

ワークフロー処理アプリケーション構築の柔軟性向上*

2N-2

綾野雅之 川崎隆二†
NTT ソフトウェア研究所‡

1 はじめに

イントラネットを活用したワークフローシステムが注目されている。我々はこうしたシステムとして、BOW(Business flow On the Web)を開発している。

BOWは組織内の帳票決裁処理など、比較的単純な(子プロセスの生成や、プロセス間同期が発生しない)ワークフローを対象としたワークフローエンジンである。

我々はBOWの設計において、ワークフローエンジンとアプリケーション間のAPIを単純化することで、アプリケーション構築を容易にすることを試みた。

本稿ではBOWのアプリケーションAPIの概要を報告し、その適用性の評価結果について述べる。

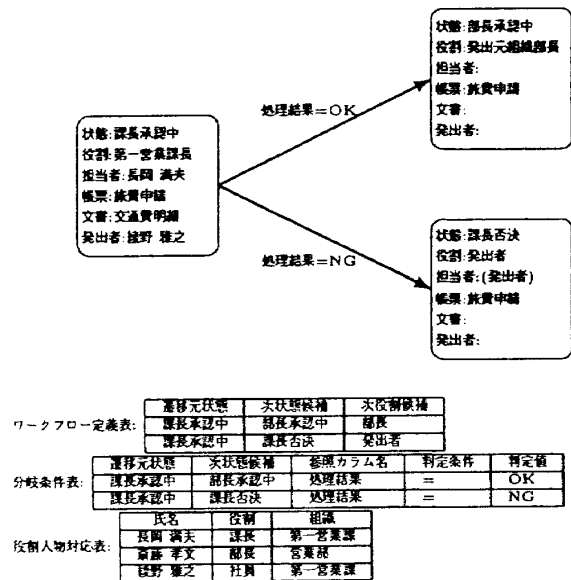


図 1: ワークフロー定義の例

2 BOW

BOWでは、ワークフローを状態の遷移として定義する(図1)。各状態には処理を担当する役割が定義され、役割の該当者が処理を行うと次の状態に遷移する。

また、BOWはFORCモデルに基づいて構成されている。FORCモデルでは「フロー」「組織」「帳票」「文書」を独立した要素として表現するため、ワークフローの変更管理が容易な事の特徴とする。[1][2]

3 ワークフロー API

BOWを利用したワークフローアプリケーションの構築を容易にすることをねらいとして、11種類のAPIを用意した。

表1はBOWのAPIの一部をWfMCのWAPI[3]と比較したものである。63種類からなるWfMCのAPIに対し、BOWのAPI数は約1/6である。

このAPIによる、標準的な処理手順を示す。

1. 帳票作成 BowFormInit(*UserID*, *WfType*, *Title*)
回覧する帳票を新規作成する。

*Improved Flexibility of the Workflow System

†Masayuki Ayano, Ryuji Kawasaki

‡NTT Software Laboratories

2. 帳票取得 BowFormlist(*UserID*)
自分に回覧されている帳票の一覧を表示し、選択する。
3. 処理の実行
アプリケーションにより決裁などの処理を行い、結果をテーブルに記録する。
4. 帳票の回覧 BowFormDispatch(*UserID*, *FormID*)
帳票の状態を遷移させ、次に処理すべき役割を取得する。
5. 役割具体化 BowGetExecutor(*Role*, *FormID*)
役割に該当する個人を検索する。
6. 処理者指定
BowSetExecutor(*UserID*, *Role*, *FormID*)
帳票を処理する個人を指定する。

BOWは、アプリケーションから利用者ID(*UserID*)、帳票番号(*FormID*)などを受け取る。次状態、役割、処理者などを決定し、帳票の状態が記録されたテーブルを更新する。

表 1: API の比較

モデル	フロー制御	リスト取得	ワークアイテム	管理	セッション
BOW	FormInit FormDispatch	Formlist	SeniDecision	—	WebArc を利用
WAPI	StartProcess CreateProcessInstance	OpenProcessInstancesList GetProcessInstance	GetWorkItem CompleteWorkItem	ChangeProcessInstances AbortProcessInstances	Connect Disconnect

表 2: 適用アプリケーション

研究成果管理システム

研究成果を外部に発表する手続きを、電子化したものである。

- 役割から、一意の処理者が自動的に決まるフロー
- 書類の記載内容により決裁ルートが異なる

ヘルプデスク支援システム

顧客からの問い合わせに対応するヘルプデスクを想定したアプリケーションである。知識レベルによりグループ分けされた担当者間で質問表を回覧する。

- 処理者を特定せず、役割宛に回覧するフロー

論文投稿受付システム

論文の投稿受付業務のためのアプリケーションである。

- 役割の該当者から、任意の1名を選択するフロー
- 処理者を特定せず、役割宛に回覧するフロー
- 役割から、一意の処理者が自動的に決まるフロー

表 3: アプリケーション開発規模

アプリケーション	項目	規模
研究成果管理	状態数	20
	役割数	8
	スクリプト (KStep)	6.1
ヘルプデスク支援	状態数	5
	役割数	3
	スクリプト (KStep)	6.2
論文投稿受付	状態数	11
	役割数	5
	スクリプト (KStep)	3.5

4 適用例

ワークフローエンジンの API を適用して、表 2 のアプリケーションを試作した。これらは BASIC 相当の制御構文と SQL、HTML が記述できる WebBASE スクリプト [4] で記述した。

アプリケーション中のスクリプト文は、モジュールの呼び出しや変数の受け渡し、画面表示用のデータ作成などで構成され、ワークフローに関する処理を直接記述する必要はない。

作成したアプリケーションのデータを表 3 に示す。

5 考察

異なるワークフロー制御を持つ、3 タイプのアプリケーションに API を適用して記述実験を行った。結果、API の機能上の不足による問題は生じなかった。これにより、単純化した API でも十分な記述力がある事、アプリケーション構築が容易な事を確認した。

これは、BOW ではアプリケーションテーブル上のデータをワークフローの状態遷移条件として記述できるため、ワークフロー API によりアプリケーションデータを扱う必要がなく、API を単純化できたことの効果のあらわれである。

その反面、アプリケーションのデータで変化するフローはテーブルに依存関係が発生し、アプリケーションとワークフロー定義との分離が十分なされない問題がある。しかし、アプリケーションのデータを利用する限り何らかの依存関係が生じることは避けられず、API の仕様として重要な問題でないと考える。

6 おわりに

今後はこのモジュールを関数ライブラリとして実装し、さらに幅広いアプリケーションに対応可能とする。また、実際に運用されるシステムに適用し、ワークフローの処理能力などを検証したい。

参考文献

- [1] 神谷, 滝野, 鳥巢, 綾野, 山本: “ワークフロー管理システムの WWW への適応”, 第 55 回情報処理学会全国大会論文集 4-179, 1997.
- [2] 神谷, 滝野, 山本: “WWW を用いたワークフロー管理システムに関する変更容易性の評価”, 電子情報通信学会技報, KBSE97-13, 1997.
- [3] WfMC: “Workflow Client Application Application Programming Interface (WAPI)”, WfMC-TC-1009, 1997.
- [4] “高機能 WWW サーバ WebBASE”, <http://robin.sl.cae.ntt.co.jp/WebBASE/>, 1997.