制御システム開発への構造化分析手法の応用

3K－9

小平和正
富士ファコム制御株式会社

１．はじめに

近年，産業用コンピュータを用いた制御システムにおいてもソフトウェア開発の品質，生産性の向上は急務となっている．ここ数年ＣＡＳＥ(Computer Aided Software Engineering) に注目が集まっているが，システム開発のトータルな生産性を期待するには，各工程でのプロセスの確立と個々を支援するＣＡＳＥツールの統合化が必要である．特に現在のソフトウェア開発で主に用いられているウォータホール型の作業工程の場合，上流工程での作業の質がその後の品質ひいては生産性を左右するため，上流での手法の確立と支援が重要である．そこで当社ではリアルタイム構造化分析手法を拡張して制御システムのソフトウェア開発へ適用し，その評価を行っている．本稿ではその拡張の概要を報告する．

２．上流工程での仕様と構造化分析手法

１９８０年にＴＲＷ社の実施した調査によればソフトウェア・エラーの６４％は要求分析と設計の段階発生し，受け入れテスト前に発見される設計段階のエラーは３０％しかないとされている[1]．分析，設計工程での仕様の作成，受渡しは図１のようにユーザと分析／設計者との間，分析／設計者と製作者の間で行われる．ユーザと分析／設計者との間では，ユーザの要求を分析／設計者が仕様書にまとめ承認を得る形式をとる．また分析／設計者と製作者の間では，分析／設計者が仕様書を提示し，製作者が設計書を作成，それをレビューして開発，テスティングを行い，ソフトウェアを完成するという，やり取りが行われる．

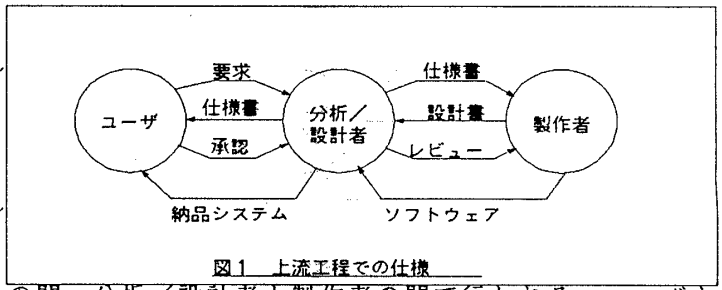

図１　上流工程での仕様

本来分析手法は前者を中心に適用されるものであるが，現状では従来の方法である言葉中心の仕様表現との間のギャップが大きい．従って前者では，構造化分析手法の実作業への適用はユーザの理解が得られるか（仕様確認が出来るか）に大きく依存する．一方，後者では仕様を受けて作成される設計書は構造化設計手法が一般になっていることもあり，仕様が構造化分析の手法で記述されついることに抵抗は少なく，仕様が厳密化されるメリットがそのまま生かされる．

また，ユーザと分析／設計者との間ではシステムが「何を」するのかを記述するため，分析図の中の機能は抽象的なモデルでかまわない．むしろ具体的なシステム化のための情報は不要とされる．一方，分析／設計者と製作者の間では具体的に「どのように」作成するかの裏付けをもとに仕様化されている必要がある．また制御システムの開発には，リアルタイムの仕様表現が必須である．

３．リアルタイム構造化分析手法の拡張と適用

構造化分析手法のリアルタイム・システム分析への拡張は，Wordらによる拡張[2]とHatleyらによる

An Extended Structured Analysis Method for Process Control Systems, Kazumasa Kodaira, Fuji Facom Corporation, 1 Fujimachi, Hino, Tokyo 191, Japan.

拡張[3]が代表的である．双方ともモデル化の観点は大枠として共通しているが，制御システムでは制御信号が多数になるため，Wordらによる拡張では図面が複雑になるという大きな問題がある．また，Hatleyらによる拡張の方が手法を習得するための資料多い，サポートするＣＡＳＥツールが多い等を考慮しHatleyらによる拡張をベースとすることとした．

