Lyee Internet Information

コンピュータサイエンス界の

学識者による討論会（前編）

日時	：	２００１年2月２２日（木）	
場所	：	ソフトウェア生産技術研究所株式会社　本社	
参加者	：	カールトン大学（カナダ）	ミッシェル・バルボー準教授
			フランシス・ボルドロー助教授
		マクマスター大学（カナダ）	リダ・ケドリ助教授
		アデレード大学（オーストラリア）	マイケル・アウズホーン上級講師
サポート	：	岩手県立大学	アイサム・A．ハミド教授

バルボー準教授　ボルドロー助教授　ケドリ助教授　アウズホーン上級講師

【ミッシェル・バルボー(Michel Barbeau)準教授　略歴】

１９５７年生まれ。現在４３歳。

１９９１年	モントリオール大学コンピュータサイエンス学科で博士号を取得。
	シャーブルック大学の助教授に就任。
１９９４年	同大学の準教授に就任。
２０００年	カールトン大学の準教授に就任。
現在	同大学にて OS、分散型オブジェクト、ワイヤレス・モバイルとコンピュータ・ネットワークに関する講義で教鞭を執る。

【フランシス・ボルドロー(Francis Bordeleau)助教授　略歴】

１９６５年生まれ。現在３６歳。
１９８９年　モントリオール大学で数学学士号を取得。
１９９１年　ケベック大学ハル校でコンピュータサイエンス学士号を取得。
１９９３年　カールトン大学コンピュータサイエンス学科で修士号を取得。
１９９９年　同大学システム・コンピュータ工学部で博士号を取得。
現在　通信会社数社（カナダ通信研究センター、CML ATC、CML Versatel、Mitel、ノーテルネットワークス、Rational）との産業プロジェクトに参加。

【リダ・ケドリ(Ridha Khedri)準教授　略歴】

１９６３年生まれ。現在３７歳
１９８２年　チュニジア チュニス大学で物理学・化学学士号を取得。
１９８５年　チュニス大学でコンピュータサイエンス工学学士号を取得。
１９９３年　ラバール大学コンピュータサイエンス学科で修士号を取得。
１９９８年　同大学同学科で博士号を取得。
マクマスター大学コンピュータサイエンス学部の助教授に就任。
現在　マクマスターソフトウェア工学研究グループ（SERG）のメンバー。

【マイケル・アウズホーン(Michael J Oudshoorn)上級講師　略歴】

１９６３年生まれ。現在３８歳。
１９８３年　アデレード大学で学士号を取得。
１９８４年　同大学で優等学位学士号を取得。
同大学コンピュータサイエンス学部の指導員に就任。
１９８９年　同大学同学部の講師に就任。
１９９２年　同大学同学部で博士号を取得。
１９９４年　同大学同学部のシニア講師に就任。
現在　同大学同学部の上級講師。
ACM（計算機協会)、IEEE（電気電子学会)、CSA（コンピュータ・サイエンス協会)、ISCA（コンピュータ・アプリケーション国際協会)、ASEI（オーストラリア・ソフトウェア工学協会）のメンバー。

【　Lyee との出会い　】

－　簡単な自己紹介と研究されてきた分野を教えて下さい。

ボルドロー助教授　私はカナダのオタワ州にあるカールトン大学でコンピュータ・サイエンス学部の助教授を務めております。
主な研究分野はソフトウェア・エンジニアリング、リアルタイム・オブジェクト指向システム開発、それにモデリング技術などです。
また、ここ数年はいろいろな企業のコンサルティングも行っております。通信分野をはじめ、様々なプロジェクトに関わっております。
年齢は３６歳です。

バルボー準教授　私も同じくカールトン大学のコンピュータ・サイエンス学部で準教授を勤めております。１９９１年にモントリオール大学でコンピュータ・サイエンスの博士号を取りました。
主な研究分野は通信です。特にモバイル、ワイヤレス・ネットワーク、それから衛星関連。この様な分野を研究の中心にしております。これをソフトウェアの観点からどう捉えていくかという事が中心課題です。
年齢は４３歳です。

