新しいシステム開発パラダイム

藤井　克磨

株式会社 ブイシンク　〒150-0001　東京都渋谷区神宮前６丁目10番11号
Tel. 03-5468-0121　Fax. 03-5468-0125　E-mail: fujii@v-sync.co.jp

本講演は新しいシステム開発パラダイムについて提案する。ゲーム等のコンテンツ系のシステム（ハードウェア、ソフトウェア、アーキテクチャー、ビジネスモデル）開発は、おもしろいかおもしろくないかという曖昧な判断基準でその存在価値が判断される。このようなコンテンツ系のシステム開発においては、要件定義をもとに開発を進めて行くことは難しい。コンテンツ系のシステム開発においては、発想を変えてコンセプト発想をベースにいかにエンジニアリングしていくかというアプローチが大事になる。このアプローチにおいてはステートトランジションダイアグラム、プロトタイピング、オプティマイズ、ルールの策定が重要な役割を果たす。現状を見ると、コンテンツ系のソフトウェア開発環境においても、いくつかの企業により寡占的な支配を受けている状態であるが、一方、最近ではオープンソースソフトウェアという既存の枠組みに対抗し得る新しい文化も着実に育ちつつある。この新しい文化とプログラマに対する著作権による保護という考え方を取り入れることにより、コンセプト発想をベースにしたソフトウェア開発をより早く、より妥当なコストでしかも品質の高いものとして実現できる。

A New Development Methodology

This key note speech shows a new paradigm for system development. Here, let me restrict the system to Content driven system (hardware, software, entire architecture and business models), such as game programs, is evaluated based upon very ambiguous criteria, i.e. amusing or boring.

For this type of system, straight forward methodology along with demand definitions could not guarantee successful results. Apart from traditional approach, concept driven engineering has provided number of successes in this business field. State Transition Diagram, Prototyping, Optimization and Rule Definition play key roles in this approach.

Open Source Software trend is emerging as new counter culture movement against oligopoly in the field of software development environment for contents driven system.

Combined with this movement and the protection mechanism of intellectual property right of software engineers will make it real, a new paradigm that can provide concept driven software development within shorter period, with reasonable costs and with excellent quality.

1．はじめに

情報システムコストの低減により、システム部門等に所属する専門スキルを有した人員ではなく、エンドユーザが直接使用することを想定したイントラネット、情報端末等のシステムが開発されることが多くなった。しかしながら、ユーザインターフェース、操作性等に問題があり放置され使われなくなったシステムが多いのではないか。このようなシステムにおいては、何ができるかという発想よりも、いかに使ってもらうかという発想が重要であるが、そのような観点からシステムを定量的に分析する手法はない。我々は、音楽産業、ハリウッド映画産業、そして日本から唯一輸出超過しているゲーム産業にその解決手段を見いだすことができると考える。

我々は、従来、ハードウェア、ソフトウェアで閉じていたアーキテクチャモデルに何をやりたいかという観点から拡張を加え、本論でいうシステムの参照モデルを UCA（Unified Communication Architecture）参照モデルとして下記のように定義している。

