4S−2

CASEツール統合化へのアプローチ(2) −統合を支援するリポジトリに関する考察−

坂本　康広　阿部　弘彰**　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

我々は通信ソフトウェア開発を一貫して支援する開発環境の構築を目指している。

中でも上流工程については，MRVアプローチに基づく設計支援システムYDS（YACⅡ-oriented Design System）を開発し，プログラミング支援システムYPS（YACⅡ Programming System）と連携させ，適用している。[1]

その結果，YDSとYPSの連携についてより高度な方式の必要性が判明し，設計とプログラミングが一体となった支援環境の実現を検討している。[2]

本稿では，このYDSとYPSとの統合化を実現する際に必要となるリポジトリ機能について，その要件と実現方式に関する考察を述べる。

2．リポジトリの位置づけと要件

(1)一般的な統合化のアプローチ

ツールの統合に関する問題を解決するアプローチとして，統合化CASE環境の必要性が一般的に論じられており，標準化も進められている。[3]

それらによると，CASEツール統合化のフレームワークとして，以下の3点の統合が必要であるとされている。

表1　CASE統合化のフレームワーク

フレームワーク	説明
①制御統合	ツールを他のツールから呼び出せるようにする。
②プレゼンテーション統合	ユーザにとってツールが同じように見えるようにする。
③データ統合	どのツールからも情報を共有でき，かつ操作可能にする。

YDSとYPSの統合化をこの図式にあてはめてみると，②プレゼンテーション統合は個々のツールの作りの問題として捉えることができるが，①制御統合，③データ統合については，YPSオブジェクトを隠蔽してYDSからダイレクトに操作可能とする上で必須の機能であることが判明した。[2]

これらをYDS／YPSの固有機能に含めることも可能であるが，将来のより上流工程支援や下流工程支援との統合化を進める上では，独立に位置づける方が有利と考えられる。本稿では，特にデータ統合の実現メカニズムとしてのリポジトリについて議論を進める。

(2)YDS／YPS統合化のためのデータ連携

YDS／YPS統合化の目的は，従来のソースファイル単位にレビューやコンパイル，試験等のプログラミングのアクティビティを行うのではなく，例えば『この共通データを変更したので関連するモジュール群をコンパイルせよ』と言った，設計時に認識される抽象化レベルでのプログラミングを実現することである。

このために管理すべき情報として，以下のものがある。

表2　リポジトリの管理情報

情報	説明
①ロケーション情報	YPSオブジェクトファイルの物理的ロケーション情報
②タグ情報	各オブジェクトファイル中に含まれる関連情報

この2点があることから，リポジトリは図1に示すように2階層の情報管理を行うものとする。

ここで，プログラム部品・関連管理機能は予めYDS／YPSの各オブジェクト（構造化ドキュメントやYACⅡチャート，YPSプログラムなど）が規定するプログラム部品（エンティティ）とそれらの間の関連を概念スキーマとして，各オブジェクト間の連携を管理する。

また，ディレクトリ管理は各々のオブジェクトの物理的なロケーションを維持管理する。

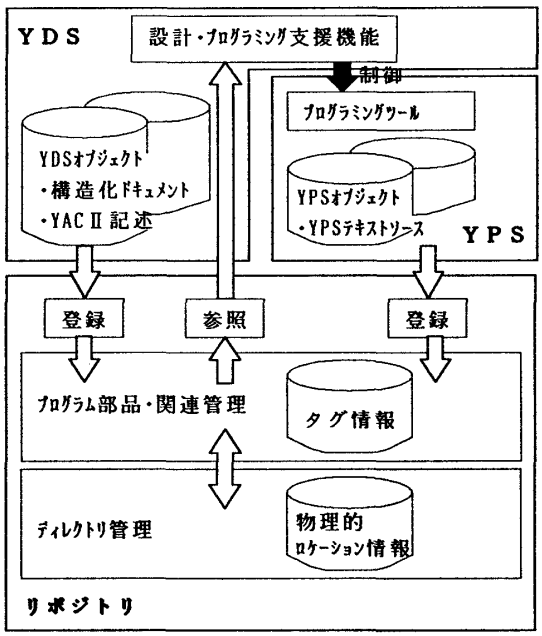


図1　リポジトリによるデータ統合

An Approach for Integration of CASE tools (2)
Yasuhiro SAKAMOTO, Hiroaki ABE, Yukiko TAKAHASHI
Itaru FUKAO, and Yoshio NISHIYAMA
FUJITSU LIMITED
FUJITSU HOKKAIDO COMMUNICATION SYSTEMS LIMITED

