社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－DPS－122（49）
2005－CSEC－28（49）
2005／3／23

関連する複数アプリケーションのデータ管理方式に関する一検討

山本英司，長島　勝，伊東輝顕，宮内直人
三菱電機(株) 情報技術総合研究所

複数のアプリケーションがフレームワーク等を介さずに連携して動作する環境において，アプリケーションが扱うデータ項目が他のアプリケーションが扱うデータ項目と関連している場合，一つのアプリケーションで加えたデータ項目の変更内容が他のアプリケーションに影響を与える場合がある．このような課題を解決するため，我々は複数のアプリケーション間で自動的にデータの変更を反映させることによって，作業量を減らし，データの不整合を防ぐ方式を検討した．

本論文では，複数のアプリケーション間で自動的にデータを同期させるデータ管理方式を提案する．

A Study on the Data Management Method for Related Multiple Application Software

Eiji YAMAMOTO, Masaru NAGASHIMA, Teruaki ITO, Naoto MIYAUCHI
Information Technology R & D Center, Mitsubishi Electric Corp.

In a computer system which is composed of multiple application software, when one software changes data contents, the modified data content might affect to another software's data contents. We investigate the data synchronization methodology among multiple application software.

This paper proposes the data management method of the multiple application software which are related each other.

1. はじめに

複数のアプリケーションがフレームワーク等を介さずに独立して動作する環境では，一つのアプリケーションが扱うデータ項目が他のアプリケーションが扱うデータ項目と関連している場合がある．その場合，一つのアプリケーションで変更を加えたデータ項目の内容を他のアプリケーションが扱うデータ中の関連するデータ項目に自動的に反映させることで，作業量を減らし，データの不整合を防ぐことが求められている．

本論文では，このような要求に応えるため，関連する複数のアプリケーション間で自動的にデータを同期させるデータ管理方式を提案する．

2. 目的

複数のアプリケーションがフレームワーク等を介さずに独立して動作する環境において，アプリケーションが扱うデータ中のデータ項目が他のアプリケーションが扱うデータ中のデータ項目と関連している場合がある．たとえば，FA(Factory Automation) 分野のアプリケーションシステムでは，図 1に示すように，

複数のアプリケーションが独立してデータ項目の入力や編集を行っている場合が多い．この例の場合，アプリケーション間で共有されるデータ項目とは，スイッチやセンサ等の FA 機器のプロパティを管理するためのタグ情報や，FA 機器の接続構成を管理するための FA ネットワーク情報である．

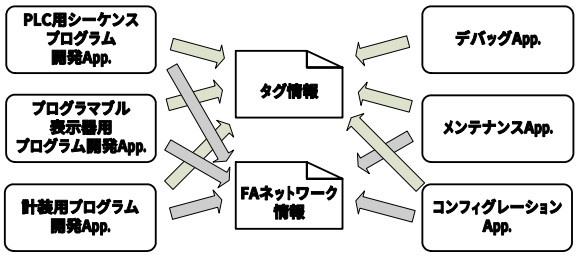

図 1　アプリケーションにおける関連するデータ項目の例

これらのデータ項目はアプリケーションが対象としている FA 設備が同じであれば，設定内容は同等のものとなるため，一つのアプリケーションでデータの入力や編集を行うと，他のアプリケーションが扱うデータ中の関連するデータ項目にその設定内容を自動的に反映することが求められている．これによって，データ変更の作業量を減らし，データの不整合を防ぐことできる．

本研究では，このような要求に応え，関連する複数のアプリケーションのデータ管理方式について検討することを目的とする．

3. アプリケーション・データ管理の課題

3.1. アプリケーションのバージョンアップに伴う互換性の確保への対応

複数のアプリケーションのデータを一元管理する場合，アプリケーションのバージョンアップに伴う互換性の確保への対応が課題となる．

アプリケーションに機能が追加されるバージョンアップの場合，アプリケーションが扱うデータの構造も変更されることが多い．一つのアプリケーションで開発が閉じている場合には，単純に既存のデータ項目のリザーブ領域にデータ項目を追加したり，旧バージョンのデータとの二重持ちをすることによって，バージョン間の互換性の問題を解決できる．しかし，複数のアプリケーションがそれぞれ独立して開発されている場合には，一つのアプリケーションのバージョンアップによってデータ構造が変更されると，他のアプリケーションのデータ構造にも影響を与えてしまうため，バージョンアップを容易におこなえず，また，それに伴う互換性を保つことが難しいという問題がある．

