Lyee　Internet　Information

対談

ドイツ　ドルトムント大学　　フォルガー・グルン教授

デリク・ピーターズ氏

ソフトウェア生産技術研究所株式会社　　根来　文生社長

ピーターズ氏

グルン教授

根来社長

日時　　：　２００１年８月２３日（木）
場所　　：　ソフトウェア生産技術研究所株式会社　本社
聞き手　：　岩手県立大学　　藤田ハミド教授

【フォルガー・グルン(Volker Gruhn)教授　略歴】

１９６３年生まれ。
１９９１年　　ドルトムント大学コンピュタ・サイエンス学部で博士号を取得。
１９９４―９６年　　中規模ソフトハウスである LION 社の主任技術者となる。
１９９７年　　ソフトウエア統合とソフトウエア開発、プロセス・モデリィング、電子ビジネスのコンサルティング会社ａｄｅｓｓｏ社を設立。
１９９７年―現在　　ドルトムント大学ソフトウエア・エンジニアリング学部の教授に就任。

＜主な研究分野＞
分散型・高度進化ベースの環境下におけるソフトウェア・プロセス支援、ビジネス・プロセスとソフトウェア・プロセス・コミュニティ間の相互発達、部品ベース・システム、E-ビジネスシステム・アプリケーションと部品ベース・ソフトウェア構造の開発

＜最近の出版＞
“Software Process Landscaping” in Special Issue “Process Simulation,” Software Process Improvement and Practice Journal, No.3, 2000
“Decision and Risk Analysis for Process Evolution” in; M. Lehman (ed.), Proceedings FEAST 2000, June 2000, London, UK

【デリク・ピーターズ (Dirk Peters)氏　略歴】

１９７１年　ドイツ・ボーフム生まれ。
１９９８年　ボーフム大学数学学部コンピュータ・サイエンス学科で修士号を取得。
１９９８年 – 現在　　ドルトムント大学で博士合過程に在籍し、同大学で研究及び教育アシスタントを務める。

＜主な研究分野＞
・コンポーネント・ベース・ソフトウェアのセキュリティ
・コンポーネント・ベース・ソフトウェアの開発
＜主な著書＞
・"Concurrent security modeling in a distributed Java-based E-Commerce Environment"
・"Requirements Analysis in Distributed Software Engineering Education: An Experience Report"

藤田教授　　それではまず一般的な質問から始めさせて頂きたいと思います。
バックグラントや現在どのような研究をやられているのかといった事について簡単に教えて頂きたいと思います。

グルン教授　**私は現在３８歳で、ドルトムント大学で教授を務めております。**
ドルトムント大学はコンピュータ・サイエンスの分野では、学生数でいうとドイツで最大の規模を誇っています。私の主な研究分野はソフトウェア・テクノロジーあるいはエレクトロニック・ビジネス、そしてコンポーネントに関するものです。

ピーターズ氏　**私の専門は数学とコンピュータ・サイエンスで、現在 Ph.D.を取得する為に研究を行っています。３年前からドルトムント大学のコンピュータ・サイエンス学部でグルン先生の下で勉強しています。研究分野はコンポーネント・ベースのソフトウェアに関するセキュリティーです。**

藤田教授　今回先生方に来日して頂いたのは我々が組織する国際学術共同研究プロジェクトに参加して頂けるか判断して頂く為ですが、我々がドルトムント大学を訪問して行ったプレゼンテーションを聞かれたのと、この数日間日本で学ばれたのを合わせても Lyee に触れたのは短い時間ですが、現時点で Lyee についてどういった印象を持たれたか、またこの方法論の将来性ついて意見をお聞かせ下さい。

