COBOL利用技術のご紹介

東京システムハウス株式会社
システムパッケージ事業部 高橋

第2回　GUI（Graphical User Interface）

1．GUIとはなにか

かつて、ビジネスアプリケーションは、数値と文字列のやりとりだけをおこなっていました。たとえば、商品コードを入力したら商品名を表示し、数量を入力したら、単価との積を金額として表示する、といったものです。

こういった、キャラクターだけで表現する画面インターフェースを、CUI(Character based User Interface)と呼びます。CUIでは、80桁25行といった制限があり、その範囲内で画面を作っていました。強調したい部分では色を変え、罫線を使い、網掛けをしました。これだけの表現力でも、数値と文字列のやりとりだけなら、まったく問題はありません。そしてCOBOLは、そういったアプリケーションを作るための最適なプログラミング言語でした。

しかし、時代とともに、要求される表現力は高くなりました。より分かりやすく、より使いやすいインターフェースを求められてきたのです。そのために、80桁25行といった制限はなくなり、イメージが登場しました。レストランのメニューに、「カルボナーラ」といった商品名に加え、その写真が載るのと同じように、数値と文字列のやりとりをより分かりやすくする情報が追加されたのです。もし、「カルボナーラ」がどういう料理なのかを知らなくても、その写真を見れば、およその見当はつきます。

「カルボナーラ」の写真のように、イメージでその特徴、内容および機能を表現する画面インターフェースを、GUI(Graphical User Interface)と呼びます。GUIは、真か偽かを表現するにはチェックボックス、処理のトリガーにはボタンといった、特定の役割を持った要素で構成されます。この要素のことを、コントロールと呼びます。

GUIの分かりやすさは、コントロールをうまく使うところにあります。さまざまなアプリケーションで、同じコントロールが使われていますが、同じチェックボックスであれば、どんなアプリケーションであれ、真偽を示すのに使われていますし、ボタンであれば、処理のトリガーとして使われています。これが、GUIの強みです。一度、そのコントロールの役割を覚えれば、どんなアプリケーションであれ、同じ意味になるからです。

また、マウスというポインティングデバイスの登場により、直感的で幅のある操作が可能になりました。チェックボックスをクリックするとレ点がはいりますし、ボタンを

クリックすると、窪んだイメージで表示されます。ユーザーの操作が、視覚的に分かりやすいものになるのです。

　インターフェースの進化とともに、COBOLも進化しました。要求される高度な表現力を持つに至っています。チェックボックスであれボタンであれ、あるいは「カルボナーラ」のイメージであれ、COBOLアプリケーションは表現できます。

2. GUIの設計手法

GUIを設計する場合、現在では各ベンダともに画面設計ツールで行うのが一般的です。CUIの80桁25行の画面だったときは、方眼紙に「ここは表示のみ、ここは入力項目、そしてここに罫線を引く」という設計方法で問題はありませんでした。しかし、GUIでは、各コントロールに設定できる属性も多く、また、80桁25行の制限を打ち破ったために、紙上での設計が難しくなったのです。とくに、アプリケーションの画面を表示する領域であるウィンドウのサイズを、実行時に変更することも可能ですから、そこまで紙上で設計するのは至難です。すなわち、設計者の意図が、実装者、すなわちプログラマーに伝わりづらくなったのです。

しかし、設計者がソフトウェアで画面を設計したらどうでしょうか。プログラマーは、そのソフトウェアで作られた画面をすぐに表示し、ユーザーからの入力を受け取るようにコーディングできます。このように、ソフトウェア上で画面の設計を可能とするツールは、次のキャッチフレーズとともに登場しました。「WYSIWYG(What You See Is What You Get)」つまり見たままを得られる、そんなソフトウェアです。

WYSIWYGを実現したソフトウェアの多くは、ただ画面を作るだけではなく、アプリケーションを作るためのさまざまな機能をも提供します。たとえば、プロトタイプ、テンプレート、さまざまなユーティリティなどです。そういった機能を提供し、高速にアプリケーションを開発するためのソフトウェアを、RAD(Rapid Application Development)ツールと呼びます。PC上で稼動するCOBOL製品も、ほとんどのばあい、RADツールを使って開発します。

