Workshop Report
ASIS – Extensions For Higher Level Abstractions
1:30-5:30 PM, Thursday, 21 October 1999
SIGAda'99, Redondo Beach, California

The Ada Semantic Interface Specification (ASIS) Working Group (ASISWG) held a Workshop on Thursday, 21 October 1999, from 1:30-5:30 P.M. in conjunction with SIGAda'99 in Redondo Beach, California. The topic was ***ASIS Extensions for Higher Level Abstractions***. The primary purpose was to address a framework to support the development of ASIS Extensions for higher level abstractions, which would be useful to vendors, developers, and ASIS users.

The standardized ASIS interfaces reflect the low-level syntactic level of the source code. A higher level abstraction might incorporate a number of ASIS queries. Such higher level abstractions could be useful to different classes of tool developers to support specialized requirements (e.g., additional OO analyses, specialized static run-time analyses). These abstractions should be implementable using the standard ASIS interfaces.

An ASIS Workshop at SIGAda'98 raised the issue that it would be useful to have a common naming scheme to name ASIS extensions to support the development of higher level abstractions. The principal reason identified at the SIGAda'98 Workshop was to provide higher level abstractions to increase productivity and effectiveness of the ASIS tool developer. It would also encourage the sharing of useful ASIS interfaces amongst vendors, developers, and users by supporting the integration of portable ASIS interfaces from multiple sources without a name clash.

Attendees

Participants included representatives from the compiler vendor community, tool vendor community, and user community. Those who attended include:

Roy Bell, Raytheon
Steve Blake, Aonix
Todd Coniam, ESC
Steve Deller, Rational
John Harbaugh, Boeing
Tony Lowe, Adamantium
Joseph Meyer, DOD
Clyde Roby, IDA
Sergey Rybin, ACT Europe
Tucker Taft, AverStar
Anh Vo, United Defense LP
Joseph Wisniewski, Commercial Software Solutions

Background on ASIS

The primary focus of the ASISWG of ACM SIGAda has been to evolve ASIS as an interface to the Ada 95 compilation environment. ASIS now provides a powerful mechanism to perform code analysis for mission-critical, high-integrity, and safety-critical applications. A variety of highly effective tools have been built using ASIS. This interface was approved as an ISO standard in 1999 as:

ISO/IEC 15291:1999 Information Technology – Programming Languages
– Ada Semantic Interface Specification (ASIS)

Those wanting to find out more about ASIS should visit the ASIS Home Page at the following URL:

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

Overview

Currie Colket, the Chair of ASISWG, had arranged for the ASIS Workshop. Steve Blake, the Vice Chair of ASISWG, chaired the workshop as Currie Colket was recovering from surgery. Steve introduced the issue of ASIS extensions by providing a background on ASIS and discussing two types of extensions in the form of Toolkit and Extension Layers. Steve Blake presented an example of tool integration using Toolkit and Extension Layers. He presented a proposal on the use of the Toolkit and Extension Layers, which was discussed by Workshop participants. After consensus was reached on the proposal, some brainstorming addressed other concerns to the ASIS community. The slides Steve used are available at the ASIS Home Page.

Toolkit and Extension Layers

ASIS Toolkit/Extensions Layers was the primary topic of discussion. Steve Blake suggested two ways that ASIS developers could extend ASIS: through a ***Toolkit Layer*** and an ***Extensions Layer***. The *Toolkit Layer* would provide higher level, abstract secondary queries built from ASIS primitives and other portable services. It would be portable to ASIS implementations provided by ASIS vendors. Such queries would be valuable as examples to reduce the learning curve in using ASIS. The *Extensions Layer* would allow vendors to fill holes and gaps in the ASIS interface and support queries that ASIS was never intended to support such as dynamic semantics. Figure 1 depicts the relationship between the Toolkit Layer and the Extensions Layer to the ASIS Interface, the Ada Compilation Environment, and the Host or Target Environment. The Extensions Layer would be able to use queries from the Toolkit Layer, as these are portable. The converse is not true; should the Toolkit Layer use queries from the Extensions Layer, the Toolkit Layer would no longer be portable.

