ＣＡＳＥツール統合化へのアプローチ(1)
4S−1 ―設計とプログラミングの統合に関する考察―

阿部　弘彰　　坂本　康広*　　高橋　幸子*　　深尾　至**　　西山　好雄*

*富士通㈱　　**富士通北海道通信システム㈱

1．はじめに

通信ソフトウェアの開発に於ける仕様の抽象化レベルの個人差や記述形式の不均質性およびこれにより派生する作業間のギャップを埋める手段としてＭＲＶ(Multi-layered Refinement by Views)アプローチ[1]を提案している。これに基づいてタスク構成からモジュールの詳細プログラミングまでの設計・製造工程を対象とする２つの支援ツールＹＤＳ(YACⅡ oriented Design System)とＹＰＳ(YACⅡ Programming System)を通信向けの実プロジェクトに適用中である。

以下では，ＹＤＳとＹＰＳを統合する上での問題点を分析し，解決へのアプローチについて考察する。

2．ＹＤＳとＹＰＳの連携

(1)概要

ＹＰＳはヘッダ部や引数宣言部，手続き部などの制御構文の図形表記と日本語によるステートメント記述を特徴とする構造化チャート（ＹＡＣⅡ）プログラミングツールであり，専用コンパイラにより実行可能コードに変換される。

一方，ＹＤＳは，設計の対象をＣＰＵ全体の持つ機能からタスク構成レベル，モジュール構成レベル，モジュールレベルへと段階的詳細化の進展に合わせて推移させていき，それによってプログラム部品群とそれらの間の関係を明らかにする過程を支援する。
各設計ドキュメントの作成・更新に対応してＹＤＳオブジェクトが作成されされる。ＹＤＳオブジェクトは，モジュール，共通データなどのエンティティとそれらの関連を持ち、各エンティティは，ＹＡＣⅡ記述によるプログラム部品（以降ＹＰＳオブジェクトと呼ぶ）に結びついている。
ＹＤＳは，このような情報に基づいて設計作業の連続性を維持するための各種機能を提供している。

最終的なＹＰＳオブジェクトは，ＹＰＳソースとして変換・編集されプログラミング工程に引き継がれる。

図１に現状のＹＤＳとＹＰＳによる連携の概要を示す。

(2)現状の問題点

以上の方式によってＹＤＳ，ＹＰＳの適用を行ってきたが，次のような運用上の問題が生じている。

①設計とプログラミングのギャップ
ＹＤＳでは，共通データや状態遷移モジュールなどにＹＰＳオブジェクトを対応付けているが，プログラミングに移行する段階で，プログラムテキストファイルという即物的な管理単位に落ちてしまう。

これは，従来のプログラミングシステムの常識に適合させた結果であるが，設計者とプログラマの間にヘッダファイルを介した共通データとモジュールの参照関係の確認など，余分な対応付けの作業を強いている。

②ＹＡＣⅡ記述の重複
①の理由から設計とプログラミングでＹＡＣⅡの記述を二重に管理することとなり，誤ったプログラム修正などに伴う一貫性を損ねる原因となる。

これらの問題は，『このタスクをコンパイルしたい』とか『この共通データをアクセスする全てのモジュールを再コンパイルしたい』といった統合ツールに期待されるユーザインタフェースを実現する上での課題でもある。

