# Image Cover Sheet

| CLASSIFICATION | SYSTEM NUMBER |
|---|---|
| UNCLASSIFIED | 154942 |

**TITLE**

A COMPARISON OF DEVELOPMENT EFFORTS USING SMALLTALK AND C++

**System Number:**

**Patron Number:**

**Requester:**

**Notes:**

**DSIS Use only:**

**Deliver to:**

# A Comparison of Development Efforts Using Smalltalk and C++

Submitted to: Defence Research Establishment Ottawa

Software Kinetics Document No. 1600-11902-001 Version 01
Copy #7 30 March 1994

# A COMPARISON OF DEVELOPMENT EFFORTS USING SMALLTALK AND C++

Contract #W7714-2-9657
Requisition #2

30 March 1994

*Prepared for:*
D. Elseasser
Electronic Warfare Division
Defense Research Establishment Ottawa

*Prepared by:*

Software Kinetics Ltd.
65 Iber Road
Stittsville, Ontario Canada
K2S 1E7

*Software Kinetics Document No. 1600-11902-001 Version 01*

Document Approval Sheet

*Document Name:* A Comparison of Development Efforts Using Smalltalk and C++

*Document No:* 1600-11902-001 Version 01

| **Approvals** | **Signature** | **Date** |
|---|---|---|
| Author | I. Graham | 30/3/94 |
| Project Manager | E.A. George | 30 Mar 94 |

Document Revision History

| Revision | Reason For Change | Origin Date |
|---|---|---|
| 01 | Original document issued. | 30 March 1994 |

**TABLE OF CONTENTS**

A Comparison of Development Efforts Using Smalltalk & C++

Version 01 #1600-11902-001

## 1. ABSTRACT

This document compares the experiences of the Data Fusion and Correlation Techniques Testbed (DFACTT) project team in developing the Smalltalk and the C++ versions of the DFACTT Intercept Workstation for OS/2. The document does not provide an objective comparison of Smalltalk versus C++; what it does do is identify considerations made in choosing to continue to use Smalltalk on the DFACTT project. These considerations may also provide insight to anyone attempting to compare the relative merits of C++ and Smalltalk.

## 2. INTRODUCTION

The Data Fusion and Correlation Techniques Testbed (DFACTT) is a multiprocessor platform designed to help develop and test multisensor data-fusion algorithms and sensor data analysis techniques. The testbed has been developed by the Defence Research Establishment Ottawa with the support of Software Kinetics Ltd.

Most of the coding of DFACTT software has been in Digitalk Smalltalk/V 286 with the support of the ENVY/Developer environment. Because Smalltalk/V 286 is being left generally unsupported and is being replaced by Smalltalk/V Windows and Smalltalk/V PM, the decision was made to port much of the DFACTT software to a better-supported environment on an OS/2 platform.

As part of the process of porting DFACTT to OS/2, one component of DFACTT, the Intercept Workstation, was partially ported to OS/2 using Borland C++ instead of Smalltalk. The limited success of what was essentially a rewrite of Intercept Workstation software resulted in a decision to port the original Intercept Workstation software to another version of Smalltalk for OS/2.

This document does not provide a general comparison of Smalltalk versus C++. What the document does do is compare the experiences of the DFACTT project team in developing the Smalltalk and the C++ versions of the DFACTT Intercept Workstation for OS/2.

## 3. THE DFACTT INTERCEPT WORKSTATION

The heart of the DFACTT system is the Analyst Workstation, on which battlefield situation-analysis is performed. Data is collected and sent to the Analyst station by various other sensors and

sensor-systems, an important one of which is the DFACTT Intercept Workstation.

The Intercept Workstation is the station at which an intercept operator operates one or more receivers to intercept enemy radio transmissions. It consists of a PC equipped with an IEEE-488 controller card controlling one or two WJ-8617B receivers, and an Ethernet card for communication with the Analyst Workstation and an audio-server. The system provides the following functions through a graphic interface: automated control of receivers, creation of intercept message reports, message interchange with the Analyst station, digital recording/playback of intercepted radio traffic via the audio-server, an integrated intercom facility, and automated radio activity monitoring and logging.

