2006－CE－83（1）
2006／2／**17**

ロボットを用いた可視化による学習支援システム

大角圭吾† 川上亮太郎† 田中裕樹† 西野洋介† 川口貴弘‡ 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)学習用の可視化を用いた学習支援システムについて述べる。本研究では、可視化を用いることで、学習者が OS の動作イメージを容易に理解することができる。また、学習対象として概念だけでなくシミュレータやロボットを用いることで、様々な学習段階の学習者に対応した学習支援を行うことができる。また、学習支援システムのフレームワークを構築することで、学習支援システムを容易に構築することができる。

キーワード　　OS　教育支援　可視化

Development Leaning Support System for Using Robot and Visualization in System Software Educational Support Environment “Minato”

Keigo OSUMI†　Ryotaro KAWAKAMI†　Yuki TANAKA†　Yosuke NISHINO†
Takahiro KAWAGUCHI‡　Eiichi HAYAKAWA‡

Abstract　　In this paper we describes the visualization based learning support system for operating system(OS) that is a part of “Minato” system software learning support environment project. Our systems features are following:(1)visualization of state transitions in OS to help learning OS behavior, (2)utilizing CPU simulators and robot hardwares to adapt several learning classes, (3)presenting a framework and components for easy construction of several OS learning support systems.

Keywords　　Operating System　Education Support　Visualization

1. はじめに

情報工学を学ぶ学生にとって、システムソフトウェアについての学習は重要なものである。システムソフトウェアは、コンピュータアーキテクチャやコンパイラ、OS などのコンピュータの中核を担う分野にあたり、コンピュータを学ぶ上で必須となっているからである。その中でも OS は、ハードウェアとソフトウェアを管理し仮想化する最重要な分野であり、OS を学習し理解することは、情報工学においてコアカリキュラムの一つとして位置づけられている[1]。

† 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、情報工学を学び始めたばかりの学習者にとって、OS の主な機能である次の三つを理解させるのは難しい。
(1)コンピュータ利用効率の向上
(2)ハードウェアの抽象化
(3)資源管理

これらのうち(1)については、コンピュータを利用していると、ユーザインタフェースなどが対応しているので、学習者にとっては、コンピュータを利用する際に接することが多く理解しやすい。しかし、(2)(3)はユーザにこれらの処理を意識させないための仕組みである。そのため、学習者はこれらの動作に意識的に接する機会が少なく、OS の動作に関するイメージを理解しにくい。このように OS 学習を行う際には、学習者が普段意識しない部分についての学習を行うということを念頭に入れ、学習者に OS の動作に対するイメージを持たせることが重要である。

そこで本研究では、学習者が OS の動作イメージを直感的に理解するための学習支援システムを構築するためのフレームワークの開発を目的とする。これを解決するために、OS の動作をアニメーションなどで可視化することで、学習者がイメージ付けできるようにする。

また、OS 学習は概念的内容から実装的内容まで幅広く対応する必要がある。この問題を解決するために、本研究では、動作概念を再現するための概念モジュール、実際の動作を確認するためのシミュレータ、実ハードウェア上での動作を学習するためのロボットを提供する。これらの学習対象を利用して、様々な学習段階に対応していく。

本研究は、拓殖大学工学部早川研究室で行われている、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」の一環である。

2.「港」プロジェクト

「港」プロジェクトは、学習者がシステムソフトウェアに関する概念的知識や、実装に関する知識を習得するための教育を支援するための環境を開発するプロジェクトである。本節では「港」プロジェクトについて述べる。

2.1 システムソフトウェア学習の問題点

システムソフトウェアに関する学習が困難な理由には次の三つの問題がある。

(1)システムソフトウェアの動作が視覚的に不明確

システムソフトウェアの動作は、普段コンピュータを利用している際に視覚的な部分に現れることは少ない。そのため、学習者はシステムソフトウェアの動作イメージを理解しにくい。

(2)概念学習と実装学習における内容の乖離

講義で学習したシステムソフトウェアの概念を、実際にシステムソフトウェアを作成するなどの実装段階の学習に利用していく場合に、動作概念と実装における動作が結びつかない。

(3)ハードウェアとソフトウェアの協調動作の理解が必要

システムソフトウェアを開発するには、ハードウェアに関する知識と、ハードウェアを実装環境にどう対応させるかを理解することが必要となる。そして、双方の能力を結びつけるために、ハードウェアとソフトウェアの協調動作を理解する必要がある。

