Project: DAZ PROJECT

Title: Example of Literate Script Development

Ref: ISS/HAT/DAZ/WRK503 *Issue:* 1.20 *Date:* 22 July 2011

Status: Informal *Type:* Technical

Author:

Name	*Location*	*Signature*	*Date*
D.J. King	WIN01		

Authorisation for Issue:

Name	*Function*	*Signature*	*Date*
R.D. Arthan	HAT Team		

Abstract: This document gives an experimental implementation "Refinement of Z to SPARK: Example of Literate Script Development", DRA/CIS/CSE3/SWI/WP/7/2.

Distribution: Library

0 DOCUMENT CONTROL

0.1 Contents List

0.2 Document Cross References

[1] DRA/CIS/CSE3/SWI/WP/5/2. *Refinement of Z to SPARK: Example - First 1000 Primes.* C.M. O'Halloran, C.T. Sennett, and A. Smith, Defence Research Agency, Malvern, 11th October 1993.

[2] DRA/CIS/CSE3/SWI/WP/7/2. *Refinement of Z to SPARK: Example of literate script development.* C.M. O'Halloran, C.T. Sennett, and A. Smith, Defence Research Agency, Malvern, 11th October 1993.

[3] ISS/HAT/DAZ/HLD501. *High Level Design: Overview.* D.J. King, Lemma 1 Ltd., `http://www.lemma-one.com`.

[4] ISS/HAT/DAZ/USR501. *Compliance Tool — User Guide.* Lemma 1 Ltd., `http://www.lemma-one.com`.

0.3 Changes History

Issue 1.1- 1.4 First Issues.

Issue 1.5 IUCT WP 2 changes.

Issue 1.6 IUCT WP 4 syntax change.

Issue 1.9 Copyright and banner updates for open source release.

Issue 1.10 DAZ-specific updates to banner for open source release

Issue 1.11 DAZ-specific updates to banner for open source release

Issue 1.12 The SPARK program is now referred to as the Ada program.

Issue 1.13 Made it output the same success message as other tests.

Issue 1.14 Compliance Notation reserved words are now prefixed by a dollar sign.

Issue 1.15 Allowed for enhancement 117.

Issue 1.16 Allowed for automated state management.

Issue 1.19 Allowed for enhancement 165.

0.4 Changes Forecast

None.

1 GENERAL

1.1 Scope

This document contains a transcription of the literate script for the "Literate Script" example in [2] suitable for input into the Compliance Tool as specified in [3] and [4].

1.2 Introduction

1.3 Background

This document contains an adaptation of the script in [1] suitable for input into the compliance tool.

2 LITERATE SCRIPT 1

SML

```
new_script {name="PACK_P", unit_type="spec"};
```

Compliance Notation

$\langle$ *package PACK_P spec* $\rangle$ (*1*)

Compliance Notation

(*1*) $\equiv$

```
package PACK_P is
   I : INTEGER;
   type COLOUR is (RED, BLUE, GREEN);
   procedure SQRT (X : INTEGER; Y : out INTEGER)

    Δ Y [X ≥ 0, Y ** 2 ≤ X < (Y + 1) ** 2];

   procedure CUBE_ROOT (N : in out INTEGER)

    Δ N [N ≥ 0, N ** 3 ≤ N0 < (N + 1) ** 3];

   function NEXT_COLOUR (C : COLOUR) return COLOUR

    Ξ
    [true,
    C ≠ PACK_PoCOLOURvLAST ∧
    PACK_PoNEXT_COLOUR(C) = PACK_PoCOLOURvSUCC(C)
```

```
    ∨
    C = PACK_PoCOLOURvLAST ∧
    PACK_PoNEXT_COLOUR(C) = PACK_PoCOLOURvFIRST];

   function PLUS_ONE (X : INTEGER) return INTEGER;

end PACK_P;
```

SML

```
output_z_document{script="PACK_P'spec", out_file="lit1.zdoc"};
output_ada_program{script="-", out_file="lit1.ada"};
new_script {name="MAIN", unit_type="proc"};
```

Compliance Notation

```
⟨ main procedure ⟩              (2)
```

Compliance Notation

```
(2) ≡

with PACK_P;
procedure MAIN is
   C : PACK_P.COLOUR;
   type PERSON is
   record
      AGE : INTEGER;
      EYES : PACK_P.COLOUR;
   end record;
   JACK : PERSON;
begin
      PACK_P.CUBE_ROOT(PACK_P.I);

       Δ C, JACK [true, C = PACK_PoRED]

      C := PACK_P.BLUE;
end MAIN;
```

Compliance Notation

```
⊑

JACK := PERSON'(25, PACK_P.GREEN);
C := PACK_P.NEXT_COLOUR(JACK.EYES);
```

SML

```
output_z_document{script="MAIN'proc", out_file="lit1a.zdoc"};
output_ada_program{script="-", out_file="lit1a.ada"};
```

3 LITERATE SCRIPT 2

SML

```
new_script {name="PACK_P", unit_type="body"};
```

Compliance Notation

⟨ *package PACK_P body* ⟩ (*1*)