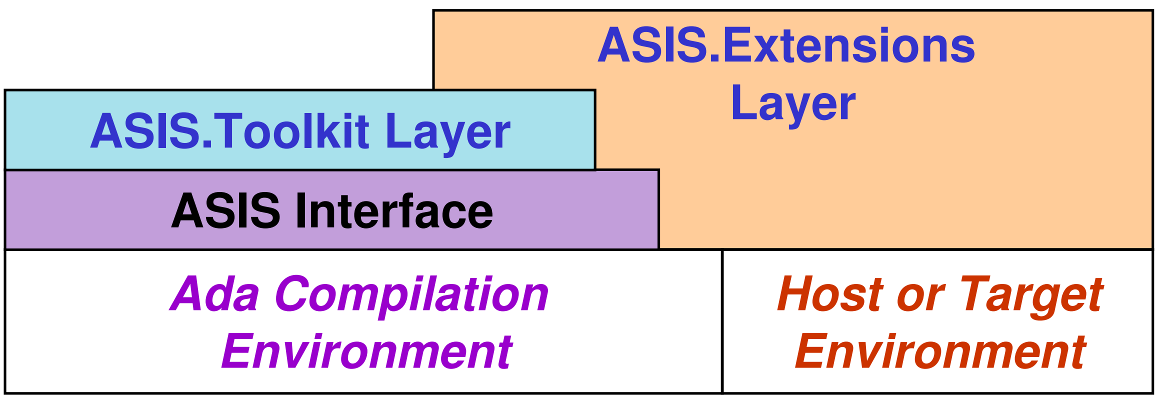

Figure 1 Relationship of the Extensions and Toolkit Layers to the ASIS Interface

Even with the limitation of using ASIS interfaces, the Toolkit functionality is limitless. It would be able to support the following types of functionality in a portable way across all Ada vendors supporting the ASIS Standard:

- Initialize and Finalize the ASIS environment and context;
- Select compilation units using mechanisms such as filters, wild cards;
- Classify elements with new queries such as: Is_Package, Is_Task, Is_Global;
- Fully classify types by cutting through derivations and private types;
- Develop queries to support Object Oriented code analysis such as Controlling_Parameter, and Corresponding_Tagged_Type;
- Support the use of Lists, Trees, Sorting, and other utilities;
- Offer generic report templates;
- Build iterative structures like Call Trees and Dependency Graphs; and
- Perform source code transformations.

Functionality from the Extensions Layer would support an even greater level of capabilities not possible with the ASIS interfaces alone, such as:

- Provide sizes and alignments for types and objects;
- Provide record layout information such as: component positions, first_bit, and width;
- Provide sizes for arrays and components;
- Provide frame sizes and stack offsets of local objects;
- Provide addresses of subprograms, tasks, or data within a target specific partition or program;
- Provide filename, directory, or other host dependent functions.

Example of Tool Integration Using Toolkit and Extensions Layers

An example of an integrated tool suite using Toolkit and Extension Layers was provided. The tool suite consists of:

- Asis.Toolkit.HTML – The HTML package is a simple interface that can glue individual tool output into an easily navigated set of reports. It can create anchors and links in Ada source code through ASIS code analysis. It would be portable amongst ASIS vendors with some tailoring to accommodate naming conventions on different hardware platforms.
- Asis.Toolkit.Call_Trees package – The Call_Trees package uses the ASIS traverse routine to build a call tree for the main program. It constructs local anchors and links to support navigation between the call tree report and the Ada sources. This also would be portable amongst ASIS vendors.
- Asis.Extensions.ObjectAda.Profiler package – The Profiler package creates a compilable copy of the source code and inserts unique controlled object declarations in callable entities. The controlled objects Adjust and Finalize procedures are invoked whenever a callee's scope is entered and exited. The callee's count and timing information is collected and saved at runtime. It also constructs local anchors and links to support navigation between the profiler report and Ada sources. The Profiler package uses ASIS interfaces, but by its very nature is vendor specific and would not be portable to other vendor environments.

