Ⓟ-29

A GUI Object Model in the PF Linac control system

Abe I., Nakahara K., *Mutoh M., *Shibasaki Y. and **Tanaka M.

National Laboratory for High Energy Physics (KEK)
*LNS, Tohoku university
**Mitsubishi Electric system & service Engineering Co, Ltd.
Oho 1-1, Tsukuba-shi, Ibaraki-ken 305, Japan

Abstract

An Object Oriented Analysis for GUI(Graphical User Interface) in the Accelerator domain has been carried out in the PF Linac. Device Objects and Generic Task Objects are analyzed and presented on the GUI stations of the Linac control console. Many features of Object Model operations are described in this paper.

加速器制御における GUI オブジェクトモデル

1．概要

従来の手続き型からオブジェクト指向型 [1,2] に手法を代えて加速器制御を見直す事で、ソフトウエア開発及び加速器制御に大きなインパクトを与える事が可能になった事を報告する。今回は、GUI オブジェクトモデルを主なる対象領域として分析を開始したが、加速器制御の全体の中で位置づける事が必要であるとの事から、加速器ドメイン全般についても各種オブジェクト・クラスの分析を行った。その概要と基本的オブジェクトの体系確立の重要性について述べる。

2．オブジェクト分析の必要性

オブジェクト分析の必要性及びその背景について概観するに、従来の加速器制御分野においては、専ら、OS、ハードウエア、インターフェース構成、ネットワーク等の観点から議論される事が多く、計算機環境の設定が最初で、ソフト開発は従属的に進められてきた。一方、技術分野の目まぐるしい進歩は、ハード・ソフト共にライフサイクルをとみに短くする方向に働いているため、この進歩に合わせて制御体系を更新、維持していく事は容易ではなく、むやみに予算と労力を浪費する。のみならず、一年もしない内に計算機系が古くさく感じてしまう様な激しい陳腐化現象を克服し技術進歩とコストパフォーマンス向上のメリットをうまく取り入れて行くための提案として、設計時のコンセプトの再考慮と共に幾つかの着目点について述べる。最近、加速器制御市販ツール等（Vista system, EPICS）を採用する事等もその対策の一つの解決策である。しかしながら、加速器本来のオブジェクトは、計算機環境に比しそれ程大きな時代変化はない。従って、出来るだけ標準オブジェクトを抽出し、そのオブジェクトは時代進歩する制御システムやマルチプラットフォーム上に対応出来る様にする事で、制御システムの陳腐化に対応出来る事に着目した。もう一つの狙いとして、最適オブジェクトの確立は、制御システムをよりシンプルに出来る可能性がある。オブジェクト分析のもたらすものとして知られている事は、再利用性、共通性の明示、体系確立、差分プログラミングへの確立等であるが、一方分析に失敗をもたらす原因としては、ドメイン問題領域への理解の不十分さ、領域外問題を多く含むシステム問題である。いずれも加速器分野で重要項目である。手続き型からオブジェクト指向型へのパラダイムシフトの難しさも現実には否定出来ない。実装化、オーバーヘッド等でもまだ問題が残っている事は事実である。これらの問題解決とBファクトリー計画へ向けて、実用化研究がこれまで推進されて来た。

3．加速器オブジェクトクラス

構造化分析手法、オブジェクト分析手法共に、幾つもの手法が提案されているがOMT [2] 法を採用した。分析・設計・実装はそれぞれ独立に考えられる事が前提であるので、まず今回の分析フェーズでは加速器制御ドメインを２つのクラスカテゴリに大別した。１）デバイス・クラスと、２）ジェネリックタスク（ノンデバイス）・クラスである。この大別が制御システムを解りやすくし、オブジェクトの保守性を良くする事になる点に着目した。かつてCERNにてコントロール・プロトコルやオペレーショナル・プロトコルの研究 [3] が進められた事があるが、これらは各種装置の状態遷移図とメッセージの定義/標準化の功績で評価される。我々は装置の状態遷移のみならず、装置をオブジェクトとして定義しそのクラスの標準化を進め、オブジェクト指向が提供している継承・派生等を有効に取り入れる事で開発生産性を上げ、保守性・柔軟性を改善している点が大きく異なっている。また、加速器分野において、これまで進められた事の無いジェネリックタスクの分析に着手した。以下２つのカテゴリについて概要を記す。

