ソフトウェア自動合成シェル - EasySOFTEX - (1)
－ 特定ドメイン向けアプリケーションジェネレータの開発プロセスと支援環境 －

3W－4

友部 実、徳岡宏樹、山之内 徹
NEC C&C メディア研究所
e-mail: {tomobe,tokuoka,yamanouchi}@ccm.cl.nec.co.jp

1 はじめに

アプリケーションプログラムの開発の効率化や高品質化を目的として、特定のアプリケーションドメイン専用のアプリケーションプログラムジェネレータ (AP ジェネレータ) を構築してアプリケーション開発を支援する手法がある [1]。ドメイン個別の AP ジェネレータの構築により、アプリケーションプログラムの開発効率並びに品質を大幅に向上することが可能になる一方、次の課題として AP ジェネレータの開発効率の向上や品質の確保、またカスタマイズの容易さや保守性の向上が大きな課題となる。

このような問題を解決するため、我々は AP ジェネレータ構築の為の問題向け設計言語 (Domain Specific Language、以下 DSL) を提供することにより、AP ジェネレータの開発を支援するアプローチを取った。ソフトウェア自動合成シェル EasySOFTEX は AP ジェネレータ構築を対象とする DSL であり、AP ジェネレータ構築に必要となる字句 / 構文解析、抽象構文木上での変換、コード生成といった機能を言語レベルで提供する。

本稿では、まず AP ジェネレータの開発プロセスについて述べる。次に AP ジェネレータ構築に固有の課題をあげ、これらを解決する為に EasySOFTEX の提供する機能について概説する。最後に、本アプローチの効果と今後の課題について述べる。

2 AP ジェネレータの開発プロセス

AP ジェネレータの開発経験 [2, 3] を通じて、AP ジェネレータ開発は以下のプロセスで行われることが判った。

リファレンスモデルの設計 AP ジェネレータの対象とする領域を決定し、AP ジェネレータの生成するプログラムの構造を決定する。

ジェネレータ生成コードの設計 前述の設計結果に基づいて、AP ジェネレータの生成する対象とするドメインのプラットフォーム (PF) 上で動作可能なコードを記述する。この段階で、実行 / 評価を行い生成コードに求められる必要な要件 (速度 / コードサイズ / メモリ使用量等) を満足するかどうか評価する。

入力仕様記述方法の設計 AP ジェネレータの入力となる入力仕様の記述方法の設計を行う。入力仕様記述は実装方法に依存せず、対象とするドメインのアプリケーションの仕様を簡単に記述できる形式とする。

ジェネレータの実装 これらの AP ジェネレータの設計情報を元に AP ジェネレータの実装を行う。

これらのプロセスはプロトタイピングによって行われる。これは AP ジェネレータの開発が適用プロジェクトに先行して行われる為、AP ジェネレータ開発時には PF の要件が決定していないケースが多い為である。この為プロトタイピングのサイクルを短縮し早期に AP ジェネレータ仕様を決定することが重要な課題であった。また AP ジェネレータ適用後にも頻繁に入力仕様記述形式や、ジェネレータ生成コードの仕様変更が発生する。大規模プロジェクトではこの変更が全てのジェネレータ適用 AP に及ぶ為、変更を迅速に AP ジェネレータに反映することが必要とされた。

3 ソフトウェア自動合成シェル EasySOFTEX

EasySOFTEX では AP ジェネレータ構築の為の DSL を提供することにより、AP ジェネレータの構築を支援する。我々は前述の開発プロセスから AP ジェネレータを構築する際に特徴的な以下の課題を考慮し、EasySOFTEX の設計を行った。

1. AP ジェネレータ開発者は AP ジェネレータの構築に必要な既存ツールを使いこなせるとは限らない。
2. 生成コードへの要件が頻繁に変更される。
3. AP ジェネレータ構築に高い生産性 / 品質が求められる。

1. に関して、AP ジェネレータの開発はジェネレータの対象とするドメインを熟知したソフトウェア開発者が行うことが必要である。このような開発者が yacc/lex 等 AP ジェネレータ構築に必要なツールの知識も持っているケースはまれである。またこれらのツールを用いても抽象構文木の生成や構文木上での変換といった AP ジェネレータ構築時に多用する機能を持たないという課題がある。2. に関して、AP ジェネレータの生成するコードは保守性の向上、カスタマイズの容易さといった観点から、AP ジェネレータの実装に依存しない形で AP ジェネレータ中に記述されていることが望ましい。3. に関して、AP ジェネレータの開発を迅速に行うことはプロトタイピングのサイクルの

Minoru TOMOBE,Hiroki TOKUOKA,Toru YAMANOUCHI
C&C Media Research Laboratories, NEC Corporation

短縮に必須である。このため実装言語は構文木上の変換を効率良く行うことができ、メモリ管理等、APジェネレータ機能に直接関連のない処理が極力不要な言語処理系であることが望ましい。