3.2. 小規模な運用環境への対応

複数のアプリケーションのデータを管理する場合，EAI(Enterprise Application Integration)[1]によるデータの一元管理がおこなわれている．この場合，EAI はアプリケーションからの要求にしたがって，EAI 内部のデータベース（以下，DB と呼ぶ）に格納されたデータを要求元のアプリケーション対応に変換してデータ入出力をおこなう．しかし，EAI 内部に通常の DB を用いる方法では，アプリケーションのバージョンアップによって機能が追加され，データ項目が増えた場合に，DB のテーブル構成の変更や，アプリケーションと EAI 間の API(Application Program Interface)の改修が必要となる．このため，互換性を保ちながらデータ構造の変更を伴うアプリケーションのバージョンアップをおこなうことは，システム改変を伴うため困難であった．この問題を解決するため，XML(eXtensible Markup Language)技術を用い，XML スキーマによるデータ構造定義をおこなうことでデータ構成をアプリケーション非依存にし，DB 構成や API の改修をせずにデータの再利用を可能にするデータ管理方式[3][3]が登場している．

しかし，これらはいずれもネットワークで接続されたクライアントーサーバ型による大

規模なシステムでの運用を想定しており，小規模な工場などのネットワークが構築されていない運用環境においては，スタンドアローンのコンピュータでDBを構築する必要があるという問題がある．例えば，FA分野においては，製造制御用のプログラムを開発するアプリケーションがあり，それをPLC(Programmable Logic Controller)と呼ばれるFA用制御計算機にダウンロードするという運用がおこなわれている．ダウンロードした後は,PLCでもデータの編集をおこなう必要がある．しかし,PLCは多くの場合EthernetのI/Fを持たないか，あるいは，持っていてもEthernetに常時接続していないため，データフォーマットを記述したXMLスキーマを入手することが困難である．また，格納できるプログラム容量の問題からDBを含むようなサイズの大きいファームウェアを実装できないという問題がある．

4. アプリケーション・データ管理方式

本章では本研究で提案している複数のアプリケーションのデータ管理方式の概要を示す．

4.1. プロジェクトデータ

本手法では，プロジェクト単位で複数のアプリケーションのデータを一つにまとめて管理する．まとめられたデータを以下，プロジェクトデータと呼ぶ．プロジェクトデータは，例えばFA分野のアプリケーションであれば，FA設備単位で切り分けられる．つまり，一つのFA設備に組み込まれたPLCやプログラマブル表示器などのそれぞれのプログラムとデバッグ用データ，メンテナンス用データ，FAネットワーク構成情報，タグ情報等を一つのデータにまとめて管理する．このとき，FAネットワーク構成情報やタグ情報等の複数のアプリケーションが共通して扱うデータ(以下，共通データ）を図 2に示すように一つに集約させる．

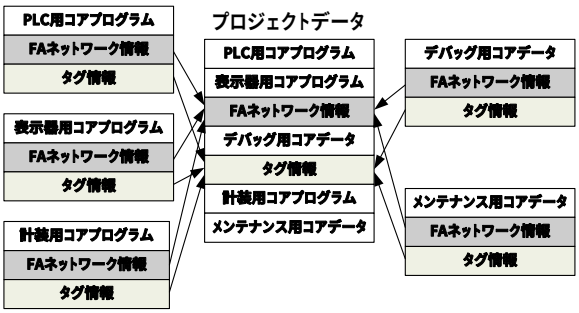

図 2　プロジェクトデータの構成

図３に示すように，それぞれのアプリケーションはプロジェクトデータが記録されたプロジェクトデータファイルから必要なデータを抽出して編集等をおこない，編集後のデータをプロジェクトデータファイルに格納する．図 3のプロジェクトデータファイル庫は，プロジェクトデータファイルを格納する機能を指している．

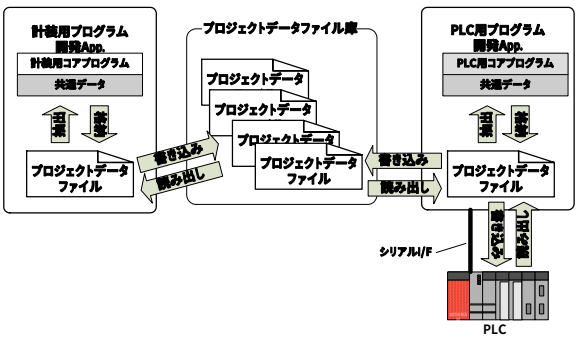

図 3　アプリケーションによるプロジェクトデータファイルからのデータ項目抽出／格納

共通データはプロジェクトデータ中で一つに集約される．全てのアプリケーションが同一データを編集／参照するため，一つのアプリケーションで編集すればその設定内容はどのアプリケーションで参照しても同じ設定内容になる．共通データを複数のアプリケーションが同時に編集可能な場合には競合が起こり得るため，これを検知する手法に関しては4.2項で述べる．

4.2. データ構造

4.2.1. プロジェクトデータファイルの要素

本手法ではプロジェクトデータを，図 4に示すように，プロジェクトデータの構造情報（以下，データ構造情報）が記録されるデータ構造記録部と，プロジェクトデータそのものが記録されるデータ記録部の，２種類の記録部からなるプロジェクトデータファイルとして管理する．

