An Interface to the Ada 95 Compilation Environment

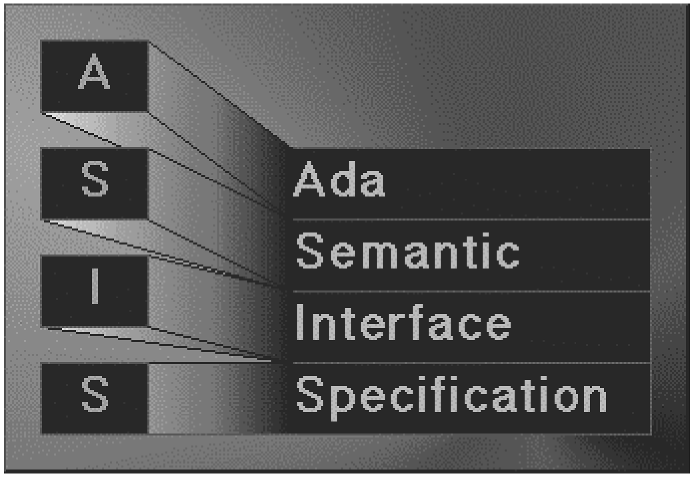
Analysis of Mission-Critical Systems Using ASIS

Mr. Currie Colket
Chair, ACM/SIGAda/ASIS Working Group (ASISWG)
Chair, ISO/IEC JTC1/SC22 WG9 ASIS Rapporteur Group (ASISRG)
Phone: (703) 242-4561; Email: ***colket@acm.org***

Dr. Bill Thomas
Vice Chair, ASISWG for Publicity/Meetings
Phone: (703) 883-6159; Email: ***bthomas@mitre.org***

21 April 1998
STC'98

ASIS Home Page => ***http://www.acm.org/sigada/WG/asiswg***



An Interface to the Ada 95 Compilation Environment

Overview

- **Why Code Analysis for Mission-Critical Systems**
- **What is ASIS?**
  - **- Syntactic & Semantic Queries**
  - **- Examples of Tools Built on ASIS**
- **ASIS Architecture**
- **ASIS Examples for Analysis of Mission-Critical Systems**
- **Expected ASIS Schedule**
- **Summary**

Why Code Analysis for Mission-Critical Systems

Safety/Quality Concerns

- Safety and security verification
- Quality assessment metrics
- Test-case generation and coverage analysis
- Coding style and standards compliance

Performance Concerns

- Timing and sizing estimation
- Dependency trees and impact analysis

Engineering Concerns

- Data flow analysis and usage metrics
- Invocation (call) trees and cross-reference
- Usage counts of language constructs
- Code browsing and navigation
- Documentation generation
- Reverse engineering and re-engineering
- Language translation and code restructuring

Addressing these concerns results in higher quality systems

Safety and Security Compliance Tools

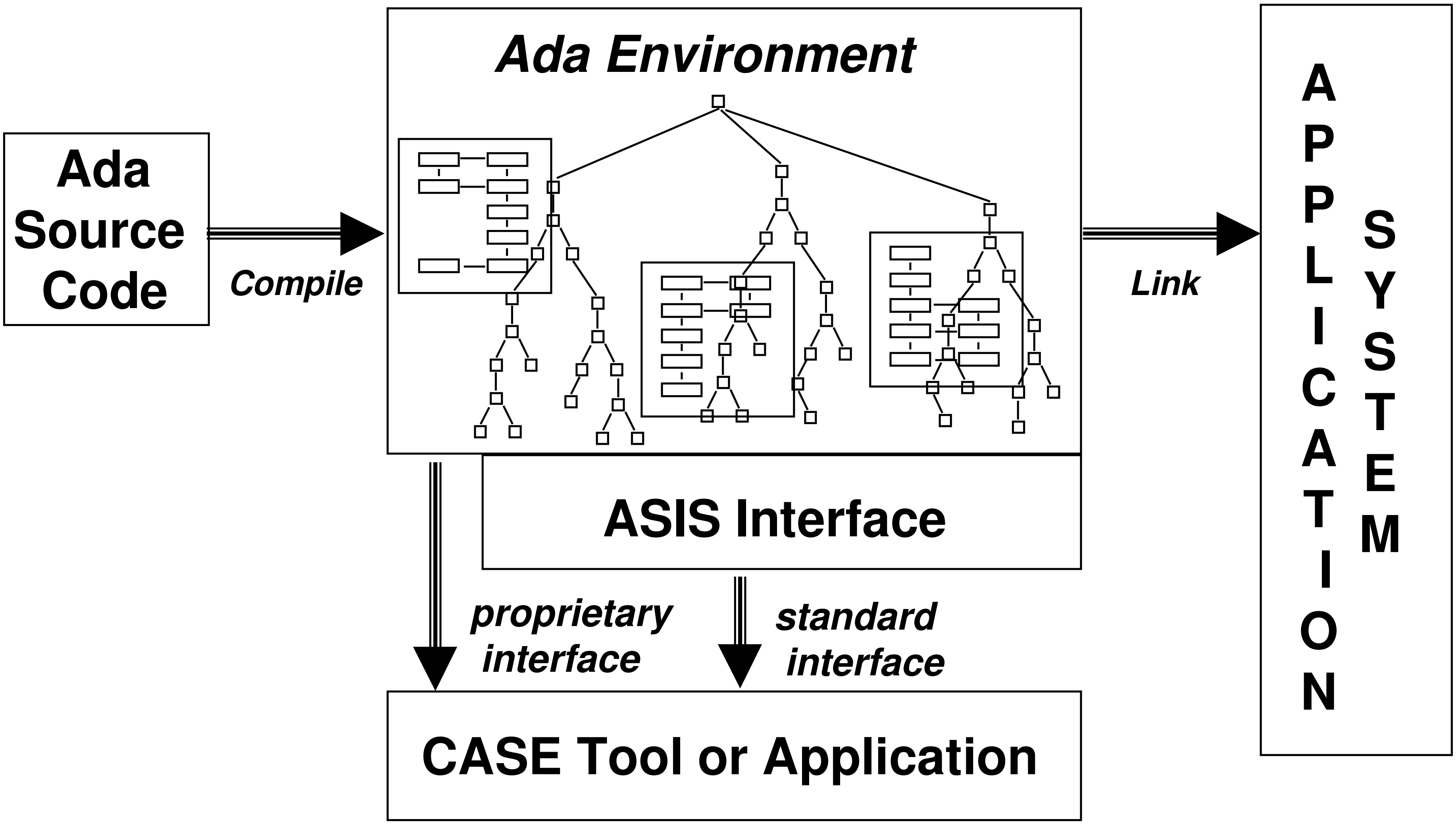
HRG (Annex H Rapporteur Group) Address safety and security issues by

- Synthesizing essential requirements of typical sector-specific standards for high integrity applications tools
- Identifying restrictions for deterministic applications
- Producing taxonomy of techniques for the construction and analysis of high integrity software:
  - Use of annotations in program construction
  - Error detection by static analysis
  - Design confirmation by static analysis
  - Static timing analysis
- Supporting the interoperation of high integrity software with tools:
  - **ASIS**, Ada compilers and RTEs, & CORBA

Simple ASIS example to identify violations of restrictions might

- **Scan your source code**
  - **Using ASIS Traverse_Element generic procedure**
  - **Actual for its Pre_Operation formal procedure should check the kind of the Element being traversed**
- **Report all occurrences of language constructs prohibited**

What is ASIS?



Provides Syntactic and Semantic Information from Ada Environment using a standard interface

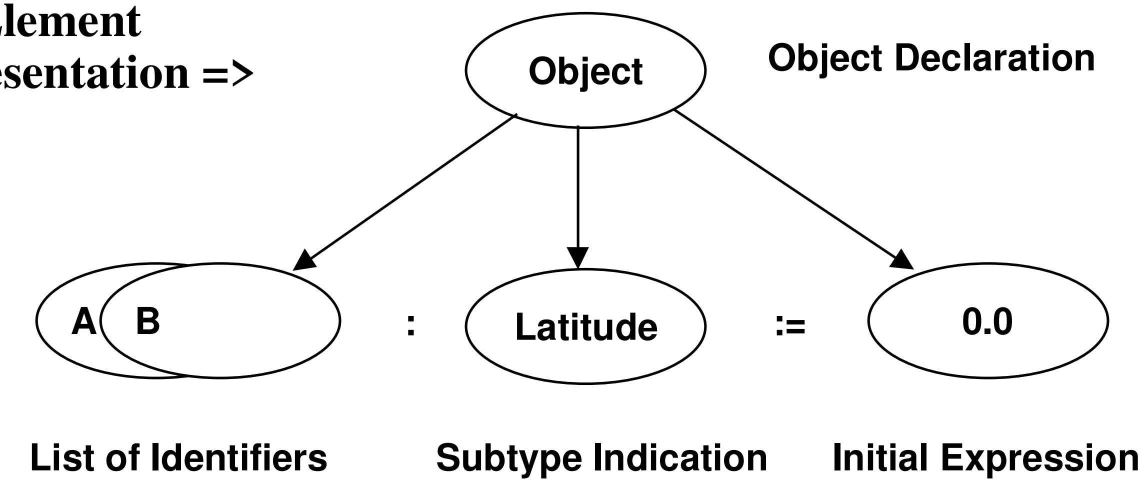
Syntactic Information

Ada syntax is summarized in Ada 95 RM, Annex P as variant of Backus-Naur Form

For example:
object_declaration ::==
defining_identifier_list : [**aliased**] [**constant**] subtype_indication [:= expression]; **|** ...

