2005－CE－78 (2)
2005／2／18

システムソフトウェア教育支援環境「港」における実装レベルのOS学習支援システム

大角圭吾† 小久保政樹‡ 西野洋介† 早川栄一‡

あらまし 本報告ではシステムソフトウェア教育支援環境開発プロジェクト「港」の一環として行われた，オペレーティングシステム(OS)の実装レベルにおける学習支援環境の開発について述べる．

本研究では，H8ボードエミュレータ上で小規模OSを実行し，H8ボードエミュレータからOSが行っている処理に関する情報を取得し，その情報を可視化学習支援教材を用いて可視化するシステムの開発を行った．学生が実装レベルの学習内容に対して容易にイメージを得られるように，内部動作をアニメーションなどを用いて可視化した．これにより，実装レベルの学習内容が抽象化され，直感的に理解しやすい形になり，実装レベルの動作の理解を促進することが可能になった．

キーワード OS 教育支援 可視化

Development OS Implementation Level in System Software Educational Support Environment "Minato"

Keigo OSUMI† Masaki KOKUBO‡ Yosuke NISHINO† Eiichi HAYAKAWA‡

Abstract In this paper, a part of The System Software Educational Support Environment "Minato" was implemented. It focuses on the development of an (OS) Operating system's implementation level to support educational environment.

We implements the board emulator H8 for the OS. H8 interacts with OS processes and accumulates information for visualization. To help the student understand, the learning content is represented with images. Furthermore, the inner operation is used with animation for visualization. As a result, the visualization content becomes an abstraction of the OS's operations, shape helps understand changes and implementation level operation accelerates understanding of the OS.

Keywords Operating System Education Support Visualization

1. まえがき

情報工学を学ぶ学生にとって，オペレーティングシステム（OS）やコンパイラなどのシステムソフトウェアに関する学習は，重要な学習項目の一つである[1]．学生がシステムソフトウェアの学習し理解するためには，ハードウェアとソフトウェア双方について学習を行う必要がある．

†拓殖大学大学院工学研究科
Graduate School of Engineering, Takushoku University
‡拓殖大学工学部
Faculty of Engineering,Takushoku University

しかし，学生にとっては，このように広範囲な内容について一度に学習することは難しい．これまでに，学生がより理解しやすい形で学習を行える環境の開発が行われてきた．

我々は，学生が理解しやすい学習環境を用意するために，システムソフトウェア教育支援環境「港」の開発を進めてきた．「港」において，OSの教育支援として可視化を用いて行ってきた．これは，ブラックボックス化されているOSの動作内容をアニメーションとして可視化することで，学生の直感的な理解を促す環境の開発を行ってきた [2][3][4].

しかし，これまで「港」で行われてきた可視化を用いた研究は概念学習を主体としており，実装段階に踏み込んだ

研究はあまり行われてこなかった.

そこで本研究では，概念レベルでの学習で理解した内容と実装レベルの動作を比較し，その違いを理解するシステムの開発を目的とする．つまり，座学などにより概念や理論として学習してきた内容が，実装ではどのように動作しているのかを理解させる．本研究は，情報工学系の学科の3年生を対象とする．1〜2年生はOSに関する知識が少なく，実装レベルの学習を行っても十分な理解が得られない．4年生は，既に卒業研究などに取り組めるだけの知識や技術を習得していると考えるからである．

なお，本研究はシステムソフトウェア教育支援環境開発プロジェクト「港」の一環と位置づけられている．

2. 問題分析

2.1. 実機を用いた学習における問題

OSの学習支援を行うためには，実際にOSのソースコードに触れるのが最も効率がいい．本学では学部3年次に情報工学実験Ⅱという科目で，実際にH8ボード用のOS（以下miniOS）を用いてOSの実際の動作を学習する．miniOSはソースコードの量が2000行弱で，学生がソースコードの解析を容易に行える．また，miniOSは学生がソースコードを改変することができる．そのため，学生が自発的にソースコードを改変し，OSの動作の違いを確認しながら，その動作内容を学習する．

