N-009

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
Development of OS Visualization Components on Systems Software Educational Support Environment "Minato"

大角 圭吾† 小久保 政樹‡ 西野 洋介† 早川 栄一‡
Keigo Osumi Masaki Kokubo Nishino Yosuke Eiichi Hayakawa

1. まえがき

我々は，学習者にとって困難であるシステムソフトウェア学習を支援するために，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」[1][2][3]を進めてきた．

「港」は次の方針を基に開発を進めている．

- 概念から実装まで幅広い学習を行える環境
- 可視化を用いることによる，視覚的教育支援を行える環境
- 異なる項目間において，協調した動作によって教育支援を行える環境

今回はシステムソフトウェアの中でも，OS に関する教育支援に注目した．これは情報工学を学ぶ学生にとって，OS の学習や理解はコアカリキュラムの 1 つとして位置づけられているからである[4].

本研究の目的は，「港」で用いる OS 可視化コンポーネントの作成である．可視化することで学習者が OS のイメージ付けを容易に行えるようにする．また，コンポーネント化することで，学習項目の組替えや連携を容易にする．

2. 設計方針

本研究では，設計方針を次のように定めた．

(1) OS を様々な視点から捉えることができる

学習者がある学習項目についての学習した際に，まずその項目単体での動作を理解する必要がある．OS 全体の動作を見たときに，その項目がどのような動作を行うかを学習しなければならない．そこで，学習項目間で協調した動作を行えるようにすることで，学習者が必要とする範囲の動作を見ることができる．

(2) 学習項目に対する視点の粒度を変更できる

OS の学習においては，一つの学習項目に対して複数の粒度が存在する．学習者が学習項目を多面的に捉えることができるようにするために，一つの学習項目に対して様々な粒度の視点を設ける．

(3) 動的な状態遷移を追える

OS は動的に状態が変化している．動的なものを静的な図を用いて学習しようとしても，動的な状態を追うことは難しい．

そこで本研究では，アニメーションや配色を工夫することにより，動的な状態遷移を動的に表現する．

3. 設計

3.1 全体構成

本研究では，設計方針で挙げた学習項目に応じた可視化環境を実現するために，MVC(Model-View-Control)を基にした 3 部分からなる構成とした。

†Graduate School of Engineering, Takushoku university
‡Faculty of Engineering, Takushoku university

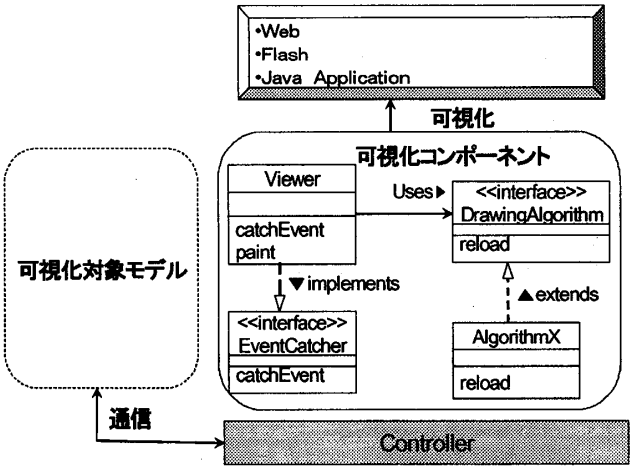

図 1 全体構成

可視化コンポーネントの構成を図 1 のような形で統一することで，可視化対象モデルや Controller を変更することなく画面への表示内容が変更することができる．

3.2 可視化コンポーネントの設計

可視化コンポーネントは，Controller から可視化対象モデルの状態遷移のデータを受け取り，可視化データに変換し出力する部分で，次のような処理手順で可視化を行う．

1. Controller からデータを EventCatcher で受け取る．
2. Viewer は EventCatcher で受け取ったデータを DrawingAlgorithm に送り，描画に必要な情報を新たに生成する．
3. Viewer は生成された描画用の情報を用いて Web, Flash, Java などの様々な出力形態で可視化を行う．

出力形態を複数持つことで，教授者が学習環境に応じた出力形態を選択することができるようになる．また，構成を統一したことで，新たなコンポーネントを作成する際に，過去に作成したものを雛形として用いることができる．これによって，教授者が新たなコンポーネントを作成する際の手間の削減ができる．

可視化コンポーネントは様々な粒度を設けることにより，学習者がある学習内容を複数の視点から捉えられることができるようにする．これは特定の視点からだけでは捉えにくい情報を別の角度から捉えることで，学習者がより深い理解を得られるようにするためである．

