



# Generation of Documentation using ASIS Tools

S.V. Hovater

# Overview

- ◆ Inception
  - Why do this?
- ◆ Elaboration
  - Prototyping, proof-of-concept
- ◆ Construction
  - Exercising all possible leverage
- ◆ Transition
  - Looking back, and looking forward

# Inception

## ◆ Why do this?

- Large projects can receive code drops without complete documentation
- Reduce the effort to maintain IDD's
- Reduce the time required to maintain IDD's
- Reduce the cost of maintaining IDD's
- Synchronize the documentation with the as-built reality
  - Documentation becomes an artifact of the code

# Elaboration

- ◆ First prototype was a 100-line ASIS program
- ◆ Written over a weekend using Rational Apex
- ◆ Extracted key information
- ◆ Expanded to handle more component types, validate the concept
- ◆ The learning curve for ASIS is steep
  - It took considerable investment to become effective
  - By the time we had validated the concept, we were hot!
- ◆ Unfortunately, there's no "ASIS for Dummies"
- ◆ Zen of ASIS: *it's a loosely typed system implemented in a strongly typed language*

# Construction

- ◆ Transitioned from stand-alone prototype to Ada Analyzer™ (Littletree Consulting) extension
  - Make use of annotation collection code already existing
  - Leverage ASIS navigation code
  - Leverage front-end, type-resolution code
- ◆ Created in-memory representation to facilitate reporting
- ◆ Created RTF output code (graphical tables)
  - We briefly considered HTML
- ◆ 80% goal expanded to full coverage
  - Tagged types, Discriminated record types
- ◆ ASIS viewer (Apex tool) proved invaluable

# IDDET input

- ◆ Record types
  - Vanilla
  - Tagged
  - Discriminated
- ◆ To construct a bitmap, needs a rep spec
- ◆ Without a rep spec, only get a list of the researched record elements
- ◆ Record Components can be varied
  - Array, scalar, record, etc.

# IDDET Output

- ◆ Component types are “researched” –
  - Records are recursively handled
  - Array bounds + element type are discovered
  - Subtypes are resolved to the base type + constraint
  - Discriminants are found & choices resolved
  - Enumeration literals are displayed
  - Specific annotations are noted
- ◆ Presence of rep spec reorders the order of the displayed components in rep-spec order, enables bitmap.
- ◆ Output format J-016-1995

# IDDET Sample Output

## 1.1.1 IDDET\_TEST\_PKG.MESSAGE\_TYPE

this is the Message\_Type annotation

0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	3	3				
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1		
X																												Word 0					
*	Spare																											Word 1					
[ 0.. 31 ] + Z (2 elements of Vrecord_Type)																												Word 2					
[ 32.. 63 ] + Z (2 elements of Vrecord_Type)																												Word 3					
[ 64.. 95 ] + Z (2 elements of Vrecord_Type)																												Word 4					
[ 96.. 127 ] + Z (2 elements of Vrecord_Type)																												Word 5					
[ 128.. 143 ] + Z (2 elements of Vrecord_Type)														Spare														Word 6					
Spare										Mode_Mask (3 elements of Enum)			Spare															Word 7					

Word	Bit	Name
1	0	Y

# Transition

- ◆ Released production version
- ◆ Common complete code base supports
  - Solaris 2.6, 2.7
  - HP-UX 10.20, 11.0
- ◆ No distinction for host/embedded source
  - ASIS is the same (but beware endian!)
- ◆ Looking forward
  - ASIS 2.0
  - NT hosting
    - (LittleTree has just produced NT version of the Ada Analyzer™)