マルチメディア通信と分散処理　109－15
電子化知的財産・社会基盤　17－15
（２００２．９．１３）

電子メールを利用したエンドユーザ向けのプログラミング環境

青木 義則　　丸山 宏
日本アイ・ビー・エム（株）東京基礎研究所

概要

インターネット上で提供される様々なサービスへのプログラミングインタフェースをXMLを用いて提供する、Webサービスが注目を集めている。また、インターネット経由で制御できる情報家電も登場しつつあり、インターネットを経由して様々なサービスを利用する環境が整いつつある。これらのサービスの中には、決済などのトランザクションサービスから、株価の通知サービスやエアコンの温度調整サービスなど様々なレベルのサービスが含まれている。今後、これらのサービスを組み合わせて、エンドユーザが自分自身で利用するための簡単なプログラムを手軽に作れる環境が必要になってくると思われる。そこで本稿では、プログラミングの経験がないようなエンドユーザでも、これらのサービスを組み合わせた簡単なプログラムを電子メールを利用して作成・実行するための環境について提案する。

Programming by E-mail: a Programming Environment for Novice Users

Yoshinori Aoki and Hiroshi Maruyama
IBM Research, Tokyo Research Laboratory

Abstract

Web services are a remarkable technology in which the programming interfaces of Web-based services are provided as XML-based interfaces. Internet-ready information appliances are also being developed which allows us to control them via the Internet. Hence, we will soon be able to control various kinds of services through the Internet. Such services include transaction services such as payments, notifications of stock prices, and controlling air conditioners in the users' homes. After such an environment has been created, it will be necessary needed to provide a programming environment with which end users can easily create their own services for themselves by combining such Web-based services. This paper proposes a novel programming environment which allows novice users to easily create programs by e-mail.

1.　はじめに

インターネット上で提供される複数のサービスを連携させる仕組みとして、Webサービス[1]が注目されている。Webサービスでは、サービス間でやり取りするデータはXMLで記述され、SOAP (Simple Object Access Protocol) [2]を用いて交換される。SOAPは、HTTP等のプロトコル上で動作する。具体的なWebサービスの例としては、決済などのトランザクションサービスから、株価の通知サービスなど、多様な応用が考えられる。

また、ネットワークに接続可能な情報家電もいくつか製品化され、インターネット経由でエアコンを制御したり、ビデオの予約を確認したりすることが可能となった。また、情報家電をネットワーク経由で制御するための仕様を標準化するため、Jini [3]、

HAVi [4]、ECHONET [5]など様々な規格が開発されている。

このように、多くのサービスがインターネット上で提供されるようになれば、それらのサービスを組み合わせて、簡単な、しかし便利なプログラムを作成するニーズが出てくると思われる。例えば、「ある特定の人から電子メールが届いたら、携帯電話に転送する」とか「自宅のセキュリティシステムが侵入者を感知したら、（犯人を驚かすために）自宅のCDプレーヤを再生し、携帯電話で知らせてくれる」といった程度の簡単なプログラムでも、容易に作成することができればユーザの利便性は大きく向上するはずである。このように、複数のサービスを連携させて新しいサービスを開発するには、通常はJavaやC++などのプログラミング言語を利用して開発する必要がある。そのため、プログラマでない一般のインターネット・ユーザでは、既存サービスを組み合わせて新しいサービスを構築することは困難であった。

本稿では、電子メールを利用して、プログラマではない一般のインターネット・ユーザでも既存のサービスを組み合わせて簡単なプログラムを作成、実行できるような環境を提案する。

以降では、我々が提案する仕組みのコンセプトを説明する。さらに、試作システムの概要について説明し、今後の課題について議論する。

2. 電子メールによるプログラミング環境

ここでは、本論文で提案するプログラミング環境の基本的なコンセプトについて説明する。

2.1. 対象とするユーザ層

ここで、本稿が対象とするユーザをより具体的に示すために、図１にインターネット・ユーザをいくつかのグループに分けて示す。図中の上位三層は、いわゆるプログラマと呼ばれるユーザ層で、下位二層はプログラムを書くことはできないが、電子メールを利用したり、簡単なWebページを記述することはできるユーザ層である。本稿で対象とするユーザ層は、図中の下位二層のノン・プログラマ層であり、以降ではこれらの層のユーザをエンド・ユーザと呼ぶことにする。