次にコンポーネントの粒度の一覧を示す．

表 1. 可視化コンポーネントの粒度一覧

	粒度大	粒度小
可視化コンポーネントの種類	タスク管理 仮想メモリ ファイルシステム	プロセッサ メモリ 排他制御

また，表示内容は学習者が必要なものを選択し，選択されたものだけを表示するようにする．これはあまり表示内容を多くしてしまうと，学習者が理解しなければならない

情報が多くなってしまい，そのとき理解したい情報を取りこぼしてしまう可能性があるからである．

可視化を行う際に，動的に状態遷移を追えるようにするためにアニメーションを用いる．これによって，学習者はOS の動的な状態遷移に対して，動的なイメージを持つことができる．また，OS の状態遷移をより明確にするために配色に工夫をつけた．具体例としては，タスク管理において実行中のタスクを緑，実行待ちのタスクを黄色，停止中のタスクは赤にすることで，タスクの状態を色という視覚情報で判断できるようにするなどした．

4. 実現

これらの設計を基に作成した試作画面を図２に示す．

本研究は Windows2000 上で Eclipse2.1.2 を用いて Java1.4.2 による開発を行った．

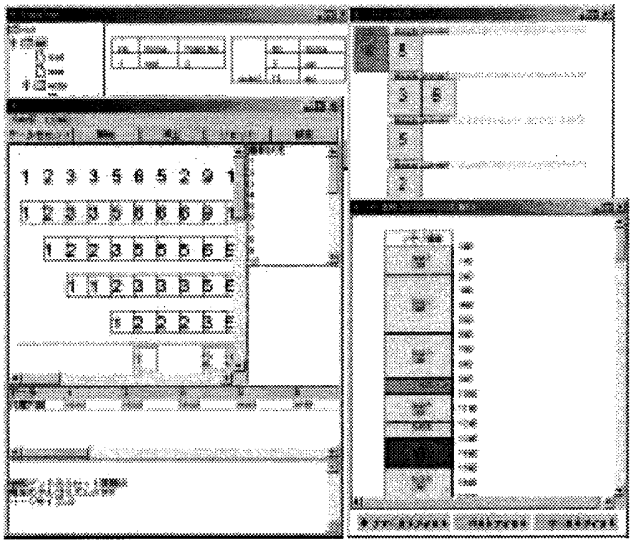

図２　試作画面

５．評価

実際に本大学の OS の講義を受講した学生に利用してもらい，本研究の設計方針についてと試作品の操作性とグラフィックについて評価を行った．

表２　設計方針についての評価

質問内容	YES	NO
1)一つの学習項目に対して，複数の視点から捉えられるほうがいい	80%	20%
2)複数の学習項目を協調して動作させることで，それぞれの項目のつながりを把握できる	60%	40%

5.1 各項目の回答内容

表２における回答では次のようなものが挙げられた．

1)で「YES」と回答した理由

- 特定の視点からでは見えない情報も，別の視点を持たせることで，その情報を見ることができる．
- 実際の動作の流れだけでなく，OS の状態遷移を時間的に捉えて追いかけることができる．

1)で「NO」と回答した理由

- 視点が増えると画面に表示される情報が増えてしまって，自分の注目したいのがどれかわからなくなってしまった．

2)で「YES」と回答した理由

- 複数の学習内容を同時に見ることができるので，OS 広範囲の動作の流れを知ることができる．
- 自分の操作した結果が，全体のどこにどのような影響を与えているかを知ることができるのがいい．

2)で「NO」と回答した理由

- どことどこが連携しているかが，いまいちつかめなかった．
- 画面がごちゃごちゃしてしまって，1 クロック時間を経過させるたびに，どの部分がどの学習項目なのかをいちいち確認しなければならなかった．

5.2　評価に対する考察

質問 1)の結果と回答理由から，ある項目の学習内容を複数の視点から捉えることができるのは，学習者にとって理解を促進させることがわかった．

また質問 2)の結果と回答理由から，学習項目間での協調した動作を行えることは学習者の理解が進む可能性が高いことがわかった．

それぞれの回答において「NO」と回答していた学生は，ウィンドウの数の多さを問題点として挙げている．これは，学習者にとって学習項目を表示するウィンドウの数が多くなったりウィンドウに含まれる情報が多くなると，解釈しなければならない情報が多くなってしまうことが問題点として挙げられていることがわかる．

5.3 操作性・グラフィックに対する意見

学生に，操作性とグラフィックについて次のような意見が挙げられていた．

- それぞれの状態が色分けされているので，状態の変化を直感的に理解しやすい．
- 1 クロックごとに動作させることができたので，自動的に処理が流れるのに比べて動作を理解しやすい．
- 項目ごとにウィンドウが表示されるので，項目を増やしすぎるとウィンドウが邪魔になってしまう．

６．おわりに

本資料では，「港」で用いる OS 可視化コンポーネントの作成について述べた．今後は可視化対象モデルをシミュレータベースのモデルだけでなく，仮想マシンベースのモデルや FPGA をベースとしたモデルまで拡張する．

また，このような学習教材の開発を容易にするために，コンポーネントを作成する環境の開発を行う．

参考文献

[1]西野,早川,石原：”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

[2]吉田,早川：”ロボットプログラミング学習支援環境の開発”FIT2003,N-026 2003

[3]米田,早川：”体験学習モデルによる OS の学習システム” 情報処理学会 コンピュータと教育研究会,2002-CE-67 2003

[4]社団法人情報処理学会：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