コンピュータと教育　16－1
（１９９１．５．１７）

# 言語ＣのＣＡＥシステムの設計と開発

玉木裕二，森田雅夫†，並木美太郎，高橋延匡
（東京農工大学　工学研究科　情報工学講座）

本研究室では，毎年新しく研究室に配属される新４年生を対象に，言語Ｃ勉強会を開いている．これは，新４年生をシステムプログラマに育成することを目標に，演習形式で行っている．本研究では，この勉強会の代わりとなる CAE システムを実現することを目標にしている．その第一段階として，学習のための実行環境を実現し，学習者の行動を分析することにした．

本稿では，言語Ｃの CAE システムとそのプログラム実行環境と，学習者の行動分析について述べる．

# Design and Development of a CAE System for the Programming Language C

Yuji Tamaki, Masao Morita†, Mitaro Namiki and Nobumasa Takahashi

Department of Computer Science,
Tokyo University of Agriculture and Technology,
Koganei-shi, Tokyo, 184 Japan

We have been giving a course of programming in the language C for senior students who join our laboratory every year. The goal of this course is to bring them up as systems programmers. A main emphasis is laid on exercises. This experience has motivated the development of a CAE system on which each student can practice his own design and understand various sorts of trade-off.

This paper describes a development of the CAE system as a programming practice environment and a desigen of the mechanism to collect statistics on learner's behavior.

† 富士ゼロックス（株）

## １．はじめに

我々の研究室では，言語Ｃを用いて，OS をはじめとしたコンパイラ，日本語情報処理研究の一貫として，DTP，手書き文字認識などの研究，開発，実用を行っている［１］．これらのプログラムは，数世代にわたって学生に引き継がれている．したがって，研究者は，効率がよく，信頼性，拡張性，保守性の高いソフトウェアを開発する技術を備えていなければならない．我々は，毎年，新しく研究室に配属される新４年生が，このような技術を習得する場として，言語Ｃ勉強会を開いてきた．“演習なくして，プログラミング言語習得はありえない”という信念のもとに，この勉強会は演習形式で進めている［２］．

この勉強会の目標は，新４年生をシステムプログラマに育成することである．システムプログラムは，アプリケーションプログラムに対し，10 倍以上のプログラミングコストがかかると言われている［２］．特に，システムプログラムには，プログラムの信頼性が徹底的に要求され，さらに，プログラムの拡張性が要求され，他の同種のものと競争するため，性能設計を十分に行わなければならないという現実がある．これらのことから，我々は，システムプログラマの必要条件として，次の要件は満たさなければならないと考えている．

（１）ソフトウェアのライフサイクルの中での，開発対象の位置づけを明確にできる設計能力を持つこと，また，プログラムの分割，インタフェースの仕様の設計能力を持つこと
（２）効率がよく，信頼性，拡張性，保守性の高いソフトウェアを開発できること

これらのことを踏まえ，勉強会では，学習者に考えさせることを中心に進め，特に次の２点に重点をおいている．

（１）実際のプログラミングを通して，学んできた知識を応用する
（２）設計から保守まで，ソフトウェアのライフサイクルを念頭においた考察を行う

本研究の目的は，この言語Ｃ勉強会の代わりとなるシステムを実現することである．我々はこの研究を，学習者の自主性と創造性を重視し，計算機に Instruction を行わせるのではなく，Education を行わせるための研究，すなわち CAE(Computer Assisted Education) の研究と考えている．しかし，このシステムを実現するためには，学習者モデル，知識表現など本質的な問題を解決しなければならない．そこで，我々は，これらの問題を解決するための手がかりを得るために，学習のための実行環境を実現し，学習者の行動を分析することにした．特に，学習者に応じた指導を行うという観点から，学習者の個別評価を行い，その結果をもとに，学習者モデルを作成することを考えている．

本稿では，言語Ｃの CAE システムとそのプログラム実行環境と，学習者の行動分析について述べる．

## ２．言語Ｃの CAE システム

### ２．１　言語Ｃの CAE システムの設計と構成

本研究の最終的な目標は，言語Ｃ勉強会の代わりとなるシステムを実現することである．そこで，言語Ｃ勉強会の流れを参考に，CAE システムの設計方針を次のように設定した．

