通信ソフトウェア向き設計支援環境†

藤本 洋†† 豊島康文†††
西山好雄†††† 深尾 至††††

企業内情報網やISDNなどの進展に伴う通信ソフトウェアの大規模化や多様化・複雑化の傾向に対応するため，その生産性や品質の大幅な向上が重要な課題となっている．以上の背景から本稿では通信ソフトウェア向けCASE環境，特に設計支援環境という課題について述べる．上流工程の問題として，仕様の記述形式や抽象化レベルが作業者ごとに一意ではないために様々な認識ギャップが存在する点が挙げられる．現在広く研究開発されているCASEでも，その多くは仕様記述の抽象化レベルが任意であるために支援効果が作業者に依存し，上述の問題に対して十分とは言えない．本稿では，上流工程において作業者が所定の時期に所定の抽象化レベルで一意に仕様の詳細化ができるような，ガイダンスに基づく設計支援方式を提案する．本アプローチをMRV (Multi-layered Refinement by Views) と呼び，大規模ソフトウェア開発のライフサイクルモデルとして一般的なV字型曲線に基づき仕様詳細化過程の抽象化レベルを多段階に分割することを基本とする．また，各段階内および段階間の詳細化のガイダンスと連続性を保証する仕組みの概念としてビューを導入した．本方式の目的は，ヒューマンファクタの影響を低減し上流工程における品質と生産性を向上させることにある．本方式はYDS (YAC Ⅱ-oriented Design System) として実現され実際の開発に適用されており，最後にその評価と課題を述べる．

1. はじめに

企業内情報網やISDNなどの進展に伴って通信ソフトウェアの開発量は急増し，要求される機能も多様化，複雑化してきている．また一方では，タイムリーにユーザニーズに答えてゆくことも強く望まれている．これに対し従来の開発環境は高級言語やクロス環境を中心に整備されてきたが，さらにその開発方法論や支援環境の全ライフサイクルにわたる革新が今後一層の効率化を図るうえで不可欠となりつつある．以上の背景から，新しい統合開発環境を研究開発中であり[1),2)]，本稿では特に上流工程の支援方式について述べる．

従来の通信ソフトウェア開発の上流工程では，仕様記述に様々の記述形式が用いられ，また，その記述内容の抽象化レベルも個人差が大きかったため，段階的詳細化のステップごとに仕様の誤解や漏れ，インタフェースの不整合等が混入しがちであった．したがって，このような仕様の記述形式や内容の抽象化レベルの属人性を極力取り除き，誰でも同レベルで的確に仕様を詳細化できる環境の実現が課題となっている．

近年，上流工程の支援ではupperCASEと呼ばれる技術が注目されている．なかでも通信ソフトウェアのようなリアルタイム分野向けには，標準的なリアルタイム構造化手法[3),4)]に基づくものや，STATEMATE[5)]などのように独自の拡張に基づくものが実用化されつつある．

前者は仕様の記述形式としてD/CFD (Data & Control Flow Diagram)[3),4)]や状態遷移図，データ構造図，プロセス仕様などをサポートし，それら相互の関係を含めて一貫性のチェックが可能である．このため，システム機能の外部仕様レベルにおける記述形式を統一し，記述全体における記述漏れや矛盾をチェックすることができる．しかしながら，これらの記述形式を用いてどのような手順でどのような抽象化レベルまで詳細化するかは任意であるため，その支援効果は作業者の経験に依存する．また，後者ではstatechartと呼ぶ状態概念を扱う表記法を導入し，記述された仕様の実行検証やソースコード生成をも可能としているが，生成されるプログラムの実現方式（プロセス管理や資源管理などの機構）は通信ソフトウェアのようなアーキテクチャには適合しにくい．

以上の課題を解決するため，ここではまず大規模ソフトウェア開発のライフサイクルモデルとして一般的なV字型の曲線モデル[6)]（以後Vモデルと呼ぶ）を導入する．Vモデルは開発の時間軸に対する仕様記述の

† Design Support Environment for Communications Software by HIROSHI FUJIMOTO (Business Switching Systems Division, FUJITSU LTD.), YASUFUMI TOYOSHIMA (Transmission Development Division, FUJITSU AMERICA, INC.), YOSHIO NISHIYAMA and ITARU FUKAO (Engineering Department, Communications Software Development Division, FUJITSU LTD.).

†† 富士通(株)交換事業本部複合交換機事業部
††† フジツウアメリカ
†††† 富士通(株)通信事業推進本部ソフトウェア開発部ソフトウェア技術部

抽象化レベルの推移を表現するものであり，上流工程はソフトウェア仕様を段階的・連続的に詳細化しプログラムとして実現してゆく過程と，下流工程はプログラム群を段階的に統合化してゆく過程とみなすことができる．

本稿では，作業者が所定の時期に所定の抽象化レベルで仕様を記述できるようなガイダンスに基づく支援方式について述べる．このアプローチを MRV(Multi-layered Refinement by Views) と呼び，V字曲線に沿って仕様記述の抽象化レベルを多段階に分割することを基本とする．また，各段階内および段階間のガイダンスと連続性を保証する仕組みを実現する概念としてビューを導入した．MRV に基づく支援によりヒューマンファクタに依存しない一意な仕様記述が可能となり，上流工程に起因するバグの減少や無駄な作業の削減による品質と生産性の向上が期待できる．

