協調ワークフロー制御エンジン WolF におけるワークフロー制御方式

4S－3

小池和弘　中野初美
三菱電機(株)　情報システム研究所

1. はじめに

我々はワークグループにおける協調作業の効率化を支援することを目的として協調ワークフロー管理システム Cooper[1] を研究開発している。WolF はワークフローを制御／管理する汎用エンジンであり、Cooper の心臓部である(図1参照)。ここでワークフローとはワークグループにおいてメンバ同士が情報の交換、配送を行ないながら協調的にある業務を処理していくこと、あるいはその業務の流れのことをいう。プロトタイプ版での WolF は、もっともシンプルなシリアルワークフロー(条件分岐などがなく1方向にフローしていくもの)の制御のみサポートしていたが、条件分岐など、より複雑なワークフローに対応し、かつワークフローのルーティング定義を容易におこなうため、制約指向に基づくワークフロー制御方式を研究している。

本稿では、WolF におけるワークフロー制御の特徴である、**制約指向ワークフロー制御言語:WFL** とその制約評価機構のメカニズムについて説明する。

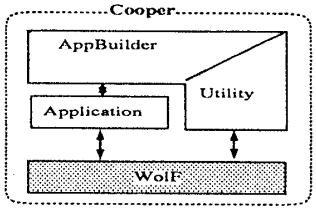

図1 Cooper における WolF の位置付け

2. ワークフローの考え方

WolF では、ワークフローを4種類の要素で表現している。すなわち(1)ワークフローの主題である問題名(業務名)、(2)その問題解決にあたるメンバと、そのワークフローにおけるメンバの役割や権限の定義、(3)ワークフローがいくつかの場面(以下ステージと呼ぶ)で構成されており、現在のステージから次のステージへ遷移するための遷移条件を定義したステージフロー定義、(4)問題解決に必要な情報や解決手段そしてアウトプットデータが格納されるデータ部、の4種類の情報である。これらの**問題**、メンバ**定義**、ステージフロー、データを1つにパッケージ化したものをワークフローオブジェクトと呼び WolF の処理対象の基本ユニットとしている(図2参照)。

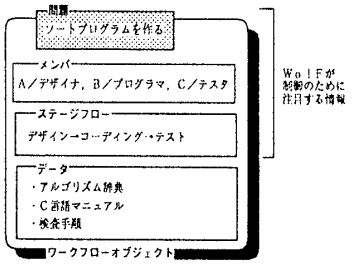

図2 ワークフローオブジェクトの構成

ここで次のような例で考えてみる。問題として、「ソートプログラムを作成する」が与えられたとする。これを解決するための情報やリソースとしてアルゴリズム辞典、C言語マニュアル、検査手順書などがあたえられ、この問題解決にあたるメンバとしてA、B、Cの3人が割り当てられた。3人の役割はそれぞれデザイナ,プログラマ,テスタである。ステージはデザイン,コーディング,テストの3つであり、各ステージでのステージ遷移条件は、デザイン終了,コーディング終了,テスト合格である。ここでテスト不合格となった場合は後向きにステージ遷移する。テスト合格すればこのワークフローは終了する。ワークフローが開始されるとA、B、Cはそれぞれの知識を駆使し問題解決にあたる。メンバはワークフローの全体的な流れをあまり意識する必要はなく、自分の担当業務に専念すれば良い。担当分が終了したらENDメッセージをワークフローオブジェクトに送る。次のステージへの遷移は WolF が自動的に行なう。次節以降では、上記の概念を基にしたワークフロー制御言語とその評価機構のメカニズム、オブジェクトの状態遷移を上記例に沿って説明していく。

3. 制約指向ワークフロー制御言語:WFL

WFL(WorkFlow Control Language) の syntax を以下に示す。

```
<WorkFlow>      ::= <WFName>(<Members>,<StageFlow>,
                                        <Data>)
<Members>       ::= {<Mem1>,<Mem2>,...,<Memn>}
<Mem>           ::= {<User>/<Role>,<AccessRight>}
<Data>          ::= {<Obj1>,<Obj2>,...,<Objn>}
<StageFlow>     ::= {<+SF1>,<+SF2>,...,<+SFn>}
<+SF>           ::= <Constraint> => <StageChange>
<-SF>           ::= <Constraint> => <StageChange>,
                                    add(Ci),del(Cj)
<Constraint>    ::= {<C1>,<C2>,<C3>,....,<Cn>}
```

The Control Method of Workflow in WolF
Kazuhiro.KOIKE, Hatsumi.NAKANO
Computer & Information Systems Laboratory, MITSUBISHI Electric Corporation, 5-1-1 OFUNA,KAMAKURA,KANAGAWA 247,JAPAN

```
<StageChange> ::= <CurStage> -> <NextStage>
<Data>        ::= {<D1>,<D2>,<D3>,....,<Dn>}
```