しかしOSの動作の変化は，具体的に目に見える違いは小さい．例えば，複数のタスクを実行したとき，どのタスクが実行されているかを視認することは難しい．学生が，改変したプログラムの動作確認をしていたとき，H8ボードがどのタスクを処理しているかがわからない，正しく動作しているかが確認できない，などの意見が上がっていた．このように，OSの動作によって，データやタスクがどのように切り換わっているのかが視認できるようにする必要がある．

2.2. 「港」プロジェクトに関する問題

「港」プロジェクトは，次の三つの項目を方針として開発が進められてきた．

- 概念から実装まで幅広く教育支援を行える環境
- 可視化を用いることで，視覚的な学習支援を行える環境
- ハードウェアとソフトウェアといった異なる学習項目間で，協調した動作によって教育支援を行える環境

2.2.1 進展上の問題

前述した方針を踏まえ，我々は主に概念学習における可視化を用いた学習支援システムの開発を行ってきた．しかし，学習範囲が概念レベルを想定して開発を行っていたことで，学習段階が実装段階まで到達しないという問題が生じた．

2.2.2 方法上の問題

これまで「港」で開発されてきた可視化を用いたOSに関する学習環境では，概念学習に対する支援に重点を置いて開発が行われてきた．しかし，学習を進めていくに従い，概念に関する内容だけを学習していても，動作を理解できなくなることがある．つまり座学で理論や概念を学習しただけでは，実際にシステムソフトウェアの開発を行えるようにはならないという問題がある．

その他に，学習が進み可視化を用いなくても学習が行える段階まで技術が身についた学生にとっては，可視化を用いた学習部分が煩わしくなってくるという問題がある．

3. 設計方針

前章で述べたそれぞれの問題を解決するために，本研究では次のような設計方針を定める．

(1) 実際のOSの動作を確認できるようにする

学生が実際にOSがどのように動作しているかを確認できるようにする．このようにすることで，学習時に状態遷移を自分で確認することが容易になる．

2.1で述べたように，本学では既にminiOSを用いた学習を行っている．本研究では，miniOSを用いて学習を行うものとする．miniOSは，学生が自分でソースコードを改変できるので，自発的に学習に取り組むことが可能である．また，改変する前後での動作の違いを確認することで，自分が変更した内容を理解できる．

(2) 可視化を用いた学習支援を行う

2.2.1で述べたように，「港」プロジェクトではOSの実際の動作を可視化することはこれまで行っていない．そこで，実機に近い環境で学習を行えるように，実際に動作しているOSの情報を可視化する．

可視化する際には，これまでに行ってきた概念レベルでの可視化に即した形での可視化を行う．これは，実際の動作を数値情報などのまま確認する形態では，学生が理解しにくい部分があるからである．動作状況をある程度抽象化することで，これまでに「港」プロジェクトで進めてきた概念レベルでの可視化を用いた学習と比較検討を行うことができる．これはOSの概念学習の際に学んだ内容と実際のOSで行われる動作の違いを理解する上で重要である．

(3)実機に近い環境で学習できるようにする

2.2.2 で述べたように，学習を進めていくうちに，抽象化した情報のみで学習していると，実装とはかけ離れた学習になってしまう．そこで，実際に OS が行った処理などにより，どのようにメモリやレジスタの内容が変化するのかといったことをを確認できる方が，より実装に即した学習を行うことができる．しかし，実際のハードウェアを用いたのではこれらの情報の取得は難しい．そこで，本研究は H8 ボードのエミュレータを用いて学習を行うものとし，可視化部分を用いなくても動作する環境を開発する．

4. 設計

本システムは，可視化対象モデル，制御部，可視化コンポーネントの三部分で構成される．また，本システムは MVC(Model-View-Control)モデルを基に構築を行った．可視化対象モデルが Model に相当し，View は可視化コンポーネント，Control は制御部が相当する．このような構造にすることで，可視化対象モデルを切り換えるだけで実装レベルのシステムと概念レベルのシステムの二つを構築することができる．

