Java によるドキュメント作成ツール

Java Technonogy in Java Application“ Data Sheet ”

櫻 井 優 子, 熊 本 厚 志

要 約 本稿では，Java で実装したアプリケーション「データシート」でいかに Java 技術を用いたかを報告する．

「データシート」と呼ぶこのアプリケーションは，ドキュメントを作成することを目的としたものである．「データシート」の大きな特徴は，ドキュメントを作成するための部品を追加することができることである．現在，「データシート」で作成できるドキュメントは，文字列とグラフィックスイメージから成るもの（文字列とグラフィックス用の部品がある）であるが，部品はアプリケーションと独立して開発でき，今後必要に応じて部品を増やすことができる．

「データシート」には，いくつかの Java 技術が使用されており，本稿ではアプリケーション開発の紹介を通して，それらの技術を報告している．「データシート」における主な技術は，JavaBeans・イベント・MVC・シリアライズである．

Abstract This paper describes how Java technology was used in an application “ data sheet ” implemented with Java.

This application called “ data sheet ” aims to prepare various types of documents. A major characteristic of “ data seat ” to be able to add parts to make a document. The document which can be made on ” data sheet ” is the availability to add parts to develop a document.

The document that can be made of data sheet is objects composed of character strings and graphics images (there are parts for both character strings and graphics), but parts can be developed independently of an application and the number of parts can be increased in future as required.

“ Data seat ” uses some of Java technology and this paper describes them through introducting the application development.

Major technology used in data seat application is JavaBeans/event/MVC/Serialize.

1. は じ め に

1995 年の発表以来，Java への関心は衰えることなく現在に至る．本稿で紹介する「データシート」開発は，1997 年度の日本ユニシスにおける「Java プロジェクト」での実証システム開発の一環として行われたものである．

本稿では，「データシート」開発を事例とし，その中で活用した Java の技術を報告する．2 章で開発したアプリケーションの要件を挙げ，3 章でその開発事例を報告する．4 章でこの開発において用いた Java 技術について報告する．本開発で用いた主な技術は，JavaBeans，イベント，MVC，シリアライズである．

2. 課　　題

2.1 アプリケーション要件と課題

あるシステムに「ドキュメントを作る」という機能が必要であり，その機能を汎用化したアプリケーション・システムを開発することとなった．ドキュメントを作ると一口にいっても，報告書や論文など，その内容は様々である．ドキュメントをどのようにとらえるかで，このアプリケーションの全貌が決まる．

「ドキュメントを作る」機能を持つアプリケーションは数多く存在する．しかし，いずれも作るドキュメントをあらかじめ想定したものである．例えば，テキストベースのドキュメントを作るもの，図の描画を得意とするもの，などである．しかし，使用者の立場から言えば,複数のアプリケーションを目的に合わせて使い分けるよりも，一つのアプリケーションで自分の思い通りのドキュメントを作成できる方が望ましい．

本稿で紹介するアプリケーションは,「作りたいドキュメントを作る」ことができる，ということを課題とした汎用的なものである．特定のドキュメントを作るためではなく，ニーズに合ったドキュメントを作るための汎用的なアプリケーションを目指した．

2.2 課題の検討

「ドキュメントを作る」とは，どういうことか．このために必要な機能には次の二つがある．

1)　ドキュメントの内容を記述（修正）する
2)　ドキュメントを保存する

ここでいうドキュメントとはテキストや図や表などから成るものである．そして，テキストや図や表はドキュメントを構成する部品ととらえることができる（以降，このような部品をドキュメント構成部品と称する）．

では，使用者の「作りたいドキュメント」とはどのようなものか．ある場合は，報告書のようなテキストが主体となるものであり，ある場合はオブジェクト関連図のような図を主体とするものとなる．使用者の「作りたいドキュメント」には様々な種類が考えられる．ドキュメント構成部品に関しても，テキストや図や表のようにあらかじめ想定できるものだけでは充分でないかもしれない．ドキュメント作成者のニーズによって，その都度必要となるドキュメント構成部品が異なる可能性もある．このことに対応するには,「ドキュメントを作る」アプリケーションに「必要なドキュメント構成部品を必要になったときに追加変更できる」ことが必要である．

