システム思考の重要性について考える

慶應義塾大学大学院
システムデザイン・マネジメント研究科 教授
白坂 成功

IPA/SEC所長
松本 隆明

Connected Industriesに代表される“つながる世界（システムオブシステムズ）”の出現により、単体のモノが価値を生み出す時代から、モノ同士やモノと人とのつながりが価値を生み出す時代に変わりつつある。今後は、製品やサービスの設計・運用は、コンポーネントベースからシステム全体で考えるシステム思考へとシフトしていかなければならない。システムズエンジニアリングに関する我が国の第一人者である白坂成功教授に、システム思考の重要性や人材育成、日本の産業界に根付かせていくために必要なことなどについて伺った。

今、なぜシステム思考が重要なのか？

松本　最近システム的に考えるということが非常に注目されています。その要因をどのようにお考えですか。

白坂 成功（しらさか　せいこう）

東京大学大学院工学系研究科航空宇宙工学専攻 修士課程修了。その後、三菱電機株式会社にて宇宙開発に従事。技術試験衛星Ⅶ型（ETS-Ⅶ）、宇宙ステーション補給機（HTV）などの開発に参加。とくにHTVの開発では初期設計から初号機ミッション完了まで携わる。途中1年8カ月間、欧州の人工衛星開発メーカに駐在し、欧州宇宙機関（ESA）向けの開発に参加。「こうのとり」（HTV：H-II TransferVehicle）開発では多くの賞を受賞。2004年度より慶應義塾大学にてシステムエンジニアリングの教鞭をとり、2011年度より現職。

白坂　基本的には三つの要因があると思っています。一つ目は価値と要素の距離が広がってきていることです。今までは単一のモノに価値があったのですが、それが普及し誰でも持っているという状態になると、今あるモノではなく、それらを組み合わせて違う価値を生み出していかなければならなくなります。つまり、求められる価値が徐々に複雑化、高度化してきているということです。例えばボールペンで言うと、単に書ければ良いということではなくて、きれいに書けるとか、書き心地が良いとか、そういった新価値が求められているときに、単に書けるだけのペンを提供しているのでは駄目だということです。モノの要素と価値の距離がどんどん遠くなっています。

二つ目は外界の影響を受けやすくなった点です。システムを構成する要素が増えてくると、システムの外部要素との関係性が増えてきます。つまり、これまでは1台の車だけの関係性を考えれば良かったものが、車と船を組み合わせるとなったら、今度は船の外部要素との関係性も全体のシステムに関係してきます。船に対する変化が車に影響し、車に対する変化が船に影響するようになります。要素が増えるとそれだけ外界が増え、外界の影響を受けやすくなるわけです。この変化対応ということを考えるためにもシステム思考が重要となってきています。

三つ目は説明責任の必要性です。単純なものならば簡単に確認できたのですが、システムになってくると、システム全体として見なければ分からないシステム特性というものを説明しなければいけなくなります。例えば、安全性などはシステム特性の典型的な例です。このように、新しい価値の創造が求められ、関係性が増えて変化対応が上がり、これらをきちんと説明しなければいけなくなってきた。このような3つのことが、システムとして捉えなければいけないという考え方の背景にあると考えています。

松本　色々なモノがつながって動くような時代になってきました。それによって、新価値を生み出せるようになってきたというところが大きいわけですね。

白坂　両面あると思います。組み合わせたから新しいことがで

きるようになったという面もありますが、新しいものが必要だから組み合わせたという面もあります。先ほど申し上げたように、新たな価値を提供しようとすると、新たに組み合わせざるを得ない。新価値創造というのは、経営学では“知の探索”と言われているもので、新たな組み合わせで起こすものです。これに対して、改善活動は“知の深化”と言われます。いったん作ったもののコストを下げていくとか、性能を良くする・軽くする・小さくするというような活動は、今の組み合わせの中でできる改善です。一方、新価値創造では新しい価値を提供し始めようとすることは、今ない組み合わせをせざるを得ないということでもあります。いずれにしても、組み合わせたから新しいことができるようになったと同時に、新しいものが必要だから組み合わせた、という面があるわけです。