（１）学習者に適した問題の出題を可能とする
（２）学習者との質疑応答を可能とする
（３）ソフトウェアのライフサイクルを見通した支援を可能とする

また，言語Ｃの CAE システムの全体構成を図１のように設計した．

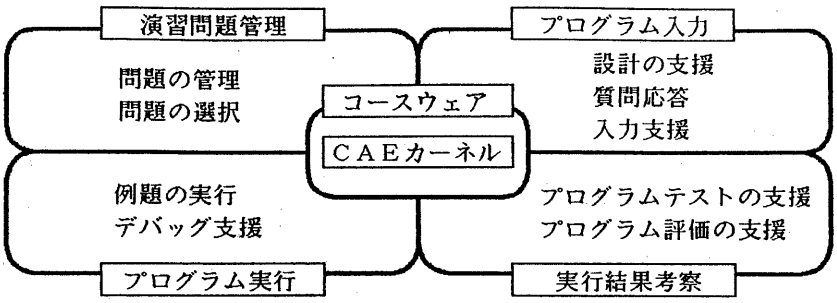


図１　ＣＡＥシステムの全体構成

## ２．２　言語Ｃの CAE システムの実現方針

前述の機能を実現するためには，多くの問題点を解決しなければならない．具体的には，

（１）コースウェア
（２）学習環境

という，技術的問題と，

（３）誤り原因同定のための学習者モデル
（４）学習者との対話を行うための知識表現

という，プログラミング言語の CAI の抱える本質的問題を解決しなければならない．

しかし，数年間の教育経験からシステムプログラミングの教育として必要なことは判ってきた．したがって，それをベースにコースウェアを作成し，また，学習環境を実現し，実際に使用する．また，実用を通して学習者の行動を分析し，それをもとに学習者モデルを作成する．さらに，それが正しかったかどうかデータを取って検証し，新しい学習者モデルを作成する．

これらのことを踏まえ，CAE システムの実現方針を次のように設定した．

（１）コースウェアを開発する
（２）学習環境を構築する
（３）学習者の行動データを収集，解析し，学習者モデルを作成する

なお，（１）については，言語Ｃ勉強会を通して得られた知見をもとに，昨年度作成した［２］．本年度の言語Ｃ勉強会は，このコースウェアに沿って進めている．（２）については，CAE システムのプログラム入力部，プログラム実行部，実行結果考察部を実現することに相当する．これらをプログラム実行環境として実現する．

# ３．プログラム実行環境の設計と実現

## ３．１　プログラム実行環境の設計方針

プログラム実行環境の設計方針を次のように設定した．

### （１）入力からデバッグまでの統合的な支援を可能とする

現在までの学習環境の改善を計るために，データ構造を図示するツールや関数のインタフェースをチェックするツールなどを作成し，使用した．この結果，これらのツールは強力なツールであることが判明した．しかし，これらのツールを一つの独立したツールとして実現したため，他

のツールの情報が使えないなどの理由により，支援には限界がある．

これらのことから，プログラムの実行やデータ構造の図示などを行うツール群を，一つの操作体系で使用でき，各ツール間でインタフェースを取れるような，統合した環境として構築し，これらを支援する．

（2）エラーの原因をユーザが自分で調べられるようにする

デバッグは，ノウハウの比率が高い．つまり，どの情報に着目すれば，何が分かるといったことを，どのように教育するかが重要である．しかし，学習者の自主性を伸ばすには，システムから積極的に情報は提示しないで，学習者自身がエラーの原因を調べた方が効果的である．この実行環境では，エラーの原因を調べらやすいように，システムが実行時に用いている情報をすべて学習者に開放する．また，これらを通して，システムプログラムで一番重要な信頼性の高いプログラム，保守性，拡張性の高いプログラムを作成するために，何をしなければならないかを考えさせる．

（3）学習者の行動分析を行うための機能を用意する

我々は，学習者モデルを作成するためには，まず，学習者の行動を分析しなければならないと考えている．そのための手段として，学習者がプログラム実行環境上で，どのような行動をとるかという，行動データを収集し，解析する．

