フェデレーション パターン

pv@inarcadia.co.jp

v.0.9, Mar. 20, 99

1. フェデレーション機構

フェデレーション（連合化）機構は、それぞれ固有のポリシの許に存在するリソースの間において、動的に相互運用を可能とする機構を提供する。フェデレーションにより、既存のリソースに対するそれまでの利用の仕方はそのままに、相互運用に必要とされる新たな属性[1]や振舞いを導入することが可能となる。フェデレーションに基づくアプリケーションでは、リソースが存在している既存の世界に対応して、その世界観が有効であるスコープ（範囲）が導入され、これは自治区と呼ばれる。即ち、既存の世界はそれぞれ自治区としてカプセル化され、それらの相互運用に必要なポリシ、属性、そして振舞いがフェデレーションをスコープとする新たな世界として構成される。

既存のリソースはそのままに、それを恰も新しい構造や振舞いを有したオブジェクトであるかのように扱いたい場合に、フェデレーションは有効である。また、自分たちとは異なる未知の世界と出会い、既存の世界はそのままに、それら未知の世界と共存するような場合にもフェデレーションが適用できる。

1.1. 例題

ある異なる国の研究機関同士が、互いの研究成果が相互に補完しあう可能性のあることが判り、試験的に研究者を相互派遣して研究活動を推進することになったとしよう。ホゲホゲ研究所に相手の研究機関から派遣されてきた研究者はメムノン博士である。優秀なメムノン博士の働きは期待以上であり、この相互派遣が研究活動の進捗に大きく寄与するだろうことはもはや間違い無いと思われる。しかし丁度一ヶ月が経過し、その研究対価を算出する段階になって厄介な問題が生じることとなった。

① ホゲホゲ研究所の規定では手当の算出に年齢に応じた加算額がある。メムノン氏は申請では35歳とのことであった。しかしメムノン氏の国では独自の暦を用いており、その電子化された個人情報の生年月日は以下のようであった。

生年	月	日	年齢
24年	6月	15日	35歳

調べたところ、以下のようになっていることが判明した。

現在	現在の暦年	月数／年	日数／月

[1] 属性の持つ値の意味（解釈）についても新しい意味付けが可能である。

メムノン氏の邦	59年	10ヶ月	30日固定
西暦	1999年	12ヶ月	28～31日

これらからメムノン博士の年齢は29歳となり、また生年月日は1970年となることが判った。

② 同じくホゲホゲ研究所の規定では妻帯者にも手当てが支給される。メムノン博士の邦では一夫多妻制であり、また女性の結婚年齢に就いての制限もないため、メムノン博士も2人の妻を有しており、それぞれ当方の年齢に換算して15歳、13歳となっている。従って16歳未満の結婚が規制されている当方では、メムノン氏は独身と見なされ、代わりに2人の養女がいることになる。

更に15歳の第一夫人は来月には満16歳を迎えるため、メムノン博士は来月にはめでたく妻帯者に復帰し、同時に第一夫人は当方においても養女から正式な妻と認められることになる。

以上のような後天的要請をどのように扱へば良いだろうか。既存システムに変更を加えること自体はさほど難しいことではないが、如何にもad-hocな上記のような対応をそのまま実現することは、システムの安定性と拡張性を大きく損なうことになるのは明らかである。また当該の問題は、

- 他の研究機関とも相互派遣が行われる可能性があり、詳細は異なるものの同様の要請は今後も発生する可能性がある。（根本的な解決策が必要）
- 当方の研究者が相手方の研究機関に出向することもあり、当方におけるのと逆の問題が相手方においても発生すると思われる。（相互利益、mutual benefitの追求）
- いわゆる給与計算や社員情報管理のシステムに上述のad-hocな処理を取り込むのは、それらシステムの本来の趣旨と照らし合わせ、明らかに過剰仕様である。（既存システムは十分に合理的[2]）

といった側面からの配慮もなされることが望ましい。

1.2. 脈絡

自分たちとは異なる世界観を背景とする世界と出会った際にも、それなりに対応が可能なflexibilityとopennessを提供する。但しa priori に全ての起こり得るであろう変異への対応を準備しておくのは現実的でない。a posteriori にそのような自由度と開放性を追加できる枠組みを用意する。