WYSIWYGのほかにも、GUIを設計するときに考慮すべき概念があります。それは、イベントと呼ばれる、OS側からのメッセージです。たとえば、マウスの右ボタンを押された、ウィンドウの閉じるボタンを押された、といったイベントがあり、アプリケーションはそれを受けて処理をおこなうこともできます。このような、イベントによって処理が動き出す仕組みを持ったプログラムを、「イベントドリブンプログラム」「イベント駆動型プログラム」と呼びます。

従来のCUIプログラムは、上から下へ水が流れるように処理が進んでいきました。しかし、イベント駆動型プログラムは違います。発生したイベントに応じて処理をおこないます。したがって、ユーザーインターフェース上の処理をフローチャートで表わすことは非常に困難になりました。そのかわり、ユーザーインターフェースの機能は飛躍的に向上しました。マウスが動くたびに画面を書き換えたり、キーボードからの入力があるたびに処理をおこなったりすることが可能になったのです。

3. 簡単なGUIアプリケーションの開発例

ここでは各ベンダのCOBOL製品の中からACUCOBOLを例として用い、簡単なGUIプログラムの開発例を説明します。

図 1　GUI画面の設計

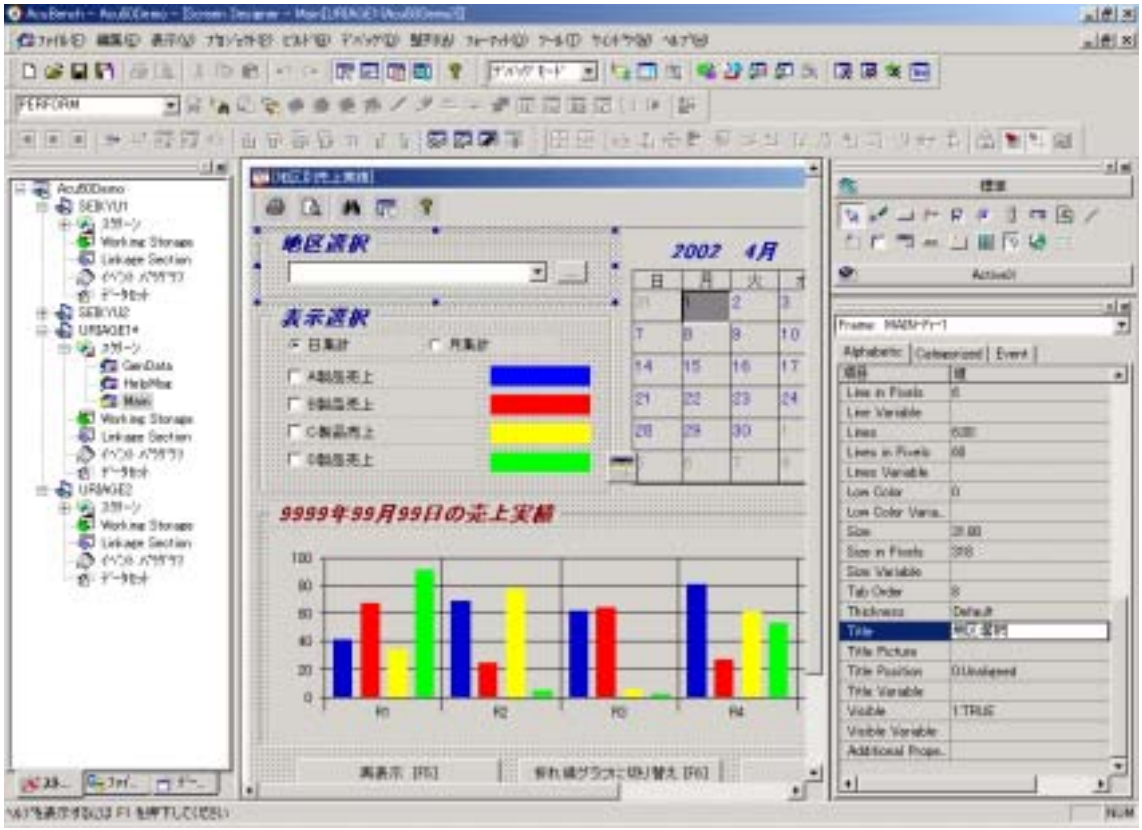

図1は、GUI画面の設計例です。ご覧のように、ツールバーにあるイメージ、画面下部のステータスバーといった、GUIならではの画面を簡単に作れます。また、カレンダーがありますが、ActiveXコントロールも使えます。このActiveXコントロールを使うことで、さらに広がりのあるGUIを開発できます。

