol 4, no 1, pages 6-16, January 1987.

ISSN: 2324-9773

Durin, B., & Rames, E. (1990). A classification model for reusable software components. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 41-58. doi: 10.7152/acro.v1i1.12463

[PRI90] R. Prieto-Diaz. Domain analysis:an introduction. In *ACM Software Engineering Notes*, vol 15, no 2, pages 47-54, 1990.

APPENDIX

The next page begins an example of a classification scheme for reusable software components.

ISSN: 2324-9773

File: durin_and_rames.l

Modified: Fri Sep 21 20:25:30 1990
Printed: Fri Sep 21 20:25:39 1990

Application_Domain

```
;Example of a Classification Schema built for indexing software components
;All the components are not necessary reusable but the scheme illustrates
;some of the notions presented in the paper

;======================================================================
;  definition du critere Application_Domain - fichier Application_Domain
;----------------------------------------------------------------------

(defcritere Application_Domain "Setting in which the REs are applied"
   (CLASSES
           (On_board_systems
                (Unmanned_spacecrafts
                    (Satellite
                        (Platform
                         Payload
                            (Communication_Payload
                             Observation_Payload
                             Scientific_Payload))
                     Spatial_carriage)
                 Manned_spacecrafts
                     (Space_shuttle
                      Space_station))
            Ground_systems
                (Control_centers
                    (Satellite_C.C.
                     Launcher_C.C.
                     Space_shuttle_C.C.
                     Space_station_C.C.)
                 Testing_benches
                    (Satellite_T.B.
                        (SPOT4_T.B.)
                     Launcher_T.B.
                        (Motor_tests
                         Thrust_chamber_tests
                         Turbo_pump_tests)
                     Space_shuttle_T.B.
                     Space_station_T.B.)
                 Simulation_and_validation_benches
                    (Satellite_S.V.B.
                     Launcher_S.V.B.
                     Space_shuttle_S.V.B.
                     Space_station_S.V.B.)
                 Mission_centers
                    (Satellite_M.C.
                     Launcher_M.C.
                     Space_shuttle_M.C.
                     Space_station_M.C.))
             Miscellaneous))
    (COMPOSANTS
       ((SPOT4_T.B. QAZWSYMB QAAS S SC SH SI SO SR SCC SCD SCE SCF SCP SCR SHT
                    SIA SIB SIC SID SIE SIF SIG SIH SIM SIR SIS SIT SIV SOA SOI SOO
                    SOS SRB SRG SRI SRR SRS HZA HZC HZD HZF HZT HZR HZS HCI HCM HCC
                    HTI HTG HTM HTW HTC HMI HMG HMW HMM HMC HJI HJG HJM HSI HSG HSM
                    QAI QMZ QASC QASH QASE QASB QASA QALC QAS QASG QAO QACC QAZ QAC
                    QAAR QAAH QASD QALE QM QAX QAL QA Q QASF QAAE QAAC QAY QAA QAG
                    QAAD QAT QAD QAE QAF QAAG QAV QAAT QAR QAAM QAAA QAST QASU QASX
                    QADR QADA QAVD QAVG QAVM QAVN QAVT QAEA QAEV QAER QAGW QAGR QAIN
                    QARM QARK QARI QARA QATA QATW QATC QATG QATR QATS QATB QAZYDIAL
                    QAZWWARN QAZWSNUM QAZWQNUM QAZWORDW QAZWORDE QAZWORD QAZWEXEC
                    QAZWCMPL QAZSTART QAZSORD QAZSENDE QAZQMODE QAZPONT QAZOPTQA
                    QAZNDIAL QAZINWAR QAZINSYM QAZINPRM QAZINERR QAZGTEXE QAZGETWA
```