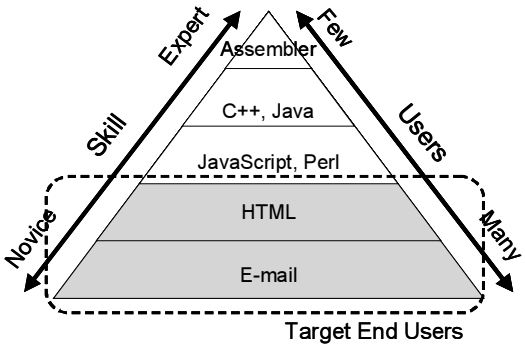

図1 対象となるユーザ層

2.2. 基本コンセプト

我々は、前節で説明したユーザ層に対して簡単なプログラムを手軽に作れるような環境を提供するためには、どのような仕組みが必要かを検討した。その結果、以下の２つのコンセプトに基づいてシステムを設計することにした。

- HTMLによるプログラムの記述
- プログラムの公開と、再利用によるプログラミング

以降では、上記２つのコンセプトについてより詳細に説明する。

2.2.1. HTMLによるプログラムの記述

通常のプログラミングでは、テキストエディタやIDE (Integrated Development Environment) を利用してソースコードを直接記述するが、エンドユーザにプログラムを直接記述してもらうのは不可能である。そのため、アイコンを並べてプログラムを記述するIconic Programmingのような視覚的プログラミング[6]の要素が必要となる。

そこで、我々はHTMLでプログラムを記述することを提案する。HTMLは、表現能力が高く、画像を配置したり、表や箇条書きなど様々な視覚表現が可能となる。そこで、HTMLの視覚表現にプログラムとしての意味を持たせることにより、Webページを作るのと同様の感覚で、視覚的プログラミングを行うことが可能となる。

このアイディアの最大の利点は、エンドユーザにとって敷居が低い点にある。ほとんどのインターネット・ユーザは電子メールを日常的に利用しており、最近の電子メール・ソフト（メーラー）の多くは標準でHTML形式のメール[7]作成をサポートしている。そのため、ユーザはプログラミングをするために新たなソフトをインストールしたり使い方を覚えたりする必要がなく、これまで利用してきたメーラーを使ってプログラムを作成することが可能になる。また、HTMLで記述されたプログラムを解釈・実行するサーバ（プログラム・サーバ）をSMTP/POP対応にしておくことで、作成したプログラムをプログラム・サーバに送信するだけで実行することが可能となる。

プログラム・ページとプログラム・アイコン

HTMLで記述された１枚のWebページ（またはHTML形式で記述された１通の電子メール）を、１つのプログラムとして扱い、そのようなHTML文書をプログラム・ページと呼ぶことにする。

また、プログラムを視覚的に記述するために、プログラム・アイコンと呼ぶ画像ファイルを利用することにする。プログラム・アイコンは１つの機能を持ったプログラムを表現しており、プログラム・ページ上にプログラム・アイコンを並べることによってプログラミングを行うものとする。プログラム・アイコンをページ上にどのように配置するかでプログラムの実行手順を制御するといったことが可能となるが、ここではその詳細については議論しない。

プログラム・アイコンとプログラム・ページ、および実際のプログラムとの関係を、図２に示す。図２に示すように、プログラム・アイコンは以下のいずれかのオブジェクトを１つ以上含んでいる。

- プログラムそのもの
- プログラムへのリンク
- プログラム・ページへのリンク

なお、ここでいうプログラムとは、テキスト形式のスクリプト・プログラムかもしれないし、Javaのクラスや実行形式のプログラムかもしれない。また、画像が上記オブジェクトを含むための手法としては、電子透かし[8]のような形で含んでもよいし、単

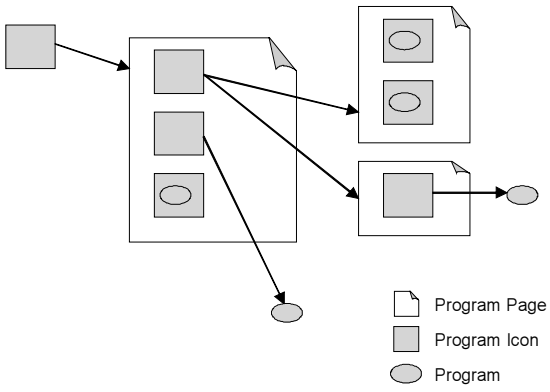

図2 プログラム・ページとプログラム・アイコン

純に画像データのコメント領域やメタデータ領域に埋め込んでもよい。

2.2.2. プログラムの公開と再利用

ここでは２つ目のコンセプトについて説明する。

検索によるプログラミング

本システムでは、プログラムをHTMLで記述する。そのため、自分が作成したプログラムをWebサイト上に公開したり、電子メールで送信したりすることが可能である（この場合、Webブラウザやメーラーでプログラム・ページを表示することは可能だが、実行することはできない）。