## 3.2　プログラム実行環境の設計

設計方針を基に，プログラム実行環境の全体構成を図２のように設計した．この図に示すように，エディタを中心に統合環境を構成する．この構成の特徴は，エディタにプラットホームの役割を持たせ，各部から出力される様々な情報を受け取らせる仕様になっていることである．この情報をやり取りするインタフェースを各部ごとに定めれば，各部の機能を細かいツールとして実現できる．また，全体を五つの部分に分け，各部の中をさらに細分化したツールとして構成することによって，保守性，拡張性を確保できる．

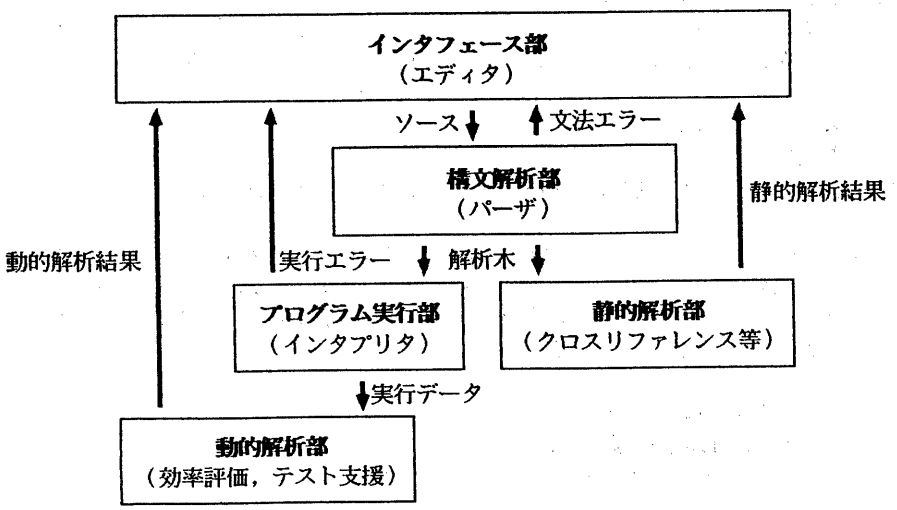


図２　プログラム実行環境の全体構成

なお，２.２で述べた CAE システムの全体構成との関係は，次のとおりである．

表１　ＣＡＥシステムとプログラム実行環境の関係

| CAE システム | プログラム実行環境 |
|---|---|
| プログラム入力部 | インタフェース部，構文解析部，静的解析部 |
| プログラム実行部 | プログラム実行部 |
| 実行結果考察部 | 動的解析部 |

次に，図２に示した各部の機能について述べる．

### ３.２.１　インタフェース部

インタフェース部では，プログラムの入力の支援を行う．具体的な支援内容を次に述べる．

（１）エラーの指摘

エディタ上から構文解析部を呼び出し，編集中のプログラムを構文解析する．構文解析部がエラーを発見した場合は，構文解析を中止し，エラーの発生した位置にカーソルを移動し，学習者はエラーの訂正を行う．このように，学習者はシステムが指摘するエラーを一つずつ，インタラクティブに取り除いていく．

（２）参照関係の指摘

プログラム中の１箇所を修正すると，その影響が複数の他の部分に及ぶことが多い．例えば，関数の引数を変更した場合，プログラム中でその関数を呼び出している箇所が影響を受ける．プログラムのサイズが大きくなればなるほど，参照関係を把握するのは困難になってくる．そこで，システムがこのような修正の影響の及ぶ箇所を指摘し，学習者が必要に応じてプログラムを修正すればよい．

また，作成したプログラムを実行するなど，プログラム実行環境の各機能を呼び出すときは，エディタ上からコマンドを発行することにより行う．これにより，学習者は，一つの閉じた環境内で学習を行うことができ，学習者の行いたいことを即座に行うことができる．なお，各機能から受け取った情報は，ウィンドウを開いてそこに出力する．

