ＴＦＦによるプログラム開発の一貫支援

2U−8

今村 紀子，岩田 誠司

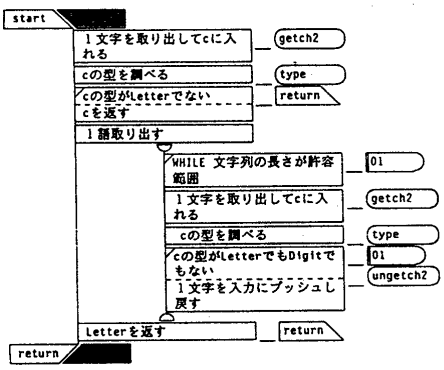
株式会社　東芝　システム・ソフトウェア技術研究所

１．はじめに

ソフトウェアの大規模化・複雑化に伴い，ソフトウェア・ライフサイクルの最終工程である「保守」における負荷が増大している。このソフトウェアの保守作業を効率的に進めるためには，ソフトウェアの内容を正確に示すドキュメントの整備が重要である。

そこで我々は，プログラム開発のモジュール詳細設計において設計書の記述形式を，当社で開発したモジュール設計記述法 ＴＦＦ (Technical description Formura for Fifty steps / module design)で統一し，ＴＦＦの中でもモジュールの処理フローをチャートで示すモジュール内部仕様書(図-1)の作成に着目した。

本稿では，保守の観点から見たモジュール内部仕様書の問題点と，それらの問題の解決を目的としたプログラム開発時の作業モデルについて述べる。



(図-1) モジュール内部仕様書 記述例

２．モジュール内部仕様書の問題点

まず，保守時にプログラムを理解する手順を，

1. モジュール詳細設計書を参照してモジュールの処理概要を把握する
2. より詳細な情報を得るためにソースコードを見る

と想定できる。この場合，モジュール内部仕様書に対して次のような問題点が挙げられる。

(1) 記述情報の不足

(2) ソースコードと内容の整合性がとれていない

(1)については，プログラム設計書が「保守」を意識して作成されていないため，記述の詳細にバラつきが生じる。例えば，モジュール内部仕様書(TFF ｼｰﾄ)を，ソースコードの自動生成を目的として記述する場合，その記述はソースコードの実行文数行程度のレベルの詳細さとなる(図-1)。この様なソースコードと等価な設計ドキュメントは理解性が悪く，保守時にあまり必要とされない。また逆に，記述内容の抽象度が高すぎると，ソースコードと設計書の対応が取れなくなり，設計書としての意味がなくなる。この様に，設計書の作成では適切な記述レベルの設定が課題となる。

一方，(2)の問題に関しては，ソースコードと設計書間の内容を相互に反映させるような適当な仕掛がないことが，大きな原因として挙げられる。手作業による設計書の修正は作業負担が大きく，これを解消するためには機械的な支援が必要である。

３．作業モデル

前述のようなプログラム設計書作成における問題点を解決するためには，以下のような支援が必要となる。

[1] プログラム設計書の記述レベルの設定

[2] プログラム設計書とソースコード間の内容を相互に反映させる機械的支援

そこで，これらの支援を考慮した，プログラム開発時の基本的な枠組みとなる作業モデルを提案した(図-2)。

以下，本作業モデル中の特徴的な支援内容について述べる。

３．１ 記述レベル

保守時に参照されることを前提に，モジュール内部仕様書を作成すると考えると，モジュール内部仕様書にはモジュールの処理フローの概要がつかめるレベルの記述を行う必要がある。記述レベルは，各分野で保守時に必要な情報がどの様なものであるかにより決められると考えられるが，今回は，モジュール内部仕様書のボックス内の記述レベルに対して，

