仮想化を超えて:ドメイン世界のモデル化と言語化

羽生田栄一†

アブストラクト

クラウドコンピューティングや仮想化技術の進展により、個々の具体的なコンピュータは表面上は表に出てこなくてよくなった。とはいえ、メモリやディスク、OS やネットワークといったデバイスやシステムソフトウェアのレベルの仮想化にとどまっていたのでは、一般の情報システムの生産性向上や新たな価値創出には直接つながりづらい。今後の仮想化の概念的な意味と方向性を論じ、モデル化や言語化の展開が業務やアプリケーションレベルでの仮想化に当たることを示す。

Over the virtualization technologies: modeling and langaging domain world

Hanyuda Eiiti (Mamezou Co., Ltd.)

1. はじめに

現在、クラウドコンピューティングが発展し、さまざまなソフトウェアやサービスの多くがクラウド上で提供され、実行され、利用されている。その際の技術的なキーワードは仮想化であり、ユーザーにわからないように、メモリやディスクやCPU、OSやネットワーク等が仮想化され、クライアントや利用者に対してはもっとも適した環境が実装上の実態とは別に用意され、あたかもそれが元からそこにあったかのごとく利用できるのである。

しかしながら、仮想化の技術や考え方は、そうした現在のクラウドコンピューティングの実装を支えるためだけにあるのではなく、よくよく考えればもっと上流のシステムの設計や組織の定義、アプリケーションの記述、さらには業務モデルそのものにまで適用していくことができる、かなり汎用のコンセプトだということがわかる。

2. 実現技術/抽象としての仮想化

2.1. カプセル化としての仮想化

仮想化されたシステムは、そのクライアントに想定された適切なインターフェースを通して利用可能性を提供する。ここでのポイントは「インターフェース」である。これはカプセル化の一種とも、実現(実装)技術という情報の隠ぺいともいえる。あくまでもインターフェースだけを意識して、クライアントはそのシステムを利用すればよい。IaaS(Infrastructure as a Service)や PaaS(Platform as a Service)のようなクラウドサービスは、実行ハードウェアや OS、ミドルウェア、言語処理系、フレームワークといったソフトウェア環境のカプセル化を行い、利用言語や共通 API といったソフトウェア開発・実行のための汎用インターフェースを提供している。

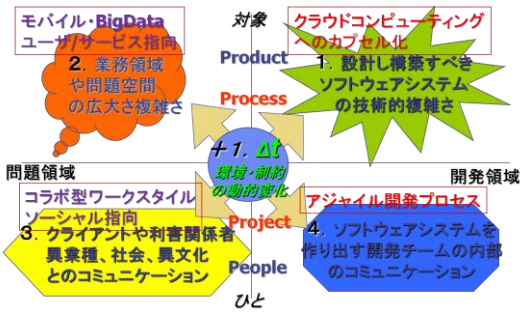

図 1. ソフトウェアシステムの複雑さの 4 象限

2.2. 抽象化としての仮想化

仮想化技術は単に実現方法(実装)とは異なるインターフェースを提供するだけでなく、そのインターフェースを通して既存の技術をより高い概念レベルで戦略的に使いこなすことを可能にする。いわゆる抽象データ型がそれに相当するが、そもそもそのようなインターフェースのセットが必要であるのは、その業務の背後に同一のアナロジーで説明できる基本的に共通の基本構造が存在するからだというわけである。アナリシスパターンや Composite のようなデザインパターンに伴うイ

†株式会社豆蔵

ンターフェースはそのような方向性を志向している。

3. 仮想化は分野を横断する

3.1. ビジネス系と組み込み系

ソフトウェアの対象領域として大きくビジネス系と組み込み系の2つが存在する。

3.2. 組織モデルと製品モデル

ビジネス系では、組織とそこでのビジネスプロセスやビジネスルールをいかに表現し柔軟に変更に対応し実行するかが問題となる。これは組織モデルであり、エンタープライズアーキテクチャのモデル化だといえる。