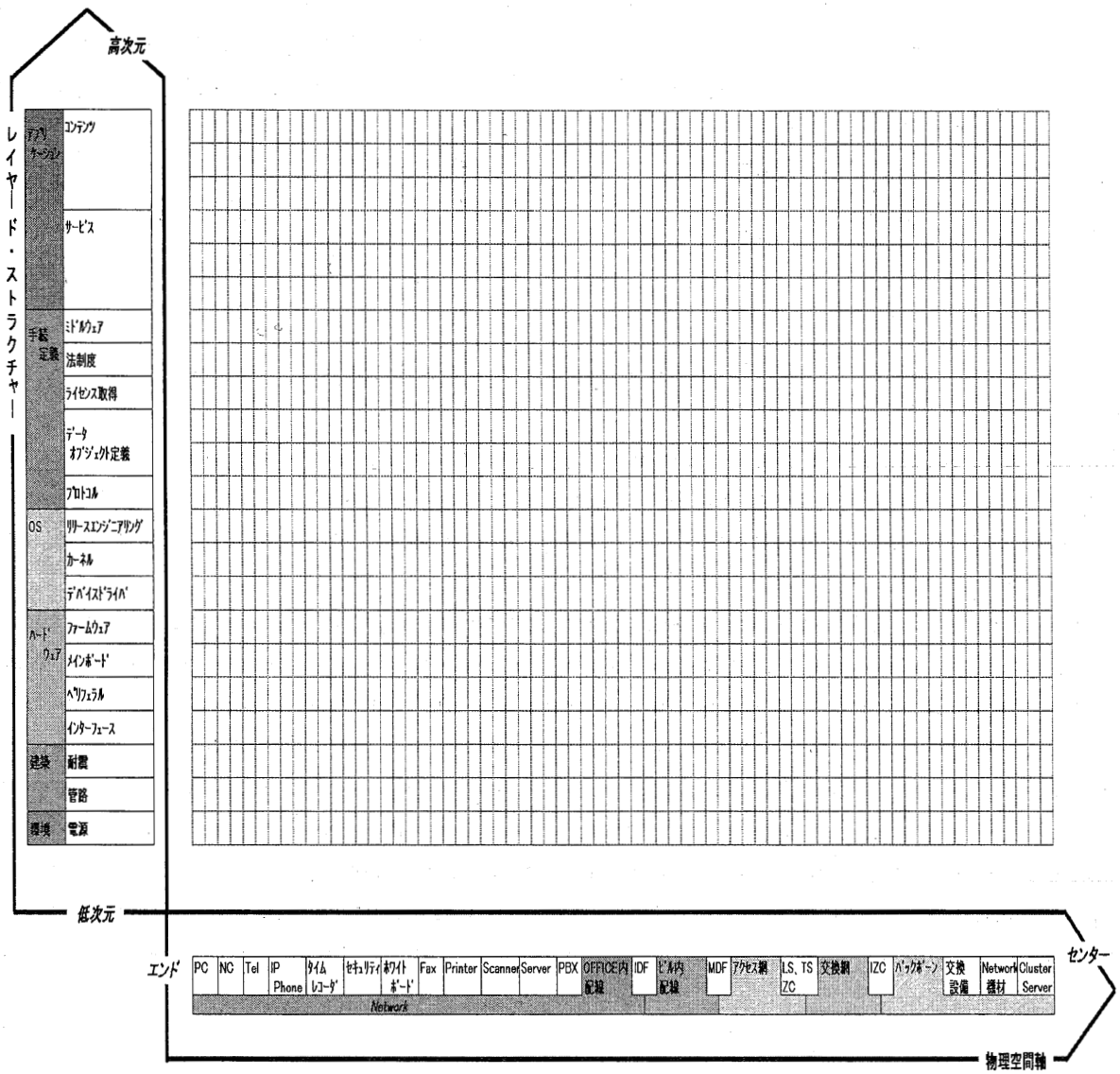

図１：UCA（Unified Communication Architecture）参照モデル

２．コンセプト発想型ソフトウェア開発と要件定義

システムを開発する上で、開発方法論の有用性は多くの人が認識するところである。多くの方法論は、現実の世界の事象を情報としてとらえ、情報をモデル化しいろいろな分析を行い要件定義に結びつけるというアプローチをとっている。この様な方法論は、現実の世界が存在する事象に対するシステムに対しては有効である。すなわち、シーズ、ニーズが現実に存在し、それらに対するアプローチとして新しいハードウェア／ソフトウェアやソリューションに裏付けされた方法論を適用することにより要件定義→問題解決という構図が成立する。ビジネスアプリケーションの分野等は典型的で、例えば、C/S、ネットワーク等のキーテクノロジーをベースにERP、SCM 等のソリューションを用い、オブジェクト指向型の方法論を適用するという明示的なトレンドに従い実装者が実装することが可能である。