2.2.1 ドキュメントの内容を記述（修正）する

ドキュメントは複数のドキュメント構成部品から成る，と考えるとドキュメントを記述することには二つのフェーズがある.一つはドキュメント構成部品の内容の記述,例えば文字の入力などである．もう一つはドキュメント構成部品をどのようにレイアウトしてドキュメントを作るかである．ドキュメント構成部品がどのような内容を記述(修正)できるかは，それがどのような操作や表現を提供するかに依存する．一方，ドキュメント構成部品のレイアウトは，その種類に関わらずドキュメントを作るうえで必要な動作であり，これはアプリケーション側で決定する．

このアプリケーションでは，いつでも「ドキュメント構成部品の追加」を行いたい．しかし，その都度アプリケーションを変更しなくてはならないようであれば，使用者が容易に「ドキュメント構成部品の追加」を行うことはできない．また，追加するドキュメント構成部品をアプリケーション開発者しか作成できないとすれば，それも使用者による「ドキュメント構成部品の追加」を妨げる．ドキュメント構成部品は，開発者が任意に開発できなければならない．とはいえ，何らかの枠組みは必要である．アプリケーションは個々のドキュメント構成部品の特性に関わらず，「ドキュメントを構成する部品」としてとらえる必要がある．「ドキュメントを構成する部品」はドキュメント全体のレイアウトに関わる操作や情報を持つ．例えば，ドキュメントのレイアウト変更操作（ドキュメント構成部品の移動やリサイズ），フォントやドキュメント構成部品の配置位置やサイズである．

2.2.2 ドキュメントを保存する

作成されたドキュメントは，その作成時にだけ存在するのではなく，何らかの形で保存できなければならない．保存はドキュメントをアプリケーション終了後も何らかの形で継続維持することを指し，当然，後に保存したドキュメントの参照や修正を考えてのことである．具体的には，ドキュメントの表示状態（サイズ，色，など），ドキュメントの内容（ドキュメント構成部品）の状態など各種の情報の保存が必要である．

ここで問題となるのは，ドキュメントの情報をどのような形式で保存するかである．保存時にはドキュメントの表示に必要な情報は全て保存しなくてはならず，読み込む時にはそれらの情報を読み取りドキュメントを表示する必要がある．保存情報の詳細を考えるまでもなく，それらが複数の情報から成り，複雑に絡み合ったものであると想像できる．この情報を書き込んだり読み込んだりするために何らかの形式を定義する必要がある．

3. データシート概要

「データシート」は，スタンドアロンで動作するエンドユーザ向けの汎用的なドキュメント作成ツールである．ドキュメントを作るには，白紙のページにドキュメントを作るために用意するドキュメント構成部品を貼り付けていく．ドキュメント構成部品には，例えばテキストを入力/表示することのできるようなものが考えられる．

図１は「データシート」起動後，新規ドキュメントを開いた状態である．メインメニューは「データシート」の操作を行うためのウィンドウで，各ボタンに対応してポップアップメニューを表示する．部品パレットはドキュメント構成部品の一覧である．ページはドキュメントの１ページに相当するウィンドウである．

