R&Dシンポジウム　基調講演

システム化・自動化の進展と人間の役割

筑波大学　大学院システム情報工学研究科　リスク工学専攻　教授
稲垣　敏之　氏

1952年、京都府生まれ。1974年京都大学工学部機械系学科卒業。1979年同大学院工学研究科博士課程修了。同年ヒューストン大学助手。1980年8月筑波大学電子・情報工学系講師。87年同助教授。94年同教授。現在、筑波大学大学院システム情報工学研究科教授。この間、1990年−1991年カッセル大学研究員（アレクサンダー・フォン・フンボルト財団奨学研究員）。人間機械共生系の研究に従事。国交省先進安全自動車推進検討会、内閣府総合科学技術会議社会基盤分野プロジェクトチームなどの委員。電子情報通信学会安全性専門委員会委員長。電子情報通信学会フェロー。

本日はシステム化・自動化の進展と人間の役割と題しましてお話します。先程の清野社長のお話にもありましたが、実は我々が狙っているのは1+1を3にすることですが、実は残念なことに1+1が0.8ぐらいにしかならないこともあるのです。システムにおいては、どのようにすれば人間と機械がチームとして本当に有効な役割を果たすことができるかが非常に大きなテーマだと思います。

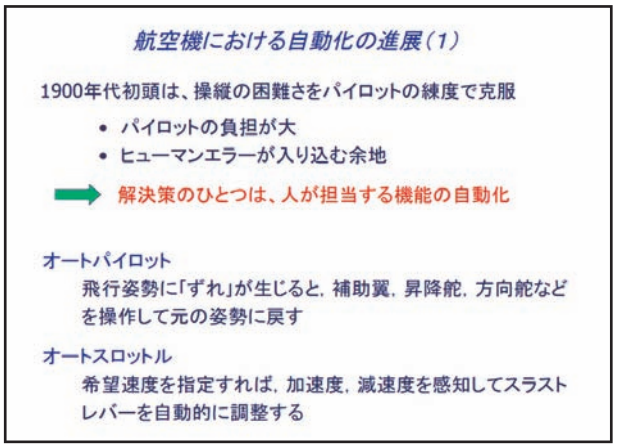
古典的なコクピット　－　Boeing 747

この図はボーイング747、いわゆるジャンボの古典的なコックピットです。一つのセンサーに一つのインジケーターが張り付いている形になっており、情報があちこちに散在してしまっています。特に、離着陸時にパイロットは100個程度の計器をずっとスキャンしながら飛ばしているという状況がありました。

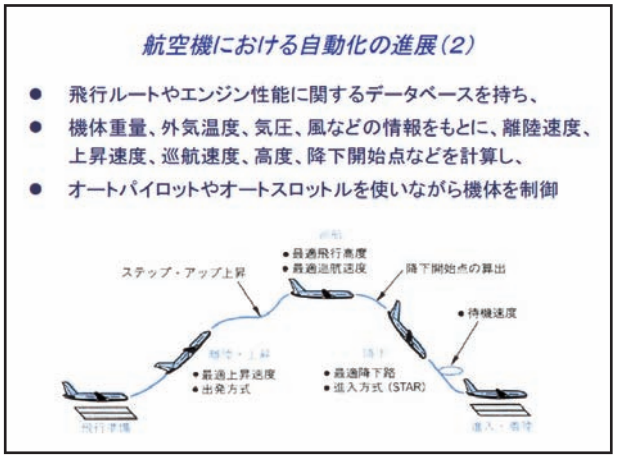
グラスコクピット　－　Boeing 747-400

それに対して、最近はグラスコックピットです。情報獲得のためのパイロットのワークロードを軽減するため、ディスプレイを使った統合型計器によって複数の情報を集約して表示しています。これはインタフェースのデザインとしては非常に大きな改善だと思われます。

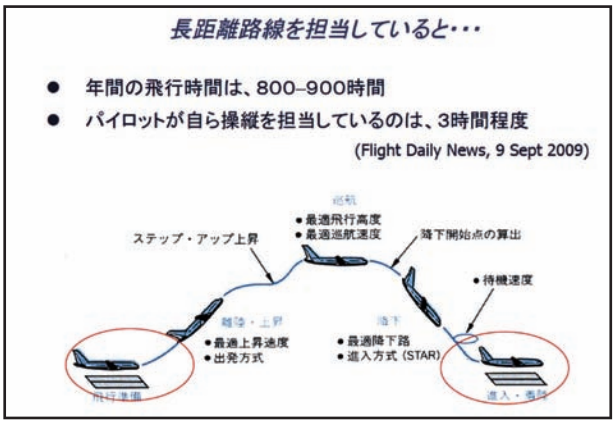
この傾向は現在も続いており、最近開発された機体、あるいは現在も開発中の機体では、ディスプレイ中心の情報表示技術にヘッドアップディスプレイを加えたり、カメラ画像を加えたりすることで、さらに多様なシーンが提供できるようになっています。



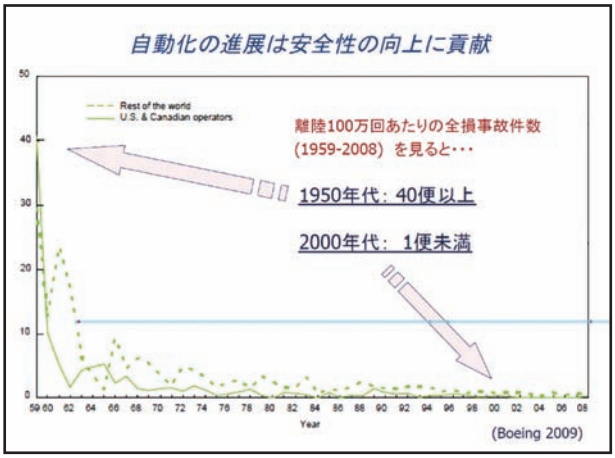
その航空機における自動化の進展ですが、航空機が登場した1900年の初頭あたりでは、その操縦は非常に難しかったと聞いています。この時代はそういう困難さをパイロットの練度で克服しており、パイロットの負担が非常に大きいものでした。この場合、当然、パイロットへの負担が大きいため、さまざまなところでヒューマンエラーが入る余地があり、実際にそのわずかな操作エラーが事故に至ったということもよく知られています。仕事そのものが困難ならば、ごく普通の順当な考え方として、その難しい仕事を自動化してしまうことをめざします。こうして登場したものがオートパイロット（航路にずれが生じたとき、それを自動的に直す）やオートスロットル（希望の速度を指定すればそれが実現される）です。



現在運航されている飛行機ではさらにその機能が優れたものになっており、1980年代後半以降に登場したグラスコックピットでは、その日の飛行機の重さ（お客さまや荷物）、飛行距離、外気温、風の吹き方などをすべて踏まえたうえで、離陸速度や上昇速度などをコンピュータが計算し、オートパイロットやオートスロットルを使用しながら機体を制御するようになっています。そうすると、離陸のところはパイロットが自ら操縦しますが、地上を離れて数百メートルあたりからはコンピュータに操縦させることが可能です。最終的には設備さえあれば着陸も自動で行うことができるようになっています。

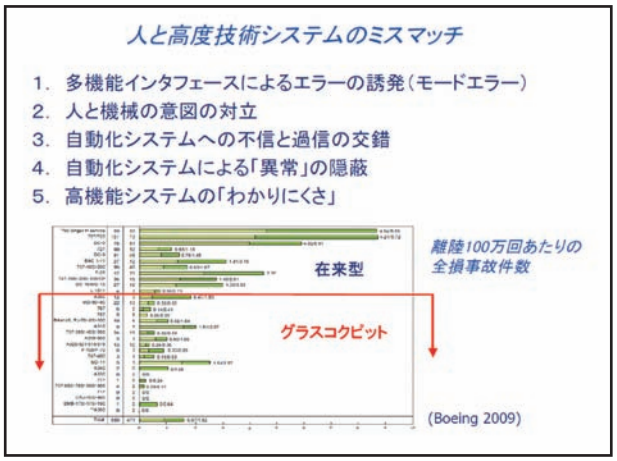


自動化システムが果たしている役割は、時間の視点からも理解することができます。Airbus社のトレーニング担当副社長の話では、長距離路線を運航している通常のパイロットの年間飛行時間は800時間から900時間ですが、その中でパイロットが自ら操縦を担当しているのは実は3時間程度だそうです。この3時間程度というのは、飛行準備から離陸・上昇初期までと、最終進入から着陸までの時間（上図の赤で囲まれた付近）の蓄積です。

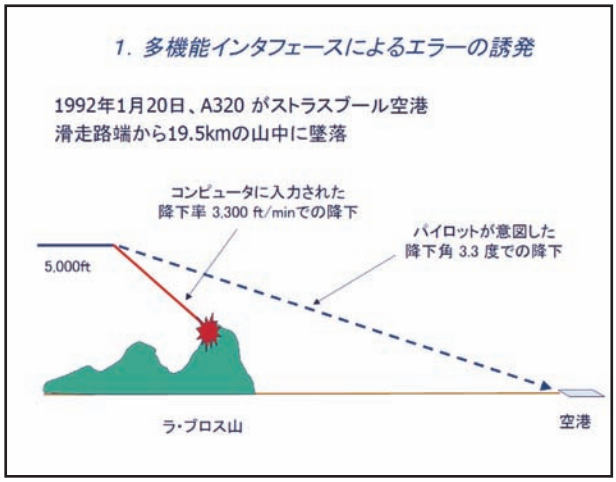


自動化がこのように進む中、当然、それが安全に貢献したかどうかということが気になります。上図はボーイングの統計ですが、縦軸が離陸100万回あたりの全損事故の件数で、100万回離陸した飛行機のうち何機が帰ってこなかったのかという統計です。1950年代では40機以上が帰ってこなかったことになります。それが