・プリミティブな関数に置き換えられるレベル

・ソースコードとの関係を一義的に定められるレベル

といった記述指針に関する規定を設けた。これにより，設計者間のドキュメント記述レベルの統一を図る。

３．２ ツール支援

機械的な支援に関しては，以下のようなモジュール詳細設計書とソースコードの内容を相互に反映させるツールを開発し，作業モデル中で運用する。

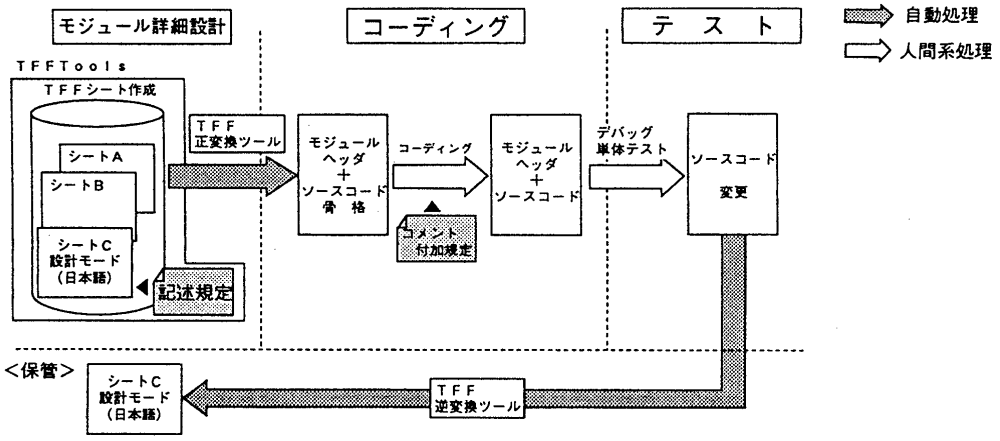
【ＴＦＦ正変換ツール】

ＴＦＦ正変換ツールはモジュール内部仕様書の処理概要の情報をコメントとしてソースコード上に埋め込む機能を持つ。この機能により，モジュール詳細設計書の設計内容をソースコード上に反映させることが可能となる。

" The consistent support of program development with TFF "

Noriko Imamura, Seiji Iwata

Systems & Software Engineering Laboratory, TOSHIBA Corporation



(図-2) 作業モデル

【ＴＦＦ逆変換ツール 】

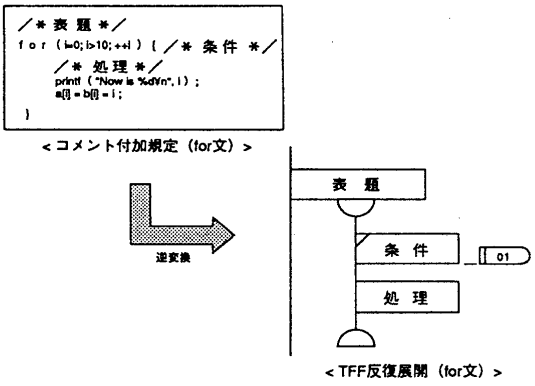
ＴＦＦ逆変換ツールは，ソースコードのコメント情報より，モジュール内部仕様書(TFF Cシート)を生成する。この様なリバースツールは従来より存在するが，それらの多くはソースコードと同等のドキュメントを生成し，その理解性は悪い。そこで今回は，前述の記述指針と同等の抽象度を持つ設計書の生成を目的とした。そのため，逆生成される処理フローの詳細さを，コメントを付加する者の意図に基づいて設定することが可能な逆変換ツールとした。

本ツールでは，ソースコードのコメントは，

・処理のまとまり毎に記述されている

・コメント付加規定に従って付加されている

と捉える。ここでのコメント付加規定は，ＴＦＦ正変換の変換ルールに準拠するもので，コーディング時にソースコード作成者が遵守する(図-3)。そして，これらのコメントとソースコードの構造を解釈して，モジュールの処理概要を示すフローが生成できるような変換方式とした。



(図-3) コメント付加規定と逆変換

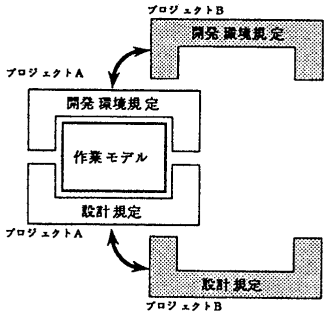
４．作業モデルの拡張性

対象ドメインによっては保守時に必要な情報が今回とは異なり，それにより設計書の記述レベルも異なると考えられる。また，前述の作業モデルを適用してプログラム開発を行う場合は，開発環境についてもプロジェクト内で規定する必要がある。この様に，作業モデルを実際運用するに当たっては，分野やプロジェクトに依存した設計上の詳細手順や注意事項，開発環境規定について，図-4で示すように個々に策定する必要がある。運用における詳細規定に関しては，手順書や規定書などのマニュアルを作成し，運用する。これにより，

・作業モデルの適用促進

・設計者間での作業手順，環境の統一化

が図られると考えられる。



(図-4) 作業モデルの拡張性

一方，作業モデルにおける作業手順や機械的な支援部分は汎用的であるため，プロジェクトに依存することなく適用が可能である。よって，分野やプロジェクトに依存する部分をカスタマイズすることにより，他分野，他プロジェクトに対して適用拡張が可能となる。

５．今後の展開

以上述べたような作業モデルを適用することにより，

・プログラム開発作業の標準化

・保守の作業効率向上

等の効果が期待される。また現在，本作業モデルは事務処理計算機システムのＣ言語プログラムの開発において適用中である。更に今後は，他システムへ適用を拡張し，モデルの妥当性，ツールの機能の充分性を検討する。

【参考文献】

・松村 他：「ソフトウェア設計記述技法」，東芝レビュー VOL.41 NO.8, 1986