グループウェア 20－6
（1996. 10. 3）

ソフトウェア開発プロセス支援システム SOFTPIE の開発

山本里枝子, 上原忠弘, 中山裕子, 吉田裕之
lisun@flab.fujitsu.co.jp
富士通研究所 ソフトウェア研究部
〒211-88 川崎市中原区上小田中4丁目1番1号

我々は, 非定型な並行協同開発作業が中心であるソフトウェア開発技術に, 定型作業を支援するワークフローシステムを適用する技術を開発し, その技術を元にしたプロセス支援システム SOFTPIE を開発した. SOFTPIE では, 開発ドキュメント管理機構で管理される複数のドキュメントに対して, 個々にワークフローを対応づけ, ドキュメント間の関連づけと, それらの開発状況を制御する. これにより, 非定型で並行に実施される協同開発作業の実際的なプロセス支援を可能とする. 本稿では, オブジェクト指向開発方法論を適用して行なったプロセス支援システム SOFTPIE の開発について報告する.

Development of Software Development Process Support System SOFTPIE

Rieko YAMAMOTO,Tadahiro UEHARA,Yuko Nakayama,Hiroyuki YOSHIDA

Software Laboratory, Fujitsu Laboratories Ltd.
1-1,Kamikodanaka 4-chome, Nakahara-ku, Kawasaki 211-88, Japan

We developed the new technique to apply workflow systems to software development supporting technologies and a software development process supported system SOFTPIE using this technique. Software development is a unfixed form activity, while workflow systems can only support a fixed form activity. We provides new facility, named the document management facility, to manage many documents information and relationships between these ones. Using this facility, SOFTPIE creates automatically a workflow data corresponding to each document, displays and controls the progress situation about each documents using the workflow data, and suggests which documents needs to create or modify in a situation. SOFTPIE enables practical development process support apply to unfixed collaborating development activity. This paper describes about development of the software development process supprot system SOFTPIE using a object oriented methodology.

1 背景

Osterweil[1] がプロセスプログラミングという概念を提唱して以来，ソフトウェア開発のプロセスをモデル化し，形式的に記述する研究が盛んに行なわれた．一方，CMM(Capability Maturity Model), ISO9000, SPICE(Software Process Improvement and Capability dEtermination) などのソフトウェアの品質保証の枠組が提案されており，契約取り引きに認証を必要する等の社会的な動きがある [2][3]. これら枠組の記述方法の提案と枠組間の比較が報告される [4] 等，ソフトウェアプロセスの技術をソフトウェア開発現場の実務作業へ適用することに期待が高まっている．しかし，これまで提案されたプロセスモデルやその環境には，市場に広く受け入れられた例が殆んどない．

我々は，実際のソフトウェア開発現場へのソフトウェアプロセスの導入を目的に，ソフトウェア開発環境においてプロセス管理ツールはそれと同様なサービスを実現する既存ツールと連係を図るべきであるという立場を提案し，プロセス管理ツールとプロジェクト管理ツールの連携の実現を報告した [5]. さらに，実際に開発現場にプロセス支援を導入する際の問題点を明らかし，解決策として，プロダクト開発を動的に計画し管理する枠組を導入し，各プロダクト開発を支援するプロセスを動的かつ並行に制御する方式，及び，プロセスモデルを解釈するエンジンとして既にビジネス市場で広く受け入れられているワークフローシステムを用いる実現を提案した [6]. SOFTPIE (SOFTware Process Inprovement Environment) は，この提案を基に開発したソフトウェア開発プロセス支援システムである．

本稿では，SOFTPIE の概要と，オブジェクト指向開発方法論を適用して行なった試作に関して報告する．

2 SOFTPIE の概要

SOFTPIE は，数十人規模の開発部隊によるソフトウェア開発作業を支援することを目的としている．大規模プロジェクト全体の管理は既存のプロジェクト管理ツールにより行なう．一般に，プロジェクト管理ツールの工程管理機能は，工程を 3 レベル程度に詳細化して管理するが，SOFTPIE はその一番詳細なレベルの作業の支援と管理を対象としている．プロジェクト管理ツールとはデータ変換により連携を図る方式を提案した [5]. マクロな進捗の把握をプロジェクト管理ツールで，ミクロなプロジェクト管理を SOFTPIE で，それぞれ分担することを想定している．