あいまい性が拡大するシステム

松本　加えて最近は、モノではなくてサービスで提供するという方向になってきました。とくにシェアリングエコノミーのような形で、自動車そのものを売るのではなく、みんなでシェアして、それを交通手段として活用する、というスタイルになっている。そういうこともシステム思考が求められる要因と考えて良いでしょうか。

白坂　その通りですね。単なるモノの組み合わせではなくなっています。例えばシェアリングエコノミーでは、法制度の問題が出てくる。システムとして提供しようとすると制度自体を変えていかなければいけません。自動運転だったら社会受容性のようなことまで含めてデザインする必要もあります。

松本　製品やサービスの提供側から見ると本当に大変な時代ですね。「VUCAの時代」と言われますが、あいまい性が増し、システムの範囲も増え、どこまでをシステムとして見れば良いのかということが定義できなくなっています。

白坂　もともとクローズドシステムで考えていたものの境界があいまいになってオープンで考えざるを得なくなってきているんですね。

松本　逆に言うと、それだけあいまい性が増え、なおかつその範囲が広域になると、システムとして捉えるのがますます難しくなってきます。むしろ、逆にコンポーネントで考えたほうが捉えやすいのではないかという発想に逆戻りしそうな気がしますね。

白坂　いや、そのアプローチでは価値につながらなくなります。上からと下からの両方が必要だと思います。何を考えているのか分からずに組み上げていくと、どこに到達するか分からないものが生まれます。更に難しいのは、コンポーネント側が変化するということです。テクノロジーが変化する。今これとこれを組み合わせました、というときに、これ自体も変化要素なので、ボトムアップで考えたらまた考え直しです。そうではなくて、上から考えておけば、手段としてのコンポーネントですから、これが変化しても違う手段で同じ価値を提供できるという話になる。

松本　なるほど。変化対応ということで考えても、全体で考えておけばコンポーネントを置き換えるだけで良いわけですね。

白坂　自動運転で言うと、前の車との距離を測るために今は双眼カメラかもしれませんが、次はライダーかもしれない。更にその次はレーダーかもしれない。しかし、前の車との距離を測らなければいけないということは変わりません。「前の車との距離を測るということを利用した安全設計」は同じなのです。手段であるセンサが変わるだけなら、そこだけを検証し直せば良い。下から積み上げていると全部を検証し直さなければなりませんが、上から考えていけば変化した一部だけを検証し直すことで済みます。そういう時代になり始めていると思います。

システム思考で考えるとは、どういうことなのか

松本　そもそも、システム思考とはどういうことだとお考えですか。

白坂　基本はすごく単純で、俯瞰的に捉える、全体として捉えるということだけなんです。ただ、俯瞰と言っても簡単ではありません。

　私は三つの軸で俯瞰するようにしています。一つ目は空間の俯瞰です。この範囲で見ましょうという空間俯瞰。二つ目は時間俯瞰。終わりまでのことを考えて見通して、今どうするかを考える。そして三つ目が意味俯瞰です。これは目的志向性ということになるのですが、ここをターゲットとして、このためにどういうふうにやっていくかという

松本　隆明（まつもと　たかあき）

1978年東京工業大学大学院修士課程修了。同年日本電信電話公社（現NTT）に入社、オペレーティング・システムの研究開発、大規模公共システムへの導入SE、キャリア共通調達仕様の開発・標準化、情報セキュリティ技術の研究開発に従事。2002年に株式会社NTTデータに移り、2003年より技術開発本部本部長。2007年NTTデータ先端技術株式会社常務取締役。2012年7月より独立行政法人情報処理推進機構（IPA）技術本部ソフトウェア高信頼化センター（SEC）所長。博士（工学）。

