福井大学
工学部 研究報告
第41巻 第2号
1993年9月

グラフィカルユーザインタフェース構築支援ツールの開発

高濱 徹行* 八木 正和** 小倉 久和* 中村 正郎*

XTSS Builder: Graphical User Interface Builder for XTSS

Tetsuyuki TAKAHAMA, Masakazu YAGI, Hisakazu OGURA,
and Masao NAKAMURA

(Received Aug. 30, 1993)

We developed the graphical user interface (GUI) builder for XTSS, XTSS Buidler, which is a GUI system for Building GUI applications described by XTSS (X Toolkit Service System). XTSS language is a simple but a powerful one for integrating textual applications on the GUI environment and similar to the X Toolkit functions. Yet, it is hard to describe XTSS program for novice users. So, we provided XTSS Builder for them.

XTSS Builder consists of two parts: Creator and Analyzer. Creator is used for constructing XTSS applications. Users can select a widget class from the widget class tree of Athena widget, place it to the user's widget tree. Creator generates XTSS program automatically from the user's tree, and realizes the GUI. Users can also select the attribute of the widget and set the value to it. Analyzer is used for reconstructing the GUI application which was not coded by XTSS. Users can easily generate XTSS program by only pressing the mouse button on the GUI application. Users can also modify it visually, if they load the program on Creator.

1 はじめに

計算機、特にワークステーションやパーソナルコンピュータにおいて、視覚的な要素を利用したグラフィカルユーザインタフェース (GUI) を有する応用プログラムの開発が盛んに行われている。GUIは、ユーザの立場からみれば、応用プログラムの使用方法を視覚的に表現するため、操作対象を直接画面上で指示でき、対象毎にどの様な操作があるかを覚える必要がなく、視覚的に情報をとらえるため直観的に分かり易い等の多くの長所を持つ。しかし、GUI環境を構築する立場から見れば、GUIアプリケーションの記述はかなり困難であり、しかも応用プログラム全体に占めるユーザインタフェース関連の部分が大きくなるため、開発の期間・コストが増大してしまうという大きな問題が存在する。[1]

これに対処するための1つの方法として、GUI記述用の簡易言語を提供することによりGUIの記述を簡単化かつ簡素化するという方法があり、OSF/MOTIFが提供するUIL[2], LISP言語であるCommon

* 情報工学科
** 大学院情報工学専攻

Lisp をベースとした CLX[3], XLISP をベースとした winterp[4], tcl 言語 [5] をベースとした tcl/tk[6], wafe[7] など様々な言語が提案されてきている。我々のグループでも同様な観点から、従来の文字入出力を基本とするテキスト型応用プログラムを GUI 環境に統合することを援助するシステムとして GUI 記述用の簡易言語をもつ 2 つのシステムを構築してきた。一つは、ウィンドウの階層を自然な形で記述するとともにウィンドウの属性および応用プログラムからの入出力を変数として取り扱うことのできる UAI/X[8] である。もう一つは UAI/X の能力を継承し、より柔軟にテキスト型応用プログラムを GUI に統合できる XTSS(X Toolkit Service System)[9] である。XTSS は X Toolkit に準拠した単純なコマンドを解釈実行するサーバと、テキスト型応用プログラムとサーバ間の通信を援助するクライアントから構成されるクライアント／サーバ型のシステムである。両システムを用いて実際にいくつかの GUI アプリケーションを作成し、いづれも良好な結果を得ている。

ところで、たとえ簡易言語を用いたとしても、実際に GUI アプリケーションを記述することは、特に初めて利用するユーザにとってはかなり困難であり、新しい言語を学習しなければならないという抵抗もある。このため、GUI の構築自体も GUI 環境下で実現する GUI ビルダーの重要性が認識されるようになってきている。GUI ビルダーは、視覚的な操作で GUI の構成を指定し GUI の記述を自動生成することによって GUI を構築するシステムであるが、その構成の指定方法には様々な形式がある。指定の形式を大別すれば、

1. 視覚的に表示された様々な部品から適切な部品を選択し白紙のウィンドウに張り付ける方法
2. メニュー等から部品名を選択し、その親子関係や位置関係を指定する方法

