マルチメディア通信と分散処理 43 － 1
（１９８９．９．２８）

ｓｕｐｅｒＣプログラム開発支援環境

勝山　光太郎　　中川路　哲男　　佐藤　文明　　水野　忠則
三菱電機株式会社　情報電子研究所

近年、ソフトウェア開発の効率化がますます重要な要件となり、オブジェクト指向言語によるプログラム開発が注目を集めている。我々は、オブジェクト指向の概念を導入しＣ言語を拡張したｓｕｐｅｒＣを開発した。本稿では、ｓｕｐｅｒＣによるプログラム開発をより効率よく行なうために開発したｓｕｐｅｒＣプログラム開発支援環境ＥＳについて述べる。ＥＳでは，プログラムの部品化、再利用を促進するために、既存のプログラムの情報を検索できるブラウザ機能や、環境自身にもオブジェクト指向の概念を導入することが重要と考え、環境自身を継承する機能をもたせた。さらにオブジェクト指向プログラムに適したデバッグを支援するために、オブジェクトの生成や消滅、メッセージパッシングの様子をグラフィックウインドウに表示し、プログラムの動作を視覚的に検証するツールも開発した。

An environment for superC programming

Kotaro Katsuyama　Tetsuo Nakakawaji　Fumiaki Sato　Tadanori Mizuno

Information Systems and Electronics Development Lab.
MITSUBISHI ELECTRIC CORP.

5-1-1 Ofuna, Kamakura-City, 247 Japan

For the purpose of developing software efficiently, an object oriented approach has been interested. We developed an object oriented language "superC", which is an extension of the C language. We also developed an environment for superC programming, named "ES". ES has a facility of an environment inheritance and provides several efficient software development tools, such as a browser, a makefile generater, a visual debugger, which shows an creation or deletetion of objects and an arrow for message passing in order to verify programs visually.

１．はじめに

情報通信システムの発展に伴い、近年、ソフトウェア開発の効率化がますます重要な要件となり、オブジェクト指向言語によるプログラム開発が注目を集めている。Smalltalkを始めとして各種のオブジェクト指向言語が開発されてきた[1]。中でも、既存のプログラミング言語にオブジェクト指向の概念を導入したものに注目すると、実装の観点からＣ言語を拡張したものとしてＣ＋＋[2]、Objective-C[3]，ＣＯＢ[4]といった言語が存在する。我々もまた、オブジェクト指向の概念を導入しＣ言語を拡張したｓｕｐｅｒＣを開発した[5]。ｓｕｐｅｒＣは、オブジェクト指向言語の特徴である継承機能や情報隠ぺい、さらにメソッドの解決を実行時に行なう動的束縛とコンパイル時に行なう静的束縛といった機能を有する。また、Ｃ言語のプリプロセッサとして実現しているので、移植性に優れている。

我々は、すでにいくつかのプログラムをｓｕｐｅｒＣを用いて開発してきた[6,7]。これらの経験から、プログラム開発支援環境の充実が重要と考え、より効率的開発を目的としたプログラム開発支援環境ＥＳ(Environment for SuperC)を構築した。ＥＳは、環境を構成するいくつかのファイル群と、ツール群とからなる。

オブジェクト指向言語の環境およびツールという観点から、smalltalkのように言語と環境が一体となったものがあり、いくつかの有用な機能を実現している。Smalltalkにあるブラウザ機能は、プログラムの部品化、再利用を行なうためには重要と考えられる。

また、コンパイル型の言語では、実行時のデバッグが問題となる。そのために、例えばObjective-Cでは、Viciとよばれるデバッガを用意して、インタラクティブなデバッグができるようになっている。しかしながら、テキスト表現でのデバッグであることと、オブジェクト指向言語であることに特化している訳ではない。そこで我々は、オブジェクト指向言語のデバッグという観点から、オブジェクトの生成、消滅、メッセージのやりとりをグラフィックウインドウに表示し、プログラムの動作を視覚的に検証できるビジュアルデバッガを実現した。