Figure 2 depicts the output of such an integrated tool suite with navigation links between the Call Tree Report and source code and the Profiler Report and source code.

Proposed Guidelines

The proposed guidelines are to add only two new child packages to ASIS: Asis.Toolkit and Asis.Extensions.

ASIS vendors or tool developers would identify themselves as children and then add unlimited descendants, for example:

Asis.Toolkits.HTML	Asis.Toolkit.CORBA_IDL
Asis.Toolkit.Call_Trees	Asis.Toolkits.Croby_Inc
Asis.Toolkit.Apex.Call_Trees	Asis.Extensions.Apex.Profiler
Asis.Toolkit.GNAT.Call_Trees	Asis.Extensions.GNAT.Profiler
Asis.Toolkit.ObjectAda.Call_Trees	Asis.Extensions.ObjectAda.Profiler

Queries/Interfaces in the Toolkit Layer would be implemented with ASIS primitives, other Toolkit queries, common portable services, and/or common portable data structures.

Queries/Interfaces in the Extensions Layer can use these resources and also be implemented with vendor specific, host/target dependent services, and other non-portable services. Of course they would be able to use queries/interfaces in the Toolkit Layer.

These guidelines would allow vendors and users to independently build Toolkit and Extensions to ASIS that will:

- Promote portability for tools using the Toolkit interfaces;
- Organize additions along clear paths;
- Expand easily while avoiding name collisions;
- Mix and match with toolkits from other vendors; and
- Provide the foundation for integrated tool sets.