各ビューの記述形式としては D/CFD と YACⅡ (Yet Another control Chart Ⅱ)[7] を拡張して採用しており，その十分性についても述べる．さらに，円滑な段階的詳細化を可能とするビューによるガイダンスメカニズムを考察し，本メカニズムを実現するために開発した YDS (YACⅡ-oriented Design System) を紹介する．最後に YDS の評価および課題を述べる．

2. 通信ソフトウェアの段階的詳細化過程

本章ではまずVモデルに基づいて，従来の通信ソフトウェアの段階的詳細化の流れと問題点を考察する．

2.1 Vモデルに基づく解釈

通信ソフトウェアは一般に少ない資源 (CPU，メモリ量等) と厳しい実時間性の制約のもとで低レベルなハードウェアの操作からアプリケーション機能までを果たす必要があるため，高度な並列多重処理構造を持ち，仕様記述における難度も高い．従来の通信ソフトウェア開発における仕様の詳細化の流れをVモデルに当てはめると図1のようになる．

① まずフリーフォーマットの日本語文により要求仕様が記述され，

② 次に何段階かの中間の記述形式を用いてその並列多重の処理構造を抽出し，

③ 逐次的な処理単位の仕様を記述する．

④ 個々の処理単位の仕様から高級言語記述を作成し，言語処理系により実行可能な機械語を得る．

本過程において，現状では記述情報の厳密さや正確さ，一貫性は作業者に依存して任意である．一般的な保証手段として作業標準やレビューの実施などがあるが，その効果も作業者の熟練度に依存するため，ソフトウェアの難度と相まって十分とは言い難い．

2.2 段階的詳細化過程における問題点

段階的詳細化の過程で記述内容の抽象化レベルが一意でないことは，実際のソフトウェア開発において，以下のような認識上のギャップとして顕在化する．

① 作業者相互（書き手と読み手）のギャップ

② プログラム相互（設計対象と関連対象）のギャップ

③ 工程相互（設計工程と試験工程）のギャップ

これらは，後工程における仕様の漏れや誤解，インタフェース誤りなどの障害に繋がることが多い．

この問題の原因としては以下の3点が考えられる．

① 個々の時点で使用される仕様記述形式において，作業者がどのような抽象化レベルで記述するべきかというガイダンスが欠如していること．

② 段階的詳細化の進行に伴う記述の抽象化レベルの適切な移行とそれらの間の連続性の保証について作業者へのガイダンスが欠如していること．

③ ユーザ要件である要求仕様からソフトウェアの実現方式である設計仕様へ変換する際の作業者へのガイダンスが欠如していること．

今回はこれらの原因のうち，①項と②項を解決するためのアプローチについて述べる．③項の解決に関する報告は別稿に譲る．

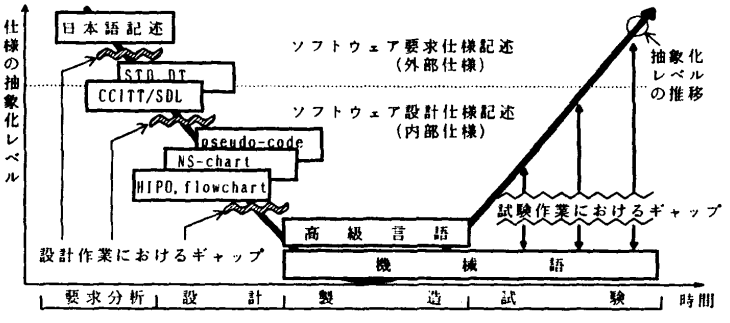


図1 従来のソフトウェア開発の流れ

Fig. 1 Conventional development process of communications software, from the aspect of stepwise refinement.

3. 段階的詳細化作業の連続的なガイダンス

前述の認識ギャップを解消するためには，単にドキュメントの記述形式を統一するだけでなく，詳細化の進行に伴って所定の時期に所定の抽象化レベルで記述するよう作業者を導くガイダンスが必要である．そこで，このような一意な詳細化を可能とするため，Vモデルに沿って仕様記述の抽象化レベルを複数の段階に分割し，それに基づいて詳細化作業の流れを誘導するアプローチを採用した．以上のアプローチをMRVと呼ぶ．

さらに，分割された各段階および段階間のガイダンスと段階間の連続性を保証する仕組みを実現する概念として，ビューを導入した．ビューとはMRVを支援するための情報の見せ方であり，作業者はビューを通じて作業を進めることにより，V字曲線に沿った連続的な段階的詳細化の道筋を自然にガイダンスされる．

MRVの狙いとする効果はヒューマンファクタに依存しない一意な仕様記述を可能とすることであり，先に述べた認識ギャップを解消することである．それによって，上流工程に起因するバグの減少と手戻りなどの余分な作業の削減を図ることができ，品質と生産性の大幅な向上が期待できる．

3.1 通信ソフトウェアの段階的詳細化過程

通信ソフトウェアの段階的詳細化過程における抽象化レベルを以下の5段階に分割する．

（1） 概念レベル

実現しようとする機能やサービスの仕様をソフトウェアの作りに依存しないレベルで記述する．通信ソフトウェアはハードウェア制御から通信プロトコル，計算処理など多岐にわたる機能を持っており，それらが相互に通信しあったり，ある時間的順序性に従って動作することが大きな特徴であるため，これらの性質を明確に定義する．