俯瞰、手段に落とし込んでいく俯瞰です。空間の俯瞰、時間の俯瞰、意味の俯瞰、この三つの俯瞰を進めるのがシステム思考だと考えています。

松本　意味で俯瞰するというのは難しそうですね。目的を達成するためにこういうことをやれば良いという、その抽象度というか、レベルが難しいような気がします。人間はどうしても、何と何をモノとして組み合わせればできると考えてしまうので、その中間的な階層構造のようなものがなかなか思い付かない。

白坂　確かに思い付きにくいのですが、空間と時間だけを見ていても目的には到達できません。システム思考は基本的には全体俯瞰ですが、目的があってやることなので、必ずゴールオリエンテッドにならざるを得ないのです。難しいのは分かっているのですが、そこは慣れや訓練もあります。

私がよく言うのは、家を建てようとしたときに、建てる側からすると初めてなので、どう考えて良いか分からないけれど、家を提供する側からすると、意匠デザインはどうするか、排水や水回り、電気回りはどうするか、お金についてはどういうふうにやろうかという、どういった視点で考えると家が建つのかということが分かっている。家を提供する人たちは「どうしますか」と、家を建てる人に聞くわけです。考えなければいけない項目が分かっている。従って、それをすべての時間と空間においてやってあげればモノはできる。

松本　それはどちらかと言うと、要素単位で考えているということでしょうか。

白坂　いえ、全体を捉える視点です。全体を時間の視点で捉えるのか、空間の視点で捉えるのか、意味の視点で捉えるのか。俯瞰すると言っても、全体をそのままでは捉えられないので、抜けや漏れがないように全体を色々な視点から捉えることが必要です。一度全体を置いて、これをそのまま見ても分からないので分けて考えていく。全体を捉えて、分割して、どう統合していけば良いかを考えながら設計していくというのが、先に進むやり方です。

松本　そうすると、どういう視点で分けていくかというのは大きなポイントですね。

白坂　重要です。例えば、システム思考で有名なのが因果関係ループですが、あれは因果の関係でシステムを捉え直しているものです。システム思考は、基本的には全体を捉え全体で見ます、と言うだけでは先に進まないので、必ず何らかの形で分けなければいけません。

松本　確かにそうですね。

白坂　そのときに、どういう視点で分けていくか。分け方はケースバイケースで毎回変わってくるので難しくなるんです。よくある例が、前から壁が迫ってきたので、壁を押し返そうとする。壁が迫ってきたので押し返そうというのは、一見良さそうに見えるけれども、実際は前の壁を押すと後の壁が押し出される仕組みで、今度は後ろから壁が迫ってくる、という因果の関係になっていれば、何の解決にもなりません。つまり、全体を捉えないで目の前のことだけを見ているから間違った対応になる。物事はつながって関係しているのだから、全体がどういう関係性になっているのかを捉えて解決策を考える――これが今起きている事象をシステム的に捉えるというアプローチです。

松本　先ほどの家の例でも、例えば明るい家が欲しいといったときに、光が色々なところから入るような仕掛けにするのか、照明をたくさん付けるのか、様々な視点から落とし込んでいきますね。

白坂　最終的に実現していくしかないわけです。それをいきなり、明るい家だからいっぱいライトを持ってこよう、では本当は合わないかもしれません。ほかとの関係性もあります。値段の問題もあるかもしれないし、安全面で窓は駄目かもしれないし、強度の問題もあるかもしれない。それらを全部統合して成立させようというのが、システム的に捉えていくという話です。部分だけで見ると家中ガラスにしよう、というアイデアも出るかもしれないけれども、それは一つの観点でしか見ていないので、結局最後は統合していかなければいけません。更に、サービスのようなものが入ってくると、人の感じ方など、必ずしもこうあるはずだと言いづらいものが出てくる。こう考えたからこうやってみた、良かった、悪かった、じゃあここをどう考えれば良いか、というふうに変えていくわけです。物理的にモノを作るのであれば、物理の法則が成り立っているかもしれないのですが、そうではないものがどんどん出てきています。

SysMLは情報の関係性を定義するもの

松本　システム的に俯瞰して捉えるという意味で、例えばSysMLのようなツールを使ってモデル図を書いていくと、全体的な俯瞰の構造が分かるようになってくる、という利点があるでしょうか。