本稿では、ＥＳの構成と機能、そして特にＥＳのもつツールとして特徴的なビジュアルデバッガの機能について報告する。

以下、第２章ではＥＳの設計方針について述べる。第３章ではＥＳの構成要素と機能について詳述し、第４章ではビジュアルデバッガに焦点をあてて述べる。第５章ではＥＳに対する評価および考察をおこなう。

２．設計方針

オブジェクト指向概念を設計からデバッグまで浸透させるために環境にもオブジェクト指向の概念を積極的に取り入れる。つまり、環境へのオブジェクト指向概念の導入とデバッグへのオブジェクト指向概念の導入である。

（１）環境へのオブジェクト指向概念の導入

プログラムの再利用を実現する際に、継承を考えずに既存のクラス情報をすべて一元的に取り込んだ環境を用意すると、次のような問題が生じる。

①クラス名の重複が許されない。
②一時的に作成したクラスが、他のクラスと同じレベルで登録されてしまう。
③環境データをもつファイルが大きくなり、動作が遅くなる。
④複数人での開発がむずかしい。

これらの問題を解決するためには、クラスの扱いに適度の局所性をもたせる必要がある。オブジェクト指向型言語は、新しいプログラムを作成するとき、既存のプログラムに変更部分の追加を行なうことによって目的とするプログラムを実現することのできる継承とよばれる機能を持っている。我々は、ＥＳにプログラムの再利用をサポートする機能および環境の局所性を組み込むために、環境にも継承機能を導入した。これにより環境もプログラムと同じ継承という一貫した考えでまとめられ、ユーザに理解しやすいものとなる。

（２）デバッグへのオブジェクト指向概念の導入

オブジェクト指向によるプログラミングでは、オブジェクトとその間で交換されるメッセージパッシングがプログラムの中心的な役割をなしている。そして、オブジェクトの生成、消滅とオブジェクト間のメッセージパッシングを把握することがプログラムの流れを把握する上で非常に重要なこととなる。

ｓｕｐｅｒＣは、コンパイル言語であるためにオブジェクトの生成、消滅、メッセージのやりとりなどを対話型の環境で実行し、その動きを動的に追うことができない。また、デバッグツールとしてＣ言語用の各種デバッガを使用できるが、オブジェクト指向プログラミングの利点であるオブジェクトとメッセージの振舞いを生かしたデバッグは行えない。

これらの問題点を改善するためにビジュアルデバッガを開発する。ビジュアルデバッガはオブジェクトの生成、消滅、メッセージの伝達をグラフィックウインドウ上に分かりやすい形で表示する。さらに、オブジェクト間のメッセージのやりとりを対話的に実行できる環境を提供する。

３．ＥＳの構成と機能

３．１　環境の構成

ＥＳが管理しているファイルには、インヘリタンスファイル、ブラウザファイル、リファレンスファイルがある。これらの環境データファイル群はシェルのもつ環境変数'REFERDIR'が示すディレクトリの下に置く。リファレンスファイルは、分割コンパイルのためのメソッドデータファイルで、メソッド名とメソッドＩＤの対応をもち、開発目標に対して１つ作成する。ブラウザファイルは、ファイル、クラス、メソッドの関係をもつファイルである。インヘリタンスファイルは、検索順序に関する情報を保持している。

環境データファイル群の構成概念図を図１に示す。

３．２　環境の継承

継承の実現はUNIXのシェルのもつパスに類似した方法によって行なっている。決められたディレクトリの下に、環境名＋サフィックス’ｒ’のインヘリタンスファイルを置き、その中に検索順序に関する情報をもたせた。