3.1 機　　能

「データシート」の主な機能は以下の通りである．

1） シートの保存と読み込み

画面上で作ったシートはファイルに保存可能である．保存したシートは，再度「データシート」に読み込み編集することが可能である．

2） ドキュメントの作成

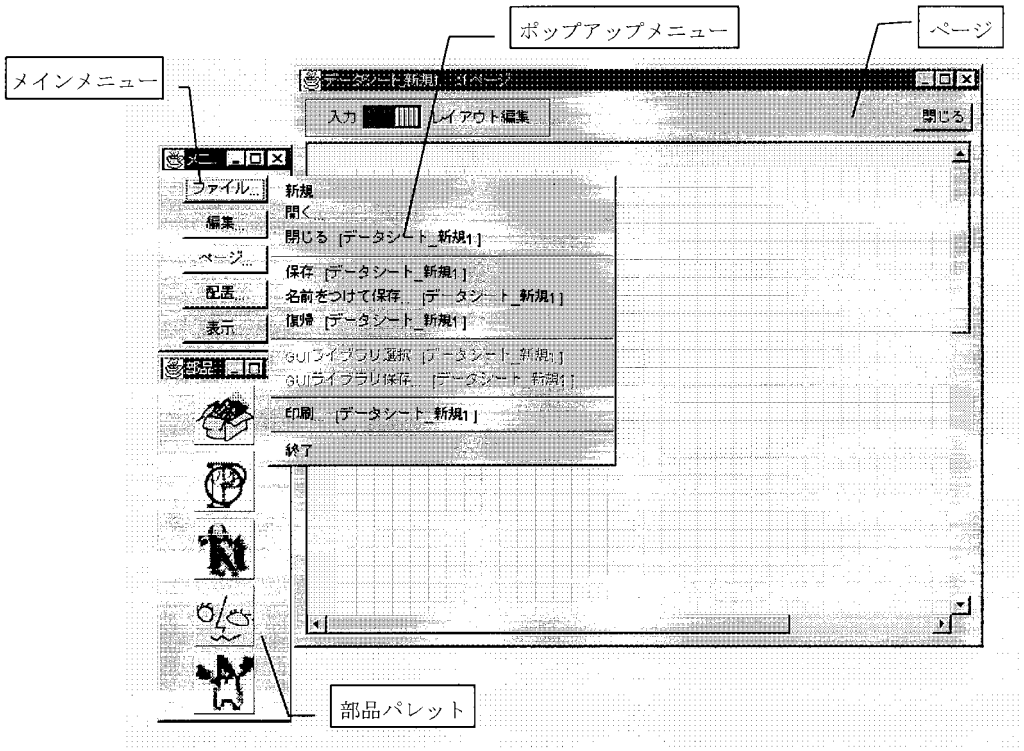

図 1　データシート

①　ドキュメント構成部品の配置

部品パレットからドキュメント構成部品を選択し，ページの任意の場所に配置できる．ドキュメント構成部品としては，レイアウトボックス・シート情報部品・テキストボックス・グラフィックスボックスの四つを提供している（各部品の詳細は 3.2 節参照）．

図 2 はある製品に対する画像イメージと，それに対するコメントから成るサンプルシートである．画像イメージにはグラフィックスボックス，コメントにはテキストボックスを使用している．そして，シート情報部品によってドキュメント作成日付を表示している．

②　ドキュメント構成部品の移動とリサイズ

ドキュメント構成部品はシート内で移動とリサイズが可能である．ドキュメント構成部品を移動するには，部品をドラッグし任意の位置にドロップする．リサイズは，選択したドキュメント構成部品上のリサイズハンドルをドラッグすることで行う．

