Lyee　Internet　Information

岩手県立大学　藤田 ハミド 教授　インタビュー

藤田 ハミド 氏　略歴

１９５５年　カナダ　ケベック州生まれ　現在４５歳
１９７９年　英国　マンチェスター大学　電気通信学部卒
１９８２年　日本に来日し、東北大学、東京大学にて学ぶ。
１９８４年　東北大学　情報工学科で博士号取得
１９８９年　東大先端科学技術研究センターで助教授としてオブジェクト指向を研究
１９９１年　９３年までカナダ　モントリオール大学情報工学科で助教授
１９９４年　再び来日し、山形東北芸術工科大学の教授に就任
１９９８年　岩手県立大学に着任し、現在ソフトウェア情報学部の教授として教鞭をとるかたわら同メディアセンター情報システム部長を勤める。

＜インタヴュー本文＞

－Ｑ　：　ハミド先生のこれまでの主な研究分野について教えて下さい。

－Ａ　：　私は主にソフトウェア工学を研究してきました。取分けイヴォーブソフトウェア（進化するソフトウェア）といわれる分野について長年研究を重ねて参りました。

簡単に言えば人工知能やオブジェクト指向、さらにオブジェクト指向に人工知能のアプローチを追加してもっと賢いシステムとして問題解決したり、賢い仕様書を作成するといったたぐいの“知的システム”の研究をしてきました。

例えば“反射言語”（リフレクティブ・ランゲージ）のようなものです。反射言

語とはあるシステムを変更する場合、つまり新たな要件を追加したり変更する場合、その部分だけでなく関連する若しくは影響されるモジュールを調査してその部分も変更しなければならない。
そのような追加・変更をする場合、そのシステムの完全性を把握し、影響度合いが多い場合は変更しない方が良いというアドバイスをするようなツールです。

－Q　：　もう少し平易に説明して下さい。

－A　：　開発支援ツールの事です。例えばプロトコル仕様書・システム仕様書などを生成するツールです。ＲＭＯＤＥＬ(Reflect Modeling Language)などに代表されるもので、オブジェクト指向で開発する場合に使用します。
特徴としては、ユーザーがある部分を変更したいと思った時に、変更した部分が、他にどの様な影響を与えるか、どの様に変化するかをある程度自動的に反映する様なツールです。

－Q　：　Lyee に巡り合う経緯について教えて下さい。

－A　：　Lyee と巡り合ったのは、１９９９年９月にロシアで行われた PACT９９（並列コンピューター工学学会）で根来さん（＝ソフトウェア生産技術研究所　社長）の講演を聞きました。続いて１０月に安比高原で行われた国際会議（戦略的知識と概念形成学会）に参加し、ディスカッションを致しました。

－Q　：　どんな感想をお持ちになりましたか？

－A　：　Lyee の理論自体は理解出来なかったのですが、研究者として“ これは何かあるな ”というか“ 面白い ”という感覚を持ちました。
根来さんが提唱する、Lyee のベースとなる、“ ユーザーの意図をそのままプログラム化する ”という事に私も大賛成だったからです。本来、ユーザーが欲しいものと、実際に出来上がる仕様書・プログラムを一致させることは非常に難しいし、実際ユーザーの要求と合致しているかどうか解りません。
各々の開発ステップにおいても作業自体は複雑で大変な労力がかかります。
Lyee はそうしたステップが一切なくなるという。その発想にとても興味を惹かれました。どこの学会でも聞いたことがないメソッドでしたし、それが本当に出来るのか出来ないのか？そこに興味がありました。

－Q　：　疑問から拒否反応を起こしませんでしたか？

－A　：　疑問と言うのは後回しです。どういう風にその理論が実現するのか？その辺を質問・ディスカッションしながら、それは正しいのか正しくないのか？本当か嘘か？見極めていく事で建設的な理解が進みます。

－Q　：　Lyee のどの部分が面白いと感じたのでしょうか？

－A　：　根来さんは「テストは必要ない」と説明されました。ソフトウェア工学上はテストは必要なものであり基本です。それなのに“ 必要ない ”とはどうしてだろうか？と。
また、今はもう変わりましたが、当時の根来氏の説明の仕方がソフトウェア工