アウズホーン上級講師　私はオーストラリアのアデレード大学で上席講師を務めております。上席講師というのはアメリカ・カナダで言いますと、準教授と同じ立場になります。１９９２年に博士号を取りました。ソフトウェアに関する私の関心事は変化してきておりますが、最近ではフォーマル・メソッドや分散システム、それから知的エージェントが研究の対象になっております。また産業界とのプロジェクトにも関与しています。

ケドリ助教授　私はカナダのマクマスター大学で助教授を務めております。
私の専門は、関係代数学を使ったソフトウェア・リクワイヤメントに関するバリデーションおよびベリフィケーションです。
また、ファンクショナルなテスティングやコンカーレンシにも興味をもっています。

― Lyee を知った経緯について教えて下さい。

ボルドロー助教授　私の場合は昨年の６月にオタワ州で行われたプレゼンテーションを聞いたのが Lyee を知ったきっかけで、それから１０月、１２月と続けて説明を聞く機会がありました。

― なるほど、それではハミド教授が行かれた時ですね。

ハミド教授　バルボー先生、ケドリ先生も同じです。アウズホーン先生は元々私と共同研究をやっておりまして、その関係で Lyee を紹介しました。ですから、今回初めて Lyee について勉強した訳です。

― それでは今回の来日の目的を教えて下さい。

ボルドロー助教授　Lyee についてもっと知りたいと思ったからです。
Lyee という開発方法論は、いわゆる普遍的な構造を持ってソフトウェアをシステマティックに捉えるという事を言っている訳ですけれども、この点に非常に興味を持ちました。
ですから、まず方法論を理解するというのが、来日の目的です。
今はこの点について、どの様な分野、どの様なアプリケーションに適用できるのか、それから Lyee の方法論が持っている、主な要素はどういうものなのかという事もかなりわかってきたと思います。
それから、もう一つの目的は、プロジェクトに関することですけれど、参加するかどうか決めるには、まず理解しなければなりませんので、今回来日して学び、参加の可否を決めたいと思います。

バルボー準教授　カナダでプレゼンテーションを聞いた時に、この方法論は一体どの様なものなのだろうと好奇心を持ちました。
最初は哲学的なアプローチを取っていることに惹かれました。
さらに、長年の知り合いであるハミド先生との話も加わって、興味が深まりました。ハミド先生が Lyee についてかなりよく理解された段階で、私にこういう方法論があると教えてくれたのです。
そして新しい考えや異なる思考法に触れる為に来日しました。
この様な機会を通して多くの事が学べます。もちろん、今この瞬間に、どういう形で Lyee に貢献できるかは、まだなんとも言えないのですが、最終的にはなんらかの形で貢献できると思います。
また、私にとっては異なる考え方について、科学的な見解を交換し合うというのが目的でもあります。

アウズホーン上級講師　私の答えは一番短くなると思いますが、Lyee とは何か、このプロジェクトはどういったものなのかといった事が知りたいと思い来日しました。

ケドリ助教授　私の大学でプレゼンテーションされた時に、私どものソフトウェア・エンジニアリング・リサーチ・グループの中では、いろいろな議論が起こり、各人が様々な意見を出しました。プレゼンテーションの中身が他の方法論と関連しているとか、自分の過去の研究と繋がっているとか、その他いろいろなディスカッションがありました。そういった中で共同研究者として参加しないかという要請を受けたので、グループの中で少しでも時間があるのは私だけだったものですから、ともかく、私が一体 Lyee というものは何なのか調べてみたい、聞いてみたいと思って、参りました。

ハミド教授　デビッド・パルナス先生という高名な先生も、来日される予定でしたが、手術がありまして、来日を延期する事になりました。しかし時期を改めていらっしゃる事になっています。

【　Lyee に対する評価　1　】

データ項目毎に独立しているという事は、対象としている問題を分離するという原理の応用で高く評価できる。（ケドリ助教授）

―　それでは、Lyee の売りについていくつか質問をしたいと思います。
まず Lyee の最大の特徴であります、プログラムがデータ項目毎に完全に独立している事に関してはどう思われますか？