グルン教授　**最初は新しいソフトウェア・エンジニアリングであるという点に興味を持ち、どういうものか見てみたいという事から始まったのですが、およそ１週間説明を聞きまして、考え方は理解する事ができるようになりました。この１０年間を振り返ってみますと、コンピュータの世界では様々なアプローチが出てきました。それらはソフトウェアの質などについて、いろいろな事ができると将来を約束するような話をしていましたが、そのほとんどは約束を果たせませんでした。**
その為、新しい方法論といわれても、これまで出てきたものが信用できなかった訳ですから最初は懐疑的でした。しかし Lyee について説明を聞いた時に、理解はできなくても新しい発想であるという事は理解できたので、どういうものか学んでみようという思い来日しました。ですがこちらに来てみて、現実的にはとても大変な時を過ごしました。なぜなら多くの馴染みのないターミノロジーの中で新しい考え方を理解しなれればなりませんので、メッセージを上手く掴むというのが非常に難しかったからです。しかし根来さんや皆さんが大変オープンでこちらが出した質問に対して丁寧に答えてくださったので、概要は理解する事ができました。
そしてこの方法が実際にビジネスとして使われている事も理解しました。今のところ詳細までは理解できていませんが、Lyee が生産性向上に繋がるという事は充分にありうるという事が見えてきました。

ピーターズ氏　**はじめにドルトムント大学で話を聞いた時に、これまでの開発方法と全く違うという事にとても驚きました。同時にこういった方法で本当にソフトウェアを生産する事ができるのだろうかとも思いました。簡単に信じる事ができなかった訳ですが、今回来日して数日間学び、ソフトウェア工場を見学して、半信半疑の状態から実際にできるのではないかと考えが変化しました。こうした方法でもソフトウェアを開発する事ができるという事が見えてきました。今後は、従来の方法と Lyee を比較して、双方の利点とマイナス点を計測する方法について、取り組んでみたいと思います。**

藤田教授　これまで何人もの同僚が多摩のソフトウェア工場を見学して、この方法論で短期間に開発を行う事を目の当たりにし、皆一様に驚かれましたが、おニ人も良い印象を持って頂いけたので大変嬉しく思います。
ピーターズ先生がおっしゃった通り、確かに Lyee と現在あるいろいろな方法論を比較していくメジャメントの問題も重要だと思います。Lyee がソフトウェアの世界のどこに位置付けられるのかという事を見出す事は興味深いと思います。グラン先生はオブジェクト指向のバックグランドを、ピーターズ氏はコンポーネントのバックグランドをお持ちだと聞いていますが、短期間の説明だけなので Lyee とその他の方法との技術的な違いについて、まだ明確ではない事は承知していますが、それでも、どういった違いがあると思うかについてお聞かせ下さい。

グルン教授　**その点は Lyee を理解する上で非常に重要なポイントだと思います。ソフトウェア・エンジニアリングの主要な考え方は、Lyee に適応できないと思います。例えばクラシフィケーションの考え方も当てはまらないでしょうし、抽象化といった非常に重要な考え方も簡単には適応できないと思います。あえて言うならば表面的でない方法でならば適応する事ができるかもしれません。**
明らかな点はセパレーショナル・コンサーンが重要であるという事です。セパレーショナル・コンサーンとは関連事項をどうやって分離するかといった事ですが、これが意味する事は独立性という問題です。Lyee はプログラムの１つ１つが独立性を持っているという特徴があります。これは非常に重要で Lyee は明確な方法で異なったリクヤイヤメントをそれぞれ独自のものとして扱います。例えば１つのリクワイヤメントから１つのシステムを作り、後にそれに追加を行い、動かしていって、以前作ったものと関連付けるという事が自由にできます。これが先程言いました関係事項の分離という事です。

