N89-16317

S38-61
167062
5P

[illegible] 536

Experience with the CAIS

Michael F. Tighe
Intermetrics, Inc.
Cambridge, Mass.

1. Overview

Intermetrics is currently using an earlier version of the CAIS (based on CAIS 1.2) in the implementation of it's line of Byron[1]/Ada[2] APSE products. This proto-CAIS provides all the Byron tools, Ada compiler, linker-driver, Ada program library manager, etcetera with a standard interface to the underlying operating system. Written in Ada and using Ada language features to separate specification from implementation, this proto-CAIS is currently implemented on four different operating systems, representing two different machine architectures including

- VAX/VMS (Digital Equipment Corporation)
- MVS/370 (IBM)
- CMS/370 (IBM)
- UTS 3.5 (Amdahl UNIX[3] derivative running under VM on the 370).

In progress is the task of moving the proto-CAIS (thus the Byron APSE) to the Sperry 1100 series (a third hardware class and fifth operating system).

Intermetrics is using this technology to permit the primary development team to proceed doing main-line development work on the Byron APSE, while *rehost* teams take either source or object modules (depending on the target hardware) and install the most recent version of the Byron APSE on these new machines for testing and

2. Ada is a registered trademark of the U.S. Government Ada Joint Program Office.
1. Byron is a registered trademark of Intermetrics, Inc.
3. UNIX is a trademark of Bell Laboratories.

ORIGINAL PAGE IS
OF POOR QUALITY

C-5

demonstration to customers. This process allows the various rehost teams to follow the primary development team very closely, at times being only two or three weeks behind the primary team in terms of supported capability. Each rehost team continues work on developing contract- or host-specific features for the rehosted compiler.

2. Proto-CAIS Usage in the Byron/Ada Compiler

The proto-CAIS is the primary database used by the functions that implement the Ada Program Library functions of the Byron APSE. The Program Library contains various representations of the Ada program as the compiler translates it from text to object code. The retention of these representations in the Program Library is under user control but usually include

- an *abstract syntax tree* (AST),
- a **Diana** representation of the program,
- an internal form used for code generation (called **Bill**),
- and the linkable object module.

Each form of the program representation is kept for each smallest compilable unit of the language, as the programmer can present his source to the Ada compiler in any of a variety of sequences and portions. It is necessary to organize these representations in an orderly fashion which is related to the name of the entity that they represent. Additionally, there are inter-relationships between the representations. For instance, each specific object module is derived from a corresponding specific Bill representation which is derived from a corresponding specific Diana representation which in turn depends on its specific abstract syntax tree representation. Compilation dependencies of the **with** statements in the source are represented as dependencies between the corresponding Diana representations. Date/time of compile and other information is kept as well.

As the Ada Program Library is also a foundation for a distributed configuration management system, it organizes the users compilations into collections and catalogs. A *collection* is a set of catalogs, and represents a higher level of grouping than the catalog. A *catalog* is a set of Ada sources (and their subsequent representations) that represent a single unit of related work chosen by the programmer. Each compile is made in within the context of a *primary* catalog which refers to various *resource* catalogs. Catalogs can created at any time, or can be *derived* from an existing catalog to form a new (incremental) version of the original catalog. Additionally, resource catalogs can be specified as *interface* or *implementation* catalogs, corresponding to the concepts of Ada **spec** and **body**. Multiple catalogs can exist as an implementation catalogs for a single interface catalog. At link time, the programmer is allowed to choose a specific implementation catalog to match a specific interface catalog.

Catalogs are composed of *units* which represent the smallest unit of compilation.

Each unit is composed of a *spec* and a *body*, with the possible inclusion of a *subunit*. Each spec or body is composed of the underlying representations *(form)* of the source (AST, Diana, Bill, Objmod).

One specific implementation detail is that all file objects (the CAIS *file node*), which represent the Diana or AST or Objmod, descend from the collection. The spec and body forms of the unit have secondary (rather than primary) links to the file nodes.

This grouping of compilation information into catalogs with all the various representations and attributes for each compilation unit represents the set of data managed by the proto-CAIS. This information is stored by the proto-CAIS in underlying host files. Each representation (AST, Diana, Bill, Objmod) of a compilation unit is kept as a separate file on the host. Relationships and attributes are stored in a single database represented by three files. The accompanying diagrams are intended to be suggestive of the use that the program library makes of the proto-CAIS rather than an exhaustively complete example.

