主たる問題
(Y)

→

G
何が問題を
起こす原因か？

→

問題の原因となる
オブジェクトの
ノブとその設定
(X_1 X_2 X_3 X_4 …)

はじめに

前章は、われわれが解決しようとしている主たる問題、すなわち、「システムが必要とする改良」を定義するのを助けてくれた。われわれはこれを、改良したい「**従属変数** (dependent variable) 」あるいは「大文字のY」と呼んだ。

$$Y$$

ここで前章の「酸の容器の例」に戻ろう。そこでは従属変数として「交換のコスト」を特定した。

$$Y = \text{交換のコスト}$$

問題解決においてよくある間違いは、改善したいものを特定するとすぐに、解決策を求めてブレインストーミング [試行錯誤による解決] を始めることである。例えばわれわれは、「容器の材質を変更して、酸の影響を受けないものにするか、より安価なものにする必要がある」という結論に簡単に飛躍してしまう。こうするときには、われわれは多くのことを当然のことと考えている。要するに、病気の症状を攻撃していても、病気そのものを無視している。これに対して、良い発明家や問題解決者は、常に、「何が問題を引き起こしているのか?」と、[その原因を] 理解しようとする。

最も単純な形式でいうと、原因-結果分析とは、改善しようとしている従属変数を制御する「**オブジェクトのノブ(knobs)**」([別の表現では] オブジェクトのパラメータ、性質または属性) を特定することである。これらの属性を、Yを与える関数の中の「**独立変数 (independent variable)**」として書くことができる。

$$Y = f(X_1\ \ X_2\ \ X_3\ \ X_4\ \ X_5 \ldots)$$

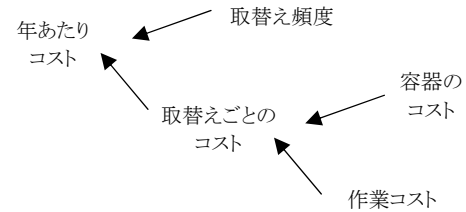
この等式は、「YはX1, X2 … の関数である」、または「Y は X1, X2 … により制御される」と読むことができる。

酸による腐食の問題は、つぎの等式の形で書ける。

$$\text{年あたりコスト} = f(\text{容器のコスト}\ \ \text{容器材質のコスト}\ \ \text{酸の作用}\ \ \ldots)$$

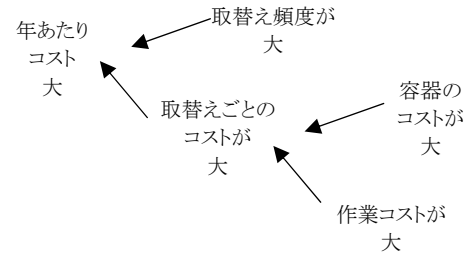
いましばらく、この関数がブレインストーミングのセッションで作られたと仮定しよう。ここで振り返って自分の作ったものを見直すと、この等式の独立変数が互いに独立ではないことに気づくだろう。例えば、容器のコストはその材質のコストに関係する。この関係はここに記した関数ではまだ明らかでない。

この相互依存の関係を考慮に入れた新しい形式で、関数を書くことができる。それをやってみると、新しい独立変数に気がつくものである。



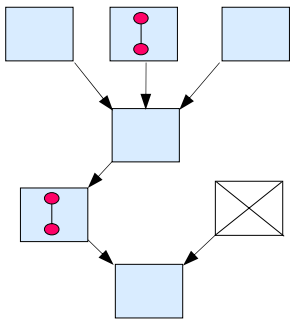
要するにわれわれは、もとの問題にまで至る、**原因-結果の依存関係**の連鎖を形作ったのである。

この各変数に対するノブの設定 [パラメータの定性的値] を記述していくと、われわれの理解をさらに詳細化できる。



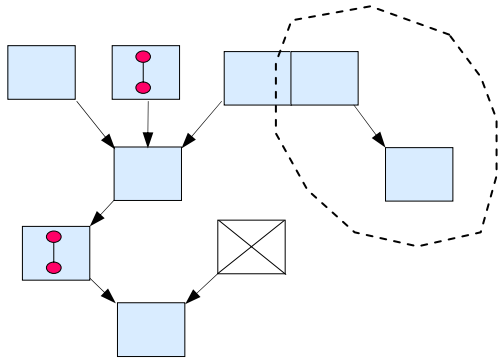
これがさらに詳細化されるのは、「多くのノブ設定が時間によって変化し、何かによって制御されている」とわれわれが認識したときである。一つのノブ設定が時間によって変化するときはいつも、一つの機能が関与している。これらの機能を取り込み、これらの機能に関与するもっと多くのノブを探すことを可能にさせる他のツールを適用するなら、問題を引き起こしているものについて一層深い理解を得るだろう。

これらの概念は本章で述べる内容の始まりであり、以下に示す形を取る「**原因-結果ダイヤグラム**」として示される。



このダイヤグラムの各箱は、ノブとその設定、またはわれわれが解決しようとしている問題に導く一つの機能を表している。これらの機能はオブジェクトの属性を介して互いにつながっている。

原因-結果ダイヤグラムを作っているとき、われわれは究極的に、他のものに影響されない、設計パラメータであるノブ（例えば何かの長さなど）に到達するだろう。ここでわれわれは価値のある教訓、「**要件は原因ではない**」という教訓を学ぶ。それらのノブがそういう設定を持っている理由は、もしそうでないなら他の何かが悪化するからである。実際、一つの矛盾が形成される。それに加えて、一つの代替問題のパスが創られる。われわれはこの代替問題を解決することによって、もとの問題を解決することができる。この代替問題のパスを例示すると、次の図に追加した二つの箱のようである。