（2） プロセッサ構成レベル

通信システムは一般にマルチプロセッサから構成されるため，概念レベルで定義された機能を複数のプロセッサに配分する．その際には，ハードウェアとソフトウェアの機能分担，プロセッサやメモリ容量の制約等の物理的な条件を考慮し，単一プロセッサ上のソフトウェアが分担する機能を決定する．また，プロセッサ間インタフェース仕様やハードウェア／ソフトウェアインタフェース仕様を具体的に記述する．

（3） タスク構成レベル

単一プロセッサのソフトウェアの持つ機能とそれらの間のやりとりの関係を，処理効率や理解しやすさ等を考慮しながら並列処理単位（タスク）へ分割し，それらの間の通信方式を決定する．タスク構造とグローバルな処理シーケンスの対応を明確にするとともに，共通データ域の割付けや，タスク間通信に伴うOS資源を明確に定義する．

（4） モジュール構成レベル

個々のタスクについて，処理シーケンスや状態遷移を考慮しながらデータや手続きなどのプログラム部品（モジュール）の単位に分割し，そのインタフェースを決定する．

（5） モジュールレベル

データや手続きとして明示されたモジュールについて，データ構造やアルゴリズムを決定する．プログラムとして実行可能なレベルまで詳細化されたところで記述が完了する．

以上の段階において，(1)はシステム要求定義工程，(2)はソフトウェア要求定義工程，(3)，(4)は基本設計工程，(5)は詳細設計工程にそれぞれ対応する．また，一般に(1)，(2)で定義されたものを外部仕様，(3)，(4)，(5)で定義されたものを内部仕様と呼ぶ．

3.2 抽象化レベルに対応した記述形式の選定

前節の5段階の抽象化レベルに対応した各ビューの記述形式が持つべき要件を順次検討する．

（1） 概念レベルの記述

この段階の記述はシステムの果たすべき機能について開発者と要求者の間でイメージを合わせるのに用いられるため，その記述形式は双方にとって理解が容易であるように以下の条件を満足する必要がある．

① 対象世界の言葉を用いて記述されること．

② 状態遷移のようなベーシックな記述だけでなく，最近の通信ソフトウェアに占めるアプリケーション機能の比率の増大に伴い，その性質をより効果的に表現可能なデータ指向の記述ができること．

③ 直感的かつ曖昧性なく表現するため，シンプルな図的表現を中心とすること．

④ ソフトウェア要求仕様の概要から詳細までを必要に応じてクローズアップできる階層性を持つこと．

以上の条件を満足する記述形式としてリアルタイム用に拡張された構造化分析手法に基づくD/CFDを採用した．D/CFDでは機能とそれらの間の通信をD/CFD

自身によって，機能間の同期や順序関係を状態遷移図等によって，通信されるデータやイベント項目をデータ辞書によって表現することができる．

ただし，標準的な D/CFD では，仕様の記述力が不足するため[8]多重処理や外部インタフェース仕様などの表記法の拡張が必要となっている．

（2） プロセッサ構成レベルの記述

この段階では一つのプロセッサに搭載されるソフトウェアの機能とそれらの関連を明確にすることにより，設計すべきソフトウェアの仕様を明確にする．したがってその記述形式への要求は理解容易性という面で概念レベルの②～④と同様であるが，さらに以下の条件を満足する必要がある．

① 対象世界での機能記述に対し，ソフトウェア世界での言葉による実現方式を対応付けること．

② プロセッサ間インタフェースやハードウェアとのインタフェースの仕様を詳細に記述できること．

そこで記述形式としてこのレベルでも D/CFD を採用することとし，上記①②のために表記を拡張する．

（3） タスク構成レベルの記述

この段階ではソフトウェアへの要求仕様がソフトウェアの構造に変換される．このため，その記述形式は以下の条件を満足する必要がある．

① 実現されたタスク（並列処理単位）や共通データ，タスク間通信に伴う OS 資源等の配置や関係を一覧でき，プロセッサレベルの要求仕様との対応が明確であること．

② ソフトウェアの構成と動的な処理シーケンスとの関係を明確に記述できること．

この段階の記述形式としてタスク関連図とシーケンスチャートを採用する．タスク関連図は処理の静的な関係と資源の割付けを表現し，シーケンスチャートは処理の動的な流れを表現することができる．

（4） モジュール構成レベルの記述

モジュール構成レベルの記述形式は，タスクに割り当てられた機能の性質に応じて異なるモジュール分割手法を反映するため，状態遷移表とモジュール構造図とを併用する．前者は制御主体のタスクをイベントの流れに基づいて分割する際に用い，後者は業務処理主体のタスクを加工の手順に沿って分割する際に用いる．

（5） モジュールレベルの記述

この段階では，それ以前に決定された実現方式を反映した個々の手続きのアルゴリズムやデータの宣言を，見通し良く記述できることが重要である．このため以下の条件が挙げられる．

① 繰返しや分岐等の基本構文や構文の入れ子が直感的に把握できること．

② 個々のステートメントが説明的であること．

これらの条件を満足する記述形式として YACⅡ を採用した．YACⅡ はヘッダ部や引数宣言部，手続き部等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し，利用者定義構文や段階的詳細化記法等により説明的な記述を可能とする（**図 2**）．詳細な YACⅡ 記述は専用コンパイラにより実行可能コードに変換される．