Page 1 of durin_and_rames.l

ISSN: 2324-9773

...Application_Domain

```
               QAZGETSY QAZGETPR QAZGETER QAZEORD QAZCVTSY QAYCRDES QAYBHEAD
               QAYWWORD QAYWSTSB QAYWSTRB QAYWREAB QAYWINTB QAYWBYTB QAYUPDBH
               QAYTESTE QAYSTWOR QAYSTSTR QAYSTREA QAYSTOWC QAYSTOIC QAYSTOFC
               QAYSTOBC QAYSTINT QAYSTBYT QAYSAWOR QAYSASTR QAYSAREA QAYSAINT
               QAYSABYT QAYINIT  QAYGTADS QAYGTADP QAYGDATA QAYCRDEB QAYCOPEC
               QAYBUFST QAYBSYTH QAXWSLEX QAXWSERR QAXWRLEX QAXWILEX QAXTZERO
               QAXTUNZE QAXTUNSC QAXTSIGN QAXTSEPA QAXTRSEP QAXTPERI QAXTOCTA
               QAXTMINU QAXTINTE QAXTINCO QAXTIDUS QAXTHEXA QAXTFMTS QAXTFMTR
               QAXTFMTI QAXTEXPO QAXTEXIC QAXTEXCL QAXTEOLN QAXTCTIM QAXTCSEC
               QAXTCOCT QAXTCBIN QAXTCHOU QAXTCHEX QAXTCDAY QAXTCAND QAXTBINA
               QAXTALPH))))
;=====================================================================
; fin de definition du critere Application_Domain - fichier Application_Domain
;----------------------------------------------------------------------

;=====================================================================
; definition du critere Functional_Area - fichier Functional_Area
;----------------------------------------------------------------------
```

Functional_Area

```
(defcritere Functional_Area ""
   (CLASSES
         (Test_preparation
            (Database_preparation
               (Monitor_tables_generator_program_MTGP
                TM_parameters_preparation
                   (TM_parameters_file_creation)
                TC_preparation_TC
                   (TC_file_creation
                    TC_message_block_generator_program)
                Database_transfert
                Database_loading)
             Synoptic_images_preparation
               (Color_picture_generator_program
                Color_picture_simulator)
             Test_sequence_production_environment
               (Supervisor
                  (Test_sequence_translation
                     (Lexical_analysis
                      Syntactical_analysis
                      Code_generation
                      Gathering
                      Assembling
                      Communication)
                   Test_sequence_creation
                   Test_sequence_editor
                   Test_sequence_emulator)
                Test_sequence_configuration))
          Test_sequence_execution
            (Preparation
             Files_management
               (T_S._Files_Management
                Compile_Files_Management
                User_Files_Access_Command)
             Symbols_resolution
             Blocks_table_management
             Interpretation
               (Scheduler
                Executor)
             Working_management
                (Routines_Calls
                 Sequence_timing
                 Sequence_monitoring
                 Operator_conversation
```

Page 2 of durin_and_rames.l

ISSN: 2324-9773

...Functional_Area

```
       Messages_management
          (T_S._Messages_Management
           Command_Messages_Management))
   Real_time_database
      (Interactive_consultation_of_real_time_database
      Modification_of_real_time_database)
   TC_monitoring_program
      (TC_equipment_management
       TC_generation
       TC_emission)
   TM_monitoring_program
       (Raw_TM_acquisition
        Raw_TM_archiving
        Simulation_and_play_back
        Decommutation)
   Communication_interfaces_with_on_board_software
   Surveillance
      (TM_parameters_surveillance
          (High_and_low_limit_control
           Variation_control
           Waiting_value_control
           state_change_control)
       Synoptic_management
       Graphic_tracing)
   Logbook_real_time_management
      (Archivage_in_the_central_logbook
       Real_time_logbook_edition))
Test_result_analysis
   (Data_processing
      (TM_processing)
   Data_evaluation
      (Functions
          (Basis_functions
           Mathematical_functions
           Graphic_functions)
       Data_analysis
          (Log_file_analysis
           Archive_file_analysis
              (Test_result_data_files
               Raw_test_data_files))
       TM_processing_bis)
    Final_data_and_report_archiving
      (Logbook_and_report_generation
       Logbook_edition))
Interface_management
   (Human_computer_interface
      (Graphics_management
          (G._processing
           G._event_processing)
       HCI_Messages_management
           (HCI_Messages_processing)
       HCI_Files_management
       HCI_Table_management)
    Input_output_supervision_program
    Front_end_equipment
      (TM_front_end_process_programs
       TC_front_end_process_programs
       TM_SCOE_front_end_driver_programs))
 Management_under_OBDH
    (OA
     Error_detection_function_management
     Error_generation_function_management
     Perturbation_function_management
```