ミクロなソフトウェア開発の大部分は非定型な並行共同開発作業であり，次の理由によりワークフローの適用が困難である．

- 最初から最後まで作業構成をあらかじめ厳密に定義することが困難．
  - 前の作業結果により次の作業構成が決まる．仕様書が完成して初めてモジュール設計書をいくつ作成すれば良いかが決まる．
- 作業の流れを強制することが困難．
  - 前の作業 (例えば設計書のレビュー) が終了しなくても，次の作業 (プログラミング) を開始する．
  - 前の作業をやり直し (プログラミング中に設計書を修正する) が頻発．

また，並行していくつも実行されている作業全体を把握することも困難であった．

我々は，開発ドキュメント管理機構の下でワークフローシステムを制御する新しいプロセス支援技術を実現した．図 1にこの技術を基にした SOFTPIE の概要を示す．開発ドキュメント管理機構は，多数の開発ドキュメント間の関係づけを管理し，それらの開発状況を制御する．ワークフローシステムは，一つのドキュメント毎の開発作業の流れ (ワークフロー) を支援する．多数の開発ドキュメントに対して，これらのワークフローが並行に実行され，開発ドキュメント管理機構がそれを制御する．

SOFTPIE は，仕様書，プログラムソース，テストスーツなどの各種ドキュメント（プロダクト）の開発状況という視点でプロジェクトの進捗を管理することを特徴とする．通常のソフトウェア開発ではミクロな作業工程よりもドキュメントの完成度で開発作業の進捗を管理することが多いためである．そして，複数のドキュメントの作業状況を取りまとめる開発ドキュメント管理機構を導入し，一開発グループの一工程程度の規模を想定した「作業計画」に関連するプロダクトの開発状況の管理を支援する．

この技術の特徴は，ワークフローを従来通り適用した場合は作業の順序関係を強制するのに対し，作業の順序づけを開発ドキュメント間の関係に基づいて，緩やかに制御することにある．開発ドキュメント管理機構のユーザインタフェースを通じて，以下が可能となった．

- ユーザはドキュメント間の依存関係として作業の順序関係を定義できる．

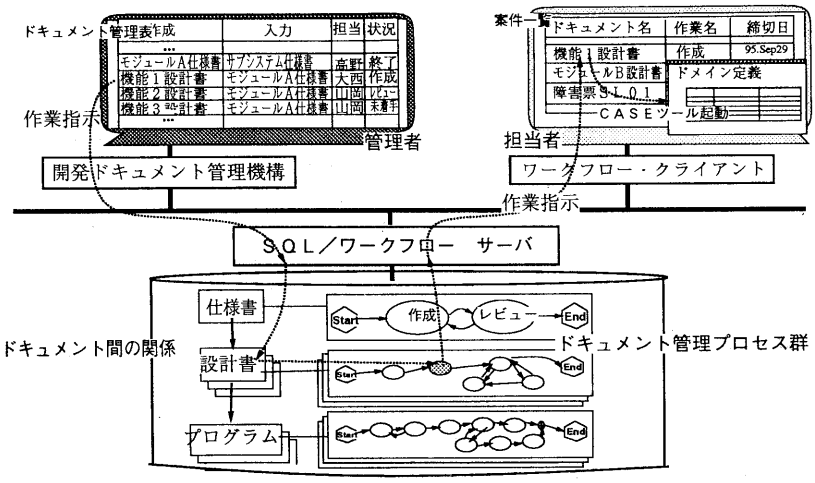

図 1: SOFTPIE の概要

- ある開発ドキュメントの作成作業を先行して始めることも指示できる.
- プロジェクト（作業計画）の進捗状況は，開発ドキュメントの作業状況一覧として把握できる.

そして, ドキュメント間に依存関係, 参照関係を設定することによりドキュメント開発の流れを穏やかに規定することができる. ドキュメントが開発完了または修正完了となると, それに依存するドキュメントの開発, 修正を提言する機能を有する.

3 オブジェクト指向開発手法の適用

我々は, SOFTPIE の開発にオブジェクト開発方法論を適用した. 大枠を OMT[7] をベースとした統一方法論 [8][9][10]（提案の仮称. 現在は，モデリング言語の提案のみ）に従い, 個別の開発技術の一部であるユースケース [11] やデザインパターン [12][13] を参考として SOFTPIE の基本部分の試作を行なった. 以降は Rumbaugh が示した作業の大枠にしたがって結果を報告する.

