ユビキタスシステムを統治するシステムアーキテクチャ

坂本 久・稲垣 嘉信・島津 秀雄

要　旨

ICカード/タグやセンサ、各種端末や装置を多数組み込んだシステムでは、システム全体で守るべきルールを末端まで津々浦々に伝達することが困難です。一部のシステムでは、内部で最新版に更新していく仕組みを持っていますが、個別のアプリケーションや装置でこのような仕組みを持つことは困難ですし非効率でもあります。本稿では、近未来のユビキタスサービスに必須な基盤として統治型アーキテクチャという新しい概念を紹介します。そして、この概念に基づいてユビキタスサービスのアーキテクチャの構想を描きます。

キーワード

●統治型アーキテクチャ　●双方向型統治モデル　●ポリシ

1. はじめに

今日、多数の各種端末や装置を組み込んだシステムが普及しています。これら端末や装置は、利用者の利用状況の変化に応じて頻繁に抜き差しされます。そうなると、システム全体で統一したルールに基づいて動作する場合に、その末端の端末や装置までのルール伝達が困難になります。例えば、システム中にインストールされている各種アプリケーションのバージョン（版）を手動で抜けなく最新版に維持することは困難です。ウィルス対策アプリケーションでは、内部で最新版に維持する仕組みを持ちますが、各アプリケーションや装置（装置の内部にもアプリケーションを内蔵していることが多いです）で独自にこの仕組みを持つことは困難かつ非効率です。本稿では、この問題を解決する「統治型アーキテクチャ」という新概念を紹介します。

2. システム全体の統治の必要性

統治が必須な例として、企業内の情報セキュリティシステムが挙げられます。例えば、イントラネット上の端末で利用するアプリケーションを統一させることを、文書や口頭で徹底させるのは困難です。これに対し、中央から実行を許可するアプリケーションとバージョンを各端末に伝達すると、各端末で、そのアプリケーションのみを許可し、それ以外は拒絶するシステムが導入されたイントラネットは、「中央から統治されている」といえます。別の例として、ウィルス対策アプリケーションのシグナチャファイル（ウィルスの動きを記述したデータ）配信があります。各端末で最新のシグナチャファイル維持のためにシステムが定期的に配信をしています。

3. ITシステムアーキテクチャの変遷

統治の観点から見ると、ITシステムは次のように変遷してきたということが分かります（**図1**）。

1) 非統治型モデル

統治をする仕組みを持たないアーキテクチャです。今日の一般的なシステムはこの形態をしています。

2) 単方向型統治モデル

サーバからクライアントのコンピュータに対して、一方的に指示が送信され、コンピュータはそれに従います。

3) 双方向型統治モデル

サーバから末端装置に対して、統治に関する指示が送信され、末端装置はそれに従います。その後、末端装置は、各

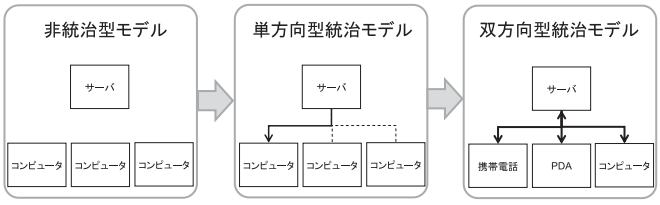

図1　ITシステムアーキテクチャの変遷

部分の状態情報をサーバに返信します。サーバは、それらを集約して、必要に応じて新たな統治に関する指示を送信します。これを続けていきます。

第2章のウィルス対策アプリケーションや版管理アプリケーションの例は、双方向型統治モデルです[1)]。今後は、この仕組みを各アプリケーションで持つのではなく、アーキテクチャの要素として組み込まれ、各アプリケーションは、その機能を呼び出すようになると思われます。

4. 統治型モデル

統治型モデルの動作を**図2**で説明します。

サーバコンピュータとクライアントコンピュータには、統治を実現するアプリケーション（例：ウィルス対策アプリケーションや版管理アプリケーションなど）が導入されているとします。管理者はクライアントコンピュータを統治するための指示の定義（ポリシ）を行います。ポリシの形式は、アプリケーションごとに異なりますが、基本的には、

