Drawrialを利用したGUI構築事例

日本ビクター株式会社　育成事業部
事業開発統括部　BD システム部　技術グループ
寺田　克彦

1．背景

AV機器の機能複雑化に伴い、その機器に搭載されるGUIが肥大化、複雑化しています。特に、BDレコーダ/BDプレーヤなどの機器では対応するメディア数が多く、さらにネットワーク機能などが追加された結果、膨大な画面数となってきています。そのため、単にGUI実装の効率化だけではなく、開発プロセス自体の見直しと効率化を迫られています。

このような状況において、市販GUI開発ツールの調査を行ってきましたが、その中で注目したのが、キャッツ株式会社の「Drawrial」です。本来このツールは、GUIの仕様書作成と、仕様レビューなどの上流工程の作業を効率化するためのツールですが、我々はこれを実装までシームレスにつなげることで、「Drawrial」本来の特徴である上流工程の効率化に加えて、下流工程の効率化を実現させたいと考えました。

2．開発プロセス

「Drawrial」を上流工程から下流工程まで利用することの最大の利点は、GUI設計仕様書を作成しレビューが終了すれば、その設計情報がそのままGUI実装のインプットになることです。これによって中間の工数が大幅に削減できることになると考えました。

従来は、図１の左図のような開発プロセスを実行していました。「要求仕様書」を受けて詳細設計を行い、そのアウトプットとして「GUI設計仕様書」を作成し、これをレビューし、完成度を高めていきます。しかしながら結局実装のフェーズでは新規に「GUI設計仕様書」と同じ内容をC/C++言語などに置き直す必要があります。

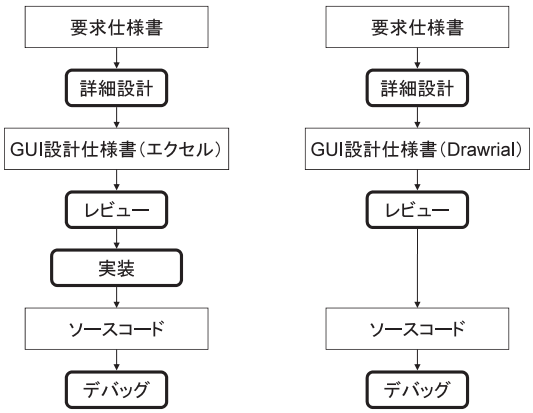

図１　従来の開発プロセスと、Drawrialを使用した開発プロセス

これに対して、「GUI設計仕様書」のフェーズにDrawrialを利用することで、レビュー済みの精度が高い設計情報をソースコードへ展開することが可能になります。

図１の右図のように、Drawrialで作成したGUI設計仕様書は、レビューを経た後ソースコードへ自動翻訳されます。もちろん全てをDrawrialで記述することはできませんので、実際には実装フェーズは残っており、完全に無くなるわけではありません。しかしながらGUIを構成する描画部品の多くは既に配置され、階層関係も決定され、画面遷移も記述されているために、従来の開発プロセスに比べると、無駄が大幅に削減されています。

Drawrialの設計情報（XML）をソースコードに自動翻訳する部分は、キャッツ株式会社の協力の下に日本ビクター株式会社が開発いたしました。（現在のDrawrial V3の機能とは異なる実装です）

3. Drawrialの応用

一般的なGUI実装ツールは、部品のオブジェクトを定義してDrag&Dropで配置していくものが多いようです。これはPCのGUIのようにボタンやスクロールバーの部品オブジェクトがデザインも含めて動作が定義されている世界では有効です。部品オブジェクトは、GUI実装ツールで既に準備されており、あとは部品オブジェクトを順次配置していくだけです。

一方、組み込みの世界においては、動作やデザインに一定のルールはあるものの、PCの世界に比較すると部品オブジェクトはより多彩です。したがって標準部品よりは、カスタム部品の方が多くなり、部品オブジェクトの作成が面倒です。しかも商品が変われば部品オブジェクトも変更になる可能性があり、PCの世界のGUI実装とは同列には論じられません。個別にカスタム部品を作成していくのも一つのアプローチだと思いますが、我々はDrawrialを利用することで別のGUI実装の可能性に気づきました。

