様式Ｃ－１９

科学研究費助成事業（科学研究費補助金）研究成果報告書

平成２５年　５月１７日現在

機関番号：３４３０４
研究種目：基盤研究(C)
研究期間：2010～2012
課題番号：２２５０００３８
研究課題名（和文）　グルーコードを削減するモジュールのバインド方式に関する研究

研究課題名（英文）　A binding mechanism between modules for reduction of glue codes

研究代表者
荻原　剛志（OGIHARA TAKESHI）
京都産業大学・コンピュータ理工学部・教授
研究者番号：90231224

研究成果の概要(和文):手続き型のプログラミング言語を用いたソフトウェア開発では､通常、モジュールを相互に連携させるための手続き（グルーコード）を書き加える必要がある。本研究では、オブジェクト指向開発で用いられているデータバインディングの機能を手続き型言語で利用するための仕組みを提案した。この仕組みを用いるとグルーコードを記述せずにモジュールの連携が可能であり、ソフトウェアの部品化を進め、バグの低減を図ることができる。

研究成果の概要（英文）：In software development with procedural programming languages, routines to connect modules each other (glue codes) are needed.　We proposed a new mechanism that could realize the same function as data binding used in object-oriented software developments.　It can connect modules each other without glue codes.　It is also useful for making module more independent, and for preventing bugs.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
２０１０年度	1,100,000	330,000	1,430,000
２０１１年度	1,000,000	300,000	1,300,000
２０１２年度	1,100,000	330,000	1,430,000
年度			
年度			
総　計	3,200,000	960,000	4,160,000

研究分野：総合領域
科研費の分科・細目：情報学・ソフトウエア
キーワード：ソフトウエア工学、組込みソフトウエア

１．研究開始当初の背景

(1) ソフトウェアモジュールを相互に組み合わせるためには、通常、グルーコードを追加する必要がある。グルーコード自体の記述にコストがかかる上、グルーコードによって新たなバグが混入する可能性もある。

(2) オブジェクト指向開発ではデータバインディングと呼ばれる機能を用い、グルーコードを記述せずに、複数のオブジェクトを連携させることができる。しかし、C 言語のような手続き型（非オブジェクト指向）言語にはこの機能が備わっていない。

(3) 組込みソフトウェアの開発では C 言語が利用されることが多い。その場合、オブジェクト指向設計ではなく、構造化モデリングによって設計、開発が行われる。

２．研究の目的

(1) 手続き型の言語を用いたソフトウェア開発において、グルーコードの記述を抑えつつ、モジュール間の連携を実現する手法を提案する。

(2) 提案した手法が、構造化モデリングによる設計、開発に適合できることを示す。

３．研究の方法

(1) C 言語において、オブジェクト指向におけるプロパティと同様にデータを格納し、共有できる機構を提案する。

(2) 上記の機構を実装し、パソコン上、および組込み環境での動作を確認する。

(3) 提案した機構をソフトウェアの設計段階から利用する開発手法を提案する。

４．研究成果

(1) 静的なバインド機構の実装と評価

本研究課題に先行して、手続き的言語における静的なバインド機構に関する提案を行っていた。本研究ではまず、この機構の実装、およびソフトウェア開発における利用を試みた（雑誌論文②に記載）。

この機構では、モジュール間の結合は実行時ではなく、リンク時に行う。コンパイル言語では、モジュール内で定義されていない変数は外部名として未定後のままオブジェクトコードに格納されている。モジュール間で共有させたい変数があった場合、各モジュールの未定義シンボルの定義を操作することによって、異なるモジュールで宣言した外部変数が、実行時に同じ変数の実体を参照するようにできる。このような変数間の関連に関する情報をソースプログラムとは別に記述しておき、リンク時の未定義シンボルの操作を自動的に行う仕組みを用意することによって、グルーコードを記述せずにモジュール間の関係を変更できることを確認した。

しかし、この仕組みによる指定は、実行形式のプログラムが作成される時にしか有効ではなく、静的なコードの間の関係を定めることしかできない。このため、プログラムの実行中にソフトウェアの構造を変化させたり、動的に構成されるデータ構造の間の連携を制御したりすることができない。実際にソフトウェアの作成を試みたが、比較的単純な構造の実現が可能になる程度であり、実用的なソフトウェアへの適用は困難であることが判明した。

(2) 動的なバインド機構の提案

静的なバインド機構は、実用面で問題があることが判明したため、オブジェクト指向言語におけるデータバインディングの考え方を取り入れた動的なバインド機構の提案を行った（学会発表②に記載）。

データバインディングは、値の間に依存関係のある複数のプロパティに注目し、オブジェクトを結合させるものである。一方が値を更新すると、他方はそれに応じたアクションを自動的に行う。プロパティ間のバインディングの指示は、そのアプリケーションの部品の組み合わせ方の記述であり、クラスやモジュールの機能からは独立している。そのため、XML 形式のデータなどとしてソースコードの外に記述することも多い。