がある。1. の方がより視覚的な操作であるが、2. に比べて一般に実現が困難である。

本研究では、記述能力は高いが UAI/X に比べてやや記述が難しくなっている XTSS 用の GUI ビルダー XTSS Builder に関する報告を行う。XTSS Builder は、出来る限り単純で実現が比較的容易でしかも充分に視覚的な操作を提供することを目標としたため、基本的には 2. の方法を選択しているが、メニューではなくツリーによる表示・操作を全面的に採用した GUI ビルダーである。GUI 用の部品を部品のクラス階層で表したツリー構造から選択し、その部品をツリーのルートノードの下に親子関係を考えながら張り付けることによって、作成する GUI アプリケーションのツリー構造を指定するという方法を取る。部品の属性については、部品の持つ属性の一覧表から選んでその値を指定することができる。さらに、XTSS 以外で記述された既存の GUI アプリケーションを XTSS で再構築する作業を援助するために、GUI アプリケーションの部品の親子関係を自動的に取得し、XTSS の記述に変換する機能も提供している。なお、ツリー状に示す部品の選択メニューでは、部品を視覚的には表示せず文字記号で示しているため、構築の途中の GUI の形状を表示する機能も提供している。

以下 2 章では、XTSS について簡単に説明し、XTSS Builder の概要について説明する。3 章では、ツリー上で GUI を構築する XTSS Builder の本体である Creator について説明し、4 章では既存の GUI アプリケーションを XTSS の記述に変換する Analyzer について説明する。5 章はまとめである。

2 XTSS と GUI ビルダー

2.1 XTSS の概要

XTSS は、C 言語などにより X のライブラリを利用するのではなく、sh, csh, perl などに代表されるシェルのようなテキストの操作を基本とする言語から X Toolkit を使用できるようにするためのクライアント/サーバ型のシステムである。XTSS は X Toolkit を操作するための各種のコマンドを提供する X Toolkit Server(XTS) と、XTS とシェル間の通信を助ける X Toolkit Client(XTC) から構成される。

XTS は、X ウィンドウに関する命令である XTS 命令を受取り、それを解釈・実行し、実行結果を XTS 応答として返す。XTS は単純なテキスト入出力を伴うプログラムであるため、パイプ、ソケットあるい

は仮想端末のようなプロセス間通信の機能を持つ perl 等の言語からは直接 XTS を使用することができ、それ以外の言語からも XTC を介して使用できる。XTS 命令は、大きく分けて特殊命令、Xt 命令、X 命令、Xaw 命令の 4 つに分かれている。これはそれぞれ、XTS 独自の命令、X Toolkit、X lib、Athena ウィジェットに関する命令である。特に Xt 命令は X Toolkit 関数名から Xt を除いた形式をとるため、X Toolkit プログラミングに習熟していれば簡単に理解できる。最も簡単な例として、Hello World を表示するための記述とその表示の様子を図 1に示す。

```
Initialize hello Hello
CreateManagedWidget "Hello World" \
    labelWidgetClass hello
RealizeWidget hello
MainLoop
```

図 1: XTS 命令の記述例 – Hello World の表示

XTC は、プロセス間通信の機能を持たないシェルなどのプロセスから XTS を利用する際に、この間の通信を助けるクライアントである。XTC は 1 つのプロセス (主プロセス) と XTS 間だけでなく、その他の複数のプロセス (副プロセス) との間の通信をも可能にする。XTC と XTS および他のプロセスとの通信の様子を図 2に示す。

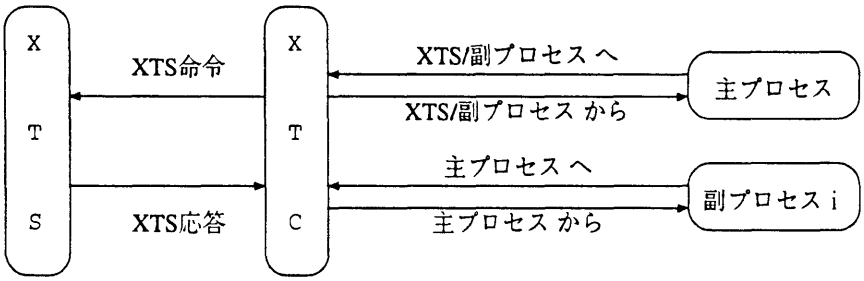

図 2: XTC による XTS と複数のプロセスとの結合

2.2 XTSS Builder の概要

XTSS Builer は、XTSS による GUI 構築を視覚的に行うための Creator と、X Toolkit によって C 言語などで記述された既存の応用プログラムを XTSS の記述へ変換するための Analyzer という 2 つの独立したシステムから構成される。図 3が XTSS Builder のトップメニューであり、Creator の起動、Analyzer の起動、XTSS Builder の終了 (Quit) を選択することができる。