学の学者達が期待するものではなく、彼のフィロソフィーであったり物理学的な説明でした。我々の常識とは違った言葉で説明していたのです。
それでは我々には説得力がありません。
説明するためにはユーザーレベルに合わせる必要があると思いました。
もちろん Lyee だけでなく説明不足なものは他にもいくらでもあります。
そんな経験はいくらでもありますので、解らないから拒否するのではなく、何か面白そうだから研究してみる、そういうスタンスでいます。
そんな折、ソフ技社も学会に論文を提出したいというアプローチがあり、私がその部分を手助けしながら、一方 Lyee の勉強するというようなギブアンドテイクで、月１～２回研究レベルのディスカッションしてきたわけです。

－Q　：　何か成果はありましたか？

－A　：　もちろんありました。IS2000 という学会（21 世紀における情報社会学会＝2000 INTERNATIONAL CONFERENCE ON INFORMATION SOCIETY IN THE 21ST CENTURY）に根来さんが作成された Lyee に関する論文が採用されました。11 月に金沢で開催される国際学会において発表されます。

－Q　：　どんな意味付けがありますか？

－A　：　とても意味のあることですよ。国際学会というものは様々なレフェリー、学者が３ヶ月くらいじっくり論文を読んで、オリジナリティーとかセールスポイントを評価しますから。オリンピックで言えば金メダルを取るようなものですよ。論文を提出したとしても、大雑把に言えば３０％程度しか採用されないとてもハードルの高いものです。
それと Lyee がソフトウェア工学の学者達に受け入れて貰えるような形で論文にまとめられたという事です。

－Q　：　では採用されたことで何らかのリアクションがありますか？

－A　：　リファー出来ます。つまり今後は大学や図書館などに学会誌が配布され、その論文を誰がいつどこにいても見ることが出来るのです。
今まではパテントのみだった訳ですが、今後はアカデミーにおいてもこのような論文があるというお墨付きを得るわけです。Lyee の存在がアカデミーにおいても認められることになります。

－Q　：　アカデミーということでは他に何かトピックはありますか？

－A　：　今 International Conference on Software Engineering（ICSE/ソフトウエア工学国際会議）＝この分野における世界トップクラスの学会ですが、ここにも論文を提出しました。これに採用されると益々今後の展開が期待できます。
あと６月にソフ技社とカナダを周ってきたのですが（オタワ大学他＝LyeeInternetInformationVOL７参照）かなり関心を惹くことに成功しました。
彼らは日本人と違ってお愛想を言いません。ハッキリしていますからね、外国人は。何もなければ 2 度と付き合ってくれません。つまり Lyee が出鱈目なもの

であったり、退屈などこにでもある理論であるならばそういう機会は 2 度と持ってくれないのです。その彼らが「とても面白かったよ。ただ何か足りない。もっと詳しく聞きたい」と言います。

そこで再度１０月にカナダに行くことにしました。モントリオール大学、ケベック大学、デュモリアール大学などを周り、Lyee についてディスカッションする約束を取り付けました。

－Q　：　Lyee の世界的研究ネットワークを組織されるそうですが、その概要、進め方など教えて下さい。

－A　：　米国、カナダ、日本、スウェーデン、イタリア、デンマーク、オーストラリア、イギリスなど１０カ国の研究者、教授３５人程度が集い、それぞれにテーマを決めて Lyee を研究し、他の方法と比較し拡張していく、そんなイメージです。当然彼らの下で学ぶ学生達も加わる予定です。

半年に１回くらいの割合で各々テーマを発表し、オープンディスカッションするようなものも検討しています。ワークショップ的なものを想定しています。

－Q　：　スタート時期はいつ頃ですか？

準備段階としては既にスタートしました。２００１年3月末までに調査を行い各先生の役割分担と詳細スケジュールを決定します。その時点で中間報告という形でセッションを行います。

その後つまり 4 月以降本格的な活動開始という予定にしています。計画の実行という事になりますね。

－Q　：　研究とはどういった形で行われるのですか？

－A　：　Lyee 本体を研究すること、例えばシナリオ関数についてもっと深く掘り下げるという事もあるかと思います。

また Lyee を用いて何が出来るのか？Lyee の可能性の模索という事もあります。例えば交換機のシステムに適用出来るか？とか飛行機のオートパイロットシステムに適用出来るのか？ＯＳに適用した場合はどうかとか、ＯＳとして成功するか否か？、オブジェクト指向との比較といったことなどいろんな事が考えられます。そういったことも含めて各国の学者達と役割分担をしながら研究したいと思っています。そうすることで Lyee の拡張性を模索し、高め強くしていきたいと思います。来年４月からそうした本格プロジェクトとして動き出す予定です。２００２年には京都で国際会議を開催するという計画もあります。