## 4. DEVELOPMENT ENVIRONMENTS

The following subsections identify the development platform and environments used for both the C++ and the Smalltalk versions of the Intercept Workstation.

### 4.1 Common Hardware and Software

Both the C++ and the Smalltalk versions of the Intercept Workstation were developed on a platform consisting of the following:

486-66Mhz PC;

IBM OS/2, version 2.1; and

IBM TCP/IP for OS/2, version 1.2.1.

### 4.2 C++ Environment

The additional software tools for development of the C++ version of the Intercept Workstation consisted of the following:

Borland C++ for OS/2, version 1.0.

### 4.3 Smalltalk Environment

The additional software tools for development of the Smalltalk version of the Intercept Workstation consisted of the following:

ENVY/Image R1.21 X9-PC; and

ENVY/Actra R2.0.

The class library included with ENVY/Image is very similar to that provided in Smalltalk/V 286; the user-interface classes, however, differ substantially from those in Smalltalk/V 286.

ENVY/Actra provides a multitasking/multiprocessing model that extends that which comes with the basic ENVY/Image Smalltalk.

## 5. OUTLINE OF THE C++ DEVELOPMENT EFFORT

### 5.1 The Time Invested

The total time spent in the development of the C++ implementation of the OS/2 Intercept Workstation was approximately 50 days. Due to difficulties quite apart from the actual programming effort, only about 30 of those days went into productive Intercept Workstation programming. So for the basis of comparison the effort should be considered to be 30 days.

Of the 30 days spent on actual design and coding, a large proportion was spent dealing with nuances of Presentation Manager programming. Though the DFACTT team members involved had extensive experience with Land E/W and the original DFACTT development, they had little experience with OS/2 PM programming.

### 5.2 The Development Process

For the C++ implementation, the process of development consisted primarily of design and coding from scratch. The user-interface of the OS/2 Intercept Workstation was to have a new look and was to provide virtually transparent control of any number of locally and remotely connected receivers (Remote receivers would be connected to other Intercept Workstations on the DFACTT network). The receiver-interface classes were developed without the support of any existing C++ class libraries.

### 5.3 The Final Product

The final C++ product fell well short of the intended goal of a complete Intercept Workstation. The only part of the Intercept Workstation that was completed fully was the window to control and respond to local receivers. The underlying interface class was completed, and the protocol and general design for the interface class for remote receivers was defined. An additional simulated interface class was also completed and used for testing. None of the gisting capabilities, channel-monitoring capabilities, nor audio-interface capabilities were tackled.

## 6. OUTLINE OF THE SMALLTALK DEVELOPMENT EFFORT

### 6.1 The Time Invested

The total time invested in the development of the Smalltalk version of the OS/2 Intercept Workstation was 32 days. Of that time, 14 days were spent porting code from the prototype Smalltalk/V 286 version, 10 days were spent developing new user-interface classes, and 8 days were spent testing and making minor modifications.

The programmer responsible for the development had received three days of formal training in programming the Common Widgets, the new windowing subsystem provided with ENVY/Image.

### 6.2 The Development Process

The Smalltalk development process differed significantly from the C++ development in that a significant amount of the original Smalltalk/V 286 code could be ported directly with little or no change. It was primarily the user-interface code that had to be rewritten, both because the user-interface classes in the new Smalltalk were vastly different and because the look of the interface was to be modeled after that designed for the C++ implementation. The actual percentage of code which was ported was 91% of the classes, 82% of the methods. So only about 18% of the total code needed to be redeveloped.

### 6.3 The Final Product

The final Smalltalk product provided a fieldable implementation of the Intercept workstation. The implementation did not support or even consider the control of remote receivers, however it provided full receiver interaction for local receivers, gisting capability,

and audio-interface capability. Channel-monitoring capability was not tackled.

## 7. ENVIRONMENT COMPARISON

### 7.1 Training Costs

Training costs for Smalltalk programmers are well-understood to be high. Though the language itself is syntactically simple, it typically takes six months or more for a programmer to become familiar enough with a Smalltalk class library to begin reaping the real benefits of using Smalltalk.