データバインディング自体はオブジェクト指向の概念と直接関係しているわけではないため、C 言語をはじめとする手続き型言語にも適用可能であると考えられるが、C 言語に適用可能なバインド機構の提案はこれまでにない。

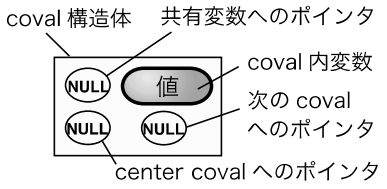
手続き型言語でもモジュール、または動的にメモリ領域を獲得した構造体に属性値を持たせ、その値によってプログラムの動作を変えたり、処理結果を値に反映させたりすることができる。このような属性値は、オブジェクト指向言語におけるプロパティと同じ役割を持つものと見なせる。この属性値が変更される時に、あらかじめ設定しておいたコールバック関数を自動的に呼び出す仕組みを作っておき、さらにこれらを相互に連携させることによって、手続き型言語を使ったシステムにおいてもデータバインディングと同じ効果を得ることが可能である。

このような考えに基づいてデータバインディングを実現するライブラリの設計を行った。この機構を coval と呼ぶ。この名称は化学の共有結合（covalent bond）から借りている。

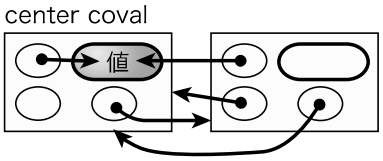
(3) バインド機構 coval の実装

実装は組込みシステムでの動作を前提として C 言語で行うこととし、構造体とマクロを組み合わせて任意のデータ型を共有できる機構とした。coval は、モジュール間で値を共有する仕組みを備えた一種の変数であり、同じ型のデータを保持する他の coval と値を共有するようにできる。この操作をバインドと呼ぶ。バインドされたすべての coval は共通の変数を参照しているため、いずれかの coval で値を更新すると他のすべての coval の値も変化したように見える。coval にはコ

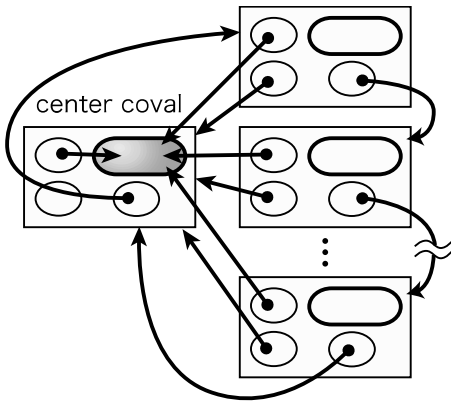
ールバック関数を対応させておくことができ、バインド中に共有変数に対して更新の操作が行われると自動的に呼び出される。



(a) 値を共有していない場合



(b) ２つの coval で値を共有する場合



(c) 多数の coval で値を共有する場合

図１　coval の構造の概念

図１に、coval のデータ構造と、バインドされた状態の概念を示す。
データ型 T の値を共有するための coval 構造体の型は、マクロ CovalTypeOf(T)によって宣言できる。また、２つの coval 構造体へのポインタ a と b があったとき、これらの間のバインド操作はマクロ CovalBind(a, b)で指定できる。同様に、値の参照と更新、コールバック関数の設定などの操作を行うためのマクロも定義されている。
coval は仕様、ソースプログラムともすべて公開しており、後述の Web ページからダウンロード可能である。

(4) バインド機構 coval の評価
coval は C 言語のライブラリとして実装され、必要なメモリ量、オーバーヘッドはわずかである。32 ビットモデルの Intel チップ向けのコードを gcc コンパイラで生成した場合、int 型を保持する coval 構造体は 44 バイトで済む。コンパイル後のコード（テキスト）は約 2.1 キロバイトである。標準関数を含め、他のどんなライブラリやシステムコールにも依存していないため、OS の存在しない組込み環境であっても問題なく動作する。

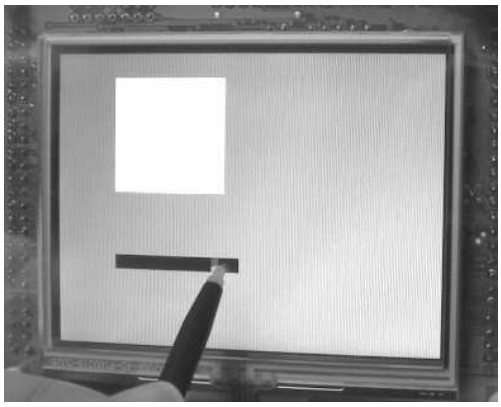


図２　組込みボード上での動作例

図２に、組込みシステムの実験用マイコンボード上で coval を使ったプログラムを動作させた写真を示す。この環境は ROM が 512kB、RAM が 40kB で OS はない。タッチパネル付き液晶からのイベント処理に coval を利用することで、ソフトウェア部品の追加、変更が容易に行えることを確認した。