2000年代になると1便未満になっています。

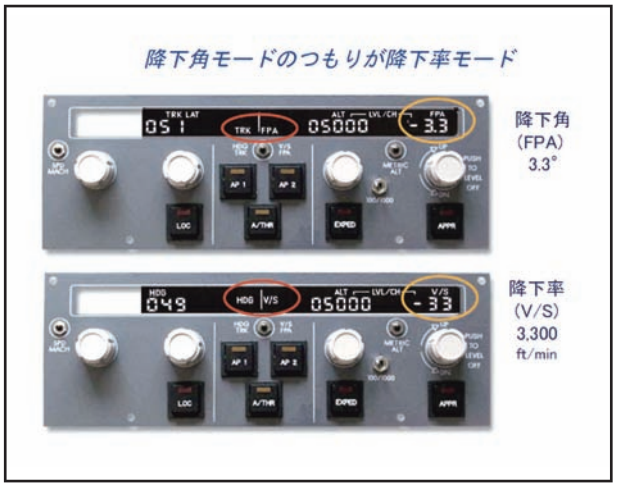


物事には多くの場合、光の面と影の面があるのと同様に、航空分野における技術にも実は光と影の両面があります。その影の面として上図を見てください。このグラフではバーが長いほど事故が多いことを示していますが、在来型のコックピットをもつ航空機がどれだけの事故を起こしているかが分かります。一方で、グラスコックピットの方が明らかに安全性に優れているとまでは言えないこともわかります。実は、グラスコックピットでは在来型の飛行機にはなかったタイプの事故が発生しており、懸念材料となっています。私自身がさまざまな事故を調べ、おおよそどのような特徴があるかを抽出したのが、上図の1番から5番になります。

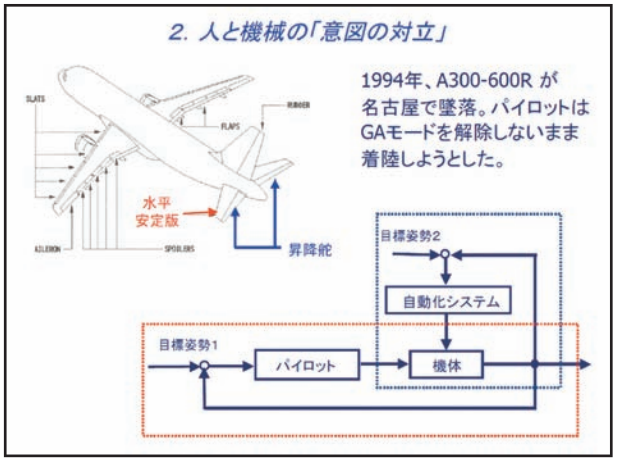


まず、一つ目ですが、多機能のインタフェースが実はヒューマンエラーを誘発してしまうというものです。多機能のインタフェースというのは、エンジニアが良かれと思って提供する非常に優れたもので、多様な使い方ができるようにコンパクトな空間にさまざまな機能を詰め込んでいます。そのような良かれと思ったことが実は裏目に出てしまったという実例ですが、エアバス320が上図の点線のパスに沿って降下角3.3度で降りようとしたところ、実は赤色で描いたパスで降りてしまい、事故になってしまったケースです。これは降下角3.3度を意図していたのに、実際には降下率3,300ft/minになってしまったというものです。

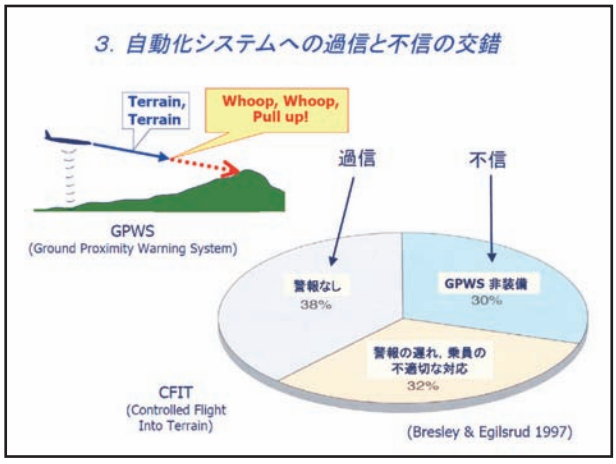
エアバス320のコックピットにおいて緑枠で囲んだ部分が人間とコンピュータがインタラクトするところです。この部分を拡大したものが次の図です。



この図の上半分は、パイロットが3.3度の角度で降りることをコンピュータに指示したときに現れる表示です。ところがコンピュータが受け取ったメッセージは、実は下半分の図に示されたものだったと考えられています。上半分の図の黄色の円で書いてあるところを見ると、フライトパスアングルとあり、降下角が－3.3度となっています。ところが下半分の図の方は、バーティカルスピード（V/S）で書かれていますが、これが3,300となっています。ここで、3,300が33と書かれている理由は、降下率の指定では10の桁も1の桁も意味がないので、「不要なものは表現しない」との趣旨によるものでした。そのモードが、降下角のモードなのか、バーティカルスピードのモードなのかは、確かに赤で書かれたところに表現はされていますが、実はそのモードは間違われることがあるのです。すなわち、モードはうっかり使うと困ったことが起こる可能性があるのです。また、「利用しないものは表示しなくてもよい」という考え方は、実は正しくないかもしれないということも意味しています。



二つ目の問題ですが、人と機械が互いに拮抗した操作をするという場面があるということです。例えば、名古屋の空港にエアバスA300-600Rが墜落した実例があります。このとき、機械が人間に反抗したのではなく、人間が与えた「Go Around」という命令を機械が忠実に最後まで実行しようとしたのです。人間は、「その『Go Around』は間違いであり、止めてもよい」ということを的確に機械に教えることができないまま、「私は降りたい」という意思で行動したのです。その状況下ではコンピュータは上がろうとする一方、人間は下がろうとするため、1つの機体を2つの意図を持ったものが同時に制御した形になりました。これは非常にシンプルなケースですが、この意図が対立するように見えるという事象は、人と機械で考え方が異なるとより複雑な場面で出てくるのです。私は、自動車のこれからの開発の中で非常に大きな問題となるだろうと考えています。



三つ目の問題ですが、自動化システムに対して人は過信も抱きますが、同時に不信も抱くため、過信だけが問題なのではなく、実は過信と不信がお互いに交錯するような形でいろんな問題が出てくることがあるのです。

航空機には、このまま進むと地表面にぶつかるという時に、あらかじめ警報を出す対地接近警報装置GPWSが必ず装備されています。上図では、高度がどのように移り変わっていくかをモニタし、降下率が大きすぎる場合、警報が出るというタイプの非常に古典的なものを示しています。このGPWSは非常に役に立つ装置なのですが、実は機能に限界があります。高度の変化だけしか見ておらず、平坦なところが長く続いて突然目の前に絶壁が出てくるという事象に対してまったく

無力となります。そのため、パイロットが、「危なくなったら警報が出るだろう」と思っていたとしても、このような場合には警報は出ないのです。

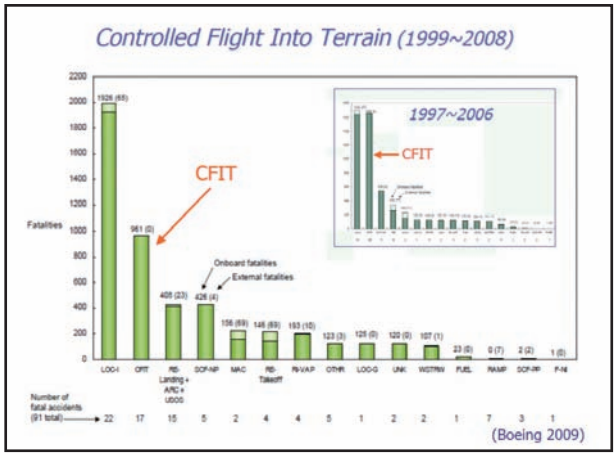
一方、地形がある一定の条件を満たすと、危険でないにもかかわらず警報が出ることがあります。つまり、実際にGPWSは誤報も発生させるし、また、欠報も発生させてしまいます。

まったく異常がない機体が山に衝突するというタイプの事故をCFIT（Controlled Flight Into Terrain）と呼んでいます。制御された地表面へ向かっての飛行という意味で、皮肉を含んだ嫌な表現になっています。CFITを起こした飛行機がどのような原因で事故に至ったかを調べた方々がおり、その結果では、3分の1ずつ、GPWS非装備、警報の遅れ・乗員の不適切な対応、警報なしとなっています。

警報がないという形で事故になったケースでは、パイロットが何もしていないということであり、異常時に警報が出るだろうと思っていたところ、何も警報が出なかったということで、この事象ではパイロットがGPWSに対して過信をしていたと言えます。