2.2「港」の設計方針

「港」は 2.1 で述べた問題点を解決するために、次のような方針を基に設計されている。

(1)可視化を用いた視覚的な教育支援を行える環境

システムソフトウェアのように、動作が視覚的に明確でないものは、学習者にとって動作を理解するのは難しい。そこで可視化を用いることで、学習者が学習対象を視覚的に捉えることができるようにする。これによって、学習者はシステムソフトウェアの動作イメージを身に付けることができ、動作を理解しやすくなる。

(2)概念から実装まで幅広く教育支援を行える環境

システムソフトウェアを学習する際に、概念と実装を切り離して学習すると、それらがどのように結びついているかを理解しにくい。そこで、概念から実装までを一連の学習環境で学習を行えるようにすることでそれらの結びつきを理解する。

(3)異なる学習項目間での協調した動作によって教育支援を行える環境

システムソフトウェアに関する学習では、ハードウェアやソフトウェアのような異なる学習内容が相互に関係していることを理解しなければならない。このことから、異なる学習項目が協調した動作をしていることを表現するような教育支援を行えることが必要となる。

2.3 学習段階と可視化レベル

「港」プロジェクトでは、概念から実装へ向かって学習を行うトップダウン形式の学習を採用している。これは、ボトムアップ形式の学習の場合、実装段階の演習で満足してしまい、概念学習まで到達しないことがあるからである。

図 1 に「港」プロジェクトで想定する学習段階と、その学習段階における可視化の関係を示す。

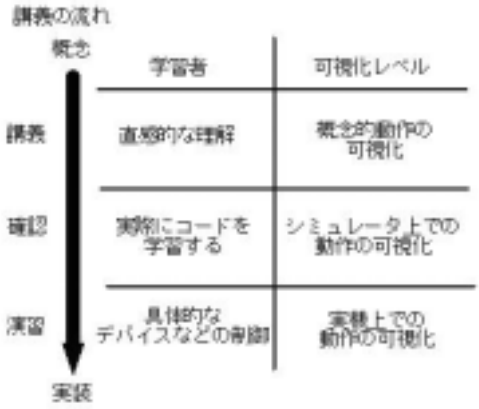

図 1 学習段階と可視化レベル

講義段階では、学習対象に対する直感的な理解を促すために、動作概念の可視化を行う。確認段階では、実際にシ

ステムソフトウェアのコードを記述して、シミュレータ上で動作させ、その動作を可視化する。演習段階では、具体的なデバイスなどの制御を学習するために、実機上でシステムソフトウェアを動作させ、その動作を可視化する。

2.4「港」における本研究の位置づけ

「港」プロジェクトは 2.2 で示した方針と 2.3 で示した学習段階を想定して設計されており、個々のシステムでの独立した動作はもちろんのこと、それぞれが強調したどうも行うことができる。「港」における学習教材の基本構成を図 2 に示す。

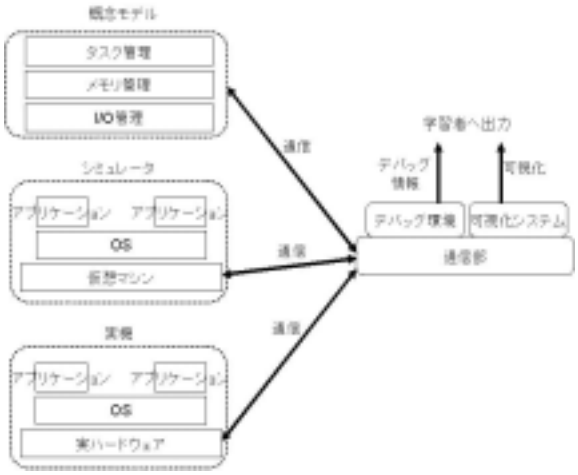

図 2 「港」における学習教材の構成

概念モデルは、システムソフトウェアの個々の機能を学習するための教材であり、講義段階で利用する。シミュレータは、確認段階で利用し、実機は演習段階で利用する。

本研究では、「港」における通信部と可視化システムのフレームワーク及びコンテンツの開発を行っていく。

3. 問題分析

3.1OS 学習の問題点

学習対象としての OS には、次のような問題点がある。

(1)OS の動作が視覚的に不明瞭

OS の動作のうち、ハードウェアの抽象化や資源管理は、普段コンピュータを利用している際には、ユーザの目に触れにくい。このように、動作を見ることができない学習対象は、学習者にとって理解しにくい。また、学習者に動作を見せる場合でも、数値の羅列のような一見して動作がわからないものでは、学習者は動作を理解できない。

