メカトロ教育のための分散処理システム

○藤田 和友（弓削商船），百垣 愛弓（弓削商船），前田 弘文（弓削商船）

Distributed processing system for mechatronics education

○Kazutomo FUJITA (YNCMT), and Ayumi MOMOGAKI (YNCMT),
and Hirofumi MAEDA (YNCMT)

Abstract: This paper describes about decentralized processing system "Marionette" for robots to facilitate device management. This system is made using the shared memory. As a result, the robot can be easily developed. By diverting this system to mechatronics education, the result more than the former is expectable. Moreover, automatic generation of this system is also described.

１．緒言

近年の飛躍的な半導体技術の発展により，小型で高性能なコンピュータが普及している．また，多くの産業機械でこれらのコンピュータを制御部分として組み込んでいる(組み込みシステム)．そのため，システムが複雑化し，プログラムや電子回路，機械要素といったメカトロ分野に強い人材が必要とされている．

高等専門学校においても，社会のニーズに答えるため，多くのメカトロ教育が実施されている．本校においても，創造性実験という名目で，組み込みマイコンを用いたシステム開発の実験実習が行われている．しかし，実際に行われている実験実習は，発注，設計など行える点では有効であるが，組み上がったシステムには簡易的なシステムが多く，簡単なセンサの読み込みやモータの動作といったほぼシーケンスに近い状態のシステムが多くを占める．本来これらの実験実習の目的は，試行錯誤を通した工学的センスを身に着けることであり，フィードバック制御などもう一歩進んだシステムを構築しなければ，身に着けることは難しい．

そこで我々はこれまでに開発してきたレスキューロボット(以下，UMRS-2010)のシステムに着目した[1]．レスキューロボットに用いられているシステムは，共有メモリを用いた分散処理システム(以下，Marionette)であり，メンテナンス性を意識したものとなっている．そのため，各機能がモジュール化した独立プログラムとなっており，それぞれの機能はいたってシンプルなものとなっている．またこれまでに，分散処理システムとレスキューロボットシステムが混合した状態で存在していたソースコードを一般化するとともに，他のロボットシステムに転用できる状態まで開発を進めてきた．このことから，メカトロ教育に対しても試行錯誤とは関係がない通信部分などはあらかじめ作成しておき，制御部分だけ開発するといった用途などに使用できると考えられる．しかし，このままではMarionetteのソースコードを全て手作業で打ち込む必要があり，膨大な時間がかかってしまう．

そこで本研究では，メカトロ教育にMarionetteを流用するため，Marionette 自動生成プログラムの作成を行った．本論文では，メカトロ教育の現状とMarionetteの実用例に少し触れ，Marionette の概要，一般化，自動生成についても述べる．

２．メカトロ教育

2.1 メカトロ製品の開発サイクル

メカトロ製品の開発は，目的とする機械の検討から始まり，制御システム・制御対象の設計および構築を経て製品として完成する．それらを分析した結果を踏まえて，検討・改善を行うことでシステム全体の完成度を高めていく(**Figure 1**)．

この開発サイクルは，マネジメントにおける PDCA サイクルに類似しており，設計＝計画(Plan)，構築＝実行(Do)，分析＝評価(Check)，検討＝改善(Action)に対応させることができる．

2.2 メカトロ教育の現状

メカトロ教育において，工学的センスを身に着けるためには，分析・検討に重点を置き，トライアンドエ

ラーを繰り返しながら, PDCA サイクルをより多く回す(スパイラルアップ)ことが重要である.

しかし，メカトロ教育の現状は設計，構築の部分に多くの時間が費やされており，思ったほどの効果が得られていない.

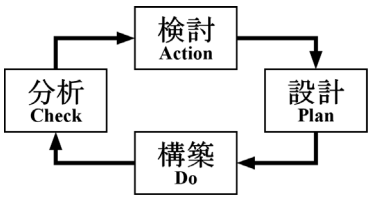

Fig. 1 The development cycle of a system

３．レスキューロボットの概要

Marionette を用いたシステムの実用例として，非営利活動法人国際レスキューシステム研究機構にて開発された UMRS-2010 を **Figure 2** に示す.