これができますと当然システム作りやメンテンスの問題にとても寄与しますし、異なるリクワイヤメントに対し、それぞれの理論やアプローチ等の情報を取得する事もできます。

Lyee を使う事によって問題が簡単に解決できるという事がわかりました。深い問題はともかくとして、システムができるだけ早くほしい場合、迅速に対応できると思います。顧客が仕様のリクワイヤメントを出してきたら、すぐにシステムを作ってしまい、最終的には全てのリクワイヤメントとシステムがマッチするといった形式がとれると思います。この方法はインクリメンタルなプロセスと呼べると思います。この点に関してはオブジェクト指向に非常に似ているところがあります。オブジェクト指向はすぐに問題を解決できる訳ではありませんが、徐々に解決していくという意味では共通点があると思います。Lyee は情報システムの開発に非常に適しているのではないかと思います。あるいはユーザ・インターフェース主体の開発に適していると思います。つまりはビジネス・アプリケーションという事になります。データのトラッキングであるとか、データベースを中心にしたシステムです。更にユーザが短期に開発したいという時に適した方法であると思います。

ピーターズ氏　私も関連事項を分離していくやり方について、非常に上手く機能しているという事に非常に驚きました。今後更に深く見ていきたい点は、こういった方法の一種の副作用と言いますか、どういった波及効果があるかといった点です。

またどのようなプログラム、アプリケーションを Lyee で開発すると最上の効果が得られるのかという事を見極めたいと思います。同時に Lyee では開発しない方がいいといったアプリケーションがあるならば、どういったものなのかといった事にも関心があります。

根来社長　まず Lyee を発明したバックグランドを申し上げたいと思うのですが、私は大学で数学とコンピュータ・ソフトウェアについて教えていたのですが、その頃からソフトウェアの開発方法論について興味を持っていました。しかし当時はまだ方法論を考える為に必要なデータがあまりありませんでした。また抽象化する為の概念もありませんでした。その当時このような研究を大学で行うといった事は考えられませんでした。また当時は学生運動が盛んで大変荒れた時代でした。そういった理由から大学を辞めて、ある企業に就職しました。そしてその会社でたくさんの開発経験を積みました。

その後、私の本来行ないたかった研究を行なう事ができるようになりました。そこで私はソフトウェアの根本を見出す事をテーマにしたのです。

ソフトウェアを極限まで抽象化するという研究を行いました。ソフトウェアを理論で作るというのが目標でした。いうなれば経験と知識を使わないでソフトウェアを作るという事です。
Lyee の説明の仕方はまだ不十分で、心理学の世界、哲学の世界、論理学の世界が混在した形になっています。そういう意味で Lyee を説明する事はとても難しいと感じています。認識というのは常に曖昧なものです。機械の世界ではロジックと構造化で曖昧さをなくしますが、ソフトウェアの世界は構造化という作用がないのです。ロジックだけで曖昧さをなくさなければならないというのがソフトウェアの最大の特徴なのです。認識の曖昧さをロジックでどうやって取り除くかという事がソフトウェアの命題です。先程いろいろな技術がでてきたが、満足な結果を得られなかったという話がありましたが、それは実は命題を満たしていなかったから上手くいかなかったのです。オブジェクト指向についても命題を満たしているかという観点で検討する必要があると思います。CASE ツールなども一時流行りましたが、はっきり言って命題を何１つ満たしてはいませんでした。私は、Lyee は命題を満たしているかという事を先生方に研究して頂ければと思います。

藤田教授　　根来さんが提唱するソフトウェアの見方というのは、従来法との違いを強調した見方であると思います。この定義の仕方自体が既に、私達が考える従来のソフトウェアとは違ったものになっています。従来のソフトウェアは構造物のイメージで開発を行なってきました。それは構造物を作る方法と同じ様な方法論でソフトウェアも作れるのではないかという仮定の上で行なわれてきたものです。私はその仮定が間違っていたから上手くいかなかったのだと思うのです。Lyee はシナリオ関数に反映されている普遍的な、またフォーマライズされた定義というものが特徴になっています。これがそもそも他の方法と決定的に違う点です。
オブジェクト指向の例を挙げて話をしますと、オブジェクト指向はクラシフィケーションという定義が必要です。先程、関連事項の分離を特徴として挙げていましたけれど、本質的な違いは普遍性にある訳で関連事項が分かれているというだけではないのです。実際、開発やメンテナンスが簡単になっており、Lyee はソフトウェアの問題を大きく改善しています。これは普遍性があるからです。昨日デモンストレーションを見て頂いたと思うのですが、プログラムの構造が単一であるのにも関わらず、コンピュータで動かした時には従来のプログラムで作った時と同様のコード・パターン、ビヘイビアができます。これが従来法との最も大きな