For the Ada object declaration => **A,B: Latitude := 0.0;**

Syntactic Element
Tree Representation =>

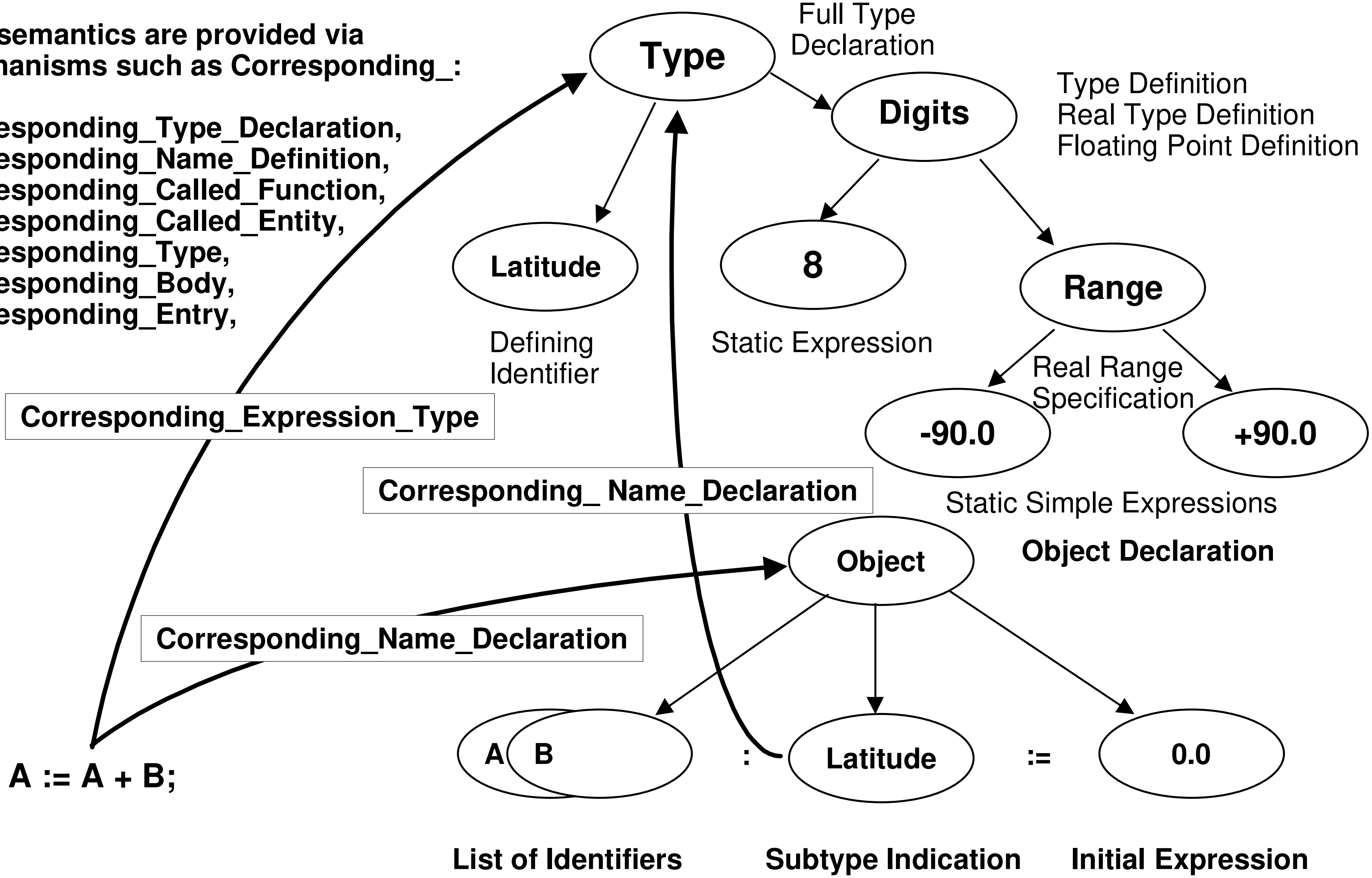


ASIS can extract desired syntactic information for every syntactic category
Of the 367 ASIS Queries, most support syntactic tree analysis

Semantic Information

Ada semantics are provided via mechanisms such as Corresponding_:

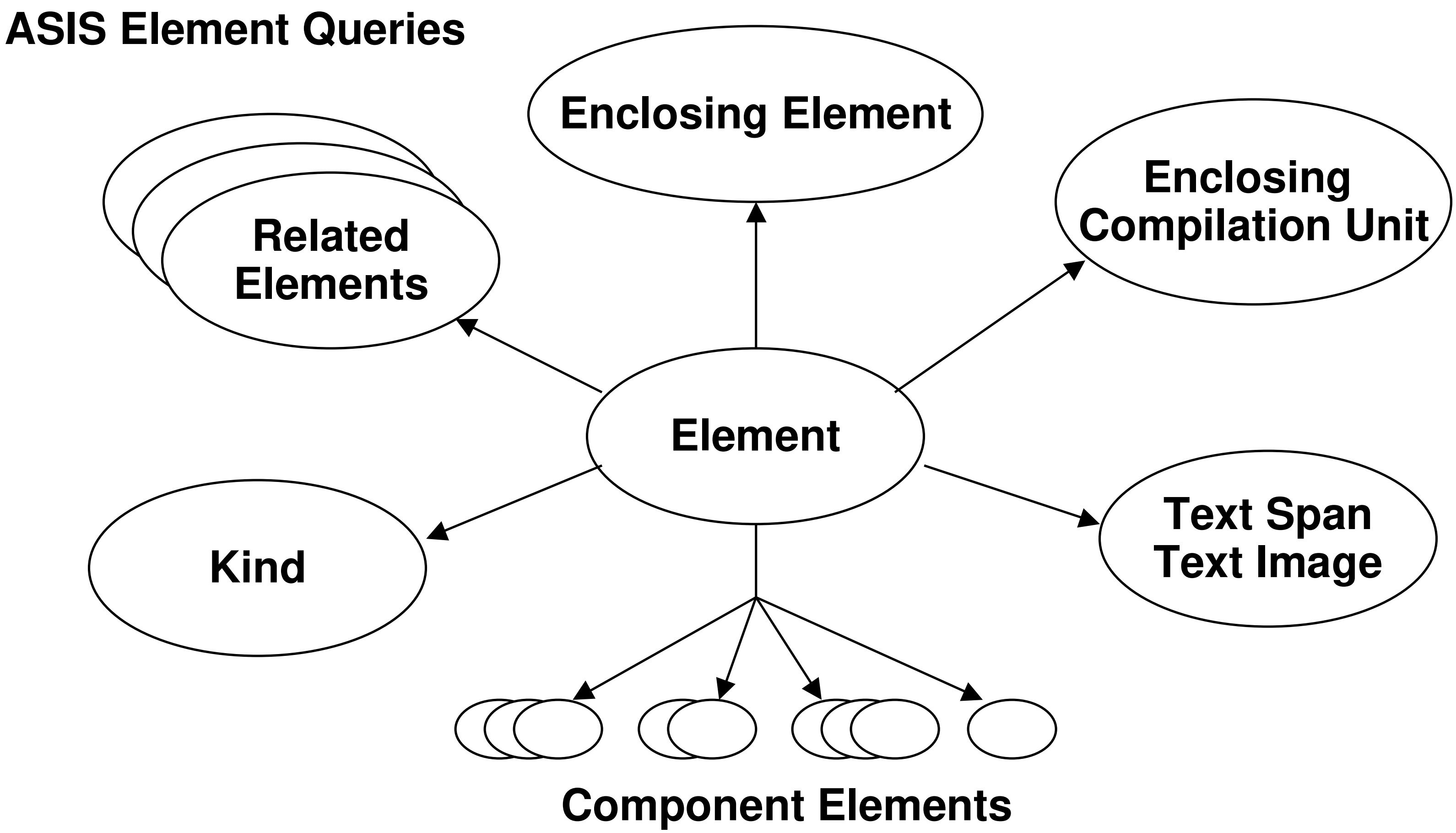
**Corresponding_Type_Declaration,
Corresponding_Name_Definition,
Corresponding_Called_Function,
Corresponding_Called_Entity,
Corresponding_Type,
Corresponding_Body,
Corresponding_Entry,
etc.**



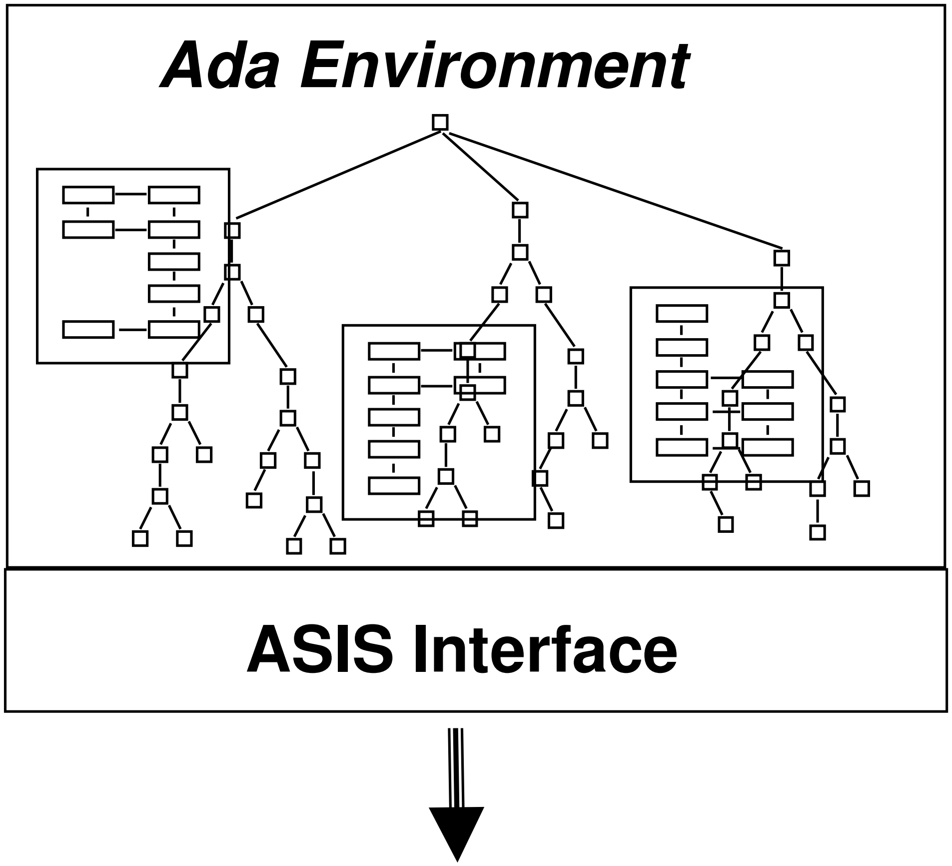
These mechanisms allow ASIS to traverse the syntactic tree like Hypertext allows one to traverse a document

Operations on Elements

Element. A common abstraction used by ASIS to denote the syntax components (both explicit and implicit) of ASIS compilation units.