これらの課題を解決する為、EasySOFTEXは関数型言語をベースにAPジェネレータ特有の字句構文解析機能、生成コード記述機能を付加した言語とした。

EasySOFTEXの提供する字句構文解析機能はyacc/lex等と異なり、入力仕様パーザに簡単な仕様記述を与えるだけで入力仕様の抽象構文木の生成が行え、既存ツールに比べて簡単にパーザを構築できる。ジェネレータ生成コードはコードテンプレートという形式で記述され、生成コードをAPジェネレータ中に直接記述することができる。これにより生成コードの変更が簡単になり、ジェネレータの保守性の向上や、容易なカスタマイズを可能にする。また高品質なAPジェネレータを高い生産性で構築する為に、変換規則の記述は関数型言語を用いた。これにより、型推論を用いた変換規則の検査による品質の向上や、データ構成子を用いた抽象構文木の表現、パターンマッチによる抽象構文木上での変換の柔軟な記述が可能になった。

図1にEasySOFTEXの入出力を示す。APジェネレータを構築する為に、以下の3つの仕様をEasySOFTEXの提供する言語を用いて定義する。

入力仕様パーザ定義 APジェネレータの入力仕様の字句ならびに文法仕様を定義する。文法定義はLL文法に基づくBNFによって定義され、EasySOFTEXの提供するパーザジェネレータが入力仕様パーザを自動生成する。この文法仕様には抽象構文木を生成する為の情報を注釈として付加することができ、この記述から自動的に抽象構文木を表現する為のデータ型と抽象構文木を生成するアクションが自動的に生成される為、yacc等に比較して簡単にパーザを構築することが可能である。

変換規則定義 入力仕様パーザで得られた抽象構文木を変換し、後述するコードテンプレート中に埋め込む処理を記述する変換規則を定義する。変換規則はStandard ML[5]等の関数型言語と同等の記述力を持つ。

コードテンプレート定義 APジェネレータが生成するコードをコードテンプレート記述言語を用いて記述する。APジェネレータ開発者は実際にPF上で動作するコードを記述し、このコードを抽象化する形でコードテンプレートを定義する。コードテンプレートの詳細は[4]で報告する。

入力仕様パーザ、コードテンプレートはそれぞれパーザジェネレータ / コードテンプレートトランスレータにより内部表現としてEasySOFTEXの変換規則で実装される。変換規則コンパイラはこれらが出力した変換規則と、APジェネレータの変換の過程を定義したユーザ定義の変換規則を直接実行可能なターゲット環境の機械語にコンパイルする。コンパイルされたコードはファイル入出力等を行うランタイムライブラリとリンクされ、APジェネレータとして提供される。

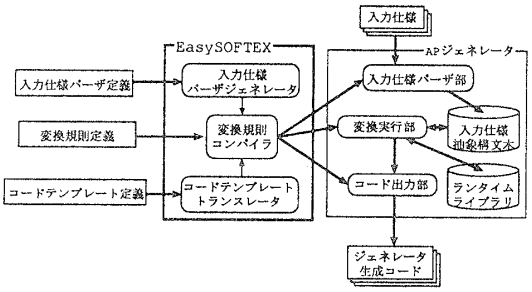

図 1: EasySOFTEXの入出力

4 評価

現在、EasySOFTEXを用いたAPジェネレータの開発が行われている。従来システム[1]に比較して、関数型言語をベースにしたコンパイラを提供している為、型推論による変換規則の静的な検査による品質の向上や、変換速度の向上が確認されている。またコードテンプレート記述を用いることにより、APジェネレータの変換過程を実現する変換規則部分と、ジェネレータの生成コード記述部分を分離することが可能になり、保守 / カスタマイズが行い易いという評価を得ている。

5 おわりに

APジェネレータの構築支援を行う為のDSLに求められる機能の分析を行った。その結果から、簡便な記述からパーザを構築できるパーザジェネレータを持ち、コードテンプレートによる保守性・カスタマイズ性の高いジェネレータ生成コード記述が可能な特長を持つ、関数型言語をベースしたAPジェネレータ向けDSL EasySOFTEXを構築し、その概要について述べた。今後、EasySOFTEXを用いた大規模プロジェクトにおけるAPジェネレータ開発を通じ、本稿で提案した機能の定量的な評価を行いたい。

参考文献

[1] Yamanouchi,T.,Sato, A.,Tomobe M.,Takeuchi, H., Takamura, J. and Watanabe, M. : Software Syntehsis Shell SOFTEX/S,7th KBSE Conf. , 1992.

[2] A.Sato,M.Tomobe,T.Yamanouchi,M.Watanabe, and M.Hijikata: "Domain-Oriented Software Process Reengineering with Software Synthesis Shell SOFTEX/S" ,10th KBSE Conf., 1995

[3] 友部、三木、中島: "問題向け設計言語によるアプリケーション開発支援環境の構築", 日本ソフトウェア科学会第14回大会, D11-3,1997

[4] 徳岡、友部、山之内 : "ソフトウェア自動合成シェル - EasySOFTEX - (2) - アプリケーションジェネレータ設計のための問題向け設計言語 -", 情報処理学会第59回全国大会,3W-05,1999

[5] L.C.Paulson : "ML for the working programmer", Cambridge University Press,ISBN 0-521-56543-X,1996