上記で"(+-)SF" がステージ遷移条件とステージ遷移情報を定義した部分である。この定義で"= >" オペレータの左辺の"Constraint" が条件部、右辺の"StageChange" がステージ遷移の実行部になる。また +SF はワークフロー終了の方向に働く定義であり、- SF は逆の方向に働く定義であり、add(ci),del(ci) は制約集合に制約を追加したり削除する指定である。 SF 定義の際、定義の順番や全体のフローについては考慮する必要がないので、フロー定義のし易さやメンテナンス性が向上する。

前述のソートプログラムの例のワークフローを **WFL** で記述すると以下の様になる。

```
※ c1=DesignEND,c2=CodingEND,c3=TestOK,c4=TestNG
MakeSortProgram(
 {A/Designer,B/Programer,C/Tester},
  {c1,c2,c3} => Author->DesignStage
  {c2,c3}    => DesignStage->CodingStage
  {c3}       => CodingStage->TestStage
  {}         => TestStage->EXIT
  {c4,c3}    => TestStage->CodingStage,
                    add(c2),del(c4)},
 {D1,D2,...,Dn})
```

4. 制約評価機構のメカニズム

WolF の制約評価の基本サイクルは以下の 4 ステップである。このサイクルを制約集合が空集合になるまでインタプリティブに繰り返す。制約集合の初期値は、ステージ遷移定義の左辺の条件部を重複を除いてマージしたものである。

Step1 ワークフローオブジェクトへのメッセージパッシングによる内部状態の遷移

Step2 現在の内部状態と制約集合を **WolF** が評価し、解消された制約があれば制約集合から取り除く。ここで制約集合が ϕ (空集合) になったら終了

Step3 制約集合とステージフロー定義のステージ遷移条件とのパターンマッチング

Step4 マッチしたステージ遷移の実行

以下ソートプログラムの例で実行の流れを説明する。(図 3 参照)

i) 制約集合の初期状態 = {c1,c2,c3}

ii) DesignerUserObj からのメッセージによって状態が"DesignEND" の状態になる。

iii) 制約集合から c1 を削除する。制約集合 = {c2,c3}

iv) 制約集合とステージフロー定義の左辺の比較

v) ステージ遷移"DesignStage → CodingStage" が実行されステージが変わる

vi) ProgramerUserObj からのメッセージによって状態が"CodingEND" の状態になる。

vii) 制約集合から c2 を削除する。制約集合 = {c3}

viii) 制約集合とステージフロー定義の左辺の比較

ix) ステージ遷移"CodingStage → TestStage" が実行されステージが変わる

x) TesterUserObj からのメッセージによって状態が"TestNG" の状態になる。制約集合 = {c4,c3}

xi) 制約集合とステージフロー定義の左辺の比較

xii) ステージ遷移"TestStage → CodingStage" が実行されステージが変わる

xiii) 制約集合に c2 を追加し、c4 は削除する。制約集合 = {c2,c3}

xiv) ProgramerUserObj からのメッセージによって状態が"CodingEND" の状態になる。

xv) 制約集合から c2 を削除する。制約集合 = {c3}

xvi) 制約集合とステージフロー定義の左辺の比較

xvii) ステージ遷移"CodingStage → TestStage" が実行されステージが変わる

xviii) TesterUserObj からのメッセージによって状態が"TestEND" の状態になる。

xix) 制約集合から c3 を削除する。制約集合 = ϕ

xx) 終了

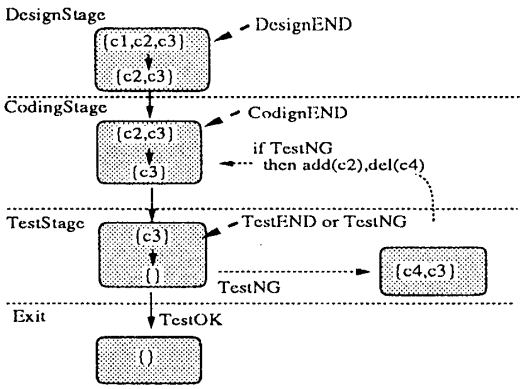

図 3 ワークフローオブジェクトの状態遷移

5. おわりに

制約指向のワークフロー制御言語とその評価機構の基本方式について説明した。課題として、制約条件のパターンマッチングと、インタプリティブな制約評価機構が速度性能面でボトルネックとなると予想される。また表層的な制約しか記述できないなど制約条件の記述力の問題がある。今後実装と評価を行なって洗練していく予定である。

参考文献

[1] 小池, 久永, 小津, 横里 (三菱電機): "協調ワークフロー管理システム 「cooper」 の概要, 情報処理学会第 47 回 (H5 年後期) 全国大会予稿集 5Q-2"