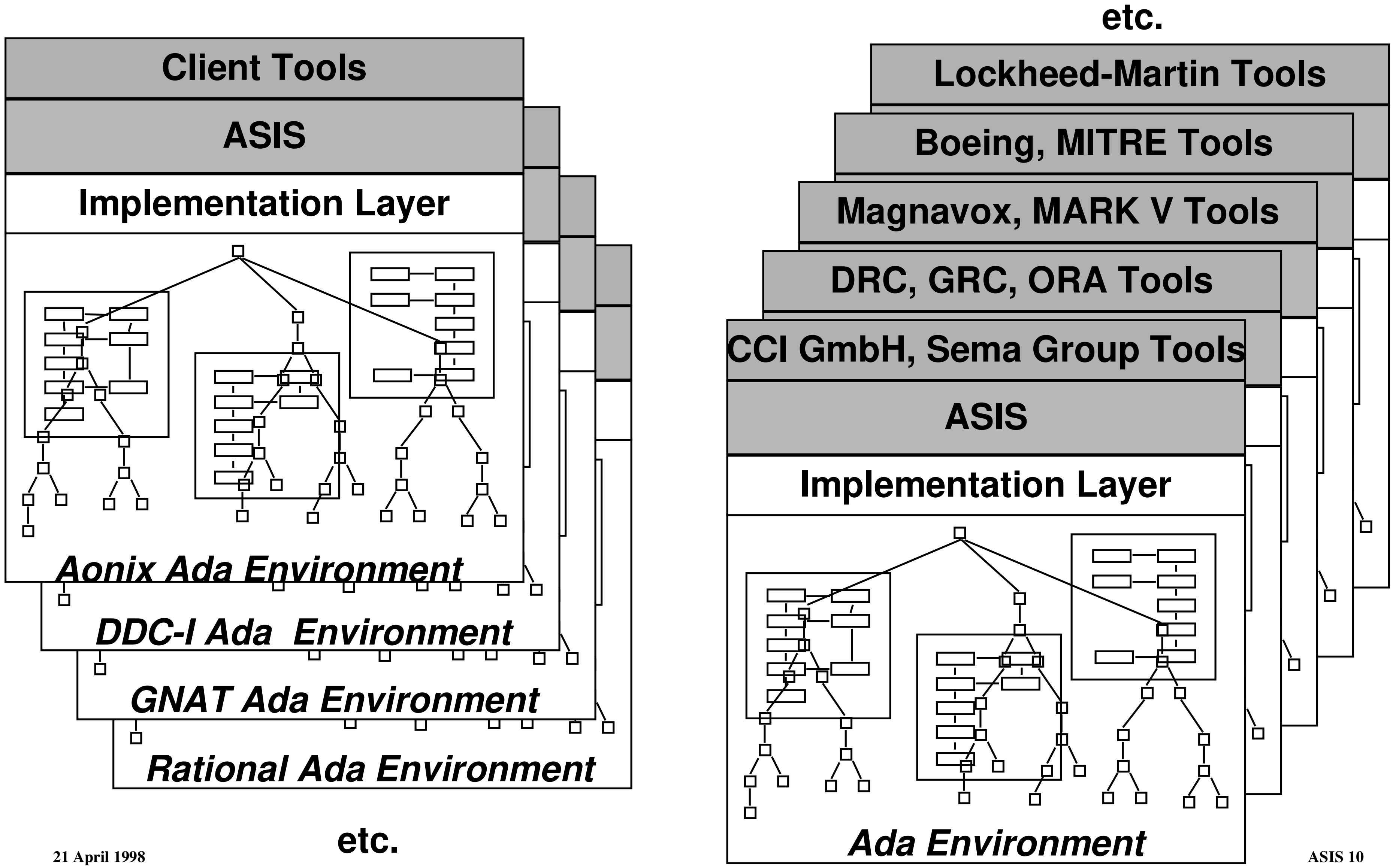
Examples of Tools Built on ASIS



Tools portable to Ada environments supporting ASIS Interface

Code Restructuring Tools
Code Browsing and Navigation Tools
Coding Style and Standards Compliance Tools
Cross Reference Tools
Data Flow Analysis Tools
Dependency Tree Analysis Tools
Design Tools
Document Generation Tools
Invocation (Call) Tree Analysis Tools
Language-sensitive Editing and Prettyprinting Tools
Language Translation Tools
Quality Assessment Tools
Reverse Engineering Tools
Re-Engineering Tools
Safety & Security Compliance Tools
Static Correctness Verifiers
Tasking Analysis Tools
Test-case Generation & Coverage Analysis Tools
Usage, Quality, & Complexity Metrics Tools

ASIS Lets Client Tools "Snap On" to Compilation Systems



ASIS Usage

Compiler Implementers:

- Aonix*
- Concurrent *
- DDC-I (Ada95)
- GNAT (Ada95)
- Green Hills *
- Intermetrics *
- OC Systems *
- Rational *

Tool Vendors:

- CCI GmbH
- DRC
- Genoa Software
- GRC
- Little Tree Consulting (Ada95)
- Mark V (Ada95)
- ORA Corp
- Rational
- Simulog
- SoHar
- WPL Labs
- Xinotech

End Users:

- Air Force
- Boeing
- FAA
- IBM
- Lockheed-Martin
- Logicon
- Loral
- Magnavox
- MITRE
- Navy
- Sema Group
- Unisys

+
HRG

Countries:

- Australia
- Belgium **
- Canada
- China
- Czech Republic **
- Denmark
- Egypt **
- Finland **
- France
- Germany
- Ireland **
- Japan
- Netherlands **
- Norway **
- Russian Federation
- Sweden
- Switzerland
- Ukraine **
- United Kingdom
- United States

* Ada 95 in near future

** Usage not known, but voted approval for ASIS CD Final on ISO/IEC JTC1/SC22 Ballot

ASIS Abstractions - Package ASIS

Ada Semantic Interface Specification (ASIS)

Package Asis provides:

- **Common types:**
  ASIS_Integer, ASIS_Natural, ASIS_Positive,
  List_Index,
  Context,
  Element, Element_List, Element Subtypes,
  Element Kinds (collection of enumeration types),
  Compilation_Units, Compilation_Unit_List,
  Unit Kinds (collection of enumeration types),
  Traverse_Control, and
  Program_Text (subtype of Wide_String)

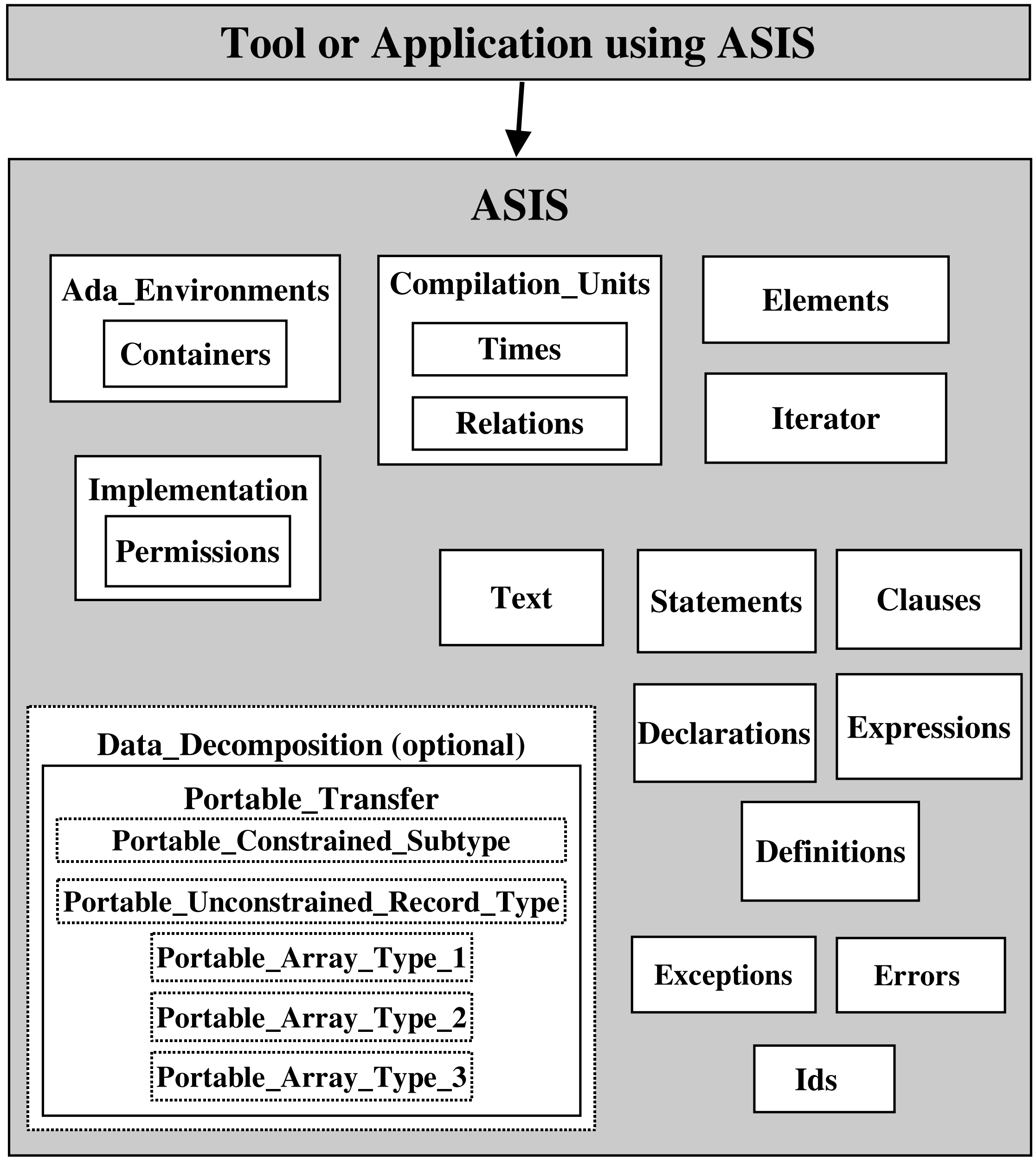
- **Queries via 20 visible child packages**