以上に述べた 5 段階のビューの各記述形式を，通信ソフトウェアの段階的詳細化の流れとそこで記述されるべき内容との対応を**図 3** に示す．

3.3 段階的詳細化のガイダンス

今まで述べてきた詳細化と呼ぶ作業は，そこで付加される情報に着目すると，**表 1** に示すような操作を単独または組み合わせて行っているものと考えられる．

タイプ 1 から**タイプ 3** までの操作は記述された意味に依存するため，人間の判断に負うところが大きく，機械的に生成するのは困難である．これに対し，**タイプ 6・タイプ 7** は構文上の操作であるため機械的に変換可能である．**タイプ 4・タイプ 5** はその中間的な性質を持っている．**図 4** にこれらの詳細化操作の概念を示す．

図 4 の流れに沿ったガイダンスを実現するために

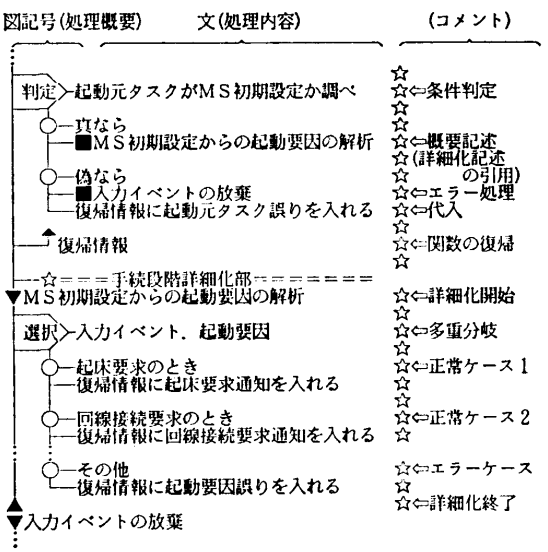


図 2 YACⅡ の例
Fig. 2 An example of YACⅡ.

は，個々のビューにおける詳細化操作の系列を明らかにしたうえで，そのタイプに応じた支援が必要である．

本稿では以下の二つの観点から支援方式を検討した．

（1） ビュー内での抽象化レベルの一意性の管理

大規模ソフトウェアの開発では一つのビューに記述される情報量も多く，複数の作業者によって様々に分担されるのが

表 1 段階的詳細化の基本操作
Table 1 Primitive operations of stepwise refinement.

種 別	名 称	内 容
タイプ1	分 割	ある対象項目を分割しより詳細な項目群を列挙する．
タイプ2	構造化	列挙された項目群をまとめ実現のための構造を与える．
タイプ3	推 敲	ある観点から記述項目の追加／削除／合成／分割を行う．
タイプ4	特 定	ある対象項目の属性値を特定する．
タイプ5	分 類	一連の記述項目をその構造に着目してクラスタ分けする．
タイプ6	変 形	一連の記述の表現形式を別な形式に変換する．
タイプ7	選 択	一連の記述項目から同じ分類のものを選別する．