違いですが、ある意味でこういった方法を取らなかったからこそ、過去の CASE ツール等が成功しなかったとも言えると思うのですが、私はあえて失敗であったというよりも、目的とするものに達する事ができなかったという言い方の方が正確ではないかと思います。

それでは次の質問に移りたいと思うのですが、Lyee が提唱しているソフトウェアの定義、シナリオ関数に表れた普遍性についてどう思いますか？ シナリオ関数はどの様なリクワイヤメントに対しても同一性を持った形で適応できる本質的なモデルになりうるでしょうか？

グルン教授　**今の質問は２つの問題が含まれていると思います。１つ目は Lyee が普遍性のあるモデルであるかという点で、一体その普遍性というのはどのレベルの普遍性なのかという事を見極めるという問題と、２つ目に過去に様々な方法論が出てきたのに何故それらは成功しなかったのかといった問題に分かれていると思います。最初の問題に答える前に、まず２つ目の問題について答えたいと思います。CASE ツールが成功しなかったのはソフトウェアの開発者に対して強要する点が非常に多かったからだと思います。従来のシステム開発は非常に創造的な仕事で、リクワイヤメントに対する対応やデザインなど、創造的な部分があるので、そこに対してあまりにも多くの制限を設けられますと、上手くいかなくなるのです。創造的な部分については様々な種類の方法によるプロセスを認めてもらう必要があります。**

しかしながら過去の方法論はソフトウェアを開発する上で、一定の順序で作業を行なわなければならないとか、ソフトウェアの総合的な環境を合わせなければならないといった制約があるため成功しなかったのです。次に最初の問題についてですが、Lyee は非常に強力なインフラを提示する可能性があると思います。何故なら異なるアプリケーションに対して、ある環境を定義する事によってインフラを整備する事ができると思うからです。現状のソフトウェア開発を見ますと、大きなチームで編成されていまして、簡単でシンプルな開発に関して似た様な事を繰り返していると言えると思います。例えばトランザクション・モニタリングやリレーショナル・ロジックといった問題です。しかし Lyee はインフラの観点からも普遍性がある故に非常に強力なものを提示する可能性があると思います。Lyee は非常にシンプルなソリューションを提示している方法論だと思います。強力なインフラを提示できると言ったのは、これまで同じ事の繰り返しで退屈な作業となっていた仕事を、Lyee は一気に解決してやってくれるからです。そういった事によってソフトウェアの開発者が、自分たちの強みを発揮できる本来の仕事に従事する事ができるので

はないかと思います。従来法もLyeeも最終的には同じ目的を持っている訳ですが、Lyeeを使う事によって成功の可能性が高まると思います。

ピーターズ氏　一般論としてチューリング・マシンに対するソフトウェアの普遍的な定義はありますが、それは普遍性について語る際の非常に低いレベルの定義にすぎないと思います。Lyeeは理論的に支援された方法論であり、普遍に非常に近い可能性があると思います。現状では上位におけるプログラミングについての普遍的な定義を行なえるものは出てきておりません。Lyeeはそういった可能性を持っていると思いますので、その点について研究してみたいと思います。

根来社長　チューリング・マシンについての評価については全く同感です。
Lyeeとチューリング・マシンとの関係について質問される方もいるのですが、それは全くナンセンスです。ソフトウェアとチューリング・マシンは何の関係もないのです。私がこれまでたくさんの開発を経験してきて感じたのは、ソフトウェア開発における根本の問題は何かという事がまだ認識されていないという事です。
例えばリクワイヤメントは様々な状態がありますが、それらについて難しいとか簡単だとか、レベルが高いとか低いといった事をよく言いますが、リクワイヤメントに難易度やレベルといったものは存在しないのです。
またクラスという概念についてですが、クラスは方法論としては成立しない概念です。私はオブジェクト指向がクラスの概念を持ち出した途端に他の方法論と同じ問題を持ってしまったと思いました。解決策を持たない概念を取り入れてしまった為にそういう状況になってしまったのです。
そういった概念がソフトウェアの世界ではたくさん生まれてしまったのです。このたくさん生まれた概念が本当にソフトウェアの世界に必要な概念なのかという事について検討する必要があると思います。