3)　シートの印刷

全ページあるいは指定ページの印刷ができる．

4)　プロパティ設定

シート上のドキュメント構成部品のプロパティ（例えば，フォント情報など）を変更することができる．シートはデフォルトプロパティを持ち，各ドキュメン

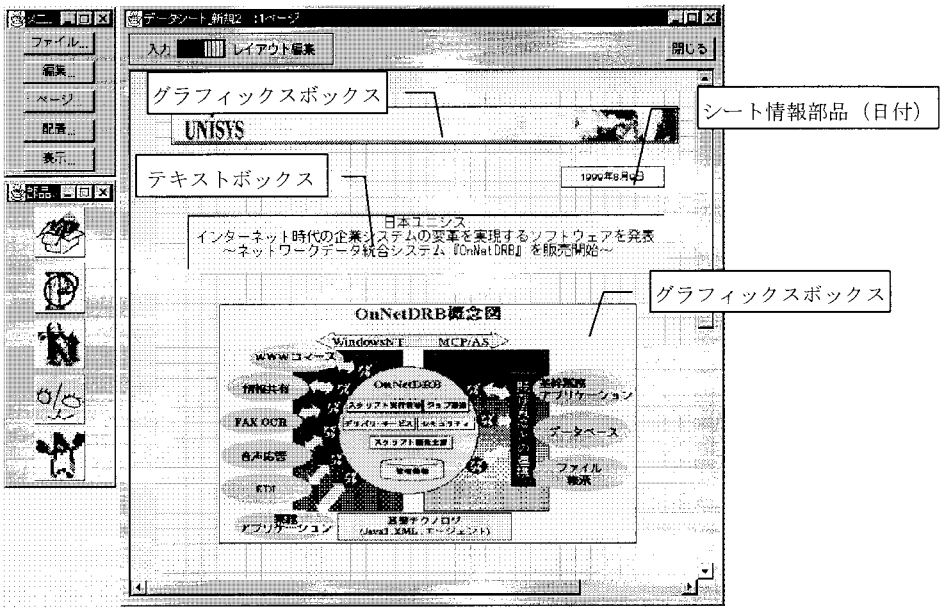

図 2　サンプルシート

ト構成部品のプロパティはデフォルトプロパティにあわせることも個別に設定することもできる．

3.2　ドキュメント構成部品

現在，「データシート」では，表 1 のドキュメント構成部品を用いてドキュメントを作ることができる．

表 1　ドキュメント構成部品

ドキュメント構成部品	概要
レイアウトボックス	他のドキュメント構成部品（レイアウトボックスを含む）を配置するための入れ物の役割を果たす。
テキストボックス	複数行の任意の文字列を表示／入力することができる。
グラフィックスボックス	シート上にグラフィックスイメージを表示する（使用可能な画像ファイル形式は GIF あるいは JPEG）。
シート情報部品	日付（シート保存日時）あるいはページ番号をシート上に表示する。

この四つ以外にも，任意の Bean を追加してドキュメント構成部品とすることができる．部品パレット（図 1）上の五つ目のボタンは，任意の Bean を追加できることのサンプルである．このボタンに対応している Bean は，BDK[*1] 付属のサンプル Bean（Juggler[*2]）である．

3.3　クラス構成

図 3 は，「データシート」のクラス構成を示したものである．

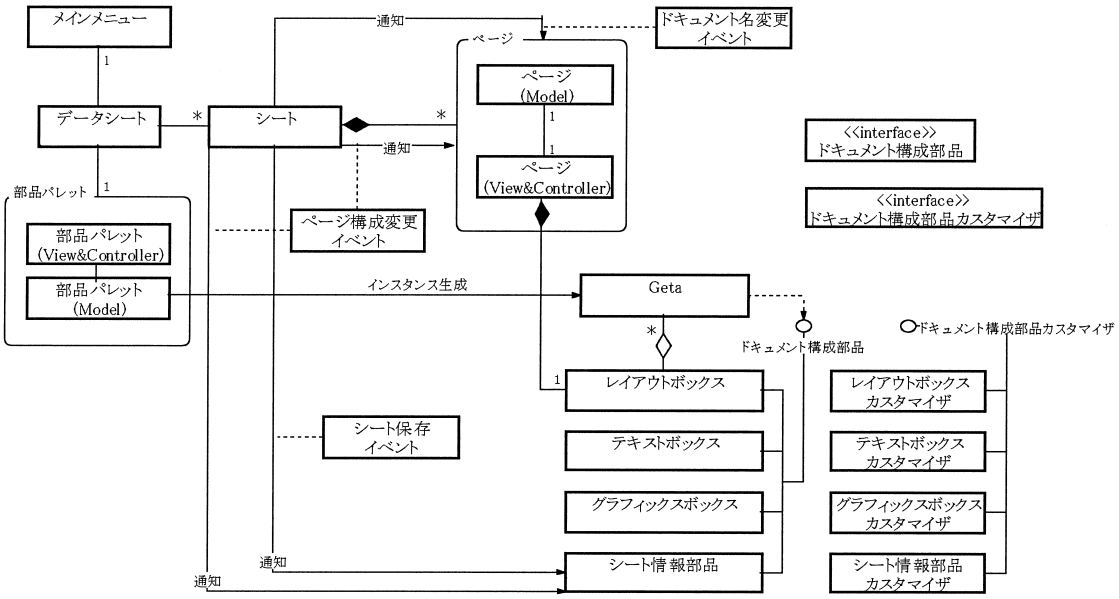

図 3　クラス構成

データシートは複数のドキュメントを管理でき，各ドキュメントの操作はメインメニューに用意する．シートクラスはドキュメントをオブジェクトとしてとらえたものである．シートは一つ以上のページからなり，ページはModelとView & Comtrollerのオブジェクトからなる．ページはレイアウトボックスから構成されており，ページ上には任意のドキュメント構成部品を配置できる．配置したドキュメント構成部品は，内部的に「Getaクラスのインスタンス＋ドキュメント構成部品」の1対になっている．Getaはドキュメント構成部品とシート間のアダプタ的役割を果たしている．ドキュメント構成部品の管理は，部品パレットが担う．部品パレットは，GUI部品とデータ部品に分かれており，データ部品は，ドキュメント構成部品のクラス管理とインスタンス生成を担う．GUI部品は，部品パレットが管理するドキュメント構成部品の一覧を表現する．