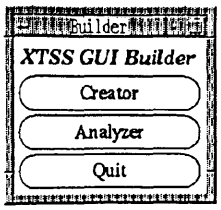

図 3: XTSS Builder トップメニュー

一般にユーザが XTSS Builder を使用する状況について考察すると、

1. 新たに独自のGUIアプリケーションをXTSSによって作成する
2. XTSSで記述された既存のGUIアプリケーション(XTSSアプリケーション)を修正する
3. 既存のGUIアプリケーションを参考にしてXTSSアプリケーションを作成する

という3つの状況がある。XTSS Builderはこれら全ての状況において利用できるように設計した。1. の場合は、Creatorにより全く新規にGUIアプリケーションを構築し、XTSSプログラムへ変換し、ファイルに保存する(save)ことができる。2. の場合も、CreatorへXTSSプログラムを読み込み(load)、GUIアプリケーションを修正し、再度保存し直せばよい。3. の場合には、まず、Analyzerによって既存のGUIアプリケーションをXTSSプログラムに変換しファイルへ出力する(dump)。次にCreatorを起動し、ファイルを読み込めば、視覚的にGUIアプリケーションを改良して行くことができる。この様子を図4に示す。

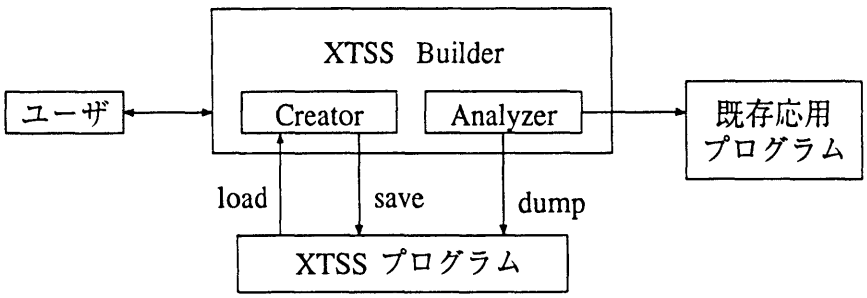

図4: XTSS Builderの利用形態

3 Creator

3.1 概要

Creatorは、XTSSアプリケーションの構築を視覚的な操作で行うのを支援するのがその基本機能である。X Toolkitを利用したGUIの構築ではウィジェットの親子関係とそのリソース(属性)値を指定することによりGUIが決定されるため、GUIビルダーの操作性は親子関係およびリソース値の指定方法によって決まる。先にも述べたように、最も視覚的な操作方法は、視覚的な部品として用意された各ウィジェットクラスの中から、ユーザがクラスを部品として選択し、マウスによりその部品を置く位置を指定すると、システムが実際にウィジェットを生成しそれを配置することにより全体のGUIを決定する方法である。しかし、この方法を実用的なものにするにはかなり大きなシステムが必要である。しかもそのようなシステムを構築したとしても、例えばリソース値の指定をマウス操作だけで行うことは無理があるため、視覚的な操作だけで全てを行うのは所詮不可能である。そこで本システムでは、かなり視覚的な操作性に富み、しかも実現が比較的簡単な方法を採用した。

Creatorでは、ユーザはAthenaウィジェットクラスのツリー構造からマウスでクラスを選択し、ユーザが構築するGUIのツリー構造の中に設定して行くことによりGUIアプリケーションを構築する。クラスの選択もGUIアプリケーション中のウィジェットの親子関係の設定もツリー構造上で行うため、統一的な操作性が保たれている。リソース値の設定は、各ウィジェットのリソース名の一覧からリソース名を選択し、それに対するリソース値を入力するという方法をとる。また、この方法では残念ながら作成中のGUIアプリケーションの形状は見えないので、作成途中の形状を表示する機能も提供している。図5にCreatorの様子を示す。