- **Ada Exceptions for errors with Status & Diagnostic information**

Asis and child packages encapsulate vendor dependencies

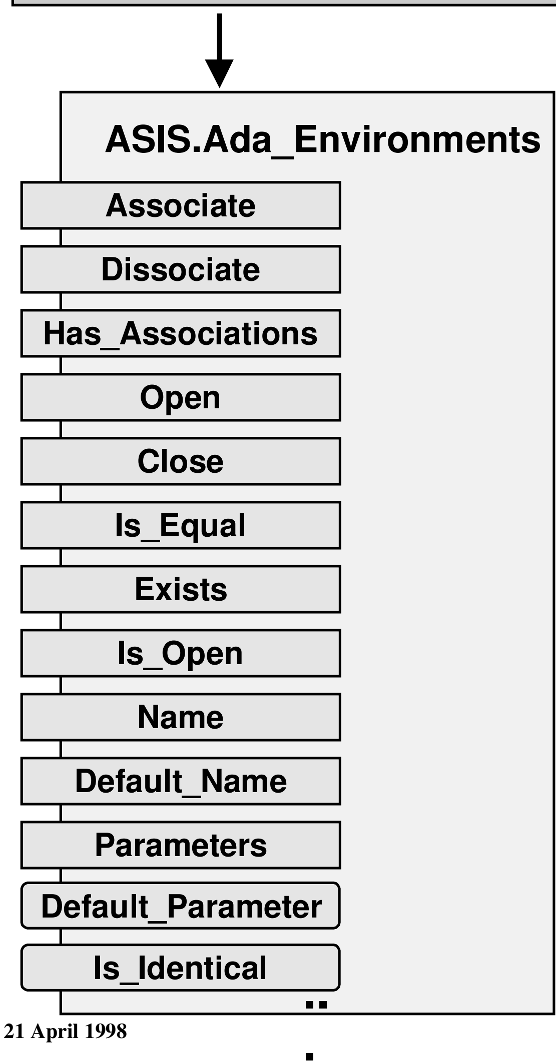
Designed to be portable for all implementations

ASIS Package Architecture



ASIS Abstractions - Ada_Environments

Ada Semantic Interface Specification (ASIS)



ASIS Context identifies an Ada Environment as defined by ISO/IEC 8652:1995 [Ada 95] which allows implementations to define methods to enter Compilation Units into the environment

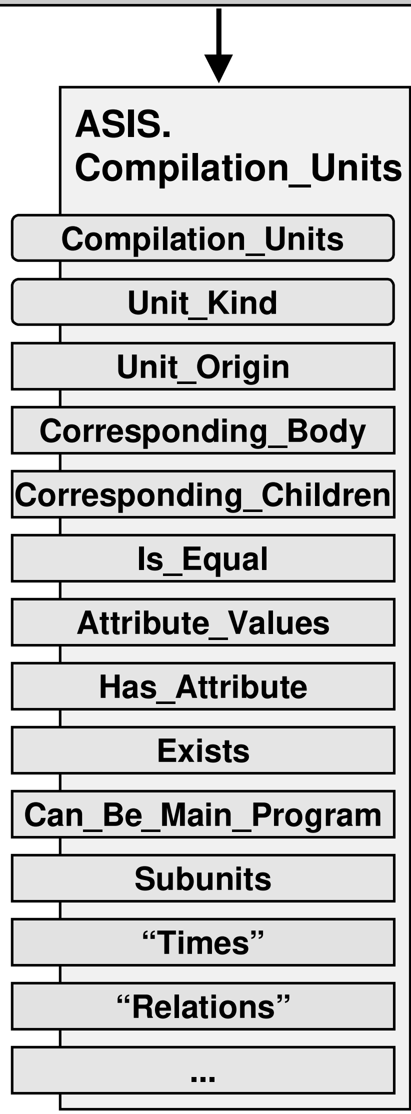
Environment Model:

- **Associates name and parameters**
- **Open, Query, and Close**
- **Provides analysis**

  free of vendor details and assumptions

ASIS Abstractions - Compilation Units

Ada Semantic Interface Specification (ASIS)



Provides external “Black Box” view

- **Attributes:**

  Date compiled, Text File, Compilation Options, Optimizations, Can Be a Main Program

- **Relationships:**

  Ancestors, Descendants, Supporters, Dependents, Family, Needed_Units

Compilation Unit Model

- **Fetch by name**
- **Query of attributes & relationships**
- **Gateway to internal view using Elements**

ASIS Abstractions - Elements

Ada Semantic Interface Specification (ASIS)

...

Asis.Elements
...
Asis.Clauses
...
Asis.Declarations
...
Asis.Definitions
...
Asis.Expressions
...
Asis.Statements
...
Asis.Exceptions
...
Asis.Iterator
..

Provides internal “White Box” view

- **Logical handle to Ada elements:**

  Declarations, Statements, Expressions, Type Definitions, With Clauses ...

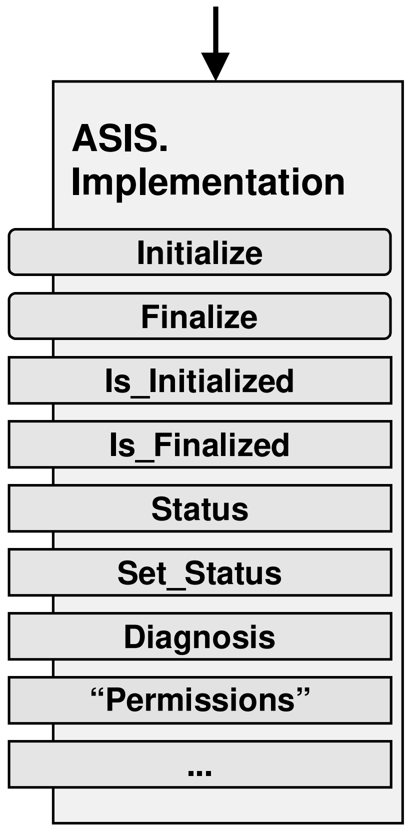
Element Model

- **Element kinds**
- **Syntactical queries to classify and decompose syntactic elements**
- **Semantic queries**
- **Elements know their enclosing Compilation Unit & Context**

Element. A common abstraction used by ASIS to denote the syntax components (both explicit and implicit) of ASIS compilation units.

ASIS Abstractions - Implementation

Ada Semantic Interface Specification (ASIS)



Using ASIS:

```
Asis.Implementation.Initialize;                  -- To setup environment
...
-- Determine ASIS implementation permissions

P := Asis.Implementation.Permissions.Is_Line_Number_Supported;
...
Asis.Ada_Environments.Associate (...);  -- To name Ada context
Asis.Ada_Environments.Open (...);        -- To gain (library) access
...
 -- Use various ASIS interfaces
...
Asis.Ada_Environments.Close (...);       -- To close (library) access

Asis.Ada_Environments.Disassociate (...);  -- To release context

Asis.Implementation.Finalize;                    -- To release all resources
```

Analysis Characteristics of Mission-Critical Systems

Typical Analysis Characteristics:

- Large-Scale Software Systems
- Analyses performed throughout the product lifecycle
- Often unique, non-standard analyses
- Inconsistent/Incomplete Documentation
- Wide variety of compilation platforms
- Development and analysis platforms often are different

We cannot depend on commercial products to support these requirements

We need the ability to develop, enhance and maintain custom software analysis applications

ASIS provides strong support for the development of such new analyses

Sample Asis-based Application: Coding Guideline Restrictions Checker

- Examines all compilation units in an Ada Environment
- Looks for violations of two guidelines:
  - Short Circuit Operators are always Used
    (i.e., OR ELSE & AND THEN are used;
    OR & AND are not used)
  - Tasks are declared at the library level
- Reports the violation and the line number of violation

But first an application template to insert the restrictions checks

This template will be useful for most ASIS applications

Restrictions Check Example: Application Template (1 of 3)

```
with Asis;
with Asis.Implementation;
with Asis.Ada_Environments;
with Unit_Processing_Templates;
with Restrictions_Check;

procedure My_Application is
    My_Context : Asis.Context;
    procedure Check is new
      Unit_Processing_Templates.Process_Application_Units_Without_State
        ( Process => Restrictions_Check.Find_Violations );

begin
    Asis.Implementation.Initialize;
    Asis.Ada_Environments.Associate (My_Context, "");
    Asis.Ada_Environments.Open (My_Context);

    Check ( My_Context );

    Asis.Ada_Environments.Close (My_Context);
    Asis.Ada_Environments.Dissociate (My_Context);
    Asis.Implementation.Finalize;
end My_Application;
```

Restrictions Check Example: Application Template (2 of 3)

```
with Asis;

package Unit_Processing_Templates is

Generic

  with Procedure Process( Unit : in Asis.Compilation_Unit ) is <>;

procedure Process_Application_Units_Without_State
   (Context : Asis.Context );

...

end Unit_Processing_Templates;
```

Restrictions Check Example: Application Template (3 of 3)

```
with Asis;
with Asis.Compilation_Units;
package body Unit_Processing_Templates is

procedure Process_Application_Units_Without_State
             (Context : Asis.Context ) is
    Unit_List : Asis.Compilation_Unit_List :=
        Asis.Compilation_Units.Compilation_Units (Context);
begin
    for I in Unit_List'Range loop
        case Asis.Compilation_Units.Unit_Origin (Unit_List (I)) is
           when Asis.An_Application_Unit =>
                Process (Unit_List (I));
           when others => null;
        end case;
    end loop;
end Process_Application_Units_Without_State;
...
end Unit_Processing_Templates;
```

Restrictions Check Example: Package Restrictions_Check

```
with Asis;
package Restrictions_Check is

    procedure Find_Violations (CU : in Asis.Compilation_Unit);

end Restrictions_Check;
```

Restrictions Check Example: Package Body

```
with Asis; with Asis.Elements; with Asis.Iterator; with Asis.Text;
with Wide_Text_Io; use Wide_Text_Io;
package body restrictions_check is

    procedure Process_Element (Elem : in Asis.Element;
                               Control : in out Asis.Traverse_Control;
                               Dummy   : in out boolean);

    procedure No_Op (Elem : in Asis.Element;
                     Control : in out Asis.Traverse_Control;
                     Dummy   : in out boolean);

    procedure Check is new Asis.Iterator.Traverse_Element
                               (boolean, Process_Element, No_Op);

    Procedure Find_Violations (CU : Asis.Compilation_Unit) is
        Control : Asis.Traverse_Control := Asis.Continue;
        Dummy   : boolean;
    begin
        Put_Line ("Processing " &
        Asis.Unit_Kinds'Image (Asis.Compilation_Units.Unit_Kind(CU))
        & ": " &  (Asis.Compilation_Units.Unit_Full_Name(CU)));
        Check(Asis.Elements.Unit_Declaration (CU), Control, Dummy);
    end;
```

Restrictions Check Example: Process_Element

```
procedure Process_Element (Elem    : in Asis.Element;
                           Control : in out Asis.Traverse_Control;
                           Dummy   : in out boolean) is

 begin

     Check_Short_Circuit(Elem);

     Check_Library_Level_Task(Elem);

 end Process_Element;
```

