ソフトウェアエンジニアリングデータベース KyotoDB におけるユーザインタフェースと協調活動支援機能の実現

5H−2

山下　薫，鯵坂　恒夫，松本　吉弘
(京都大学工学部)

1．はじめに

KyotoDB は、現在われわれが開発中のオブジェクト指向ソフトウェアエンジニアリング支援環境である［1］。KyotoDB の主な目的は、ソフトウェア開発の各段階で生産されるソフトウェア構成要素と、それらの意味的な関係を統合的に管理してソフトウェア生産の効率化と品質の向上を図ることである。

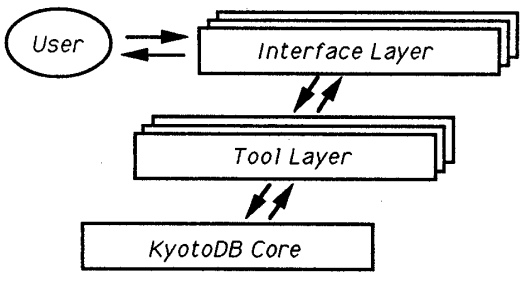
本稿では最初に、KyotoDB におけるユーザインタフェースの実現と、そのためのデータの可視化の方法について述べる。次に、ソフトウェア構成要素間の「関係」を用いて、グループによるソフトウェア開発を支援する機能（コラボレーション支援）の考え方と、その分散環境における試作について説明する。

2．KyotoDB の構成とユーザインタフェースの位置付け

KyotoDB は3つのレイヤから構成されており、それぞれ Interface Layer、Tool Layer、KyotoDB 核と呼んでいる。これを［図1］に示す。

KyotoDB 核はオブジェクト指向データベースであり、ソフトウェア構成要素とその可視化イメージ、それら相互の関係および管理データを、種々のオブジェクトの階層構造として管理している。

Tool Layer は、KyotoDB 核によって提供されるオブジェクトのメソッドを呼び出す。後述するコラボレーション機能は、主として Tool Layer で実現される。

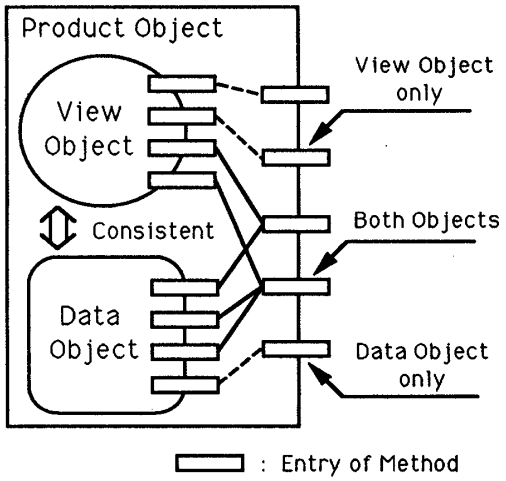


［図1］KyotoDB の3つのレイヤ

Interface Layer はユーザインタフェースのみを担当し、このレイヤだけを変更することができる。また、同じ Interface Layer を異なる Tool Layer と組み合わせることも可能であり、これによって、複数のユーザインタフェースの選択や、異なる機能を統一されたインタフェースで利用することが可能になる。

3．データの可視化

KyotoDB 核では、ソフトウェア構成要素の基本単位は Product Object と呼ばれるオブジェクトに対応し、木・表・グラフなど種々のデータ構造の形で表現される。Product Objectはさらに、構成要素そのものである Data Objectと、対応する可視化イメージである View Objectから構成される。明らかに、この方法では Data と View の一貫性が問題になる。そのために、双方の Object に意味的に等価なメソッドを用意し、Product Objectが双方の一貫性が常に保たれるようにメソッド呼び出しを行なう。従って Data/View Object は Product Object に包含され、外部からは隠蔽される。この考え方に基づくオブジェクトの階層構造を［図2］に示す。