Creatorは、以下の3つのウィンドウから構成される。

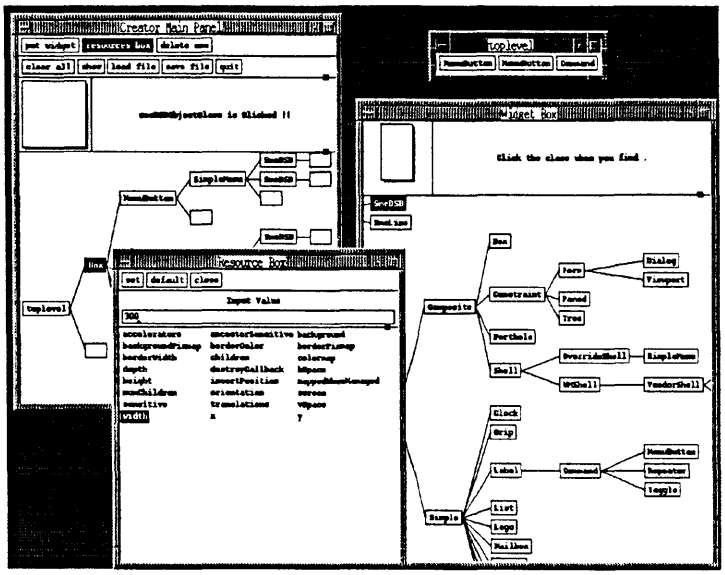

図 5: Creator

メインパネル 次の 4 つの領域から構成される。ウィジェットの配置 (put widget)・削除 (delete one)、リソースの設定 (resource box) のうちどの操作を選択するかを指定する操作指定領域。ツリーの初期化 (clear all)、GUI の表示 (show)、XTSS プログラムの読み込み (load file)、XTSS プログラムの格納 (save file)、Creator の終了 (quit) という常に使用できるコマンドを用意したコマンド領域。エラーメッセージなどの様々な情報を表示するメッセージ領域。構築しようとする GUI アプリケーションをツリー構造で表現するための配置領域。

ウィジェットボックス Athena ウィジェットのクラスの階層関係をツリー構造で表現したウィンドウ。ユーザは適切なクラスをマウスにより選択することができる。

リソースボックス 次の 3 つの領域から構成されている。リソース値の設定 (set)、リソース値を暗黙の値に戻す (default, リソース値の削除)、およびリソースボックスの終了 (close) というコマンドを用意したコマンド領域。リソース値の入力を行なうための入力領域。設定可能なリソースを表示し、値を設定しようとするリソース名を選択するリソース名リスト表示領域。

操作方法について以下に簡単に触れておく。

GUI アプリケーションにおけるウィジェットクラスの親子関係を指定するためには、まず GUI アプリケーションで使用するウィジェットクラスをウィジェットボックスから選択し、次に、メインパネルの操作指定領域で put widget を指定する。そして、配置領域中のツリー構造上の適切なノードをクリックすることにより指定したクラスのウィジェットを配置できる。ツリー構造上で配置可能な位置には自動的に空白のノードが長方形で表示されるため、ユーザはスムーズにウィジェットを配置できるようになっている。

リソースを設定するには、操作指定領域で resource box を指定する。この後、配置領域のノードをクリックすれば、リソースボックスがポップアップされる。設定したいリソース名をリソース名リストから選択し、入力領域にマウスを合わせ、リソース値をキーボードから入力する。

XTSS プログラムを Creator に読み込むためには、メインパネルのコマンド領域の load file をクリックする。XTSS プログラムを生成するには、GUI の構築をした後、save file をクリックすればよい。こ