Restrictions Check Example: Short Circuit Operators

```
procedure Check_Short_Circuit( Elem : in Asis.Element) is
       Op_Kind : Asis.Operator_Kinds :=
       Asis.Elements.Operator_Kind (Elem);

begin
    case Op_Kind is

    when Asis.An_And_Operator =>
       Put_Line("Violation of Short Circuit Operator guideline:");
       Put("-- Use of AND Operator at line ");
       Put ( Asis.Text.Line_Number'Wide_Image                      -- 20.2
                            (Asis.Text.First_Line_Number (Elem)));   -- 20.8
       New_Line;
    when Asis.An_Or_Operator =>
       Put_Line("Violation of Short Circuit Operator guideline:");
       Put("-- Use of OR Operator at line ");
       Put ( Asis.Text.Line_Number'Wide_Image                      -- 20.2
                 (Asis.Text.First_Line_Number (Elem)));            -- 20.8
       New_Line;
    when others =>
               null;
    end case;

end Check_Short_Circuit;
```

Restrictions Check Example: Tasks at Library Level (1 of 2)

```
procedure Check_Library_Level_Task (Elem : Asis.Element) is
    begin
      case Asis.Elements.Declaration_Kind (Elem) is

          when Asis.A_Task_Type_Declaration |
               Asis.A_Protected_Type_Declaration |
               Asis.A_Single_Task_Declaration |
               Asis.A_Single_Protected_Declaration =>
                  If not Is_Library_Level
                      (Asis.Elements.Enclosing_Compilation_Unit(Elem))
then
                    Put_Line("Violation of Tasking guideline:");
                    Put("-- Non-Library Level Task at Line:");
                    Put ( Asis.Text.Line_Number'Wide_Image
                        (Asis.Text.First_Line_Number (Elem)));
                    New_Line;
                  end if;
          when others =>
              null;
         end case;

    end Check_Library_Level_Task;
```

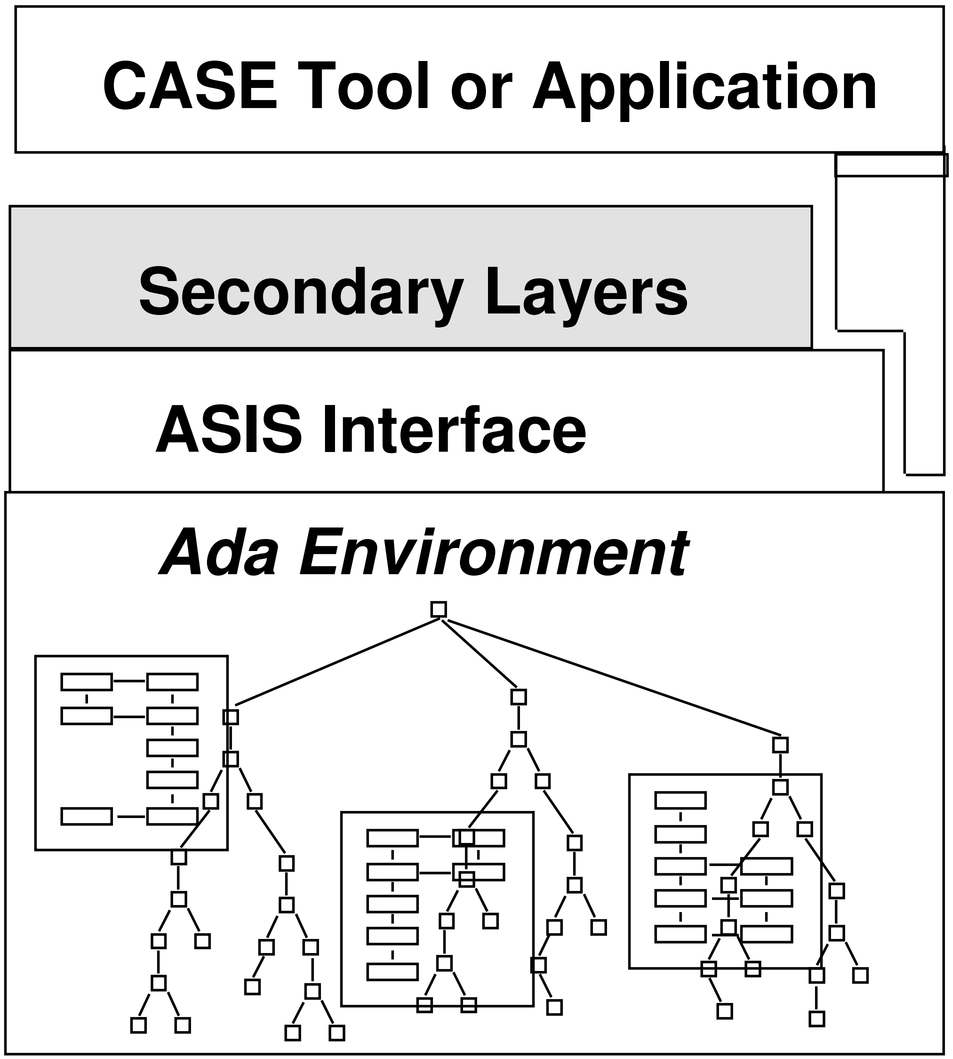
Restrictions Check Example: Tasks at Library Level (2 of 2)

```
function Is_Library_Level ( CU : Asis.Compilation_Unit ) return boolean is
    begin

      Case Asis.Compilation_Units.Unit_Class(CU) is
          when Asis.A_Public_Declaration =>
              return true;
          when others =>
              return false;
      end case;

   end;
```

ASIS Secondary Layers



- Widget -like layer of abstractions that model commonly used views of Ada program structure
- Each view provided in set of packages with operations for constructing and querying views
- Offer CASE tool builders same kind of benefits enjoyed by X-users:

  Productivity
  Consistent results
  Reliability
  Portability

- ASIS 83 Secondary layers available:

  Scanners
  Namespace View
  Reference View
  Region View
  Control Flow View

- Work is in progress for ASIS95

How to Get ASIS Artifacts

ASIS Home Page =>

http://www.acm.org/sigada/WG/asiswg

Also mirrored on sw-eng host for ftp on Internet =>

sw-eng.falls-church.va.us

ASIS 95 specification is complete and ready for use:
(Microsoft Word, postscript, ASCII, & HTML)

ASIS is a Registered ISO Committee Draft (CD) and has
has passed the Final CD Ballot

Also available are tutorials, papers, examples, bibliography

ASIS Schedule

Dec 93	AJPO recommends ASIS V1.1.0 (ASIS83) be used as interface to Ada 83 Program Library
Mar 94	Design Goals for ASIS95 identified
Jun 94	ASISWG finalizes ASIS 83 as V1.1.1 with test suite
Jun 94	Evaluate design approaches for ASIS 95
Nov 94	Finalize approach for ASIS 95
Mar 95	Skeleton ASIS 95
28 Apr 95	ASISRG created unanimously by ISO/IEC JTC1/SC22 WG9
Jun 95	Skeleton ASIS 95; new library/environment model
Nov 95	Public Review of ASIS 95 initiated (Version 2.0.E)
2 Feb 96	ISO/IEC JTC1/SC22 approved NWI
Feb 96	ASIS available for GNAT Ada 95 Compiler
Mar 96	ASISRG vote to submit ASIS 95 as ISO Working Draft
Jun 96	ISO/IEC JTC1 approval of NWI - ASIS assigned ISO number 15291
Dec 96	WG9 ballot on ASIS 95 Working Draft commenced
1 Mar 97	WG9 ballot on ASIS completed (6 Affirmatives; 1 Abstain; 0 Negatives)
1 May 97	Disposition of Comments delivered to WG9
2 Jun 97	Disposition of Comments, ASIS to SC22 for Balloting approved
Aug 97	ASIS to SC22 for Registration as ISO CD& CD Final Ballot
9 Mar 98	ASIS approved by SC22 for CD Registration & CD Final
Sep 98	**Possible ASIS 95 approved as ISO Standard**

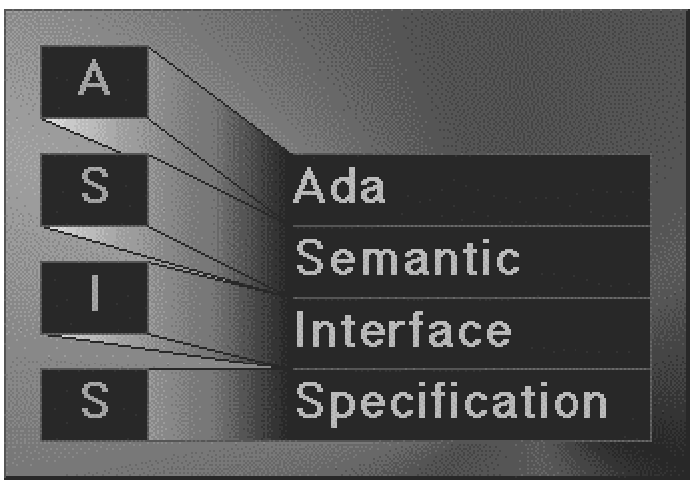
Final Notes on ASIS

ASIS - Interface for obtaining information from Ada environments

- Provides both syntactic and semantic information
- Supports building powerful and highly portable CASE tools
- Supports many requirements for code analysis

  By tool vendors AND especially by code developers

- Used in approximately 10 countries
- ASIS mature with 3 implementations today; standard ~ 98-99



To get involved with ASISWG and/or ASISRG, send email to:

asis-request@sw-eng.falls-church.va.us

to join asis@sw-eng.falls-church.va.us

asis-technical-request@sw-eng.falls-church.va.us

to join asis-technical@ sw-eng.falls-church.va.us

or call Currie Colket @ +1 (703) 242-4561

or Clyde Roby @ +1 (703) 845-6666

ASIS Home page => *http://www.acm.org/sigada/WG/asiswg*

Simple ASIS Example - 1

```
with Wide_Text_IO; use Wide_Text_IO;
with Asis;                  -- also with necessary child packages
Procedure Asis_Application_Example is

      My_Context         : Asis.Context;
      My_Unit            : Asis.Compilation_Unit;
      Unit_Name          : Wide_String ( 1 .. 100 );
      Unit_Name_Length   : Natural;

   procedure Report_Declarations (Unit : in Asis.Compilation_Unit)
   ... -- See Next Page

begin -- Asis_Application_Example

   Asis.Implementation.Initialize;
   Asis.Ada_Environments.Associate(My_Context, "My_Context");
   Asis.Ada_Environments.Open ( My_Context );

   Put_Line ("Type the name of an Ada package specification");
   Get_Line (Unit_Name, Unit_Name_Length);

   My_Unit := Asis.Compilation_Units.Library_Unit_Declaration
             ( Unit_Name ( 1 .. Unit_Name_Length), My_Context );

   if Asis.Compilation_Units.Is_Nil ( My_Unit )
   then
      Put ("Context does not contain the requested unit ");
      New_Line;
   else
      Put ("Context contains the requested unit "); New_Line;
      Report_Declarations ( My_Unit ); New_Line;
   end if;

   Asis.Ada_Environments.Close( My_Context );
   Asis.Ada_Environments.Dissociate( My_Context );
   Asis.Implementation.Finalize;

end  Asis_Application_Example;
```

```
For =>

package Asis_Test is

  type T is ( A, B, C);

  S : integer := T'BASE'SIZE ;

end Asis_Test ;
```

Simplified Example to ASIS 2.0.Q

```
Result =>

Type the name of an Ada package specification
Asis_Test
Context contains the requested unit
Asis_Test (is kind) A_PACKAGE_DECLARATION
T (is kind) AN_ORDINARY_TYPE_DECLARATION
A (is kind) AN_ENUMERATION_LITERAL_SPECIFICATION
B (is kind) AN_ENUMERATION_LITERAL_SPECIFICATION
C (is kind) AN_ENUMERATION_LITERAL_SPECIFICATION
S (is kind) A_VARIABLE_DECLARATION
```


Example prompts user for name of Ada package specification, traverses that compilation unit, and prints all explicit declarations and their kind.