UMRS-2010 は，地下街で発生した災害に対応することを目的とした探査ロボット UMRS-2009 の後継機(**Figure 3**)にあたる [2)3)]. しかし，UMRS-2010 は新たなテーマとして防爆機能を搭載することを目的としているため，中身は火花が発生しないブラシレスモータを使用するなど，外見は酷似しているもののまったく別のロボットシステムとなっている.

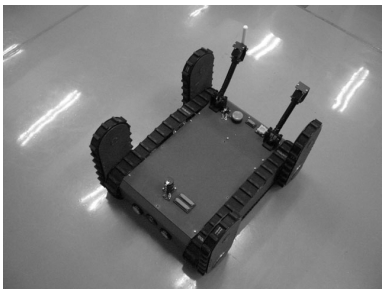

Fig. 2 UMRS-2010

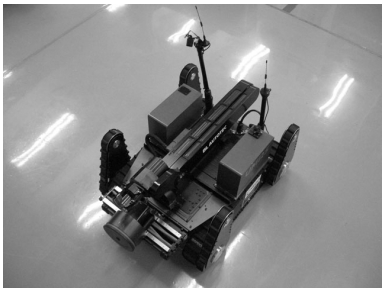

Fig. 3 UMRS-2009

Figure 4 に UMRS-2010 の内部構造を示す．Fig. 4 において，全てのブラシレスモータ(マッスル株式会社 COOL MUSCLE 2 CM2-C-56B20A-K ，CM2-C-60A10A-R)は，デイジーチェーンによって結合され，シリアル(D-Sub9pin)によって制御用 PC(ICOP I.T.G. 株式会社 VDX-6314-512)に接続される．また，4 つのカメラ(MAXWIN 小型カラーバックカメラ CAM11A)はビデオサーバ(ACTi ACD-2000QT)を介して，LAN により接続される．同様に，光ファイバージャイロ(日立電線株式会社 HOFG-OLC-1)についても, 組み込み用超小型デバイスサーバ(Lantronix 株式会社 XPort)を用いてシリアルを LAN に変換した後接続される．さらに，加速度センサ(クロスボー株式会社 CXL04GP3)および LED(三菱電機オスラム株式会社 DP03A-W4-754)については，それぞれ CR フィルタと LED ドライバを介した後, COOL MUSCLE 2(以下，CM2)経由で VDX-6314-512 に接続される．なお，操作卓には NEC Corporation ShieldPRO FC-N22A/BX6SS1B を，無線通信には株式会社バッファローWLI-UC-GNHP を使用している.

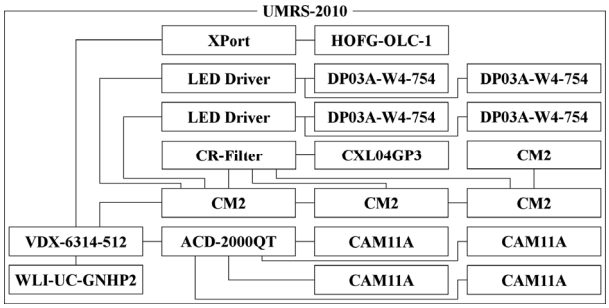

Fig. 4 Internal structure of UMRS-2010

3.1 一体型サーボシステム

CM2 は，一体型サーボシステムであり，モータ・ドライバ・エンコーダ・コントローラ・電源ユニット・PLC 機能の全てを内蔵している(**Figure 5**)．これにより，モビリティロボットに必要とされるモータ制御を全て CM2 に委ねることができる．結果，制御用 PC である VDX-6314-512 はシリアルを介して CM2 に目標指令を送るだけとなる．また，A/D 変換や I/O といったロボットに必要とされる機能も搭載されているため，ハードウェアに依存する(特殊デバイスへのアクセス)部分については，全てを CM2 に委ねることができる．これに