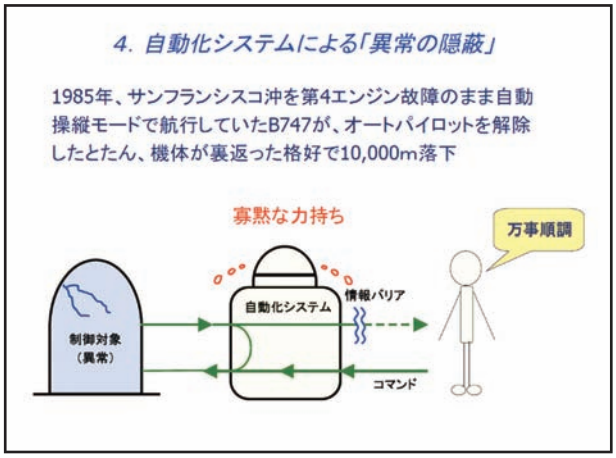
また、GPWSを装備していない飛行機は、当時、世界中で3%しかありませんでしたが、その3%しかないGPWS非装備の飛行機が30%のCFITを起こしているということは非常に重要な意味を持っています。非装備になった経緯は、「GPWSなんて搭載しても仕方がないですよ」というGPWSに対する不信感の表れでもあります。

次に、パイロットがGPWSに対応しないことがあり得ることをビデオで紹介します。この事例では、8回程度GPWSが鳴っているにもかかわらず、パイロットは、「自分は安全な高度を飛んでおり、管制官に言われたところまではまだ降りきっていない」と勘違いをしてしまっています。図の上方に2400フィートのラインが表示されていますが、管制官は「2400 (two thousand four hundred)フィートまでは降りてもいいですよ」と言っています。ところが、パイロットはtwo thousandを聞き落とし、「400まで降りてよい」と言われたと勘違いしたのです。そのため、「こんなところで警報が鳴るはずがない」と思い、ランディング・ブリーフィングを続けています。その中でGPWSの警報が鳴っても、パイロットは「何でこんなところで鳴るのですかね？」と思い、対応しないということがあり得るのです。パイロットは、山のわずか手前で初めて「GPWSが危ない」と正しく伝えていてくれたことに気づいています。

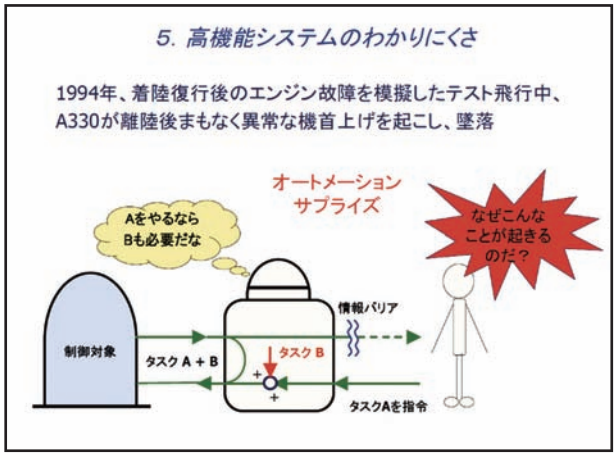


上図は、飛行機がどのような形態で事故を起こし、死者を出してきたのかを示しています。一番左の一番長い棒は、制御できない機体が落ちてしまったという

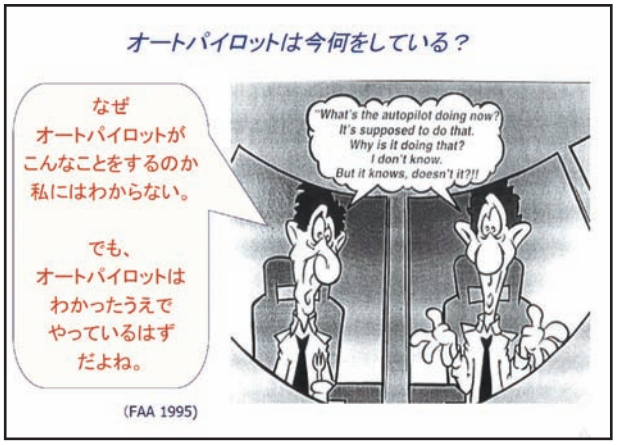
事故による死者数です。このような事故は、機体が制御できない以上、いたしかたない面があります。しかし、左から2番目の棒はCFITによるもので、機体そのものはどこにも問題がないにもかかわらず、どこを飛んでいるのか、どの高度を飛んでいるのかなど状況を的確に認識できていなかったために衝突してしまった事故となります。この場合、コックピットのボイスレコーダーを調べると、最後まで平穏な会話しか出てきません。最後の数秒だけが「あっ」というような感じで、会話の緊迫感が上昇するものの、その時はもはや手遅れとなってしまっています。まったく異常のない機体がこのように墜落してしまうのはたいへん虚しいものです。このボーイングのデータに基づいたグラフは、10年間でどれだけの死傷者がどのタイプで出たかということを表していますが、右上の図は2年分、前方にシフトした10年間を対象にして同様のグラフを示しています。2年前方にシフトすると、CFITがさらに多かったということが分かります。CFITがどのようにして現在の数字まで減ったかについては、後述の機能強化型GPWSの貢献があるのではないかと考えています。



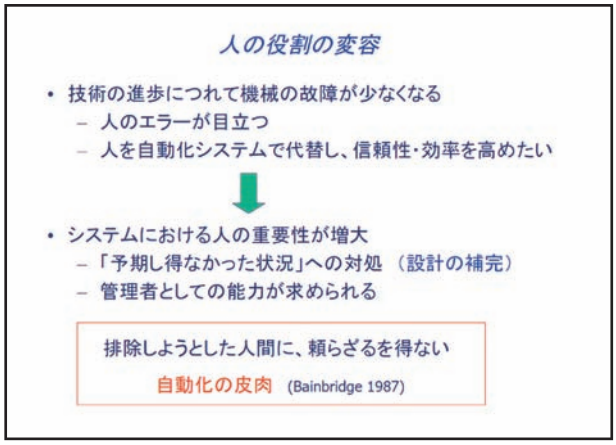
四つ目の問題ですが、自動化システムは非常に寡黙で「何をやれ」と言われても絶対文句を言わずに、どれだけ長い時間かかっても一生懸命頑張ります。どんな汗をかいていてもコンピュータは絶対に人には不平を言わず、寡黙で力持ちなシステムになっています。そのためコンピュータが、何をやっているのか、どのような状態にあるのかということを、もし人間がインタフェースを通じて、的確に把握することができなければ、実は異常があるということすら気がつかないということになります。これは、サンフランシスコ沖を飛んでいた飛行機がコンピュータによりかろうじて姿勢を保っていたところ、パイロットがオートパイロットを解除してしまい、その瞬間に機体が裏返って1万メートル落下したという、典型的な事例の中にも見ることができます。パイロットは、コンピュータがどれほど大きな力でようやく機体を支えていたかということを分かっていなかったのです。



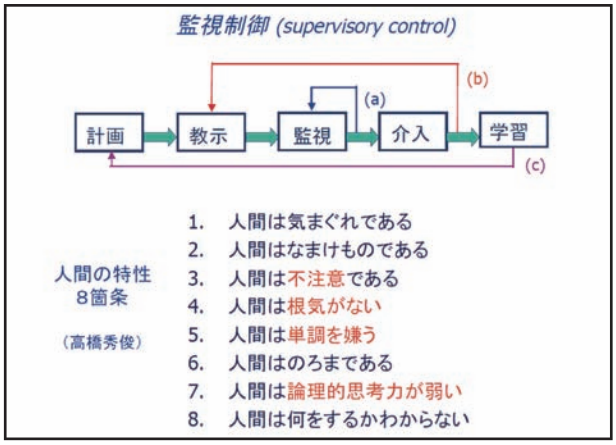
五つ目の問題は、「高機能システムのわかりにくさ」ですが、非常に厄介な問題だと思っています。いまのコンピュータは、高い知能をもって自分の判断を自分の手足を使って実行することができる能力を持っています。その能力により、人間にとって非常にわかりにくい行動をする、あるいは、人間から見てそのように思えることがあります。例えば、人間がAという仕事をやってほしいとコンピュータに指示をすると、コンピュータは非常に気が利くので、「Aを行うのならば、一緒にBもやってあげよう」と考え、AとBを同時にするということがあります。AとBを同時に行うと、Aを行った時にどのような現象が見られるのかに関するパイロットの予測と違った現象が出てきます。そのとき、「何でこのようになっているのだ、なぜこんなことが起こっているのだ」となり、現象に対して理解ができないのです。自動化システムの良かれと思った行為が、パイロットをびっくりさせてしまうという意味で「Automation Surprise」と表現しています。これはトレーニングを十分積んでいるパイロットにすら起こり得るため、トレーニングを十分積んでいない人を対象とする自動車ならば、深刻な問題になる可能性があると考えています。非常に知的であるがゆえに、コンピュータが分かりづらいことをすることがあるのです。



上図はFAAによって作成されたCFITを防止するためのマニュアル教材の中に表れているマンガの１つです。コンピュータが操縦を行っており、二人のパイロットは共に操縦していないのですが、「コンピュータはなぜこのようなことをやっているんですかね。私には全然よくわからないけれども、コンピュータは分かってやっているはずだよね？」と言ってそのままにしているという状況です。これは「高機能システムの分かりにくさ」ももちろん表していますが、コンピュータが分かりにくいことを行っている時に、人間はコンピュータに対して、「任せておいて大丈夫だろう」という気持ちを抱いてしまうことがあることを示しています。これはある意味で過信であり、「自分がわからないことを相手が行っていることに対しても、良かれと思って行っているのだろうから問題ないのでは」という解釈をしているのです。このような過信に対しても、これからの人間と機械との役割を設計していくうえでは非常に重要な問題になっていくと思われます。