(2)統合的学習環境の不在

SOsim[2]や VisualOS[3]といった既存の学習支援システムは、OS の動作を抽象化して動作概念に基づいた学習支援を行うシステムである。また、教育用コンピュータ TeC[4]のように、ハードウェアを提供し、提供されたハードウェアを改変することで学習を行っていくものも存在する。

しかし、このような既存の学習対象は、概念と実装の両端に偏った学習支援を行うものであり、統合的な学習支援を行うものではない。このように既存の学習環境には、概念と実装を結びつけた環境は少ない。

(3)新規教材の開発にかかる手間の大きさ

OS の学習は、講義段階における理論学習と演習段階における実践学習を繰り返し行うことが、学習者の理解を促進させる。OS はハードウェアとソフトウェアの両方に関する知識が必要であるため、理論学習だけでは実践が伴わなくなるといった問題が起こる。しかし、個々の学習項目ごとに演習環境を用意したり、その後の理論学習につながる教材を用意するのは、教授者側の作成にかかる手間が大きい。また学習者にとっても、新規教材が作成されるたびに操作方法を覚えるのは、大きな負担となる。

3.2 要求

本研究では、3.1 で述べた問題点を「港」の方針に沿って解決するために、OS 学習支援環境の要求仕様を次のように定めた。

(1)視覚的・動的な学習支援を行える

OS は常に動的に状態が変化している。これに対して静的な図を用いた状態繊維の表現だけでは、動的に状態を追いかけることが難しい。それに加え、OS の動作は視覚的に明確でない。

これに対応するために、アニメーションなどの動的に変化するものを用いて、視覚的・動的な学習支援を行っていく。しかし、実行するたびに動作内容に違いがあっては、動作の再現性が低く学習に不向きなものになってしまう。そこで、動作を何度も再現できることが必要となる。

(2)学習支援環境に連続性を持たせる

OS は一つの学習項目に対しても様々な視点や粒度が存在する。個々の視点や粒度に対して教材を作成していくことは可能だが、同じ学習内容を表している教材では、表現方法に違いがあっても動作は同じものでなければならない。動作が変わってしまうと、同じ内容を学習しているという認識がなくなり、視点や粒度が変わるたびに始めから学習しなおすことになってしまう。これでは、学習の連続性は保たれなくなってしまう。

このようなことから、視点や粒度が変わっても学習内容の動作は協調したものになっていなければならない。

(3)学習項目に応じた学習教材が容易に構築できる

OS 学習においては、講義と演習を繰り返して学習を進めることが理想だが、学習項目が変わるたびに学習教材を再構築するのは、学習者と教授者の双方に手間がかかる。

そこで、学習教材を容易に構築できることが必要である。また、(2)を満たすために個々の学習項目に関する教材が単独でも動作し、連動した際には教材同士が強調した動作を行う必要がある。また、学習者の手間を軽減するために、教材のユーザインタフェースを共通化するなどして、新規教材の操作性を既存の教材に近いものにすることも必要である。

4. 設計方針

3.2 で述べた要求に合わせ、本システムでは次のように設計方針を定める。

(1)学習対象の動作の可視化

学習者が OS の動作を視覚的に理解できるようにするために、本システムでは OS の動作を可視化する。この際に、数値を羅列するような形式ではなく、アニメーションなどを用いた可視化を用いる。学習対象である OS の動作をアニメーションなどで抽象化し可視化することで、講義段階では動作概念のイメージ付けができる。また、確認段階や演習段階の学習者にとっても、動作を視覚的に捉えることができるので、実際の OS がどのような動作をしているかを理解することができるようになる。

(2)学習者の習熟度に応じた学習支援

本システムでは、図 1 で示した「港」における学習段階に合わせて、学習者の習熟度に応じた学習支援を行う。

講義段階では、OS の動作を抽象化し動作概念として可視化を行う。この際に、OS のすべての機能を一度に可視化すると、学習者が動作を把握できない。そこで、OS の個々の機能ごとに可視化していく。こうすることで、機能に注目した学習を行わせる。また、個々の機能が連携した動作を行えるようにしておくことで、機能のつながりを学習できるようにする。