－Q　：　では技術的なお話を伺います。Lyee に関してユーザーはその実態をなかなかつかめないようです。例えば「ＤＯＡと何が違うの？」とか「オブジェクト指向と一緒じゃないか？」とか。そのあたりどの様にお感じになられますか？

－A　：　日本は何か新しいものが出ると、何かと比較できないと理解出来ないような風潮があるようですね。（笑）確かにアプローチするには難しいものがあります。例えばオブジェクト指向の一種ではないのか？安比高原の学会でもさまざまな

先生が同じ事を言いました。
“ シナリオ関数 ” はオブジェクト指向で言うところの “ スペシャライゼーション ” と似ています。カバレッジ（適用範囲）を見るとなるほどその通りです。アウトカム（結果）としてはそうなのです。
けれども実際は違います。オブジェクト指向が作り出すソースプログラムはディテールを作ることが出来てないんですよ。一方シナリオ関数の作り出すアウトプット＝つまりソースプログラムはきれいに出来ます。
もう一つシナリオ関数の特徴は、部分集合のようなものを使います。けれどもオブジェクト指向のスペシャライゼーションは抽象的なものしか見ていないのです。ディテールがないのです。ただ抽象的なアプローチ、当然ですよ、ジェネラライズコンセプトにとっては “ アメリカ、アメリカ、アメリカ！ ” アメリカはただのフレームメーカーですから。
抽象的な事は神様に聞け！ですから。（笑）
Lyee はもっと具体的です。しかしながらオブジェクト指向の概要しかわかってない人は Lyee と似ていると感じるかも知れません。

－Q　：　ではもう少しオブジェクト指向との違いについて詳しく説明して下さい。

－A　：　オブジェクト指向で UML を使ってソフトを開発する場合、UML のユースケース図というものを使います。ユーザーとシステムの間でどういうやり取りをするかをユースケース図で定義します。そこには９つのダイヤグラムがあって、そこで要求仕様を決めていく。
Lyee の場合要求仕様はユーザーが決定することです。そこが全然違います。
オブジェクト指向は必ずリクワイアメントエンジニアがユーザーの様々な要求を聞きながら要求仕様を作るんです。UML にもひとつの方法論があります。OMT（オブジェクトモデリングテクニック）というメソドロジーです。
ご承知のように、要求仕様を記述する際はユーザーの要求とマッチしているかわかりません。何故か？ユーザーモデルがないからです。そこで、要求仕様はリクワイアメントエンジニアを使って、今度はシステムの中に、この要求仕様に基づき、要求仕様に基づくですよ！ユーザーモデルがないのに、ないものに基づくというのは矛盾でしょう！！
Lyee はこのプロセスはありませんが、OMT は様々なダイヤグラム、例えば状態遷移ダイヤグラム、データフローダイヤグラムとかを使ってシステム設計をする。それが出来上がったらシステム仕様書を作成する。そしてテストをし、ドキュメンテーションする。また、Lyee と同じように、UML にも自動化ツールはあるのですよ。
ただそこではオブジェクトとオブジェクトの間で交換するメッセージが解らない。そこでそれはプログラマーがやらなければならない。プログラマーはオブジェクト同士で交換するメッセージ、つまり他のオブジェクトにどういうデー

タを渡し、取るかなどはプログラマーが決める。
ところがプログラマーは要求モデルを知らない。ですから出来上がったインプリメンテイションに要求がマッチしているかどうかはわからないのです。
そうしたプロセスを見ると Lyee とは全く違うものだという事がわかります。
Lyee の思想はエンジニアではなく、ユーザーをとても尊重している。オブジェクト指向はエンジニアが尊重されている。なぜならば要求仕様はエンジニアが決定してるのだから。このギャップが大きく違います。