そこで、各ユーザが積極的に自作プログラム・ページを公開することにより、ユーザ同士がお互いのプログラムを参考にしたり、あるいはそのまま再利用することが可能となる。プログラム・ページ上にコメントとしてプログラムの内容を記述しておけば、Webブラウザやメーラーでプログラムの説明を読むことができるだけでなく、既存の検索エンジンのキーワード検索を利用してインターネット上から所望のプログラムを探すことができるようになる。もちろん、HTMLの<META>タグなどを利用して（あるいは独自のタグセットを定義して）プログラムの内容を記述することが可能となれば、より精度の高い専用のプログラム検索エンジンを構築することも可能となる。重要なのは、プログラミングを始める前に、既存プログラムをインターネットで検索し、そ

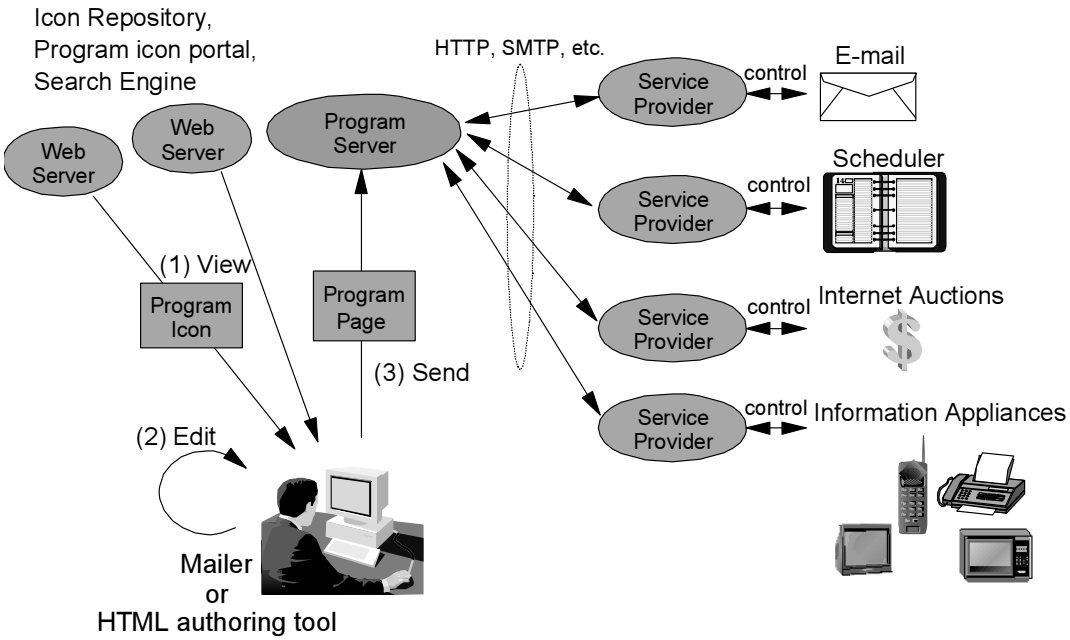

図3 提案システム概要

れらを再利用することでプログラミングの作業を軽減するような環境を構築することである。

スキル・レベルに応じたプログラミング

このコンセプトは、ユーザ同士が各自のスキル・レベルに応じたプログラミングを行うことを想定している。例えば、通常のプログラミング言語を熟知したプログラマは、プログラム・アイコンの中に埋め込まれるプログラムそのものを開発することになると思われる。一方、プログラムそのものは記述できないが、コンピュータには詳しいヘビー・ユーザや論理的思考力の高いユーザであれば、既存プログラム・ページやプログラム・アイコンをインターネット上から探し出し、それらを自由に組み合わせて独自のプログラム・ページを作成することが可能となるであろう。また、より簡単なプログラミング・スタイルとして、既存プログラムのパラメータのみを変更して再利用する手法や、既存プログラムをそのままコピーしてきて再利用するというスタイルもあるであろう。