確認段階では、シミュレータ上で OS を動作させ、その動作を可視化する。学習者は、OS の既存の機能を改変したり新たな機能を追加することで、実際の OS の動作を学習していく。また、講義段階で利用した個々の機能の可視化と比較することで、動作概念と実際の OS の動作の違いを学習する。

演習段階では、ロボット上で OS を動作させ、その動作を可視化する。学習者は、OS を改変したり、ロボットの制御プログラムを記述することで、具体的なデバイスの制御について学習したり、実機とシミュレータでの動作の違いを学習する。

(3)機能のモジュール化

教授者が新規教材を作成する際に、すべての機能を作成しなおすのは手間がかかる。また、新規教材と既存の教材の操作性が大きく異なる場合は、学習者に操作を覚えなおす手間がかかる。

これらの問題を解消するために、ユーザインタフェースや可視化の実行・停止といった、どの教材でも必要となる機能をモジュール化することで、教材の再利用性を高める。

5. 設計

5.1 全体構成

本システムでは、学習対象をアニメーションやグラフを用いて可視化する。これによって、講義段階における動作概念のイメージを学習したり、確認段階や演習段階において OS の動作を確認したりすることができる。

学習対象は、講義段階で利用する概念モジュール、確認段階で利用するシミュレータ、演習段階で利用するロボットの三つを用意する。概念モジュールは OS の機能の一部を再現して学習に利用し、シミュレータとロボットは実際に OS を動作させることで、OS の動作を学習する。

可視化部は、学習対象の動作を可視化する部分である。学習者は、学習したい内容に合わせた可視化部を選択し実行する。可視化部の表現から、動作概念のイメージや、実際の OS の動作を確認する。

システムの全体構成を図 3 に示す。

図 3　全体構成

5.2 学習対象

学習対象は、学習者が実際に学習する内容となる部分である。本システムでは、講義段階で利用する概念モジュール、確認段階で利用するシミュレータ、演習段階で利用するロボットの三つを用いる。

5.2.1 概念モジュール

講義段階の学習者はOS学習を始めたばかりなので、OSの全体構成を把握するのは難しい。また、OS の主な役割についても、その動作イメージがわからないため理解できないことが多い。

そこでOSの機能全体ではなく、タスク管理やメモリ管理といった個々の機能に注目し、それらの機能を再現したものを概念モジュールという学習対象として提供する。これらは、一つ一つを独立した内容として学習に利用するだけでなく、並列に動作させることでそれぞれの概念モジュールが協調した動作を行えるようにする。このようにすることで、まずは小さい機能の動作を学習させ、学習が進むに従い、概念モジュール同士を協調して動作させることで、それぞれの機能のつながりを学習することができる。

概念モジュールの構成を図4に示す。

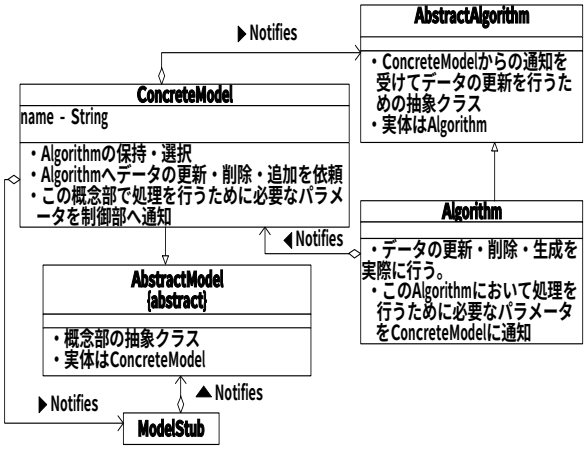

図4　概念モジュール

AbstractModelは、可視化制御機構と通信するための抽象クラスであり、ConcreteModelは概念モジュールを管理する実体である。学習者は、ConcreteModelから学習したいアルゴリズムを選択し実行する。また、ConcreteModelはその概念モジュールで必要となるパラメータを、可視化制御機構に通知する機能を持つ。これは、概念モジュールごとに必要なパラメータが異なるからである。例えば、タスク管理では、タスクの名前や優先度、周期といったパラメータが必要となるが、メモリ管理ではこういったパラメータは必要なく、データのサイズや先頭番地といった情報が必要となる。ConcreteModelは、必要なパラメータを可視化制御機構に通知し、ユーザからの入力を受け付けてパラメータを設定できるようにする。