[2] 現行制度において、一夫一婦制、並びに婚姻可能年齢の見直しがなされる可能性は殆ど無視できるからである。

1.3. 課題

どのようなシステム（あるいは世界観、モデル）であれ、時間の経過とともにいつかは当初全く予想もしなかったような利用の仕方と出くわすものである。従ってシステムは本来、そのような出会うかも知れない要求にも対応できるような自由度と開放性を有していることが望まれる。他方、遭遇するかも知れないあらゆる変更要求に前もって備えることは徒にシステムを複雑にし、それはシステム開発の質自体を低下させる危惧を孕む。そして何よりも現実的ではない。望ましいのは機能変更や機能追加が必要になったその時に、それが行えるような開放性をシステムの基本機構として備えておくことである。つまり適応できることが望まれる分けだが、それは単純にあるものから別のものへと変態するだけでは済まない。望ましいのはその機能変更や機能追加を欲したその脈絡、ないし環境においてはそれに適した変態がなされ、そうでない場面においては元の単純さが保持されていることなのである。背景色に応じて自らの体の色を変化させる Chameleon のように、「適切さ」に対応した適応の機構は以下のような特性を備えることになるだろう。

- 一般にソフトウェアに対して変更を加えることは、瑕疵を持ち込み易く、ソフトウェアが当初有していた一貫性や無矛盾性を損なう傾向があり、またコスト的にも高価となることも少なくない。変更伝播の影響を既存システムに及ぼすことがなければ安全であり、品質の維持と試験コストの抑制につながる。
- 変更を、その背景をなす主要な動機（意図）に応じて局所（カプセル）化することができれば、所謂「関心事の明確な分離[3]」が達成でき、それにより複雑性が抑制され、保守容易性が維持できる。
- 自由度と開放性を齎す様々な意匠を個別に持ち込むことは複雑度を増加し、保守性を損なう。枠組みとしてそれに従ったこれら意匠の適用により、理解容易性と変更容易性が維持できる。
- 微視的なレベル[4]から巨視的なレベル[5]まで、多様な変更のスケールに対し、同一の枠組みで対処できる。

1.4. 解決策

システムはある時点での世界観を反映したスナップショットと見なせる。すると変更とは時間経過に伴う世界観の変化にシステムを追随させようとする要求のことである。変化は一様ではない。世界観の数だけ存在する。つまり世界観に対応させ、変更は即ち新しい世界観に従ったシステムの実現として表現する。変わらない部分はそれまでの世界観を表現した

[3] “clean separation of concerns (Dijkstra)” のこと。
[4] 例えば、ある属性を更に分割した部分毎に扱うような場合。
[5] 例えば火星人と相互運用する必要が生じた際には、我々は「地球」という自治区になる。

システムをそのまま「継承」すればよい。

このような指針に基づいた枠組みは自ずから世界観の有効なスコープを意識することになる。数学における関数が、それが意味をもつ「有効範囲」と対になって存在するように、ある世界観の実現であるシステムにも、同様にその有効範囲を導入する。

ある有効範囲を有したシステム（ないし世界観）を自治区と呼ぶ。自治区同士の相互運用を可能とする上位の構造をフェデレーションと呼ぶ。従ってフェデレーション自体もまた自治区となり得る。世界観に対応する自治区は、全ての認識可能な分節の単位で（可能性として）存在しうる。

フェデレーションはある世界観に就いての定義を保持し、自身に属（参画）している自治区に対し、原則として各自治区に存在するリソースが、フェデレーションの定義する世界観に従ったリソースとして振舞うことを要請する。従って、フェデレーションは参画している自治区のリソースを、自身が定義する普遍的なリソース[6]として提供することができる。

自治区は特定のフェデレーションに参画し、当該フェデレーションが定義する世界観に則り、自身が保持しているリソースの対応づけ（写像）を保証する義務を負う。代わりに、各自治区は個別に相互の世界観の相違を補完する労から開放され、唯一、フェデレーションに対してのみ、差分を保証すれば済む。