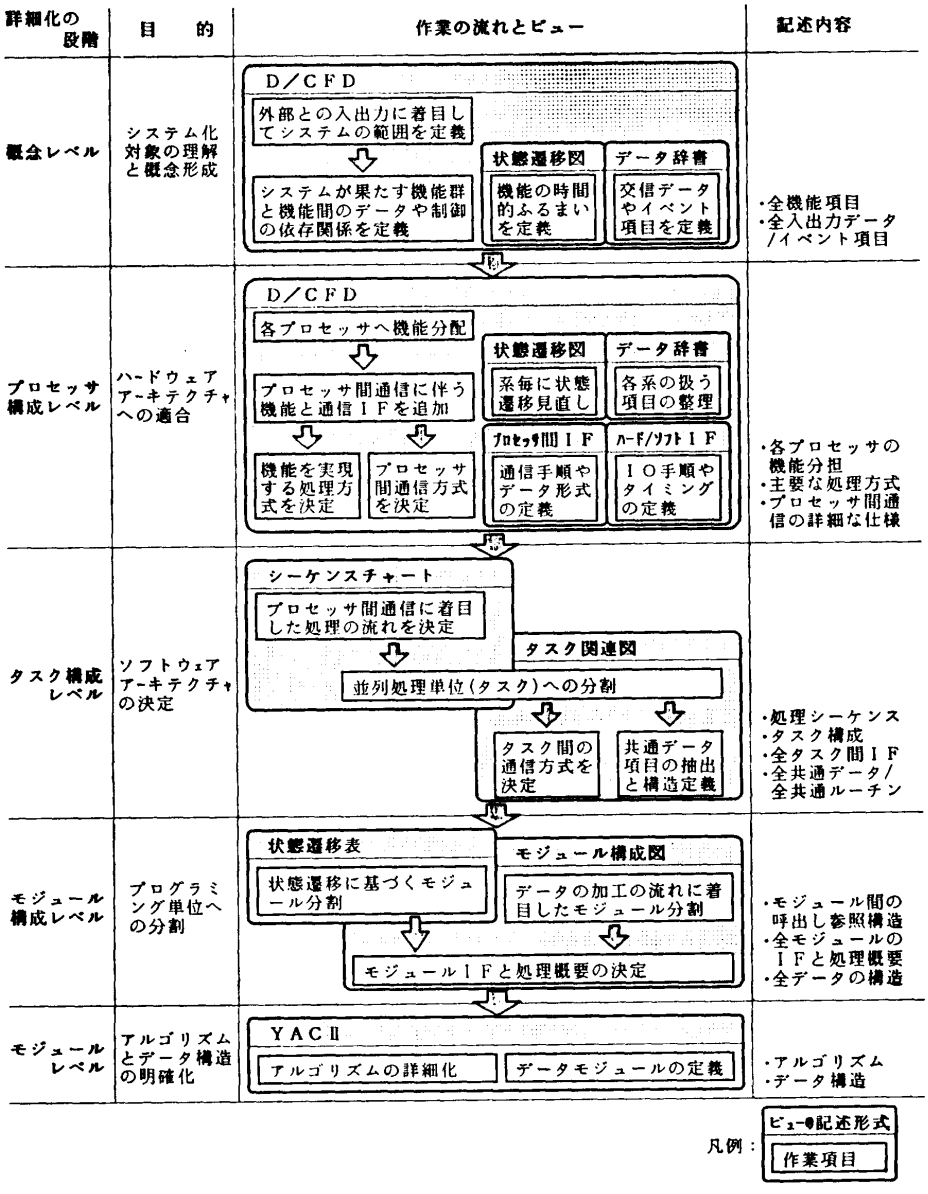


図 3 段階的詳細化の流れとビューの対応
Fig. 3 Allocation of views correspondent with stepwise refinement processes.

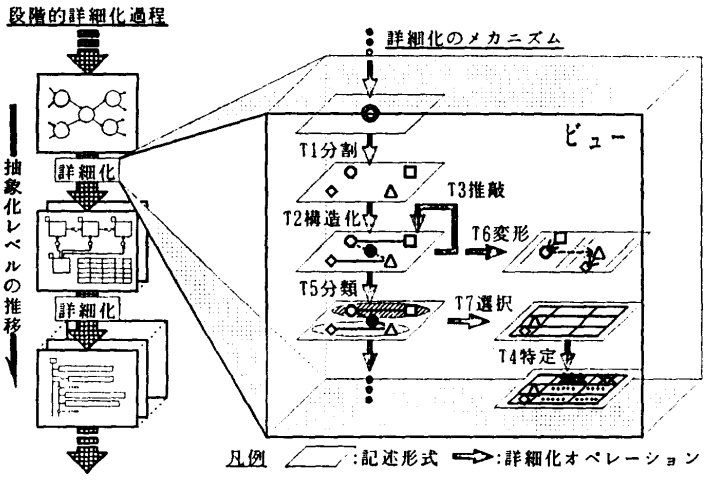


図 4 段階的詳細化操作の系列
Fig. 4 Combination of operations for stepwise refinement.

普通であるため，同一のビューにおいては一意な抽象化レベルで記述されるよう管理する必要がある．この観点から以下の支援が可能である．

① 各記述形式の構文規則に従って，構文誤りや属性値の型や値域の制約違反を可能な限り記述時にチェックする．

② 構造化された結果におけるグローバルな参照関係の矛盾や一貫性をチェックする．

③ 次の段階のビューへの移行に先立ち，現在のビューでの仕様記述項目の漏れなどをチェックすることにより，次の抽象化レベルの詳細化作業へ移行可能か否かを指摘する．

記述の抽象化レベルの一意性を厳密に保証するためには，さらに記述内容の意味付けとそれらの相互関係の解析も必要となるが，必ずしも形式的な仕様記述を前提としていない場合にはそれは困難である．そこで，本方式では，コンテキストによって期待する意味内容を示唆するよう記述形式群を配置することにより，記述の意味的な分散が極小となるよう作業者をガイドするというアプローチをとる．

（2） ビュー階層間での記述の依存関係の管理

各ビューにおける個々の部分の情報が，上位のビューのどの部分の記述からどのようなタイプの操作を経て獲得されるかをビューのメタな定義として保持することにより，以下が可能となる．

① ある段階の詳細化の完了後，次にどの段階のどのビューに移れば良いかを環境側から提示する．

② 次の段階の詳細化を開始するときに，上位のビューから継承する情報を自動的に挿入する．

③ 操作のタイプにより可能であれば，上位と下位の記述の間で情報の矛盾チェックや相互変換を行う．

④ 意味的な詳細化操作を介在する場合でも，記述された内容間の対応を識別できるため，仕様記述の変更の影響範囲を提示する．

以上のメカニズムを実現するため，設計支援ツールを開発した．さらに，本ツールを実際のソフトウェア開発に適用し，その有効性を評価した．

4. YACII指向設計支援システム：YDS

YDS は図3に示す5段階の詳細化過程の中で，タスク構成レベル以降の3段階を対象とした設計支援ツールであり，YACII によるモジュールレベルの設計に至る円滑な詳細化のガイダンスを目的としている．

YDS は**図5**に示すように，各ビューの記述形式に対応した設計書エディタ群，詳細化作業の進展に沿って必要な設計書エディタを起動するエディタ連携機能，および進捗管理などの補助機能から構成される．