4.　データシート開発における Java 技術

4.1　Bean によるドキュメント構成部品の実現

「データシート」では，JavaBeansをサポートすることで「ドキュメント構成部品の追加機能」というツールの機能を実現している．JavaBeansはJava開発者にとって共通の仕様であるため，「データシート」実装の詳細を意識することなくドキュメント構成部品を開発できる．Beanとしてドキュメント構成部品を作成すれば，「データシート」に必要な以下の要件を満たすことになる．

1)　「データシート」は，アプリケーション起動時に読み込むアーカイブに存在するBeanをドキュメント構成部品と認識する．そして，その詳細に関わらず，ドキュメント作成者の指示によってインスタンスを作りシート上に配置することになる．アプリケーションからは各ドキュメント構成部品クラスの持つメソッド(コ

ンストラクタも）に関して考慮することができないので，インスタンス生成に関して条件（コンストラクタの引数）が必要ないクラスでなければならない．

2） ドキュメントの保存/読み込みにはシリアライズを利用している．保存ではドキュメント全体をファイルに保存し，読み込みではファイルに保存した情報からドキュメントを再現している．ドキュメント（シートオブジェクト）を保存すると，ドキュメント上の部品オブジェクトも一緒にシリアライズされるので，ドキュメント構成部品はシリアライズが可能なクラスでなければならない．

また，「データシート」の実装において，部品パレット上のボタンにグラフィックスを使用したり，部品プロパティ変更用のカスタマイザ[*3]を使用している．これらはJavaBeansによって定義されるBeanの構成を利用している．具体的には各BeanにはBeanInfo[*4]という構成規則を定義するクラスを用意することができ，このBeanInfoからBeanを表すアイコンやカスタマイザを知ることができる．

4.2 Beanと「データシート」の融合

各ドキュメント構成部品は各々ドキュメントの表現の一部を担う一方，ドキュメント作成のための操作ができる必要がある．具体的にはドキュメントの内容をレイアウトする操作，つまりドキュメント構成部品の配置やサイズの自由な変更を指す．「データシート」にとってドキュメント作成の操作性は重要なポイントである．マウスによってシート上にドキュメント構成部品を配置し，それらを自由にレイアウトできる必要がある．この為には，ドキュメント構成部品に発生するマウスイベントを処理する必要がある．

シート上のドキュメント構成部品に発生するイベントは，2種類に分けることができる．一つはドキュメント構成部品（Bean）自身が処理するイベントで，例えばテキストボックスへのキー入力などである．もう1つはドキュメント構成部品の移動やリサイズという「データシート」が処理しなければならないイベントである．前者の場合，どのようなイベントをどう処理するかはドキュメント構成部品任意の仕様である．後者のイベント処理はGetaと名づけた部品のアダプタクラスで行っている．

シートに配置された部品は図4のような構造になっている．

①と③がGetaの部分である．①はダミーのドキュメント構成部品で，②はその本体（Bean）である．③は①と②を内部的に一つにまとめるための入れ物になってい

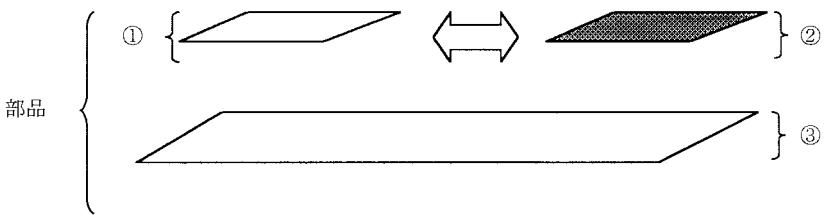

図4 シート上の部品構造

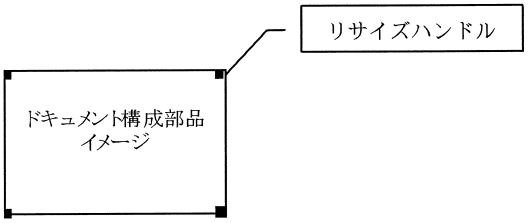

図 5　ダミーの部品

る．ダミーの部品上には，対応する本体のドキュメント構成部品と同じイメージ（ハードコピー）を表示し，4 隅にはリサイズハンドルとなる黒い四角を表示する（図 5）．ダミーのリサイズハンドル部分ではリサイズのためのマウスイベントを受け，それ以外の部分では移動のためのマウスイベントを受けている．これによってドキュメント構成部品は移動/リサイズを行う．