4．デバイス・カテゴリとサブ具象クラス

制御の観点からボトムアップ的に、各種のデバイス（装置）をオブジェクト定義し、汎化クラスを模索し標準化を進めた。電子銃、加速管系、ビーム輸送系、ＲＦ系、真空系、トリガー系、各種ビームモニター、CAMAC/VME/PLCインターフェース系、計算機・通信系、安全系等装置オブジェクトとして存在しており、これらはクラス分類・汎化が可能である。デバイス・カテゴリでは、分類単位は機能的に分割出来る最適単位をスーパークラスとし、殆ど具象サブクラスを持つ。その為、継承機能等を有効に利用でき、ソフト体系を簡単明確にできる。スーパークラスは実際にデザイン上又は運転保守上便利な分類と言える。GUIレイヤーにおいて、総てのデバイスクラスに関しインスタンスは派生により容易に画面上で追加削除変更が可能となる。つまり加速器装置の変更はツール上での簡単な変更のみで、プログラムへの変更は殆ど不要に成るなど、従来の制御システムとは大きく異なる点で特徴的なものになった。デバイスカテゴリで重要な事は、コンポーネントクラスの汎化であり、これによってGUIレイヤーにおいても制御卓用画面は極めて作りやすくなった。

5．ジェネリックタスク（ノンデバイス）・カテゴリ

デバイスそのものは直接含まないが、デバイスカテゴリーのオブジェクトにメッセージを送ること、若干の継承を受ける事によって、加速器運転、診断、保守を行うオブジェクトの集約をこのカテゴリの対象とした。運転支援クラスの抽出を特徴とし、他にデバイス単独運転、デバイスクラス運転、ビーム調整（エネルギー、カーレント、プロファイル調整）クラス、運転診断クラス、運転記録クラス、アラームクラス、ヘルプ（ドキュメント）クラス等に分類される。ジェネリックタスク・クラスでは具象クラスを持たないものが多いのがデバイスクラスと大きく異なる。また、エキスパートシステムで言うヒューリステック知識を取り込むべきクラスとして位置づけている。以下に述べる２つのクラスはジェネリックタスクカテゴリから外した。

5-A）シミュレーションクラス

深層知識を含む計算モデルで、ビーム軌道計算、回路シミュレータなどは計算用コンピュータにパラメータを渡して計算結果をもらうシステムとして存在する。一般にインスタンスを持たない抽象クラスに位置づけられる。計算モデルをオブジェクト指向で行うかどうかは、今後担当者が決める問題である。又、表層知識は運転クラスに入れた。

5-B）データベースクラス

それぞれのオブジェクトは抽象化され、適当な単位で独立性が高く機能するまとまりとして存在するが、データベース（ＤＢ）の中で管理されれば便利である。しかしながら、データベースは歴史的に見て、第一世代ＤＢ（1960年代）のファイルシステムをベースにしたものから、第二世代の階層型ＤＢであるIMS、又第三世代のネットワークＤＢとしてCODASYL等があり、第四世代（1970年代）としてRDB、1980年代のクライアントサーバー型ＤＢとしてSybase, Oracle, Informix等があるが、現在では第五世代ＤＢ（1990年代）に分類されるOORDB或いは連邦型ＤＢとしてO2, UniSQLなどの市販データベースに至っている。マルチメディア・データや加速器オブジェクトは、OORDBで管理する事が不可欠になるであろう事から第五世代ＤＢにてデータベースドリブン型制御を試みて来た。速度の点やブジェクトの管理の点で不満が残っており、まだ課題が多い。