図 1 に本システムの基本構成を示す．

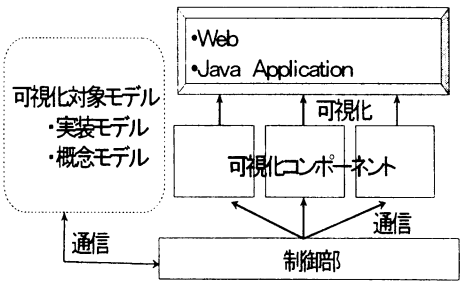

図 1 基本構成

4.1. 可視化対象モデル

可視化対象モデルは，実際に学習する内容に当たり，本研究では，実装モデルと概念モデルの二つがある．

過去に開発を行ってきたのは，主に概念学習を念頭に置いた学習支援教材であった．つまり，可視化対象モデルが概念レベルの学習内容を支援しているだけだった．今回は，可視化対象モデルを H8 ボードエミュレータを用いることができるようになっているので，概念と実装との動作比較を行える形になっている．

4.1.1. 実装モデル

H8 ボードエミュレータ上で miniOS を動作させることで，実機に近い状態の OS の動作を確認するためのモデルである．図 2 に実装モデルの構成を示す．

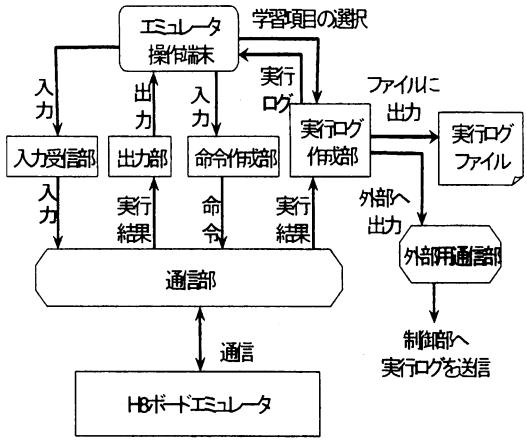

図 2 実装モデル

実装モデルは，単体でも動作することを想定して設計を行った．これは，可視化に関連した部分を省いて動作させることができるようにするためである．可視化を行うとアニメーションなどにより処理量が増加するので，個々の動作が遅くなってしまう．実装についての初期段階の学習では，処理量が増えたとしても個々の動作を順を追って確認する方が理解しやすい．しかし，学習段階が進み，可視化を必要としなくなった学生にとっては，可視化を行っていると処理の多さが煩わしい．このような問題を回避するために，実装モデルは単体でも動作するようにする．

各部の機能は次の通りである．

(1)エミュレータ操作端末

ユーザが H8 ボードエミュレータに対しての入力を行ったり，H8 ボードエミュレータからの出力を受け取り表示するためのインタフェースである．

この部分で行う入出力は次の二つである．

・H8 ボードエミュレータに対しての入出力

これは H8 ボードエミュレータに対して直接行う入出力である．具体的には OS を実行や停止，各種仮想ハードウェア情報の取得などがこれにあたる．

・実行中のアプリケーションに対しての入出力

これは，miniOS 上で実際に動作しているアプリケーションに対しての入出力である．

(2)入力受信部

(1)から miniOS 上で動作しているアプリケーションに対しての入力を受信する部分である．受け取った入力内容を，(6)へ送信する．

(3)出力部

(2)からの入力情報に対して，miniOS 上で動作している

アプリケーションが返してきた応答を，(1)へ出力する部分である．

(4)命令作成部

ユーザが(1)から入力したH8ボードエミュレータの制御命令を，(6)へ送信する部分である．

(5)実行ログ作成部

(2)や(4)が行った入力に対して，H8ボードエミュレータが返してきた応答を受信して，実行ログとして編集し，(1)や制御部へ出力する部分である．ユーザはこの部分に対して，あらかじめ学習項目を指定しておく．実行ログ作成部は，指定された学習項目に従ってH8ボードエミュレータが出力した応答を編集し，実行ログを作成する．作成された実行ログは，(1)で表示したり制御部に送信される．制御部は実行ログを解釈して，可視化コンポーネントへ可視化に必要な情報を送る．また，実行ログをファイル形式で保存しておくこともできる．これは，それまでに実行した動作との比較を行えるようにするためである．