⇒ 例題では便宜的にメムノン博士の邦の世界観をそのままフェデレーションの世界観としよう[7]。従って当方がそのフェデレーションに参画し、そこに定義された世界観を遵守するための差分をフェデレーションに対して提供することになる。

当方の自治区としては2つの要請に応えなければならない。一つはフェデレーションに定義されたリソースとして提供される対象を、自身のリソースとして扱うための**サロゲート**の用意である。もう一つは自身のリソースをフェデレーションに定義されたリソースとして振舞わせるための**アダプタ**の提供である。

フェデレーションに定義される普遍的なリソースは**ピボット**と呼ばれ、それを支点としたリソース、サロゲート、アダプタ、そしてフェデレーションと自治区の関係を模式的に示す（図 1）。

---

[6] 当然、ここで云う「普遍性」はスコープ付きである。即ち、フェデレーションがスコープの範囲である。

[7] 現実的には他にも同様のしかし詳細は異なる変換の必要性が生じる可能性もあるため、そのような潜在的な当事者を含め、合意の上でフェデレーションの有する世界観を決定することになるだろう。どのような世界観が定義されるかはそこに参画する自治区の合意に完全に依存するが、実際は参画する自治区全体の mutual benefit を最大にするような選択がなされることになるだろう。この詳細に関しては実装の節で議論する。

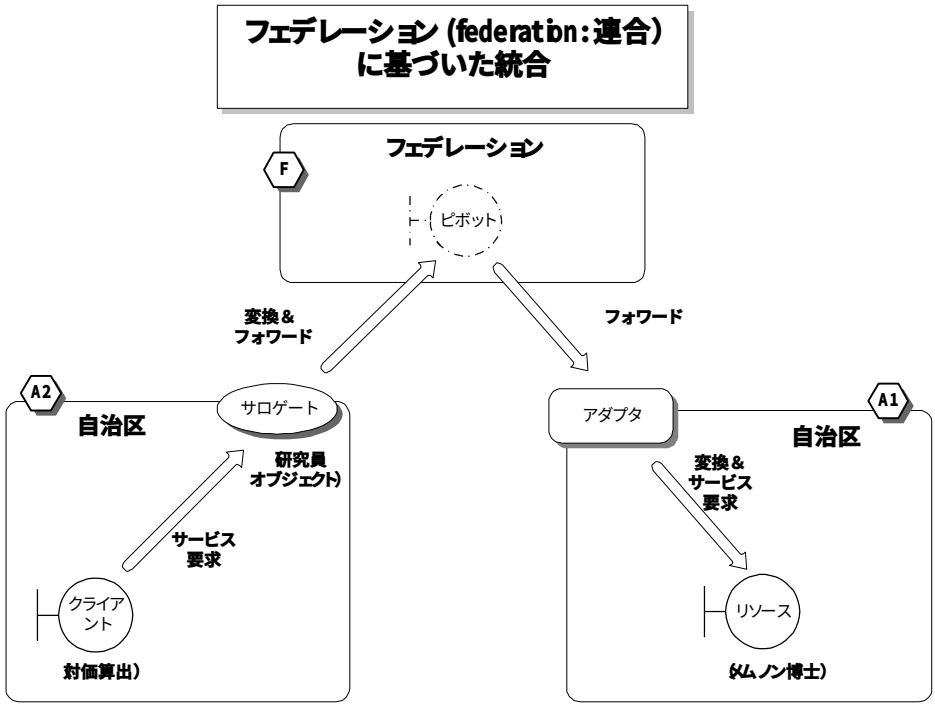

図 1 フェデレーションと自治区

1.5. 構成

フェデレーションの構成は大きく2つのレイヤから構成される。一つは自治区とそららが参画するフェデレーションに関するオブジェクトの構成であり、もう一つは実際のランタイムに出現するサロゲート、ピボット、そしてアダプタの各オブジェクト群である。

フェデレーションと自治区の構成に関しては単純である。フェデレーションは以下のサービスを参画している自治区に対し、提供する。

- 参画を望む自治区の登録
- 参画している自治区の管理
- フェデレーションに定義されているモデル情報の開示
- 利用できるリソースを提供している自治区の照会
- リソースの写像対象となっているピボットの提供