藤田教授　私はモデレータですので本来自分の意見を言ってはいけないのですが、あえて１つだけ申し上げたいと思います。今チューリング・マシンの話が出てきましたが、チューリング・マシンと現在のプログラムにどういった関係があるのかという問題は重要だと思います。それはチューリング・マシンによってソフトウェアの理論的な背景が出てきたのは事実ですし、みなその点について研究してきたからです。ソフトウェアを研究していく為には、どうしても理論的な背景を求めるものです。理論があるという事は普遍性という問題に行き当たり、様々な条件の中でプログ

ラムが決定されていくという事を求めていくことになります。ソフトウェアに関わる人達はみな、理論を打ち立てようとしてきましたが、私が知る限り本当のソフトウェアの理論というものは結局どれも成功していません。私の知っている限り、根来さんが初めて本当の理論を打ち立てる事ができたと思います。ですから一般的にソフトウェア・サイエンスに理論というものはないとは言えず、ある事はあるのですが、ただ本当の理論には至っていないということだと思います。今までのソフトウェアはまずリクワイヤメントや目標があり、それをエンジニリングを使って目標に到達するという過程を経て作った総合的な構造物、あるいはモデルでした。エンジニアリングを行なう過程の中でできたプログラムの中に目的がどれくらい反映しているかといった問題もあるのですが、更に目的や状況が変わったりすると、今まで作ったものがフィットしないといった事が起きてしまいます。それを理論で解決しようとしたができませんでした。

ところが Lyee はシナリオ関数に代表される普遍性を持つ事で、プログラムの中のリクワイヤメントからどういったビヘイビアをとるのかといった事など、全て理論によって明解にしたのです。普遍性があるというのはメモリの量など様々な制約の中で、プログラムを決定できる方法という事です。

先生方ももちろん頭の中にある種の普遍的なモデリングというものをイメージとしてお持ちだと思いますが、それは１つにはチューリング・マシンによって作られてきたものだと思います。

根来さんがチューリング・マシンとソフトウェアは全く違うもので比較するのはナンセンスだとおっしゃった事はよくわかりますが、チューリング・マシンそのものがある種の理論的な普遍性を定義してきたという事は事実だと思います。

ピーターズ氏　**確かにソフトウェア・サイエンスの世界にもチューリング・マシンの様な理論はありましたが、本当の理論と言える様なものはなかったというのはあきらかです。また Lyee に普遍性があるかという点についてですが、例えばアインシュタインの相対性理論は普遍的な理論であり、この理論により他の分野の様々な問題を理解し解決するのに役立ちました。相対性理論の登場により世界に対する理解が深まった訳です。理論が出てきた当初はなかなか受け入れられませんでしたが、時を経て今では受け入れられるようになりました。アインシュタインがどのようにしてこの理論を受け入れさせたかと言いますと、従来の物理学で説明できる事象を全て相対性理論で説明し、更にそれ以上の事象もカバーする事ができる**

という事を実証してみせたのです。その事により相対性理論がこれまでの理論よりも優れた理論であるという事を証明しました。現在量子力学という新たな理論が出てきていますが、まだ、相対性理論の更に先を行くものとして１００％受け入れられたとはいえる段階にはありませんが、アインシュタインが取った方法と同じ手法で受け入れさせようとするのは間違いありません。