以下では YDS の持つ各機能を紹介する．

4.1 設計書エディタ

設計書エディタは各ビューにおける記述の枠組みを提供するものであり，ビューを構成する記述形式に対応して連携動作する複数の要素画面（エディタ）から構成され，個々のビューの中での詳細化を支援する．**図6**に YDS を構成するエディタの例を示す．

ビューを構成する各要素エディタの果たす役割は以下の2種類に分類できる．

（1） 入力情報の抽出・整理の支援

前段階のビューに記述された情報から必要な情報を

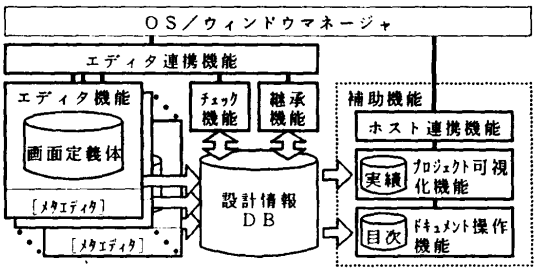


図 5 YDS の機能構成
Fig. 5 Functional configuration of YDS.

抽出・整理し，後段階のビューにおける詳細化対象項目を一覧できるようにする．ここで行われる作業は3.3節で述べた詳細化操作の変形（**タイプ6**）や選択（**タイプ7**）に対応するものであり，エディタは主に一覧表の形式から成っている．

（2） 設計者による決断の支援

各段階のビューにおいて(1)項で抽出された対象項目に基づいて詳細化の決断を行い，その結果を記述する．ここで行われる詳細化の決断とは分割（**タイプ1**），構造化（**タイプ2**），推敲（**タイプ3**），特定（**タイプ4**），および分類（**タイプ5**）といった意味的な操作であり，設計の中心的な作業である．このため，各エディタは3.2節で述べたような各ビューで詳細化されるべき内容に適合した図表形式から成っている．

これらのエディタの実現にはYACⅡエディタを除いて，構文駆動のメタエディタを共通基盤として採用している．このため，各エディタの操作方法を統一し，さらに記述情報をその論理構造に従ってデータベースに格納することができる．

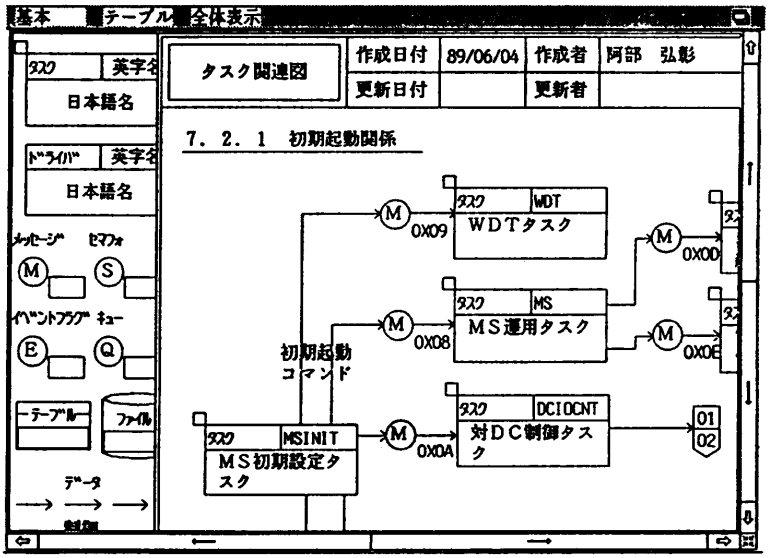


図6 YDS設計書エディタ例
Fig. 6 An example of YDS document editors.

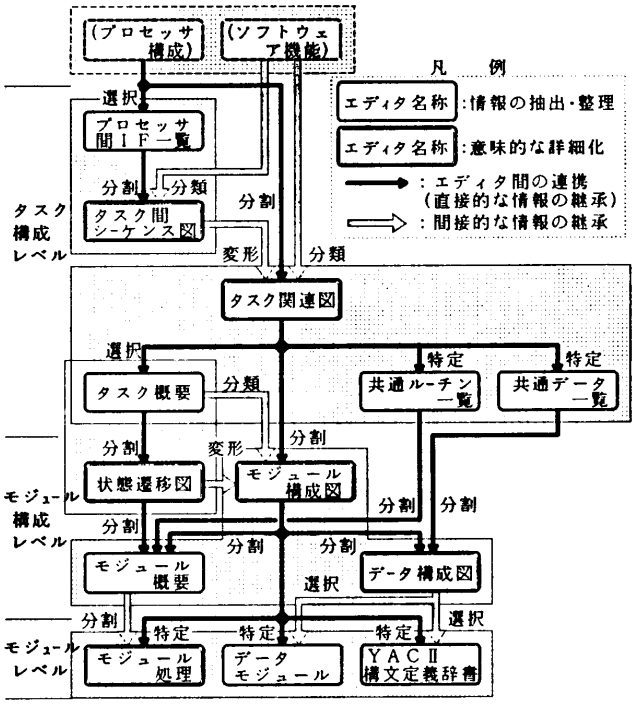


図7 YDSのエディタ間の連携
Fig. 7 Cooperative support for stepwise refinement by YDS editors.