Simple ASIS Example - 2

```
procedure Report_Declarations (Unit : in Asis.Compilation_Unit) is

     My_Element              : Asis.Element;
     My_Control              : Asis.Traverse_Control := Asis.Continue;
     My_State                : Boolean := True;


     procedure Process_Element
                   (Elem : in Asis.Element;
                    Control : in out Asis.Traverse_Control;
                    State : in out Boolean);

     procedure No_op
                   (Elem : in Asis.Element;
                    Control : in out Asis.Traverse_Control;
                    State : in out Boolean);


     procedure Find_and_Print_Declarations is new
                    Asis.Iterator.Traverse_Element
                      (Boolean, Process_Element, No_Op);

     procedure No_op
                    (Elem : in Asis.Element;
                     Control : in out Asis.Traverse_Control;
                     State : in out Boolean) is
     begin
        null;
     end;

     procedure Process_Element ... -- See Next page

begin -- Report_Declarations

     My_Element   := Asis.Elements.Unit_Declaration ( Unit );
     Find_and_Print_Declarations (My_Element, My_Control, My_State);

end Report_Declarations;
```

Generic Procedure Traverse_Element
3 generic parameters:
1. **Type for State Info**
2. **Pre-Operation**
3. **Post_Operation**

Traverses Element & all component elements

Simple ASIS Example - 3

```
procedure Process_Element
        Elem : in Asis.Element;
        Control : in out Asis.Traverse_Control;
        State : in out Boolean) is

    package Kind_Io is new Text_Io.Enumeration_Io (Asis.Declaration_Kinds);
    Decl_Kind : Asis.Declaration_Kinds := Asis.Elements.Declaration_Kind (Elem);

begin -- Process_Element

    case Decl_Kind is

        when Asis.Not_A_Declaration => null;

        when others =>

            declare

                Name_List : Asis.Defining_Name_List  := Asis.Declarations.Names (Elem);

            begin

                for I in Name_List'Range loop
                    Put (Asis.Declarations.Defining_Name_Image (Name_List (I)));
                    Put (" (is kind) ");
                    Kind_Io.Put (Decl_Kind);
                    New_Line;
                 end loop;
            end;

    end case;

end Process_Element;
```

```
For =>

package Asis_Test is

   type T is ( A, B, C);

   S : integer := T'BASE'SIZE ;

end Asis_Test ;
```

```
Result =>

Type the name of an Ada package specification
Asis_Test
Context contains the requested unit
Asis_Test (is kind) A_PACKAGE_DECLARATION
T (is kind) AN_ORDINARY_TYPE_DECLARATION
A (is kind) AN_ENUMERATION_LITERAL_SPECIFICATION
B (is kind) AN_ENUMERATION_LITERAL_SPECIFICATION
C (is kind) AN_ENUMERATION_LITERAL_SPECIFICATION
S (is kind) A_VARIABLE_DECLARATION
```

Example is missing context clauses & exception handler

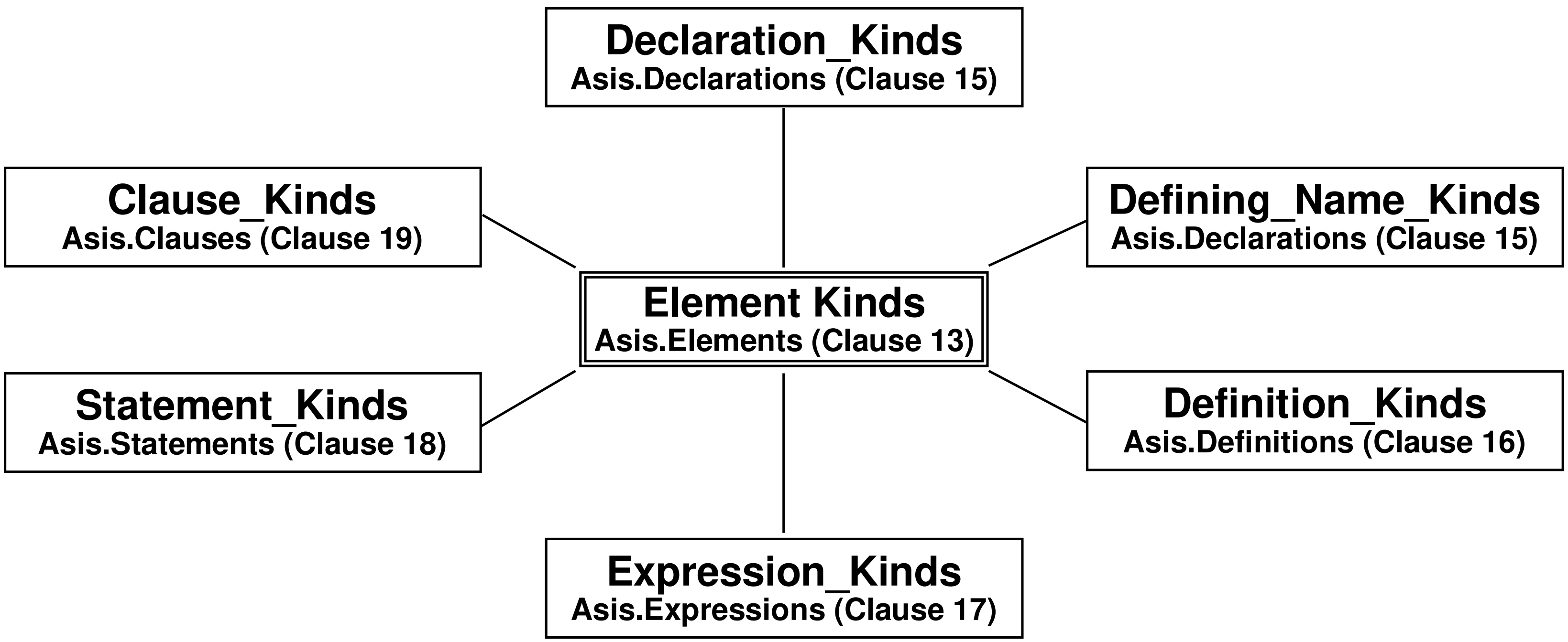
See ASIS Specification, Annex B.1 for complete compilable example

Simple ASIS Example - 4

Decl_Kind from example is of type Declaration_Kinds defined in Subclause 3.9.4
Queries on Declaration_Kinds are found in Asis.Declarations (Clause 15)

General Element processing queries in Asis.Elements (Clause 13)

Similar syntactic processing can be performed on other Element Kinds defined in 3.9:



Ada Board Resolution About ASIS

Motion at the Ada Board 10-11 September 1992:

> The Ada Board recognizes the potential benefits to the DOD Ada community of an ASIS standard and therefore it recommends that the AJPO director support by whatever means possible, the development of an ASIS standard and its submission to ISO/WG9 for publication.

Vote: Passed unanimously

ACM ASIS Working Group established to develop ASIS for Ada 83 and Ada 95 with ISO WG9 ASIS Rapporteur Group for standardization of ASIS for Ada 95

ACM's SIGAda's
ASIS Working Group Charter

The purpose of this working group is to develop and participate in standardizing an implementation independent application programming interface to retrieve information from an Ada environment.

The Ada Semantic Interface Specification (ASIS) is an interface between an Ada environment (as defined by ISO/IEC 8652:1995) and any tool requiring information from it. An Ada environment includes valuable semantic and syntactic information. ASIS is an open and published callable interface which gives CASE tool and application developers access to this information. ASIS has been designed to be independent of underlying Ada environment implementations, thus supporting portability of software engineering tools while relieving tool developers from needing to understand the complexities of an Ada environment's proprietary internal representation.

The working group is currently developing ASIS for Ada 95, and is working towards its adoption as an international standard. The working group successfully developed ASIS for Ada 83, which is now complete and in the public domain.

Membership in the working group is open to any interested party. Members are responsible for their own expenses. Current members include representatives of Ada compiler implementors, CASE tool developers, application developers, and other interested users.

Original February 8, 1993, Updated June 28, 1994, Updated November 7, 1995

ASISRG-1

ISO/IEC JTC1/SC22 WG9 ***voted unanimously to create ASISRG on 28 April 1995***

Scope

Standardize the interface between an Ada 95 compilation environment and tools/applications requiring information from this environment.

Purpose and Justification

The Ada Semantic Interface Specification (ASIS) is an interface between an Ada environment as defined by ISO/IEC 8652 and any tool requiring information from this environment. An Ada environment includes valuable semantic and syntactic information. ASIS is an open and published callable interface which gives CASE tool and application developers access to this information. ASIS has been designed to be independent of underlying Ada environment implementations, thus supporting portability of software engineering tools while relieving tool developers from having to understand the complexities of an Ada environment's proprietary internal representation.

ASISRG-2

Purpose and Justification (Continued)

As an international standard, ASIS will benefit the Information Technology community by facilitating the development of powerful CASE tools portable amongst the various environments provided by Ada vendors. This portability can only be achieved through the standardization of ASIS at the international level. A standardized ASIS will promote the development of powerful tools for the software engineering environment by providing access to important semantic information otherwise available only through proprietary interfaces. Further, ASIS will benefit the Information Technology community as a valuable resource for application development (e.g., decoupling system to system interfaces). The international standardization of ASIS will facilitate the use of this important capability in the development of system software applications.