一方で例えば、「パソコンを買って何かをやりたい」という人がいたとしよう。このような漠然とした何となく「こんなことしたいな」という世界に対しては、方法論に基づいた要件定義→問題解決というアプローチは極めて困難である。上記のような場合、実際には以下の様な QA を繰り返すことになろう。

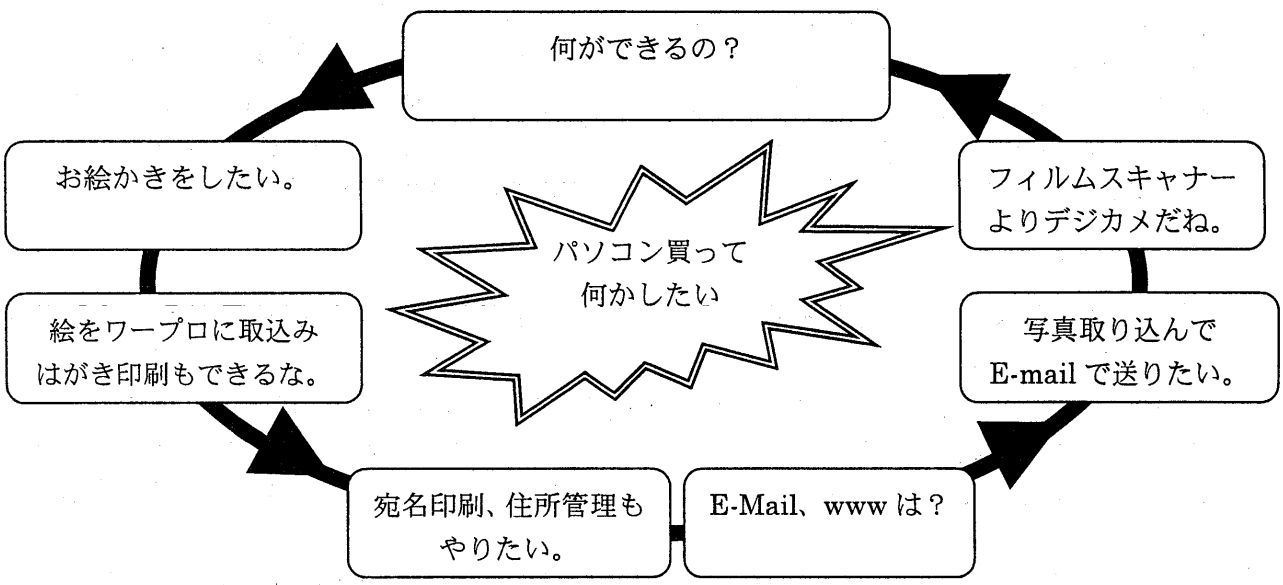

図２

注意して欲しいのは、上記の作業により何か問題が解決されているわけではなく、「どんなことをしたいのか」というテーマに対して同心円的に試行錯誤を繰り返しながら、新たな需要を創出しているということである。（「パソコンを買って何かをやりたい」という人は相変わらず問題解決の為に「何かをやる」というのではなく、「こんなことができるのか」という理解が得られたのみである。）

何となく「こんなことしたいな」というようなタイプのシステムは、コンテンツ系のシステムに多く見られ、今までになかった新しいことをやるコンセプトドリブンのシステムである。このようなコンセプト発想型のソフトウェア開発プロジェクトにおいて顕著な特徴としては、以下のようなものが挙げられる。

・新しいメンバが加わる度に全員が同じ議論をもう一度最初から行う。
・プロジェクトのメンバーにより、そのプロジェクトに対するコンセンサス、熱意に温度差がある。
・同一の問題について同じ議論を繰り返しながら、問題解決のレベルをアップしていく。(あたかも、螺旋階段を上って行くが如く)
・成功事例を分析して成功したもっともらしい理由を後付で述べることはできるが、企画時点では成功するか否かは当事者にも不明であった。(成功事例と同一アプローチをとってもほぼ成功しない)