AbstractAlgorithm はデータの更新を行うためのアルゴリズムの抽象クラスで、その実体であるConreteAlgorithmが実際にデータの更新を行う。教授者は、ある学習内容について、新たなアルゴリズムを追加したい場合は、Concrete Algorithm を作成することで、学習内容にアルゴリズムを追加していく。

5.2.2 シミュレータ

実際にOS を動作させることで、OS の具体的な動作を学習したり、動作概念と実際の動作の比較を行うための学習対象である。

本システムでは、2004年度に拓殖大学早川研究室で開発されたH8/3069F ボード(以下H8 ボード)のシミュレータとH8 ボード用のOS(以下miniOS)を利用する。miniOSは次の機能が実装されている。

(1)メモリ管理

2 メガバイトの外部RAM を管理する。実際には、仮想ベクタテーブルやOS 自身のコードを管理する。ユーザの要求によってメモリ領域を提供するが、解放し再利用する機能はない。

(2)割込み管理

仮想ベクタテーブルを管理する。ユーザがデバイスドライバなどの割込みを扱うプログラムを記述する場合、割込み管理が提供するインタフェースを介してハンドラの登録を行う。

(3)タスク管理

タスク管理では、ごく単純なタスクモデルを提供する。タスクはCPU コンテキストだけを持ち、スケジューリングは10 ミリ秒ごとのラウンドロビンスケジューリングが実行される。同期及び吐いた機能としてPV セマフォが提供される。

(4)ファイルシステム

H8 ボードシミュレータにはディスクデバイスがないため、外部RAM の一部をディスクの代用として扱うメモリファイルシステムとなっている。内部構造はFAT16 が採用されている。

miniOS の構成は、このようなシンプルなものとなっており、コード量が2500 行程度と可読性もよい。このような点から、確認段階の学習者が利用する学習対象として適

切である。

学習者は、これらの機能を改変したり、不足している機能を追加することで、OS の構造を学習していく。例えばタスク管理において、タスクに優先度を付加することで、優先度付きラウンドロビンスケジューリングを実装するといった拡張を行っていく。

また、2004 年度に H8 ボードシミュレータ用の入出力インタフェースも開発されており[5]、これを利用することで、学習者は H8 ボードシミュレータの操作を容易に行うことができる。図 5 に入出力インタフェースを示す。

本システムでは、miniOS の動作ログを抽出し可視化することで、学習者に実際の OS の動作を学習させる。

図 5　入出力用インタフェース

5.2.3 ロボット

本システムでは、演習段階の学習対象としてロボットを利用する。このロボットは「港」プロジェクトの一環として行われている、ロボットを用いたシステムソフトウェア教育支援環境「港 Ver..R」[6]で利用されているロボットである。ロボットは、各種センサなどの入力デバイスやモータなどの出力デバイスをもつので、デバイス制御に関する学習対象として適している。また、制御プログラムや OS で不具合が発生した場合に、挙動から誤動作が発生したことを視覚的に確認することができる。

また、ロボットに搭載されている OS の構造を次に示す。

図 6 ロボット用 OS

OS は教育用 OS とスケジューラ、ファームウェアの三つから構成される。学習者がコードを改変する部分は、教育用 OS とスケジューラの部分である。ファームウェアには、演習段階に進んだばかりの学生が扱うには難しい機能を集め、改変できないようになっている。これは、すべての機能を提示しても、規模が大きくなってしまい学習者が把握できないということが考えられるからである。コード量は、スケジューラと教育用 OS を合わせて 2000 行程度であり、OS の基本的な機能も満たしているので、学習対象として適切である。