On the other hand C++ is a very complex language, and class libraries are less standardized. In developing the C++ implementation of the Intercept Workstation it was also found that learning to program OS/2 PM applications with C and its variants is potentially as complex as that of learning Smalltalk.

The brief Common Widgets training that was provided before the Smalltalk development, and the subsequent productivity (10 days for development of the Intercept Workstation user-interface) demonstrates at the very least that the basics of Common Widgets are fairly easy to learn for a competent Smalltalk programmer.

In summary, the cost of training a programmer to be proficient in programming ENVY/Image Smalltalk may not be vastly different from the cost of training a programmer in programming OS/2 PM. The Intercept Workstation experience has demonstrated the difficulty of learning PM, though a fair comparison cannot be made because of the previous experience of the Smalltalk programmer. However, without doubt the DFACTT team's experience has been that for a Smalltalk programmer to become productive with Common Widgets is far less costly than for a C++ programmer to become productive with OS/2 PM. A very rough estimate places the ratio of OS/2 PM : Common Widgets training time at about 5 : 1.

### 7.2 Code Management

An important issue in the development of any large software product is the management of source and object code. This code-management issue extends beyond initial development into follow-up development and version-management.

The ENVY/Image Smalltalk is delivered with ENVY/Manager, a sophisticated package of source- and object-code management tools.

ENVY/Manager is specially designed for a multi-user object-oriented environment. It implements a unique philosophy of class-ownership that still provides developers with the system-wide malleability for which Smalltalk is well-known. This approach is very suitable for an object-oriented environment where the integrity of complex classes might be difficult to achieve with a more traditional class check-out/check-in approach. Version-control is built in to the class-ownership implementation.

Borland C++ is delivered with a single-user file-based development environment. This environment's file-management tools are good for a single-user environment, however, they lack support for configuration-management and version-control. Most third-party source-code-management tools that are available for C++ were designed for simple file-management of straight C code. However, more sophisticated environments for C++ are available, though their strengths and limitations are outside the scope of this document.

Establishing a metric for comparison of the ENVY/Manager environment with a traditional file-based class check-out/check-in approach would be difficult. But without doubt the single greatest gain with ENVY/Manager is in programmer productivity: no programmer need ever wait for check-in of a class before beginning modifications or additions of her own. Re-integration of simultaneous changes to a class is performed quickly and easily with ENVY/Manager. Another gain with ENVY/Manager, even less easily quantified, is that class-ownership based code-management encourages quality and integrity within a class.

The impact of source-code-management issues on the DFACTT Intercept Workstation development discussed in this document was minimal. In both the Smalltalk and C++ cases a single developer worked on the project, so configuration-management was not an issue.

### 7.3 Debugging

Both ENVY/Image Smalltalk and Borland C++ come with powerful debugging tools. They both allow incremental source-level debugging with stack-viewing and extensive variable-inspection capability. This section will focus on significant differences recognized during the DFACTT Intercept Workstation development.

Recovery from errors is far better with Smalltalk, with environment crashes being significantly less frequent. Though Borland C++ includes many valuable compiler-checks for potential sources of errors, C++ programs are still vulnerable to bad pointers and their subsequent damage.

Another significant benefit to debugging Smalltalk is its capacity for context reuse. With C++ it is usually necessary after an error is encountered to restart the program, forcing the developer to reestablish similar context for additional testing. With Smalltalk, not only does most of the context usually remain intact after an error occurs, but code-changes may be made without terminating the program and losing context. Smalltalk's capacity for context reuse remarkably eases and shortens debugging time compared to that for debugging C.

One area in which Borland C++ exceeds ENVY/Image Smalltalk is in built-in variable monitoring. During debugging, the developer may request that the program break when the state of a variable is reassigned. With ENVY/Image Smalltalk, much more powerful variable-monitoring can easily be built into the classes being developed, however no variable-monitoring facility is provided by the environment itself.