これらの特徴は、明確なシーズ、ニーズがなく、そのプロジェクトの必然性を説明する方法論がないことに起因するものと考えられる。

コンセプト発想型のシステムに類するものとしては、タマゴッチ、プリクラ等のアイディア商品と呼ばれるものがある。これらの成功要因は、「ひらめき」のひらめきの一言で説明されてしまう。京都大学 大島 清 名誉教授によると「ひらめきとは、人間が突然または一瞬のうちに言葉や理屈ではなく、視覚的なイメージによって浮かび上がる事柄である」と定義付け、左脳の前頭前野の活性化に起因するドーパミンによる右脳の活性化により説明している。しかし、我々実装者レベルでは脳科学ではなく、何らかの方法論に基づき「ひらめき」に相当するものを実現し、コンセプト発想型のシステム開発を一定の打率で成功させる必要がある。

3．ステートマシンに対する設計手法ステートトランジションダイアグラムの適用

我々は先ず、コンセプト発想型のシステムが「ひらめき」ではなく、一定の妥当性をもって「ひらめいた」ことを説明し、かつ、「ひらめく」べき領域を合理的に探し出す為に、ステートマシンに対する設計手法であるステートトランジションダイアグラムをシステム開発の分析手法として用いた。

ステートトランジションダイアグラムでは、状態で物事の現象を網羅的に表現する。例えば、人間に対しては、簡略化すれば例えば以下の状態で表現が可能である。

・生きている状態 or 死んでいる状態
・生きている状態であれば、家庭に居る or 通勤途上 or 会社に居る
・上記のそれぞれの状態に対して立っている or 座っている or 寝ている