3.1 分析結果

ユーザとの要件獲得のコミュニケーション手段として, ユースケースを用いた. アクタは作業担当者と作業管理者の 2 つ, ユースケースは 9 個を抽出した. 作業担当者を主なアクタとして,「ワークリストを確認」「ドキュメント開発の作業情報を確認」「ファイルを編集」など, 作業管理者を主なアクタとして,「作業計画を新規作成」「作業計画を変更」など. さらに, これらユースケースの共通部分を別ユースケースに抽出して, 最終的に 1 6 個とし, これらについて Jacobson の提案を基に, 主に「シナリオ」と呼ばれる部分に要件を定義した.「作業計画を変更」の単純化したシナリオ例を以下に示す. これらユースケースとともに, これらの語彙を定義する辞書を作成した.

1. ドキュメント群の情報を確認する／変更する.
   - その一覧表に関して, 名前, 作業管理者の名前, プロジェクト名／システム名／サブシステム名等を確認する／変更する.
   - 各ドキュメントについて, 名前, 種別, プロセステンプレートの名前, 作成作業の担当者, スタート／締切日時, 現在の状況を確認する.
   - 指定したドキュメントの作業情報を確認する.
   - 指定したドキュメントの作業履歴を確認する.
   - 各ドキュメントについて, 現在の状況は自動的に変更される.「スタート日時」は自動的に設定される.
   - ドキュメントを追加する, 削除する.
   - ドキュメントを指定し, 参照するドキュメントを設定する.

- ドキュメントを指定し，その行に対応するワークフローの各作業ステップのロールに割り当てられた作業者名を確認し変更する．

2. 実行する．
   - 依存関係に基づき，作成／変更作業を行なうべきドキュメントが提案される．自動的に行なわれる．
   - 作成／変更作業を提案されたドキュメントを選択する．選択したドキュメントに関して作業指示する．

ユースケースから，SOFTPIE の対象世界のモデル（ドメインモデル）を作成した．作業計画に関連するオブジェクトを図 2に単純化して示す．

SOFTPIE では，開発ドキュメント類にあわせて障害票などの帳票類をも扱えるように，ドキュメントのスーパクラスとして「ファイル」を，作業計画のスーパクラスとして「作業管理単位」を定義している．「作業計画」は複数の「ドキュメント」を有する．「ファイル」はそのファイルの開発手順を示す「プロセス」と一対一に対応づけられる．これにより個々のドキュメントの開発を独立に進める事ができる．「プロセス」はいくつかの開発「作業」で構成されており，現在行うべき作業が担当者の「ワークリスト」に対応づけられる．「ファイル」は複数の「参照」を持つことができる．「参照」は属性「依存？」をもつ．もし，「依存？」が真ならば，「参照」に「対象」で関連づけられている「ファイル」は元のファイルに依存されているファイルである．

「作業計画」がドキュメントの作業状況を取りまとめる．「作業計画」は、各ドキュメント毎に依存しているドキュメントの作業状況を調べ，作業を行うべきドキュメントの提言を行う．SOFTPIE は複数人数での共同作業を支援するために，ファイル情報を共有する必要があり，情報の編集における排他制御が必要となる．そこで「ファイル」に「編集開始」「編集終了」の操作を付加する．

ドメインモデルを作成後，各ユースケースの実現するためのアプリケーションオブジェクトを抽出し，ユースケースのシナリオに基づきドメインオブジェクトとアプリケーションオブジェクトとのインタラクション例を作成して，ドメインオブジェクトの妥当性を検証した．

3.2 設計

設計では他システム，インフラの利用などを考慮して分析モデルを詳細化する．

まず，SOFTPIE のシステム構成について 3に示す．SOFTPIE のワークフローエンジンに当社製品である TeamWARE Flow[14] を用いる．TeamWARE Flow はクライアント・サーバ型のシステム上でワークフローを管理，自動化する．

よって SOFTPIE はクライアントサーバ型のシステムである．サーバ側には TeamWARE Flow サーバと SQLServer を用いてそれぞれワークフロー情報とドキュメント管理情報を管理する．クライアント側は TeamWARE Flow クライアント API，SQLServer クライアント API を用いて SOFTPIE クライアントを構築する．SOFTPIE クライアントは C++ の本体と VisualBasic の GUI 部からなる．