### ３.２.２　構文解析部

構文解析部は，インタフェース部から呼び出され，編集中のプログラムを構文解析し，解析木を生成する．この構文解析部の特徴を述べる．

（１）生成する解析木は，対応するソースプログラムへの逆ポインタを持つ

文法エラーや実行時エラーをソースレベルで指摘するためには，解析木が対応するソースプログラムへの逆ポインタを持っていなければならない．また，解析木がソースプログラムへの逆ポインタを持つことにより，これを入力とするツールが，出力とソースプログラムを対応づけることができる．

（２）参照関係を管理している

前述した参照関係の指摘や，後述する静的解析ツールがプログラムの参照関係を抽出するために必要となるデータを管理する．

（３）任意の構文要素を構文解析の対象として指定できる

一度構文解析した結果を保存していて，ソースプログラムの修正が加えられた部分だけを構文解析を行う．これにより，３．２．１で述べた，インタラクティブな文法エラーの除去を，効率的に行うことができる．

### ３．２．３　プログラム実行部

プログラム実行部は，コースウェアの例題や学習者の作成したプログラムの実行を行う言語 C インタプリタで構成される．これは，教育用のシステムであることを考慮すると，システムが実行時エラーを把握し，それを学習者に通知する必要があると考えたからである．インタプリタでないとこのようなことはできない．

プログラム実行部の特徴を述べる．

（１）内部状態の変化をアニメーション表示できる

学習者にプログラムの内部動作を解説する場合，実際の動作を図解したほうが効果的である．ここでは，データ構造を図示するとともに，変数の内容が書き変わったときに，変数の内容や構造が動的に変化する様子を表示する．また，関数コール時にスタックに値が積まれる様子を動的に表示する．

（２）デバッガとしての機能を持つ

数人の学習者のプログラミングを観察したところ，学習者は，学習時間の約８割をデバッグに費やしている．ここでは，これを支援するために，学習者の欲しい情報をいつでも得られるように，システムが持つすべての情報を公開する．これにより，学習者がプログラムの間違いを，学習者自身が調べることができる．

（３）実行時データの計測を行う

プログラムの動的解析を行うために必要となるデータを計測する．動的解析については，後述する．

（４）オブジェクトモジュールの追加リンクを行う

インタプリタは実行速度が遅いという欠点を持っている．この実行環境では，実行速度を向上させるために，オブジェクトモジュールを後からリンクし，実行できる．これにより，ライブラリや，デバッグする必要のないモジュールなどは高速に実行できる．これは，我々の実記憶系の OS で，動的リンクを行うことである．この方式を追加リンクと呼んでいる．

### ３．２．４　静的解析部

静的解析部は，構文解析部の出力した解析結果を入力とし，各処理を行うツール群により構成される．これらのツールは，おもに保守の支援を目的としている．具体的な内容を次に述べる．

（１）静的解析ツール

ファイルの構成，ファイルの包含関係，関数の定義を行っている位置などの情報や，定義した関数とその関数が呼び出す関数，その関数が使用している変数など，プログラム内の参照関係を抽出して出力するツールである．これらは主に，プログラムを読むときに有効な情報となる．

（２）仕様書生成ツール

関数名，関数の型，引数の型などの関数の仕様，変数名，変数の型などの変数の仕様を抜き出し，出力するツールである．これらは主に，文書化のときに有効な情報となる．

（３）プログラム整形ツール

プログラムを指定した書式にしたがって整形するツールである．プログラムを読むとき，そのプログラムがある形式に沿ってインデントされ，定まった書式で美しく記述されたプログラムのほうが可読性が高い．

### ３．２．５　動的解析部

動的解析部は，プログラム実行部が出力した実行時データを用いて，実行効率の評価支援とプログラムのテスト支援を行う．

（１）実行効率の評価支援

プログラムを評価する尺度として，実行効率が挙げられる．プログラム実行部が出力した実行時データを比較することによって，相対的な評価が可能になる．例えば，演習問題の模範回答と学習者の作成したプログラムの実行時データを比較すれば，両者の実行効率の差を数値で示すことができ，学習者に改善の方針を示すことができる．

また，同じ処理を行うプログラムを，複数の書き方で作成し，メモリの使用量や演算回数などの比較を行うことにより，トレードオフの問題を説明できる．

（２）プログラムテスト支援

トレースデータと統計量データを用いて，未実行の関数やパスを指摘する．学習者は，まだ通っていないパスを通るようなデータを入力として，プログラム実行部を呼び出し，全パステストを行うことができる．