一方、自治区は以下のサービスをフェデレーションに対し、提供する。

- フェデレーションに定義されたモデル情報に対応し、アクセスを認めるリソースの写像に関する情報の開示（アダプタのモデル情報）
- ランタイムにおけるアダプタ オブジェクトの提供

またリソースを利用するクライアント側の自治区は、

- ピボット（の先のリソース）をラッピングし、既知インタフェースによるアクセスを代理するサロゲートの提供

を行う。

他方、ランタイムに出現するオブジェクト群の構成は以下のようになる。

- ピボット（の先のリソース）をラッピングし、クライアントに対して既知のインタフェースを提供するサロゲート
- リソース（をラッピングしたアダプタ）が写像されているピボット
- リソースをラッピングし、ピボットに対して既知のインタフェースを提供するアダプタ

ピボットを除き、サロゲート、およびアダプタはロール呼称であり、実態はラッパ、ないしコンバータである。以上の様子を図 2に示す。

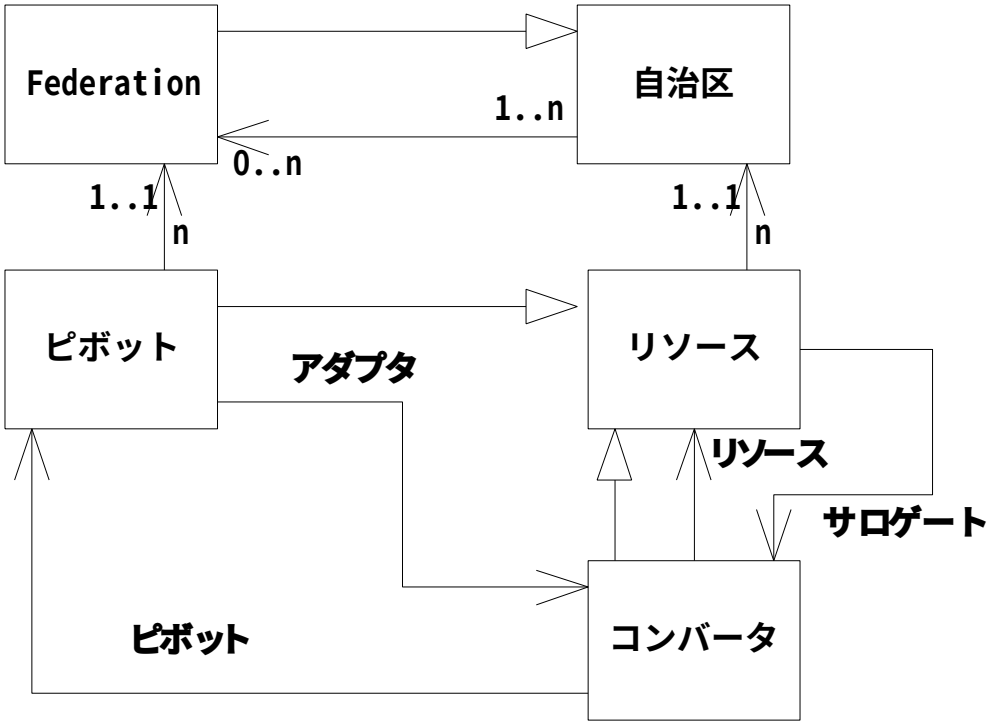

図 2 フェデレーションの構成（クラスグラフ）

1.6. 動的側面

フェデレーション機構の中核をなす振舞いは、リソースの利用を、フェデレーションを経由し、自治区間に跨って可能とすることである。それは即ち、クライアントに指示されたリソースに至るチャネルを、ピボットを最小不動点としてサロゲート、およびアダプタを介して確立することである。典型的なシナリオを示す。

① クライアントは、リソースを示す情報（オブジェクトID）、およびそのアクセスに用いる既存クラス型[8]を引数として、自身の自治区にリソースへのチャネルの確立を依頼する。