Lyee はソフトウェアの普遍的な定義を行なえるといっていますが、私はまだ数日間説明を聞いただけですし、様々な疑問が頭の中を渦巻いている状態ですので、同意する訳にはいきませんが、アインシュタインの相対性理論のように、Lyee がソフトウェアの世界で従来行われてきた方法の全てを網羅し、更にそれ以上の事ができる事を示すことによって、説得力を持つと思います。

１つ付け足しますと相対性理論は従来の古典的な物理学の更に一歩先に進んだ理論であるという事を証明する際に、従来の理論の延長線上のやり方で説明をしました。しかし強調したい事はだからといって相対性理論が従来の理論の延長上の理論では全くないという事です。物理の３大要素である時間と重さと距離に対して全く新しい定義をしています。相対性理論は３大要素の全く違う定義をしながら、効力については従来のやり方で証明したのです。私は Lyee が相対性理論に匹敵しうるものだと思いますので、Lyee も同じ様に従来のやり方で証明するべきだと思います。

根来社長　大変参考になるご意見だと思います。ニュートン力学とアインシュタインの相対性理論、そして量子力学などは要約すれば全て座標に関する問題や距離をどう扱うかという問題を取り扱っています。それに対して Lyee は意味の問題、意味は距離を持つかといった視点に立っています。そういった意味では物理学の理論の系譜に近い論理性を持っていると私は思っています。ここで大切な事は、ニュートン力学はニュートン力学として完成した論理性を持っていたという事です。だからアインシュタインは相対性理論を研究する際、ニュートン力学をはっきりとイメージする事ができました。ですからニュートン力学の上に相対性理論が成り立っている訳です。ソフトウェアの世界ではそういったベースとなる理論は何も無いのです。ですから従来的な世界とは何かというと、実は何もないのです。ペーパーはたくさん出ていますがほとんどが無意味なペーパーです。それらはほとんど役に立ちません。

だから悪いというのではなく、それを事実をして認識する事はとても大事なのです。決して学者の先生を責めている訳ではありません。

グルン教授　**私は１つの事にあまりにもいろいろなものを付与して話をするのは、かなり慎重にした方がよいと思います。世の中は簡単に言いますと優れたソフトウェア・エンジニアリング、ソフトウェア・プロセスを求めているという事に尽きると思います。広大な野望を持っている訳ではなく、控えめな願望を持っているのです。それは安く・早く・より良くできるという事です。ですから、そういったレベルで見ていきたいと思います。**

根来社長　**ソフトウェアは原料がありません。原料のない世界ですからソフトウェアの問題は生産性だけなのです。ですから生産性の問題は一番大事な問題なのです。**

藤田教授　もう１つ質問をさせて頂いたと思うのですが、話の流れが変わってしまいましたので、少し軌道修正をしたいと思います。根来さんはソフトウェア・サイエンスの世界に哲学的科学とも言うべき考え方を持ち込みました。それはソフトウェアを科学する為の基盤が必要であると考えたからです。今のお話の中で物理学との比較といった話がでてきました。これ自体は非常に面白いトピックを含むものですし、矛盾をはらむものでもありますので、このテーマについては後にもっと時間をとって話をしたいと思います。

それではもっと具体的な話に移りますが、先程グラン先生がオブジェクト指向と Lyee の共通点として、インクリメンタルなプロセスにより、プログラムを作っていくという特徴を挙げていました。

この点についてもう少し述べますと、まず、オブジェクト指向は問題解決の為に、クラスという概念を使って抽象化を行い、クラス間のメッセージのやりとりを行なう方法を使ってプログラムを作り、実行するといったプロセスを取ります。

Lyee はこの点については全く違う方法を取ります。普遍的な構造を持つシナリオ関数を使ってプログラムを作りまして、どの問題に対しても同じ構造を持ったプログラムが生産されます。この点は従来の方法ともオブジェクト指向とも全く違います。従来の方法ではエンジニアリングの違いというのが大きな問題として出てきます。例えばオブジェクト指向というエンジニアリングで作られたシステムについて、何かを少し変えようとするだけでも、そのエンジニアリングに沿って変えていかなくてはならないという事が起きます。