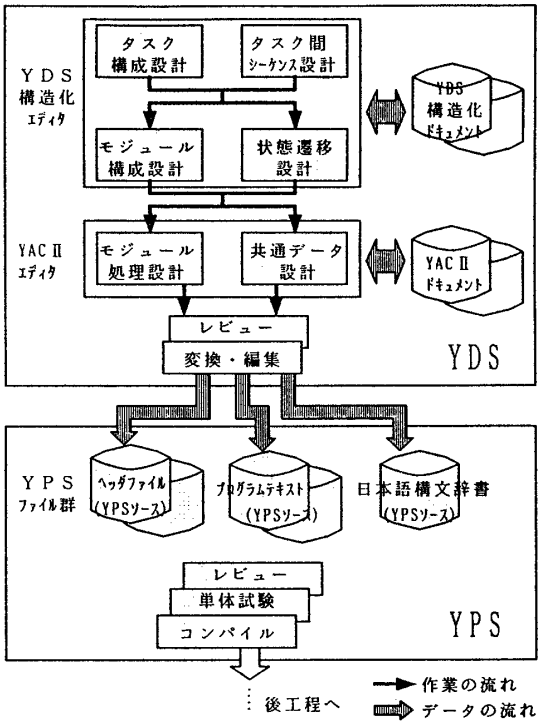

図１．現状のＹＤＳとＹＰＳの連携

3．設計とプログラミングの統合化

(1)考え方

以上から，一連のプログラミングの作業を独立のものでなく，設計作業の一環と捕え，設計レベルの概念によるプログラミングの実現を検討した。
本方式によるＹＤＳ／ＹＰＳ連携の様子を図２に示す。

ここでは，プログラミング部品に対応付けられたＹＰＳオブジェクト自身をソーステキストとしても扱い，プログラミングの作業を全てこれらを対象に行う。即ち，ＹＰＳにおける従来の即物的なプログラミングのオブジェクトやそれを扱うツール群を背景に隠蔽している。

An Approach for Integration of CASE Tools(1)
Hiroaki ABE,　Yasuhiro SAKAMOTO,　Yukiko TAKAHASHI
Itaru FUKAO,　and Yoshio NISHIYAMA
FUJITSU LIMITED
FUJITSU HOKKAIDO COMMUNICATION SYSTEMS LIMITED

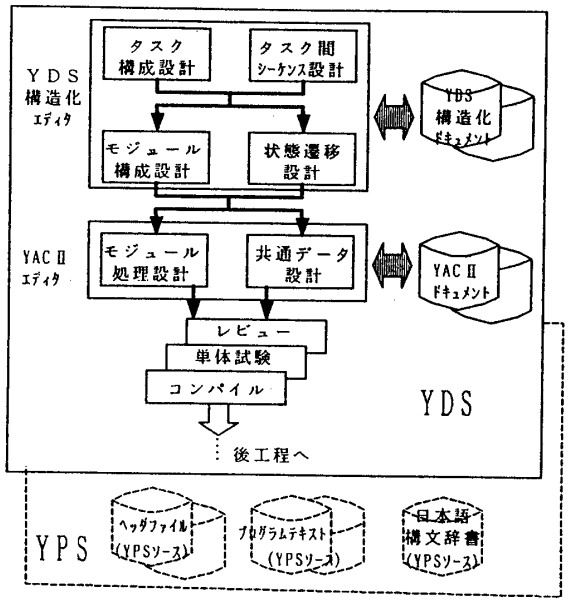

図2．本方式によるＹＤＳとＹＰＳの連携

⑵処理過程

ここでは，図３に示すモジュール，共通データおよびそれらの間の"参照"関連のみからなる仮想的なＹＤＳオブジェクトの例を用いて『共通データＤ１を詳細化してコンパイルしたい。』という要求を実現するまでの処理過程について考える。ここではＹＰＳオブジェクトに対するプログラミングが全て完了しており，データ構造エンティティとの関連に関する取扱は自動的に追随するものと仮定する。