図 2　GUI プログラムの実行例

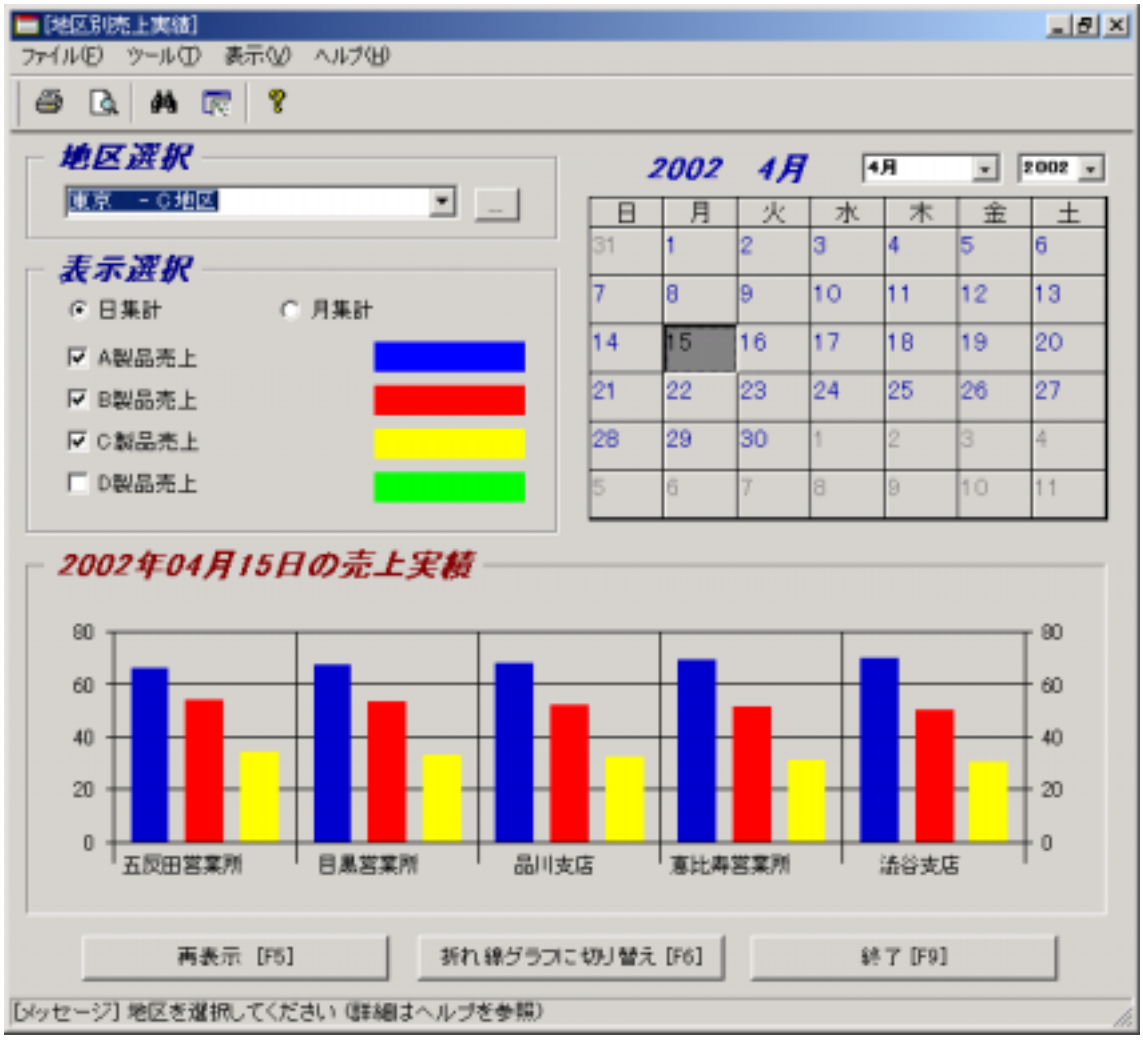

図2は、実行時の画面です。設計時の画面と比べてみてください。表示されているデータが違うだけで、位置、サイズ、色といった画面の属性は、設計時のままです。

図 3　GUI コントロールとイベント定義

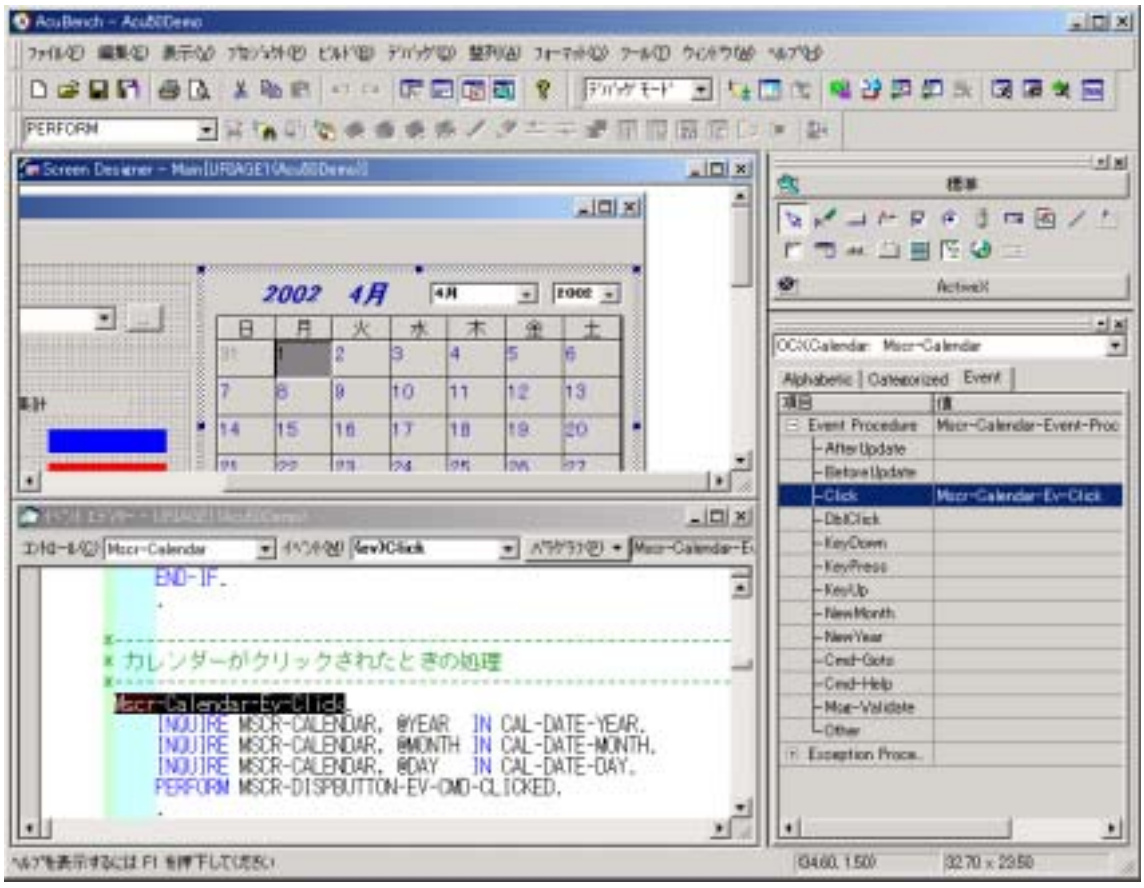

図 4　GUI プログラムのコード例