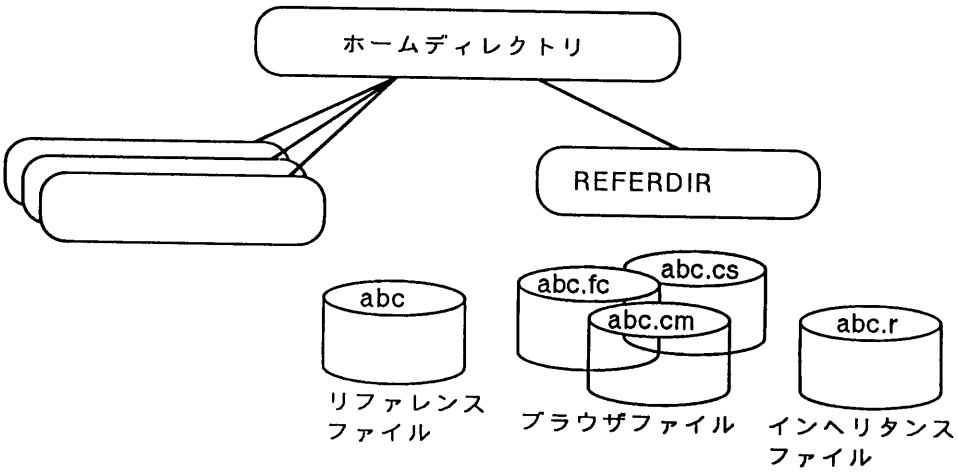

図１　環境の構成

3．3　ツール群

ＥＳでは、図２の表示例にしめすように、メニューを選択することによって以下の機能を実現するようにした。図２ではブラウザとリファレンスファイルの参照とを行なっている。

（１）ブラウザ（browser）

ブラウザファイルに基づき、ファイルとクラスの関係、クラスとメソッドの関係、メソッドのインタフェースなどをメニューの選択指示によって表示する機能を提供する。

ブラウザの機能を表１に示す。

（２）リファレンスファイル参照（showdb）

メソッド名とメソッドＩＤの対応を保持しているリファレンスファイルの内容を表示する。

（３）リファレンスファイルの削除（rmdb）

リファレンスファイルを削除する。

表１　ブラウザの機能一覧

選択ﾒﾆｭｰ	指定	機能
File to Class	ﾌｧｲﾙ名	指定ﾌｧｲﾙに定義されているｸﾗｽ名すべてを出力する
Class to File	ｸﾗｽ名	指定ｸﾗｽが定義されているﾌｧｲﾙ名をすべて出力する
Class to Method	ｸﾗｽ名	指定ｸﾗｽに定義されているﾒｿｯﾄﾞ名をすべて出力する
Method to Class	ﾒｿｯﾄﾞ名	指定ﾒｿｯﾄﾞが定義されているｸﾗｽ名をすべて出力する
Method to Interface	ｸﾗｽ名 ﾒｿｯﾄﾞ名	指定クラスに定義されている指定ﾒｿｯﾄﾞのｲﾝﾀﾌｪｰｽを出力する
vi+	ｸﾗｽ名 ﾒｿｯﾄﾞ名	指定ｸﾗｽが定義されているﾌｧｲﾙについてvｴﾃﾞｨﾀを起動しｶｰｿﾙを指定ﾒｿｯﾄﾞの定義されているところにもっていく

