# Designing for adaptability and evolution in system of systems engineering

## GCSL syntax, semantics and meta-model

**D_6.3.3**

<table>
<thead>
<tr>
<th>Nature</th>
<th>N/A</th>
<th>Dissemination Level</th>
<th>PU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>DANSE</td>
<td>Contract Number</td>
<td>INFSO-ICT-287716</td>
</tr>
<tr>
<td>Deliverable ID</td>
<td>D_6.3.3</td>
<td>Date</td>
<td>15/02/2015</td>
</tr>
<tr>
<td>Status</td>
<td>Final</td>
<td>Version</td>
<td>1.00</td>
</tr>
<tr>
<td>Contact Person</td>
<td>Valerio Senni</td>
<td>Organisation</td>
<td>ALES S.r.l.</td>
</tr>
<tr>
<td>Phone</td>
<td></td>
<td>E-Mail</td>
<td><a href="mailto:valerio.senni@utsce.utc.com">valerio.senni@utsce.utc.com</a></td>
</tr>
</tbody>
</table>
AUTHORS TABLE

<table>
<thead>
<tr>
<th>Name</th>
<th>Company</th>
<th>E-Mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benoît Boyer</td>
<td>INRIA</td>
<td><a href="mailto:Benoit.Boyer@inria.fr">Benoit.Boyer@inria.fr</a></td>
</tr>
<tr>
<td>Jean Quilbeuf</td>
<td>INRIA</td>
<td><a href="mailto:jean.quilbeuf@inria.fr">jean.quilbeuf@inria.fr</a></td>
</tr>
<tr>
<td>Christoph Etzien</td>
<td>OFFIS</td>
<td><a href="mailto:christoph.etzien@offis.de">christoph.etzien@offis.de</a></td>
</tr>
<tr>
<td>Marco Marazza</td>
<td>ALES S.r.l.</td>
<td><a href="mailto:marco.marazza@utsce.utc.com">marco.marazza@utsce.utc.com</a></td>
</tr>
<tr>
<td>Valerio Senni</td>
<td>ALES S.r.l.</td>
<td><a href="mailto:valerio.senni@utsce.utc.com">valerio.senni@utsce.utc.com</a></td>
</tr>
<tr>
<td>Francesca Stramandinoli</td>
<td>ALES S.r.l.</td>
<td><a href="mailto:francesca.stramandinolil@utsce.utc.com">francesca.stramandinolil@utsce.utc.com</a></td>
</tr>
<tr>
<td>Thomas Peikenkamp</td>
<td>OFFIS</td>
<td><a href="mailto:peikenkamp@offis.de">peikenkamp@offis.de</a></td>
</tr>
</tbody>
</table>

CHANGE HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Reason for Change</th>
<th>Pages Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>May 15, 2014</td>
<td>Initial version</td>
<td>all</td>
</tr>
<tr>
<td>2.00</td>
<td>January 16, 2015</td>
<td>New GCSL patterns to support MTBF</td>
<td></td>
</tr>
<tr>
<td>3.00</td>
<td>February 6, 2015</td>
<td>Final version, transformed into a Deliverable</td>
<td></td>
</tr>
</tbody>
</table>
CONTENTS

1  INTRODUCTION .................................................................................................................................................. 4
   1.1  OVERVIEW, PURPOSE AND SCOPE ............................................................................................................. 4

2  GCSL SYNTAX ........................................................................................................................................................ 5
   2.1  GCSL CONTRACTS ............................................................................................................................................. 5
   2.2  OCL ATOMIC PROPOSITIONS .......................................................................................................................... 6

3  GCSL SEMANTICS ..................................................................................................................................................... 7

GCSL META-MODEL ..................................................................................................................................................... 8

4  GCSL TOOL CHAIN DEVELOPMENT DETAILS AND REMARKS ..................................................................... 13
   4.1  THE TIME BOUNDS IN BLTL: ............................................................................................................................ 13
   4.2  BLTL ENCODING OF OPERATOR MEAN, SUM AND PROD .............................................................................. 13
   4.3  THE BLTL GRAMMAR USED IN PLASMA ............................................................................................................ 14
   4.4  THE OPERATOR PRECEDENCE (TOP-DOWN) IN BLTL ...................................................................................... 15