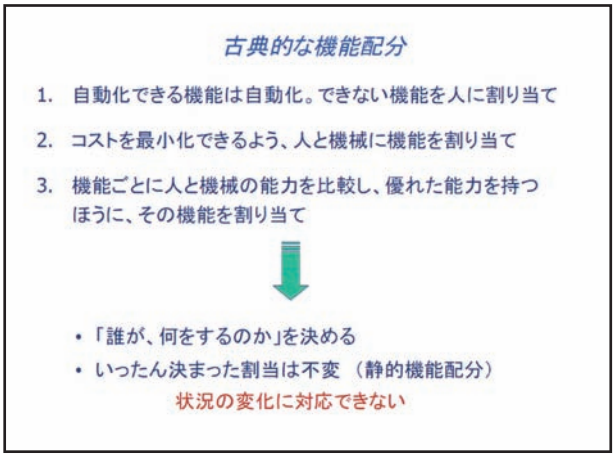
人と機械をどのようにして協調させるかという、非常に初歩的な段階に振り返って考えてみます。技術が進歩するにつれて機械の故障がだんだん少なくなり、信頼性が高くなります。すると、当然の流れではありますが、相対的に人のエラーが目立つようになり、「エラーをするような人はシステムから排除してしまえ」ということが考えられました。人の作業を自動化システムで置き換え、信頼性・効率を上げたいという趣旨で「工場も無人化しましょう」と、自動化を積極的に推進してきた時代がありました。ところがその結果、システムの中から人がいなくなったわけではなく、むしろ、システムの中における人の役割が昔に比べてはるかに重要なものになってきたのです。システムは一旦作られれば、例えば、10年あるいは20年と使われていくものです。10年や20年の間に一体どのような場面に遭遇するかをあらかじめ設計の段階ですべて網羅し、それに対する対策をすべてとることは、ほとんど不可能です。これはエンジニアとして絶対不可能と断言してもよいくらいだと思います。そうすると、「デザインで対応できないような場面が出現した際は、システムの中にいるオペレータに頼らざるを得ない」ということになります。つまり、システムの中にいるオペレータは、デザイナーの仕事の残りを補完するという重要な役割を果たすことになります。さらに管理者としての能力も求められます。したがって、初期の時代ではどちらかといえば人間は労力を提供するのに対し、現在の形態では、人間が知力を提供するという形になっています。排除しようとした人間に最終的には頼らざるを得ないという形態が現在のシステムであり、これを「自動化の皮肉」ということがあります。



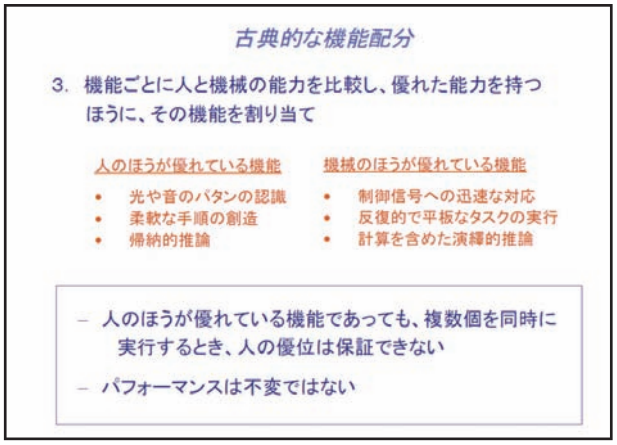
一方、現在のシステムの多くは、監視制御というモデルで表現することができます。現在のシステムでは、人間が何をするかについて計画し、それをプログラムに書いたりボタンを押すなどコンピュータに教えることで、コンピュータが動作を始めます。長い時間にわたってコンピュータが稼動し、自動化システムが動いている様子を常に人間がモニタし、もし何か異常があれば、直ちに人間が対応をとらなければならないのです。上図の(a)というループは人間がモニタしている時に、何か変だなと思ったら、別のところをいろいろ監視しながら、一体何が起こっているかを理解しようとするループです。そして継続が困難だと判明すれば、自動化のシステムを一旦解除する必要があります。そこで人間の介入が必要となります。介入後、新しく必要になった行動を改めてコンピュータに教え込まなければならないのです。

上図の1番から8番は、高橋秀俊先生が「人間の特性8箇条」としてお書きになったものです。高橋先生は東京大学の物理学の先生で、日本の情報科学の草分けとなった著名な先生です。

「1.人間は気まぐれである」、「2.人間はなまけものである」、「3.人間は不注意である」、「4.人間は根気がない」「5.人間は単調を嫌う」、「6.人間はのろまである」、「7.人間は論理的思考力が弱い」「8.人間は何をするかわからない」と列記されており、スライドの赤字部分はまさに監視制御に求められているものの対極にあります。したがって、私は監視制御、つまり現在のシステムは、人間の特性に合っていないのではないかという気がするほど、人間にとって非常に過酷なシステムだと思っています。例えば、通常は何事もなく継続され、いつ異変が起こるということもほとんど分からないような状態で、いつ起こるかもしれない異常を見つけるためにずっと画面を凝視していなければならないのは、非常につらい仕事だと思うのです。これが現在の監視制御であるならば、これからは人と機械がもう少しうまく協調し、「1+1が3になる」ようなシステムを築くためにも、人間と機械はどのように役割分担をすべきであるかということを考えていかなければなりません。

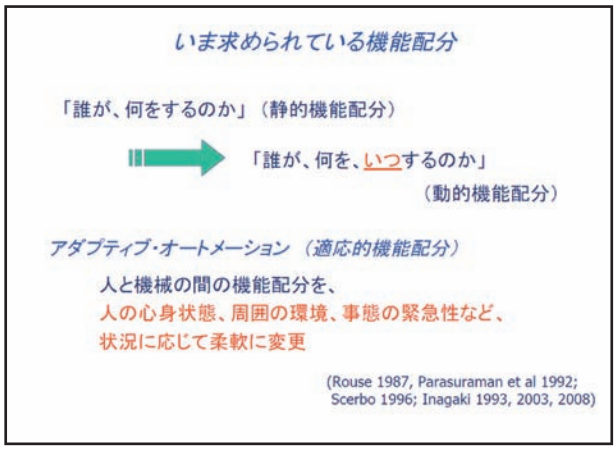


古典的な機能配分には3つの方式があります。機能配分とは人が何を行って機械が何を行うのかを決めるということを表わします。一つ目の方式は、自動化できる機能は全部自動化してしまい、できなかったところは人が担当するというものです。二つ目は、コストを最小化できるように、人と機械に機能を割り当てようというものです。例えば、自動化しようと思えば自動化が可能なのですが、お金がかかってしまうので、それをするくらいなら、人を一人雇った方が安いという場合です。人と機械はほとんど同列に扱われてしまい、あたかも人は機械の一部であるかのように扱われてしまいます。これを「技術中心の自動化」、つまり、人のことは何も考えないで技術の観点からのみで作られたシステムです。どのようなシステムを作れば、一番作りやすくて一番安価であるかという考え方でできています。

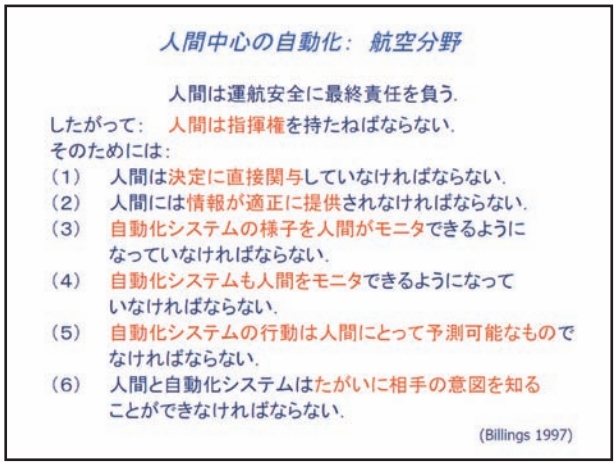


それに比べ、少々よい方法が実は古くからありました。それぞれの機能ごとに人と機械のどちらがその機能において得意なのかということを考慮し、得意な方に割り当てるという方法です。上図の赤字部分がその例ですが、人が優れている機能にはどのような機能があり、機械が優れている機能はどのような機能があるのかということをリストアップするのです。これはもちろん部分的なリストで完璧なものではありません。このリストのなかで優れている方に仕事を割り当て、人は人が優れている仕事だけを行えばよく、人にとって幸せなように見えますが、実はそうではありません。例えば、人の方が優れている機能であっても、複数作業を同時に実行できないからです。2つくらいならば同時にできるかもしれませんが、3つ、4つ、あるいは5つ同時にやってくださいと言われてもできないかもしれません。また、仕事を行っているうちに、人はだんだん疲れてきます。人のパフォーマンスは不変ではないという問題も出てきます。

古典的な機能配分の3つの方法は、基本的に「誰が、何をするのか」ということを決めるためのものであり、一旦決めた割当ては不変で永久に変わることはありません。不変という意味で静的な機能配分となりますが、状況変化に適応できないというのはすでにご推測のとおりです。