バルボー準教授　私はこの点について、ソフトウェアのタイプによって違うのではないかと思います。
Lyee がカバーしている範囲というものがあって、その範囲の中では正しいと思いますが、範囲外のものでは、ソフトウェアの単位を独立した単位で見るだけでは十分ではなく、前後関係が必要になると思います。
全てのソフトウェアでプログラムが完全に独立させられると言い切ってしまうには、まだ充分ではない気がします。
例えば、今朝、根来さんに前後関係考慮しなくてはならない通信の例を出して話しをしたのですが、ソフトウェアの全ての範囲でデータ毎にプログラムを独立させられるという仮説が正しいとは

思いません。あるものに対してなら正しいと私は思います。

― あるものといいますとどういったものでしょうか？

ハミド教授　この問題に関しては、バルボー先生も根来さんもまだ答えを出していません。私はこの問題についてはもう少し研究する必要があると思います。ですので、今回のプロジェクトで扱いたいと思います。これについてはバルボー先生も同意見です。

― バルボー先生は感覚的にどの部分に適合すると思いますか？

バルボー準教授　現状では、Lyee のもつ可能性がどのくらいのものなのか、はっきりしていません。
データ中心の双方向アプリケーションには適していると思うのですが、システム・プログラミングや通信サブシステム、オペレーティング・システムなどの分野は、Lyee がそれほど力を発揮しないのではないかと思います。
何故かと言いますと、こういったシステムは個々に関係するソフトウェア・モジュールでできているからです。これは Lyee の開発方法とは合い入れないものでしょう。

アウズホーン上級講師　プログラムが分かれているのは、大きなシステムを扱う時に、ものすごい量のプログラムを扱う事になるので、データ結合の問題に対応しなくてはならなくなり、これが最終的には問題になると思います。ですから大きなアプリケーションや大きなシステムを扱う時に、この方法が適しているかどうかという事についてはちょっと判断を留保します。ですが結局難しいのではないかと思っています。
これからこの方法が最も向いているのはどういったシステムなのかという事について検証していきたいと思います。今回は時間がなかったので、なんとも言いきれないので、この問題については、答えを出さないままにしておきたいと思います。

ケドリ助教授　私はプログラムがデータ項目毎に独立しているという事を、高く評価しています。対象としている問題を分離するという原理の応用で、この考え方は何の問題もなく受け入れられるものです。
ですが３日間のセミナーでは、どのようにこの技術を使ってデザインの問題を解決するか、学生に伝えることはできません。
つまり、どの分野に合っているかという事については、今まで聞いた話だけではわかりません。ただ、発想は本当にすばらしいと思います。特にメンテナンスの事を考えた時に、修正が必要な

ところだけを取り出すというのは素晴らしいと思います。

― パレット連鎖関数のメカニズム，W02，W03，W04 の動きのメカニズムについてはどの様に評価しますか？

バルボー準教授 それは Lyee のもっとも根本的な特徴を示していると思います。
Lyee に対してはこのポイントを聞いて興味が湧きました。

ボルドロー助教授 確かに重要なポイントですね。

アウズホーン上級講師 基本的にこの点が Lyee メソッドだという感じがします。

ケドリ助教授 この点について１つコメントしたいのですが、Φ・・・・・といった式が出てきますが、率直にいって表示の仕方がよくないので、数学者が見ても意味がわかりません。あの式そのものは数学的に書かれているかの様に見えるのですけれども、普通の数学者が見て、式の内容や意図がわかるものではありません。Lyee の特徴を聞いた後にそれを見ると、何を言いたいのかという事が見えてくる訳ですけれども、あの式そのものは、数学者が理解できるものではないと思います。
これからいろいろなリサーチをしていきますけれども、内容をどうやって数学者が見てわかる様な書き方にするかというのも１つの研究対象になっていくのではないかと思います。