れによりダイアログウィジェットがポップアップされるので、ファイル名を入力すればXTSSプログラムを読み込みあるいは出力することができる。

一度配置したものを削除したい場合には、操作指定領域でdelete oneを指定して、配置領域のノードをクリックする。

現状のGUIを確認するには、メインパネルのコマンド領域でshowをクリックすればよい。

3.2 XTSSプログラムの解析・生成

Creatorは、load fileが指定されれば、XTSSプログラムを解析し、各ウィジェットの情報をWInfo構造体に、リソース情報をResourceCell構造体に格納する。また逆に、save fileが指定されれば、WInfoとResourceCellの2つの構造体を参照し、XTSSプログラムを生成する。

WInfo構造体は、ツリーのノード間の関係を表すために、ノードを表現しているウィジェットを指すwidget、ツリー構造上の親のWInfoを指すparent、子のWInfoを指すchild、兄弟を指すbrotherという要素を持つ。そのノードのリソースの状態を示すResourceCell構造体を指すのはresourceである。その他に、ウィジェットの名前を示すname、クラス名を示すclass、クラスの番号を示すclass_num、生成されるウィジェットの種類を示すtypeがある。typeとしては、INITIALIZE, APPINITIALIZE, CREATE, MANAGE, SHELL, POPUPがあり、それぞれXt命令のInitialize, AppInitialize, CreateWidget, CreateManagedWidget, CreateShell, CreatePopupShellというウィジェット生成用命令に対応する。

ResourceCell構造体は、リソース名を示すresource_name、リソース値を示すresource_value、次のリソース構造体を指すnextという要素を持つ。

XTSSプログラムの解析は、以下のような手順で行う。

1. CreateManagedWidget等のウィジェット生成用命令の場合には、
   (a) 新しいWInfo構造体を生成する。命令に対する引数を解析することにより、name, class, typeをWInfoに設定する。
   (b) 親の名前をキーにして親のWInfoを検索し、取り出した親のWInfoとの間で親子関係のリンクを張る。
   (c) ウィジェット名をキーにしてWInfoを保存する。
2. それ以外の命令は、一行単位に単純な文字列として保存する。ただし、現在のところ、この情報はXTSSプログラムを生成する際には使用されない。この点は課題として残されている。

XTSSプログラムの生成は、以下のような手順で行う。

1. ウィジェット階層のトップのWInfoのname, class, resourceを取り出し、typeに応じて

   ```
   Initialize <name> <class> NULL <resource>
   AppInitialize appcon <class> NULL NULL <resource>
   ```

   を出力する。ただし、<resource>は、ResouceCellのresource_nameとresource_valueを順に、

   ```
   <resource_name> <resource_value> ...
   ```

   と出力したものである。
2. 子あるいは兄弟のWInfoがある場合には、まず子について、次に兄弟について、そのWInfoのnameとclassを取り出し、typeに応じて、

```
Create(Maneged)Widget/CreatePopupShell <name> <class> <parent> <resource>
AppCreateShell <name> <appclass> <class> <parent> <resource>
```

を出力する。ただし、<parent>は親の WInfo のウィジェット名であり、<appclass>は現在のところ<name>で代用している。

3. 全ての子および兄弟に対して同様の処理を再帰的に行なう。
4. 全ての子を生成したら、トップの WInfo の name, type を参照し、

```
RealizeWidget <トップの name>
MainLoop あるいは AppMainLoop appcon
```

を出力する。

なお、GUI の表示は以上のようにして XTSS プログラムを生成し、そのファイルに対して、

```
xtc -f ファイル名
```

を実行することにより実現している。

図 6に図 5のツリーから生成された XTSS プログラムとその表示結果を示す。

```
Initialize toplevel Toplevel
CreateManagedWidget Box boxWidgetClass toplevel width 300
CreateManagedWidget MenuButton menuButtonWidgetClass Box
CreatePopupShell menu simpleMenuWidgetClass MenuButton
CreateManagedWidget SmeBSB smeBSBObjectClass menu
CreateManagedWidget SmeBSB smeBSBObjectClass menu
CreateManagedWidget MenuButton menuButtonWidgetClass Box
CreatePopupShell menu simpleMenuWidgetClass MenuButton
CreateManagedWidget SmeBSB smeBSBObjectClass menu
CreateManagedWidget SmeBSB smeBSBObjectClass menu
CreateManagedWidget Command commandWidgetClass Box
RealizeWidget toplevel
MainLoop
```