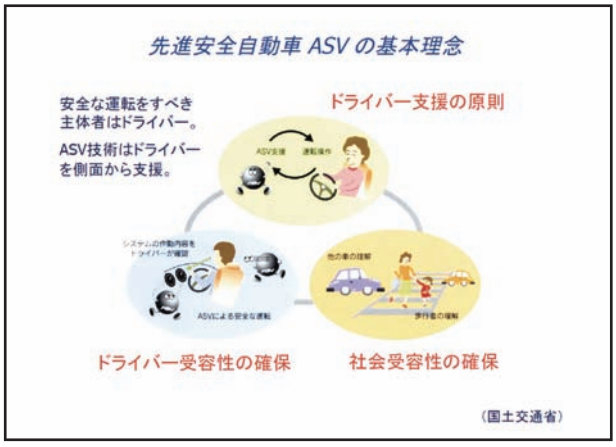
上述のように、「誰が、何をするのか」という決め方、つまり、静的な機能配分ではよくないということが分かります。いま求められていることは、「誰が、何を、いつするのか」を決めることです。これは、時と場合によってその役割分担が変わることを前提としています。つまり、人が行っていたところを機械が行う、あるいは機械が行っていたところを人が行うことがあり得るのです。これを動的な機能配分と称しますが、その中でも特に我々が求めているのはアダプティブ・オートメーション、あるいは適応的機能配分と呼ばれているものです。これは人と機械の役割分担を、人の心身状態、周囲の環境、あるいは事態の緊急性などを加味し、状況に応じて柔軟に変えるというものです。実際には、古くからアイディアはあったのですが、実はこれが可能になったのはごく最近のことです。いろいろな先進技術が利用できるようになって初めてこれが実現できるようになりつつあります。

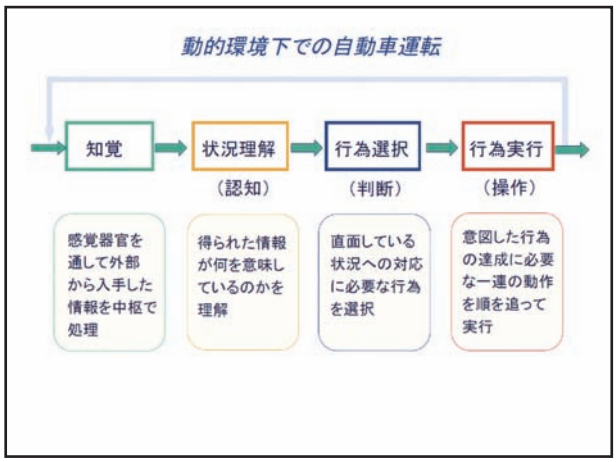


ただし、まだ完成しているわけではありません。その中で最も参考になることは、航空の領域でも人間と機械の役割分担について、1980年代から研究されていたということです。つまり、自動化が盛んに導入されていった航空の領域で考えられてきたことになりますが、現在は、「人間中心の自動化」というタイトルのもとで、ある程度の形態がまとまっています。簡単にお話すると、「人間は運航安全に最終責任を負っている。したがって、最終責任を負う人に対しては全権限を与えておかないといけない」ということがポイントになっています。「人間は指揮権を持たねばならない」という表現で書かれていますが、人間に最終決定権を与えるという言い方をすることもあります。

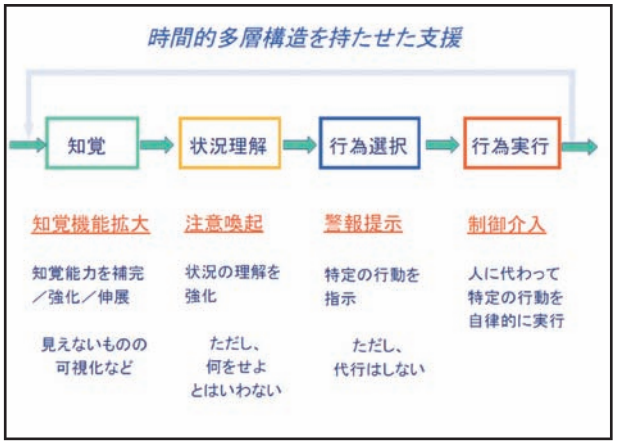
上図の1～6番までに、これを実現するためにはどうすればよいのかということの概略があります。決定権をもつ以上、決定に直接関与するのは当たり前ですが、情報が適正に提供されない限りその決定権は行使できません。自動化システムの様子を人間がモニタできるようになっていて、「危ない」と思ったら、直ちにそのようなことがわかるようになっていなければならないのです。ところが4番目になりますと「自動化システムも人間をモニタできるようになっていなければならない」とあり、少々異質なものとなっています。5番目では、「自動化システムの行動は人間にとって予測可能なものでなければならない」とあり、これは上述でいえば、Automation Surpriseをなくさなければならないということになります。6番目では、「人間と自動化システムはたがいに相手の意図を知ることができなければならない」とあり、チームを構成しようとしても、相手が何をやろうとしているのか分からないようではチームは組めない、ということを意味します。

航空の領域では、人間中心の自動化という概念でこのように動いているのが現在の状況ですが、実は、同様のことが現在、国土交通省でも進められています。国土交通省および、自動車メーカーが1991年から取組んでおられる「先進安全自動車（ASV:Advanced Safety Vehicle)」の基本理念の中にも同じようなことが述べられています。大きく3つにまとめられた基本理念の原則として、「ドライバー支援の原則」、「ドライバー受容性の確保」、「社会受容性の確保」があげられています。いま、人間中心の自動化の観点から一番大きな関連性があるのは「ドライバー支援の原則」です。安全な運転をすべき主体者はドライバーであり、決して機械ではないということがうたわれています。機械はあくまでも側面から支援するという位置づけになっています。実はこの部分が非常に大きな問題になることがあります。





例えば、自動車を運転することを想定してください。おおよそ人間の頭の中では4つのフェーズで情報処理が行われています。まず自分の感覚器官を通じて外から情報が中枢に入り、その中枢に入ってきた情報がいったい何を意味しているのかを理解します。これが2番目の状況理解です。認知という言い方がされることもあります。つぎに、その状況下で私は何を行わなければならないのかを考えますが、それが行為の選択ということになります。その行為を選択し、実際に一連の動作を順に追って実行していくという形で行為の実行がなされます。



この4つのフェーズの中で、実際に機械がどのようにサポートすればよいのかということが考えられています。例えば、知覚をサポートしようとした場合、人には見えないもの、あるいは見えにくい物を見えやすくするというのが典型的です。状況理解では、「これには注意をしていてくださいね」といったことを人間に教える注意喚起があります。次に、ある状況で本当は行わなければならないことがあるにもかかわらず、人間の行動が少々遅れている時は、「これを行った方がいいですよ」など、ある特定の行為を明確にして人間に行動を指示する、警報が必要となります。最後に、指示しても人間が対応できない場合に、自動的に行動を代替して実行するというのが制御介入です。

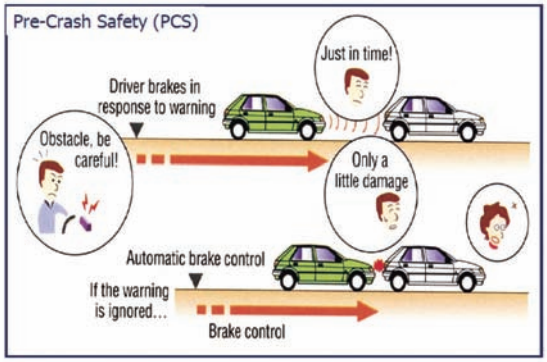
知覚機能拡大

実例ですが、夜間走行中に小動物がいるのにもかかわらず、人間の目にはよく見えない場合、センサーを使ってドライバーの目の前にあるディスプレイにそれを強調して表示しようというシステムも現在は実現できています。これは知覚機能の拡大の例です。

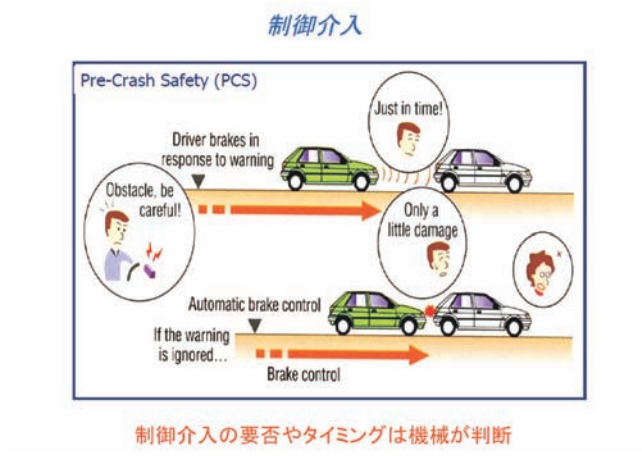
注意喚起

同じようなケースですが、前方に歩行者がいるのですが、画面でも見えないとおり、運転者は認識できていません。その状況下で、例えば「ディスプレイの中に、歩行者がいます」ということを明確にして、枠で囲って表示しています。これは、「ここに注意して下さい」ということを明確に言っており、「歩行者がいることに注意して下さい」という意味で注意喚起になっています。しかし、何を実行しろということを指示しておらず、警報ではありません。