Page 3 of durin_and_rames.l

ISSN: 2324-9773

```
          Profile_detection_function_management
          Static_simulation_function_management
          Environment_parameters_management
          Rhythm_synthesizer_management)
       Test_and_simulation_tools
         (Bench_and_test_sequence_tests
            (Real_time_database_test_tools
             Logbook_real_time_test_program
             Compiler_test
             Interpretor_test
             Executer_test
             Test_tolls_for_intertask
             block
             transfert
             Simulation_TC_programs
             Simulation_TM_programs
               (AGATE_simulator)
             Network_management
               (Network_block_emission
                Network_block_reception)
             Intertask_block
             processing
             tools
               (Intertask_block_generation_tools
                Intertask_block_reception_simulator
                Intertask_block_tracing_tools))
          Simulation_tools_for_testing_equipments
            (Orbital_simulation
               (Environment_simulation
                Sensing_device_dynamic_simulation
                Actuator_dynamic_simulation)
             Failure_management
               (Thermic_function_dynamic_simulation
                Supply_dynamic_simulation
                Payload_dynamic_simulation)))
       Miscellaneous))
(COMPOSANTS
 ((Human_computer_interface HZA HCI HTI HMI HJI HSI)
   (HCL_Table_management HZC HZR)
   (Graphics_management HTG HTW HMG HMW HJG HSG)
   (G._processing HZD)
   (HCL_Messages_management HCC HTC HMC)
   (HCL_Messages_processing HZS HTM HCM HMM HJM HSM)
   (HCL_Files_management HZF HZT)
   (Test_sequence_execution S)
   (Interpretation SIR SIC SI)
   (Scheduler SOO SO)
   (Executor SRS SOS)
   (TC_monitoring_program SIV)
   (Blocks_table_management SRB)
   (Preparation SRG SHT SCP)
   (Symbols_resolution SCR SCC)
   (Files_management SIF)
   (User_Files_Access_Command SCF)
   (Compile_Files_Management SCE)
   (T_S._Files_Management SC)
   (Working_management SRR SIG)
   (Messages_management SIS SIM SR)
   (Command_Messages_Management SCD)
   (T_S._Messages_Management SIE)
   (Routines_Calls SIT SIH SID SIB SIA)
   (Sequence_timing SH)
   (Test_sequence_translation QATB QATS QATC QAIN QAGR QAGW QAEA QAVT QAVN QAVD QADA QADR QMZ QM QAI QAG QAF
```

ISSN: 2324-9773

```
QAE QAD QA Q)
        (Syntactical_analysis QATA QASX QASU QAST QASH QASG QASF QASE QASD QASB QASA QASC QAV QAS)
        (Lexical_analysis QARA QARI QARK QARM QALE QALC QAX QAT QAR QAL
                QAXWSLEX QAXWSERR QAXWRLEX QAXWILEX QAXTZERO QAXTUNZE QAXTUNSC
                QAXTSIGN QAXTSEPA QAXTRSEP QAXTPERI QAXTOCTA QAXTMINU QAXTINTE
                QAXTINCO QAXTIDUS QAXTHEXA QAXTFMTS QAXTFMTR QAXTFMTI QAXTEXPO
                QAXTEXIC QAXTEXCL QAXTEOLN QAXTCTIM QAXTCSEC QAXTCOCT QAXTCBIN
                QAXTCHOU QAXTCHEX QAXTCDAY QAXTCAND QAXTBINA QAXTALPH)
        (Communication QAZYDIAL QAZWWARN QAZWSYMB QAZWSNUM QAZWQNUM QAZWORDW
                       QAZWORDE QAZWORD QAZWEXEC QAZWCMPL QAZSTART QAZSORD
                       QAZSENDE QAZQMODE QAZPONT QAZOPTQA QAZNDIAL QAZINWAR
                       QAZINSYM QAZINPRM QAZINERR QAZGTEXE QAZGETWA QAZGETSY
                       QAZGETPR QAZGETER QAZEORD QAZCVTSY QAER QACC QAZ QAC)
        (Assembling QATR QATG QATW QAVM QAVG QAAA QAAD QAAG QAAT QAAR QAAM QAAH
                    QAAS QAAE QAAC QAA)
        (Code_generation QAEV QAO)
        (Gathering QAYCRDES QAYBHEAD QAYWWORD QAYWSTSB QAYWSTRB QAYWREAB
                   QAYWINTB QAYWBYTB QAYUPDBH QAYTESTE QAYSTWOR QAYSTSTR
                   QAYSTREA QAYSTOWC QAYSTOIC QAYSTOFC QAYSTOBC QAYSTINT
                   QAYSTBYT QAYSAWOR QAYSASTR QAYSAREA QAYSAINT QAYSABYT
                   QAYINIT QAYGTADS  QAYGTADP QAYGDATA QAYCRDEB QAYCOPEC
                   QAYBUFST QAYBSYTH QAY)))))
;==================================================================
; fin de definition du critere Functional_Area - fichier Functional_Area
;-----------------------------------------------------------------------


;==================================================================
; definition du critere Function - fichier Function
;-----------------------------------------------------------------------
```