それは、Drawrialと同様に操作イベントによってGUI部品を配置した画面を次々に切替えていくという手法です。たとえば、図2で言えば、「フォーマットしない」ボタンにフォーカスが当たっている画面から、左に移動するキーを押下することで「フォーマットする」ボタンにフォーカスが切り替わります。この2つの画面の遷移を、部品オブジェクトに状態変化を指示すると

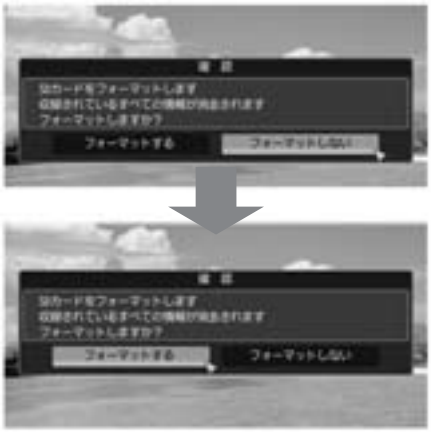

図2　画面遷移の例

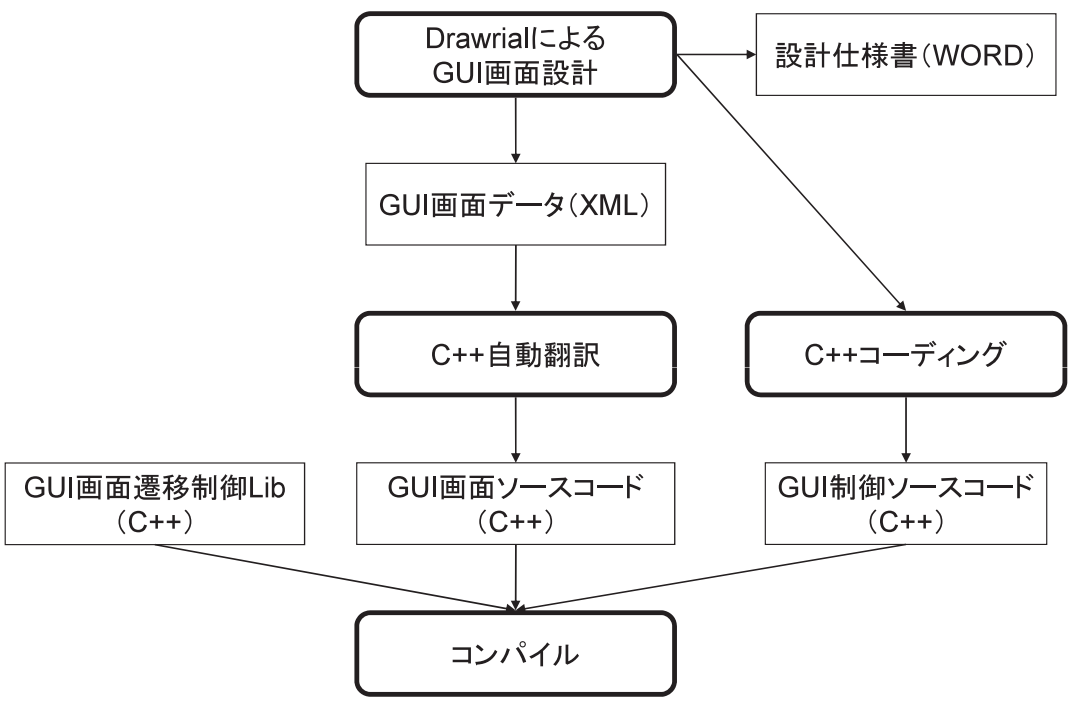

図3　ソースコード生成の流れ

いう方式ではなく、単なるビットマップ画像で構成された画面の遷移だと捉えることにします。このような簡単な例だと、ほぼDrawrialでの記述のみでソースコードを生成することができます。

4. システムの概要

実際の工程とグラフィックシステムをソースコードの生成という観点からもう少し詳細に説明します。

図3のようにソースコードとして最終的にアウトプットされるものは下記の3種類のソースコードです。

1） GUI画面遷移制御Lib
2） GUI画面ソースコード
3） GUI制御ソースコード

GUI画面遷移制御Libは、本グラフィックシステム向けに新規に開発されたライブラリーで、Drawrialによって生成されたGUI画面ソースコードを実際に描画する機能を有しています。図4のように複数の画面ABCの情報を保持し、定義されたイベントによって画面を遷移させていきます。遷移の判断は、GUI画面ソースコードに含まれる遷移情報に基づいて決定されます。