② 当該自治区は、リソース情報を引数として、当該リソースの写像が可能なピボット型候補の照会をフェデレーションに対して行う。

③ フェデレーションは参画している各自治区に対し、以下を行う。

a) 当該リソースが存在するかどうかの検出を依頼する

b) 存在が確認されたなら、その照会先自治区に対し、当該リソースのピボットへの写像を可能とするアダプタ型候補を要求する

④ 得られたアダプタ型候補から、それぞれが対応するピボット型候補を取得し、照会元の自治区に回答する。

⑤ 照会元自治区は得られたピボット型候補より、クライアントの要求したアクセスのためのクラス型として振舞え、且つピボット候補のいづれかとの写像が可能なサロゲート型を一つ選択する。当該サロゲート型に対応したピボット型候補を選択する旨をフェデレーションに指示する。

⑥ フェデレーションは指示されたピボット型のピボットを生成し、そのピボットとリソース情報とを引数として、照会先自治区にアダプタを要求する。

⑦ 照会先自治区は、指示されたピボット（の型）へとリソースを写像できるアダプタを生成し、アダプタにピボットとリソースを記憶させ、当該アダプタを応答する。

⑧ フェデレーションは応答されたアダプタをピボットに記憶させ、当該ピボットを照会元自治区に応答する。

⑨ 照会元自治区は選択済のサロゲート型よりサロゲートを生成し、応答されたピボットを記憶させる。

⑩ 以上までで、サロゲート⇒ ピボット⇒ アダプタ ⇒ リソース へのチャネルが確立された。以降はクライアントのアクセスに応じ、要求の変換／フォワード[9]が当該チャネル上においてなされることになる。（図 3）