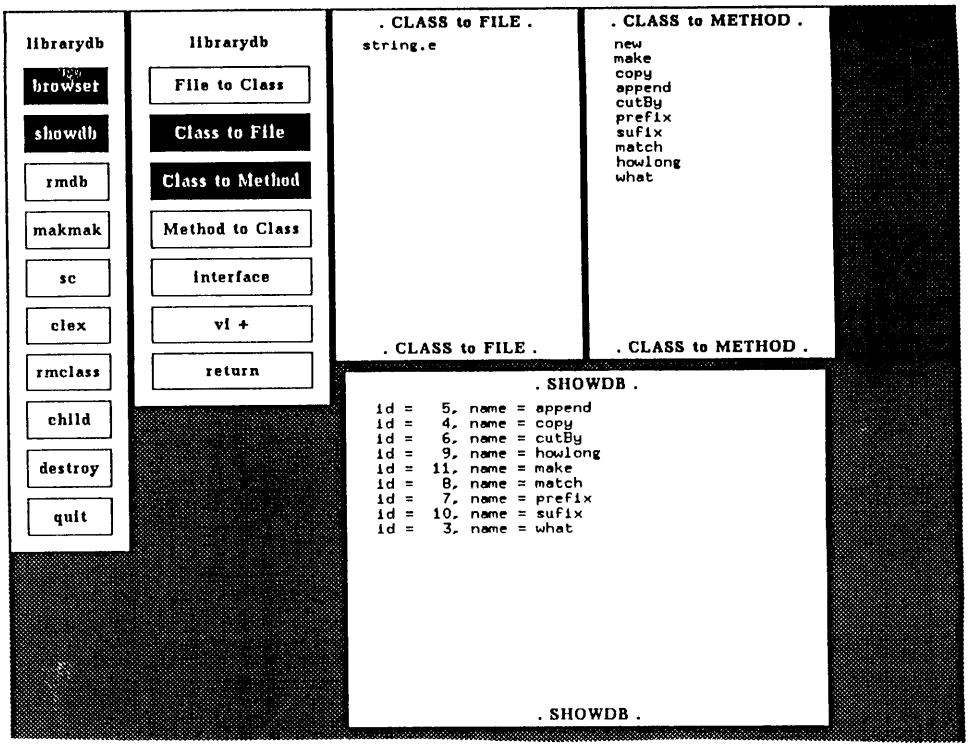

図２　ＥＳの表示例

（４）makefile自動生成（makmak）

ｓｕｐｅｒＣの構文規則とＥＳがもつデータとを利用してmakefileを自動的に生成する。

構文規則としては、externclassによりクラスの外部参照を宣言する。externclass文を検索すれば、プログラムの中で使用されるクラス名を得ることができる。また、ＥＳはブラウザファイルの中にクラス名とそのクラスが定義されているファイル名の情報を持っている。これらの情報によりmake fileを自動的に生成する。

（５）ｓｕｐｅｒＣプリプロセッサ（sc）

superCのプログラムからＣのコードを生成する。プリプロセスの過程で、クラス情報、レファレンスファイル、ブラウザのための情報を生成する。

（６）環境からクラスを削除（rmclass）

環境から指定のクラスを削除する。

（７）環境の継承（child）

childをクリックし新しい環境名を入力するとその環境は、現在の環境を継承する。

（８）ＥＳの終了（quit）

ＥＳを終了させる。

４．ビジュアルデバッガの機能

ビジュアルデバッガは、オブジェクトの生成、消滅、メッセージの伝達をグラフィックウインドウ上に分かりやすい形で表示するものである。図３にビジュアルデバッガの表示例を示す。