より，ロボット製作においてハードウェアとのやり取りをモジュールという形で外部に設けることで，ロボット内部におけるシステムを簡略化することができる．

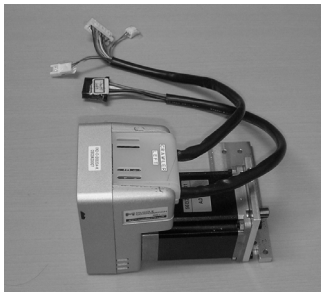

Fig. 5 COOL MUSCLE 2

3.2 制御用 PC

制御用 PC である VDX-6314-512 は, CPU として DM&P Vortex86DX 800 Mhz SoC を搭載しており，メモリが 512 MB，IDE/ATA フラッシュストレージモジュールとして記憶容量が 8 GB と組み込み用ボードとしては十分なスペックを要している(Figure 6)．そのため，本ロボットでは Linux(Ubuntu 10.04 LTS)をインストールして使用する．また，基板のサイズも 100×66 mm とコンパクトである．さらに，I/O 関係も充実しているが先に述べたように，CM2 に外部制御を任せているため，USB およびシリアル以外は使用しない．

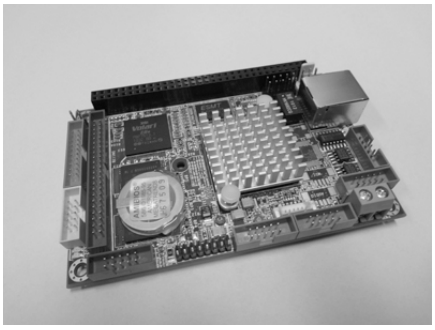

Fig. 6 VDX-6314-512

3.3 ソフトウェア

本ロボットシステムのプログラム構成の概略を **Figure 7** に示す．ソフトウェアの開発には，操作卓，ロボット本体ともに Ubuntu 10.04 LTS 上で動作するものとし，開発言語には C 言語を用いる．なお，Marionette は操作卓側のプログラムに使用されている．

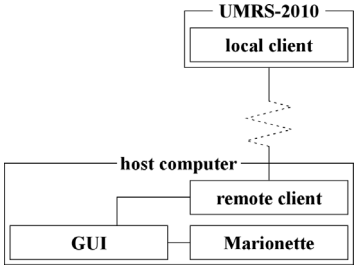

Fig. 7 View of a robot system program

４．分散処理システムの概要

分散処理システム Marionette は, 糸を使った操り人形を語源としている．操り人形の糸は，各部位を繋ぐ横糸と人形全体を操るための縦糸に分かれる(**Figure 8**)．また，縦糸は人形を操作するために一本もしくは複数の棒によって接続されている．そのため，棒を操作することで人形全体を操ることが可能となる．

本 Marionette も操り人形を模擬しており, 各プロセスと Marionette は縦糸で, 各プロセス同士は横糸で接続された形をとる(**Figure 9**)．これにより，一つ一つのプロセスは，人形の腕や足のように，それぞれの担当する小規模な機能を実装するだけで済み，プログラムをシンプルに構成することが可能となる．

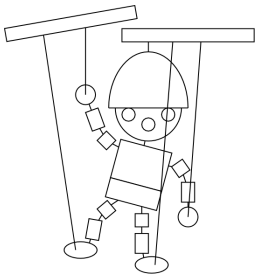

Fig. 8 Marionette of a doll

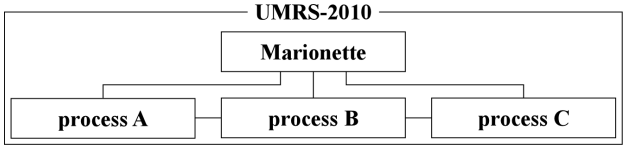

Fig. 9 View of Marionette

実際にプログラムを実装する際は，縦糸と横糸は共有メモリとなり, プロセス同士間, プロセス・Marionette 間はそれぞれ 1 対 1 の関係となっている．そのため，**Figure 10** で示すような 1 対多数や多数対多数という接続は発生しない．もし複数の接続を行いたい場合は，

