3J−4

ソフトウェア設計･製作支援システム　CADRIS（1）
ー　開発思想　ー

大脇　隆志**，　大槻　繁*，　高橋　勇喜**，　林　利弘**
（株）日立製作所　*システム開発研究所　**大みか工場

1．はじめに

計算機制御応用システムのソフトウェア開発・保守を対象として，設計工程以降の一連の意思決定，及び，これに伴う作業を支援するCASEツール：CADRIS(Computer Aided Design and Reuse Environment with Intelligent Support system)を開発，運用を行っている．本稿では，この支援環境の開発思想について述べる．

2．CASEツール開発の背景

ソフトウェア工学という学問分野が提唱されて以来約三十年を経ているが，実用的な開発支援環境を構築し，これを快適に運用するにはまだまだ技術革新が必要である．

当初はエディタ，コード生成，ドキュメント生成等の個別ツールを開発者が適宜個別に選別，使用していた．しかし，近年，ソフトウェアの規模，複雑さの増大にも拍車がかかり，プロジェクトワークでなくては対処できなくなって来ている．これに伴い，支援環境も，全ての成果物（仕様，ソースコード等），さらには，これ等に関連した個別ツールさえも完全に計算機管理することができる統合支援環境が必要になって来ている．

例えて言えば，フルコースの料理を要求されているようなものである．どの料理にも粗相があってはならないし，各料理の組み合わせ，サービスの順序も非常に重要なファクタになっているわけである．しかも，当たり前のことを，さりげなく，当たり前に提供しなくてはならない．

我々は，このような状況の中で統合支援環境を構築するために，次のように技術開発の重点化と特性付けを行ってこれを進めている．

（1）対象分野，工程の限定

支援分野をオペレーティングシステムやデータベースシステム等の基本ソフト，オンライン銀行システム等の情報システムではなく，計算機制御ソフトウェアシステムに限定している．こうすることにより，分野限定しつつも，実用的な視点からは，ツール導入に際しての文化的な衝突も避けられ，また，標準パッケージの整備や，再利用の促進が図れる．現に，CADRISが制御分野の設計・制作の支援をしている証左は，その対象分野向けのライブラリの充実化にある．

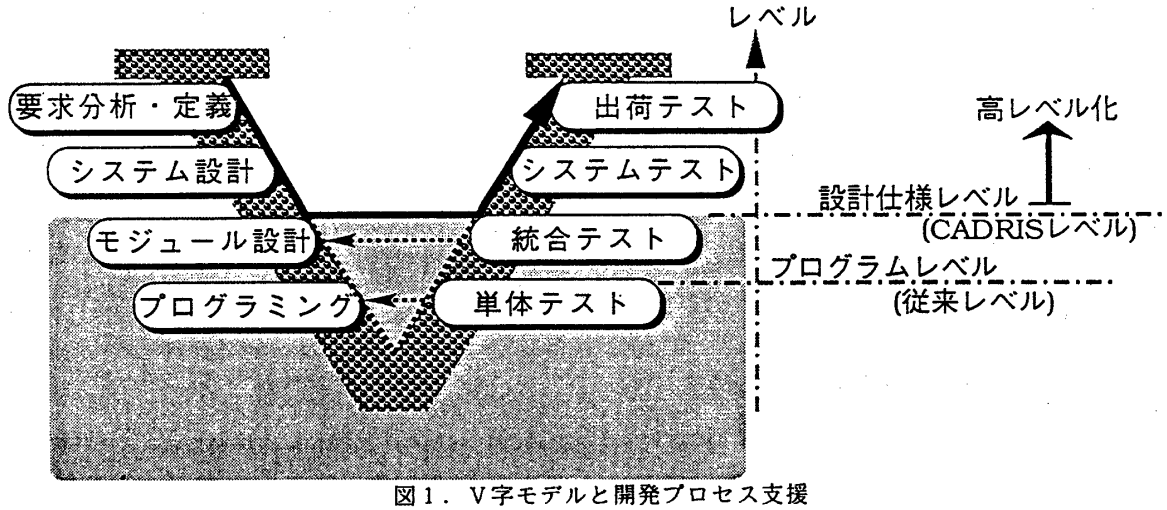

図1．V字モデルと開発プロセス支援

Design Support Environment for Realtime Software System:CADRIS -the Concept-
Takashi Owaki**, Shigeru Otsuki*, Yuki Takahashi**, Toshihiro Hayashi**
*Systems Development Lab.,Hitachi,Ltd., **Omika Works,Hitachi,Ltd.