4.2 エディタ連携機能

YDSでは**図7**に示すように，各エディタに記述された情報に基づいて関連する下位のエディタを起動することができる．起動の関係は上位のエディタに記述された項目について，より詳細に分割（**タイプ1**）したり，その属性値を特定したり（**タイプ4**）する方向である．このようにエディタ間の連携をとることにより，自然な詳細化の流れをガイダンスすることができる．

4.3 情報の継承とチェック機能

段階的詳細化をより効果的に支援するためには，上流で記述された情報を継承したり，記述の誤りを早い時点で指摘できることが重要である．このためYDSでは以下に示すような情報の継承やチェック機能をサポートする．

（1） 上位ビューからの情報の継承

4.1節(1)項で述べた上位ビューから下位ビューへの移行の際に，上位のエディタの記述内容から次の詳細化対象となるべき情報を変形や選択操作により抽出し，下位のエディタに自動的に継承することができる．このような関係には，直接親のエディタの情報を継承する場合と，上位ビューを構成するエディタから

間接的に継承する場合との2種類がある.

(2) エディタ起動時の継承

各エディタは連携する上位エディタからの起動時に連携のキーとなる名前を継承する.

(3) ビューにおける記述結果のチェック

これらはさらに2種類に分類でき,一つのビューにおける記述の抽象化レベルの一意性を保証すると共に,次のビューに進むか否かの判断基準となる.

① 一つのビューに属するエディタ群により記述された情報の中から,未参照や未定義の項目の有無をチェックする(一貫性).

② 上位のビューから選択・変形され継承された情報が詳細化後もすべて反映されていることをチェックする(トレーサビリティ).

(4) エディタ編集時のチェック

個々のエディタは構文規則を持つため,図形の接続規則や個々の文字フィールドに書かれるべき文字列の命名規約や型・桁数などの制約を記述時点でチェックすることができる.

(5) 修正の影響範囲のチェック

下位ビューに選択・変形して継承された情報とそれに基づいて詳細化された情報との間において,一方が変更されたならば影響の及ぶ範囲として他方を指摘することができる.

以上の機能の中で(1)項と(3)項のトレーサビリティのチェックおよび(5)項は現在開発中である.また,『なぜそのようにタスク分割をしたか』といった,分割操作や構造化操作などの本当にセマンティクスに関わる内容のチェックは人間に委ねている.

4.4 YDS の補助機能

YDSでは,実際の設計作業を補助するため,さらに以下の機能をサポートしている.

(1) ホスト連携機能

複数人による分散開発を支援するため,各エディタにより記述された情報はホストコンピュータ上で一括管理し,共有することができる.

(2) プロジェクト可視化機能

すべての設計情報は構造化されてデータベースに格納されるため,新規投入や修正の様子をモニタし,プロットすることにより作業の進捗を可視化できる.

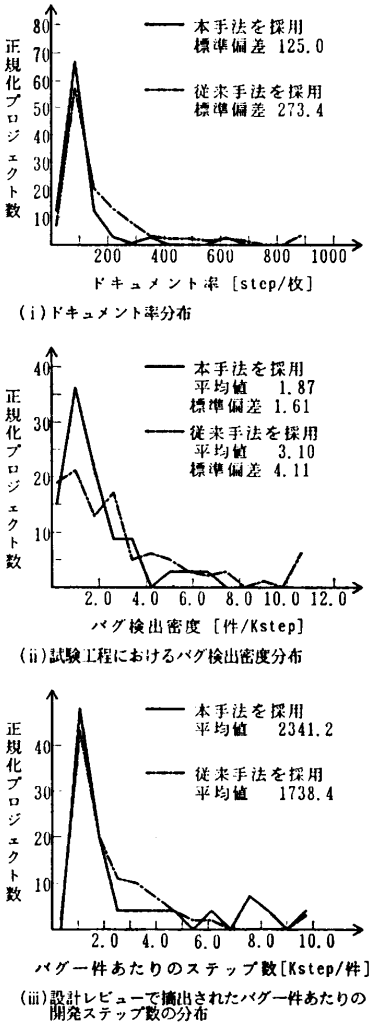
(3) ドキュメント操作機能

作成されたすべての設計書を目次管理しており,目次から直接更新したい設計書エディタを起動したり,印刷したい設計書を選択することができる.

5. 評価と今後の課題

5.1 評 価

図8に従来手法を用いたプロジェクト群と本手法を採用したプロジェクト群とを比較した3種類のデータを示す.(i)はドキュメント率を比較したもので,各々のプロジェクト群における設計ドキュメント1枚当たりのステップ数の分布を表している.(ii)は各々の試験工程での検出バグ密度の分布を比較したものである.また,(iii)は設計レビューで摘出したバグ1件当たりの開発ステップ数の分布を比較したものである.



(i)ドキュメント率分布

(ii)試験工程におけるバグ検出密度分布

(iii)設計レビューで摘出されたバグ一件あたりの開発ステップ数の分布

注) 比較のため,各図におけるプロジェクト数は100に正規化した.

図8 MRV手法と従来手法とを採用したプロジェクト間の比較

Fig. 8 Comparison between projects employing MRV and conventional approach.