ISO/IEC JTC1 assigned ASIS project number 15291 in May 1996

Successful Ballot for ISO Committee Draft in March 1998

ASISWG/ASISRG Officers

Chair ASISWG/ASISRG:	Currie Colket (SPAWAR)
Vice-chair ASISWG:	Steve Blake (Aonix)
Recorder ASISWG:	Clyde Roby (IDA)
Vice-recorder ASISWG:	Dan Cooper (Boeing)
Publicity/Meetings ASISWG:	Bill Thomas (MITRE)
ASISRG Co-Project Editors:	Steve Blake (Aonix) Clyde Roby (IDA)

ASIS Interface - Funding

FY88 & FY89 Funding provided by STARS

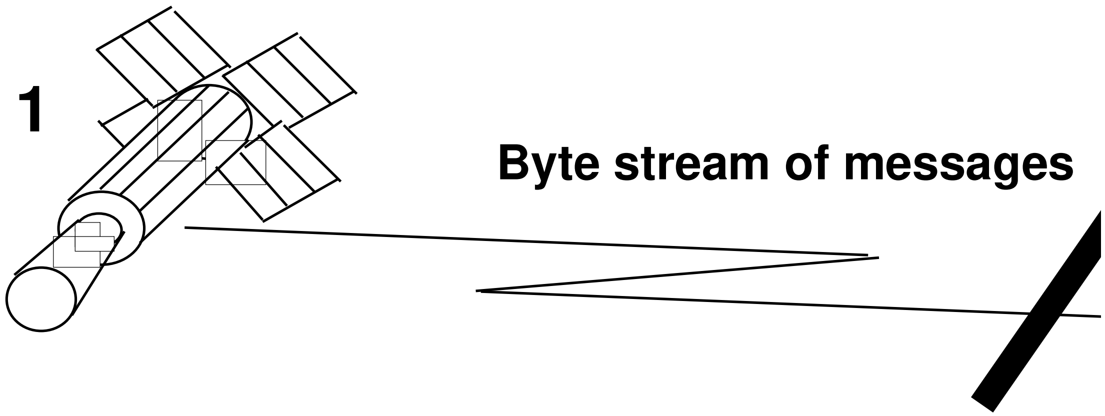
Later in FY89, STARS initiated a policy not to fund any standardization efforts and ASIS funding was halted along with all other standardization activities

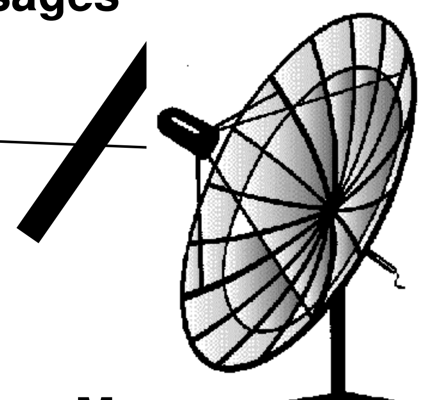
There has been no STARS, no AJPO, and no DOD funding for ASIS Standardization since FY89

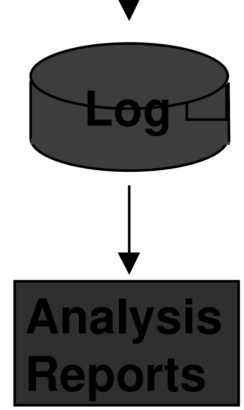
All post FY89 ASIS standardization funding has resulted from industry investment - user community working closely with the compiler implementor community

Example of Application Built on ASIS-1

System 1







Problem

How do you delog data so an analyst can view data in a meaningful way?

Alternatives

1. **Delog routine per message ID.**

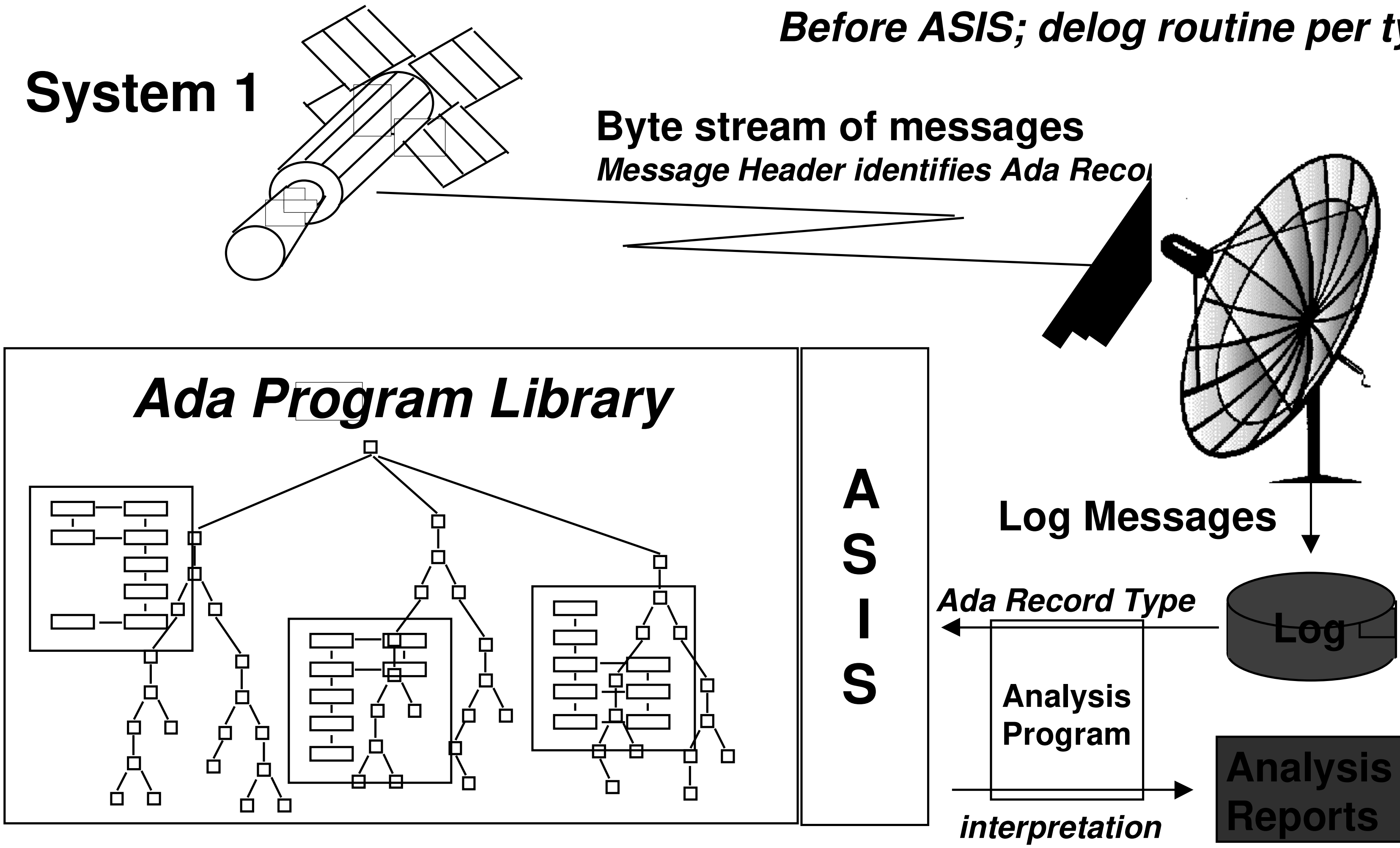
   Expensive, inflexible, must be constantly updated

2. **Universal delogger, able to interpret any message ID.**

   Effective, highly flexible, never needs updating

Example of Application Built on ASIS-2

Before ASIS; delog routine per type



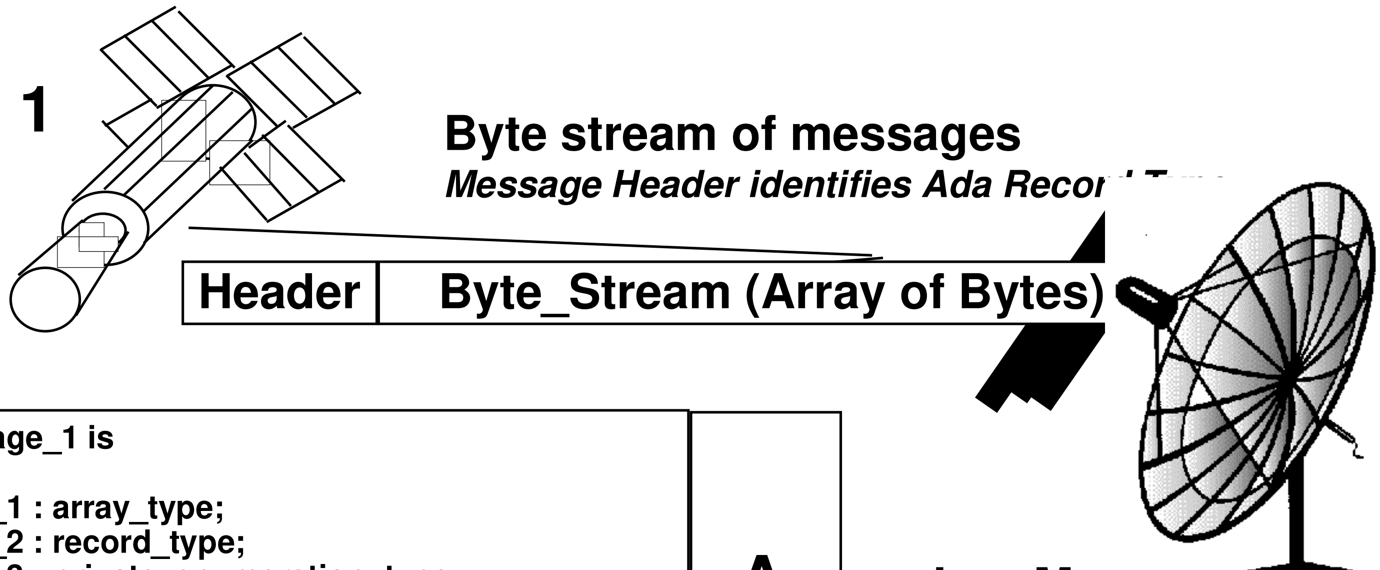
With ASIS; universal delogger, able to interpret any type;
Time reduced to build type dictionary from 704 hours to 2 hours