－Q　：　テストが必要ないという部分は今どう感じていますか？

－A　：　テストは今でも必要だと思っています。ただ根来さんの言う“テスト”と従来法で言う“テスト”には違いがあります。Lyee の構造から見た時には確かに従来法で行っていた様なテストは必要ありません。
Lyee を使ってソフトウェアを設計した場合、ユーザーの要求、つまり言葉、単語を LyeeALL を使って定義すれば良い。そしてシナリオ関数の仕組みは、W04、W02、W03 のパレット（プログラム）が順番に動作し、W03 のパレットはユーザーが開発要望として表明したインテンション（意思）の妥当性を定義の段階でテスティングしているのです。どの様にか？シナリオ関数では W04、W02、W03 という順番に動作させ、また繰り返して W04 に戻るメカニズムになっています。この繰り返しを行う事で、自分が開発要望として表明したインテンションをユーザーは無意識のうちに確認したことになります。W03＝自分のインテンションに一致するまで、このアイトレーションと出来たモデルを確認している。つまりバーチャル的にテストしている。従来法のテスティングとは考え方が違います。
テストとはどういうものか？（従来法の）仕様書にユーザーから様々な要求を聞いて反映する。でＳＥがユーザーの「何をしたい」「どんなシステムにしたい」という事を吸い上げて、要求仕様にする。
そこで理解している部分。Lyee はこの辺はアイトレートしている。従来法はアイトレートしない。ここは聞いた後“終わり”です。出来上がったインプリメンテイションが理解出来てないから、プロトタイプを作ったとしても不可能なのです。プロトタイプが出来上がってみると「違うぞ」「どこが違う？」それはアイトレート的に Lyee はシナリオ関数がやっている。ユーザーの要求を聞いて、何とかを作って、その何とかを今度コーディングする。要求仕様とシステムを踏まえてシステム仕様書を作る。出来上がった場合、そこにチェッキングが必要なのです。システム仕様書を確認する。何を確認するのか？それはロジカルな問題、ロジカルチェックをする。ロジカル的に何が問題なのか？それを行っている。
Lyee はこのステップをエンジニアではなく、シナリオ関数というメカニズムが行っている。シナリオ関数は W04、W02、W03。そのインテンションはユーザ

ーの言ったものになっている。

－Q　：　とすると従来法の方がテスト負荷は高いとお考えになりますか？

－A　：　もちろんですよ。従来法のテスティングはとても大変です。
どうしてか？システムの要求仕様とは、ユーザーから上がって聞いた話しに基づいて、何かあるものが出来る。このリクワイアメント＝システム仕様書(Specification)をリクワイアメントとマッチングして、作る訳です。それを、うまく行ったか否かテスティングしなければいけない。
ここには Inconsistent（矛盾）が一杯あります。あるユーザーの要求と一致しているか否かも、まだわからない。そこにロジカル的なチェッキングはしなければいけない。設計書もメカニカルプルービング（実験）もやらなくてはならない。そしてこれが終った後も完全に終ったどうかもわからず、また戻ってくるのです。従来のソフトウェアでは一番大変な部分の一つです。

－Q　：　Lyee でも一緒ではないですか？

－A　：　Lyee でも人間のミス、人為ミスをチェックするテストは必要でしょう。しかしながらあくまでも人為的なミスのチェックです。
テスティングの本来の意味は設計書作成までに様々なロジカルチェックを行う事です。設計書に基づいてコーディングを行う。＝これら入力ミス・バグエラーを取る事は、ソフトウェア工学の世界ではテスティングとは言わない。これは「確認」と言います。もちろん確認する事は必要ですが。

－Q　：　では「テストをしなくても良い」という事は理論的には正しいですか？

－A　：　正しいです。テスティングの部分については Lyee はとても単純です。
どうしてか？従来法にはコンプリート メカニズム フォー テスティングはありません。例えばテスティングに従来法ではユースケース図を使います。ユースケース図は完全ではありません。すべてが正しいとは言えません。それではフルテスティングにはなりません。あるユースケースは良くてもあるユースケースは心配になってしまう。何らかのメカニズムを使ったとしても、それはケースバイケースでしかありません。オールマイティではありません。
Lyee はシナリオ関数のお陰で自然にテストを行ってる。知らないうちに。
Lyee にはフォーマルモデルは必要がありません。シナリオ関数があるから。
従来法ではわざわざテスティングしなければなりません。テスティングとして。