(6)通信部

H8ボードエミュレータとの通信を行う部分である．

この部分は，H8ボードエミュレータとソケットで接続し相互通信を行う．

H8ボードエミュレータは，TCP/IPプロトコルで通信を行うことが可能で，テキストベースの通信を行って，入出力を行う．(2)(4)から送られてきたユーザ入力をH8ボードエミュレータに送信し，その応答を(3)(5)に送信する．

(7)H8ボードエミュレータ

ユーザは(1)を操作して，H8ボードエミュレータに自分が改変したminiOSのソースコードをロードし実行する．実行したminiOS上でアプリケーションを実行して動作させる．H8ボードエミュレータへの入出力は，通信部を通じて行う．

4.1.2. 概念モデル

概念モデルは，OSの基本機能についての学習内容を，高水準言語を用いて構築した，データなどの状態を遷移させるものである．図3に概念モデルの基本構成を示す．

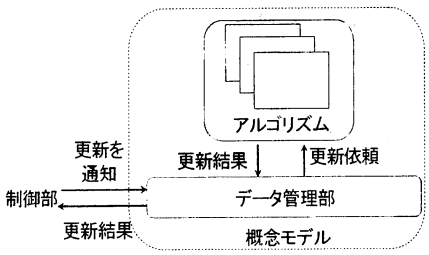

図3　概念モデル

各概念モデルは次の二つの部分で構成されている．

(1)データ管理部

それぞれの学習内容を表すのに必要なデータを管理する部分である．データ管理部では，制御部からの更新依頼などの通知を受けて，通知内容に従ってデータの更新を(2)に指示する．データが更新された場合は，データを制御部に送信し，可視化コンポーネントを更新するよう依頼する．

(2)アルゴリズム

制御部からの依頼を受けた(1)が，更新を依頼する部分である．つまり，実質的な学習内容にあたる部分である．(1)からのデータ更新依頼に対して，更新したデータを返す．

ユーザは，制御部からどのアルゴリズムを用いて更新を行うかを選択する．

本研究では，次の六つの学習内容についての概念モデルが用意されている．

・タスクスケジューリング
・メモリ管理（フィッティングアルゴリズム）
・メモリ管理（ページングアルゴリズム）
・セマフォ
・iノード
・FAT

4.2. 可視化コンポーネント

可視化コンポーネントは，可視化対象モデルにおける動作状況をアニメーションを用いて可視化する部分である．ユーザは制御部から，自分が学習したい内容に関する出力をもつコンポーネントを選択し，アニメーションを実行させる．

可視化部分をコンポーネント化することにより，様々な形態で可視化対象モデルの情報を出力することが出来るようになっている．つまり，教授者が新たなコンポーネントを作成することで，同じ学習内容であっても，違った視点からの表現で示すことが可能になっている．コンポーネント化されていることにより，他の部分の構築を作成しなおす必要がないため，教材の作成時間の短縮というメリットもある．また，一つの可視化対象モデルに対して，複数の可視化コンポーネントを実行することも可能になっている．これは，同じ学習内容を違った表現で表すことで，各学習項目間の協調した動作を学習できるようにするためである．

4.3. 制御部

ユーザインタフェースと各部との通信機能を担う部分である．また，可視化対象モデルから受信したデータの表示

などもこの部分が行う．ユーザはここから，可視化対象モデルの制御を行う．可視化コンポーネントに対しては，アニメーションの実行や停止といったどのコンポーネントでも必須となる機能は，制御部が行うものとする．これは，4.2で述べたように一つの可視化対象モデルに対して，複数の可視化コンポーネントを起動することも可能な形になっているからである．コンポーネントごとに実行や停止といった機能を実装してしまうと，相互に協調した動作を行えなくなってしまうからである．

