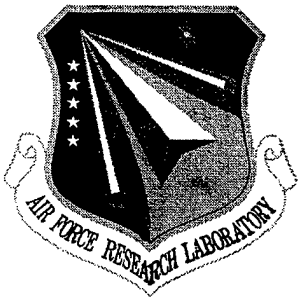
AFRL-IF-RS-TR-1998-62
Final Technical Report
May 1998



OBJECT ABSTRACTOR ADA 9X

Xinotech Research, Inc.

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. B123

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-1998-62 has been reviewed and is approved for publication.

APPROVED: James R. Milligan
JAMES R. MILLIGAN
Project Engineer

FOR THE DIRECTOR:

NORTHRUP FOWLER, III, Technical Advisor
Information Technology Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFTD, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

OBJECT ABSTRACTOR ADA 9X

Romel Rivera
Subash Shankar
Susan Seaholm

Contractor: Xinotech Research, Inc.
Contract Number: F30602-94-C-0175
Effective Date of Contract: 30 June 1994
Contract Expiration Date: 30 June 1997
Program Code Number: 6D10
Short Title of Work: Object Abstractor Ada 9X
Period of Work Covered: Jun 94 -Jun 97

Principal Investigator: Romel Rivera
Phone: (612) 379-3844
AFRL Project Engineer: James Milligan
Phone: (315) 330-3013

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by James Milligan, AFRL/IFTD, 525 Brooks Road, Rome, NY.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1998	Final Jun 94 - Jun 97

4. TITLE AND SUBTITLE

OBJECT ABSTRACTOR ADA 9X

5. FUNDING NUMBERS

C - F30602-94-C-0175
PE - 61101E
PR - B132
TA - 01
WU - 01

6. AUTHOR(S)

Romel Rivera, Subash Shankar, and Susan Seaholm

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Xinotech Research, Inc.
1313 Fifth Street Southeast
Minneapolis, MN 55414

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFTD
525 Brooks Road
Rome NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-1998-62

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: James R. Milligan/IFTD/(315) 330-3013

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes the work done by Xinotech Research on the Xinotech metalanguage-based object abstractor for ADA 9X program transformation. The goal of this project was to developed a knowledge-based metalanguage-based tool to support object abstraction for forward and reverse engineering of ADA programs. The capabilities of the object abstractor include those for improving software analysis, software understanding, software reengineering, and automatic program transformation. The tools developed under this effort support automatic translation from other languages to ADA, automatic generation of graphical architectural models from source code, user-assisted refinement and tuning of the graphical architectural model, and automatic transformation of the source code to the final model.

14. SUBJECT TERMS

ADA, Software Transformation, Reverse Engineering, Architecture, Reengineering, Code Generation

15. NUMBER OF PAGES

20

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

O. Abstract

This report describes the work done by Xinotech Research on the Xinotech Metalanguage-Based Object Abstractor for Ada 9X Program Transformation. The goal of this project was to develop a knowledge-based metalanguage-based tool to support object abstraction for forward and reverse engineering of Ada programs. The goals of the Object Abstractor include software analysis, software understanding, software reengineering, and automatic program transformation.

Our tools are based on the ARMY Missile Command Software Engineering Directorate (MICOM SED) methodology for reengineering. Xinotech is working very closely with MICOM. The MICOM SED methodology provides a well-defined sequential process for creating an organized object-oriented Ada architecture from legacy source code. This sequential process is implemented in the steps of the Architectural Transformation phase (OA-ARCH) of the Object Abstractor. These steps are described explicitly in Chapter 4 of the manual for the Object Abstractor [3]. A copy of the manual is enclosed with this report.

The tools developed in this project support automatic translation from other languages to Ada, automatic generation of graphical architectural models from source code, user-assisted refinement and tuning of the graphical architectural model, and automatic transformation of the source to the final model.

1. Object Abstractor (SOW 4.1.1)

The objectives of the Xinotech Object Abstractor are to: 1) provide flexible, evaluative support for the changing ARMY MICOM SED methodology, 2) provide extensive libraries of rules ("plans") to support automated extraction of a variety of new architectural models from the original sources, 3) support visual manipulation of the proposed models, 4) provide comprehensive transformation libraries to support automatic transformation of the original Ada sources into the new architectural model, 5) have a product that generates correct compilable Ada code, and 6) comfortably handle files of 50K lines.