つぎに，このシステム構成を考慮して，分析時のオブジェクトモデル図を詳細化するオブジェクト設計，永続化実現のために利用する RDBMS のスキーマ設計 [15]，GUI とつなぐための本体部分の API 設計を行なった．

設計時のオブジェクトモデル図の簡略版を図 4に示す．

新たに 2 系統のクラスが追加されている（太線のクラス）．一つは TeamWARE Flow から提供されるクラス（RC**）である．これらを分析により得られたクラスに対応づけ，提供される機能を利用するように設計する．ドキュメント情報の編集における排他制御については，TeamWARE Flow の各プロセス単位での排他制御機能を利用するため，「ファイル」にあった操作「編集開始」「編集終了」を「プロセス」に移し，点線のクラスのオブジェクトを変更する時は「プロセス」のこれらの操作を実行してから行うよう設計した．これによりドキュメント情報の編集に関する排他制御が実現される．

もう一系統は永続性を保証すために追加されたクラス [15] である．永続記憶として RDB を用いるが，RDB との通信を行うクラス（RDBInterface）と特定のクラスのオブジェクトと RDB 内の情報との変換を行うクラス（**DataStore）を導入する．これにより RDB 固有の記述を隠蔽することができる．

また，これらのオブジェクト設計に際して，Singleton パターン，Template Method パターンなど [12] の一般的なデザインパターンを導入し，また，RDBMS の利用に関するパターン [11][13] を参考とした．

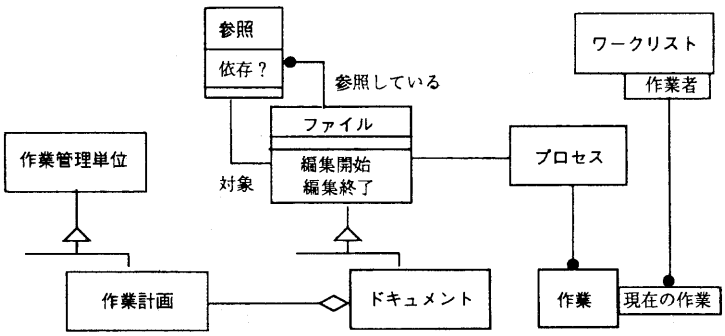

図 2: 分析結果の例

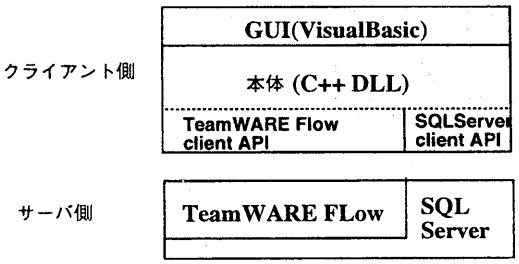

図 3: システム構成

4 まとめ

我々は，開発ドキュメント管理機構の下でワークフローシステムを制御する新しいプロセス支援技術を実現した．開発ドキュメント管理機構は，多数の開発ドキュメント間の関係づけを管理し，それらの開発状況を制御する．ワークフローシステムは，一つのドキュメント毎の開発作業の流れ (ワークフロー) を支援する．多数の開発ドキュメントに対して，これらのワークフローが並行に実行され，開発ドキュメント管理機構がそれを制御する．

本技術に基づいてソフトウェア開発プロセス支援システム SOFTPIE を開発した．開発手法にオブジェクト指向技術を適用し，分析結果と設計結果について，「作業計画」に関連する部分の概略を報告した．ワークフローシステムに TeamWARE Flow を利用した SOFTPIE の試作に際し，38 のクラス，約 200 の関数，約 150 の VisualBasic への API を作成した．うち，設計時に新たに加えられたクラス数は 17 である．また TeamWARE Flow client API のクラスを 10 クラス利用した．

今後は，ソフトウェア開発における SOFTPIE の有用性の検証および検証方法の検討を通じて，ソフトウェアプロセス導入による作業効率の向上を確認し，組織のプロセス改善を図る試みへと発展させていく．

参考文献

[1] Osterweil, L. J.: Software Process are Software Too, *ACM Trans. Comput. Syst.*, pp. 2–13 (1987).

