X Toolkit上でのアプリケーションプログラム作成支援ツール

7N-3 Weiss-XT

杉本直樹* 武田博隆* 高田司郎* 松浦敏雄** 吉岡信夫***
*CSK総合研究所 **大阪大学 ***大阪工業大学

1. はじめに

UNIX上の代表的なウィンドウシステムであるX Window[1]では、プログラマの負担を軽減するためにX Toolkit[2]という画面上での部品のライブラリを提供している。しかしX Toolkitを用いても、画面の構成を確認しながら部品を配置することができず、部品のパラメータなどの設定も面倒であった。

部品の配置をインタラクティブに行えるツールとして、Weiss[3]やWidget Editor[4]があるが、どちらも取り扱える部品集合が固定されていて、ユーザが部品を追加することは容易ではなかった。

本稿では、画面上で部品をインタラクティブに配置することで、そのようなユーザインタフェースを実現するプログラムを生成するシステムWeiss-XTについて説明する。Weiss-XTは、X Toolkitの部品集合(Widget)を対象とし、画面上での部品配置時にも実際のWidgetを使用している。従って、生成されるプログラムの画面イメージを確認しながら、部品の配置を行える。また、Weiss-XTでは、Toolkitの利用を容易にするToolkit記述言語(XTL)を用意している。XTLを用いることでプログラムの修正、デバッグが容易になった。また、Widgetとして新しい部品を作成した場合、Weiss-XTに追加することも容易にできる。

2. Weiss-XTの構成と特徴

Weiss-XTは、

- ❑ 部品レイアウトツール WET (Widget Editing Tool)
- ❑ X Toolkit記述言語 XTL (X Toolkit Language)
- ❑ XTLからCプログラムへのトランスレータXTLtoC

からなる。

アプリケーションプログラマは、まずWETを起動し、画面上で部品の配置及び属性を決定する。WETはXTLによるプログラムを出力する。出力されたプログラムをXTLtoCトランスレータで変換すると、X Toolkitを用いたCプログラムになる。出力されたプログラムは、コンパイルすることにより、それだけで配置した部品を持つウィンドウとして起動できる。あとはプログラマが、Callbackルーチンをアプリケーションの仕様に従って記述すればよい。このプログラミングはXTLかまたはCレベルで行えばよい。(図1参照)

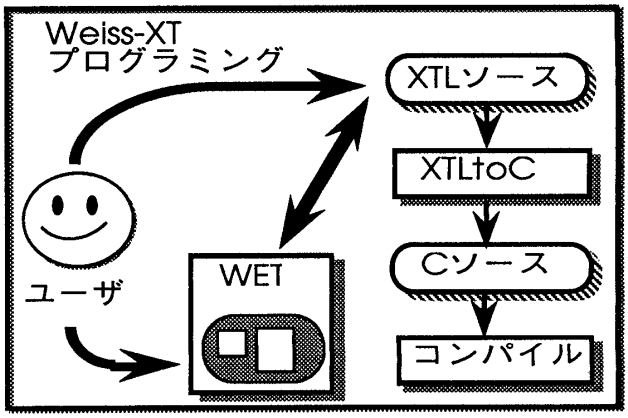

図1 Weiss-XTプログラミング

Weiss-XTを用いることにより

- ❑画面上で部品のレイアウト、属性の変更が可能で、その結果が即座に画面に反映される (→3.参照)
- ❑ X Toolkitのパラメータの設定が容易 (→4. 参照)
- ❑ X Toolkitレベルでのエラーチェックが容易 (→4.参照)

3. WET - 部品レイアウトツール

WETはXToolkitの部品(widget)を画面上でインタラクティブにレイアウトするツールである。(図2参照)

WETに関する操作はマウスによって行われる。部品を作成するには、図2の左上の部品一覧ウィンドウで作

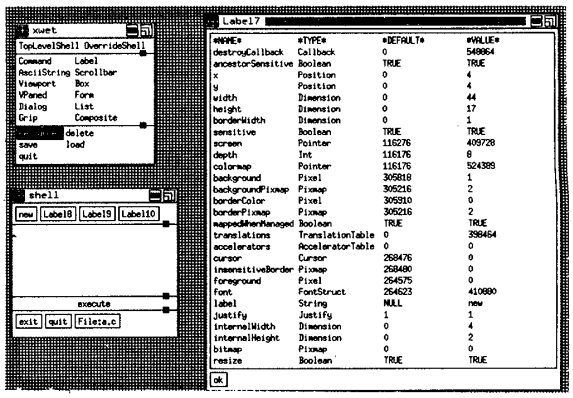

図2 画面上での使用例

Weiss-XT : User Interface Program Generator on X Toolkit
Naoki SUGIMOTO[1], Hirotaka TAKEDA[1], Shiro TAKATA[1], Toshio MATSUURA[2], Nobuo YOSHIOKA[3]
1. CSK Research Institute, 2. Osaka Univ., 3. Osaka Institute of Technology