ENVY/Image Smalltalk does not come with any breakpoint-setting facility. However, because of the ease with which code can be changed to insert a halt without losing a program's running context, there is actually little need for a breakpoint facility.

In conclusion, debugging in ENVY/Image Smalltalk is much easier than in Borland C++ for OS/2. The context reuse possible in Smalltalk provides a tremendous advantage that is not compensated for by any attribute of the C++ environment. The difference would be difficult to quantify because Smalltalk development can rarely be broken down into separate code-writing vs. debugging phases.

### 7.4 Class Libraries

A factor that significantly slowed development of the Borland C++ Intercept Workstation software was the lack of a rich class library. The only classes provided with Borland C++ are stream classes and container classes (a C++ Container being equivalent to a Smalltalk Collection).

ENVY/Image Smalltalk comes with a very rich, highly reusable set of class libraries. Classes are available to support many types of streams and collections, comparative magnitudes and numbers, user-interface development, multi-processing, object storage and retrieval, parsing, file-system access, and more. These classes and libraries are easily extended by adding additional methods and/or by using inheritance to define subclasses. The richness of the Smalltalk class library comes not only from the breadth of

functionality provided but from the extensibility and malleability possible with the existing classes.

There are numerous third-party C++ class libraries commercially available. However, the DFACTT project team has not researched what is available or how easily any of the products can be extended and adapted. The team's experience has been, though, that Smalltalk's pure object-model contributes significantly to the reusability of the provided classes; the lesser adaptability of C++, with its two-level mix of objects and lower-level types, is likely to limit at least somewhat the value of any C++ class library.

To effectively establish metrics for the availability and usefulness of Smalltalk and C++ class libraries would require an application-specific and source-specific study. The unique requirements of a given application and the availability of company-internal class libraries make any more general comparison unhelpful.

### 7.5 Design Consistency vs. Performance

The issues of design consistency and performance are discussed here together because both Smalltalk and C++ offer one of the two at the cost of the other. Smalltalk provides the consistency of a pure object-oriented model in which everything is an object, but with the cost of significantly slower iteration over large arrays. C++ offers the possibility of much higher performance working with lower-level types of data, but with the cost of the inconsistency of a non-uniform data-model.

Advocates for Smalltalk often argue that implementors of Smalltalk virtual machines have optimized Smalltalk message-sends so well that they come close to being as fast as a C function-call. However, the typical C++ application uses coarser-grained objects than does any Smalltalk application, the result being that many fewer C++ function-calls are performed than Smalltalk message sends; there is also typically less dynamic memory-management in a C++ application. The DFACTT team has found that when performance is critical great care must be taken to write fast Smalltalk code, but it is generally possible. The only large performance-drawback to Smalltalk virtual machines available today is slow iteration over arrays.

Smalltalk provides a homogeneous object-model, which, typically, makes code easier to understand and, as described in the previous

section, pre-built classes are typically more easily reused and adapted to suit the specific demands of a particular application.

Because the DFACTT Intercept Workstation is primarily user-interface, the code was not performance-critical so no objective comparison was made. It should be born in mind, however, that any portion of the DFACTT software which comes up against performance limits could be written in C as a DLL, which can easily be called from within Smalltalk.

### 7.6 Portability

For the DFACTT team, portability between radically different hardware platforms has been an issue from the beginning of the project. Much of the original DFACTT development was done on IBM PC compatibles running DOS, with the target platform being an embedded system consisting of a multi-processor VME-cage with Intel 680x0 processors running Harmony. More recently, the demand for support of modern user-interface standards such as Microsoft Windows, OS/2 Presentation Manager, and X/Windows has highlighted software portability as a priority.

Software developed in C++ for OS/2 will run under OS/2, only. Tools are available to help convert software developed for OS/2 Presentation Manager into software that runs under Windows, but the portability of the software is limited not so much by the language but by the application's extensive use of system-calls.

The architecture of ENVY/Image Smalltalk is such that software developed for one supported platform can be developed so that it is 100% portable to another supported platform. And the look and feel of the application blends in completely with that of each target platform. The platforms currently supported include Windows, OS/2, and ITI/Motif. This portability under ENVY/Image Smalltalk is invaluable when software being developed in a research environment must also be fielded on various other platforms, as has been the case with the DFACTT software.