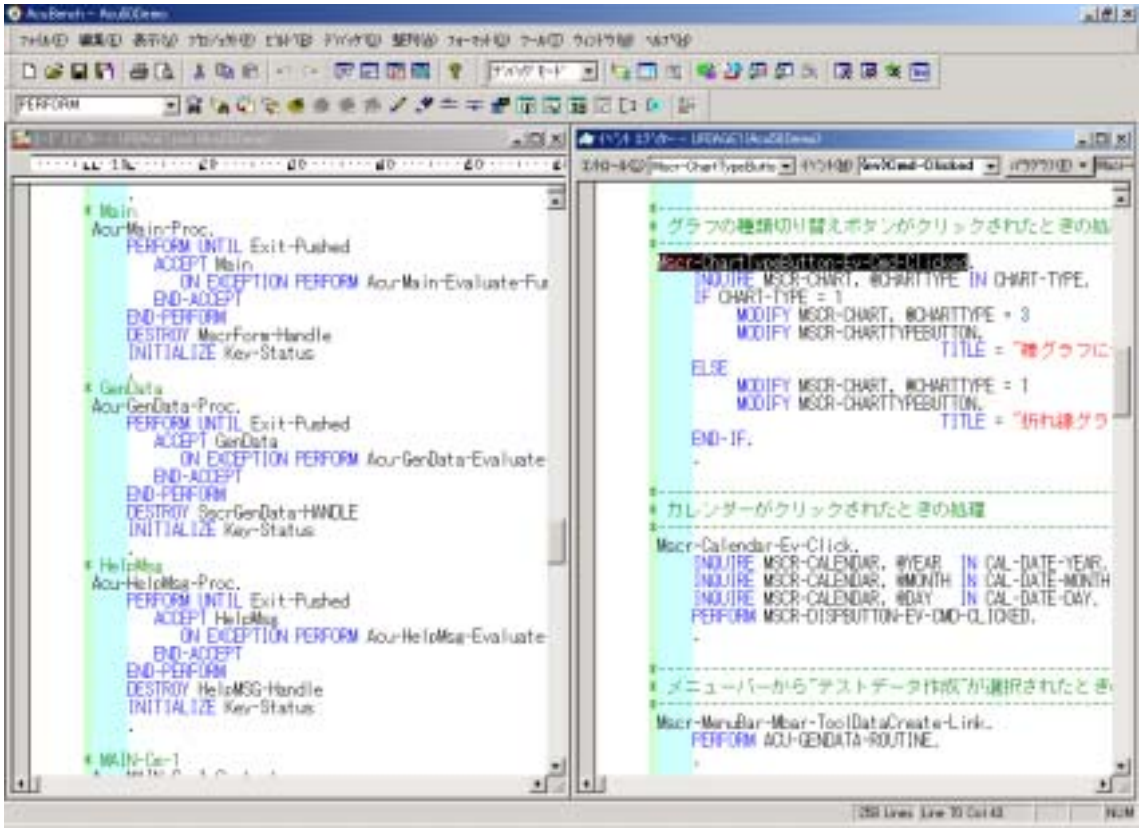

図３は、Active-Xのカレンダーコントロールの Click イベント手続きを作成している例です。イベント手続きの中では当然ですがCOBOLの命令を記述することが出来ます。図４は、GUIを扱うプログラムコードです。画面の表示と入力は、READ/WRITEあるいはACCEPT/DISPLAYといった、COBOLの言語仕様に沿って記述します。GUIだからといって、CUIとまったく違うわけではありません。また、RADツールを使うと、画面の表示と入力といった基本的な部分を隠蔽し、開発者はビジネスロジックに集中できます。

このようなRADツールの登場により、GUIはCOBOL開発者にとって、かなり身近なものになります。従来のCUIアプリケーション開発手法から考えれば、設計手法、概念、そしていくつかの新しい構文を覚える必要があるかもしれません。しかし、そのわずかな労力を補って余りある機能と表現力を、GUIアプリケーションでは得られるのです。