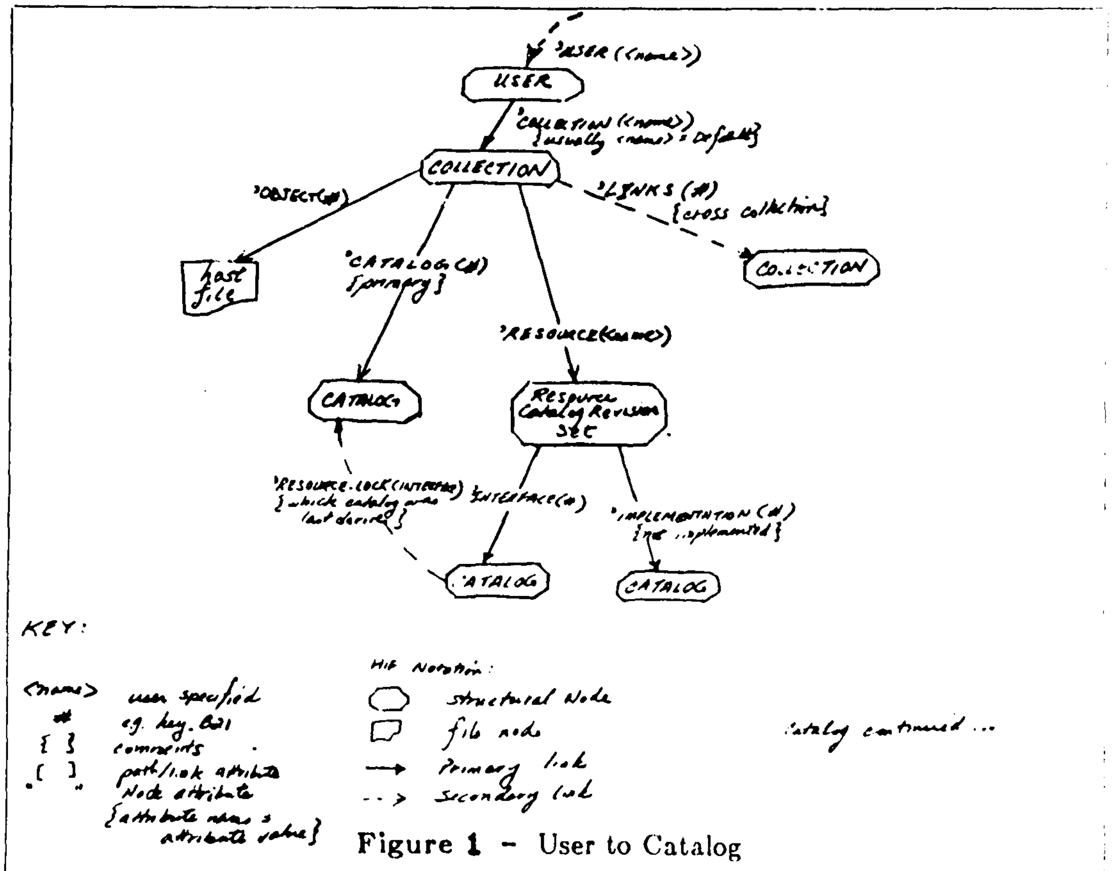

Figure 1 - User to Catalog

3. Implementation Details

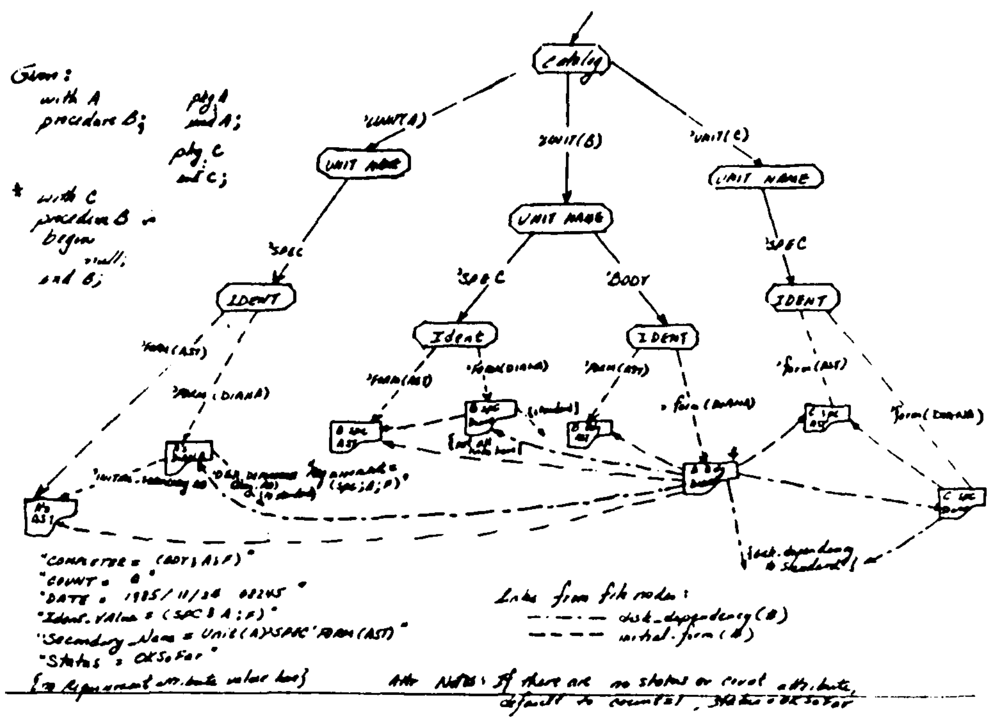

Figure 2 - Catalog Internals

Intermetrics has recently had experience in replacing the proto-CAIS with a totally re-written implementation to improve performance of the tool set using the proto-CAIS. Preliminary performance analysis indicated that the initial implementation of the proto-CAIS was a drain on the performance of the system, and it was targeted for a major upgrade in performance. The entire underlying implementation of the proto-CAIS was redesigned and reimplemented in light of the performance data, and the new implementation is currently installed in the latest version of the Ada compiler and its tools.

At present the new implementation is performing up to expectations with no anomalies reported due to differences in implementation. It is important to note that only minor changes (less than 500 lines of Ada code, excluding the new proto-CAIS code) were made in the compiler and tool sources (400KSLOC) to allow this new implementation to be installed. Most of these source changes were required by changed functionality of the new implementation of the proto-CAIS which were intended to improve performance without loss of portability. Some small number of changes were made to clear up anomalies in the preliminary implementation and definition of the proto-CAIS. Had no changes in functionality been required, there would have been no source changes required in the sources of the tool set.

4. Conclusion

Intermetrics experience is that the Ada package construct, which allows separation of specification and implementation allows specification of a CAIS that is transportable across varying hardware and software bases. Additionally, the CAIS is an excellent basis for providing operating system functionality to Ada applications. By allowing the Byron APSE to be moved easily from system to system, and allowing significant re-writes of underlying code, Ada and the CAIS provide portability as well as transparency to change at the application/operating system interface level.