概念モデルに対しては，データの生成や削除，更新などの要求を行う．しかし，実装モデルに対しては，実装モデルを起動する以外の操作は行うことができない．これは，5.1.2で述べたように実行ログ解釈機構にH8ボードエミュレータの操作端末としての機能を備えさせているためである．

5. 実現

本研究はWindows上でJava5.0を用いて開発を行った．各部の詳細を次に示す．

5.1. 可視化対象モデル

5.1.1. 実装モデル

実装モデルの実行画面を図4に示す．

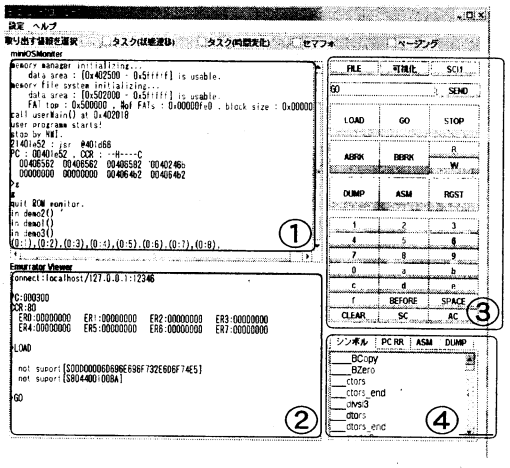

図4 実装モデル

①はminiOSの実行画面で，miniOSに対する命令を入力したり，miniOS上で動作しているアプリケーションの入出力を行う部分である．

②はH8ボードエミュレータに対して送信した命令に対する応答を表示する部分である．

③は，ユーザがエミュレータに対して送信する命令を作成したり，命令を送信したりするためのコンソールである．

④は，miniOSの情報を管理するためのタブである．それぞれのタブで，関数などのシンボル，プログラムカウンタやレジスタの値，ディスアセンブルリスト，メモリのダンプ結果を表示する．

ユーザが③を使ってH8ボードエミュレータに対して送った命令の応答が，このタブに保存される．また，③で命令を作成する際に，メモリアドレスの代わりにこのタブからシンボルを選択することもできる．

実行ログ作成部は，このタブに保存されている情報から，制御部が解釈できる形のデータを生成し，要求に応じて制御部にデータを送信する．

5.1.2. 概念モデル

概念モデルは，4.1.2で述べた六つのモデルににについて開発を行った．ユーザは制御部からどのアルゴリズムを用いてデータの生成などを行うかを選択し，必要なパラメータを設定する．

5.2. 可視化コンポーネント

可視化コンポーネントの規模は，平均して300行程度である．この程度であれば，新たな表示方法を作成する場合も，大きな手間がかかることもなく，容易に作成可能であると考える．図5にタスクスケジューリングにおいての可視化コンポーネントを示す．

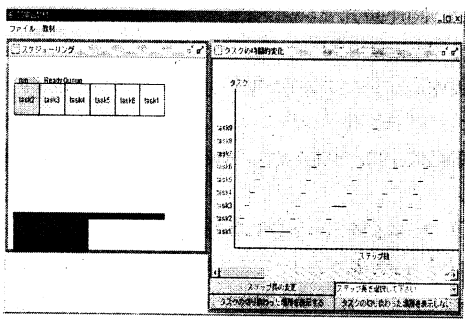

図5 可視化コンポーネントの例

図5は左右で同じモデルを表示している．左側は，タスクを一つのブロックで表現し，ブロックをアニメーションさせることで，タスクの状態遷移を表現したものである．右側は，タスクが実行中状態である時間をグラフ化して表したものである．