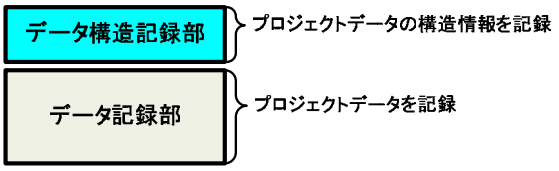

図 4　プロジェクトデータファイルの要素

データ構造記録部が常にプロジェクトデータと共にあるため，アプリケーションはこれを参照し，後述するデータ項目 ID を元にデータ記録部中の操作対象のデータ項目の位置を特定して目的のデータを抽出する．この際，アプリケーションはデータ構造記録部中の未知のデータ項目を無視するため，バージョンアップにより新規のデータ項目が追加しても，操作対象のデータ項目の抽出が可能となる．

4.2.2. データ項目構造情報

データ構造情報において，各データ項目に対する構造情報は，図 5に示すように，データ種別 ID とデータ項目バイト数，およびデータ項目の版番号を表すシーケンス番号の 3 種類のデータから成る．以下，これらをデータ項目構造情報と呼ぶ．

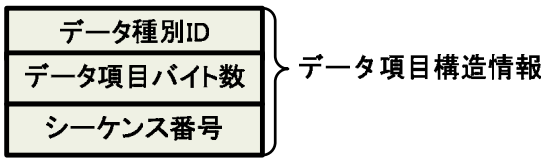

図 5　データ項目構造情報の構成

データ種別 ID はデータ項目の種別を表す ID であり，データ項目バイト数は対応するデータ項目に格納されるデータのバイト数を表している．

シーケンス番号は，共通データを複数のアプリケーションが同時に編集をした場合の競合を検知するために用いられる．図 6に共通データの競合を検知するための手法を示す．

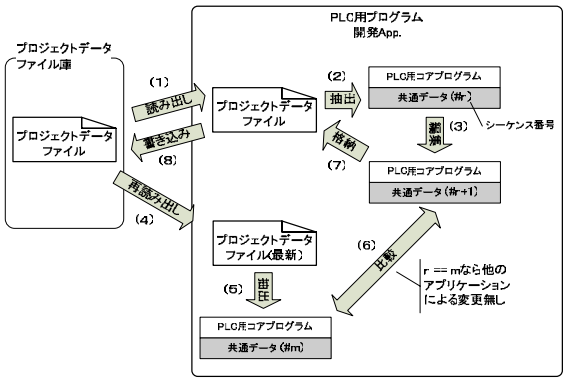

図 6　シーケンス番号による共通データの競合の検知

それぞれのアプリケーションはプロジェクトデータファイル庫からプロジェクトデータファイルを読み出し，アプリケーションが扱うデータを抽出する．データ項目を編集した場合には対応するシーケンス番号が 1 増やされる．編集後にデータ項目をプロジェクトデータファイルに格納する際に，最新のプロジェクトデータファイルをプロジェクトデータファイル庫から再度読み出し，対応するデータ項目のシーケンス番号を比較して最新データにおけるシーケンス番号が 1　少なければ他のアプリケーションによる編集がないことを示しているため，プロジェクトデータファイル庫に書き込み可能となる．それ以外の場合には他のアプリケーションによって編集されたこと示している．なお，競合を検知した場合の処理に関しては，アプリケーションの設計ポリシーに依存するため，本論分では述べない．

4.2.3. データ構造情報の表現方法

プロジェクトデータに含まれるデータ項目がネスト化されていないフラットな構成であれば，単純にデータ項目構造情報をスタック

することでデータ構造情報が表現できる．しかし，通常はプロジェクトデータに含まれるデータ項目は，いくつかのデータ項目が集まってより大きい単位のデータ項目を形成し，それがネスト化されているため，データ構造情報の表現方法もネスト化に対応していなければならない．ネスト化されたデータ構造情報に対応するための表現方法として，本研究では，（1）含まれる全データのデータ構造を一つのデータ構造記録部に記録して，データ記録部と完全に分離する方法と，（2）ネスト単位ごとにデータ構造記録部とデータ記録部で表現してネスト化する方法の２種類の方法を検討した．

4.2.3.1. 全データのデータ構造を一つのデータ構造記録部に記録する表現方法

含まれる全データのデータ構造を一つのデータ構造記録部に記録する表現方法では，図7に示すように，データ項目記述情報は，それ以上小さいデータ項目に分解できない最小単位のデータ項目構造情報だけからなる．複数のデータ項目から構成されるデータ項目に関しては，それを構成するデータ項目のスタックとして表現されるため，そのデータ種別 ID がデータ構造記録部に現れることはない．