－Q　：　Lyee はどの様なシステムに適合すると思いますか？

－A　：　様々なアプリケーションへの適用性を確認しました。
具体的には、ビジネスソフトでは４０１ｋシステムを、また、制御系ソフトではデジタル・ウォッチ制御ソフトとプロトコル制御ソフト等です。この確認から、プロトコル制御ソフトに適用できることだけで Lyee はすべてのソフトに適用が可能であると思っています。
　周知のようにソフト化にはスペシフィケーション（要件定義）が必要不可欠

で重要です。なぜならば、ユーザーの要件をコンピュータシステムとしての様々な条件を踏まえて現実性のあるものとして実現しなければならないからです。実現のためにはその要件をある形にまとめなければなりません。そのことをスペシフィケーション（要件定義）と言います。たとえば、UML などでユースケース形式として表現することです。

しかし、ユースケース形式などではすべての現象を捉えるスペシフィケーションは最後の最後まで出来ません。なぜならばそのスペシフィケーション（要件が遭遇する様々な事象の想定とその事象に対する対応の仕方；即ちプロトコルという概念と同義）は経験も知識も限定的であるエンジニアが行わなければなりません。限定的なものが全体を捉えるということは矛盾ですから原理的に不可能です。結局、すべての事象を捉えるスペシフィケーションは最後の最後まで人間技（エンジニアおよびユーザー）では困難です。すると、次にそのスペシフィケーション（要件定義）から現実性のあるものにするための取り組みが必要になります。現実性があるためには、要件が遭遇する様々な事象のコンフリクト（衝突）やライフレスを引き起こすデッドロックやループを回避しなければなりません。そのためのスペシフィケーションに対する確認手法が「フォーマルモデル化」、「標準化」、「ベリフィケーション」、「テスティング」です。これらの作業は従来法では必須です。なぜならば、スペシフィケーションのすべての事象を把握することは難しく複雑であるからです。

ところが Lyee はこれらのすべてを、シナリオ関数が機械的にしてくれます。（即ち、仕様書の分析と設計を必要とせずに）Lyee のシナリオ関数構造には永久ループが 2 箇所あります。

ひとつはパレット連鎖関数が行う W04→W02→W03→W04 のループです。このループには W04 と W02 の間にユーザーのインテンション定義（スペシフィケーション）を含みます。

他のひとつはパレット関数が行う基底論理を再起させるループです。このループは先に述べたループのユーザーが定義したインテンションを「意味であること」と確認するループです。すなわち、セマンティックチェックを行います。ご承知のようにループは従来法では許しがたい現象です。寧ろ、このループを回避しようとする作業が「スペシフィケーション」、「フォーマルモデル化」、「ベリフィケーション」、「テスティング」、「標準化」です。

にもかかわらず Lyee のシナリオ関数はループが骨子となっています。しかし、そのループは唯一の状態に遭遇すると自然に解けるのです。

これはユーザーの要件のコンポーネントがユーザー自身によって満足される状態です。つまり、ユーザーの意志を構成する最小の単位である単語をシナリオ関数に与えます。それを W02 が受け取り、次にその単語の意味になるものが成立するまで W03 のパレット関数がその単語ごとの基底論理をループで再起させ

ます。単語の意味になるものが成立するとループが解けて W04 を介してユーザーに報告されます。その報告をユーザーが確認し、与えた単語によって要件が満足できるかどうかを評価します。満足できなければユーザーは単語を変更したり追加したりします。するとパレット連鎖関数が W02→W03→W04 とループさせます。満足できればループが解けるのです。従来のプログラムのようなリセットに拠らずにループが解けるのです。

　このようなメカニズムは言葉を変えれば、機械的な「スペシフィケーション」、「標準化」、「フォーマルモデル化」、「ベリフィケーション」、「テスティング」であると思います。

　また、ユーザーの要件のコンポーネントである単語をシナリオ関数に与えますと機械的にソースプログラムが出来ます。

プロトコルは従来法では状態遷移図やデシジョンテーブルでエンジニアが設計します。しかし、すべての状態とそのアクションを捉えられません。また、状態遷移図やデシジョンテーブルで定義されたスペックをプログラムにするととても大きく複雑なものになります。しかし、Lyee のプログラムは小さくてシンプルです。そして、状態遷移図やデシジョンテーブルでプロトコルのスペックを設計しなかったのに、W03 のなかに状態遷移図やデシジョンテーブルのようなものが出来上がるのです。