⑶運用面からの要件

近年の通信ソフトウェアの大規模化に伴い，開発要員は増加し，開発拠点も分散化する傾向にある．（図２）このような環境では，ＹＤＳ／ＹＰＳの統合化を図る上で，多数の開発者が異なる地点から情報がアクセスされることを想定する必要がある．この点もリポジトリを独立な機能として実現することが必要な所以である．

また，リポジトリを集中型・分散型の何れの形態で実現するにしても，そのデータ内容が破壊されて一貫性が損なわれたりすることを想定し，リカバリ対策を考慮しておくことが重要である．

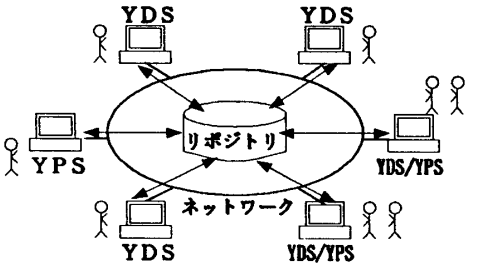


図２　分散環境におけるリポジトリ

３．実現方式

以上の要件に基づく，リポジトリの実現方式を述べる．

⑴データ統合インタフェース

個々のツールはそれ自身に固有なオブジェクトを生成すること，ツールとリポジトリの役割分担はシンプルな方が良い，との考えから，インタフェースとして以下の方式を採用する．

①オブジェクト登録

オブジェクトが生成・更新される都度，その内部に含まれるプログラム部品とそれらの関連情報の抽出は各ツール側で行うものとし，リポジトリには，１オブジェクトに関する関連情報と物理的ロケーションのみを通知する．（ｔａｇ方式）リポジトリは通知された情報を，予め規定された概念スキーマに従って内部に展開し，他のオブジェクトの情報とリンクする．

本方式により，リポジトリは各ツールの全てのオブジェクトからの抽出情報を収集することにより，何時でも再生可能なため，データ破壊にも簡単に対応できる．

②オブジェクト参照

参照に際しては，所定のプログラム部品の型や関連の種類，具体的な部品名や関連の名前等をキーとして該当する情報をリスト形式で出力する．

⑵リポジトリの形態

前述のように，分散型開発環境において用いることを想定すると，リポジトリとして集中／分散の何れの形態を採用するかは議論が別れるところである．各々の得失を比較すると表３のようになることから，以下の理由によりここでは分散形態を採用する．

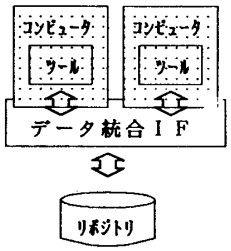
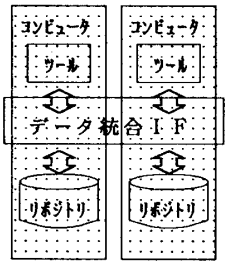
①ネットワーク環境による要件

現状ではネットワーク環境の整備が万全ではない場合が多く，それに影響されない方式が必要である．

②情報の一元管理に関する要件

分散型においてはリポジトリ間に矛盾が発生する場合があるがマスタやサブマスタの設置やそれらの矛盾のチェック機能などの作成により解決する．

表３　リポジトリの形態

型	概要	比較	
集中型		利点	・情報の一元管理ができる． ・情報の変更が容易．
		欠点	・登録／参照にネットワークが介在し性能に影響する．
	プラットフォーム内で唯一のリポジトリを持つ．		
分散型		利点	・ネットワークに関係なく登録／参照が可能．
		欠点	・リポジトリ間の整合性を保つ機能が必要． ・情報の変更時にリポジトリ間にタイムラグが発生する．
	プラットフォーム内で複数のリポジトリを持ち，それらの整合性を保つ．		

⑶効果と課題

上記のような実現方式により以下のような効果が期待でき，同時に以下のような課題の発生が予測される．

①　ＹＤＳとＹＰＳの連携において従来，人的作業で行っていた作業を殆どなくすことができる．

②　分散型のリポジトリの採用で性能面，安全面に於いて実用レベルのリポジトリが実現できる．

③　各コンピュータにリポジトリを置くため，各々のコンピュータのディスク容量と情報の変更時のオーバヘッドの解消が課題になる．

４．おわりに

本稿においてリポジトリに関する要求仕様と実現方式について考察した．今後はこれを用いて実際のリポジトリの実現を行う予定である．

また実現後は，上記の要求仕様の正当性や実現方式の実用性などを検証していきたいと考える．

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 (July. 1990)

[2]阿部弘彰，坂本康広，深尾至，西山好雄：”ＣＡＳＥツール統合化へのアプローチ「設計とプログラミングの統合に関する考察」”

[3]松本正雄：”ＣＡＳＥ統合のためのインタフェースの標準化現状”，情報処理学会論文誌 Vol. 31 No. 8 pp. 1086-1094 (Aug. 1990)