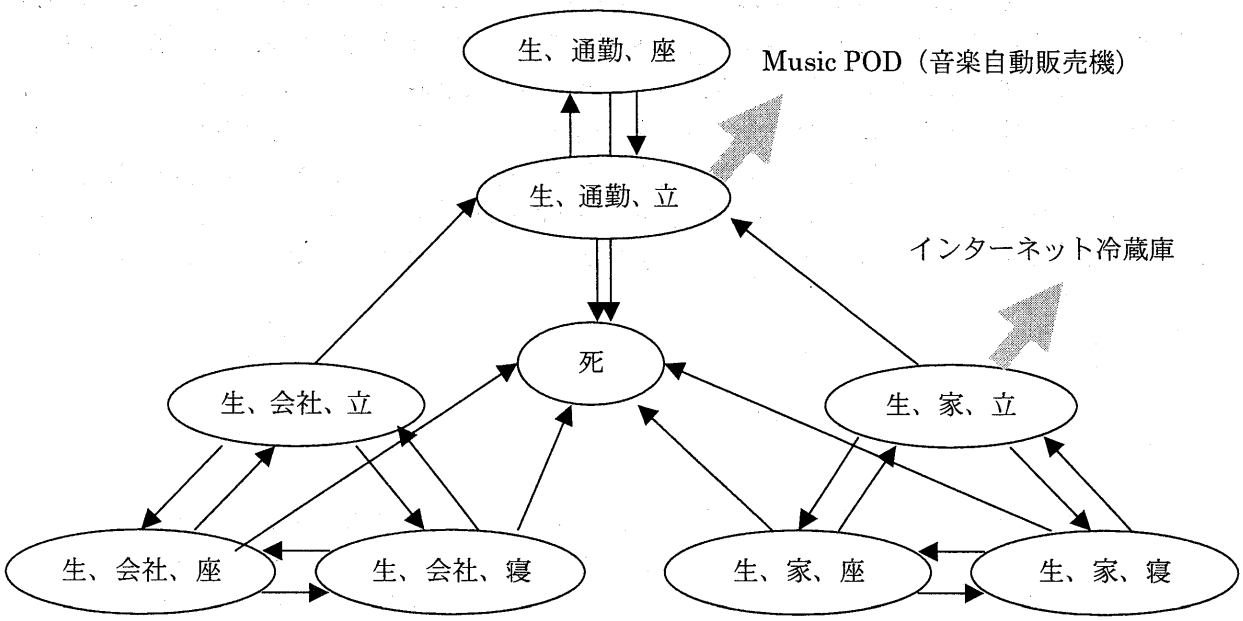

図３ステートトランジションダイアグラム（一例）

我々の製品の一つにインターネット冷蔵庫（図４）がある。これは、人間の家庭に居る状態でかつ立っている状態に着目してコンセプト発想したものである。現在、PC、インターネットに関する実装は家庭に居る状態もしくは会社に居る状態でかつ座っている状態に対する実装（デスクトップPC)、通勤途上で立っている状態に対する実装（モバイル）に対し各メーカー、ベンダー等が過当に参入している状況である。そこで、我々は未だ実装が盛んでないに家庭に居る状態でかつ立っている状態に着目し、コンセプト発想を行いインターネット冷蔵庫を企画・作成した。

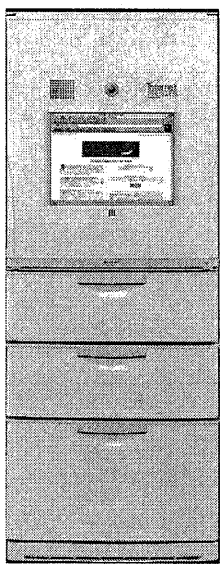

図４：インターネット冷蔵庫

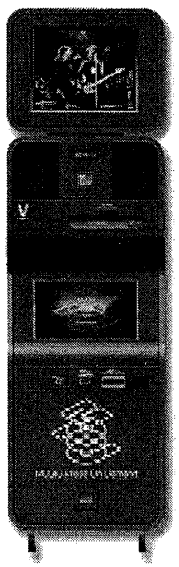

図５：Music POD（音楽自動販売機）

又、我々の製品のもう一つにMusic POD（音楽自動販売機）（図５）がある。これは、人間の通勤途上に居る状態でかつ立っている状態に着目してコンセプト発想したものである。（モバイルは正確には通勤途上に居る状態でかつ歩いている状態であり異なる）例えば、EC に関して言えば世間では先ず企業間 EC でその次は当然家庭内 EC という流れになっているが、ステートトランジションダイアグラムに基づけば通勤途上に居る状態でかつ立っている状態に着目した街頭端末という発想は必然であり、また、現実の世界（街頭での販売）やネットワークトポロジー（NOC（ネットワークオペレーションセンター）→街頭経由→企業、家庭）を参照しても、街頭の場を無視する理由はどこにもない。

４．プロトタイピング、オプティマイズ

ステートトランジションダイアグラムによる分析手法に基づき、「ひらめく」べき領域を合理的に探し出しコンセプト発想を行った次に行うことはプロトタイピングである。先に述べたようにコンセプト発想型システムの場合、企画時点で成功するか否かは不明である。又、コンセプト発想型システムは何となく「こんなことしたいな」という発想であるため、企画だけでは多くの人が「自分も似たようなアイディアを持っていた」と主張する場合が多い。そこで大事なのは、企画を十分に詰め成功の可能性について十分に検討した後に製作する（ヒット、バント、ヒットで１点というパターン）のではなく、とりあえず片っ端から製作してしまう（ホームラン狙いで３打席に１回は１点というパターン）という「プロトタイピング」である。いち早く発想したものをプロトタイピングしてしまうことにより、唯一現存する物をベースに製品、サービス、ビジネスモデルの良し悪しを判断することが可能になり、また、当該領域のルール策定のリーダーシップを取ることが可能になる。ただし、片っ端から作成するといっても、闇雲に作成するのではなく、実現不可能な事（星に行く事）なのか目の前にある困難を解決すれば実現可能な事（スカートめくり）なのかについては十分検討する必要がある。例えば、Music POD の場合では音楽 CD のビットレート×世の中にある音楽 CD の枚数を計算し最大バンド幅を算出し、現状のシステム技術によって電送可能であり、物理法則を越えていないことを事前に確認した。