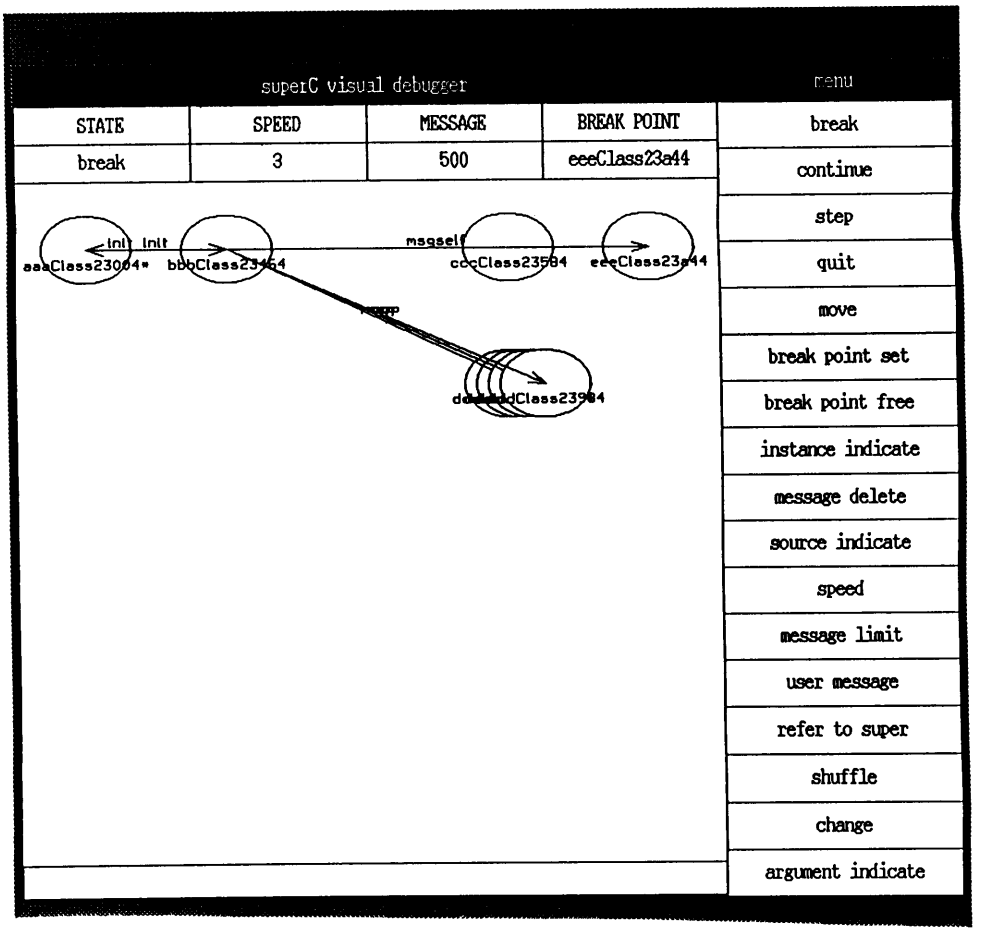

図３　ビジュアルデバッガの表示例

ビジュアルデバッガは、ユーザのプログラムとリンクすることにより、単一プロセスとして動作する。ユーザは、オブジェクトを単位として対話的に実行し、デバッグをおこなうことが可能である。ビジュアルデバッガのもつ主な機能を以下に説明する。

（1）オブジェクト表示

現時点で生成されているオブジェクトをすべて表示する。表示のタイミングは、newのメッセージが送られたときであり、purgeによりオブジェクトが消滅した場合には、表示も消去される。

表示位置は、ウインドウを格子状に分けた各格子点であり、同じクラスから生成されたオブジェクトは少しずつずらし重ねて表示される。オブジェクトの形のディフォルトは円であり、クラスの階層構造は表示しない。ただし、マウスの操作により、オブジェクトを拡大表示することができ、その時は、クラスの階層構造を見ることができる。拡大時のオブジェクト表示では、オブジェクトへのメッセージが解決された階層に矢印が向くことにより、どのクラスのレベルでメッセージが解決されたが分かる。