[8] インタフェース型も可。

[9] 一種のアプリケーションレベルでのマーシャリングだと捉えても構わないだろう。

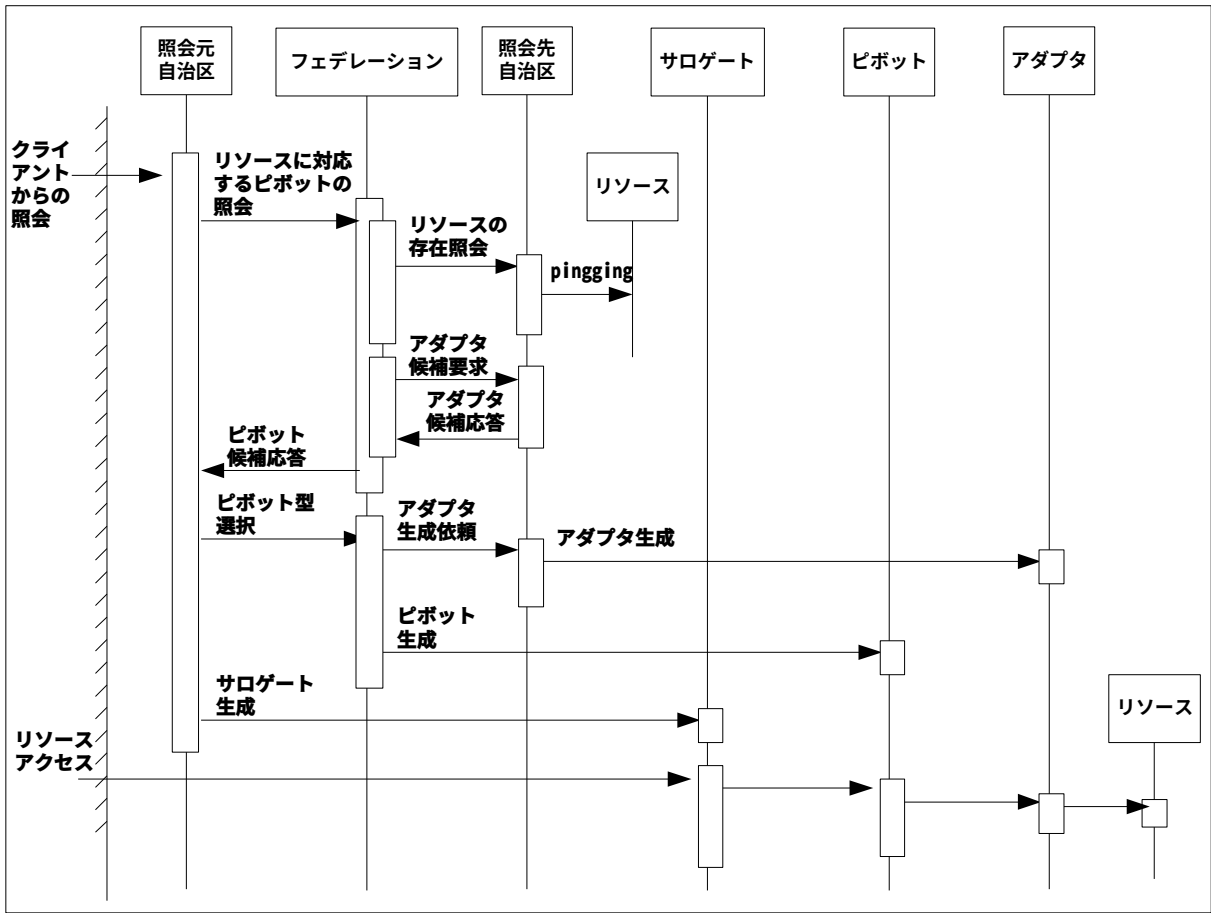

図 3 フェデレーションを介したリソースへのチャネルの確立

1.7. 実装

実装においては、アダプタ、ピボット、そしてサロゲートの適切な組み合わせによるリソースへと至るチャネル確立のために、フェデレーション、自治区とも各オブジェクトの型情報を用いる必要がある。開放性を有したシステムの実現のためには、このようにランタイムにおける型情報など、メタプロトコルに関する情報[10,11]が不可欠である。

実際、よりエレガントなフェデレーションの実装は、リフレクションやオープン インプリメンテーション、あるいは AOP などで試みられているアプローチが有効と思われる。

フェデレーションを比較的規模の大きなスコープで適用する際には、分散オブジェクトへの配慮が必須となる。現実的には分散環境の substrate layer としてフェデレーション機構が提供されることが望ましい。

---

[10] G. Kiczales, R. Deline, A. Lee, C. Maeda: Open Implementation — Analysis and Design of Substrate Software, Tutorial #21 of OOPSLA '95, 1995

[11] G. Kiczales, J. des Revieres, D. Bobrow: The Art of the Metaobject Protocol, MIT Press, 1991

フェデレーションにおいてマイクロアーキテクチャとして適用されるパターンに、プロクシ、あるいはコンバータパターンがある。また明示的な言及を行っていないが、本論で用いたピボットを支点とする m:n 変換系は、これもマイクロアーキテクチャとしてパターン化が可能と思われる。

1.8. 解法の事例

オブジェクト指向に基づくフェデレーションパターンの範疇に分類できる解法の事例の試みとして、OMG／OS の Trader Service における federated service が指摘できる。複数の ORB に跨って trading service が実行される際に、各 ORB が定義するポリシ[12]が優先されるようになる。

また federated database として DBMS の分野では幾つか研究が行われてきている。

1.9. 亜種

多重視点を考慮することで複数のモデルを同時的に満足するシステムを実現する機構として、Harrison らの Subject Oriented Programming、および Minsky らの Means of Surrogate などの試みがある 。

システムの柔軟性と開放性を確保するアプローチとして、reflection の研究は類似するものが少なくない。しかしリフレクションはシステム自身が変貌を遂げ、適応することに主要な焦点があり、変貌以前との互換性の保証には余り配慮がなされていない。フェデレーションはオンザフライなモデル（世界観）の導入と、それまでの世界観とそこでのリソースがそのまま有効であるという点で、前者の試みとは異なっていると云えよう。

1.10. 適用事例

実用レベルでは CORBAmed などでもフェデレーションサービスが意識されている。また米国 Vitria 社の EAI[13]パッケージである BusinessWare は、企業レベルにおいて既に導入されている各種の ERP パッケージを「そのまま」に、全社レベルのモデルに基づく統合を可能とするため、明示的にフェデレーション機構を導入している。

1.11. 帰結

省略。

---

[12] 例えばホップカウント上限値、応答待ち時間上限、などのプロパティが ORB 毎に動的にオーバライドされる。

[13] Enterprise Application Integration の略。

1.12. **参照**

省略。