幸いにも我々は実装力の技術・スピードに強みがあり、リアルタイムでの検証により、インターネット冷蔵庫及びMusic POD については、立った状態でのレスポンスタイムの重要性を識別し、オプティマイズを行った。

５．ルールの策定

ステートトランジションダイアグラムによる分析手法を用いるもう一つのメリットに、ルールの違いを認識し、かつ、ルールが策定されていない領域を認識できるということがある。前述の例で言えば、立っている状態でのルールと座っている状態のルールは異なり（ユーザーインターフェース、許容レスポンスタイム等）、家庭に居る状態で立っている状態のルールの策定は不十分であることがわかる。ルールは守ることが大事である反面、一方でルールを策定している人がいることも忘れてはならない。F-1 グランプリ、スキージャンプ等に見られるようにルールの変更により、結果として特定のプレーヤーに有利になることがあり、ルールの策定、変更が大変重要

であることが判る。しかしながら、闇雲にルールを策定、変更してもそれが受け入れられる可能性は大変低い。ルールが受け入れられるためには、コンセプト発想により新しいものを提案しつつ、それと同時にルールを策定・提示していくことが最低限必要である。

6．オープンソースソフトウェア

ルールの策定は大事であるが、コンピュータ業界においては、現在、数社による寡占状態が続いており、特定のSDK、アーキテクチャーに依存していることによる制約が大きく、現状のままでは、例えコンセプト発想による新しいものを提案しても、ルールの策定・提示に至るまでには、多くの障害が待ち受けていることは容易に想像できる。しかし、近年、オープンソースソフトウェア（OSS）によるシステム開発が新しい動きとして台頭しており、これにより、既存の制約を打破できる可能性は大変大きくなった。インターネット上で公開されている（http://www.post1.com/home/hiyori13/freeware/halloween.html）いわゆる、「Halloween 文書」と呼ばれる文書の中で、オープンソースソフトウェアの最右翼であるLinuxを以下のように分析している。

> GPL（General Public License）であるLinuxは、OSSにとって非常なステップであり、提供されたソースコードから開発されたすべての資産を、GPLコードに還元するように求めている。この開発スタイルによって、世界中のプログラマたちが共同で一つのソフトウェアを開発しているようなスタイルになり、結果としてそのソフトは商用ソフトより高品質なものになってしまうのである。非常に複雑で大規模なソフトの開発も、Linuxの場合、x86のカーネル部分だけでも50万行のコードからなり、すべての配布ソフトを加えると、1000万行に達する。インターネットを利用したOSSの開発に対抗するには「企業ではなく（開発）プロセスに対抗しなければならない。（一部抜粋、要約）

このようにOSSの世界においては、全てのソフトウェア開発者がルール策定、変更に参画できるのである。

7．プログラマに対する著作権による保護

我々は、Music POD（音楽自動販売機）を製作するにあたり、楽曲1曲販売毎に著作権処理を行い、正当な権利者に対して適切な課金処理を実現している。同様の考え方をOSS及びOSSを支えるプログラマに適用できると考える。すなわち、ある特定のプログラマが作成したモジュールには著作権があるという考えに基づき、当該モジュールが1回起動する度毎に利用者に対し課金処理を行い、プログラマに対して支払いが行われるような仕組みが実装可能であると考える。これにより、1ステップあたりいくら、プログラマ1人月いくらといった従来のソフトウェア課金体系ではなく、プログラムの使用に基づくより合理的な課金とOSSを支えるプログラマへの還元を行う。

8．将来のプログラマ像

将来的にプログラマは、ミュージシャンのように個人がフリーで活動し、システムプロダクションに所属し、レコード会社がレコードを販売しているようにソフトウェアハウスからソフトウェアを販売するようになるのが理想であると考える。

謝辞：講演の準備に当たって、手助けをして頂いた株式会社 ブイシンクの永島 道夫 氏並びに
福谷 大 氏に感謝いたします。