ソフトウェア工学　139－2
(2002. 10. 10)

階層的部品定義に基づく組込用ＵＩ設計ツール

小中　裕喜　　中川　隆志　　津高　新一郎　　西川　泰浩　　有田　英一

三菱電機株式会社 先端技術総合研究所

Email: {Konaka.Hiroki, Nakagawa.Takashi, Tsudaka.Shinichiro, Nishikawa.Yasuhiro, Arita.Hidekazu}@wrc.melco.co.jp

概要：携帯電話などの組込機器における、画面の切替を伴うユーザインタフェースの設計再利用性の向上、設計開発コスト低減を実現する設計ツールを開発した。複数の画面状態とそれらの間の状態遷移、及び各状態におけるＵＩ部品のレイアウトを定義可能な、ステートチャートオブジェクトという概念に基づき、個々のカスタム表示部品からアプリケーション、機器本体に至るまで、すべて同一の枠組みで設計し、組み合わせてシミュレートすることが可能である。またシミュレーション結果からの表示操作仕様書作成や設計データからのコード生成をサポートする。

UI Design Tool for Embedded Systems based on Hierarchical Component Definition

Hiroki Konaka, Takashi Nakagawa, Shin'ichiro Tsudaka, Yasuhiro Nishikawa, Hidekazu Arita

Advanced Technology R&D Center, Mitsubishi Electric Corp.

Abstract: The authors have developed a UI design tool to improve productivity and quality of user interface that involves scene transitions. It seamlessly supports UI development from rough flow design to S/W development for embedded systems such as mobile phones. Based on the notion of *State Chart Object* that allows the hierarchical definition of UI components, the tool enables the user to design small indicators to applications and device bodies within a single framework while checking the usability of designed UI via immediate simulation. It also supports the edition of operational specification from simulation results as well as the generation of target code in C++ from designed data.

１．はじめに

携帯電話をはじめとする組込機器のユーザインタフェース S/W は、他社製品との差別化要因であり、その多機能化や仕様変更に伴う設計開発コストの増大、品質低下が大きな課題となっている[1]。そこで UI 設計のラピッドプロトタイピングの実現、設計再利用性の向上、設計データからのコード自動生成等により、上記課題を解決するツールを開発した。

２．ステートチャートオブジェクト

本ツールは、複数の画面状態とそれらの間の状態遷移、及び各状態における UI 部品のレイアウトを定義可能な、ステートチャートオブジェクト(SCO: State Chart Object)という概念をベースとしている。図１に示すように、SCO の各状態にはボタンやラベルといった既製の UI 部品だけでなく他の SCO を配置することも可能である。これにより、画面を構成するカスタム表示部品からアプリケーション、機器本体に至るまで、様々な階層の設計物を部品化し、組み合わせることを可能にしている。

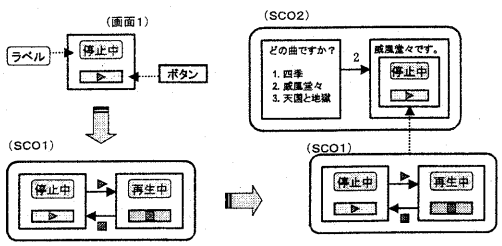

図１ ステートチャートオブジェクト(SCO)

3．ツールの構成

図2に示すように、本ツールは以下のモジュールから構成される。

(1) Chart Editor : SCO の各状態における、状態遷移などのイベントハンドリングの設計を行う。データベースアクセスなどの非 UI 部との連携や、それに伴う動的な UI 部品のプロパティ設定などを可能とするため、必要に応じて Java ™ のサブセットをベースとした簡易スクリプトを記述することも可能である。

(2) Layout Editor : SCO の各画面における UI 部品のレイアウト及びプロパティ設定を行う。ある UI 部品のプロパティが他の UI 部品のプロパティやプロパティデータベースの値を自動的に参照するように設定することも可能であり、SCO による部品化や、言語などに応じた表示の切替を容易にしている。

(3) Simulator : Chart Editor 及び Layout Editor で設計した SCO の動作をシミュレートし、ユーザによる動作確認を可能とする。シミュレーション時のユーザ操作をロギングしてリプレイすることも可能である。

(4) Flow Editor : シミュレーション結果からの表示操作仕様書作成をサポートする。またリプレイによって表示操作仕様の画面イメージを自動的に更新することも可能である。

(5) Code Generator : 設計データから C++ソースコードを出力する。UI 部品のレイアウト情報だけでなく、SCO の状態遷移やスクリプト記述を含むイベントハンドリングも生成対象としている。本ツールの提供するランタイムと組み合わせ、実機上で動作させる。移植性を考慮して、コード生成系及びランタイムはそれぞれ機器独立部と依存部に分離されている。

4．UI 設計開発の流れ

図2に本ツールを利用した UI 設計開発の流れを合わせて示す。

ラフフロー設計の段階では、シミュレート時に簡易操作で生成可能な仮想的イベントや、暫定的な表示イメージなどを用いて、紙芝居的な設計を行う。そして暫定的な画面遷移やレイアウト設計を、より具体的なイベントハンドリングや UI 部品で置き換えながら Simulator で確認する過程を繰り返し、順次設計を詳細化する。

SCO は既存の UI 部品にはないルックアンドフィールをもつカスタム表示部品の設計を

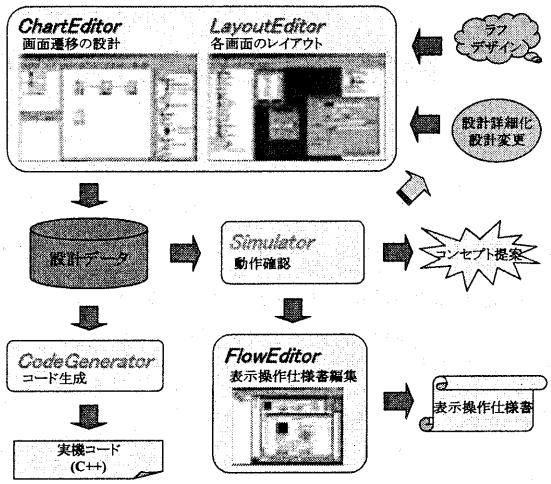

図2　ツールの構成と UI 設計開発の流れ

容易にする。また「はい／いいえ確認」や「暗証番号入力」などといった、複数のアプリケーションで共通に用いられる一連の操作シーンを部品化することも可能とする。このような SCO の設計／修正は、その SCO を利用するすべての場面に反映されるため、アプリケーション間の操作性統一を図ることが容易となる。

実機用コードは設計データから直接生成される。また表示操作仕様書は、シミュレートされた画面フローを機能ごとにまとめて作成するが、画面イメージはリプレイによる自動更新が可能であり、いずれも設計変更に伴うオーバヘッドの軽減に寄与している。

5．おわりに

画面切替を伴う UI の生産性向上を目的とした UI 設計ツールの基本概念と構成、そしてツールを用いた UI 設計開発の流れについて説明した。

本ツールを用いれば、デザイナ、開発者、顧客の間で「動く仕様」を共有しながら、簡易な上流設計から実機レベルの詳細設計まで連続的に作り込むことが可能である。また SCO による UI 設計の部品化や、コード生成機能のサポートなどにより、設計データの効率的な活用が可能となっている。

参考文献

[1] 今井, 三宅: ケータイ・ソフト開発 人海戦術の破綻, 日経エレクトロニクス, 2001年5月7日号, No.795, pp.117-137, 2001.