Function

```
(defcritere Function ""
   (CLASSES
        (Management
           (Initialization
            Up_to_date
            Preparation
              (Allocation
               Creation
               Simulation
               Copy
               Convertion)
            Production
              (Integration
               Development)
            Translation
              (Scanning
               Parsing
                 (Recognizing
                  Declarations_analysis
                     (Global_data_managing
                      Local_data_managing)
                  Expressions_analysis)
               Binary_code_generation)
            Execution
              (Start
               Timing
                 (Synchronization
                  Dating
                  Wait)
               Stop
               Sequence)
            Communication
              (Export
```

ISSN: 2324-9773

Durin, B., & Rames, E. (1990). A classification model for reusable software components. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 41-58. doi: 10.7152/acro.v1i1.12463

...Function

```
                      Interfacing
                      GST
                         (quote
                          s_environment)
                      Connection
                      Transfer
                         (Dispatch
                          Transmission
                             (Emission
                              Modulation
                              Demodulation
                              Reception))
                      Loading
                      Unswitch
                      Access)
                  Supervision
                     (Writing
                          (Writing_in_buffer
                           Writing_in_file
                           Display
                           Printing
                           Tracing)
                      Input
                         (Acquisition
                          Validation))
                  Processing
                     (Analysis
                          (Comparison
                           Search
                           Filtering
                           Choice
                           Classification
                           Estimate)
                      Calculation
                         (Coding
                          Decoding)
                      Working
                      Archiving)
                   Configuration
                      (Reading
                       Modification
                       Suppression)
                   Control)
              Schedule
              Test
              Debugging
              Call
              Miscellaneous))
    (COMPOSANTS
       ((Debugging QAE QAD SIB)
        (Schedule SO)
        (Management SRB SIS SIM SIG SHT SH S HZC HZF HZT HZR HCC HTW HTC HMW HMC)
        (Acquisition QAYGTADP QAYGTADS QAZGETER QAZGETPR QAZGETSY QAZGETWA QAZGTEXE
QAZOPTQA QAZQMODE QAZWORDE QAZWORDW QAAR QAAM)
            (Writing_in_file QAZNDIAL QAZWCMPL QAZWEXEC QAZWQNUM QAZWSNUM QAZWSYMB QAZWWARN
                                 QAXWSLEX QAXWSERR QAXWRLEX QAXWILEX)
            (Display QAYBUFST QAZSENDE)
            (Writing_in_buffer QAYSABYT QAYSAINT QAYSAREA QAYSASTR QAYSAWOR QAYSTBYT
QAYSTINT QAYSTOBC QAYSTOFC QAYSTOIC QAYSTOWC QAYSTREA QAYSTSTR QAYSTWOR QAYWBYTB
QAYWINTB QAYWREAB QAYWSTRB QAYWSTSB QAYWWORD)
             (Modification QAGW QAEA QADA)
             (Reading QATR QAGR QAER QADR)
             (Processing SRR SOO SCF SCE HZD HZS HCM HTG HTM HMG HMM HJG HJM HSG HSM QAXTRSEP)
```