Example of Application Built on ASIS-3

System 1

Byte stream of messages

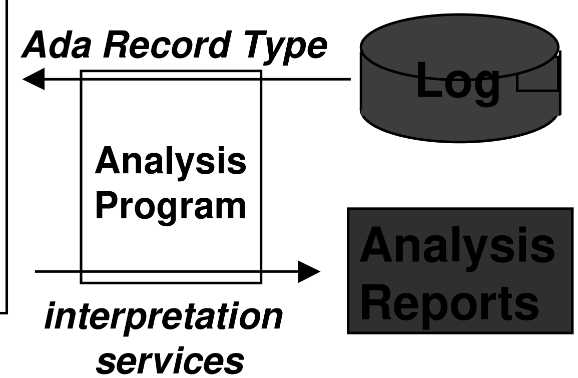
Message Header identifies Ada Recor




```
    type message_1 is
        record
            field_1 : array_type;
            field_2 : record_type;
            field_3 : private_enumeration_type;
        end record;
    type message_2 (discrim_2: integer) is private;

private
    subtype  range_type is positive range 10 .. 20;
    type varying_array_type is (integer range <> of float);
    type message_2  (discrim_2: integer);
        record
            field_1 : range_type;
            field_2 : varying_array_type (1 .. discrim_2);
        end record;
```

A S I S

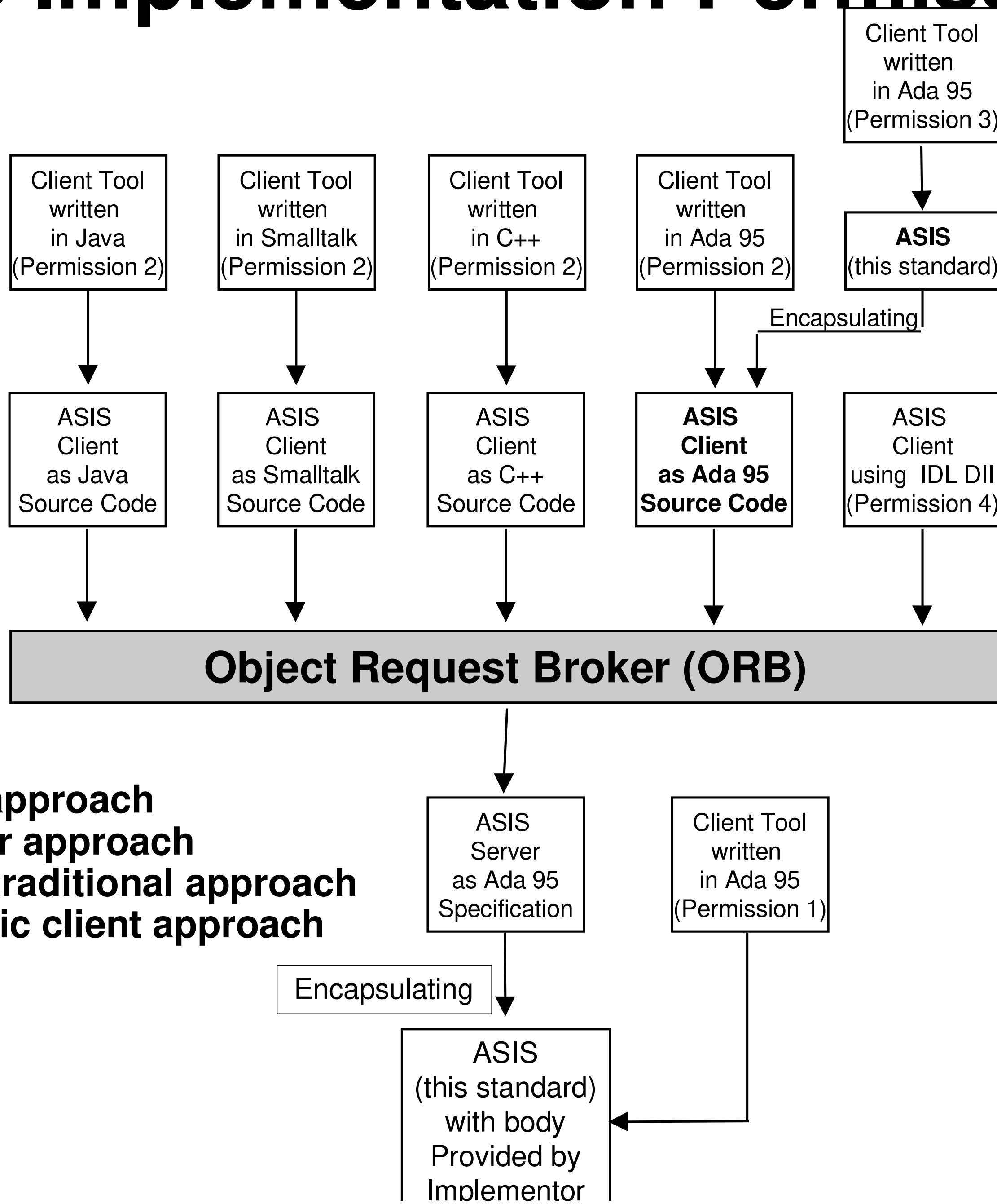


ASIS provides ‘size, ‘position, ‘first_bit, ‘last_bit

Services index Byte_Stream to primitive types using Unchecked_Conversion

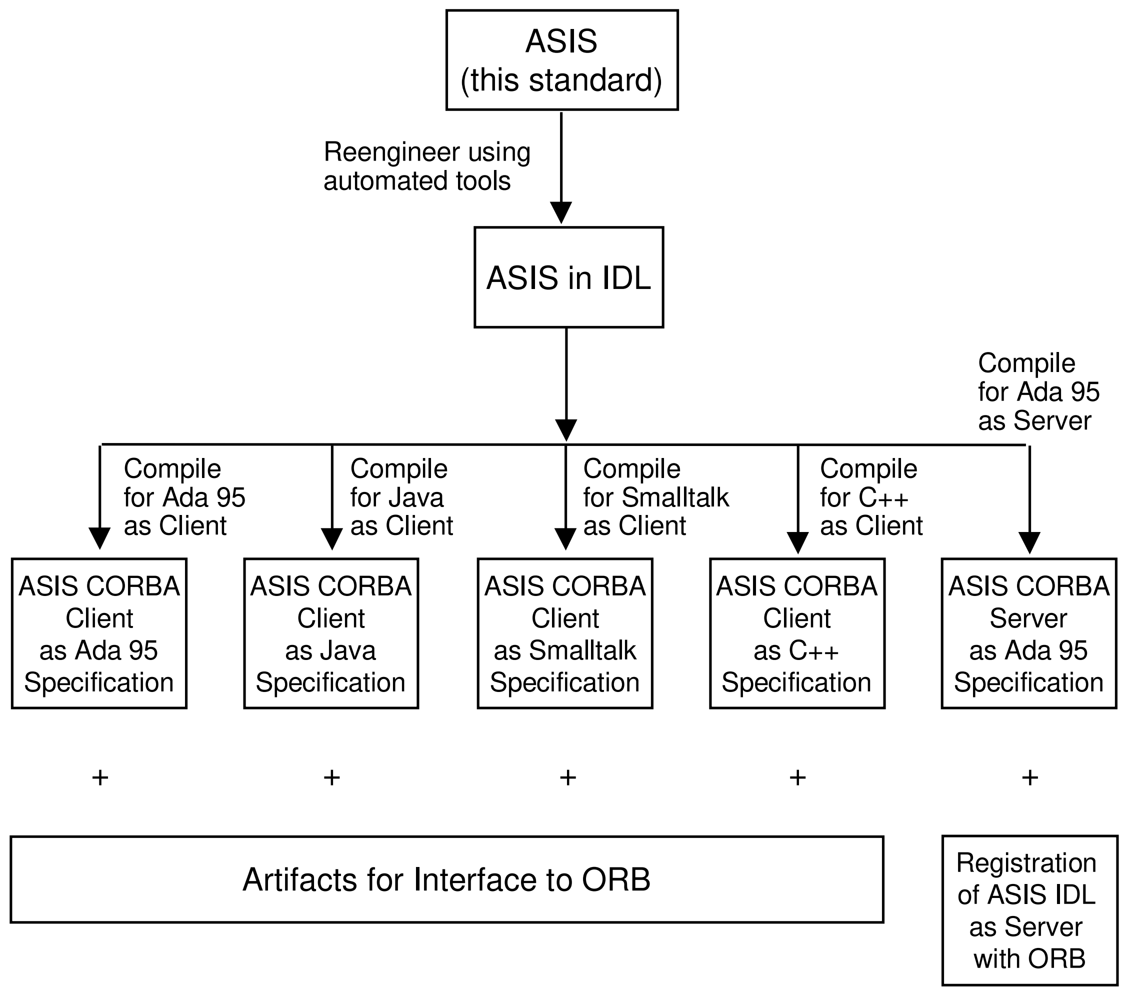
Services return values for analysis as variant record discriminated by ASIS Type_Kind

ASIS Implementation Permissions



Permissions:
1. **Traditional approach**
2. **Client/Server approach**
3. **Distributed traditional approach**
4. **ASIS dynamic client approach**

Creating an ASIS IDL



ASIS-for-GNAT Implementation

GNAT, the GNU Ada 95 compiler, originally developed at NYU, is distributed and maintained by Ada Core Technologies (ACT): available at: *http://www.gnat.com*

- ASIS for GNAT was developed by:
  - Dr. Sergey Rybin (Moscow State University)
  - Professor Alfred Strohmeier (Swiss Federal Institute of Technology)
- First free ASIS 95 implementation prototype in February 1996
- Now full ASIS implementation distributed and maintained by ACT.

ASIS has ISO Final CD Approval!!!

Concurrent ballots for Committee Draft (CD) Registration and Final CD (FCD) Approval for FCD 15291 - Information technology - Programming languages - Ada Semantic Interface Specification (ASIS) were completed on 9 March 1998.

ASIS had a successful ballot. ASIS is now a registered Committee Draft and the ASIS Version with FCD comments resolved will be a DIS.

- Fourteen nations voted to Approve FCD 15291: Belgium, Canada, Czech Republic, Egypt, Finland, France, Ireland, Japan, Netherlands, Norway, Russian Federation, UK, Ukraine, and USA.
- Four nations Abstained and five nations did not vote.
- There were no ballots voting to Disapprove.

Only Final DIS (FDIS) Ballot is needed for ASIS to be an ISO Standard.