また，プログラム実行部でエラーが発生しなくても，実行結果が正しいとは限らない．そこで，入力と変数の途中結果の取るべき値を与え，実際のトレースと比較する．両者の間に矛盾が生じた位置が，プログラムの間違いの位置であり，それを指摘する．

## ４．学習者の行動分析

### ４．１　学習者の行動分析の目的

ここでは，学習者の行動分析の目的を述べる．

（１）学習者の個別評価を行う

CAE システムでは，学習者の自主性と創造性を重視している．このため，システムは学習者に応じた指導ができなければならない．この前段階として，学習者ごとの行動を分析し，学習者によってどのような誤りのパタンがあるかなど，学習者間の差を明らかにする．

（２）学習者モデル作成，評価のための基礎データとする

行動分析の結果をもとに，学習者モデルを作成する．また，作成したモデルが正しかった検証を行い，評価を行うためにも，学習者の行動分析を行う．

### ４．２　学習者の行動分析のためのデータ収集

学習者の行動を分析する方法として，学習者がプログラム実行環境上で，どのような行動を取っているかをデータとして記録し，それを解析する方法を採用した．ここでは，データとして何を収集するかについて述べる．

（１）学習者が発生させたエラー

学習者によってどのようなエラーを犯すのかを調査し，分類する．しかし，単にエラーメッセージを記録するだけでは，正確な分類は行えない．システムの出力するエラーメッセージは，必

ずしも同じエラーを同定しているとは限らないからである．したがって，エラーメッセージを学習者がどう解釈したかを何らかの形で残さなければならない．これには，後述する編集の履歴との関係を解析することによって行う．

（２）学習者が実行したコマンドとその結果

学習者がデバッグを行うとき，どのように誤りを発見するのかを調査する．一般にデバッグは，内部状態を何らかの形で知り，それが正しい状態か否か判断し，正しい状態でなければ，ソースプログラムを編集するという形で行われる．したがって，学習者が内部状態をどのように調べ，どう解釈したかを記録する．具体的には，プログラム実行環境に用意したどのコマンドを使用したか，また，その結果とソースプログラムの編集の記録を解析することによって行う．

（３）ソースプログラムをどう編集したかという，編集の履歴

上記（１），（２）で述べたように，システムが出力するエラーメッセージや，実行環境に用意した機能を使用した結果を，学習者がどう解釈したかを調査するために，ソースプログラムをどう編集したかという，編集の履歴をデータとして取る．これは，学習者が，それらをどう解釈したかが，ソースプログラムをどう編集したかに，必ず反映すると考えたからである．

具体的には，エディタ上でのキーストロークを記録することによって行う．

なお，これらのデータは，大容量の２次記憶（追記型光ディスク）に，タイムスタンプ付きのデータとして記録する．タイムスタンプ付きで取ることにより，学習者が，システムの出力する情報を解釈するのに要した時間や，誤りを発見するとき，ソースプログラム中の一部分をどのくらい注目していたかなど，どのくらいの時間考えていたかが分かる．さらに，プログラミングを始めてから，完成するまでに要した時間を計測することもできる．

これらは，学習者によって大きな差が出ることが予想される．

## ５．おわりに

言語Ｃの CAE システムとそのプログラム実行環境，学習者の行動分析について述べた．現在，プログラム実行環境の構築を行っている．現在までは，

（１）構文解析部（パーザ）
（２）プログラム実行部（インタプリタ）

が完成している．今後は，他の部分を完成させ，プログラム実行環境を実現し，学習者の行動分析を行いたいと考えている．

なお，プログラム実行環境は，まだ実現していないが，学習者の行動分析の第一歩として，パーザのコンパイルエラーと，一世代ごとのソースプログラムをデータとして収集している．

## 参考文献

［１］高橋延匡：“研究プロジェクト総説 OS/omicron の開発”，情報処理学会オペレーティングシステム研究会報告 39-5，1988
［２］森田雅夫他：“言語 C の CAI システムの設計とそのプログラム実行環境の開発”，シンポジウム報告集 pp.177-186，情報処理学会，1991.1