ISSN: 2324-9773

Durin, B., & Rames, E. (1990). A classification model for reusable software components. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 41-58. doi: 10.7152/acro.v1i1.12463

...Function

```
          (Decoding SCD)
          (Analysis HZA)
          (Comparison QAZEORD QAZSORD QAZWORD)
          (Communication QAZYDIAL QACC QAZ QAC)
          (Access QATS QATG QATC SIF)
          (Interfacing QAZPONT QAF)
          (Export QARA QARK QARM QASU QAST QAV QAT QAR QAG)
          (Loading SC)
          (Reception SR)
          (Emission SIE)
          (Initialization QAYINIT QAZINERR QAZINPRM QAZINSYM QAZINWAR QAIN QAI SRG
                          SCP HCI HTI HMI HJI HSI)
          (Control QAEV SCC)
          (Sequence SRS SOS)
          (Start QAZSTART)
          (Up_to_date QAYUPDBH QATW QATA)
          (Translation QM QA Q SIV SIT SIR SIH SID SIC SIA SI SCR)
          (Parsing QASF QASA QAX QAS)
          (Recognizing QARI QASX QASB)
          (Declarations_analysis QASG)
          (Global_data_managing QASD)
          (Local_data_managing QASE)
          (Expressions_analysis QASH)
          (Scanning QALE QAL)
          (Binary_code_generation QAAS QAY QAO QAA)
          (Convertion QAZCVTSY QAVT QAVN QAVM QAVG QAVD QALC)
          (Allocation QAYCRDEB QAYGDATA)
          (Simulation QAYTESTE)
          (Copy QAYCOPEC)
          (Creation QAYBSYTH QAYBHEAD QAYCRDES QATB QAAT QAAH QAAG QAAE QAAD QAAC QAAA)
          (Test QAXTZERO QAXTUNZE QAXTUNSC QAXTSIGN QAXTSEPA QAXTPERI QAXTOCTA
                QAXTMINU QAXTINTE QAXTINCO QAXTIDUS QAXTHEXA QAXTFMTS QAXTFMTR
                QAXTFMTI QAXTEXPO QAXTEXIC QAXTEXCL QAXTEOLN QAXTCTIM QAXTCSEC
                QAXTCOCT QAXTCBIN QAXTCHOU QAXTCHEX QAXTCDAY QAXTCAND QAXTBINA
                QAXTALPH)))))
;====================================================================
; fin de definition du critere Function - fichier Function
;--------------------------------------------------------------------

;====================================================================
; definition du critere Entity - fichier Entity
;--------------------------------------------------------------------
```

Entity

```
(defcritere Entity ""
   (CLASSES
           (equipment
               (translator
                  (t_s._translator)
                payload
                supply
                sensing_device
                _actuateur_)
             information
               (programming_languages
                assembly_instructions
                compile_informations
                   (compile_errors
                       (errors
                        warnings)
                    compile_mode
                    compile_option
                    compile_phasis
                    occurences
```

ISSN: 2324-9773

...Entity

```
     (symbols_occurences
      sequence_parameters_occurences)
   symbols
   syntax_tree
   lexem
     (symbol_lexem
      error_lexem
      number_lexem
         (real_lexem
          integer_lexem))
   internal_types
     (characters
         (numeric_c.
             (null_c.
              not_null_c.
              octal_c.
              decimal_c.
              hexadecimal_c.
              binary_c.)
          under_score_c.
          sign_c.
          separator_c.
          period_c.
          inverted_commas_c.
          alphabetic_c.
          ampersand
          special_c.
             (format_c.
                 (hexadecimal_f_c.
                  binary_f_c.
                  octal_f_c.
                  time_f_c.
                  string_f_c.
                  real_f_c.
                  integer_f_c.)
              real_mantisse_c.
              end_of_line_c.))
      numbers
         (integers
          reals)
      address
         (branch_address
          symbol_descriptor_address
          sequence_parameter_descriptor_address
          data_address)
      array
         (word_array
          string_array
          real_array
          byte_array
          integer_array)
      string
      word
      byte))
debug_informations
constant
date
   (date_of_passage)
parameter
   (sequence_p.
    configuration_p.
    environment_p.
    convention_p.
```