6．表示系（GUI）具象クラス群

デバイスカテゴリ及びジェネリックタスクカテゴリとは別に扱ったが、表示系クラスは運転時にオペレータから見えることで密接である。また、加速器運転・実験に応じ開発運用に於いて柔軟性の高いシステムである事が不可欠に成ってきており、加速器の操作性をも決める重要項目である。多くのインスタンスはクラスから派生しメッセージ交換又は継承で運転され、必要なプロパティを変更・記入するだけで、クラス継承したインスタンスを画面に簡単に張り付け、必要によってはメソッドを言語記述で差分プログラミングするだけでよい様に、オブジェクト指向の機能を多少なりとも活用出来る言語またはツールによって実装する事が必要であった。具体的には Visual Works、Visual C++, Visual Basic 等で、windows NT 等のマルチプラットフォームOSの上で運転可能な開発環境を目指して来た。既にWindows 上では Visual C++ を使い、継承を許すcontrol packを作成し、Visual Basicの中でユーザーはプログラムが不要なシステムを構築した。加速器全般のオブジェクト分析をしたことでGUIクラスの定義は勿論、実装化は比較的容易でスムーズであった。実装化されたGUIオブジェクトは部分的に運用を開始した。

7-1．問題点（データ・手続きの抽象化）

手続き型従来手法の様に、デバイスやノンデバイス・クラスに於いても機能又はサブ機能としての抽象化が可能なものは、これまでは構造化チャートで表現されて来た。すなわち機能分割法であって、従来の加速器制御は全てこれに基づいていると言っても良い。しかし、手続きとデータをカプセル化したオブジェクトによる抽象化で機能化分析は行われてこなかった。データ抽象の概念に基づいて分析を行う際、特に加速器分野でデバイスオブジェクトはその属性を定義する事は容易であったし、継承機能と共にオブジェクト指向は加速器ドメインに適した手法である事を確認した。特にGUIオブジェクトモデルの実装でその実証ができた。機能分割による抽象化はデータフローとバブルで写像するよりも簡単な方式であることを実感した。しかし、オブジェクト抽象化と標準化については、切り口が人によって複数存在する事以上に、実装問題と概念理解が固定していない等残された問題が多い。

7-2．問題点（オブジェクトのレイヤー分散）

線形加速器に於いては、エネルギーを上げるため基本的要素として加速管や真空装置、マイクロ波源、電磁石系、モニター等などの基本的装置を必要なだけ直線的に繰り返し配置されている。共通する装置の属性とサービスは上位クラスとして定義が容易であったし、あとは拡張と特化を行えば良く、オブジェクトの継承機能を非常によく適用出来る事をこれまで述べた。しかし、ネットワーク分散環境に於いての継承に関しては、満足して現在（1995年で）即使用できる程のものは無く、近々予定されているCORBA 等が目下検討中である。従って、一番大きな問題であったのが、GUIクラス階層のどの層までを物理層（３層構造の場合）のどこに配置するのがよいかであった。現状では層間メッセージが一番少ない所で分離すのが望ましいとした。今後、変更が予想される。

8．まとめ

加速器ドメインのオブジェクト分析を基にGUIオブジェクトモデル分析を進めて来たが、結果として特にコンポーネントクラス等の運用によって、より

１）汎用的な表現を可能にした事「標準オブジェクトの確立」、

２）再利用がし易くなった事「開発生産性向上」、

３）加速器構成装置の変更・追加・削除が極めて容易で数分で完了し、運転オブジェクトに変更を必要としないで可能になった等の特徴をもたらした。実装手法についてはまだ幾つかの興味深い研究分野が残されている。主にスタテックな分析を中心に取り上げたが、動的モデルについてのオブジェクトの振る舞い等については、重要であるが紙面不足で省略した。

参考文献

[1] オブジェクト指向分析（OOA）
Peter Cord, Edward Yourdon著, 株トッパン

[2] オブジェクト指向方法論OMT、Ｊ．ランボー著
1992, Prentice Hall, Inc. 株トッパン
OMT: Object Modeling Technique,

[3] Control Protocol:The proposed new CERN standard access procedure to accelerator equipment.
G.Baribaud, ICALEPCS1991, p591