2.3. 提案システムの概要

図３に、提案システムの構成を示す。図３に示すように、サービス・プロバイダは各サービスへのインタフェースを提供し、それらのインタフェースへはHTTPやSMTPなどのプロトコルを利用してインターネット経由でアクセスすることができる。

ユーザはプログラミングをはじめる前に、検索エンジンやプログラム・ページのポータルサイトへアクセスし、既存プログラムで再利用できそうなものを探す。また、必要であれば、プログラム・アイコンを提供しているWebサイトにWebブラウザでアクセスし、パラメータを変更した新しいプログラム・アイコンを生成してもらう。次に、メーラーで新規メッセージを作成し、探してきたプログラム・アイコンをコピー&ペーストで貼り付け、プログラムのフローを定義する（メーラーのかわりにHTMLオーサリング・ツールを利用してもよい）。次に、作成したメッセージ（プログラム・ページ）の送信先にプログラム・サーバのメール・アドレスを記入し、送信する。プログラム・サーバは、受け取ったメッセ

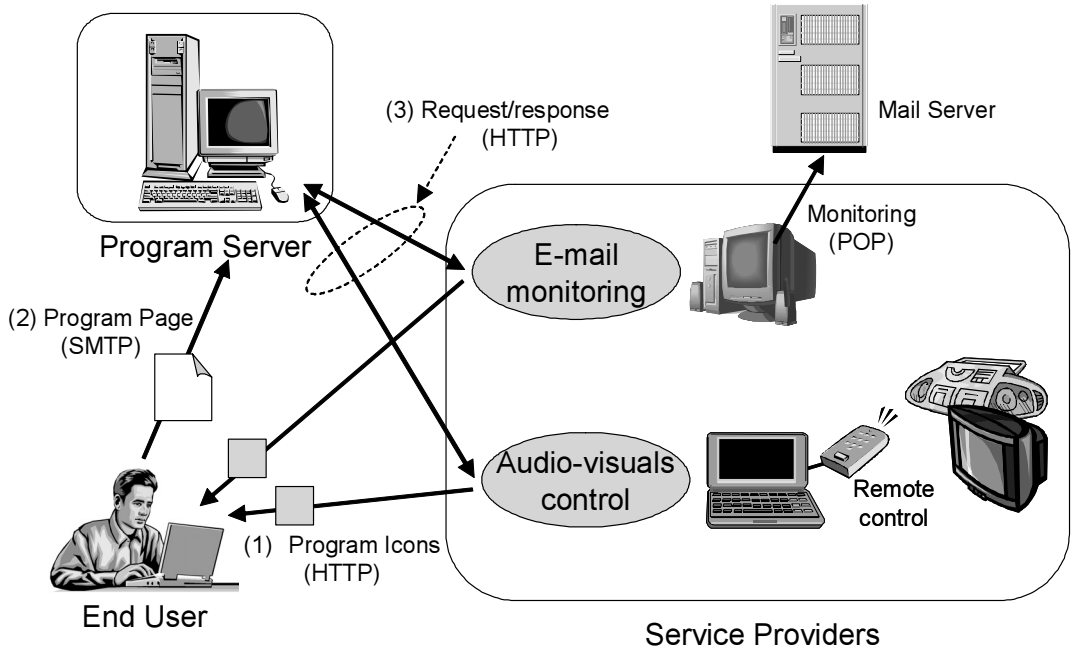

図4 試作システム

ージを解析し、そこに記述されたプログラムを実行する。プログラムを実行する際には、プログラムで利用されているサービスを提供しているサービス・プロバイダと連携しながらプログラムを実行する。