ボルドロー助教授 しかし、例えばソフトウェア・エンジニアリングや方法論という観点から言うと、大切なポイントはその方法論の核がなんであるかという事になるので、そういう表記については、あまり重要な問題ではありません。もちろんフォーマル言語で表現されれば、それはそれですばらしいですけれども、それよりも実際に適用される時に、定義がきちんとなされているかという事の方が重要であって、数学的表現は別の問題です。

― 従来法に比べてソースが非常に大きくなるので、レスポンスの問題などを気にされる方が多いのですけれども、その点についてどの様にお考えでしょうか？

ボルドロー助教授 今朝のセッションでもその事について話題になったのですが、
Lyee にとってコードの大きさは問題だと思います。ある環境では大きさは問題ないのかも知れませんけれども。しかし、根来さんもおっしゃっているように、ソース自体を小さくするという事は可能でしょう。しかし現在作られているプログラムのサイズはやはり Lyee にとって問題だと思います。

アウズホーン上級講師 Lyee で作ったプログラムのユーザーがアプリケーションに問題を

見つけた場合を考えてみてください。プログラムのサイズが５００でも５００万でも、ともかく大きかったとしても、自分達でコードをデバッグする事はないので、これを問題だと感じる事は必ずしもありません。しかし、いずれにしても、サイズの問題は、パフォーマンスに関わる可能性がありますので、今後リサーチの対象にしていくべきだと思います。

ボルドロー助教授　またその一方で、Lyee の開発方法を変えずに、コードの数を減らすためにコード生成を変更する事は可能でしょう。しかし、そうした試みがひょっとして何かマイナス面とか損なうところがあるかどうか見ていく事も、プロジェクトで取り上げるべき課題だと思います。

バルボー準教授　コードのサイズは、やはり問題だと思います。
特にプログラムのフットプリントが問題となるパーベイシブ・コンピューティング・アプリケーションのためのソフト開発に Lyee を使うことを考えると、問題だと思います。けれども他の方がおっしゃった様に、これは最先端のコード生成技術を使えば、実は簡単に解決できる問題です。この問題は、私が Lyee に貢献できる興味あるテーマではないかと思います。

ケドリ助教授　私はそういう見方そのものについては、基本的にあまり賛成しません。と言いますのは、皆いわゆる長いプログラムが好きではなく、長いプログラムを問題だと考えています。ではいったい何を指して問題としているのでしょうか？従来法で作られた長いプログラムというのは、確かに問題とされますが、それは何故かと言うとメンテナンスやテストが難しくなるからです。
つまりパフォーマンスが良くないので、長いプログラムは悪いという訳ですね。ところが Lyee というのは、メンテナンスやテストの問題、特に構造テストの問題をクリアしているので、そういう事であったら、じゃあ一体問題とは何かという事になるでしょう。しかし、私もこの点について検証していくというのは非常に重要だと思います。
今は、まだ理解していないところがあるので、Lyee をサポートしきれませんが、この問題について研究をしていくという事は意味があると思います。

Lyee によって短時間のバリデーションに
なるという事は確か。（ボルドロー助教授）

― Lyee ではプログラムを作って動いたのならテストは必要なく、Lyee の構造によりプログラムが動いた時点で、プログラムの問題は解消されているという事については、どう思われますか？

ボルドロー助教授　それは言葉の使い方という気がします。テスティングやバリデーションと呼ぶ人もいますが、Lyee ではコードを自動生成で作っていく過程の中に、コードレベルでしなければならなかったテストが消去されています。
しかし、このこと自体は他のツールでもされますが、Lyee ではイタレーションのプロセスによってサイクルが短くなって、ユーザーが検証しなくてすむようになっています。テストとは何かの解釈によりますが、Lyee がテストを必要としないとは言えるでしょう。また、バリデーションがプロトタイプ技術の場合の様に短時間のバリデーションになるというのは、確かなことで、大変すばらしいと思います。

バルボー準教授　私も同感です。

アウズホーン上級講師　テストがいらないと言いきってしまわれると、それではとても単純な質問をしたくなります。例えば航空機の製造会社から「うまく動くから信用しろ、でもテストはしていないよ。」と言われて、それで、飛行機に乗る気がするでしょうか。