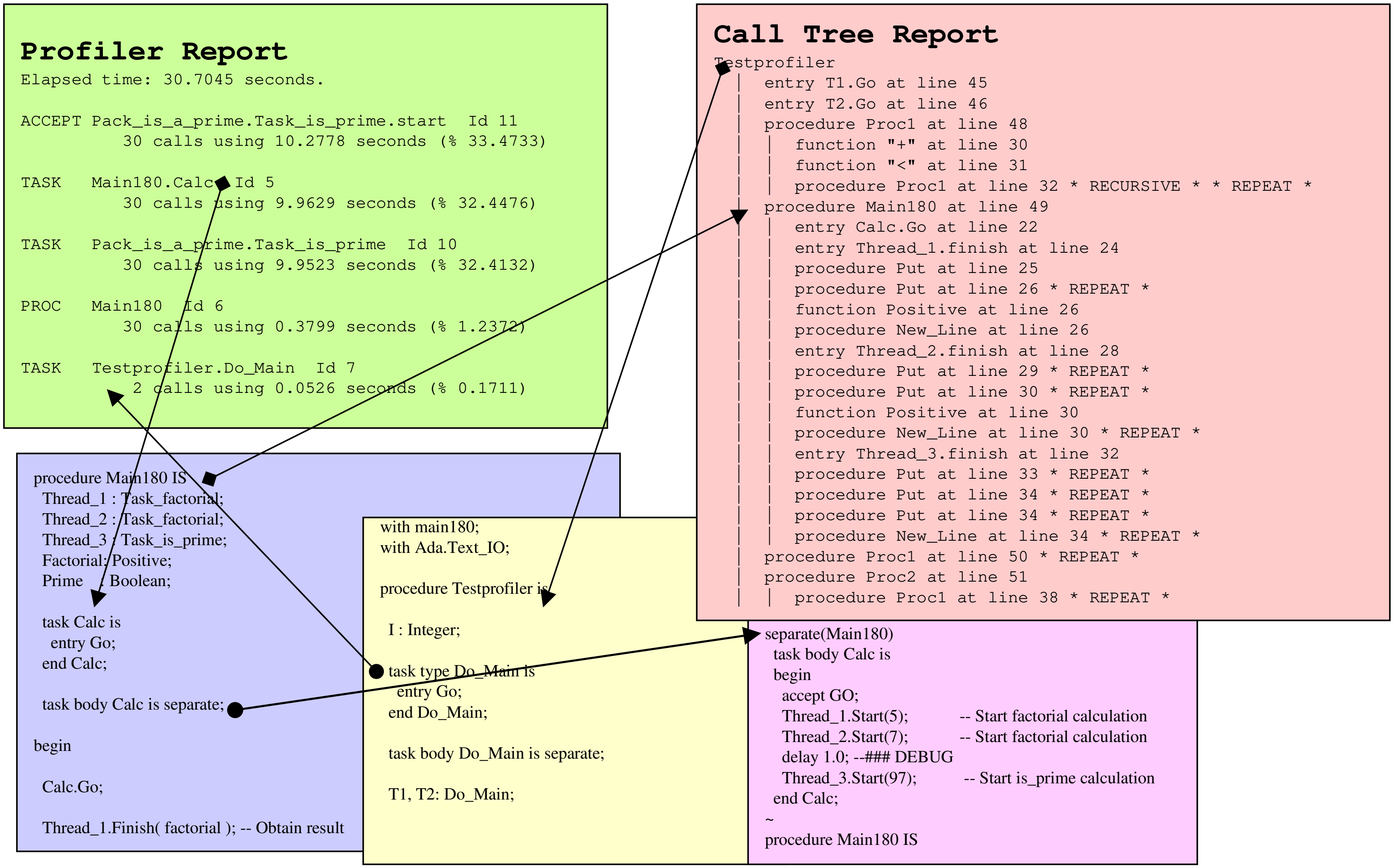

Figure 2 Integrated Tool Output with Navigation Links

Discussion on Proposed Guidelines

One of the things that ASISWG can facilitate is the sharing of tests among ASIS vendors – they can be placed on the ASIS Home Page. It was suggested that initially a package or a subprogram might be made available using the nomenclature:

Asis.Toolkits.Aonix.DoSomething

Then when it is shown to be useful to different users and portable amongst different vendors, it can be “elevated” to:

Asis.Toolkits.DoSomething

and placed in a different directory on the ASIS Home Page. Some discussion followed which indicated that there was really no reason to have the intermediate step (vendor-specific sub-packages) – simply name the package Asis.Toolkits.DoSomething and place the specification and body on the ASIS Home Page. It was recognized that there might be some value in using the vendor-specific sub-package naming convention as it also implies that the package has been tested in that environment. Also, it would eliminate the possibility of name clashes for ASIS vendors prior to registration on the ASIS Home Page. The ASISWG would accept contributions in both forms to be placed on the ASIS Home Page. Both package specifications and bodies should be provided. Comments should be used to explicitly identify vendor environments to which the code was tested. As artifacts are provided, vendors could identify that code is portable to their environment. It was noted after the workshop that it would be easier to maintain the compatibility information in a spreadsheet than in comments. The spreadsheet would identify which environments the code has been tested and who tested the code.

It was noted that defining an ASIS toolkit package and placing it on the ASIS Home Page is not a requirement that other vendors implement that particular package. But, if in the future, a vendor decides to implement the package, it must be implemented as described.

Brainstorming

The following areas were brainstormed.

- **Data Decomposition Annex –** ASIS Data Decomposition appendixes can be used for data dictionary applications. Steve said that a specific customer is still using Data Decomposition for one of their specific applications.

- **Tool Launchers –** Tool Launchers would allow a common way to start one or more ASIS-based tools. The user would first select the particular units or files or context and the tool launcher would execute. Steve Deller said that Rational uses a Configuration Management argument line, containing one or more compilation units/files and a switch indicating closure. There should be a minimal set of actions that the tool launcher should perform, e.g., identification of context, closure, etc. After much discussion, Joe Wisniewski took an action item to propose the functionality of a tool launcher.