5  REFERENCES ......................................................................................................................................................... 16
1 Introduction

This working document reports on the work done by ALES, INRIA and OFFIS for the refinement of the GCSL syntax and semantics to support the integration of GCSL in the tool-chain. This document is not a replacement for the official Deliverable D6.3.2, which remains the reference document. This document contains only a refined version of the GCSL syntax and semantics, to be considered as the reference for tools implementation and users, thus replacing the syntax and semantics of D6.3.2.

1.1 Overview, Purpose and Scope

GCSL is a very expressive logic [ref. D6.3.2] parameterized by a language for expressing atomic propositions, based on OCL [http://www.omg.org/spec/OCL/]. Depending on the type of application and the version of the OCL language, it is possible to express various forms of atomic properties. The purpose of this document is to fix a reference version of GCSL syntax and semantics, including the fragment of OCL upon which it is based. For the purpose of the tool-chain integration, this document also reports about the development of a Meta-Model of GCSL. The final version of the GCSL language is going to be presented in full details in the forthcoming deliverable D6.3.3. The current document serves as a support to the development and integration of the tool-chain and as a reference for users to write contracts for use-cases.
Within DANSE, the GCSL language is supported by various analyses techniques. The set of GCSL operators supported by each analysis may vary, depending on the kind of analysis and the maturity of the underlying tools. The exact fragment of GCSL supported by the specific tool is going to be specified within the deliverable describing that tool.
2 GCSL syntax

2.1 GCSL contracts

Here we provide the formal grammar for the GCSL logic, based on the grammar from the PLASMA website https://project.inria.fr/plasma-lab/gcsl/, which has been extended with additional behavioural patterns in order to cover the GCSL deliverable D6.3.2. Note that the GCSL grammar does not specify which fragment of OCL it covers. In particular, it does not provide a specification for the terms OCL-coll, OCL-prop, and OCL-expr below. This is addressed in the next section.

GCSL ::= 
  OCL-coll -> forAll ( variableName | pattern ) |
  OCL-coll -> exists ( variableName | pattern ) |
  MEAN(OCL-expr,interval)  a-relop  OCL-expr |
  SUM(OCL-expr,interval)  a-relop  OCL-expr |
  PROD(OCL-expr,interval)  a-relop  OCL-expr |

pattern ::= 
  [ OCL-prop ] implies [ OCL-prop ] holds forever |
  always [ OCL-prop ] |
  whenever [ OCL-prop ] occurs [ OCL-prop ] holds |
  [ OCL-prop ] implies [ OCL-prop ] during following interval |
  [ OCL-prop ] during interval raises [ OCL-prop ] |
  [ OCL-prop ] during interval implies [ OCL-prop ] during interval then [ OCL-prop ] during interval |
  [ OCL-prop ] occurs int times during interval raises [ OCL-prop ] |
  [ OCL-prop ] occurs at most int times during interval |
  whenever [ OCL-prop ] occurs [ OCL-prop ] holds during following interval |
  whenever [ OCL-prop ] occurs [ OCL-prop ] during following interval |
  whenever [ OCL-prop ] occurs [ OCL-prop ] does not occur during following interval |
  whenever [ OCL-prop ] occurs [ OCL-prop ] occurs within interval |
  always during interval [ OCL-prop ] has been true at least [ OCL-expr ] % of time |
  at the end of interval [ OCL-prop ] has been true at least [ OCL-expr ] % of time |

interv ::= lp ( time ,)? time rp
lp ::= [ ] ( 
rp ::= ) ] )
time ::= N unit | inf
unit ::= s | min | hour | day | year   // year =365 days
2.2 OCL atomic propositions

The grammar below describes the fragment of OCL 2.0 (http://www.omg.org/spec/OCL/) that we support within GCSL. The grammar defines OCL propositions that are Boolean formulas, among which there are patterns that allow to quantify over collections of objects. It defines also OCL expressions, that are arithmetic expressions over integer and real values.