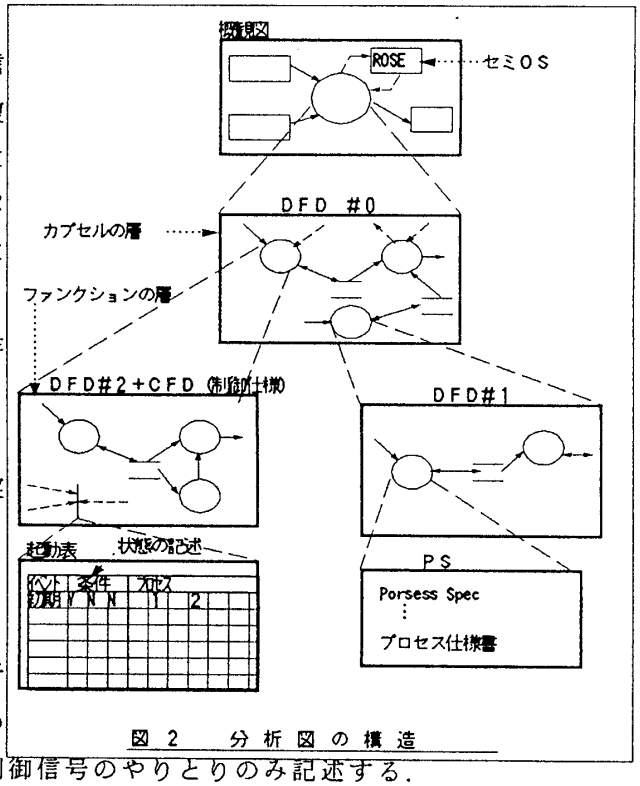

図２　分析図の構造

我々はまず，社内で評価できる分析／設計者と製作者の間での適用と評価を開始した．この際以下のような，規範の設定と手法の拡張を行った．

① 通常，制御システムを汎用のＵＮＩＸ計算機に実装する場合リアルタイム処理を実現するためのセミＯＳが利用される．当社ではＲＯＳＥ(Real time system Operating management & Support Environment)パッケージを用いることが多い．分析において，このセミＯＳ内の機能は既知であるため分析の対象とせず，概観図のターミネータとして制御信号のやりとりのみ記述する．

② ＲＯＳＥでは，複数の実行単位（ＵＮＩＸのプロセス相当，ＲＯＳＥではファンクションと呼ぶ）をグループ（カプセルと呼ぶ）として管理する機能があり，この割付は重要な設計項目のある．分析図では，第一階層のＤＦＤのバブルを分割する際カプセルに見立てて分割する．ただし，カプセルが多くなりすぎる場合は複数カプセルをグールピングした抽象的な機能（バブル）を導入し階層化する．

③ 上記②のバブルの直下は，実装時のファンクションの階層とする．ただし，１カプセル内にファンクションが多すぎる場合は，複数ファンクションをグループ化した抽象的なバブルによる階層を間にもうけ，そのバブルの下にファンクション相当のバブルを持つＤＦＤを作成する．

④ 起動表は本来その制御仕様の含まれるＤＦＤ内のバブルに対し起動をかけるが，実装上は他のカプセル配下のファンクションに起動をかけることは常であり，これを可能とする．即ち，他のＤＦＤ内のバブルも起動表に含めて良いこととする．

⑤ システム内にフラグを持ち，そのフラグの状態あるいはフラグの組み合わせで処理の流れを制御している場合，そのシステムは有限状態マシンである事が多い．手法では，状態遷移図と起動表を別々に作成するのが，記述と検証の容易性から両表を一つにまとめて記述することとする．

４．今後の予定

現在，実業務に上記の拡張を取り入れた分析手法を適用中で，これまでのところ従来より厳密な仕様の受渡しと，打合せの効率化が図れるという評価が出ている．今後，システムの保守まで含めた総合的な評価およびユーザとの仕様確認での適用評価を行う予定である．またセミＯＳのオブジェクト指向環境への対応に伴い，構造化分析手法からオブジェクト指向分析手法への移行も課題となる．

参考文献　（１）A.S.Fisher，黒田・中島訳，CASEソフトウェア開発にはCASEツールをつかって，共立出版，1990
（２）P.T.Ward and S.J.Mellor，Stractured Development for Rwal-Time System，Prentice－Hall/Yourdon Press，1985.
（３）D.J.Hatley and I.A.Pirbhai，立田監訳，リアルタイムシステムの構造化分析，日経ＢＰ社，1989.