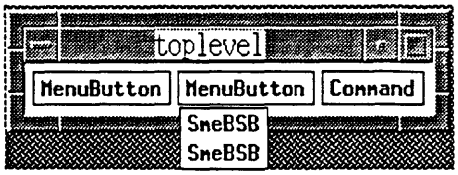

図 6: 生成された XTSS プログラムと表示結果

4 Analyzer

4.1 概要

Analyzer は、既存の GUI アプリケーションの GUI 部をそのまま、あるいは多少改良して利用したい場合などに使用する。このような場合に、利用したい GUI アプリケーションを起動しておき、そのアプリケーションをマウスで指定することによりウィジェット情報を取得し、GUI ビルダーに反映させることができるものが望ましい。これを実現するためには次のような動作が必要となる。

1. 対象となるGUIアプリケーションにAnalyzer側から、ウィジェット情報の取得を依頼するイベントを送る。
2. GUIアプリケーションはそのイベントを理解し対応する処理を行ない、Analyzerにウィジェット情報を送る。

すなわち、他のGUIアプリケーションのウィジェット情報を取り出すには、GUIアプリケーションがイベントを理解できるようにアプリケーションにイベントハンドラを登録しておかなければならないのである。しかし一般に目的のアプリケーションは不特定であるので、その都度登録することは非常に困難である。ところが、Athenaウィジェットには editres(X Toolkit アプリケーション用ダイナミックリソースエディタ) 用のイベントハンドラが既に登録されており、editres はこの editres 用プロトコルを用いてクライアント間通信を行ない、ウィジェット情報の取得を行なっている。このためAnalyzerは editres を利用し、editres のコマンドメニューに、XTSSプログラム生成用のメニューを追加するという形で作成した。図7がAnalyzerのトップウィンドウである。

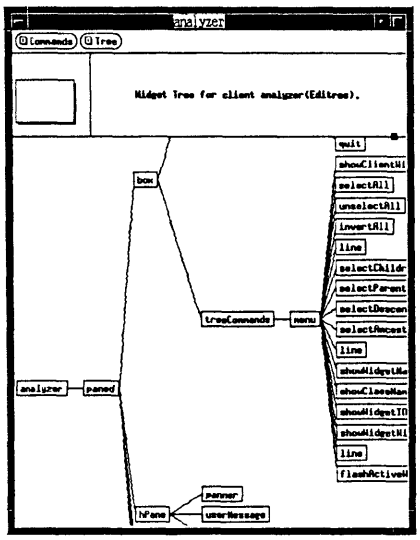

図7: Analyzer

操作方法について以下で簡単に説明する。

Commandメニューの項目のGet Widget Treeを選択すると、マウスポインタカーソルが十字型に変わる。調べたいGUIアプリケーションのウィンドウをクリックすると、そのアプリケーションが editres プロトコルを実装している場合は、クライアントのウィジェットツリーが表示される。一度ウィジェットツリーが得られれば、他のメニュー項目を選択してクラス名やウィンドウIDのツリーを見たり、リソースの設定などを行なうことができる。これらは、全て editres 本来の機能である。

表示されたウィジェットツリーをXTSSプログラムに変換するには、Commandメニューの Dump XTSS source to a File を選択する。ファイル名を問い合わせるダイアログボックスがポップアップされるので、ファイル名を入力すれば、そこへXTSSプログラムが出力される。これの部分が付加した機能であるが、非常に簡単にXTSSプログラムを生成できることが分かるであろう。ただし、Athenaウィジェットを使用していないGUIアプリケーションは、通常 editres プロトコルが実装されていないので、全てのGUIアプリケーションには対応できないという問題がある。

4.2 XTSSプログラムの生成

editresはGet Widget Treeを実行するとWNode構造体にウィジェット情報を格納するが、Analyzerはこの WNodeに格納されている情報を基にXTSSプログラムの生成を行なっている。

WNode構造体は、ウィジェットの名前を示すname、ウィジェットのクラス名を示すclass、ウィジェットのIDを示すid、ウィンドウのIDを示すwindow、親ウィジェットのWNodeを指すparent、子ウィジェットのWNodeの配列を指すchildren、その要素数を示すnum_childという要素を持つ。