本システムでは、教育用 OS とスケジューラの動作ログを抽出し、そのログを可視化することで、学習者に実機上での OS の動作を学習させる。

5.3 可視化制御機構

学習対象と可視化部の管理と通信、ユーザ入力を担う部分である。図 7 に構成を示す。

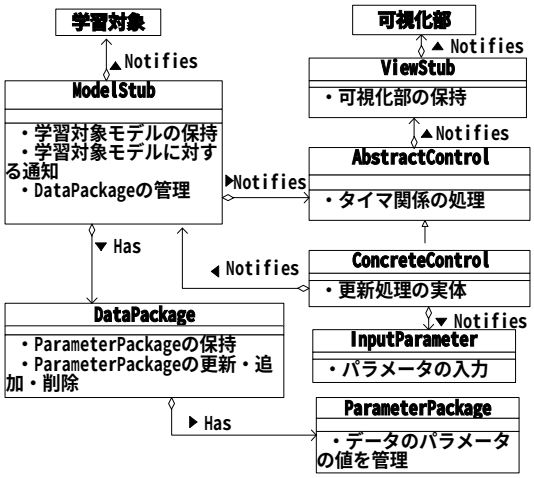

図 7　可視化制御機構

学習対象では次の三つの機能を持つ。

(1)時間管理

本システムは様々な学習対象を持つが、各学習対象において時間の扱いが異なるため、それらの整合性を取る必要がある。

そこで、可視化制御機構がタイマを保持し、そのタイマが更新依頼を出してから次の更新依頼を出すまでの時間を 1 コマと定める。各学習対象は自身のログデータを、1 単位時間分のデータごとに、指定された形式に基づいて区切った形にして可視化制御機構に送る。可視化制御機構は、学習対象から送られてきたデータから 1 コマ分のデータを抽出して可視化部へ送信する。

タイマは AbstractControl が保持し、タイマに関する制御は AbstractControl の実体である ConcreteControl が行う。

(2)ユーザ入力とデータ管理

ユーザからの入力や学習対象からのデータを受信し管理する機能である。

概念モジュールのデータは、学習内容によって必要な情報が変わるため、ユーザの入力によって生成される。そのためのユーザ入力は InputParameter が行う。入力されたデータは、DataPackage に変換され可視化部で可視化される。

シミュレータやロボットが学習対象の場合は、データが形式に従って、1 コマ分のデータに区切られた形でログデータが送られてくる。これを、1 コマごとのデータに分割し、それを更にパラメータ単位まで分割して、可視化部で利用できる形にデータを整形する。ログデータは ModelStub が解析して、DataPackage に変換する。各種パラメータは ParameterPackage として、DataPackage がそのリストを保持する。

(3)学習対象と可視化部の管理

学習対象のうち、独自で動作する機能を持たない概念モジュールと、可視化部を管理する機能である。学習者が、自身が学習したい内容を選択しそれぞれの管理部分に登録しておくことで、学習者にあった学習内容を実行することができる。概念モジュールの管理は ModelStub が行い、可視化部の管理は ViewStub が行う。

5.4 可視化部

学習対象の動作を可視化する部分である。

5.4.1 機能

学習対象での処理内容を可視化する部分である。可視化部は、可視化制御機構と通信を行うための抽象クラスである AbstractView と、処理を行う実体である ConcreteView で構成される。

可視化部の表示方法は、フレーム形式・内部フレーム形式・タブ形式が候補として考えられたが、本システムではフレーム形式を採用した。これは、フレーム形式は、表示範囲が制限されず、複数の可視化部を同時に表示することができるからである。

5.4.2 新規作成時のガイドライン

本システムにおいて、可視化部を新たに作成する際のシステム上の制約は設定していない。これは、システム上の制約を設定してしまうと、可視化の表現方法が狭まってしまい、表現について様々な粒度を用意できなくなってしまうからである。

そのため、システム上の制約は存在しないものの、学習者にとって、表現形式に統一性が無いと学習時の混乱の原因となる。そこでレイアウトに関するガイドラインを示す。

(1)配色

同じ学習内容を表現するものは、同じ色を使って表示する。これは、表現の粒度が変わっても、同じ内容を学習していることを理解しやすくするためである。

(2)レイアウト

学習者が教材を利用する際には、学習者に操作に定型化した流れを持たせることで、操作性を統一することができる。学習者が操作するにあたって、ある操作から次の操作に移るときに、その間に操作対象以外のものが目に入ってしまうと、意識が分散して学習効率が低下するからである。

そこで、本システムでは可視化部でのレイアウトを次のように規定する。

- 可視化表現はフレームの中央に配置する。
- 表示の縮尺変更といった表示に関する操作機能はフレームの上部、メニューバーなどに配置する。
- データリストはフレーム右側に配置する。
- データの情報はフレーム下部に配置する。

6.実現

本システムは WindowsXP(SP2 適用)、Eclipse3.0、Java2 Platform Standard Edition 5.0 を用いて開発を行った。