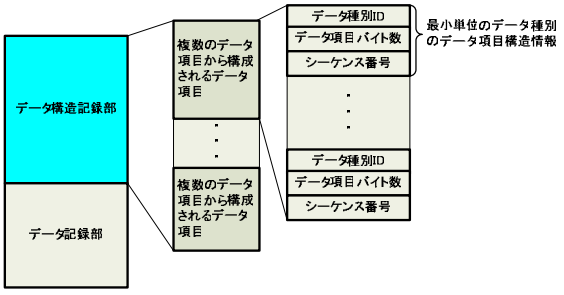

図 7　全データのデータ構造を一つのデータ構造記録部に記録する表現方法

この表現方法では，データ種別 ID はそれだけで，どのアプリケーションの，どのデータ項目を構成するデータ項目なのかが判る必要がある．このため，データ項目の持つ意味が異なれば必ず異なるデータ種別 ID を持つ．例えば，機器の名称を格納するデータ項目であれば，FA ネットワーク情報中に存在する機器名称なのか，PLC 用プログラム中に存在する機器名称なのかによってデータ種別 ID は異なっていなければならない．このため，データ種別 ID 自体の管理方法が問題となる．また，4.2.2項で示した共通データの競合の検知を実施する際に，他のアプリケーションによって最新のプロジェクトデータファイルが書き換わっている場合には，全データ構造情報の中から対応するデータ項目を見つけなければならないため，シーケンス番号を比較する相手のデータ項目の検索が困難であるという問題がある．

4.2.3.2. ネスト単位ごとにデータ構造記録部とデータ記録部で構成する表現方法

ネスト単位ごとにデータ構造記録部とデータ記録部で構成する表現方法では，図 8に示すように，データ記録部に格納された個々のデータ項目の実データが更にデータ構造を持つことができる．データ項目の実データが更にネスト化されたデータであるかどうかを判断するためにネスト化識別子が加えられる．

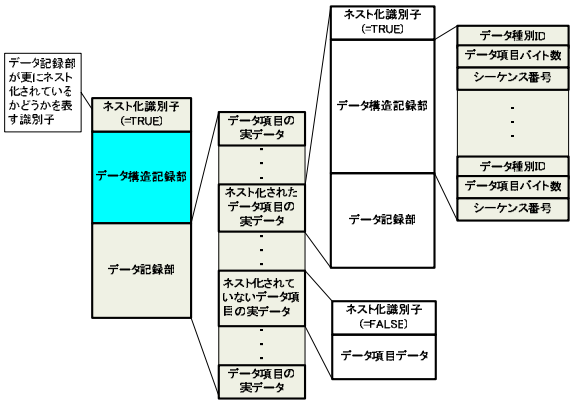

図 8　ネスト単位ごとにデータ構造記録部とデータ記録部で構成する表現方法

この表現方法では，同等のデータが格納されるデータ項目には同じデータ種別 ID を与える．例えば，機器の名称を格納するデータ項目であっても，FA ネットワーク情報中に存在する機器名称であっても，"機器名称"を表すデ

ータ種別IDを与えることができる.このため,データ種別 ID の管理が容易になる．また,4.2.2項で示した共通データの競合の検知を実施する際に，他のアプリケーションによって最新のプロジェクトデータファイルが書き換わっている場合でも，一つ一つのデータ構造情報は小さいため，シーケンス番号を比較する相手のデータ項目の検索が容易である.

以上の点から，この表現方法は，前述の全データのデータ構造を一つのデータ構造記録部に記録する表現方法に比べて優れていると言える.

ただし，ネスト化識別子が必要となるため，データ量が増加するという問題や，ネスト化されたデータ項目では，構成要素であるデータ項目のバイト数が増えた場合などには，上位のデータ項目のバイト数もそれに応じて増やす必要があるという問題がある.

5. おわりに

本研究では，関連する複数のアプリケーションのデータ管理方式を検討している．本論文では，プロジェクト単位で複数のアプリケーションのデータを一つにまとめたプロジェクトデータとその構造について提案した.

本手法を用いることで，アプリケーションのバージョンアップに伴う互換性を確保し，小規模な運用環境で運用可能なデータ管理が可能となる.

今後は，データ構造情報とプロジェクトデータの整合がとれていない場合の処理など，実運用時に発生する可能性のある問題への対処方法について検討を進める．また，FA 用アプリケーションのプロトタイプを開発して本方式の効果について確認する予定である.

参考文献

[1] 日本ユニシス社 EAI トップページ:
http://www.unisys.co.jp/solution/eai/

[2] TIS 株式会社 Enterprise Knowledge Portal:
http://www.tis.co.jp/product/XML/SOLUTION/KNOWLEDGE.html

[3] NEC 社 BizConnector/Converter :
http://www.ace.comp.nec.co.jp/bizcon58/conv.htm