7H－6

シナリオ変更の容易化を目指した GUIアプリケーション開発環境GUI-SIDERの開発*

清水 奨　荒川則泰†
日本電信電話株式会社　ソフトウェア研究所‡

1　はじめに

GUIアプリケーションでは、機能の利用手順(以下、シナリオ)への変更要求が多い。提供する機能が完成した後でも、シナリオの変更は多発する。従来、シナリオに関するプログラムコードは、他の機能と混在して記述されていたために、その変更には多くの時間が必要であった。シナリオ変更の効率化でGUIアプリケーション開発期間の大幅な短縮が期待される。

シナリオの形式的記述による正当性確認から、プログラム生成までの自動化を検討している[2]。本稿では、X Window上のGUIアプリケーションを対象とした開発環境 GUI-SIDER (GUI Scenario-based Interaction Design Environment) のシナリオ単位の設計法とその実装について述べる。

2　シナリオを単位とする設計法

機能の利用手順をシナリオと呼ぶ。例えば「終了する」という機能に関するシナリオとして以下がある。

1. プルダウンメニューからQUITを選ぶ。
2. 「終了しますか」とのダイアログウィンドウが現れる。
3. ダイアログウィンドウ内でOKを選ぶと、終了する。

「終了」では、ダイアログでの確認の有無や、ファイルセーブ確認の有無等様々なシナリオが存在する。実際のアプリケーションでは、シナリオ全体数は数百に及ぶ。

シナリオは、ユーザとGUI部品間のインタラクション時系列としてモデル化できる。時系列の表現手段としてメッセージシーケンス図(図1)がある。シナリオをMSCで記述することで、既存シナリオとの整合性確認、プログラム生成等のMSC処理既存技術[1]の流用が可能になる。また、機能処理関数群をプロセスプログラムのサブルーチンとして記述するモデル化により、シナリオと機能は明確に分離される。終了シナリオのMSC記述である図1では、Quit()が機能にあたる。

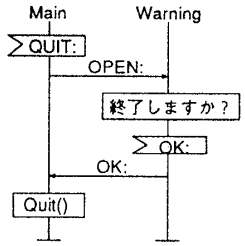

図 1: Message Sequence Chart

3　X Window上での実装

3.1　並行プロセスプログラム実行制御

シナリオのMSC記述では、個々のダイアログウィンドウをプロセスでモデル化する。各プロセスを内部状態を持つサブルーチンとしてプログラム生成し、以下のような機構で実行する。

1. Xのイベントを対応するプロセスへのメッセージ送信に変換。送信されたメッセージはメッセージキューに入る。
2. メッセージキューに信号がある間は、プロセスにメッセージをディスパッチ。
3. メッセージキューが空になったらXのイベントループに戻る。

3.2　イベントとのI/F

Xのイベントでシナリオを駆動するには、コールバック関数(イベントハンドラを含む)にメッセージキューへの送信記述を行なう必要がある。

シナリオの観点から、コールバック関数は次の2種類に分類できる。

シナリオ依存型： シナリオ上でイベントとして現れるもの。それ自身では機能が完結しない。プルダウンメニューのボタン選択イベントによりシナリオが駆動され、ダイアログを出すなどの状態へ遷移する時、このボタンのコールバック関数はシナリオ依存型である。

---

*GUI-SIDER:a Scenario-based design environment for easy modifications

†Susumu Shimizu and Noriyasu Arakawa
‡NTT Software Laboratories, 5-314B, 3-9-11 Midori-cho Musashino-city TOKYO 180 JAPAN

独立型： それ自身で機能が完結しており、間接的にシナリオに影響を及ぼすもの。例えばラジオボタンによる選択はそれ自身で状態変数を変えるという機能が完結する。

このうち、シナリオ依存型のコールバック関数にシグナルを生成させ、プロセスプログラムとイベントとのI/Fとして用いる。この実装法により、コールバック関数をライブラリ化できる。ほとんどのイベントに対して、コールバック関数はシグナルとデータをキューイングするだけでよい。ライブラリは十数種類の関数からなり、それぞれは数行〜40行程度の規模である。