この原因-結果の道筋は、後に解決策を探すときに大変役に立つ。第一に、われわれはすべての面から問題を観察してきて、問題を引き起こしている関係をすべて発見するように努めてきた。第二に、多くの矛盾がすでに明らかである。これらの矛盾のどれを解決しても、少なくとも二つの問題を解決するだろう。そのうちの一つはそれほど明らかでなかったものだ。

問題について深く知っているほど、この技法は一層有用になる。長年悩まされてきた懸案の問題に取り組むとき、これは一層明らかになる。通常は、「その道のエキスパート」が周りにいる。問題についての情報の断片はすべて、ばらばらでこのエキスパートの頭にいつも浮かんでいる。この頭に浮かんでいる情報を組織することが問題解決の重要な一ステップである。このような「その道のエキスパート」でも、問題を解決できないのが普通である。彼らは矛盾があることを熟知しており、それを解決できないことに無力さを感じている。解決プロセスの中のより後のステップがこのハードルを越えさせるのである。

問題があまりよく理解されていないと、原因-結果の関係を見つけることは大変時間を要する。実際、われわれが最初にスタートするとき、理論を下に置いてしまっているのがふつうである。原因-結果ダイヤグラムはわれわれの理論を記録する効果的な手法である。これは良い思考プロセスマップとして役立ち、われわれのエネルギーを最も重要なテスト等を行うために集中させてくれる。原因-結果ダイヤグラムは、仕事をチームで行うときに特に役立ち得る。

このダイアグラムに図［ポンチ絵など］を追加して、他の人々が［後から］参加して容易にフォローできるようにすることが大事である。そのような図がなければ、これらのダイヤグラムは「印刷された麻酔薬」になりかねない。読者はあなたが話を終えるまでに普通眠ってしまうだろう。 著者の経験では、原因-結果ダイヤグラムの使い方を習ったことがなかった人々が、急速に追いつき、重要な貢献をすることができる。

問題を引き起こしているものが何かが分からないなら、その問題は極めていらいらするものであろう。「原因の検出が困難な場合に何をすべきか」に関して［後に］特別な項を設けている。もしあなたが多くの資源を持つ大組織の一員なら、恐らく、一時に一つの変数だけを変えたプロトタイプについてスクリーニングテストをするべきだろう。これらのテストは各ノブの相対的な重要性を分からせてくれるだろう。

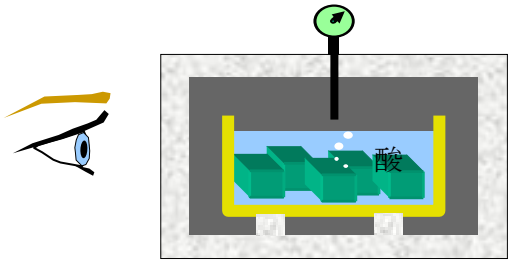
このステップを追加した利点は、問題を解決するのに容易に回せるノブを発見できることである。できるだけ多くのノブを見つけられるようにステップが作ってある。これらの方法は「相対的（“relative to”）」と呼ばれるものと、アルトシュラーの「発明標準解」のほとんどを含む「**ノブの一覧表**」［付録L］とである。

いくつかの分野では、本ステップを「**根本原因分析(“Root Cause Analysis”)**」と呼ぶ。「根本原因」という名前は、われわれが「なぜ?」と何回も問い続けるなら、ついには[真に]「根本の」原因に到達することを含意している。シックスシグマを身につけた人なら、「特殊原因」をもつ問題と「共通原因」をもつ問題には違いがあることを思い出すとよい。「特殊原因」は、プロセスの出力が制御できる限界の外にあり、そのためほとんどありそうにないことを意味している。そのような問題に対しては、しばしば一つか二つの「根本」原因を見つけることができる。しかしながら、[プロセスの出力が] **制御限界内**にある変化の場合は、原因の連鎖が存在する。[そこで]「特殊原因」でない問題を理解しようとしている場合には、「根本原因分析」という用語を避けることを薦める。

本ステップの出力は、ノブとその設定、および主問題をもたらす鍵となる機能である。この出力は原因-結果ダイヤグラムの形式で表せる。

簡易版

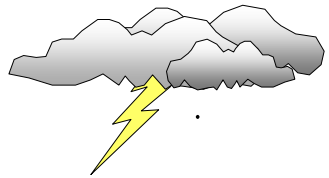
状況/システムを観察せよ



1. 人がシステムを使う様子を観察せよ。可能なら、彼らがシステムを使うやり方に影響を与えないような形で観察せよ。
2. システムの予期しなかった特性を見つけよ。

オブジェクトのノブ (Knobs) を列挙せよ (ブレインストーミングで)

$Y = f(X_1 \quad X_2 \quad X_3 \quad X_4 \cdots)$



われわれが改善しようとしている結果を制御する重要なオブジェクトのノブ (群) は何か？

「なぜ?」と数回問え

なぜ？
なぜ？
なぜ？
なぜ？
なぜ？

各ノブについて、「なぜこの設定をもっているのか?」と問え。

・「なぜ?」と十分な回数問い、いくつかの設計パラメータにまで行きつけ。