[2] 飯塚悦功：ソフトウェア品質保証モデル ISO9000-3 の意義, 情報処理学会ソフトウェアプロセス・シンポジウム, pp. 107–114 (May).

[3] 堀田勝美：プロセス成熟度モデル, 情報処理学会ソフトウェアプロセス・シンポジウム, pp. 115–122 (1994).

[4] Inoue, K., Watanabe, A., Iida, H. and Torii, K.: Modeling Method for Management Process and its Application to CMM and ISO9000-3, in *Proceeding of the 3rd International Conference on the Software Process*, pp. 85–98,

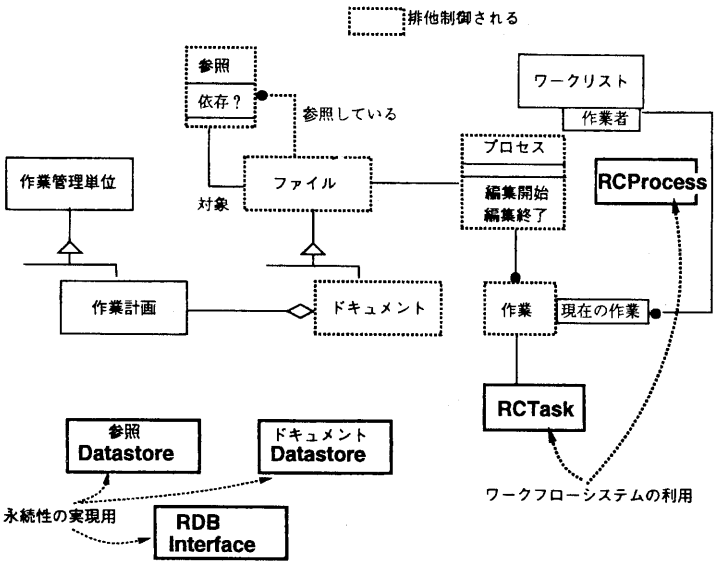

図 4: 設計結果の例

Reston (1994).

[5] 山本里枝子, 吉田裕之：プロジェクト管理ツールとプロセス管理ツールの連携の実現, 情報処理学会 ソフトウェア工学研究会, pp. 147–152 (1995).

[6] 山本里枝子, 上原忠広, 森偉作, 吉田裕之：プロダクトベース・プロセス支援の提案, 情報処理学会 ソフトウェア工学研究会, pp. 33–40 (1996).

[7] E.Rumbaugh, J., R.Blaha, M. and J.Premerlani, W.: *Object-Oriented Modeling and Design*, Prentice Hall (1991), [羽生田栄一監訳: 「オブジェクト指向方法論 OMT」, トッパン, 1992].

[8] Rumbaugh, J.: OMT:The development process, *Journal of Object-Oriented Programming*, Vol. 8, No. 2, pp. 8–76 (1995).

[9] Rumbaugh, J.: OMT:The object model, *Journal of Object-Oriented Programming*, Vol. 7, No. 8, pp. 21–27 (1995).

[10] Rumbaugh, J.: OMT:The dynamic model, *Journal of Object-Oriented Programming*, Vol. 7, No. 9, pp. 6–14 (1995).

[11] Jacobson, I., Christerson, M., Jonsson, P. and Overgaard, G.: *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley (1992), [西岡利博／渡辺克宏／梶原清彦監訳: オブジェクト指向ソフトウェア工学 OOSE − use-case によるアプローチー, トッパン, 1995].

[12] Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design Patterns : Elements of Reusable Object-Oriented Software*, Addison-Wesley (1995), [本位田 真一／吉田 和樹監訳: オブジェクト指向における再利用のためのデザインパターン, SOFTBANK, 1995].

[13] Yourdon, E., Whitehead, K., Thoman, J., Oppel, K. and Nevermann, P.: *Mainstream Object: An Analysis and Design Approach for Business*, YOURDON Press (1995).

[14] 富士通： TeamWARE Flow 使用手引書 (1995).

[15] 山本里枝子, 上原忠広, 中山裕子, 吉田裕之：オブジェクト指向開発における RDBMS の利用, 青山幹雄, 深澤良彰（編）, オブジェクト指向最前線（情報処理学会 オブジェクト 指向 96 シンポジウム）, pp. 155–158, 朝倉書店 (1996).