...Entity

```
   orbital_p.
   control_p.
     (format
      high_and_low_limit
      variation
      extreme_value
      waiting_value
      state_change
      bracket)
   tm_p.)
variable
area
  (t_s._data_area)
data
  (symbol
     (quote
      s_data_type)
   global_data
   d._for_dating
   d._for_localization)
descriptor
  (symbol_d.
   area_d.
   format_d.
   processing_d.)
block
  (t_s_global_data_b.
   t_s_header_b.
   intertask_b.
   network_b.)
telemetery_data_tm
  (raw_tm
   line_of_tm
   hole_of_tm
   archives_tm)
telecommand_tc
function
  (functor
   transfert_f.
   mathematical_f.
   error_detection
   error_generation
   perturbation
   profile_detection
   static_simulation
   thermic_f.)
table
  (conversion_t.
     (data_type_c_t.
      passing_method_c_t.
      symbol_nature_c_t.
      symbol_type_c_t.)
   reserved_words_t.
     (interpretor_reserved_words_t.
      compiler_reserved_words_t.)
   debugging_blocks_t.
   delay_t.
   configuration_t.
   scattering_t.
   calculation_laws_t.
   symbols_t.
     (symbols_t._header
      command_symbol_t.)
```

ISSN: 2324-9773

...Entity

```
      warnings_t.
      errors_t.
      messages_t.
      parameters_t.
      mnemonics_t.
      masks_t.
      databases_t.
      formats_t.
      connections_t.
      descriptors_t.)
   tests_sequences
     (global_t_s.
      executable_t_s.
      t_s._buffer
      t_s._schedule
      t_s._context
      t_s._timing
      t_s._statements
        (t_s._control_statements
         debug_statements))
   file
     (compile_f.
      temporary_f.)
   database
     (real_time_database
      central_database))
communications
  (messages
   input
     (graphic_inputs
      command
        (analyser_c.
         operator_c.
         working_management_c.
         access_file_c.))
   output
     (test_results
        (chronogram
         graph
         logbook
         statistics
         synoptic)
      image
         (window)
      catalogue
      alarm
      listing
        (compilation_listing)))
tools
  (tool_box
     (syntaxic_analysis_t_b.)
   graphic_tools
   tool_routines
     (syntaxic_analysis_t.r.
      tc_t.r.
      tc_monitor_t.r.
      tch_t.r.
      oba_debug_t.r.
      cyclic_acquisitions_t.r.))
others
  (interface_rules
   cycle
   shareable_image
```

ISSN: 2324-9773

```
            configuration
               (satellite_c.
                material_c.)
            program
               (foreground_p.
                background_p.
                online_p.
                offline_p))
         miscellaneous))
(COMPOSANTS
   ((interface_rules QAZPONT)
    (t_s._translator QAIN QAI)
    (communications QAZSTART QAZYDIAL QAZ SIS)
    (command SCD HZA)
    (access_file_c. SCF)
    (graphic_inputs HSG HJG HMG HTG)
    (window HTW HMW)
    (compilation_listing QAZNDIAL QAZWCMPL)
    (messages QAC SRR SIM SIE SR HCC HCM HTC HTM HMM HMC HJM HSM)
    (t_s_header_b. QAYUPDBH QAYBHEAD QAAR QAAH)
    (t_s_global_data_b. QASD)
    (tests_sequences QALE QM QAX QAL QA Q SRS SOS SO S)
    (t_s._buffer QAYBUFST)
    (t_s._context SIG SCP)
    (t_s._timing SH)
    (global_t_s. QASF)
    (t_s._schedule SOO)
    (t_s._statements SI)
    (debug_statements SIB)
    (t_s._control_statements SIC)
    (executable_t_s. QAYCOPEC QAZGTEXE QAZWEXEC QAAE QAAC QAY QAA)
    (s_data_type QAZCVTSY)
    (global_data QATA QAGR QAG)
    (file SIF SC)
    (temporary_f. HZT)
    (compile_f. SCE HZF)
    (descriptor QAYCRDES SCR)
    (symbol_d. QATC QAVG QAAD QAT SCC)
    (sequence_p. QAZGETPR QAZINPRM QATS)
    (debug_informations QADR QAD)
    (function SIR)
    (functor QATB)
    (variable QAGW)
    (symbols_occurences QAZWSNUM)
    (sequence_parameters_occurences QAZWQNUM)
    (syntax_tree QAO)
    (compile_mode QAZQMODE)
    (lexem QASX QASH QASE QASB QASA QALC QAS)
    (integer_lexem QAXWILEX)
    (real_lexem QAXWRLEX)
    (error_lexem QAXWSERR)
    (symbol_lexem QAXWSLEX)
    (symbols QAZGETSY QAZSORD QASG)
    (compile_errors QAZSENDE QAER QAE)
    (warnings QAZGETWA QAZWORD QAZWORDW QAZWWARN)
    (errors QAZEORD QAZGETER QAZWORDE)
    (compile_option QAZOPTQA)
    (compile_phasis QAYINIT QAYTESTE)
    (reals QAYSTOFC QAYSTREA QAYWREAB)
    (integers QAYSTINT QAYSTOIC QAYWINTB)
    (word QAYSTOWC QAYSTWOR QAYWWORD)
    (byte QAYSTBYT QAYSTOBC QAYWBYTB)
    (characters QAXTEXIC QARI)
```