（2）再利用中心

同一の組織でいくつものソフトウェア開発を進めていると再利用の比率は必然的に高くなってくる．新規作成の情報分析や要求仕様定義を含めた技法中心のアプローチよりも，成果物や開発プロセスの徹底的な管理指向の再利用に重点を置いている．

（3）大衆技術

もともとソフトウェア工学というのは，大衆技術を開発することに目的がある．天才が造る限られたソフトウェアにはおそらくＣＡＳＥツールは不要である．数多くの人々がプロジェクトワークとして大量のソフトウェアを開発することを支援するところにこそ意義があると考える．

（4）分散・協調ソフトウェア開発

多人数，複数組織での開発は避けられず，それぞれの役割を持った人々が協調して作業を進めてゆかなくてはならない．その上，物理的に離れた場所での作業も多い．こういった状況での支援では，計算機どうしを単にネットワークで結合するだけでは不十分で，ソフトウェア工学上の仕掛をそれ等の上に構築しなくてはならない．統合支援環境としても，各個人の内部での作業はもとより，より，開発者間のインタフェースに力点を置かざるをえない．

3．アプローチ

ＣＡＤＲＩＳの開発に当たっては，上記要求を踏まえ次に示すようなアプローチをとっている．

（1）開発プロセス支援

ソフトウェア開発のライフサイクルモデルとして図1に示すＶ字ライフサイクルモデルを採用している．これは，従来のウォータフォール型のモデルに近いものであり，その欠点を説明するためによく使われるが，現状の実用的な開発では，この生産形態を免れることはできない．

こういった生産形態で一番の問題は，レビューやテストの後の手戻り作業にある．特に，最初に行った意思決定の妥当性が最後にならないと検証できないという原理的な問題は深刻である．

我々は，このＶ字モデルを開発プロセスの抽象化の階層の観点から，ＣＡＳＥツールのインタフェースレベルを可能な限り高く設定し，下位のレベルの作業を隠ぺいし機械化・自動化することを解決の手段としている．

（2）プログラミング言語独立

Ｖ次モデルの抽象化の企てによって，まず，隠ぺいされる開発プロセスがプログラミングレベルのものである．実際に設計成果物というのは，その組織での技術蓄積の成果物であるから，なるべく汎用的な形で残して置くべきである．従って，プログラミング言語に依存したものでは再利用の範囲等も限られてしまうことになる．我々は，全ての仕様をまず設計仕様のレベルを起点として考え，その仕様の体系の中に，プログラミング言語とは独立なロジックやデータ構造等を記述する高位の設計言語インタフェースを設定している．

（3）一つの事実は一箇所に(One Fact in One Place)

これは，データベース設計でよく言われることであるが，一つのデータや事実は，当該世界に唯一の場所に蓄積しないと，不整合の問題を引き起こすことになる．従来，ソフトウェアの世界はファイル中心の蓄積形態をとっているために，種々の依存関係にあるデータが異なる場所に蓄積されてしまう傾向がある．我々は，ソフトウェアに関わる全成果物を実体・関連モデルによってモデル化し，これに論理レベルで仕様データベースを構築して来た．

さらには，「一つの作業は一度だけ(One Act in One Process)」といった方向にまで持っていくことを目指している．ソフトウェアというものは作らないに越したことはない．まして，一度行った作業と同一のものを何回も繰り返すことは絶対に避けなくてはならない．

概ね，上記のアプローチを概念上簡潔にまとめると，次の三つに要約される．

Concept 1: Check for Step（開発プロセスの検証）
　抽象度の高いレベルでの正しい開発プロセスの支援
　開発者間のコミュニケーション支援
Concept 2: Constraint and Controle（制約と統制）
　言語非依存のインタフェースと，誘導
Concept 3: Configuration（構成管理）
　成果物，及び，これ等の間の関連（個別ツール）の関係管理．版管理等．

4．おわりに

実用レベルの支援環境では学界の先進的な提案を考慮しつつも，そのユーザである開発担当者の文化を配慮し，彼らの意思決定作業を真に支えるものを提供しなくてはならない．今後ともこういった実践的立場で，ソフトウェア開発に関する理論の検証を進めて行く．

参考文献

野木，中所：プログラミングツール，昭晃堂，1989