Figure 11 に示すような接続方法を用いる．

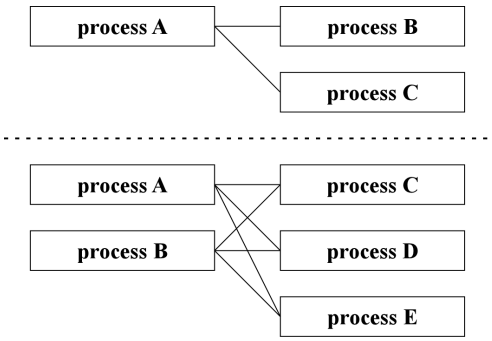

Fig. 10 Prohibited matter of Marionette

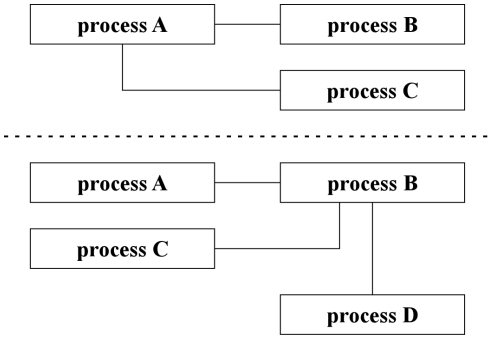

Fig. 11 Example of connection

以下に，Marionette における制約を示す．

・プロセスは 1 対 1 の関係を維持する．
・共有メモリはセマフォを使った排他制御によって管理する．
・プロセスの起動および終了は同期する．

これらの制約を実装するために，Marionette 本体では，それぞれのプロセスを同時に起動するとともに，終了の際も同期を取って同時に終了を行う．この管理を行うものが，先に示した縦糸に該当する共有メモリである．また，プロセス間のデータ通信に用いる共有メモリについても Marionette 本体が管理する．そのため，Marionette 以外が共有メモリを管理できないよう，Marionette は各セマフォと共有メモリのキーをそれぞれ 2 個ずつ，プロセスの起動時に必要なプロセスのみに受け渡す形を取る（**Figure 12**）．これによってシステム全体におけるバグの発生を抑制している．

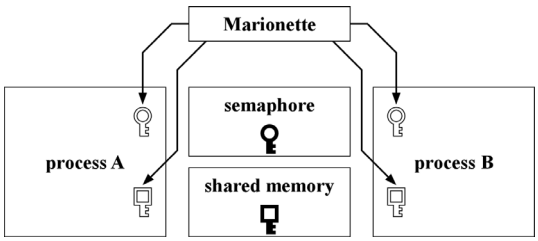

Fig. 12 Delivery of the key in Marionette

５．分散処理システムの一般化

本論文においては，Marionette を含む 4 つのプロセス間通信を例として述べる（**Figure 13**）．

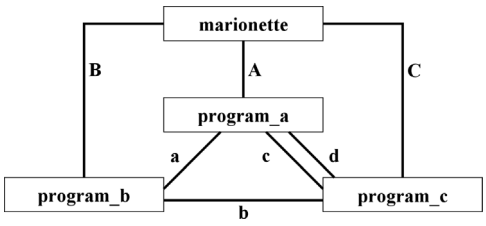

Fig. 13 Process configuration

また，それぞれのプロセス間通信の共有メモリは，**Figure 14** に示す構造とした．なお，それぞれの共有メモリにはセマフォを設け，型宣言も行っている．

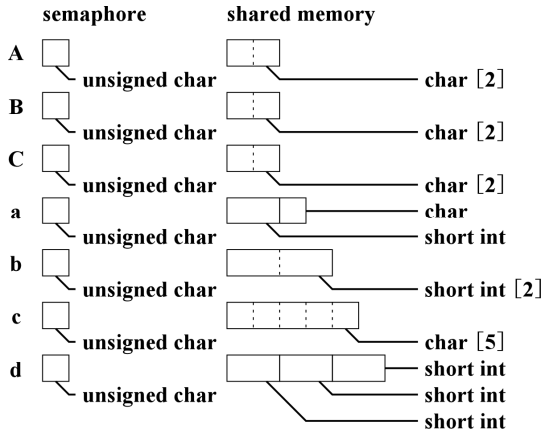

Fig. 14 Semaphore and shared memory

今回，ソースコードの一般化にともない以下の 4 つの作業を行う必要があった．これは，Marionette の汎用性を高めることはもちろんのこと，ユーザが Marionette を熟知しなくても容易に扱える必要があったからである．