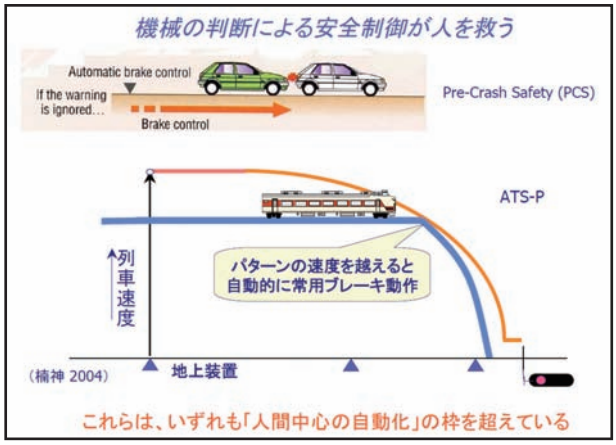
警報提示



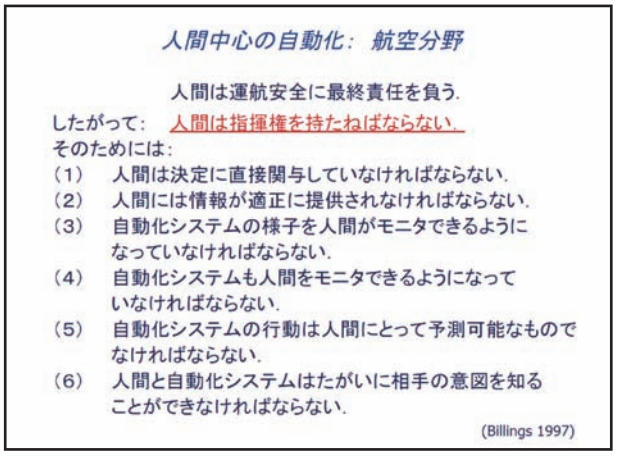
警報が出てくる典型的な例ですが、前の車に対して自分が運転している車が接近しすぎるとき、「ブレーキを踏め」という意味の警報が出ることがあります。自分が加速しているというより、前の車が減速して自分に近づいているようなケースを想定してください。そこで運転手が対応すれば、コンピュータはそれ以上出る幕はありません。



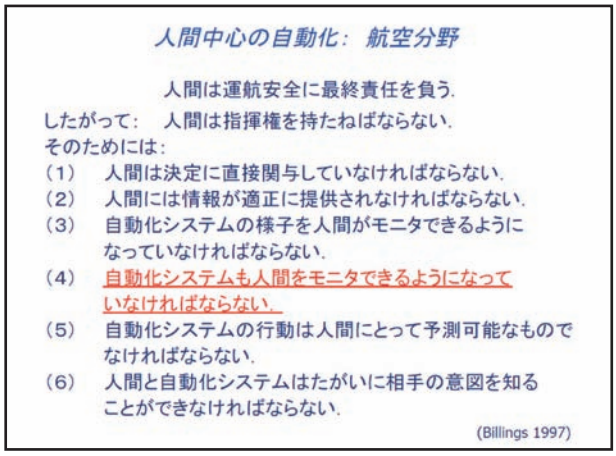
ところが、警報を出したにもかかわらず、まだ人間の行動が見られないというとき、衝突を避ける、あるいは被害を軽減しようとするなら、自動でブレーキをかける必要があります。その「自動でブレーキをかける」ことを実現したものがPre-Crash Safetyと呼ばれているシステムです。ここで重要なことは、制御介入が必要かどうか、あるいは介入するとすればいつ介入するのかというタイミングを人が決めておらず、機械が決めているということです。機械は人に「いま介入していいですか」ということも聞きません。何も聞かずに機械は実行しているのです。



機械の判断によって安全が確保される、機械の判断による安全制御は人を救うという側面は、Pre-Crash Safetyシステムだけではなく、鉄道にもあります。鉄道のATS-Pというタイプでは、停止信号からの距離に応じて決められた最大速度のパターンが描かれ、そのパターンにぶつかれば、自動的にブレーキがかかるというシステムになっています。ブレーキをかけるかどうかの判断はすべて機械が行っています。つまり、安全確保のためには、人間中心の自動化の枠を超えることが必要となる可能性があることを理解しておく必要があるのです。

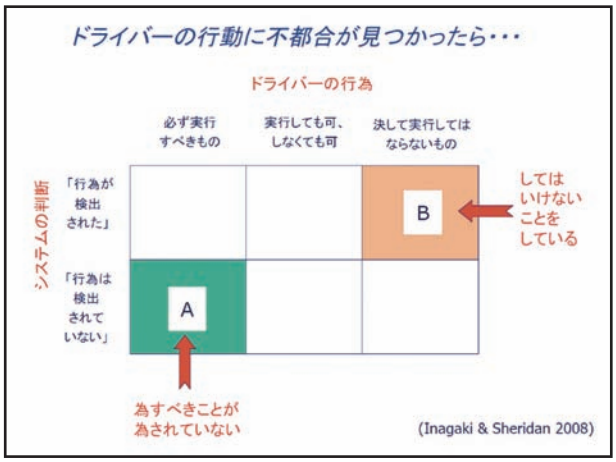


人間中心の自動化で、「人間は指揮権を持たねばならない」ということを述べましたが、いまのケースの場合、人間は指揮権を持っておらず、何も命令は出していません。それでも機械はそれを実行することがあるということです。ある意味で、Pre-Crash Safetyシステムは人間中心の自動化に反している面があるということに注意いただきたいと思います。

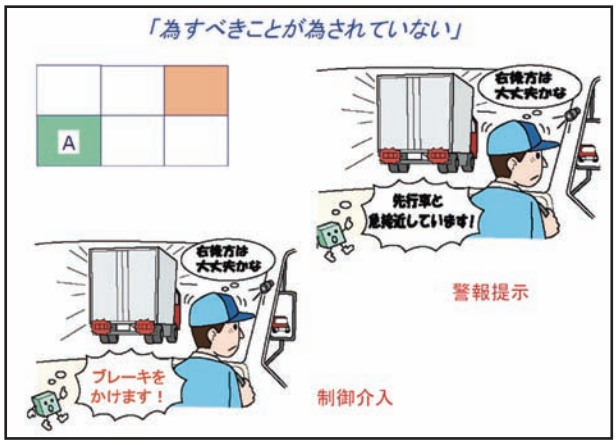


次に、4番目の「自動化システムも人間をモニタできるようになっていなければならない」ということを、「人間中心の自動化」のなかでも、他のものに対してやや異質なものとして紹介します。「自動化システムはなぜ人間をモニタする必要があるのか」ということを考えてみます。ヒューマンエラーには、オミッションとコミッションがあります。オミッションは「行うべきことを行っていない」、コミッションは「行わなくてもよいことを行っている」ということです。1つの解釈とし

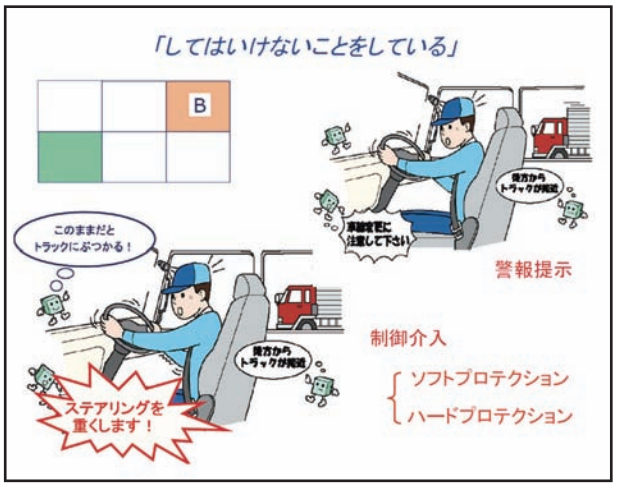
て、この4番目は、人間がオミッションしていたり、あるいはコミッションしていたりすることがないかということを機械でモニタする必要があるのではないかということです。また、もしそれが見つかれば、何らかの手を打つ必要があるだろうというものです。



例えば、自動車での領域で、ある状況において、ドライバーが絶対に行わなければならない行為と、絶対に行ってはいけない行為の2種類があることはすぐに分かります。ところが、行ってもよいし行わなくてもよい、どちらでもよいという行為があり、実は3通りに分けることができます。ドライバーの行動をモニタしている機械があると仮定した場合、ドライバーが何の行動をしているかを検出することができます。例えば、コンピュータが、ドライバーのある特定の行為を検出した、あるいは検出されていないという判断を下すということがあります。その場合、全部で6通りの組み合わせができます。我々が考えるべき項目は、このうちのAとBで書かれたところです。例えば、「必ず行わないといけないことを確かにその人は行っている」とコンピュータが認めているという場合、何もする必要がありません。ところが、「行わないといけないことを行っていない」とコンピュータが判断している場合、どうすればよいかということが問題になります。Aは、このように「なすべきことがなされていない」とコンピュータが判断した場合であり、一方、Bは「行ってはいけないということを人は行っている」とコンピュータが判断した場合です。