一方、独立型のコールバック関数はオーバーヘッドを避けるため、シナリオ記述とは独立に実装する。これは従来通りの実装法である。

まとめると、イベント処理の流れは図2となる。

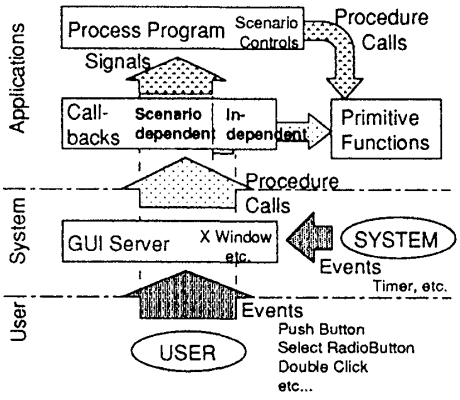

図 2: Event flow

図中、Primitive Functions が個々の機能のエントリである。Callbacks から直接 Primitive を呼ぶのが独立型 (Independent) のコールバック関数で、それ以外 (Scenario dependent) は Signal を生成して Process Program に制御を任せる。

4 実験による評価

本設計法に基づく実装機構 GUI-SIDER(図3) を実装し、評価実験を行なった。図中 Signal Dispatch Libs 及び Signal Generator Callbacks はそれぞれ前章で議論したライブラリである。また SDE (Systems Design Environment) は、メッセージシーケンス図の集合を合成・検証するシステムである [1]。GUI 画面設計には市販の UI ビルダーを利用した。

実験対象はスケジュール管理アプリケーションであり、そのコード量は画面設定が 1.2k 行、シナリオ部が 1.6k 行、その他が 1.2k 行の合計 4k 行である。

シナリオ部コード (1.6k 行) を 25 枚の MSC から自動生成することにより、シナリオ変更 (シナリオ5枚

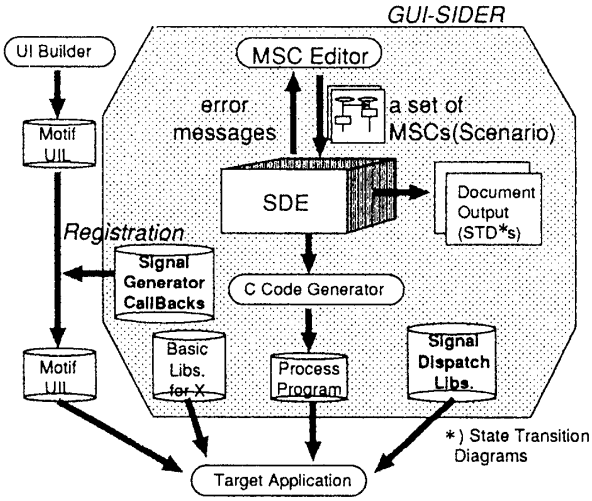

図 3: The Structure of GUI-SIDER

追加, コード量100行増加) にかかる作業時間を従来比にしておおむね2割削減できた。

5 まとめ

従来分散していたシナリオ記述を集約し、変更の効率化を図るため、シナリオ単位の設計を行なう設計法を述べた。本設計法によりシナリオ設計〜評価〜修正のターンアラウンドが短くなるため、ユーザ満足度の高いシナリオ設計が行える。

シナリオ実行ライブラリを構築し、既存の MSC 処理環境と組み合わせることで X Window 上の GUI アプリケーション設計環境 GUI-SIDER を実現。シナリオ変更作業時間の 2 割削減を確認した。

今後はより多くの実装実験を基に MSC 記述の有効範囲を検討し、定量的な評価を行なう予定である。

謝辞

最後に、本研究を進めるにあたり議論して頂いた NTT ソフトウェア研究所サービスソフトウェア方式研究グループ、市川 晴久グループリーダ及び加藤 順 主任研究員に感謝します。

参考文献

[1] H.Ichikawa, M.Itoh, and others, : SDE: Incremental Specification and Development of Communications Software, *IEEE Transactions on Computers*, **40** No. 4, pp. 553-561 (1991).

[2] 清水, 荒川: インタラクション変更の容易化をめざした GUI アプリケーション設計法の提案, インタラクティブシステムとソフトウェア I (WISS '93), 近代科学社 レクチャーノート (1994), 日本ソフトウェア科学会インタラクティブシステムとソフトウェア研究会 編.