［図2］オブジェクトの階層構造

Prototyping of User-Interface and Collaboration supporting facilities on Software Engineering DataBase KyotoDB
Kaoru YAMASHITA, Tsuneo AJISAKA, Yoshihiro MATSUMOTO
Faculty of Engineering, Kyoto University

これにより、双方のオブジェクトがそれぞれ適切な内部構造を選択でき、データの可視化に関するメソッドがカプセル化できる。さらに、一つのData Object に対して複数の View Objectを対応させることにより、同一のソフトウェア構成要素に対して複数の視点を提供できる。

4．コラボレーションによる一貫性保持

KyotoDB 核では、ソフトウェア構成要素間の関係を、２つの Data Objectの間の１対１の関係に限定し、Relation Object という形で扱う。Relation Object の１つの項目が１つの関係に対応し、関係付けられた Data Object の要素（ノード）、関係の内容、関係を定義した理由などを記録している。 現時点の KyotoDB では、関係の定義は全てユーザに任され、関係の内容についても、システムがそれを単独で判定する機構を備えていない。

すなわち、ある関係が新たに定義されたり、変更や削除を受ける場合、そのような関係に対する操作が意味的に正しいものであるかどうかの判断は、全てユーザに委ねられることになる。

KyotoDB 核は、 Relation Object と Data Object を検索することにより、特定の Data Object のノードが他のどのノードと関係付けられているかを調べる機能を持つ。これによって与えられる依存関係の表を Dependency List と呼ぶ。Tool Layerは、関係に関する操作に先立ち、Dependency List によって依存関係の有無をチェックする。依存関係が全くなければ操作を受け入れる。

そうでない場合は、Dependency List に列挙されたユーザの間の話し合いを支援し、一貫性を保持する。このような話し合いによる問題解決を、KyotoDB における「コラボレーション」と呼び、コラボレーションを開始するユーザを「ホスト」と呼ぶ。[図３] にコラボレーションのシナリオの一例を示す。

a. 問題となる操作の検出と、それに対する Dependency List の作成
b. ホストによる、関連するユーザ（ゲスト）の選択
c. ゲストの呼び出しと、参加の有無の問い合せ
d. 話し合い（セッション）
e. ホストによる結果報告
f. e. に従ったデータ操作の実行

[図３] コラボレーションのシナリオの一例

5．インプリメンテーション

以上の設計に基づきプロトタイプを作成した。Interface Layer および View Object は、Ｘウィンドウ上のユーザインタフェース構築ツール 鼎 [２] を用いて実現した。

また、グループによるソフトウェア開発では、１人に１台ずつのワークステーションが割り当てられることが望ましい。KyotoDB はこのような環境をターゲットとし、ユーザ毎に Tool Layer と Interface Layer を用意する。協調活動のためには、それぞれの Tool Layer が、KyotoDB 核や他の Tool Layer と通信を行なう必要がある。そのためにプロセス間通信の種々の方法を用いている。

6．おわりに

現在、KyotoDB を実用的なものにするため、われわれは以下のようなアプローチを試みている。

A. KyotoDB の Data Object のデータ構造の拡張と、対応する View Object の拡張
B. A.を用いて対話できるコラボレーション支援システムの実現
C. 関係の表現の拡張
D. 協調活動のモデル化

特に D.が重要な課題である。[図３] で示したようなシナリオ以外の形態の協調活動の支援や、コラボレーション参加要求に対して返答がない場合の対処法、実時間ではない（非同期）コラボレーションの支援の実現のためには、モデル化が欠かせない。[３]

参考文献

[１] Yoshihiro Matsumoto,Tsuneo Ajisaka,"A Data Model Applied in Software Project Database KyotoDB", Advances in Software Science and Technology Vol.2,Japan Society for Software Science and Technology,1990 (to appear)

[２] 暦本,菅井,他,「Ｘウィンドウ上のマルチメディアユーザインタフェース構築環境：鼎」,第30回情報処理学会プログラミングシンポジウム予稿集

[３] 石井 裕,「ＣＳＣＷ --- コンピュータを用いた人間の協調活動支援」,日本ソフトウェア科学会 ウインターチュートリアル,1990年1月18日