- **Static versus Known_Value –** There was some discussion concerning the meaning of static expressions and expressions with a known value. The interface "Is_Static" was originally in Rational's LRM interface (one of the early ancestors of ASIS). Known_Value is implementation-dependent. Extensions (note -- not toolkit) should try to be made a "standard" extension -- if there is no agreement, then it is a "non-standard" extension.

- **CORBA IDL and ASIS** – Roy Bell brought up an issue concerning the ASIS database and its possible usefulness for CORBA IDL. As we know, Ada is central to ASIS. The good news is that it fully supports all of the syntax and semantics of Ada. The bad news is that it may not be too adaptable to other languages. Roy is specifically interested in CORBA IDL. It would be really great if we had a way to load the contents of CORBA IDL into an ASIS database. We could then query the database for the contents of the IDL. Perhaps some of the queries would not have a meaning with IDL, but IDL is surprisingly similar to an Ada specification. In fact, we should be proud that the CORBA chapter that describes how IDL should be mapped to Ada is the shortest of all of the mapping chapters in the CORBA standard.

One of the facilities that a CORBA vendor must supply is an "IDL repository". The CORBA standard specifies the queries that can be made on the repository. These queries are sufficiently powerful that a sophisticated client can "discover" services that have been specified in IDL. This in turn means that a CORBA client can use the results of the queries to formulate a request for a service. If we could load IDL into an ASIS repository it should be very easy to use ASIS queries to implement the CORBA requirements.

Roy admits that he is not too familiar with all of the details of ASIS, so he does not know all of the problems that can occur if we tried to load another language into an ASIS database. Hopefully, the mapping would not be too bad, but we don't know how bad it would be. Roy said that he would assume that queries on Ada elements that do not have a corresponding equivalent in IDL would result in a null response. A bigger problem would be CORBA IDL elements that do not have a corresponding equivalent in Ada. Two that come to mind are 1) IDL exceptions have data elements, and 2) IDL exceptions are associated with specific functions or procedures. Roy doesn't think that these two issues would present a problem to an implementer of an IDL repository, but they are information that some CORBA-ASIS users may wish to have. Those wishing to obtain more information should contact Roy Bell at RMBell@ACM.Org.

- **Future Directions –** Currently, the major players are Aonix, ACT, and Rational. Suggested packages should be placed on a "neutral" site – SIGAda's ASIS Home Page can be this location. Whenever a package is placed on the Home Page, there should also be a test driver -- a unit test driver -- which can test more than one thing in the package.

Conclusion

The use of Toolkit and Extension Layers appear to be very important to support the ASIS community. The Toolkit Layer will be portable and interchangeable with different vendors providing more powerful analysis capabilities to the ASIS community. It will provide higher level abstractions to more directly support the analysis community. It will serve as a valuable teaching resource to those new to ASIS to help reduce the learning curve. The Extension Layer can extend ASIS to serve many needs, which are out of scope for ASIS, such as dynamic semantics. It can also be used to fill in a number of holes and gaps in the ASIS international standard.

Clyde Roby will prepare the ASIS Home Page to receive Toolkit Layer contributions from the ASIS community. All are encouraged to contribute packages, which will benefit the Ada and ASIS communities. We would like to thank the Workshop participants for the valuable insight they provided.

Respectfully submitted,

Clyde G Roby Jr

Clyde G. Roby Jr.
ASISWG Recorder
roby@ida.org

Steven Blake

Steven Blake
Workshop Chair
sblake@aonix.com

The ASISWG Chair gratefully thanks Steve Blake and Clyde Roby for conducting a very successful workshop during my recovery from hip replacement surgery. The outcome of this workshop is valuable in providing the ASIS community with clear direction as to how to work together to provide more powerful analysis tools for the Ada community.

William Currie Colket

William Currie Colket
ASISWG Chair
colket@acm.org | colket@mitre.org