すべてのソフトはプロトコル制御と同義です。そのプロトコル制御を上記のようなメカニズムを持った Lyee のシナリオ関数で例題として確認できました。

－Q　：　大規模な業務アプリケーション開発に Lyee は向いていると思いますか？

－A　：　ええ、向いていると思います。

－Q　：　それはどのような点が？

－A　：　４０１ｋの様なプロジェクトは従来法を使うと予算（費用）と時間が多くかかります。Lyee が一番ショートカットしている部分はユーザーから聞いた要求から仕様書を作っていく、この一番時間のかかる部分を Lyee は省いています。必要ありません。Lyee は単語を入れる事によって、W03＝つまり業務仕様がどうなってるか、このアイトレートするメカニズム(W04→W02→W03→W04)＝（シナリオ関数）によって、単語の妥当性を確認する事ができます。従来法では確認出来ないのです。いろんな単語・仕様・要求をお客様から聞く時は、ブラックボックスとして聞いた人の頭の中に Lyee で言うところの W03 ロジックが出来ているのです。それを持って帰って、その想像を作る訳です。処理順序や業務仕様の。ですから要求と出来上がったものが合致しているかどうか解らない。従来法では。

一方、Lyee の場合は、そうではなくてたとえ間違っていても良いんです。単語を聞いて、W03 はどうなってきたか確認しながら、単語をどんどん増やしていけばいいのです。それでシステムをグローイングアップをしていく。

ですからユーザーの要求から外れる心配がないのです。

－Q　：　Lyee とウォーターフォール型、プロトタイピング型を比較してください。

－A　：　従来法はウォーターフォールモデルに準じます。手戻りが発生した場合、全てを最初からやり直さなければなりません。

プロトタイプ型は確かに表現的には Lyee と似ているかもしれません。ユーザーの要求仕様に基づき、想像でプロトタイプを作っては直し、作っては直す。確かに最終的にユーザーのリクワイアメントに近づきます。ただし、いずれも方法論ではありますが、メカニズムがないのです。Lyee にはシナリオ関数があります。きちんとしたメカニズムです。従来法は常にケースバイケースである事は否めません。

－Q　：　Lyee の最大の長所はどこにあると思いますか？

－A　：　私が感じるのは、Lyee の最大の長所はテスティングが必要ないという事ももちろんその一つですが、ある要求から設計書を作って、ある成果物を作った時に、それが要求と合っているかどうか確認するというのは、とても大変な作業です。従来の世界では、ここはシステムによっても方法は変わっていきます。仕様書によっても目的によっても変わります。つまり先ほども申し上げた通り常にケースバイケースです。

それに比べると Lyee はこういった事は全く無関係なのです。どういうシステムを使いたいとか全く関係ありません。そこが良いポイントと感じています。

－Q　：　悪い点・矛盾している点はありますか？

－A　：　未だ判断できません。ただ何でも同じシナリオ関数で良いのか疑問を持っています。それを証明したいと思っています。これから研究が必要と感じています。他の先生もそう感じているのではないでしょうか？

－Q　：　LyeeALL の使い勝手はいかがですか？

－A　：　面白いですね。

今までは自分のイマジネーションを精一杯使いながら仕事を作ってきました。従来ならユーザーの要求を一生懸命聞いて“理解したもの”と考え、次のステップに移る。で、ちょっとでも自分の想像と外れると終わりです。

LyeeALL なら、仮に違っていても直ぐに修正出来るし、実際は希望通りに出てきます。

－Q　：　学生にも使わせてみますか？

－A　：　う～ん（笑）あまりに簡単にコードが出来てしまうので心配です。あまり簡単な道を教えると、難しい道は進まなくなってしまうから。LyeeALL だけを使わせるという事はしません。

－Q　：　メンテナンスビリティーは向上すると思いますか？

－A　：　確かに出来上がるコードはシンプルですが、Lyee のメカニズムをもっと深く掘り下げ、何に適用できるのか証明しながら判断したいです。

未だその結論を出すステップではありません。私の目的は開発する事ではなく研究する事ですから。

一方ビジネスとして捉えた場合、一番良い方法論を選択しなければなりません。アプローチを変更すればビジネスも変わってきます。

彼らにとっては辛い事です。あるところを開発して、プラットフォームがいつの間にか古いバージョンになりました。そうなると使った物も新しくしなければなりません。それは辛い。この業界は人に依存する部分、ヒューマンリソースに頼る部分もあってそれが交替するのも辛い。あるメソドロジーから違うメソドロジーにもっていく。それも辛い。様々な問題があります。Lyee は全て解決できると言う。だから研究が必要です。まあビジネスでは理論などどうでも良くてアウトプット・アウトカムが全てですが。