XTSSプログラムの生成の手順は、

1. ウィジェット階層のトップのWNodeのnameとclassを取り出し、

   ```
   Initialize <name> <class>
   ```

   を出力する。
2. 子の WNode の class が composite クラスに属していれば CreatePopupShell、そうでなければ CreateManagedWidget 命令を生成する。

   (a) classの先頭文字は大文字になっているので小文字に変換する。

   (b) このclassからウィジェットであるかオブジェクトであるかを判断し、ウィジェットであれば <class>WidgetClass、オブジェクトであれば<class>ObjectClassとする。

   以上のように変換した後、

   ```
   CreateManagedWidget/CreatePopupShell <name> <変換後の class> <parent>
   ```

   と出力する。ただし、<parent>は親のWNodeの名前である。
3. 全ての子に対して同様の処理を再帰的に行なう。
4. 全ての子を生成したら、

   ```
   RealizeWidget <トップの name>
   MainLoop
   ```

   を出力する。

5 おわりに

本研究では、GUIの構築を更に容易にする目的から、テキストレベルでのGUIプログラミング機能を提供するXTSSの支援ツールとして、視覚的操作でGUIの構築を行なえるGUIビルダー XTSS Builderを提供した。本GUIビルダーは、ツリーからツリーへの統一的な操作によって視覚的にGUI部を構築する機能、GUIの構築中にいつでもGUIを生成して確認できる機能、設定可能なリソースの一覧を表示する機能、既存のアプリケーションのGUIを利用する機能等によりGUIの構築を大幅に簡単化した。

更に充実した支援を行なえるように、今後の課題として以下のようなものが挙げられる。

1. XTSSプログラムの解析の充実
   ウィジェット生成用命令にしか対応できていない。XTSSプログラムを直接記述した場合には、生成命令以外の命令が含まれていることがあるので、それらの命令を解析し、修正を援助し、そして出力できる機能を完備する必要がある。

2. ツリーの親子関係操作機能の充実
   基本操作として、ノードの追加・訂正、ノードとその子ノードの一括削除の機能はあるが、あるノードだけを削除する機能、ノードを移動する機能なども必要である。
3. 誤り検出および訂正機能
   ウィジェットの親子関係を定義するときに子を持てないクラスのウィジェットに子を繋いだり、あるいは、リソース値として不適切な値を指定してしまうことがある。現在のところこのようなエラーの判定は全てXTSSに任せているが、XTSS Builder自体で適宜エラーメッセージを出力したり、エラーを自動的に訂正する機能が必要である。

参考文献

[1] 久野靖: 新しいプログラマ・インタフェースの利用, 情報処理, **Vol.30**, No.4, pp.396-405(1989)

[2] Open Software Foundation: *OSF/Motif Programmer's Guide, Revision 1.1*, Open Software Foundation, inc. (1990)

[3] Scheifler,R.W., Oren,L., etc.: *CLX Programmer's Reference*, Texas Instruments Inc. (1989)

[4] Mayer,N.P.: The WINTERP Widget INTERPreter - An Application Prototyping and Extension Environment for OSF/Motif, *Proceedings X Into The Feature, The European X Users Group Autumn Conference 1990*, pp.33-55 (1990)

[5] Ousterhout, J. K.: Tcl: An Embeddable Command Language, *Proceedings of the 1990 Winter USENIX Conference* (1990)

[6] Ousterhout,J.K.: An X11 Toolkit Based on the Tcl Language, *Proceedings of the 1991 Winter USENIX Conference* (1991)

[7] Neumann,G., Nusser,S.: *Wafe - A Tcl Interface to the X Toolkit or A Graphical Frontend for Applications in Various Programming Languages* (1992)

[8] 高濱徹行, 中谷洋一, 小倉久和, 中村正郎: 応用プログラム群を統合するウィンドウ型ユーザインタフェースシステム：ＵＡＩ／Ｘ, 電子情報通信学会論文誌 *D-I*, **Vol.J75-D-I**, No.7, pp.479-487 (1992)

[9] 高濱徹行: ＸツールキットサービスプロトコルによるＸウィンドウプログラミング, 福井大学情報処理センター *NETWORK*, **Vol.7**, No.1, pp.19-44 (1993)