ケドリ助教授　私もボルドロー先生のおっしゃった事に賛成なのですけれども、もう少し具体的に申し上げますと、テストというのは２種類のテストがあると思います。１つはホワイトボックス、言い換えますと構造に関するテスト、もう１つはブラックボックス、機能に関するテストです。構造に関するテストについては、Lyee は構造が明解だという事ですから、こちらの問題に関しては、テストは必要ないというのは、合っていると言えるでしょうが、機能面については実際テストをやっているでしょう。ですが、構造上のテストがないということは、コストの削減につながりますね。

― 実は今その質問をしようと思っていたのですけれども、あきらかに従来法に比べテストの負荷は減るというのは認めますか？

ボルドロー助教授　コード生成自体は確かにテストの負荷を減少させます。特に Lyee の方法論では、コード生成は論理要素などを指す高次レベルの表

現から出たものですよね。コード生成がそうした高次の表現から
自動生成され、そのコード生成ツール自身も正しいのであれば、
当然テストはしなくてよくなります。これは全くその通りです。
ただ、ケドリ先生がおっしゃる様に機能的なテストは必要です。
ともかく、Lyee で使われているようなコード生成技術は一般に
コードテストを減少させます。

アウズホーン上級講師　私はまだ Lyee を勉強して間もなく、使ってもいないのですけれ
ども、テスト面が減るという事に１００％納得していません。
その理由としましては、アクションやワードから、実際のボタン
操作に至るまで、やはり、いろんなチェックやテストが必要では
ないかと思います。特に自然言語で書かれた顧客の要件から
ワードを引き出すだけだと言っても、他の人がそれを明確にする
ために翻訳していると思います。そういう意味で、ちゃんと書か
れているかどうかチェックする必要があるでしょう。ですから、
全体として、テストがそれほど減るのかという点については、
１００％納得していません。

ケドリ助教授　私もある意味で、同じ意見で、機能面のテストは絶対していると
思います。どこの部分がテストを必要としないか例をあげますと、
あるシステムで問題が起きたので、新しく作り直したとします。
従来では新しく作ったものが後退する事がない様に、前にやった
ものと同じテストを繰り返すという事をやります。そして経路の
カバーチェックをして、プログラム内の全ての部分をチェック
しますけれど、Lyee の場合は、前のプログラムも新しいプログラ
ムも同じ部分があり、その部分のテストはしなくてもよくなると
思います。

ボルドロー助教授　私はそうは思いません。シーケンシャル・システムですので。
私は、インター・シナリオ・リレーション（内部シナリオ関係）
に関する問題について長年、研究してきましたが、Lyee のプログ
ラムには、インター・シナリオ・リレーションがありません。
リアルタイム・システムやドリブン・システムの場合と違います。
これは、結局シーケンスがあるという事は、シーケンスがきちん
と定義されていれば問題はないというのが、長い間研究をして
きて感じた事なのですが、Lyee の場合は、作り直したものは毎回
データ結合が行われていますね。ケドリ先生がおっしゃった様に
同じ部分はテストをしなくていいというのには、賛成できません。
前と一緒だからといって、それが Lyee の特徴によって関係なく

なるとは言えないと思います。

－　ハミド先生、今お二人がおっしゃったのはそういう違いですか？

ハミド教授　　人によって取り方は違うと思います。見方の違いだと思います。どっちも正しいと思います。

－　基本的に言ってらっしゃる事は同じですよね。

ハミド教授　　だいだい同じです。ただ取り方の違いです。

－　概ね負荷が減るという事ですよね。

ハミド教授　　結論としてはそういう事です。

バルボー準教授　　私の見解としましてはテストの負荷が減るというのは、その通りだと思います。ユーザーの意図というのはバリデーションという形で、チェックされていく訳ですけれども、ともかく意図をソースコードにシステマティクに翻訳生成していくのですから、それに対するテストは当然いらなくなります。しかし、ソースコードは意図を反映したものであり、意図の形成に問題があった場合には、それがソースコードに反映されてしまうのではないかと思います。

後編につづく