3. 試作システム

前章で示したコンセプトの有用性を確認するとともに、実現する際の課題を洗い出すために、簡単な試作システムを作成した。試作システムの概要を図４に示す。

試作システムでは、以下の２つのサービス・プロバイダを実装した。１つは電子メールのモニター・サービスで、メール・ボックスに到着した電子メールを監視し、（1）事前に登録したユーザから電子メールが届いたら通知する、（2）事前に登録したキーワードが件名か本文に含まれていたら通知する、といったサービスを提供する。２つ目はオーディオ制御サービスで、CDプレーヤやテレビといたオーディオ機器を制御するサービスである。サービス・プロバイダはHTTPでサービスの要求を受け付け、処理結果をHTTPで返す。電子メールのモニター・サービスは、処理結果として検出されたメールの本文を返し、オーディオ制御サービスでは制御結果を返す。オーディオ制御サービス・プロバイダはJavaアプリケーションとして実装されており、実際のオーディオの制御はPCにシリアル接続した学習リモコンから各種オーディオ機器に制御信号を送ることで実現している（これは、HTTPで直接制御できるオーディオ機器がまだ市販されていないため、PCに接続したリモコンでHTTPで制御可能な情報家電をシミュレートしている）。また、電子メールのサービス・プロバイダはPerlで実装した。電子メールのモニター・サービスを利用する場合、パラメータ（例えばモニターしたい電子メール・アドレスなど）を変更して、自分専用のプログラム・アイコンを作成する必要があるため、パラメータを受け取って、それらを埋め込んだプログラム・アイコンを生成するプログラムをJavaサーブレットで実装した。ユーザはWebフォームに必要なパラメータを入力し、「アイコン生成ボタン」をクリックすることで、それらのパラメータが埋め込まれたプログラム・アイコンをWebブラウザに表示することができる。

図5 試作システムによるプログラム例

プログラム・サーバは、電子メールとして受け取ったメッセージを解析し、HTML文書の部分（プログラム・ページ）とそこに埋め込まれた画像ファイル（プログラム・アイコン）をMIMEエンコードされたメッセージから取り出し、プログラムを解釈・実行する。プログラムサーバはJavaアプリケーションとして実装した。

試作システムでは、プログラム・サーバはプログラム・ページ上に並べられたプログラム・アイコンを順番に実行する。メール本文（またはWebページの本文）にテキストとして記述されたメッセージはコメント文として無視する。また、各プログラム・アイコンにはPerlで記述されたプログラムが画像フォーマットのコメント領域に埋め込んであり、プログラム・サーバはプログラム・アイコンから取り出したPerlプログラムを実行する。

図５に試作システムで実際に作成したプログラムの例を示す。図５の例では、「事前に登録したユーザからメールが届いたら、CDプレーヤを再生して音楽で通知する」というプログラムを表現している。メール中に書かれている文章はプログラムの内容を説明するコメント文で、プログラムサーバからは無視される。また、メール本文に貼り付けてある２つの画像はプログラム・アイコンで、Perlで書かれたプログラムが実際に埋め込まれている。

4. 議論

試作システムの設計、および実装を通して、様々な課題が明らかになった。ここでは、主な課題について議論する。

4.1. セキュリティ

提案システムの最も重要な課題は、セキュリティをどうやって確保するかという点である。例えば、悪意のあるプログラム・ページによって勝手に自宅のエアコンやお風呂のスイッチを入れられては危険である。また、ウィルスを含むようなプログラム・ページを如何にして判定するのかといった点も問題となる。また、プログラム・サーバは電子メールでプログラム・ページを受け付けるため、悪意のある第三者が本来利用権限のないプログラム・サーバに他のユーザになりすまして電子メールで送りつけるかもしれない。そのため、ユーザ認証の仕組みや、プログラムの内容を証明するような仕組みが必要となる。ただし、エンド・ユーザを対象としているため、それらのセキュリティの仕組みでユーザの手順が複雑になり過ぎないように注意しなくてはならない。

また、プログラム・アイコンによってはパラメータに個人情報を埋め込む必要がある場合もある。例えば、試作した電子メール・モニター・サービスで利用するプログラム・アイコンに埋め込まれたPerlプログラムは、ユーザのメール・ボックスを監視するために、ユーザのメール・アドレス、メール・サーバのアカウントとパスワードといった情報を含んでいる。このように、プログラムのパラメータとして個人情報が含まれるような場合は、プログラムを暗号化するなどして埋め込む必要がある。

4.2. 言語モデル

試作システムは、視覚的プログラミングの形態をとっており、その背後にある言語モデルは非常に単純なものである。プログラム・ページに埋め込まれたプログラム・アイコンは順番に実行され、各プログ