The Object Abstractor provides the following functions:

- Language-based environment : Program analysis and transformation is based on the Object Abstractor's knowledge of the syntax and semantics of Ada.
- Knowledge-based environment: Program analysis and transformation is also supported by the Object Abstractor's knowledge of rules outside the language in use, such as rules for object-oriented design, domain-specific programming, architecture-specific support, programming standards, and reengineering methodologies.
- Language-independent environment: The Object Abstractor supports multiple languages and applications. All the language and outside-language knowledge is specified in XML, the Xinotech Metalanguage.
- Concept recognition, abstraction, and transformation: Pattern abstraction is used to support the recognition of implied concepts in the source code, and their transformation to higher-level models. Pattern abstraction is supported through an XML component called XPAL, the Xinotech Pattern Abstraction Language.

- Support for reengineering methodologies: Libraries of analyses, guidelines, and transformations can be organized to support specific methodologies.
- Support for multiple models (e.g. design and implementation): Multiple models of the same system can be maintained and compared.
- Multiple textual and graphical views: When a view is modified, all other views are updated. The Object Abstractor supports program and dictionary navigation across graphical and textual views and through declaration dependencies. Graphical and textual views also support levels of abstraction.
- Integrated forward- and reengineering: The Object Abstractor supports both interactive development as well as the reengineering analysis and transformation of a changing source code system.

These functions are made accessible to the user through three major activities available from the Object Abstractor: guidelining, modeling, and transformation. Guidelining is the ability to verify a software system's adherence to a standard. The Object Abstractor can verify adherence to design philosophies, programming disciplines, and metrics. Reports may be generated to document violations and the user may browse the corresponding positions in the source code.

Modeling involves analysis of source code and application of strategies for creating abstract representations of relevant relationships within the code. The Object Abstractor can also display the resulting models graphically, and allow the user to fine-tune the model or propose an alternative model. Transformation of source code may be carried out by the Object Abstractor, to reflect the new or proposed model, or to force adherence to guidelines which have been violated. These activities are described in greater detail in Chapter 1 of the Object Abstractor manual [3].

Under this contract, Xinotech achieved the following accomplishments:
- The Development of the Ada 95 Object Abstractor, for program sizes in the 30K lines of code range found in the test cases used.
- The use, testing, and evaluation of the Object Abstractor by the ARMY MICOM SED. This group applied the Object Abstractor to transform the architecture of legacy code translated from Jovial to Ada. It was determined that the tool successfully applied complex, global transformations to the source code, consistent with MICOM's methodology for re-engineering of Ada. The graphical modeling and turning of architectures was found particularly useful and powerful. The resulting code was evaluated as reaching "98%" of its desired state; engineers would modify about 2% to achieve compilation or other goals. Further lessons learned from this project are deferred to section 3.
- The pre-release of the Object Abstractor on a demo basis to selected commercial entities such as Bill Hodges of the Boeing (STARS) group. The Object Abstractor has not been used extensively yet in commercial environments, and has not been evaluated formally by this group.
- The commercialization of the Object Abstractor to other markets and languages, including in particular the COBOL Year 2000 problem.
- The specification of TypeL, a language for type prospecting and evolution to support type-based software reengineering. TypeL is described further in section 1.1.2.
- Demonstration of the Object Abstractor at the ARPA SIS Environment Days in January 1995, at the ARPA SISTO Symposium in August 1995, and the EDCS Demo Days in July 1997.

Particular notable features of components of the Object Abstractor are discussed below in the appropriate sections.

1.1 Object Extractor (SOW 4.1.1.1)

1.1.1 XML Ada Extractor (SOW 4.1.1.1.1)

The XML Ada Extractor was implemented. The Extractor extracts syntactic and semantic information from Ada source code. This capability is based upon a description of the syntax and semantics of the Ada language written in XML, the Xinotech MetaLanguage. The latter language is used to describe each element of the Ada programming language, including its external syntax or "views" and its semantic attributes, in an abstract grammar for the language. The XML description of the Ada language is then used by Xinotech's tools. For example, the Composer uses the description of Ada to provide formatting, interactive browsing and intelligent editing of Ada source code. As source modules are imported into the Composer, they are parsed and the Extractor extracts and records syntactic and semantic attributes. This information may in turn by used in the analyses of the Object Abstractor. The role of XML for Ada is described further in the attached document "Xinotech for Ada" [4].