しかし Lyee ではそういうことをする必要は全くありません。つまりソフトウェア・エンジニアリングというものが必要ないといった言い方もできるのです。従来の開発方法ではソフトウェアの開発には知識や経験が

必要で、それによってユーザーの要件が固まりました。知識なくしてプログラムの生成はできませんでした。しかしLyeeではそれが可能だというのです。

グルン教授　**現時点でオブジェクト指向とLyeeの違いを明確に言う事は難しい事だと思いますが、Lyeeは抽象化を行なっていないと言われていますが、私はやはりなんらかの方法で抽象化を行っていると感じています。Lyeeはユーザの意図がリクワイヤメントになり、それから属性を定義して、必要がないものは排除し、即座にプログラムを開発するというプロセスで開発を行います。しかしユーザから出てきたリクワイヤメントには様々な属性があり、またユーザがしゃべっていない属性も背後にあるので、扱っている表示された情報、それからデータベースに入れられる情報は確かに少ないかもしれませんが、なんらかの形で抽象化は行われていると思います。また、Lyeeのインフラが抽象化を助けているのではないかと思います。もちろんLyeeの扱うデータはユーザの使うリクヤイヤメントと緊密な関係にあるという事は理解していますし、全体の作業が非常に改善されていると思います。**

Lyeeの開発手順についてはたいへん明解にプロセスを説明して頂けたので、ユーザのリクワイヤメントに直結した形でシステムが完成するという事は理解しました。しかし全くエンジニアリングがいらないという言い方をするとちょっと極端ではないかと思います。今日見学したソフトウェア工場では数ヶ月のトレーニングでLyeeによる開発が行なえるとの説明がありましたが、いずれにしても学ばなければならないエンジニアリングはあると思います。今後Lyeeが様々な形で使われていけばいくほど、やはりもっとバックグランドや何がエンジニアリングに相当するものかを含め、明解にする事が有効だと思います。

ソフトウェア工場で話をした人の中には経験のある人もいましたし、非常に若い人もいました。またエンジニアリングの専門の人もいると思いました。ですから、そういった役割はやはりあり、エンジニアリングのプロセスも含まれていると思いますので、それを明らかにするという事は長期的にみて有効だと思いました。

根来社長　**Lyeeがエンジニアリングという意味で抽象化しているところは基本的にありません。何故そういう事ができるのかと言いますと、述語構造がその代役を果たしているからです。述語構造という７つのボックスの構造が実行時にエンジニアリングの役割を果たしているのです。それからソフトウェア工場で働いている人達はかなりの時間を費やして勉強をしています。年間５０％は教育に時間を使っています。何故教育にそんなに**

時間をかけるのかというと、エンジニアリングの為に教育が行なわれているのではありません。将来彼らはクライアントと話をする事になるのですが、クライアントは従来の世界でものを見ている訳で、その時にクライアントにLyeeを理解させる必要が出てきます。極端な話をしますとクライアントに従来行っていたような仕事はする必要がないという事を説明する事ができるようになる為に勉強しているのです。
具体的な例を申しますと、従来の世界で換算すると２００万ステップ、単語の数が１８０万を超えている大きなシステムをLyeeで開発したのですが、それをユーザの要望で１０月からもう一度作り変える事になりました。そこで生産性がどれくらい改善されるかという予想を立てたのですが、前回開発した時よりも更に５０％改善できると予想しています。大変大きな数値ですが、何故そのような事が可能かと言いますと、前回の開発経験でユーザ側が余分な仕事をしなくてもいいという事をユーザに理解してもらえたので、ユーザ側の生産性が上がるという予想を立てたのです。具体的に言いますと要件を定義するという作業について、前回の開発では、ユーザはそれだけの情報では開発は行なえないと考えて余分な仕事をしてしまったのです。今度はそういった仕事をしなくてもいいという事をユーザもわかったので、リクワイヤメントを定義する時間が短縮できるのです。

以上