「データシート」は，必要に応じてドキュメント構成部品のダミーと本体の表示/非表示を切り替える．シート上では通常，ダミーのドキュメント構成部品を表示している．ダミーをクリックすると選択状態になり，リサイズハンドルを表示する．その状態でさらに該部品をクリックするとドキュメント構成部品本体を表示する．つまり，選択状態ではダミーがレイアウト処理を行ない，選択状態の部品をクリックすることをきっかけに本体が部品固有の処理を行う．しかし，ダミー上には本体のドキュメント構成部品と同じイメージが描画されているので，アプリケーション使用者にダミーと本体の切り替えを意識させることはない．

4.3　Bean の拡張

「データシート」がドキュメント構成部品として取り込むことが可能な Bean には，以下の 2 種類がある．

① 単純な Bean

② 「データシート」が定義するインタフェース[*5] を実装した Bean

ドキュメント構成部品の提供者は，最低限 “ Bean ” を作成すればドキュメント構成部品として「データシート」に追加できる．しかし，「データシート」の提供するインタフェースを実装することによって，作成するドキュメント構成部品に，ユーザにとって，便利な機能を追加できる．具体的には，“ プロパティの一括更新 ”，“ ショートカットメニューへの項目追加 ” がある．「データシート」は Bean を取り込む際に，クラス定義を検査し，該インタフェースが実装されていれば，そのメソッドを用いる．Java では実行時にクラスの定義を検査することができるのでこのようなことが可能である．

インタフェースを実装するドキュメント構成部品は以下のように動作する．

- カスタマイザからのプロパティ変更タイミングは，各プロパティの値を入力した時と全プロパティを入力し終わった時の二つが考えられる．しかし JavaBeans には，プロパティを一括更新するための仕組みは用意されていない．これを補うために「データシート」提供インタフェースには，プロパティの一括変更メソッドを用意した．このインタフェースを実装しているドキュメント構

成部品に対しては，プロパティの設定を確定するタイミング（OK ボタンクリック時）でこのメソッドを呼び出しプロパティを変更する．

- JavaBeans にはショートカットに関する定義はないが，「データシート」ではプロパティ変更のためにショートカットを使用したい．このため，インタフェースにドキュメント構成部品のショートカットメニュー項目を取得するメソッドを用意した．「データシート」は，ドキュメント構成部品をマウス右ボタンでクリックすると，その部品に対するショートカットメニューを表示する．この時，単純な Bean であればデフォルトのメニュー項目を表示するが，インタフェースを実装した Bean の場合は，デフォルトの項目に加えて任意のメニュー項目を表示する．

4.4 MVC

MVC とはオブジェクト指向を GUI 構築に適用する為の指針である．MVC において GUI は Model・View・Controller の三つの役割を分担するオブジェクトで構成される．各オブジェクトは表 2 のような役割を担っている．

表 2　MVC におけるオブジェクト

オブジェクト	役割
Model	ウィンドウに表示されるべき内容を表現するオブジェクト。 アプリケーションの機能部分を実行するデータを保持し、ユーザインタフェースに依存しない方法で動作する。
View	Model の表示を行うオブジェクト。
Controller	View や Model に対する操作や指示を処理するオブジェクト。

GUI では様々な操作性を提供することが可能である．内部的には同じデータ処理を行うことに対して種々の画面操作を提供することもある．一方，ユーザが画面から行う操作にかかわらず，本来システムとして必要なデータ処理があるはずである．このように GUI を持つシステムの内部処理は煩雑になりがちである．MVC では GUI を意識してデータ部分と表示部分に分けて設計することになる．MVC モデルを用いて設計することは，UI の処理によって煩雑になりがちな GUI システムに対して見通しのよい設計を行うことができるという利点がある．