1.1.2 Object Prospector (Object Identifier) (SOW 4.1.1.1.2)

The Object Prospector identifies objects in Ada source, and provides for their graphical display and manipulation. Major functions include the automatic identification of candidate objects in Ada source code, use of a type-based object prospecting algorithm, graphical display of the Ada software architecture, graphical manual tuning of the architecture, and type prospecting and evolution.

The Object Prospector is compatible with the MICOM Snider-Lewis methodology as well. This methodology establishes an Ada architecture by identifying objects based upon "read/write" relationships between procedures and data items. A state data item is defined as a data item which must retain its value after the scope of any procedures which modify it have been exceeded. State data items are protected within objects or packages, along with procedures which must access them. The methodology provides steps to prepare code for subsequent identification of objects, and for iterative analysis and definition of objects and relationships between procedures and data items . For more detail, refer to Chapter 4 of the manual for the Object Abstractor [3].

To support type-based reengineering, Xinotech specified and implemented a language called TypeL (Type Prospecting and Evolution Language). Recognizing that a fundamental paradigm for the reengineering of legacy software is as a type evolution problem, TypeL allows the user to specify rules that describe data structure evolution. Major classes of rules include:

- Type prospecting: finding types that correspond to typeless declarations, even in untyped languages.
- Data structure definition transformation: Correction of deficient data structures through transformation to new improved target types.
- Data structure reference transformation: Context-dependent upgrading of references to data structures whose definitions were changed.
- Operation transformation: Transformation of operations on updated data structures.
- Data conversion: Migration of data associated with updated data structures.

1.1.3 Pattern Abstractor or XPAL Evaluator (SOW 4.1.1.1.3)

The Xinotech environment is structured around the concept of pattern abstraction: the process of automatically condensing or abstracting low-level source code patters found in existing software into high-level program concepts. XPAL, the Xinotech Pattern Abstraction Language, is a declarative constraint

language used to express these program patterns and their transformations. XPAL is a component of XML, the Xinotech Meta-Language, which is a language for specifying the abstract grammar, external syntax (views), and semantics of languages. The relationships between the Xinotech languages are depicted in the attached figure.

As an example, consider implementing a guideline which recommends avoiding the use of "go to" statements. XPAL may be used to describe programming practices involving such statements, for example using a "go to" to jump out of a loop. XML would be used to describe each language element, including the syntax of the "gotoStmt", and that of the "loopStmt". An XPAL plan could then be written to describe a "go to" found within the body of a looping construct, by expressing the constraint that the go to is contained within a loop: "contains(loopStmt, gotoStmt)". XPAL provides a syntax for describing such constraints. The XPAL Evaluator identifies occurrences in the source code which match the constraints.

XPAL also provides syntax for describing changes which should be made to the program tree; for example the "gotoStmt" as identified could be eliminated from the program tree, and the "loopStmt" and its body modified to provide the logical equivalent of the original code. These modifications would be expressed in terms of the XML descriptions of the relevant language elements.

As part of this project, Xinotech made a number of improvements to the XPAL language and its implementation.

A. Improvements to the XPAL Language

Major improvements to the XPAL language include new data types, new predefined control procedures, and new predefined functions. The syntax of the language was enhanced for readability, clarity, and ease of use.

B. Improvements to the Implementation or Evaluator

A number of changes were made to improve the XPAL implementation. Major areas of change include faster execution, reduced memory usage, garbage collection, and XPAL profiling. Problems that became evident when scaling the technology were corrected.

1.1.4 XPAL Object-Oriented Library for Ada 9X (SOW 4.1.1.1.4)

Xinotech explored and implemented features related to a number of issues concerning Ada 9X. These include updating of the Xinotech Ada 9X specification, transformations to support reengineering to target object-oriented designs, and the adaptation of Hollingsworth's principles to Ada 9X. The latter principles are described in the Ph.D. dissertation of Hollingsworth [2], which describes a methodology for producing highly reusable components using Ada. The methodology is organized into a collection of numbered principles. The Xinotech library provides guidelines and transformations organized into the same numbered set of principles.