① Marionette 部分のソースコード抽出
② セマフォの配列部分の切り離し
③ ソースコードの関数化
④ マクロによるソースコードの隠蔽化

5.1 Marionette 部分のソースコード抽出

最初の処理として, UMRS-2010 のソースコードから, Marionette 部分のソースコードを抽出する作業を行った．その際，UMRS-2010 のソースコードと一体化している部分については新たにソースコードの書き換えを行った．

5.2 セマフォの配列部分の切り離し

以前までは，セマフォが配列で定義されていたために，**Figure 15** のような形を形成していたことになる．このため，セマフォのキーを全てのプロセスが持つこととなり，セマフォの配列番号を間違えた場合，違う共有メモリをロックするという問題が生じる（**Figure 16**）．

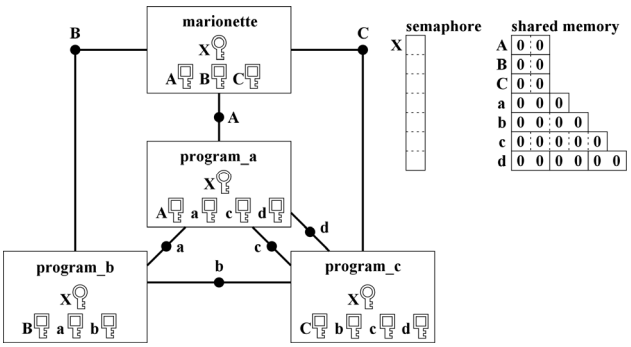

Fig. 15 Former semaphore structure

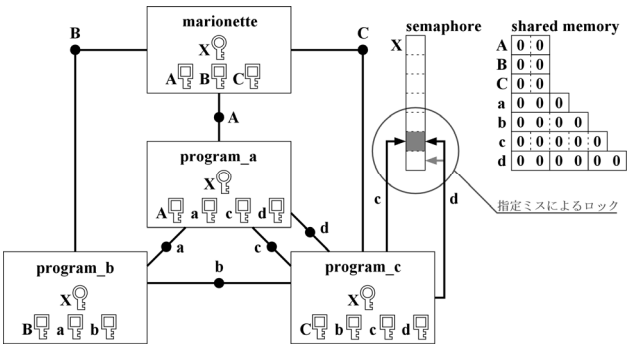

Fig. 16 Specification mistake

以前であれば，Marionette を熟知したものがソースコードを記述していたため，このような問題は発生しなかったが，一般化するにあたっては配慮する必要がある．そこで，Fig. 14 のように，それぞれのセマフォを切り離し，キーもそれぞれが独立する形を採用した．

5.3 ソースコードの関数化

Marionette の関数部分をセマフォ・共有メモリを中心に見直し，改編を行った．以下が，Marionette で用いられている関数である．

・メッセージ表示関数
・シグナル初期化関数［SIGINT］
・シグナル処理関数［SIGINT］
・セマフォバッファ初期化関数
・セマフォロック関数
・セマフォアンロック関数
・共有メモリ読み込み関数
・共有メモリ書き込み関数
・共有メモリ表示関数

5.4 マクロによるソースコードの隠蔽化

ユーザが Marionette を意識することなく使用するために，hiding.h を設けマクロを記載することでソースコードの隠蔽を行っている．

1 つには，ユーザが使用する構造体と Marionette が使用する共有メモリを共用体によって結びつけることで，ユーザは自分自身が定義した構造体のみを使用するだけでプログラムが組める形を取っている．そのため，ユーザが共有メモリ側を意識しないように，共用体を意味する部分を置換し，構造体の形に見えるように隠蔽している．

2 つ目には，データの読み書きにおいて，以下のマクロを使用することで，Marionette の内部構造を意識させない形式を取っている．

・READ_［共有メモリ名］: 共有メモリのデータを配列に格納する．
・WRITE_［共有メモリ名］: 配列データを共有メモリに格納する．

さらに，main 関数においても，ユーザに Marionette の内部構造を意識させないため，以下のマクロを使用して隠蔽を行っている．なお，変数宣言部分については，変数の二重定義によるエラーを発生させないためにマクロによる隠蔽は行っていない．