－Q　：　シナリオ関数は適用業務に関わらず普遍的な関数という事ですが、その点どのように評価されますか？

－A　：　マスマティカル（数学的）フォーマーとしては、関数の証明が必要です。抽象的にはわかります。けれどもソフトウェアに適用するとどんなアプリケーションでも適用できる。“何でも出来る。”というのはどうでしょうか？言葉だけではなく証明が必要です。色々なユースケースを与えて証明していくべきでしょう。プロコトルもＯＫ、銀行システムＯＫ・・・といった様に、色んなケーススタディを検証していく必要があると思います。マスマティックと応用は別物です。

－Q　「プログラムのソースコードの決定」が関数で解けるという事についてはどの様に思われますか？

－A　：　システムのやり方も発想としてはとても面白いし、その発想は今までソフトウェアの世界になかった。そこがキーポイントです。

ソフトウェアは実際にはハードウェアから生まれでたものですが、Lyee はハードウェアは全く無関係です。意識していません。たぶんこれはキーポイントの発明だと思います。

ウォーターフォール型はハードウェア存在から生まれたコンセプトです。だから設計書・仕様書などが出てきます。

状態遷移ダイヤグラム・データフローダイヤグラム等、これも元々はハードウェアを作り出すコンセプトです。繰り返しになりますが（笑）

オブジェクト指向と Lyee では何が違うか？

オブジェクト指向で Lyee に似ているのは、全てオブジェクトを使っている事です。単語もオブジェクトの１つですから。それだけです。似ている部分は。

残りは関係ありません。オブジェクト指向は全てオブジェクトです。Word もオブジェクト。そして Word と Word にはメッセージを交換する事でコンセプトを作るのです。コンセプトとは、オブジェクトの一般性・専門性・クラスなどの

属性を定義する。この部分は似ています。ではどの様にオブジェクト指向を使ってインプリメンティションを作るのか？例えば、UML を使います。しかし UML を使っても抽象的なクラスしかできません。様々なダイヤグラムを作る必要があります。そうしているうちにダイヤグラムの関係が見えなくなってしまう事があります。Lyee は違います。もっと単純です。概要モデルとしては Lyee とオブジェクト指向は似ていますが、オブジェクト指向だからオブジェクト指向のアプローチを使いなさい！その様な事に Lyee はなっていません。コンセプトとしては、オブジェクトですが、オブジェクト指向のアプローチを利用している訳ではありません。

－Q　：　では Lyee はオブジェクト指向よりも優れている？

－A　：　そこまでは未だ断定出来ません。お互いの優位点・問題点をアカデミックに洗い出して比較したい。Lyee の弱いポイントを見つけ出したいと思います。（笑）ただ感覚的には優れていると思います。感じます。まあ、私は学者ですから感覚的に物事を語りたくありません。しかしながら敢えて何故その様に感じるかといえば、私自身オブジェクト指向と Lyee 両方使ってみました。何かおかしいです。変です。苦労する時間と楽な時間、Lyee は具体的なソースコードが生まれる。オブジェクト指向では答えにならない。

更にドキュメンテーション、プログラミング構造、空のクラスで中に色々やって色々メッセージ交換も決めて、様々な過程・属性も入れて、ダイヤグラムも９つのダイヤグラム使って、ユーザーの値を入れて→そこまでやっても、Lyee で言うところの W03（ロジック）は未だ出来ない。W03（ロジック）はその次のフェーズとしてプログラマーの仕事になる。

それも考えてみればおかしな話です。そしてインプリメンテイションが出来上がって、見せる。ユーザーが違うというと、またぞろプログラマーが変更する。前にも述べましたが矛盾してますよね。そうした部分を勘案すると、感覚的には Lyee が優れているように思います。ただあくまでも感覚的なものですよ。

このあたりは是非 Lyee 研究ネットワークの成果に期待してください。

以上