6.1 可視化システム

今回は、テストケースとしてタスク管理に関する可視化システムと、メモリ管理に関する可視化システムを作成した。図 8～9 に作成した可視化部の表示例を示す。

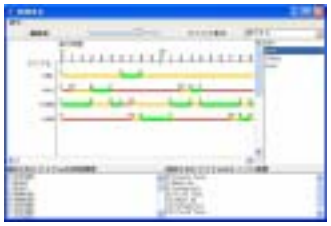

図 8.1 グラフ表現

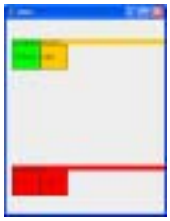

図 8.2 状態遷移

図 8 タスク管理に関する可視化例

図 9.1 フィッティング

図 9.2 ページング

図 9　メモリ管理に関する可視化例

タスク管理では、図 8.のように時間に注目した形で表現したり、状態遷移に注目した形で表現するなど粒度を切り換えることができる。メモリ管理では図 9 のように、それぞれの学習内容の違いによって大きく表現形式が変わる。

このように、学習対象の動作内容を数値の羅列のような具体的な表現ではなく、アニメーションやグラフのような抽象化した表現を用いた可視化部を作成していく。学習者に対してこのような抽象化した動的表現を提示することで、学習者に動作イメージを学習させ、その動作を直感的に理解できるようにする。一つの学習内容に対して複数の可視化表現を作成することで、学習者は学習内容を複数の粒度や視点で捉えることができる。

また、同じ可視化表現を使って、概念モジュール・シミュレータ・ロボットの動作を表現できる。これによって、学習段階に合わせた学習支援を行うことができる。また、各学習段階の実行内容を比較することで、学習段階における動作の違いを学習することができる。

6.2 規模

本システムのソースコードの規模を表 1 に示す。

表 1　システムの規模

表 1.1 可視化システムの規模

構成部分		共通部分(行)	規模（行）
可視化制御機構		698	981
概念モジュール	アルゴリズム	51	161～243
	その他	104	402～482
可視化部		347	406～778

表 1.2 各学習対象におけるログ抽出機能の規模

構成		ログ抽出部/全体(行)	割合(%)
シミュレータ	miniOS	0/2368	0
	入出力用インタフェース	413/4188	9.84
ロボット	OS	230/1849	12.44
	デバッグツール	370/14000	2.64

本システムでは、新たな学習教材を作成する場合は多くとも 1000 行程度のコードを記述することで、新規の教材を作成することが可能となった。また、概念モジュールを作成するだけなら、400~600 行程度、可視化部を作成する場合でも 100~400 行程度の記述で追加することが可能となっている。

また、シミュレータやロボットを学習対象として用いる場合、それらのコードに400～600行程度追加することで、本システムに組み込むことができる。ログ抽出部の割合も、各学習対象の10%程度の規模で追加できる形となっており、本システムに学習対象を追加するのが容易になっている。

7.おわりに

7.1 成果

本研究では、可視化を用いた OS 学習支援システムを構築するためのフレームワークの開発を行った。可視化を用いることで、学習者は OS の動作イメージを身に付けることができるようになった。また、学習対象として、概念モジュール・シミュレータ・ロボットを用意したことで、学習者の学習段階に合わせた楽手支援を行うことができるようになった。そして、機能をモジュール化することで新規教材を作成する手間を軽減することができた。

7.2 今後の課題

今後の課題として、OS が持つ多くの機能に対応するために、新規教材を開発していくことが必要である。また、近年の e-Learning の普及に対応するために Web に対応した環境の構築を行うことが挙げられる。

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]Luiz Paulo Maia : SOsim , http://www.training.com.br/sosim/

[3]Ramon Rey Vicente : VisualOS , http://visualos.sourceforge.net/

[4]重村哲至,守川和夫,力則晃,新田貴之,原田浩二,山田健仁：教育用マイコンボードを用いた HDL 演習環境の実現, 情報処理学会研究報告 2005-CE-78,pp.43-48,2005

[5]小久保政樹：H8 ボードエミュレータを用いたオペレーティングシステム学習支援教材の開発,拓殖大学 2004 年度卒業論文,2005

[6]西野洋介,田中裕樹,川上亮太郎,大角圭吾,川口貴弘,早川栄一：ロボットを用いたシステムソフトウェア教育支援環境「港 Ver.R」の実現,八王子産学公連携機構第 5 回研究成果等発表講演会,pp.180-181,2005