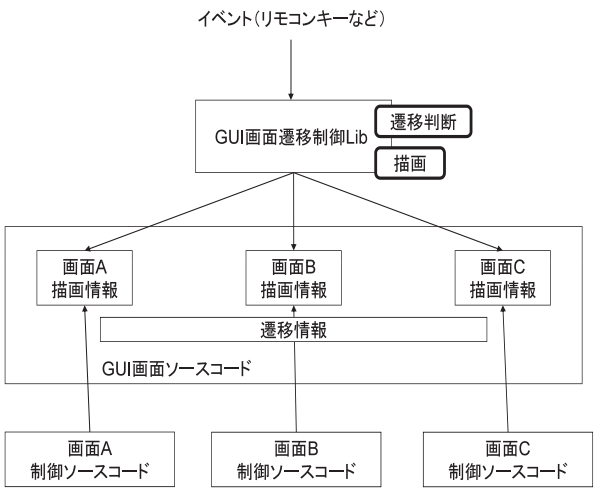

図4　GUI制御の関係

このライブラリーは汎用的に設計されているため、一度作成すれば別の商品開発にも適用できます。

このGUI画面遷移制御Libに画面描画情報と遷移情報を与えるのが、GUI画面ソースコードです。これは、Drawrialで作成されたGUI画面の構成情報、遷移情報をDrawrialがネイティブで管理しているXMLドキュメントから自動翻訳されて生成されます。

GUI制御ソースコードは、Drawrialでは記述しきれないGUIの動作をコーディングしたものです。理論上はDrawrialで記述することもできますが、たとえば画面と部品の組み合わせが膨大になり、画面遷移だけで記述しないほうが有利な場合もあります。このような場合は、画面部品を動的に変更可能な状態にして、GUI制御ソースコードによって描画を管理していきます。図4のように、画面ABCのそれぞれの制御を受け持つGUI制御ソースコードが複数存在します。

5. 今後の課題

今回ご紹介した開発プロセスとグラフィックシステムは、実際に日本ビクター株式会社のBlu-ray/DVD製品の一部に適用されています。今後の課題として、

1） Drawrialでどこまで記述すべきかの切り分け
2） Drawrialで記述する際の記述ルールの明確化
3） GUI画面遷移制御Lib描画速度の改善

などが考えられます。

1）の切り分けは、2）の記述ルールとも密接な関係があります。Drawrialで記述しない部分の制御をGUI制御ソースコードで行うためにDrawrialの記述ルールを設定する必要があるからです。たとえば、条件分岐などをGUI制御ソースコードで記述するために、Drawrial上では、条件分岐を意味する記述をする必要があります。自動翻訳ツールはこのような記述を見つけ出すと、GUI制御ソースコードに条件分岐のためのフックポイントを提供します。

切り分けと記述ルールによって、GUI制御ソースコードの内容は、大きく異なるものになり

ます。これは、複数のGUI開発者が参加するような規模の大きいプロジェクトではソースコードのばらつきなどを押さえるために考慮しなければならない課題です。

また、本システムでは画面の動作を画像の遷移として捉えるために、まともに全画面を描画していくと、ほんの一部の画像の遷移でも全画面を再描画することになります。これを避けるためには、画面内の差分を抽出して差分のみを描画する手法が考えられますが、これの最適化を図ることによって、高速なグラフィックシステムを実現できる可能性があります。

6. おわりに

ソフトウェアの開発のなかでもGUIに絞って検討を進めてまいりました。現状はまだまだ効率的な手法として確立されたとはいえませんが、今後もZIPCなどとの連携を検討しながらより効率的な手法に進化させて行きたいと考えています。

WATCHERS Back Numberで振り返るユーザ会

第12回を迎えた「ZIPCユーザーズカンファレンス」では、特別セミナー「形式手法と組込みソフト」におきまして、北陸先端科学技術大学院大学の二木厚吉先生、青木利晃先生をお招きし、形式手法の最新動向についてご講演頂きました。

基調講演では、「エンピリカル・ソフトウェア工学」について、奈良先端科学技術大学院大学の松本健一先生より、ソフトウェアの生産性、品質の向上実現に向けてご講演頂きました。

事例講演では、家電製品やPHSなど開発支援ツールの適用効果が発表されました。