＜対象となるアプリケーションまたは装置＞

＜守らなくてはならない条件＞

＜守られていない場合の対処方法＞

の3つ組集合になっています。ポリシは、適当な時間間隔でサーバからクライアントコンピュータに配信されます。クライアントコンピュータからポリシを要求することもあります。

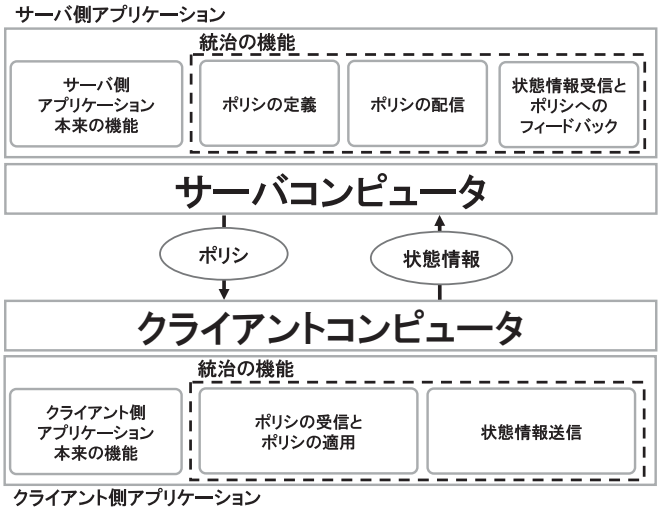

図2　双方向型統治モデルの構成

クライアントコンピュータは、入手したポリシの中で、自分の管理下にあるアプリケーションや装置に関する記述を見つけ、＜守らなくてはならない条件＞が守られているかを調べ、守られていればそれで終了しますが、守られていない場合は、＜守られていない場合の対処方法＞を実行します。また、これらの実行結果である状態情報をサーバに返信します。

サーバコンピュータはクライアントコンピュータから受信した、またはクライアントコンピュータに要求して取得した状態情報を総合的に判断し、最新のポリシに反映して、ポリシを配信することがあります。このように、サーバコンピュータとクライアントコンピュータが共同して、システム全体に統一したポリシを配信することが可能になっています。

5. ポリシの記述例

ここでは、社内で使用されるクライアントコンピュータ内のファイルを、管理者が定義したポリシに従って、フォルダごとに暗号化するシステムのポリシ記述例を説明します。ポリシには、以下の内容が記述されます。

- ポリシ自身の版管理をするためのバージョン番号
- 暗号化対象となるフォルダ群
- 暗号化対象フォルダ内のファイルに与えるアクセス権限（誰にどういうアクセス権を付与するかということ）
- 暗号化対象となるフォルダ内のファイルを調べ、ファイルがポリシで指定した状態ではないときにする処理

このシステムのポリシ例を**図3**に示します。POLICYタグで、ポリシ自身のバージョンを記述します。クライアント側でポリシを受信したプログラムは、受信したポリシと現在適

```
<POLICY VER="1.0.0">
    <CODE>
        <TARGET PATH="C:¥A_INTERNAL">
        <ACCESS RULE="A社社員:rw">
        <FILEFOUND ACTION="encapsle.exe –x">
    </CODE>
    <CODE>
        <TARGET PATH="C:¥A_PARTNER">
        <ACCESS RULE="A社社員:rw、A社パートナー社員:r">
        <FILEFOUND ACTION="encapsule.exe –x -d">
    </CODE>
</POLICY>
```

図3　ポリシの記述例

用中のポリシを比較し、同じ場合は何もしませんが、新しければ受信したポリシを最新ポリシとします。CODEタグ内部では、対象ごとに守るべき条件と守られていないときの対処方法が記述されます。<TARGET PATH>では、対象物として、フォルダ群が記述され、<ACCESS RULE>では、守るべき条件を規定します。ここでは、C:￥A_INTERNALフォルダ下のファイルは、「A社社員は編集可能だが、それ以外の人はアクセス許可しない」ですし、C:￥A_PARTNERフォルダ下のファイルは、「A社社員は編集可能だが、A社のパートナー社員は閲覧のみ可能、それ以外の人はアクセスを許可しない」という定義です。<FILEFOUND ACTION>には、守られなかった場合の対処方法を規定します。おのおのの場合で、異なる引数形式でファイル暗号化プログラムを実行します。