2. Extended Ada 9X Philosophy Library (SOW 4.1.2)

A set of transformations was created to provide consistency with Ada 9X philosophy: these transformations go beyond simple language compliance and help to produce code which takes advantage of special features of the language. For example, guidelines describe methods for turning procedures and functions into object-oriented packages, and variant types into tagged records which are more extensible object types. Other transformations take advantage of access subprogram types and abstract subprogram declarations. Yet others aid in reorganizing program decomposition, by splitting out child library units from existing Ada 83

packages. Guidelines supporting the philosophy of Ada 9X were derived from the document "Ada 95 Rationale" [1]. Xinotech has also created an additional set of guidelines which identify code in Ada 83 which is not compliant with Ada 9X. Criteria for non-compliance come from Appendix X of the above document [1], entitled "Upward Compatibility".

Additionally, Xinotech researched a number of program transformations supporting the reengineering of legacy Ada code to Ada 9X. Particular transformations studied include: identification of Ada 83 monitor tasks, transformation of Ada 83 monitor tasks to Ada 9X protected objects, and translation of an Ada 9X object-oriented subset to Ada 83. These transformations provided further information about Ada 9X philosophy; some were implemented in the library for the recognition and transformation of monitor tasks.

3. Conclustions and Future Research Possibilities

The technology as described above is operational. It produces consistent, predictable results when applied to source code. The greatest benefit is the time saved via automated extraction of models and transformations of source code. Compared to manual approaches, the use of the Object Abstractor involves less human effort, is more uniform and predictable, and is less error-prone. The application of the technology is practical in that the resulting source code is readable and the process relatively user-friendly. We conclude that the project has met its forecasted objectives with the approaches selected and as implemented.

The Army MICOM group found the automatic transformations to be helpful for improving the architecture of Ada code, and recommends they be extended to allow for additional steps in the process or other methods. When applied to Ada translated from legacy Jovial, it would be helpful to have first transformed the Jovial code where hardware-specific optimizations were used, including to "pack" or "overlay" items in memory. This in turn produces fewer lines of Ada code which provide a better basis for the re-architecturing efforts. Conditioning of legacy code prior to translation is an area of activity at Xinotech; however re-architecturing efforts should be concentrated on the target language, as concluded by MICOM.

This group also found the graphical models and transformations to be particularly "powerful and impressive". It is suggested for future work that it be easier to select or "grab" nodes within the graph, and to "unselect" items without selecting others. It would also be helpful to be able to separate items graphically at a finer level of granularity; some of the graphical operations for removing items were deemed overly "heavy-handed", producing too many changes at once. The user should be able to remove a single type declaration from a body into a package specification, then to another package, and then to reverse the procedure if desired. These improvements are left for future work.

During the project, name resolution schemes were implemented based on the semantics of the Ada language. An attempt was made to use the ASIS libraries for name resolution instead. The ASIS libraries provide an interface to semantic information derived by Ada compilers. It proved difficult to use these libraries, in part because they were poorly supported by the compiler vendors and contained bugs and errors. The name resolution schemes developed by Xinotech proved adequate for the task.

The greatest need for improvement involves scaleability. This was also the conclusion of the ARMY MICOM group. For larger applications, the time for analyses and transformations should be reduced. The technology should be able to handle larger code modules, and be tested further for robustness across a wide variety of Ada applications. The interactive capabilities for producing graphical object-oriented models should be enhanced. In addition, features supporting collaborative use by large teams should be improved. These goals are to be addressed in future projects with some of the current collaborators.

4. References

[1] *Ada 95 Rationale, The Language, The Standard Libraries.* Intermetrics, Inc. Cambridge, Massachusetts, 1995.

[2] Joseph E. Hollingsworth. *Software Component Design-for-Reuse: A Language-Independent Discipline Applied to Ada.* Ph.D. Dissertation, Ohio State University, 1992.

[3] Xinotech Research. *Object Abstractor.* Minneapolis, Minnesota, 1997.

[4] Xinotech Research. *Xinotech for Ada.* Minneapolis, Minnesota, 1995.