\[
\text{OCL-prop ::= } \text{true} | \text{false} | \text{path} | \text{not OCL-prop} | \text{OCL-prop b-relop OCL-prop} | \\
\quad \text{OCL-expr a-relop OCL-expr} | \\
\quad \text{if OCL-prop then OCL-prop else OCL-prop} | \\
\quad \text{OCL-coll -> forAll( variableName | OCL-prop )} | \\
\quad \text{OCL-coll -> exists( variableName | OCL-prop )} | \\
\quad \text{OCL-coll -> empty()} | \\
\quad \text{OCL-coll -> notempty()}
\]

\[
b\text{-relop ::= and} | \text{or} | \text{xor} | \text{implies} | \text{iff}
\]

\[
a\text{-relop ::= } > | >= | = | <= | << | <
\]

\[
\text{OCL-expr ::= path | OCL-coll -> sum() | OCL-coll -> size()} | \\
\quad \text{if OCL-prop then OCL-expr else OCL-expr} | \\
\quad \text{const} | \text{- OCL-expr | ( OCL-expr )} | \\
\quad \text{OCL-expr a-binfn OCL-expr}
\]

\[
a\text{-binfn ::= } + | - | * | /
\]

\[
\text{OCL-coll ::= itsClassNames | its(ClassName) | } \\
\quad \text{connectedClassNames | connected(ClassName) | } \\
\quad \text{basicTypeAttributeName | } \\
\quad \text{objectName . OCL-coll | variableName . OCL-coll | } \\
\quad \text{itsClassNames . OCL-coll | its(ClassName) . OCL-coll}
\]

\[
\text{path ::= basicTypeAttributeName | objectName . path | variableName . path}
\]

\[
\text{const ::= integers | reals}
\]

REMARKS:

- **basicTypeAttributeName** is a name of an attribute of type Boolean, Int, Float, … (i.e. Primitive Data Types.)
- **itsClassNames** / **its(ClassName)** are (equivalent) expressions used to navigate the relation of containment
- **connectedClassNames** / **connected(ClassName)** are (equivalent) expressions used to navigate the relation of being-connected-to (through a port)
3 GCSL Semantics

From GCSL deliverable D6.3.2: “Let $\tau: \mathbb{N} \rightarrow \mathbb{R}$ be a monotonically increasing time sequence, i.e. for all $i \geq 0: \tau_i \leq \tau_{i+1}$ and let $\Sigma$ be an alphabet. A (infinite) sequence $\sigma_0 \sigma_1 \sigma_2 \ldots$ of symbols of $\Sigma$ is called a trace. A trace $(\sigma, \tau) = (\sigma_0, \tau_0)(\sigma_1, \tau_1)(\sigma_2, \tau_2)$ where $\sigma$ is a trace and $\tau$ is a time sequence is called a timed trace.” Behaviors are timed traces. The semantics of a GCSL formula is a (possibly infinite) set of behaviours. In this section we define the semantics of the GCSL temporal patterns by translation to corresponding B-LTL formulas. This is done following the paper [AiSoS2013] but with some modifications to B-LTL formulas corresponding to patterns 4,5,6 plus the addition of pattern 1 which is present in D6.3.2 but not in [AiSoS2013]. Below we assume that $k$ is the current simulation interval bound: this bound constraints the patterns intervals bounds as indicated in the table below.