組み込み系では、エレキ・メカを含むハードとソフトウェアを合わせた製品モデルをシリーズやバージョンアップという形でいかに設計・生産管理するかが問題となる。これはいわゆるプロダクトラインエンジニアリングのモデル化であり、PLE アーキテクチャを考える必要がある。

3.3. モデリングと仮想化

いずれにせよ、なんらかのモデリング技術を用いた共通プラットフォーム化が必要であり、ビジネス系の場合は SOA アーキテクチャ、組み込み系の場合はリアルタイム性も視野に入れた MARTE(Modeling and Analysis of Real-Time and Embedded Systems)といったプロファイルが想定され、UML/BPML や SysML といったモデリング言語を前提に実現技術の仮想化が図られることになるものと思われる。

4. 言語やモデルと仮想化

4.1. モデリングとシミュレーション実行

ビジネスプロセスを UML や BPMN でモデル化しそれをワークフローエンジンや Web アプリケーションとして実現することは容易である。また実装前に実行条件を与えてモデリングツール上でシミュレーション実行することも可能である。同様に、組み込み系でも SysML や Matlab/Simulink などを用いて対象システムのシミュレーション実行が行え、場合によっては MDA（Model Driven Architecture）や MDD（Model Driven Development）といわれるような、最終コードの自動生成まで提供する開発環境も存在する。

4.2. 仮想化の手段としての言語処理系

Java 言語は JVM という仮想マシンを通して実現環境を隠ぺいし、アプリケーション開発者に一種の共通 API をアプリケーション言語という形で提供している。JavaScript も Web ブラウザ環境やクライアントプラットフォームを隠ぺいし言語という形で抽象化された共通 API を Web アプリケーション開発者に提供している。

4.3. ドメイン特化言語 DSL

特定の業務・問題ドメインに特化した言語をそのドメイン利用者やアプリ開発者に提供する DSL(Domain Specific Language)はその業務環境の仮想化と言える。

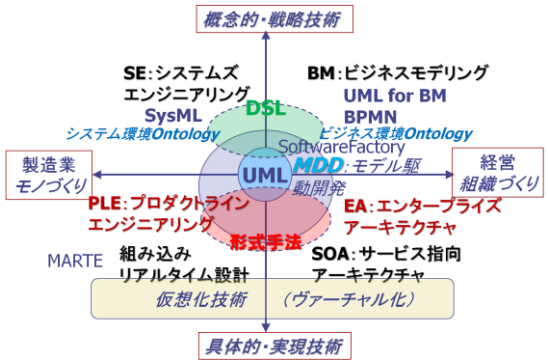

図 2. モデリングと仮想化の可能性

5. 考察

5.1. 仮想化の定義

さまざまなレベル・次元・抽象度・分野での仮想化が存在するため、今後「仮想化」そのものの定義をより明確にしていく努力が必要だと思われる。

5.2. オントロジーの重要性

DSL が有効に機能するためには、その問題ドメインの重要概念が無定義語および定義語として明確に位置づけられ DSL を通して参照・関連付けできる必要があり、ドメインオントロジーの重要性が増す。

5.3. 今後の展望

共通アプリケーション言語を超えて、ドメイン毎の DSL の普及が今後加速するものと予想できる。DSL の標準的な開発手法、利用手法の研究が今後ますます重要になるであろう。

参考文献

[1] 日経 BP 社, すべてわかる仮想化大全 2012 (日経 BP ムック),日経 BP 社(2012)

[2] Martin Fowler, ドメイン特化言語 パターンで学ぶ DSL ベストプラクティス 46,ピアソン桐原(2012)

[3] Debasish Ghosh, 実践プログラミング DSL ドメイン特化言語の設計と実装ノウハウ,翔泳社(2012)

[4] David S.Frankel, MDA モデル駆動アーキテクチャ, SIB アクセス(2003)

[5] サンフォード フリーデンタール, システムズモデリング言語 SysML, 東京電機大学出版局(2012)