### 7.7 Programmer Productivity

From the DFACTT team's experience alone, no conclusions may be formed about the general productivity of a competent ENVY/Image programmer against a competent C++ and OS/2 PM programmer. However, productivity in developing the Smalltalk implementation was far greater than that in developing the C++ implementation. So for the purposes of future DFACTT development, the advantages of being able

to port much of the original Smalltalk code make ENVY/Image the far more attractive choice.

Establishing metrics for comparison of programmer productivity in Smalltalk and C++ is difficult. Traditional means such as counting lines of code tell little: the only truly effective comparison is one which includes the entire life-cyle of a project from preliminary design to disposal. Even an entire-project comparison neglects the reusability of software components in other projects.

Worth mentioning, though acknowledged as being subjective, is that all team members on the DFACTT project believe that their productivity with Smalltalk is significantly greater than it ever was with more conventional languages. However, none of the team members have ever been proficient in C++.

## 8. CONCLUSION

Two main conclusions may be drawn from the comparative experience of developing DFACTT Intercept Workstation functionality in both C++ and Smalltalk:

1) Training costs to produce competent ENVY/Image Smalltalk programmers are likely not significantly greater than training costs to produce competent C++ OS/2 PM programmers; and

2) Productivity in porting the DFACTT prototype to ENVY/Image is far greater than productivity in redeveloping in C++ and OS/2 PM.

SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

## DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Software Kinetics Ltd.
65 Iber Road
Stittsville, Ontario K2S 1E7

2. SECURITY CLASSIFICATION (overall security classification of the document. including special warning terms if applicable)

Unclassified

TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

A Comparison of Development Efforts using Smalltalk and C++ (U)

AUTHORS (Last name, first name, middle initial)

I. Graham

DATE OF PUBLICATION (month and year of publication of document)

30 March 1994

6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)

14

6b. NO. OF REFS (total cited in document)

None

DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Report

SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

Defence Research Establishment, Ottawa
Shirley's Bay, Ontario
K1A 0Z4

PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)

02J03

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)

W7714-2-9657/01-QC

a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

#1600-11902 Version 01

10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)

DREO Contractor Report 95-634

DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availabilty (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

As for 11

13. ABSTRACT ( a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both offical languages unless the text is bilingual).

A subjective comparison of experiences in the development in C++ and Smalltalk of the DFACTT Intercept Workstation. Comparisons are made of the basis of the development environments, training costs, code management, effort to debug, class libraries, design consistency and performance, portability and programmer productivity.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Smalltalk
C++
Training Costs
Code Management
Effort to Debug
Class Libraries
Design Consistency
Performance
Portability
Productivity.

#154942

| NO. OF COPIES<br>NOMBRE DE COPIES | COPY NO.<br>COPIE N° | INFORMATION SCIENTIST'S INITIALS<br>INITIALES DE L'AGENT D'INFORMATION SCIENTIFIQUE |
|---|---|---|
| 1 | 1 | |
| AQUISITION ROUTE<br>FOURNI PAR ▶ | DREO | |
| DATE ▶ | 8 Jan 96 | |
| DSIS ACCESSION NO.<br>NUMÉRO DSIS ▶ | | |

DND 1158 (6-87)

National Defence / Défense nationale

**PLEASE RETURN THIS DOCUMENT TO THE FOLLOWING ADDRESS:**

DIRECTOR
SCIENTIFIC INFORMATION SERVICES
NATIONAL DEFENCE
HEADQUARTERS
OTTAWA, ONT. - CANADA K1A 0K2

**PRIÈRE DE RETOURNER CE DOCUMENT À L'ADRESSE SUIVANTE:**

DIRECTEUR
SERVICES D'INFORMATION SCIENTIFIQUES
QUARTIER GÉNÉRAL
DE LA DÉFENSE NATIONALE
OTTAWA, ONT. - CANADA K1A 0K2