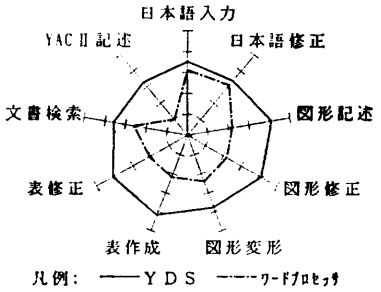


図 9 YDS の操作性評価（アンケート結果）
Fig. 9 A usability evaluation of YDS.

ここで，(i)ドキュメント率，(ii)試験工程でのバグ検出密度とも確かに標準偏差が小さくなっており，プロジェクト間でのばらつきの減少を示している．このことから，本方式によるガイダンス効果の有効性が認められる．

さらに，(iii)は設計工程レビューで摘出されたバグ件数の平均値の向上を示しており，本方式が上流工程での高品質の確保に有効であることが分かる．このことは，(ii)における試験工程での平均検出バグ件数の減少からも裏付けられる．

また，**図 9** は YDS と従来手法で用いていたワードプロセッサとについて，その操作性に関するユーザの評価を比較したものである．ドキュメント作成・更新の全項目で YDS が良好な評価を得ており，作業効率向上にも寄与していることが分かる．

5.2 今後の課題

上流工程を一貫した MRV アプローチにより支援するために，今後の課題として以下の点が挙げられる．

① 現在開発中の要求分析支援ツールと YDS との円滑な連携を図るうえで必須な，要求仕様から設計仕様への変換ガイダンスの実現．

② トップダウンの開発だけでなく，既存ソフトウェア資産の保守や新規システムへの再利用を支援するリバース方向のガイダンスの実現．

③ デバイスドライバなど，対象システム固有のアーキテクチャに依存するソフトウェアにも適用可能とするための，ガイダンスのカスタマイズ機能の実現．

6. おわりに

本稿では，ソフトウェア仕様の段階的詳細化の流れに着目したビューと呼ぶ概念によりその流れをガイダンスする手法である MRV アプローチについて述べ，その実現と効果を示した．

MRV に基づく開発環境では，ヒューマンファクタの影響を最小限に抑えることにより，ソフトウェア開発の上流工程における品質と生産性の飛躍的向上を達成することができる．

今後は，要求分析から設計へのガイダンスや既存資産からのガイダンスなどの課題の解決に向けて，順次検討を進める予定である．

謝辞 本研究に当たり多大な御支援を頂いた通信事業推進本部中村部長，有益な議論を頂いた鈴木課長，貴重なデータを提供頂いた菅原課長，ならびに YDS の開発評価に尽力頂いた川島公子氏，阿部弘彰氏他の関係各位に対し謝意を表します．

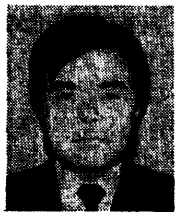
参考文献

1) Fujimoto, H., Toyoshima, Y., Fukao, I. and Kiryuu, S.: Visual Development Environment for Embedded Software, *36th ISMM*, pp. 224-227 (1988).
2) 深尾，西山，豊島，藤本：通信ソフトウェア開発用統合環境，CASE 環境シンポジウム論文集，pp. 73-80 (1989).
3) Ward, P. T. and Mellor, S. J.: *Structured Development for Real-Time Systems*, Yourdon Press, New Jersey (1985).
4) Hatley, D. and Pirbhai, I. A.: *Strategies for Real-Time System Specification*, Dorset House Publishing, New York (1987).
5) Harel, D. et al.: STATEMATE: A Working Environment for the Development of Complex Reactive Systems, *10th ICSE*, pp. 396-406 (1988).
6) Fujimoto, H., Suzuki, T., Katoh, H. and Yoshida, H.: Telecommunications Software Design Support System Based on Specification Languages, *EUROCON'84*, pp. 271-275 (1984).
7) 富士通(株)：FACOM SDEM YAC II 記述手引書 (1986).
8) 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).

(平成元年 10 月 2 日受付)
(平成 2 年 4 月 17 日採録)

藤本　洋（正会員）

昭和16年生．昭和40年日本大学理工学部電気工学科卒業．同年より東京大学生産技術研究所に勤務．昭和44年富士通(株)に入社．以来，通信システム（特に電子交換システム）のソフトウェア開発に従事．現在，ヒューマンベースのソフトウェア生産技術について興味を持つ．電子情報通信学会，IEEE 各会員．



豊島　康文

昭和25年生．昭和48年大阪大学工学部通信工学科卒業．同年富士通(株)に入社．以来，応用通信システムの開発に従事．昭和61年より，通信ソフトウェアの開発技術に関する研究開発に従事．現在，フジツウ・アメリカに勤務．HMI 技術および CASE 技術に興味を持つ．IEEE 会員．

西山　好雄（正会員）

昭和26年生．昭和49年大阪大学工学部通信工学科卒業．同年富士通(株)に入社．以来，無線システムの開発に従事．昭和62年より通信ソフトウェアの開開発技術に関する研究開発を行っている．CASE を始めとするソフトウェア統合開発環境に興味を持つ．電子情報通信学会会員．

深尾　至（正会員）

昭和32年生．昭和54年京都大学工学部情報工学科卒業．以来，応用通信システムの開発に従事．昭和62年より通信ソフトウェアの開発技術に関する研究開発を行っている．通信ソフトウェアの要求分析・仕様化および設計プロセスの支援技術に興味を持つ．AAAI 会員．