ISSN: 2324-9773

...Entity

```
        (hexadecimal_f_c. QAXTCHEX)
        (string_f_c. QAXTCSEC QAXTFMTS)
        (real_f_c. QAXTFMTR)
        (octal_f_c. QAXTCOCT)
        (binary_f_c. QAXTCBIN)
        (time_f_c. QAXTCDAY QAXTCHOU QAXTCTIM)
        (integer_f_c. QAXTFMTI)
        (real_mantisse_c. QAXTEXPO)
        (end_of_line_c. QAXTEOLN)
        (separator_c. QAXTRSEP QAXTSEPA)
        (decimal_c. QAXTINTE)
        (octal_c. QAXTOCTA)
        (null_c. QAXTZERO)
        (binary_c. QAXTBINA)
        (not_null_c. QAXTUNZE)
        (hexadecimal_c. QAXTHEXA)
        (under_score_c. QAXTUNSC)
        (ampersand QAXTCAND)
        (period_c. QAXTEXCL QAXTPERI)
        (alphabetic_c. QAXTALPH QAXTIDUS)
        (sign_c. QAXTMINU QAXTSIGN)
        (inverted_commas_c. QAXTINCO)
        (integer_array QAYSAINT)
        (byte_array QAYSABYT)
        (string_array QAYSASTR)
        (real_array QAYSAREA)
        (word_array QAYSAWOR)
        (string QAYSTSTR QAYWSTRB QAYWSTSB)
        (sequence_parameter_descriptor_address QAYGTADP)
        (data_address QAYGDATA)
        (symbol_descriptor_address QAYGTADS)
        (branch_address QATG QATW)
        (programming_languages QAF)
        (warnings_t. QAZINWAR)
        (debugging_blocks_t. QAYCRDEB QAEV QAEA QADA QAAG SRB)
        (calculation_laws_t. HZR)
        (errors_t. QAZINERR)
        (delay_t. SHT)
        (conversion_t. QAV)
        (symbol_type_c_t. QAVT)
        (symbol_nature_c_t. QAVN)
        (data_type_c_t. QAVD)
        (passing_method_c_t. QAVM)
        (symbols_t. QAZINSYM QAZWSYMB QATR QAAT)
        (symbols_t._header QAYBSYTH)
        (command_symbol_t. SRG)
        (reserved_words_t. QARK QAR)
        (compiler_reserved_words_t. QARA)
        (interpretor_reserved_words_t. QARM)
        (t_s._data_area QAAM QAAA)
        (cyclic_acquisitions_t.r. SIA)
        (tc_t.r. SIT)
        (tc_monitor_t.r. SIV)
        (oba_debug_t.r. SID)
        (syntaxic_analysis_t.r. QAST)
        (tch_t.r. SIH)
        (syntaxic_analysis_t_b. QASU)
        (graphic_tools HZD)
        (miscellaneous HCI HTI HMI HJI HSI))))
;==============================================================
; fin de definition du critere Entity - fichier Entity
;--------------------------------------------------------------------
```

ISSN: 2324-9773