図4にクラス階層構造の表示例を示す。

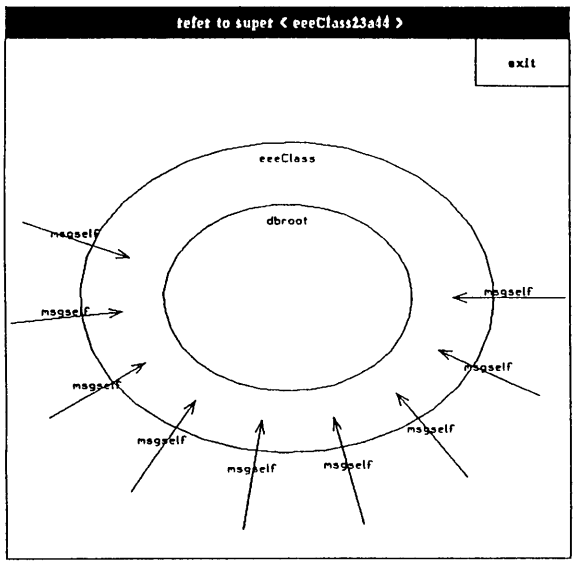

図4　クラス階層構造の表示例

（2）メニューによる対話処理

オブジェクトの表示と別のウインドウにメニュー画面がある。メニューの選択により以下の機能が選択できる。図3の右側に並んでいるのがメニューである。

①表示速度指定（speed）

オブジェクトの表示間隔を秒単位で指定できる

②オブジェクトの表示形態の変更（change）

注目すべきオブジェクトの形状を変更できる。

③矢印の削除（messege delete）

メッセージパッシングの矢印が増えてくると、見にくくなるので削除する。

④オブジェクトの移動（move）

オブジェクトを見やすい位置に移動できる。矢印もラバーバウンドで追随する。

⑤メッセージの強制発生（user message）

オブジェクトに対して任意のメッセージを送ることができる。

⑥ブレイクポイントの設定（breake point set）

オブジェクトにブレークポイントを設定する。なんらかのメッセージを受けたときに、ブレークする。

（3）インスタンス変数の表示

クリックによって、指定したオブジェクトに対してインスタンス変数を参照するメッセージを発行可能である（instance indicate）．これは、ブレークポイントで停止中のオブジェクトに対してのみ有効である。なお、インスタンス変数を参照するメソッドはプリコンパイラが自動的に生成し、ソースコードに組み込む。

（4）ソースコードの表示

プログラムの流れを知るためには、オブジェクトおよびメッセージに注目することが重要である。しかしさらに細かな情報を知るためにはソースコードを見る必要がある。特に現在送られているメッセージは、ソース上のどの部分に当たるかを知ることが重要である。そのために、メッセージを送ったソースのラインを表示することができる（source indicate）。

5．考察

環境の継承を利用してプログラミングを行なう場合、まず上位の環境を何にするか考える必要がある。その時、最上位にはそのシステムを利用している全員が使うような、環境を定義するのが望ましい。また、数人で１つのターゲットを作る際に共通して使う環境がある時それを上位の環境とし、それらの性質を受け継ぐ環境を生成してその中でプログラミングを行なう。目標とするプログラムを作るに際に適した環境を継承すれば、過去に作ったクラスの情報（クラスを含むファイル名、クラスが含むメソッド名、メソッドのインタフェース、クラスの親子関係）が得られ、しかもそのクラスをプログラムの中で使うことができる。ただし、環境の継承機能によって使えるクラスは、動的な束縛の利用に限定される。

もし、ある環境内で過去に作ったクラスを利用して目標とする機能を得たが、実行速度の点において仕様を満足しない場合、時間的に多くを費やしている部分を静的な束縛に置き換えることによって実行速度の向上を得ることができる。

このようにＥＳ上でのプログラム開発は、環境の継承機能を利用して、手早くプログラミングし、実行を確かめた後、実装にそくした形になおすことによって進められる。

現状では、ブラウザ機能を使うほど、クラスライブラリが充実しておらず、今後充実を図っていく必要がある。

ビジュアルデバッガでは、オブジェクトの生成、消滅が確認できるため、オブジェクトを解放しないためにおこるメモリ不足を防ぐことが可能となる。

ビジュアルデバッガにおける課題としては、表示可能なインスタンス変数に制限のある点がある。

また、グラフィックウインドウを利用しているため、ウインドウを利用したプログラムをデバッグするような場合には、ウインドウが重なってしまうため、操作が繁雑となる。

6．おわりに

オブジェクト指向言語ｓｕｐｅｒＣを利用したプログラム開発を効率よく行なうための、プログラム開発支援環境ＥＳの構成と機能について述べた。今後、ＥＳを利用したプログラム開発の経験を積むことによって、評価改良をおこない、ＥＳの機能充実を図る。

＜参考文献＞

［１］Goldberg, A. and Robinson, D.:Smalltalk 80:The Language and Its Implementation, Addison-Wesley(1983).

［２］Stroustrup, B.:The C++ Programming Language, Addison-Wesley(1986).

［３］Cox, B.J.:Object Oriented Programming-An evolutionary Approach, Addison-Wesley(1986).

［４］上村他：ＣＯＢにおけるオブジェクト指向機能、コンピュータソフトウェア、Vol.6, No.1, pp4-16(1988).

［５］勝山他：通信ソフトウェア向けオブジェクト指向言語ｓｕｐｅｒＣ，情報処理学会論文誌、Vol.30, No.2, pp234-241(1989).

［６］中川路他：国際標準に準拠したファイル転送プロトコルの実現と評価、情報処理学会論文誌、Vol.29, No.11, pp1071-1078(1988).

［７］中川路他：ＯＳＩ　ＣＣＲの実現、マルチメディア通信と分散処理研究会、ＭＤＰ３７－２（１９８８）．