今回は，注目するポイントを実際の動作と時間的な変化の二つに切り換えることで，別の視点から捉えられる形態になっている．このように，同じモデルであっても表示形態を複数用意することで，学習内容の幅が大きく広がると考える．

5.3. 制御部

制御部では，概念モデルの操作及びアニメーションの実行・停止を行う．新たな概念モデルを追加する際に，そのモデル専用の操作がある場合は，その操作用の機能を追加する必要がある．図6に操作パネルの実行例を示す．

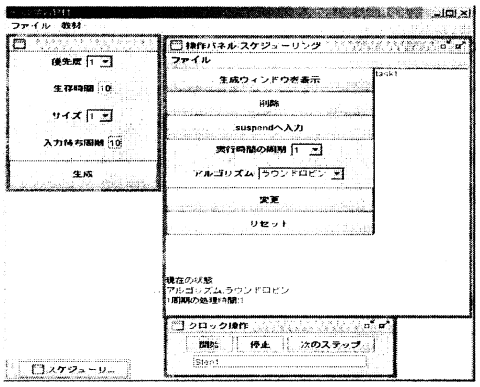

図6　操作パネル

制御部は可視化対象モデルから送られてきたデータを解釈し，可視化コンポーネントにアニメーションに必要な情報を送信する．可視化コンポーネントは，受信した情報を基に可視化対象モデルの動作状況を可視化する．

6.　関連研究

6.1　Visual OS [5]

仮想的なハードウェアを規定し，それらの動作を確認するという手法が用いられている．しかし，表示される情報が複雑なために，学生が把握するのが難しいという問題がある．

6.2. たとえ話を用いたOSの学習支援システム[6]

たとえ話が基本となっているために，概念的には理解しやすい．しかし，実際の動作に即していない部分があるという問題がある．

7.　おわりに

本報告では，システムソフトウェア教育支援環境「港」における，可視化を用いた実装レベルのOSの学習支援システムについて述べた．

本研究によって，実装レベルと概念レベルの差異を，視覚的に確認しながら理解する環境を実現することができた．また，OSの実際のソースコードを改変した際の動作なども確認することができるようになったことで，実装レベルの学習を支援することができるようになった．また，概念モデルと実装モデルを用意したことで，相互に比較検討が行えるようになった．そして，概念モデルを複数用意することで，様々な概念学習に対応することができるようになっている．

各部分の規模を表1に示す．miniOSは学習する側が改変可能なのでソースコード量は可変だが，初期状態は表1で示した程度の規模である．

表1　各部分の規模

構成部分		行数
実装モデル		2000
miniOS		2000
可視化コンポーネント		200~400
制御部		1000
概念モデル	データ管理部	300~350
	アルゴリズム	100~250

今後の課題としては，学生がminiOSの機能を拡張した場合への対応がある．miniOSは学生が機能を拡張することで，OSの実装を学習していくことを想定しているため，OSに求められる機能は必要最小限の機能しか実装されていない．つまり，学生が新たな機能を追加した際に，その追加した内容によっては，本システムで取得できない情報が発生する可能性がある．

今後は，課題の改善と本システムのシステム評価及び実際の教育効果の評価を行う．

参考文献

[1]情報処理学会：大学の理工系学部情報系学科のためのコンピュータサイエンス教育カリキュラムJ97(第1.1版)，1999

[2]西野洋介，早川栄一：OS教育支援環境における可視化設計と実現，第2回FIT、N-023、2003

[3] 西野洋介，大角圭吾，早川栄一：OS教育支援における可視化環境の開発，電子情報通信学会技術研究報告，Vol.103 No.536，pp.83-88，2003

[4]大角圭吾，小久保政樹：システムソフトウェア教育支援環境「港」におけるOS可視化コンポーネントの開発，FIT2004，N-008、2004

[5] Manuel Estranda Seinz : Visual OS
http://visualos.sourceforge.net/

[6]及川　聡，並木　美太郎：たとえ話を用いたOSの学習支援システムの開発，情報処理学会研究報告，No.057-006、2000