Compliance Notation

(*1*) ≡

```
package body PACK_P is
   HAIR : COLOUR;
   J, K : INTEGER;
   type ARR_COLOUR is array(COLOUR) of COLOUR;
   NEXT_COL : constant ARR_COLOUR :=
                  ARR_COLOUR'(BLUE, GREEN, RED);
   function PLUS_TWO (L : INTEGER) return INTEGER

   Ξ [true, PLUS_TWO(L) = L + 2]

   is
   begin
      return L + 2;
   end PLUS_TWO;
   procedure SQRT (X : INTEGER; Y : out INTEGER)

    Δ Y [X ≥ 0, Y ** 2 ≤ X < (Y + 1) ** 2]

   is separate;
   procedure CUBE_ROOT (N : in out INTEGER)

    Δ N, J [N ≥ 0, N ** 3 ≤ N0 < (N + 1) ** 3]

   is
```

```
   L : INTEGER;
begin

   Δ N, J, L [N ≥ 0, N ** 3 ≤ N0 < (N + 1) ** 3] (2)

   L := PLUS_TWO(L);
end CUBE_ROOT;

⟨ function NEXT_COLOUR body ⟩                          (3)

function PLUS_ONE (X : INTEGER) return INTEGER

 Ξ [true, PLUS_ONE(X) = X + 1]

is
   L : INTEGER;

     ⟨ procedure PLUS_FOUR body ⟩                      (4)

begin

     Δ L [true, PLUS_ONE(X) = X + 1]                   (5)

end PLUS_ONE;
end PACK_P;
```

Compliance Notation

```
(3) ≡

function NEXT_COLOUR (C : COLOUR) return COLOUR

Ξ
[true,
C ≠ PACK_PoCOLOURvLAST ∧
NEXT_COLOUR(C) = PACK_PoCOLOURvSUCC(C)
∨
C = PACK_PoCOLOURvLAST ∧
NEXT_COLOUR(C) = PACK_PoCOLOURvFIRST]

is separate;
```

SML

```
open_scope "PACK_P.PLUS_ONE";
```

Compliance Notation

```
(4) ≡

procedure PLUS_FOUR (A : INTEGER; B : out INTEGER)

Δ B [true, B = A + 4]

is
begin
   B := A + 4;
end PLUS_FOUR;
```

Compliance Notation

```
(5) !⊑
PLUS_FOUR(X, L);
L := L − 3;
return L;
```

SML

```
output_z_document{script="PACK_P'body", out_file="lit2.zdoc"};
output_ada_program{script="−", out_file="lit2.ada"};
```

4 LITERATE SCRIPT 3

SML

```
new_script {name="PACK_P.SQRT", unit_type="proc"};
```

Compliance Notation

```
⟨ procedure PACK_P.SQRT body ⟩ (1)
```

Compliance Notation

```
(1) ≡

separate (PACK_P)
procedure SQRT (X : INTEGER; Y : out INTEGER)

Δ Y [X ≥ 0, Y ** 2 ≤ X < (Y + 1) ** 2]

is
```

$LO : INTEGER;$

$\langle$ *local vars* $\rangle$ (2)

begin

$LO := 0;$

$\Delta\ LO\ [X \geq 0 \wedge LO = 0,\ LO\ **\ 2 \leq X < (LO + 1)\ **\ 2]$

$Y := LO;$

end SQRT;

Compliance Notation

(2) $\equiv$

$HI : INTEGER;$

Compliance Notation

$\sqsubseteq$

$\Delta\ LO,\ HI\ [X \geq 0 \wedge LO = 0,\ LO\ **\ 2 \leq X < (LO + 1)\ **\ 2]$

Compliance Notation

$\sqsubseteq$

$HI := X + 1;$

$\$still\ [\![LO\ **\ 2 \leq X < (LO + 1)\ **\ 2]\!]$

loop

$\Delta\ LO,\ HI\ [LO\ **\ 2 \leq X < HI\ **\ 2,\ LO\ **\ 2 \leq X < HI\ **\ 2]$

end loop;

Compliance Notation

$\sqsubseteq$

exit when $LO + 1 = HI;$

$\Delta\ LO,\ HI\ [LO\ **\ 2 \leq X < HI\ **\ 2,\ LO\ **\ 2 \leq X < HI\ **\ 2]$

SML

```
output_z_document{script="PACK_PoSQRT'proc", out_file="lit3.zdoc"};
output_ada_program{script="-", out_file="lit3.ada"};
```

5 LITERATE SCRIPT 4

SML

```
new_script {name="PACK_P.NEXT_COLOUR", unit_type="func"};
```

Compliance Notation

```
separate (PACK_P)
function NEXT_COLOUR (C : COLOUR) return COLOUR
Ξ
[true,
C ≠ PACK_PoCOLOURvLAST ∧
NEXT_COLOUR(C) = PACK_PoCOLOURvSUCC(C)
∨
C = PACK_PoCOLOURvLAST ∧
NEXT_COLOUR(C) = PACK_PoCOLOURvFIRST]

is
   C1 : COLOUR;
begin

  Δ C1
  [true,
  C ≠ PACK_PoCOLOURvLAST ∧
  NEXT_COLOUR(C) = PACK_PoCOLOURvSUCC(C)
  ∨
  C = PACK_PoCOLOURvLAST ∧
  NEXT_COLOUR(C) = PACK_PoCOLOURvFIRST]

end NEXT_COLOUR;
```

Compliance Notation

```
⊑

C1 := C;
C1 := NEXT_COL(C1);
return C1;
```

SML

```
output_z_document{script="PACK_PoNEXT_COLOUR'func", out_file="lit4.zdoc"};
output_ada_program{script="-", out_file="lit4.ada"};


diag_line "All module tests passed.";
```