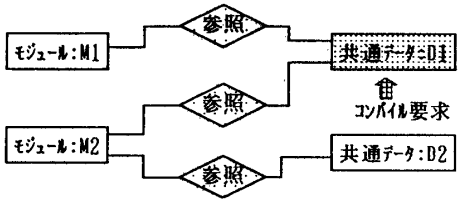

図３．モジュールと共通データの関連例

①ＹＰＳソースの生成までの手順

コンパイル実現にあたってあらかじめ必要となるＹＰＳソースの作成までの手順を以下に示す。ここではモジュールＭ１に対応するＹＰＳソースの生成を考える。

[1]抽出モジュールの共通データ参照関係を抽出
　抽出結果のモジュールエンティティＭ１についてそれぞれ共通データ参照関連を抽出する（Ｍ１はＤ１を参照している）。
[2]ＹＰＳオブジェクト相互の結合による関連の反映
　ＹＰＳモジュール部品としてのＭ１のＹＡＣⅡ記述を取り出し，参照関連である共通データ（Ｄ１）の外部参照宣言構文を該当構文個所に挿入する。
[3]ＹＰＳソースファイルの格納
　[2]の結果としての共通データ外部参照宣言を含むＹＡＣⅡ記述をＹＰＳソースファイルとして格納する。

このような手順でＹＤＳオブジェクトが保持するモジュール（Ｍ１）と"参照"関連を反映した結果としてのＹＰＳソースが完成できる。

②コンパイルまでの手順

共通データＤ１に対してコンパイル要求が届いた場合次のような手順でコンパイルが実現される。

[1]ＹＤＳオブジェクトからの参照関連抽出
　Ｄ１と"参照"関連を有する全てのモジュールエンティティを抽出（Ｍ１，Ｍ２）する。
[2]モジュールエンティティのＹＰＳソースの生成
　①に従って抽出したＭ１，Ｍ２のＹＰＳソースファイルを生成する。
[3]コンパイルの実施と結果としての実行形式の格納
　Ｍ１，Ｍ２に対し，コンパイルを起動し実行形式を該当ファイルとして格納する。

⑶前提となる機能要件

本方式とその処理過程を実現する上で，前提となる機能要件をまとめると以下のものがある。

[1]エンティティと関連の抽出機能
　ＹＤＳ／ＹＰＳオブジェクトからそれに含まれているエンティティと関連の抽出のための機能が必要となる。
[2]オブジェクトのロケーション管理機能
　ＹＤＳ／ＹＰＳオブジェクトやコンパイル結果の実行形式を格納するロケーションの管理が必要である。
[3]関連に従った制御の伝搬機能
　共通データへのコンパイル要求に対し，実際にはそれと特定の関連で結ばれているモジュールエンティティにコンパイル制御が渡っている。このようなエンティエィ間での制御の伝搬ができる必要がある。
[4]オブジェクト間のエンティティの整合性保証機能
　図３の例におけるモジュールエンティティＭ１はＹＤＳ／ＹＰＳオブジェクトおよびＹＰＳソースとして複製される。ＹＰＳソースについてはツール利用者から遮蔽されるため不整合は生じないが，ＹＤＳ／ＹＰＳオブジェクト間に重複して存在するエンティティ間の整合性を保証できる必要がある。
　これには，ｔａｇ方式に基づくチェック機能[2]あるいは，チェック結果に基づいた該当競合オブジェクトへの特定エンティティ／関連の置換，削除のための機能が必要である。

4．まとめと今後の課題

設計およびプログラミングのためのＣＡＳＥツールの統合化について，現状の問題点分析，解決に向けての方針，実現へのアプローチについて考察した。

今後の課題として上記考察結果に基づく統合ツールの実現とその過程での適用評価を行いたいと考えている。

参考文献

[1] 深尾至，西山好雄，豊島康文，藤本洋："通信ソフトウェア向き設計支援環境"，情報処理学会論文誌 Vol.31 No.7, pp.1113-1122(1990.1)

[2] 今野，阿部，中川，深尾，西山："構造化された設計情報の静的解析に関する考察"，平成元年度電子情報通信学会春季全国大会，p.6-65(1990.3)

[3] 嶋川，川島："通信ソフトウェアへのＣＡＳＥ適用の考察"：平成元年度電気関係学会北海道支部連合大会論文集，pp.388-389(1989)