6. ポリシで統治するシステムアーキテクチャ

ここまでは、各アプリケーションで、統治型モデルを実現してきた例でした。ここでは、その仕組みを基本システムの一部として、各アプリケーションに提供する統治型アーキテクチャを説明します。統治型アーキテクチャは、サーバ側とクライアント側に、基盤層とその基盤を用いてサービスを構築するサービス層を持ちます。基盤層では低水準な実装や環境を隠ぺいし、サービス層にインタフェース（APIライブラリ）を提供します。**図4** で、サーバ側、クライアント側それぞれで実行される統治型処理を説明します。

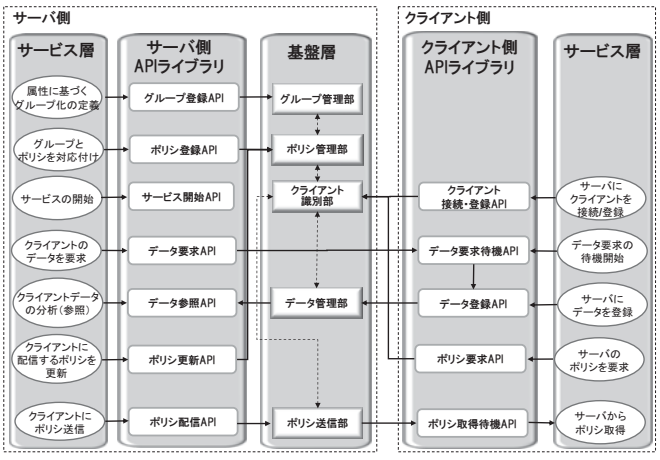

図4　統治型アーキテクチャのAPIと動作の流れ

サーバ側では、クライアント側を個々ではなく、クライアント側の属性に従って分類し統治するため、サーバ側APIライブラリの「グループ登録API」によりグループを定義します。サーバ側は、グループごとにポリシを保持します。　サーバ側でサービスが開始されると、クライアント側はクライアント側APIライブラリの「クライアント接続・登録API」によりサーバ側に接続し、クライアント側の属性情報を送信し、サーバ側に登録を要求します。サーバ側ではクライアント側からの登録要求を受信すると、クライアント識別IDを発行し、属性情報をもとにクライアント側を既存のグループに分類します。これ以後、クライアント側は、「ポリシ取得待機API」を実行しポリシを受信できるように待機します。

サーバ側は、各クライアント側から状態情報が送信されると、「データ参照API」を実行し、送信されてきた状態情報を参照します。サーバ側は、「データ要求API」を実行し、任意のクライアント側の状態情報を要求することもできます。収集した状態情報を、事前に登録してある分析プログラムで分析し、分析の結果、現在のポリシを更新する場合があります。このときは「ポリシ更新API」によりポリシを更新し、「ポリシ配信API」により、必要なクライアント側にポリシ配信します。

統治型アーキテクチャでは、これらAPIにより、各アプリケーションに、統治型モデルを容易に組み込めます。

7. 統治型アーキテクチャの適用例

ここでは、統治型アーキテクチャを用いてさまざまなシステムが動作する仮想的な例を説明します。**図5** は、版管理アプリケーション、ウィルス対策アプリケーション、情報漏えい対策アプリケーションが独自のポリシ制御の仕組みを持つのではなく、統治型アーキテクチャを用いて動作する仮想的な例です。

サーバコンピュータ上の各サーバプログラムは統治型アーキテクチャのサーバ側APIライブラリ及び基盤層を共有します。一方、クライアントコンピュータ上の各クライアントプログラムは、クライアント側APIライブラリを共有します。

各サーバプログラムは自身のクライアントプログラムのポリシとして、クライアントプログラムの種別や条件に応じ、アプリケーションごとに1つ以上のポリシを保持します。各サーバプログラムは、クライアントプログラムの種別や条件