(5) バインド機構を用いた分析と設計
本研究では、手続き型言語で開発を行う場合に広く利用されている構造化モデリング手法に基づき、coval によるデータバインディングの利用を考慮に入れて設計を進める方法を提案した（雑誌論文①に記載）。ここで用いたモデリング手法は、データフロー図(DFD)に制御フローの概念を導入し、リアルタイムシステムの制約にも対応できるように拡張したものである。
例題として、タンクの水位を管理する制御システムのシミュレータを考える。このタンクには常に水が蓄えられ、必要に応じて取水口から水が取り出される。水位がタンクに設定された最低値を下回ると給水栓が開き、タンクへの給水が行われる。水位が設定された最高値に達すると給水は停止する。このシミュレータでは、取水栓を開く動作を、利用者がボタンをタッチすることで表す。シミュレータは一定の時間間隔でタンクの水位と取水栓、給水栓の状態を調べ、次の動作と状態を決めるものとする。この考え方に沿って記述した DFD を図３に示す。実線の矢印はデータフロー、破線の矢印は制御フローを表す。
図３の DFD では、プロセス１とプロセス３が取水栓の状態を共有し、プロセス４とプロセス５が給水栓の状態を共有している。このよ

うな、プロセス間で値を共有している部分を coval で実装することにして DFD を書き直したものが図４である。ここでは平行な二重線で coval を表している。coval からプロセスに向かう破線の矢印は、値の更新によってプロセスが活性化されることを示す。

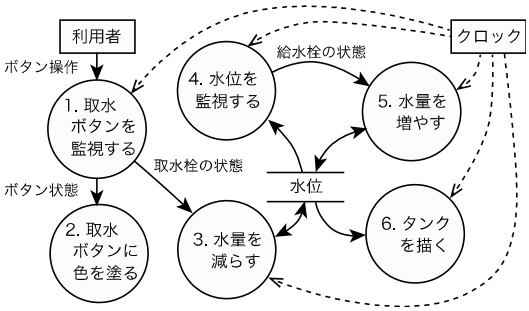


図３　通常のデータフロー図

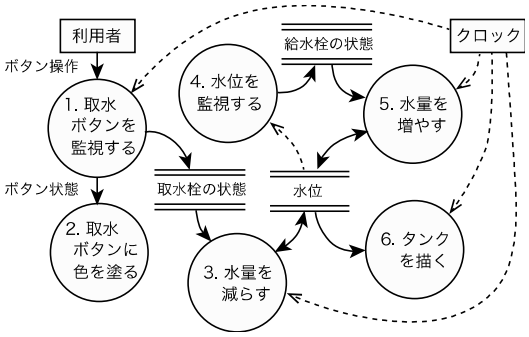


図４　coval を利用したデータフロー図

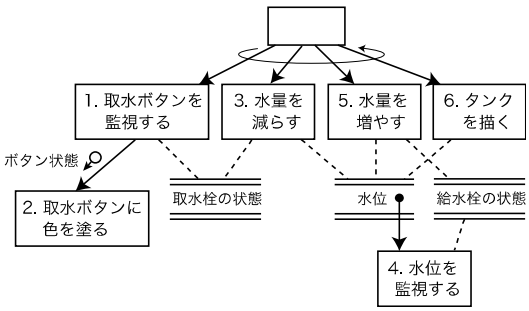


図５　coval を利用した構造図

さらに、図４をもとに作成した構造図を図５に示す。ここでも coval は平行な二重線で示している。coval とモジュールの関連は破線で示され、手続き呼び出しではなく、バインディングで関連づけられることを表現している。ただし、コールバック関数を呼び出すことが決まっている場合には、coval からモジュールへ向けて矢印を書く。モジュール 4 は親モジュールから直接呼び出されるのではなく、coval の値が更新されることで起動される。この構造図を参照することによって、coval によるバインディングが利用可能なモジュールを意識してコーディングを行うことができる。

このように、coval を設計段階から利用してソフトウェア開発を行うことが可能であることを示すことができた。組込みシステムをはじめ、手続き型言語を用いるソフトウェア開発において、部品化を促進し、バグの混入を防ぐ技術として広く利用できるものと考えられる。

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計２件）

① 荻原剛志、手続き型言語におけるデータバインディング機構の提案と構造化設計への適用、情報処理学会論文誌、査読有、54 巻、4 号、2013、pp.1573-1580、http://id.nii.ac.jp/1001/00091591/

② 荻原剛志、C 言語への静的なバインド機構の実装、京都産業大学論集 自然科学系列、査読有、第 40 号、2011、pp.109-123

〔学会発表〕（計２件）

① 荻原剛志、共用変数によるバインド機構を用いた組込みシステムの開発手法について、電子情報通信学会技術研究報告、Vol.111, no.481, SS2011-58, 2012、pp.7-12

② 荻原剛志、共有変数を用いたバインド方式の提案とソフトウェア開発への応用について、情報処理学会ソフトウェア工学研究会報告、SE-171(23)、2011、pp.1-8

〔その他〕

ホームページ等

http://www.cc.kyoto-su.ac.jp/~ogihara/coval/

６．研究組織

(1)研究代表者

荻原　剛志（OGIHARA TAKESHI）

京都産業大学・コンピュータ理工学部・教授

研究者番号：90231224