成したい部品を選択し、左下のレイアウトウィンドウで指示すればよい。現在のところ基本的な部品として、"Intrinsics"と"Athena Widget"の部品を用意している。既に配置した部品の移動、変形もレイアウトウィンドウ上で行える。作成した部品に関する属性情報も右のような属性ウィンドウでみることができる。属性ウィンドウ上のリソースデータを変更することも可能である。変更結果はレイアウトウィンドウ上に即座に反映されるので、画面上で容易に確認できる。レイアウトウィンドウでの部品群の編集結果は、saveによりXTLプログラムリストとして出力される。

WETへの部品の追加は、その部品のプログラムと部品のリソース定義ファイルを準備し、WETを再構成すればよい。

WET自身はXTLを用いて記述されている。

4. XTL - X Toolkit記述言語

XTLはToolkitの部品の操作に関する記述を行うための言語である。（図3参照）

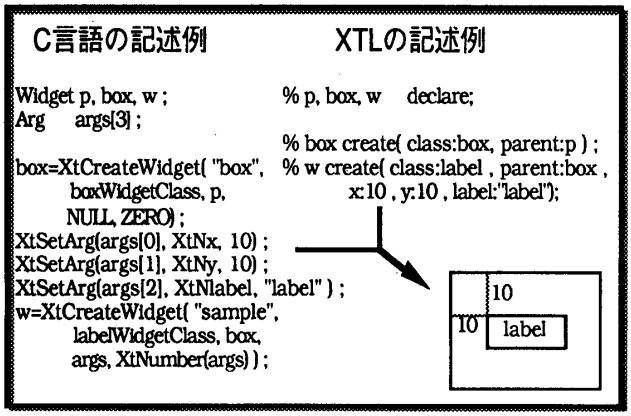

図3 C言語とXTLの対応例

特徴として、

- ❑ キーワード付パラメータなので、
  - ◆ 引数の数や順序が自由
  - ◆ デフォルト値が与えられるので パラメータの省略が可能
  - ◆ パラメータの配列渡しを無くし 直接引数で書くことが可能
- ❑ C言語に混在して使用可能

XTLのこれらの特徴により、XTLプログラムリストは読みやすく、一般にCのリストより短くなった。Cで記述すると図3の左のようなプログラムが、XTLでは右のようになる。また、XToolkitのプログラミングでよく起こりがちな引数の順序の間違いや、パラメータの設定ミスを減らすことができた。これらのミスはXToolkitのレベルでのエラー情報がでないため、デバッグが非常に困難であったが、XTLで記述することにより、XTLレベルでエラーチェックが行えるようになった。XTLプログラムリスト中にCの制御構造や演算もそのまま使用可能である。XTLプログラムリストはXTLtoCによりCのプログラムリストに変換される。

5. おわりに

Weiss-XTは現在プロトタイプがX11R3上で稼働中で、WETがXTLで約1500行、XTLtoCトランスレータがC言語で約2500行となっている。Weiss-XTにより、画面イメージを確認しながら部品のレイアウトが実行でき、またXTLで記述する場合もCで記述するのに比べてプログラムが短く簡明になった。特に初心者の使用には非常に役に立つであろう。

問題点として、部品のリソースがポインタやユーザ定義のリソース、関数等の場合、どのように変更できるかがわかりにくいことがある。これは、上記のデータの場合、整数でしか値をとってこれないためである。全部のリソースを修正可能にするには、Xレベルで整数からリソース名への逆変換を行うルーチンが必要である。

もう一つの問題点は、レイアウトを変更したとき、それが他の部品にも影響を及ぼす可能性があることである。これはAthena Widget自身が簡単なレイアウト機能を持っているためである。その他に、画面表示以前の指定しか有効でないリソースや、Shell,Viewportのように例外的規則を持つ部品もあるため、Atena Widgetを基本として詳細なレベルのレイアウティングを行うのは難しい。もし詳細なレベルのレイアウティングを行いたいならば、Athena Widgetを捨てて、自分で新しい部品体系を作ったほうが良いかもしれない。ただし、汎用ということとの兼ね合いが難しい。

今後の方針としては、Widget Class Editorの作成を考慮中である。部品定義自体をインタラクティブに作成することを目標にしたいと考えている。

参考文献

[1] Sceifler, R. W. and Gettys,. : "The X Window System ", MIT , Laboratory for Computer Science (Jul . 1986)

[2] " X Toolkit Intrinsics - C Language X Interface , X Window System , X Version 11, Release 3 (1988)

[3] 杉本他：" X Window上のユーザインタフェースプログラムジェネレータ", 情報処理学会第37回全国大会講演論文集 2K-7 (1988)

[4] Richard Carling : " Widget Editor ", Massachusetts Computer Corpration (1988)