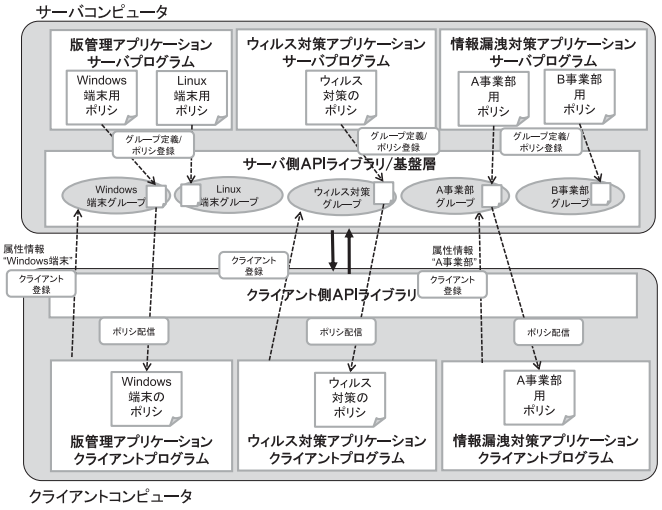

図5　統治型アーキテクチャとアプリケーションの動作

に応じ、クライアントプログラムを分類するグループを定義します。例えば、版管理アプリケーションは、クライアントプログラムがWindows端末用かLinux端末用かで異なるポリシを準備し、情報漏えい対策アプリケーションは、A事業部用のファイル保護ポリシとB事業部用のファイル保護ポリシを別々に準備します。そのため、版管理アプリケーションサーバプログラムはWindows端末グループとLinux端末グループを定義し、それぞれのポリシ（Windows端末用ポリシとLinux端末用ポリシ）を定義します。情報漏えい対策アプリケーションプログラムは、A事業部グループとB事業部グループを定義し、それぞれのポリシ（A事業部用ポリシとB事業部用ポリシ）を定義します。

クライアントコンピュータ上の各クライアントプログラムは、起動時にクライアント側APIライブラリにより、クライアントプログラムの種別や条件を表す属性情報を付加した上で、自分自身を登録します。版管理アプリケーションでは、クライアントプログラムはWindows端末の版管理をする場合、“Windows端末”という属性を送信します。また、情報漏えい対策アプリケーションでは、A事業部の権限で情報を保護する場合は“A事業部”という属性を送信します。クライアントプログラムからの登録要求を受信したサーバ側基盤層は、属性情報に基づき、各クライアントプログラムをグループに分類し、対応するポリシを配信します。

このように、各アプリケーションは、属性やポリシを定義するだけでよく、クライアントプログラムのグループ化やポリシの配信機能などを個別に実装する必要がなくなります。

8. むすび

統治型モデルの最初の応用領域は、情報セキュリティ分野であり、既に各アプリケーションで統治型モデルに設計されています[2~4]。今後、セキュリティシステム同士の緊密な連携が必要になるときに、個々のアプリケーションを超えた横断的なポリシ制御ができれば、セキュリティ強度の向上に貢献します。一方、近未来の統治型電力システムであるスマートグリッドも、風力発電装置や太陽光発電装置を電力系統に接続する場合、統治型モデルはシステムの安全性や柔軟性のためにも必須になります。このように、ユビキタスサービス基盤として、統治型アーキテクチャの商用化が待たれます。

参考文献

1) ポリシベースによるQoS制御 (小泉, 平島, 三宅 オーム社 (2001))
2) NECグループにおけるITによる情報漏えい防止の取り組み～情報漏えい防御システムARGUS～（田村他 NEC技報,Vol.60 No.1, 2007/1）
3) 企業におけるコンテンツセキュリティ（足尾他 情報処理学会 第70回全国大会 2008/3）
4) コンテンツセキュリティにおける網羅性の実現（坂本他 情報処理学会 第70回全国大会 2008/3）

執筆者プロフィール

坂本 久
NECシステムテクノロジー
システムテクノロジーラボラトリ
研究エキスパート

稲垣 嘉信
NECシステムテクノロジー
システムテクノロジーラボラトリ
情報処理学会会員

島津 秀雄
NECシステムテクノロジー
システムテクノロジーラボラトリ
所長