Using ASIS to Generate C++ Bindings

Gachia -

Generating a C++ Header Interface Automatically

Background / Motivation

- One component (FLTS) of the FAA ERAM (En Route Automation Modernization)
- ERAM primarily developed in Ada
- Utilizes components from previous ATC developments, also primarily in Ada
- **Need to provide access / interface for C++ client programs** (primarily display interfaces)
- Rapidly changing Ada API during development – **C++ API lagging severely behind**

Approach

- Use ASIS to generate API headers
  - Package / File
  - Supporting Utilities (Data Serialization - incomplete)
- Determine language construct mapping
  - Packaging
  - Types
  - Functions / Procedures
  - Exceptions
- Integration with other components

Ada / C++ Language Mapping

- Packaging
- Types
  - Intrinsic
  - Composite / Structured
  - “Abstract” (includes abstract, tagged, variant, discriminant)
- Functions / Procedures
- Exceptions

Ada / C++ Packaging

- Ada packages
- C++ namespaces
- Both have “child” capabilities
- Really straightforward…
  - Ada parent-child.ads, .adb
  - C++ parent-child.h, .cpp

Ada / C++ Types

- Intrinsic
  - Constants
  - Enumeration types
  - Integer types
  - Arrays (including strings)
- Composite / Structured
  - Records / structs
- “Abstract”
  - Abstract
  - Tagged
  - Variant records
  - Discriminant types

Intrinsic

- Constants
- Enumeration types
  - C++ requires unique per namespace
- Integer types
  - C++ “expansion” to int
- Arrays (including strings)
  - C++ “0” based

Composite / Structured

- Records / structs
  - Ada variant
  - C++ union (not addressed)
  - Bit layout issues **NOT** addressed at this point

“Abstract” Types

- Private Types Mapped to C++ Classes
- Interesting issues
  - Variants (often different constructors)
  - Tagged (“protected” rather than “private” data members)

Ada / C++ Types (contd)

- Type Issues
  - Dynamic Types
    - *E.g.* C++ vectors, lists
  - Booch Components
- Approach
  - Use the “natural” C++ style in header
  - Use collections / “type tool” in wrapper implementation (presentation tomorrow by Matt Mark)

Ada / C++ Exceptions

- Defined in base level packages for both languages
- Implemented cross-language exception passing in wrappers via an “errno” like approach
- Another compiler dependent “pop-stack-and-call option” used elsewhere

ASIS Overview

- Abstract interface to underlying compiler's internal representation of code
- Tutorial Information on the Web – start at http://www.sigada.org/wg/asiswg/
- Requires Ada code (specifications on for this project) be compiled
- Requires ASIS implementation by vendor
  - May require compilation with special flags (e.g., GNAT's "-gnatt" to retain parse tree)

ASIS Components

- Compilation Unit
- Declaration
- Definitions
- Elements

ASIS – Compilation Unit

- Contains queries about compilation units
  - What declarations are contained
  - Whether in specification or body
  - Private or public (visible)
  - Whether body is required
  - Name

ASIS – Declarations

- Queries for declaration objects
  - Type (variable, constant, number, package, procedure / function, generic / not, etc.)
  - Name
  - Parameter profile (for functions / procedures)
  - Result (return) profile
  - Visibility

ASIS – Definitions

- Typing
  - Parent type
  - Structure / record components
  - Enumeration literals
  - Integer / real range constraints
  - Mark
  - Array bounds
  - Discriminants and variants

Elements

- Context (“with”) clauses attached to packages
- Function call parameters
- Type kinds

Generated outputs

- Basic Interface:
  - C++ header file (*.h)
- Support interface:
  - Ada interface wrapper files (*-w.ad[bs])
  - C++ wrapper files (*-w.h)
  - C++ skeleton “implementation” (*.cpp)
  - Data Serialization (TBD)

Basic Interface

- Iterate through the package declarations
  - Recurse through “with’d” packages
- Generate corresponding intrinsic types
- Collect functions / procedures for C++ classes
- Generate C++ class interfaces

Support Interface

- Ada interface wrapper files (*-w.ad[bs])
  - Pragma export C++ names
  - Create everything procedurally
  - Wrap an “errno” around existing interfaces
- C++ wrapper files (*-w.h)
  - Use “externs” to limit C++ compiler name mangling
- C++ skeleton “implementation” (*.cpp)
  - Like a “stubsrc”, to link against
- *Data Serialization (TBD)*
  - *Again – attend tomorrow’s talk by Matt Mark*

Integration with other Components

- Some other components already had C++ interfaces defined
- Standard C++ features used in order to be more natural for the C++ programmer
- Effectively drew a “line in the sand” and implemented with textual substitution and manual editing

Limitations

- Development process required inspection of **all** code (generated or manual)
- Some rough edges not completed
- Ada API included some components not required by C++ clients
- Optimizations possible when clients access was limited to “reading” the data

Results

- Effective for bulk of header files
  - Most simple types
  - Generated classes and methods
  - Created ~ 500 header files for FLTS
- Not fully implemented
  - Serialization deferred
  - Inherited types incomplete
  - C++ “Bodies” unimplemented

Use of ASIS

- Intuitive interface
- Straightforward implementation
- Reasonable performance (5 – 6 minutes execution time for hundreds / thousands of files)
- Much nicer to be working in Ada rather than awk, perl, grep, sed, cpp!