上図のようなとき、コンピュータはどうすればよいのでしょうか。これは易しい問題ではありません。まず、「なすべきことがなされていない」というAの場合を考えてみます。例えば、前にいる大きなトラックを追従しているドライバーがいたとします。また、そのドライバーが一体何をしようとしているのかについてコンピュータがセンシングでずっと見ていると仮定します。このドライバーはサイドミラーを頻繁に見ており、前を見ていないということをコンピュータが認識したとします。その時、先行車が減速してきたならば、警報を出して、「先行車が減速していますからブレーキかけてください」と伝えることができます。もう1つのやり方として、警報はもちろん出しますが、警報を出した直後にブレーキを自動的にかけてしまうという方法もあります。この場合、警報を出すだけの警報提示の方がよいのか、あるいは自動ブレーキを含めてコンピュータが自律的にブレーキをかける制御介入の方がよいのかがポイントとなります。人間中心の自動化という観点からすれば、厳密には警報提示の方が適合していると言えます。ところが我々の行ったさまざまな実験によれば、実は事故を回避しようと思うならば、制御介入がどうしても必要であるという場面が出てきます。このようなとき、人間が命令を下していないにもかかわらず、コンピュータが何かを行うということに対して、人間はどう思うかということも調べる必要があります。それが許容できるのか、できないのか、そういうことはあってもよいのか、自分としてはそれを認められるかという受容性を調べたところ、実はこのケースに対してそれほど人は拒否感を感じないということが、我々の実験の範疇では分かりました。



一方、Bの領域ですが、例えば、車線変更をしようとしてドライバーが右にハンドルをきろうとしたにもかかわらず、コンピュータが「後ろから車が来ているのだからいまハンドルを切ったら駄目だよ」と判断する場合を考えます。この場合、「後ろから車が来ていますからハンドルを切らないでください」といった警報を出すだけでよいか、あるいはハンドルが切れなくなるようにする、例えば、ハンドルがすごく重くなる、あるいはハンドルを切っても車はまっすぐ動くようにするという選択があります。後者については、新しい「バイ・ワイヤ」の技術を使えば、ステアリングの動きと車輪の動きを独立させることができます。例えば、ドライバーが右にいくらハンドルを切っても、車はまっすぐに進むということが実現可能となります。この選択に対して、さまざまな形態を作り、シミュレーターで実験しました。その結果、制御介入を行った方が事故は減るということが分かった場面でも、その制御介入を嫌うドライバーが多いことが分かりました。「警報だけにしてほしい」、「制御介入までされるのは嫌だ」、「ハンドル操作までやってほしくない」といった意見が多かったのです。制御介入が安全であるにもかかわらず、やはり機械に権限を奪われるということに対する違和感が随分ありました。この結果から言えることは、安全の確保をしながら、どうやってドライバーに認めてもらえるものを作るかということが、非常に大きなポイントになるということです。

我々は、制御介入を行うとしても、少なくともソフトプロテクションとハードプロテクションの2通りのやり方があると考えています。例えば、ソフトプロテクションでは、本当はハンドルを切ってはいけないという時にハンドルを切ろうとしても普段よりもずっとハンドルが重くなるという制御があげられます。この制御では、ハンドルをきろうとした時、「何かおかしいな、これは切ってはいけないということかな」とドライバーが感じることにより、ハンドルをきることを思いとどまってくれるドライバーがいるかもしれません。しかし、前に飛び出してきたものがあったりしてどうしてもハンドルを切らないといけない時には、ドライバーが大きな力をかければ、コンピュータによって重くされているハンドルをオーバーライドすることができます。このように、最終的にはドライバーの判断を優先するというデザインがソフトプロテクションとなります。一方、ハードプロテクションは、「バイ・ワイヤ」の技術を使って絶対にハンドルは切れないようにするなどの制御があげられます。例えば、航空機では、少々単純な図式になりますが、分かりやすく画一的にいえば、ソフトプロテクションがボーイング型で、ハードプロテクションがエアバス型だと考えて差し支えないかと思います。

このように、人と機械との間で安全を確保しながら、その人にとって使いやすいシステム、あるいは、受け入れられやすいシステムにしようと思った時、権限の問題がたいへん重要な役割を果たします。その権限の問題は、実は人間中心の自動化の中でまだ十分には検討されていない部分があり、我々としてはまだ多くの検討課題が残されていると考えています。

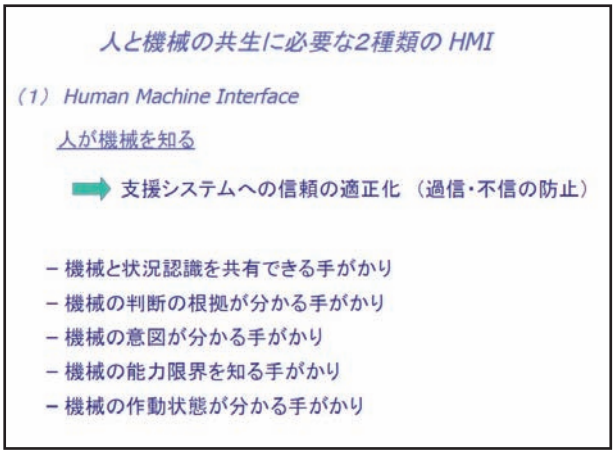
これまで航空機と自動車を中心にして述べてきましたが、我々の身の回りには、交通移動体としてさまざまなものがあり、鉄道や船舶もあります。一般的に、「人間中心の自動化」という言い方をされることが多いですが、実は「××の人間中心の自動化」と言わないと正しくないと私は考えています。つまり、「航空機のた

めの人間中心の自動化」は、「自動車のための人間中心の自動化」とは等しくないということです。同様に、「鉄道のための人間中心の自動化」、「船舶のための人間中心の自動化」もそれぞれ異なるのです。「人間中心の自動化」と言いながらも、どこで交通移動体に依存する部分が出てくるかは、おおよそ3つあります。

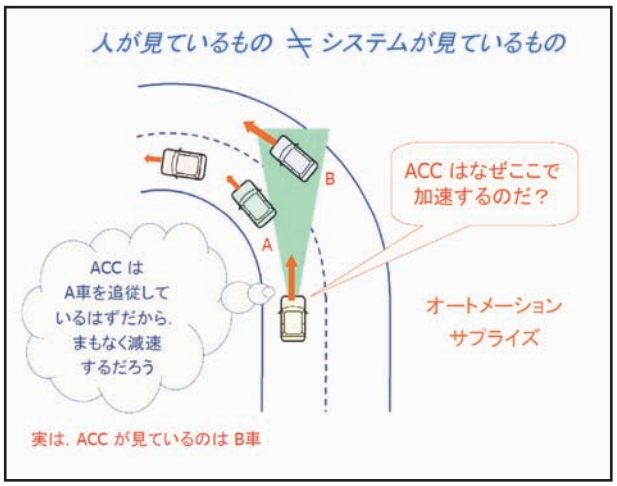
1つ目は、オペレータに知識あるいは技量があることを仮定することができるか、ということです。例えば、航空機のパイロットはトレーニングを十分積んでおり、しかも継続的にトレーニングが行われます。自動車の場合だと、一旦免許を取得すれば、余程のことがないかぎり、「もう一回教習所に行くように」と言われることはありません。また、高度なシステムが自動車に積まれても、おそらくドライバーはマニュアルを読まないでしょう。すなわち、システムの限界、あるいはシステムの特徴を全部理解したうえで初めてシステムを使うという自動車のドライバーはおそらくいないでしょう。このようなことがオペレータの知識とか技量の差となります。つまり、「自動車の自動化」を考える際、航空機のものとは全然違ったものを考えなければならないということになります。

2つ目は、操作開始までに許される時間余裕が違うということです。これは警報が出た時を想定すれば分かると思います。山にぶつかって行く前に警報が出るというGPWSではおおよそ20秒、あるいは30秒前に警報が出ます。また、空中で2つの飛行機が衝突しそうになるかもしれないというときに、片方の飛行機には「上がれ」という警報が、もう片方には「下がれ」という警報が出ます。この警報はレゾリューション・アドバイザリと呼ばれています。このような場合でも、警報が出てから5秒以内に操作を開始すれば、十分回避できるようなタイミングで警報が出されます。このように、確かに航空機は高速度の移動体ですが、時間的にはある程度ゆとりがある乗り物です。それに対して自動車では、「このまま進むと前の車に衝突しますよ、ブレーキを踏んでください」という接近警報が出て5秒程度経過した頃、「あ、ブレーキを踏むのか」とおもむろにブレーキを踏んでいるようでは、もうすでに衝突した後となってしまいます。すなわち、飛行機と自動車で時間の余裕が全然違うのです。

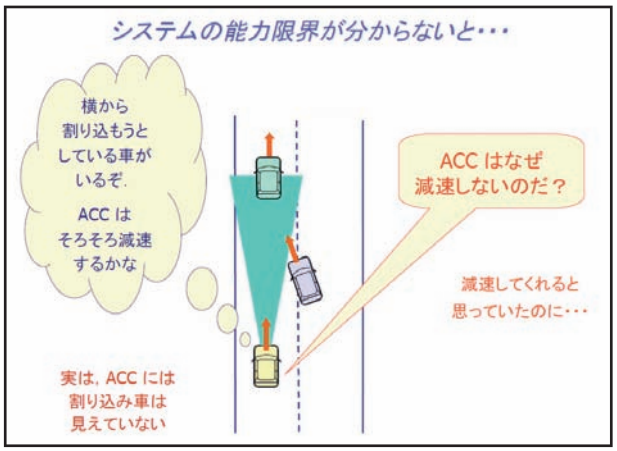
さらに、移動体によって応答特性も異なります。例えば、船舶では「あて舵」をうまく行わないといけないということがあり、私も体験したことがありますが、船舶をうまくコントロールするのはたいへん難しいです。相当予測をしたうえで操作をしないと、とても操縦できるものではありません。飛行機にも同様の側面があると思います。一方、鉄道では、例えば、ドライバーが前方に障害物があると認識し、ブレーキをかけてももはや間に合いません。実際にブレーキをかけて止まるまでには、実際のドライバーに見えるところよりも先のところを見越すだけの能力がないと実は止まることができないのです。やはり移動体の特性が相当異なっているのです。また、鉄道と航空機が大きく違うところは、鉄道は基本的に一人で運転しますが、航空機は原則的に二人で運転することです。飛行機では一人は操縦にあたりますが、もう一人は情報をモニタしている役割を果たします。一方、鉄道のドライバーは一人で全部行わないといけないのです。この状況を加味すると、鉄道ではドライバーに情報をたくさん提供しても実はあまり意味がないというところもあるのです。必ずしも航空機のうまく行ったサクセスストーリーを鉄道に適用できるとは限らないのです。