「データシート」の開発においても，いくつかの個所で MVC を意識した設計を行っている．例えば，「データシート」はいくつかのドキュメント構成部品クラスを管理する．この部品クラス群を Model，各部品クラスに対応するボタンを持つ部品パレットを View，と位置づけている．部品パレット（View）をクリックすると，部品パレット（Model）は該当部品のインスタンスを生成する．具体的には部品パレット（View）上のボタンのクリックイベントを処理する時に部品パレット（Model）の持っているインスタンス生成メソッドを呼び出している．部品パレットの場合，View は Controller を兼ねている．

4.5 イベントメカニズム

「データシート」では，オブジェクト間の情報伝達手段として，JDK[*6] の AWT[*7] で用いられているイベントメカニズムを踏襲することにした．ここでいうイベントと

は例えば“ GUI のボタンが押下された ”，“ キー入力が発生した ”，“ 値が変更された ” などの事象を指す．

このイベントメカニズムには 3 種類のオブジェクトが関係している．

- イベント発信者
- イベント受信者（Listener）
- イベントオブジェクト

イベントオブジェクトは発生した事象をオブジェクト化したもので，事象に関連した情報（値変更イベントであれば新しい値，など）を持つ．イベント発信者（以下発信者）は事象の発生を監視し，事象が発生したらイベント受信者にそれを通知する役割を担う．イベント受信者（以下受信者）はイベントの発生を引き金として何らかの処理を行うオブジェクトである．

イベントの伝達は以下のような手順で行われる．

① 受信者（または，その代理人）は発信者に対して，発信者が管理している受信者リストに受信者の登録を依頼する．

② 発信者は伝達すべき事象が発生したら，イベントオブジェクトを生成する．

③ 発信者は受信者リストに登録されている受信者にイベントオブジェクトを渡す．

④ イベントオブジェクトを受け取った受信者はそれぞれの処理を行う．

（受信者は複数存在し得る（登録できる）ので③，④が繰り返される．）

「データシート」では，例えば，このイベントメカニズムをページの挿入/削除に伴うページの構成変更イベントの伝達に使用している．「データシート」のドキュメント構成部品には，ページ番号を表示することができるシート情報部品を用意している．この場合，発信者はページを管理するシート（Model），受信者はページ上に配置されたシート情報部品（該ページのページ番号を表示している），イベントオブジェクトはページ構成変更イベント（属性としてページ番号を持つ）である．これにより，シートにページが挿入/削除されると受信者であるシート情報部品にそれが通知され，シート情報部品は表示している値を新しい値に変更する（図 6）．

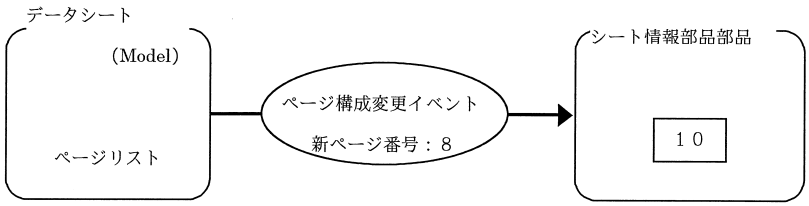

図 6　シート情報部品のイベント伝達

このイベントメカニズムを用いる利点は，オブジェクト間の依存関係を減ずる（オブジェクトの独立性を高める）ことにより拡張を容易にすることにある．

コーディング時に，発信者が受信者について知っている情報は“ 所定のメソッド（この例では，名前が changePageStructure で引数が PageStructureChangeEvent）を

持っているオブジェクト”ということだけで良く，受信者がどのようなクラスのオブジェクトであるかといったことについては知らなくて良い．また，受信者が何人いるのかといったことも知らなくて良い．発信者は，単に，プログラム実行時に受信者リストに登録されている受信者オブジェクトの changePageStructure メソッドを呼ぶだけで良いのである．

これにより，各ページにページ番号を表示するシート情報部品を配置していれば，ページの挿入/削除によって全ページのシート情報部品の表示が変更される，といったことが可能になる．また，ページのウィンドウタイトル部分にも該ページ番号を表示しているが，この部分もページオブジェクト自体がページ構成変更イベントの受信者になることによって実現している．