ラム・アイコンは必ず何らかのアクションを実行し、その結果をイベントとして次のプログラム・アイコンに渡す、というモデルを採用している。このとき、イベントはテキスト・メッセージとして次のアイコンに渡される。渡されるイベント・メッセージをどのように処理するかは次のプログラム・アイコンのアクション部分に依存するが、単純に表示する、読み上げる、転送する、無視するなどの処理が考えられる。ただし、イベント・メッセージの内容を解析し、その内容に応じて次のアイコンのアクション部分の処理を選択するような利用の仕方をしてはならない。これは、直前のアイコンから特定のイベント・メッセージが送られてくることを期待しているようなプログラム・アイコンが存在すると、アイコン間の接続に制約ができてしまう。そのような制約は、通常のプログラミング経験のないようなエンド・ユーザを困惑させるかもしれない。そのため、試作システムではイベント・メッセージをユーザに対するメッセージ伝達にのみ利用するようなモデルにすることで、任意のアイコンをどのような順番で並べてもプログラムが動くようにした。

現状の言語モデルは、プログラム・アイコンを順番に実行し、アクション実行とイベント発行を繰り返すような単純なモデルである。そのため、エンドユーザにとってはプログラム・アイコンを並べていくだけなので使いやすい反面、あまり複雑なことはできない。今後は、実ユーザにプログラムを作ってもらう実験などをすることで、複雑性を排除しつつ、どの程度まで高機能にするべきかを探っていき、より洗練された言語モデルを構築する必要がある。

4.3. プログラムのメタ情報

公開されているプログラム・アイコンやプログラム・ページを高い精度で検索するためには、プログラム・ページやプログラム・アイコンにメタ情報を埋め込むことが必要となる。

そのためには、検索のためにはどのような情報をどのような形態で埋め込むべきか検討する必要がある。また、悪意のあるコードの流通を防ぐためにも、埋め込まれたメタ情報やプログラムそのものが正しい内容であることを証明する必要がある。そのためには、プログラムの内容を検証し、証明するような仕組みが必要となるであろう。もしくは、インターネット・オークションのユーザ・ランキング等のように、正しいプログラムを多く公開してきたユーザを他のユーザが格付けしていくような仕組みになるかもしれない。

また、試作システムでは画像ファイルのコメント領域にプログラムを埋め込んでいる。通常のコピー&ペーストで電子メールのメッセージにプログラム・アイコンを貼り付ける場合は、プログラムやメタ情報も画像に埋め込まれたままメールに貼り付けられる。しかし、いったんイメージ編集ソフトで画質調整などの処理を施すと、埋め込まれたプログラムやメタ情報が消失する場合がある。そのため、画像ファイルとプログラムやメタ情報の関係を、埋め込み以外の方法で実現する必要があるかもしれない。

5. おわりに

本稿では、インターネット上で公開された様々なサービスを連携させることで、簡単でしかも便利なプログラムを手軽に作ることができるような環境を提案した。この仕組みは、（1）HTMLでプログラムを記述する、（2）既存プログラムを再利用してプログラミングする、という２つの基本コンセプトに基づいている。エンドユーザは、通常使っているWebブラウザでインターネット上からプログラムを検索し、既存プログラムをそのまま再利用したり、組み合わせたり、パラメータを変更することでプログラムを作成することができる。また、プログラムの作成は、普段利用している電子メールのソフトを利用して行うため、ユーザにとって敷居が低い。

これらのコンセプトの有用性を確認するために、試作システムを設計し、実装した。その結果、手軽にプログラムが作れるという点では非常に使い勝手がよいものの、実用上は様々な課題があることが明らかとなった。今後は、これらの課題を解決するための仕組みを検討する必要がある。

謝辞

試作システムの実装に協力していただいた、学生研究員の山中晋爾氏とEthel Chen氏に感謝いたします。

参考文献

[1] Graham, S. et al., *Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI*, Sams, December 2001.

[2] Box, D. et al., "Simple Object Access Protocol (SOAP) 1.1," W3C Note, May 2000. Available at http://www.w3.org/TR/SOAP/.

[3] Arnold, K. (Ed.), *The Jini™ Specifications 2nd Edition*, Addison-Wesley, December 2000.

[4] HAViホームページ, http://www.havi.org/.

[5] ECHONETコンソーシアム・ホームページ, http://www.echonet.gr.jp/.

[6] Shu, N.C., *Visual Programming*, Van Nostrand Rheinhold, NY, 1988.

[7] Palme, J., Hopmann, A., and Shelness, N., "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)," RFC 2557, March 1997.

[8] Bender, W., Gruhl, D., Morimoto, N., and Lu, A., "Techniques for data hiding," *IBM Systems Journal*, Vol. 35, No. 3&4, 1996.