・INITIALIZATION：初期化，キーの取得，シグナルの設定，共有メモリのアタッチを行う．
・SHUT_DOWN：終了要求があるかチェックし，ある場合にプロセスを終了させる処理に移る．
・ENDING：終了フラグを 1 にセットし，共有メモリのデタッチを行う．

６．分散処理システムの自動生成

Figure 17 に Marionette のソースコードを自動生成するプログラム（Marionette World Creator：MWC）のファイル構造を示す．

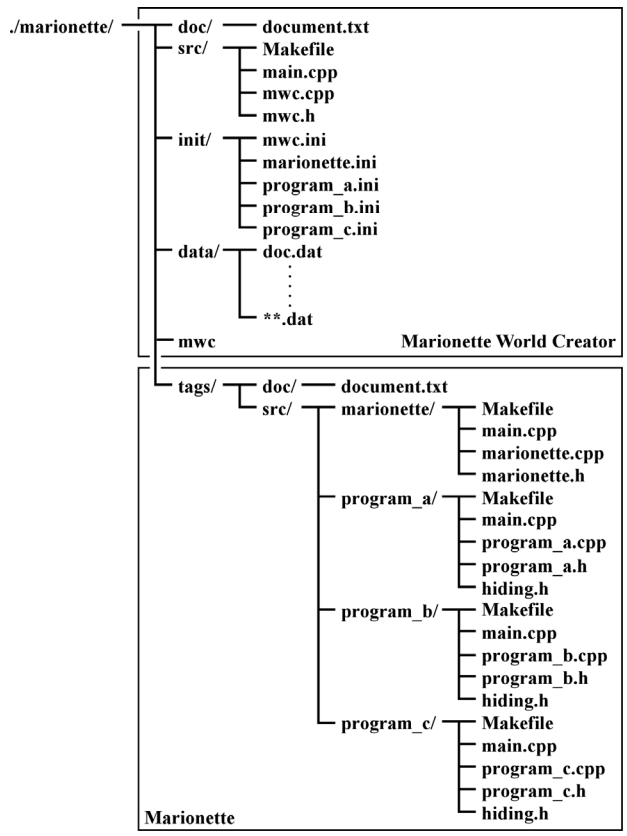

Fig. 17 File composition of MWC

mwc が自動生成プログラムの実行ファイルであり，init ディレクトリに収められた ini ファイルの情報をもとに，data ディレクトリのデータファイルと合成することで，ソースコードを生成する．src ディレクトリ内は mwc のソースファイル，doc ディレクトリ内はドキュメントファイルが格納されている．実際に自動生成されたソースコードは全て tags ディレクトリに格納され，ユーザはそれぞれのディレクトリにある main.cpp を変更することでプログラムの改良を行う（marionette ディレクトリを除く）．

７．結言

今回，レスキューロボットで使用されていた Marionette をメカトロ教育に適用する方法について述べた．また，システムの構築を容易に行うための自動生成プログラムについても触れた．今後は，自動生成プログラムの設定部分に GUI を用いるなど，よりユーザに使いやすいものに改良していく予定である．また，この Marionette をベースとしたメカトロ教育の枠組みを構築していく予定である．

参考文献

1) 前田 弘文，小林 滋，高森 年：レスキューロボットにおけるデバイス管理を容易にするためのシステム開発，弓削商船高等専門学校紀，第 34 号，pp.48～53，（2012）
2) 石井 良典，大坪 義一，小林 滋，小林 泰弘，山本 祥弘，梅田 栄，海藻 敬之，前田 弘文，高森 年，田所 諭：閉鎖空間内探索ロボットのための遠隔操縦システムの開発，第 11 回システムインテグレーション部門講演会（SI2010），pp.1238～1241，（2010）
3) 前田 弘文，藤長 大祐，五百井 清，大坪 義一，小林 滋，高森 年：レスキューロボットにおけるデバイス管理を容易にするための分散処理システムの開発，第 12 回システムインテグレーション部門講演会（SI2011），pp.60～63，（2011）