4.6 シリアライズによる保存機能の実現

「データシート」は，ドキュメントを作成するためのアプリケーションであり，作成したドキュメントの「ファイルへの保存と読み込み」は必須の機能である．ドキュメントの保存は，ドキュメントの情報（名前，ページ数など），ドキュメントの内容（ページ上のドキュメント構成部品，部品のサイズ，部品の位置など）の保存が必要である．ドキュメントの読み込みは，保存した情報を元に再度保存時の状態を画面上に再現する必要がある．「データシート」では，シリアライズを利用することで作成したドキュメント全体を保存/読み込みしている．

シリアライズはストリームに対するオブジェクトの入出力を可能にする．つまり，ネットワークであれ，ファイルに対してであれ，ストリームを取得できる媒体であればどこにでも入出力が可能になる．ファイルを対象とした場合，オブジェクトはプログラムの終了後も保存されるため永続化が実現できる．シリアライズでは，ターゲットにした一つのオブジェクトだけを保存するのではなく，そのオブジェクトから参照されている全てのオブジェクトをまとめて保存できる．保存したオブジェクトを再構築するとそれが参照していたオブジェクトも一緒に再構築される．仮にシリアライズ対象としたオブジェクトしか保存されないのであれば，開発者はそのオブジェクトと関連する（参照している）全てのオブジェクトを管理し，保存したい（シリアライズに必要な）オブジェクトの全てをシリアライズしていかなくてはならない．その時，ひとつでもオブジェクトをシリアライズし忘れると，オブジェクトの再構築ができなくなってしまう．各オブジェクトの保存と再構築のタイミング（順番）も自分で管理していかなければならない．

新規に定義するクラスをシリアライズ可能にすることは簡単なことである．単に，Serializable インタフェースを実装し，シリアライズ可能であることを宣言するのみである．また，シリアライズ対象が参照する全てのオブジェクトがシリアライズされるが，その中でシリアライズ対象外であることを明示することも可能である．シリアライズされたオブジェクトのバージョン管理の手段も提供している．

シリアライズによって，「データシート」のドキュメント保存/読み込み機能はシンプルに実装できた．

5. おわりに

本稿で紹介した開発は，1997 年 12 月から 1998 年 3 月にかけて行われたものである．開発当時の JDK のバージョンは 1.1 であったが，本稿執筆時点では既に Java 2 がリリースされている．そして，Java の技術は発展し続けている．Java は様々な技術を提供し多くの可能性を持っており，それらの技術をいかに利用するかが問題である．本稿で紹介のアプリケーションは，JavaBeans をサポートするフレームワークと位置づけることができ，Java 技術の 1 実践例である．

＊ 1　Beans Development Kit. JavaBeans 開発ツール.
＊ 2　BDK 付属のサンプル Bean の一つで，お手玉をするアニメーション.
＊ 3　JavaBeans で定義の Bean プロパティをカスタマイズする Wizard.
＊ 4　Bean に関する情報（メソッド，プロパティ，イベントなど）を定義するクラス.
＊ 5　インタフェースとはメソッドシグニチャ（メソッド名，返り値の型，引数の型）の集合体である．メソッドの中身は“インタフェースを実装する”と宣言したクラスに任せられている．クラスはインタフェースを，必要なだけ，いくつでも実装することが出来る.
＊ 6　Java Development Kit. Java 開発環境.
＊ 7　Abstract Window Toolkit. ボタンなどを使った GUI を作るためのツールキット.

参考文献　［1］　マイケル・モリソン，ジェリー・エイブラン著，福井眞吾，久野禎子，久野靖訳『続 Java 言語入門新しいフレームワークと API』プレンティスホール 1998 年 12 月 pp.120–124, pp. 275–291.
［2］　青木淳著『オブジェクト指向システム分析設計入門』ソフト・リサーチ・センター 1993 年 2 月 pp. 120–126.

執筆者紹介　櫻　井　優　子（Yuko Sakurai）
1987 年追手門学院大学文学部心理学科卒業．同年日本ユニシス(株)入社．OS/3 to OS 1100 コンバージョン，分散システム運用管理プロダクト開発を経て，1997 年 Java プロジェクトに参加．現在，関西支社開発室所属.

熊　本　厚　志（Atsushi Kumamoto）
1985 年岡山大学経済学部卒．同年日本ユニシス(株)入社．金融系の AP システムの開発を経て，92 年よりオープン系のシステム開発に従事. 現在，関西支社開発室所属. 情報処理学会会員.