DISTRIBUTION LIST

addresses	number of copies
JAMES MILLIGAN AFRL/IFTD 525 BROOKS ROAD ROME NY 13441-4505	5
XINOTECH RESEARCH,INC. 1313 FIFTH STREET SOUTHEAST MINNEAPOLISH MN 55414	2
AFRL/IFOIL TECHNICAL LIBRARY 26 ELECTRONIC PKY ROME NY 13441-4514	1
ATTENTION: DTIC-OCC DEFENSE TECHNICAL INFO CENTER 8725 JOHN J. KINGMAN ROAD, STE 0944 FT. BELVOIR, VA 22060-6218	2
ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
RELIABILITY ANALYSIS CENTER 201 MILL ST. ROME NY 13440-8200	1
ATTN: GWEN NGUYEN GIDEP P.O. BOX 8000 CORONA CA 91718-8000	1

AFIT ACADEMIC LIBRARY/LDEE
2950 P STREET
AREA B, BLDG 642
WRIGHT-PATTERSON AFB OH 45433-7765 — 1

ATTN: TECHNICAL DOCUMENTS CENTER
OL AL HSC/HRG
2698 G STREET
WRIGHT-PATTERSON AFB OH 45433-7604 — 1

US ARMY SSDC
P.O. BOX 1500
ATTN: CSSD-IM-PA
HUNTSVILLE AL 35807-3801 — 1

NAVAL AIR WARFARE CENTER
WEAPONS DIVISION
CODE 4BL000D
1 ADMINISTRATION CIRCLE
CHINA LAKE CA 93555-6100 — 1

SPACE & NAVAL WARFARE SYSTEMS CMD
ATTN: PMW163-1 (R. SKIANO)RM 1044A
53560 HULL ST.
SAN DIEGO, CA 92152-5002 — 2

COMMANDER, SPACE & NAVAL WARFARE
SYSTEMS COMMAND (CODE 32)
2451 CRYSTAL DRIVE
ARLINGTON VA 22245-5200 — 1

CDR, US ARMY MISSILE COMMAND
REDSTONE SCIENTIFIC INFORMATION CTR
ATTN: AMSMI-RD-CS-R, DOCS
REDSTONE ARSENAL AL 35898-5241 — 2

ADVISORY GROUP ON ELECTRON DEVICES
SUITE 500
1745 JEFFERSON DAVIS HIGHWAY
ARLINGTON VA 22202 — 1

REPORT COLLECTION, CIC-14
MS P364
LOS ALAMOS NATIONAL LABORATORY
LOS ALAMOS NM 87545 — 1

AEDC LIBRARY
TECHNICAL REPORTS FILE
100 KINDEL DRIVE, SUITE C211
ARNOLD AFB TN 37389-3211 1

COMMANDER
USAISC
ASHC-IMD-L, BLDG 61801
FT HUACHUCA AZ 85613-5000 1

US DEPT OF TRANSPORTATION LIBRARY
FB10A, M-457, RM 930
800 INDEPENDENCE AVE, SW
WASH DC 22591 1

AWS TECHNICAL LIBRARY
859 BUCHANAN STREET, RM. 427
SCOTT AFB IL 62225-5118 1

AFIWC/MSY
102 HALL BLVD, STE 315
SAN ANTONIO TX 78243-7016 1

SOFTWARE ENGINEERING INSTITUTE
CARNEGIE MELLON UNIVERSITY
4500 FIFTH AVENUE
PITTSBURGH PA 15213 1

NSA/CSS
K1
FT MEADE MD 20755-6000 1

ATTN: OM CHAUHAN
DCMC WICHITA
271 WEST THIRD STREET NORTH
SUITE 6000
WICHITA KS 67202-1212 1

AFRL/VSOS-TL (LIBRARY)
5 WRIGHT STREET
HANSCOM AFB MA 01731-3004 1

ATTN: EILEEN LADUKE/D460 1
MITRE CORPORATION
202 BURLINGTON RD
BEDFORD MA 01730

OUSD(P)/DTSA/DUTD 2
ATTN: PATRICK G. SULLIVAN, JR.
400 ARMY NAVY DRIVE
SUITE 300
ARLINGTON VA 22202