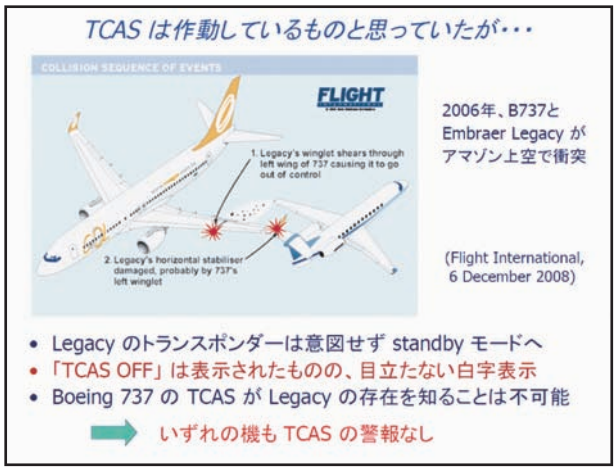
これまで人と機械の共生に必要な2種類のHMI を紹介しました。1つ目がいわゆるHuman Machine Interfaceです。これは人が機械を知るという側面をもっています。機械が人間と意図を共有できるのか、機械は何を考えているのか、機械は何をしようとしているのか、あるいは機械にはどこに能力の限界があるのかなどを的確に知らせるインタフェースが必要だという意味で、2つのHMIのうちのひとつであるHuman Machine Interfaceが重要なのです。



例えば、私の実体験ですが、前の車を自動的に追従していく黄色の車に乗っており、これまでAという緑色の車を追従していたとします。このとき、私の前方にあるレーンには渋滞があるのでそろそろ自動的に減速するだろうと思っていたら、あるところで加速したのです。非常に稀なケースだったと思いますが、たまたま、ターゲットをAからBに切り替えてしまっており、切り替わったということが私にはわからなかったのです。人が見ているものとシステムが見ているものが違うと困るということを示す実例の一つですが、状況認識が共有されていないことが問題でした。

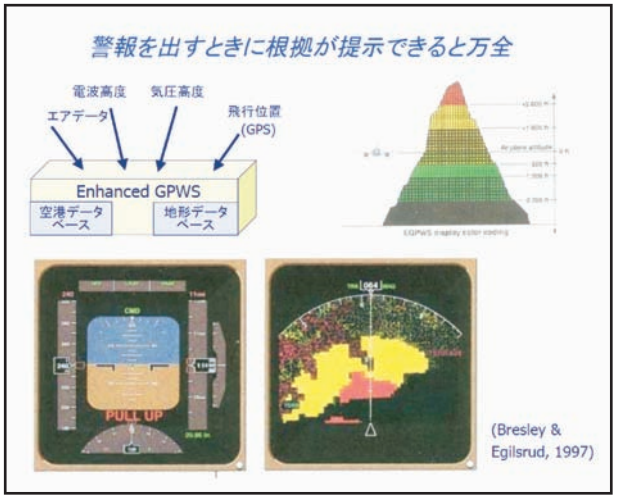


次に、システムの能力限界が分からないと何が困るのかという例ですが、黄色い車を運転していて緑の車を追従しているときに、私の前方の右側から私の前に割り込もうとしている、不審な挙動をしている車がある場合、私の車のコンピュータにはそろそろ減速操作をしてほしいなと思うのですが、減速してくれないケースがあげられます。私には見えている紫色の車が、実はコンピュータには見えていなかったのです。コンピュータがどこまでが見えて、どこまでが見えないのかというのは、現在、実はドライバーにはあまりよく分かるようになっていません。

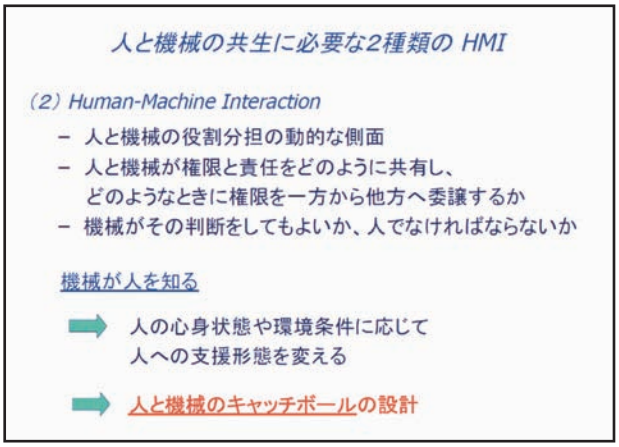


最後に、飛行機の空中衝突を防止する切り札であるTCASに関する事例を紹介します。2006年、ボーイング737とエンブラエルのレガシーという小さな飛行機がアマゾン上空で衝突し、ボーイング737の乗員・乗客全員が死亡したという事故がありました。両方の飛行機は、ともにTCASを積んでおります。TCASは、毎秒1回信号を発し、その信号に対して近くにいる飛行機がレスポンスを返してきたとき、そのレスポンスを見ながら、自分にとって脅威なのか脅威でないのかということを判断するという機能をもっています。そして、「このままでは異常接近あるいは衝突の可能性がある」と判断したときには、2つの飛行機がお互いに反対側の操作を行うことになるように双方のTCASが協調し、一方には「上がれ」、他方には「下がれ」というレゾリューション・アドバイザリを出すという非常にパワフルなシステムとなっています。当時、レガシーのトランスポンダー（信号を発するシステム）が、ある時、スタンバイになってしまいました。信号が発せられない状態になったため、TCASは「オフになっていますよ」という情報をコックピットに表示しましたが、非常におとなしい表示で目立たず、パイロットはTCASがオフになっていることに気がつきませんでした。一方、ボーイング737にもTCASがありますが、電波を発してもエンブラエル・レガシーのトランポンダーが応答しないため、「自分の近くには飛行機はいない」と認識してしまったのです。そのため、ボーイング737側でも警報は出ませんでした。この例では、システムが作動しているかどうかということをどのようにしてパイロットに

知らせるかがいかに重要であるかを示しています。



航空分野におけるすばらしいサクセスストーリーの例として「Enhanced GPWS」をあげたいと思います。これは古典的なGPWSと違い、機能強化型のGPWSであり、全世界の90数パーセントをカバーする地形データベースを持っています。GPSから水平方向の位置が分かり、気圧などから高度も分かるため、自分の前方のどこにどんな山があるのか、すべて把握できることになり、前方が予測できるのです。さらに、ナビゲーションディスプレイ上に、自分は今、三角形の印で描かれたところを飛んでいるが、「前方のここに危ないところがありますよ」ということを図示しながら警報を出します。したがって、その警報が出た時に、「これはウソだろう」とは誰も思わないのです。実際にEnhanced GPWSでは、不信感などのために警報に対応しなかったという例はなく、CFITになった事例もありません。これは非常に重要なことであり、我々は現在、警報を出す時にどうにかしてその根拠を表示したいということを考えています。



2つ目のHMIですが、Human Machine Interactionがあげられます。人と機械の役割分担を動的な側面で見ていこうというもので、ある時には人が権限をもつかもしれないが、ある時には機械に権限を渡さなければならないかもしれないということです。これを実現しようとした場合、「機械が人を知る」必要があります。機械が人の状態を知る、つまり心身状態や環境条件に応じて支援形態を変える必要があります。人間が何を行った時に機械がどのようにするのか、またその機械が行ったことに対して人間はどのようにするだろうか、さらに、もし何か必要であったらこのような手も打たなければならないという意味でのキャッチボールが実現されなければならないこととなります。

映画「モダンタイムズ」の中に、人に効率よく昼ごはんを食べさせるための機械が、口をふいてあげようとナプキンを動かしたときチャップリンがびっくりするシーンがあります。自分のペースに合っていない、要するに機械のペースですべて進んでいるのです。デザインとしては非常に親切に作られているのですが、お仕着せであり、人に合わず、人のペースにも合わず、して欲しくないことまで行ってくれるものになってしまっています。

「タスクアナリシスが必要である」とは言いますが、どんなタスクをしないといけないのかは、もちろんデザインをする時に考えなければなりません。そして、「どのようなシナリオの中でいつどのようにして人はそれを行おうとするのか」ということを考えたうえでないと、デザインはできません。「このシステムを作った時、人の行動はシステムによってどのように移り変わっていくのだろうか、短期的、中期的、長期的に行動はどのように変容していくのだろうか」ということまで踏まえなければなりません。そのうえで、もしも望ましくないような行動変容が出るようなデザインであれば、それは実現

してはいけないのです。このような考えを持ってデザインを進めて行かなければならないことは、我々がいま考えているところでもあります。

以上です。ご静聴どうもありがとうございました。