<table>
<thead>
<tr>
<th>P.id</th>
<th>Patterns</th>
<th>B-LTL Semantics</th>
<th>Consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\Psi_1$ implies $\Psi_2$ holds forever</td>
<td>$G_{sk}(\Psi_1 \rightarrow G_{sk}(\Psi_2))$</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>always $\Psi$</td>
<td>$G_{sk}\Psi$</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>whenever $\Psi_1$ occurs $\Psi_2$ holds</td>
<td>$G_{sk}(\Psi_1 \rightarrow \Psi_2)$</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>$\Psi_1$ implies $\Psi_2$ during following $[a,b]$</td>
<td>$X_{sa}G_{sb-a}(\Psi_1 \rightarrow \Psi_2)$</td>
<td>$a \leq b \leq k$</td>
</tr>
<tr>
<td>5</td>
<td>$\Psi_1$ during $[a,b]$ raises $\Psi_2$</td>
<td>$(X_{sa}G_{sb-a}\Psi_1) \rightarrow X_{sb}F_{sk-a}\Psi_2$</td>
<td>$a \leq b \leq k$</td>
</tr>
<tr>
<td>6</td>
<td>$\Psi$ during $[a,b]$ implies $\Psi_1$ during $[a,c]$ then $\Psi_2$ during $[c,b]$</td>
<td>$(X_{sa}G_{sb-a}\Psi) \rightarrow (X_{sa}G_{sc-a}\Psi_1 \land X_{sc}G_{sb-c}\Psi_2)$</td>
<td>$a \leq c \leq b \leq k$</td>
</tr>
</tbody>
</table>

**EXTENDED Patterns**

| 7    | $\Psi_1$ occurs $n$ times during $[a,b]$ raises $\Psi_2$ | $occ(\Psi_1, a, b) \geq n \rightarrow X_{sa}F_{sk-b}\Psi_2$ | $a \leq b \leq k$ |
| 8    | $\Psi$ occurs at most $n$ times during $[a,b]$ | $occ(\Psi_1, a, b) \leq n$ | $a \leq b \leq k$ |
| 13   | always during $[a,b]$, $\Psi$ has been true at least $E\%$ of time | $G_{sb}(TimeOf(\Psi) \geq \frac{E}{100} \cdot \#Time \lor \#Time < a)$ | $a \leq b \leq k$ |

**at the end of $[a,b]$, $\Psi$ has been true at least $E\%$ of time**

| 9    | whenever $\Psi_1$ occurs $\Psi_2$ holds during following $[a,b]$ | $G_{sk-b}(\Psi_1 \Rightarrow X_{sa}G_{sb-a}\Psi_2)$ | $a \leq b \leq k$ |
| 10   | whenever $\Psi$ occurs $\Psi_1$ implies $\Psi_2$ during following $[a,b]$ | $G_{sk-b}(\Psi \Rightarrow X_{sa}G_{sb-a}(\Psi_1 \Rightarrow \Psi_2))$ | $a \leq b \leq k$ |
| 11   | whenever $\Psi_1$ occurs $\Psi_2$ does not occur during following $[a,b]$ | $G_{sk-b}(\Psi_1 \Rightarrow X_{sa}G_{sb-a} \lnot \Psi_2)$ | $a \leq b \leq k$ |
| 12   | whenever $\Psi_1$ occurs $\Psi_2$ occurs within $[a,b]$ | $G_{sk-b}(\Psi_1 \Rightarrow X_{sa}F_{sk-a}\Psi_2)$ | $a \leq b \leq k$ |
The latest version of the GCSL Meta Model can be retrieved from the DANSE SVN repository: 


The GCSL Meta Model supports the integration of the DANSE Analysis tools, in particular the integration of the Plasma Statistical Model Checker. For this reason, every class of this model contains a getBLTL function that allows translating the entire sub-tree starting from that class into an equivalent BLTL formula as indicated in Section 3. The getBLTL function has an integer argument containing the simulation time, which is required for proper translation into BLTL.

The picture above illustrates the modelling of a GCSL contract, having an ID of type string, a threshold of type double indicating the minimal probability of the contract truth, the referredComponentPathName of type string that contains the path to the Component to which the contract is attached and containing two GCSL assertions that are the Assume and the Guarantee parts of the contract. GCSL assertions are marked in red. They can be either TopLevelQuantifier or Accumulator (MEAN/SUM/PROD) or PatternComposer (AND/OR composition). They correspond to the syntactic elements provided in the grammar in Section 2.1. Then, nested, there are OCL Propositions, OCL quantifiers and variables. For some classes (Pattern, Binary Op and UnaryOp) the implementation of the getBLTL function was provided by using OCL constraints. This choice proved to be useful to allow effortless extension of the Patterns to support the needs of the users.
The picture above illustrates how OCL Propositions and Expressions are modelled. This part of the Meta Model is a little less restrictive than the grammar provided in Section 2.2. However, the GCSL grammar is tested by the GCSL Parser plugin and it will therefore produce correct instances of the Meta Model. The GCSL Parser plugin can be found in the DANSE repository at the following address:


Finally, the picture above illustrates how Navigation Expressions are modelled: paths are sequences of navigable features that can be either model elements (i.e. Components and subcomponents identifiers) or variables (i.e. Components’ and subcomponents’ attributes).
Figure 1 Overview of the GCSL Meta-Model components
Figure 2. GC SL Meta Model Exploded View – part 1
Figure 3. GCSL Meta Model Exploded View – part 2

Figure 4. GCSL Meta Model Exploded View – part 3
4 GCSL Tool chain development details and remarks

The purpose of this section is to collect the technical details that are relevant for the tool chain developers, i.e. ALES, OFFIS and INRIA. Therefore this section is not meant for the normal users, concerning mainly the GCSL compiler output alignment with the inputs of analysis tools.

4.1 The Time bounds in BLTL:

Each temporal operator has a time bound that define the duration on which it must hold. This time bound is always a **constant value** and it must be expressed in the **basic time unit** used in simulator, or in **number of states**.

In the DANSE settings, the basic time unit provided by Desyre is the **second**. For instance, the property like 
\( (G\leq b \ P_1) \) means that P1 must hold during the next **b seconds** in the simulation. It means that every time bound used in a GCSL contract must be converted in seconds.

A GCSL contract may also be defined over interval using **steps** of simulation. This notion of steps match exactly the notion of **states** that can also be used to bound the temporal operators. In BLTL, the distinction between time of the simulation and the steps of simulation is done by prefixing the number of steps with the character ‘#’. For instance, the property like 
\( (G\leq \#b \ P_1) \) means that P1 must hold during the next **b states** of the simulation.

4.2 BLTL encoding of operator MEAN, SUM and PROD

<table>
<thead>
<tr>
<th>MEAN(OCL-expr,interval)</th>
<th>SUM(OCL-expr,interval)</th>
<th>PROD(OCL-expr,interval)</th>
</tr>
</thead>
</table>

MEAN(OCL-expr, [low, up]) is translated into BLTL as follows:

\[
(X\leq low \ X< len \ MEAN< len \ (BLTL-expr) )
\]

where

1. BLTL-expr is OCL-EXPR encoded in BLTL
2. len is a constant defined as len = up – low
3. the time bound low, up and len are constants defined in seconds.

The encoding of SUM and PROD operators follows exactly the same construction.
4.3 The BLTL grammar used in Plasma

Expression ::= IfThenElse
   | IfThenElse ( U | W ) <= Bound Expression
   | ( F | G ) <= Bound Expression
   | X ( <= Bound )? Expression

Bound ::= # integer | number
IfThenElse ::= Implication ? Implication : Implication
Implication ::= Disjunction (=> Disjunction)?
Disjunction ::= Conjunction (| Conjunction)*
Conjunction ::= Equality (& Equality)*

Equality ::= != RelExp (Eop RelExp)?
Eop ::= = | !=

RelExp ::= NumExp (Rop NumExp)?
Rop = < | <= | >= | >

NumExp ::= Term (Nop Term)*
Nop = + | -

Term ::= Factor (Top Factor)*
Top = * | /

Factor ::= -. ( | ( Expression )
   | Function
   | ident | number
   | true | false

Function ::= MEAN (<= Bound)? ( NumExp )
   | FtoT (<= Bound)? ( Implication )
   | MIN ( NumExp )
   | MAX ( NumExp )
4.4 The operator precedence (top-down) in BLTL

- (unary minus)
*, / (multiplication, division)
+, - (addition, subtraction)
<, <=, >=, > (relational operators)
=, != (equality operators)
! (negation)
& (conjunction)
| (disjunction)
=> (implication)
? (ternary operator test ? a : b means "if test = true then a else b")
U W F G X (temporal operators)
5 References