白坂　SysMLは考えたことを可視化するための表記法でしかないので、ただモデル図を書いたからと言って全く新しいものが見えてくることはないように思います。単純化して言うと、以前、3DCADがなかったときには、三面図というものを使っていました。3つの面でそれぞれ表現して、頭の中で3次元がどうなっているかを想像していました。これは単純でした。視点が90度ずつずれている、という状態です。

今、MBSEと呼ばれるモデルベースシステムズエンジニアリングで何をやろうとしているかと言うと、システムズエンジニアがやることをモデルを活用して支援しようとしているのです。その重要なアプローチの一つが、システム全体で考えライフサイクルを俯瞰することです。そうすると多数のステークホルダが見えてきます。使っているときはユーザがいて、メンテナンスのときにはメンテナンスをする人がいて、もしかしたら保険の人がいるかもしれない。廃棄のときには廃棄業者がいます。安全設計上だったら安全の人がいます。このように、様々な人たちが関連したときに、おのおのにとって重要な情報というのがあるはずです。メンテナンスのしやすさを考慮するためや、廃棄のしやすさを重視するための情報があったときに、それを先ほどの三面図のように、頭の中で一つの図にすることはなかなかできません。何らかの関係性がみんなある。それがどう関係しているかというのをどこかで決めてあげなければいけません。この部分を表現するためにSysMLがあるのです。システムモデルと言われているものは、それらを統合的に表した情報の構造体のことを指しますが、SysMLは関係性を定義するために使うものです。情報構造体を定義する人たち、どうなっているのかを見る人たちのための表記法です。

松本　どちらかと言うと、システムを設計すると言うよりも、実際にそれを実装するときの一つのモデルとして考えたほうが良いということでしょうか。

白坂　どちらかと言うと実装よりも設計です。設計するときの情報の関係性を定義するという感じです。つまり、何と何がどう関係するか、ということを決めていくための言語なので、初期の段階で決めてあげるものです。その機能を使ったりメンテナンスしたり、その機能を基に安全性を評価する人たちにとっては、どの機能がどの機能とどう関係しているのか、という情報が必須になります。これは良い、悪い、ここはこう変えたいというのがあれば、この人たちはそれをベースに変えていくことになります。

松本　一つのモデルを色々な観点から見たときに、それの相関関係を示しやすい表記法だということですね。

白坂　おっしゃる通りです。

モノとルールを同時に設計する

松本　俯瞰的に設計するということは、実際にはどのように進めて行けば良いでしょうか。何かツールのようなものがあって、とりあえずそれに従えば近いことができる、というようになっているととても助かるのですが。

白坂　残念ながらそのようなツールはありません。たぶんソフトウェア設計も、当初はそれほど簡単ではなかったと思います。それがソフトウェア分野だけでなく、複数の分野を超えてやっているわけですから、もう一段難しくなっている。つまり、機械設計や電気設計であれば、物理の制約でできることは限られています。それがロジックだけで好き勝手に何でもできるようになってきました。ソフトウェア単独で見てもそうだったのが、今度はハードウェアも含み、システム全体で考えて、ライフサイクルでメンテナンスの人たちも複雑度が増し、関係者がどんどん増えている中でみんなの要求を満たした設計をやろうとしているわけですから簡単ではないものになります。

松本　確かにそうですね。ソフトウェアも昔はそのままダイレクトコーディングでやっていたのが、フローチャートのようなものを使って論理的にやりましょうということになり、やがてフローチャートを書いてもデータ構造がどうなっているか分からないから、今度はデータ構造設計書のようなものができて、更に今は要素間の関係性が増えてきてインターフェース仕様書をきちんと決めないといけない、ということになってきました。

白坂　それが更に違う技術分野も統合せざるを得なくなり、人との関係性も出てきて、サービスやビジネスが関係する。メンテナンスをどうするのか、ということなどを考慮すると更に複雑です。先ほどお話ししたように、人がどう感じるのかとか、ルールなども同時に設計しなければいけない。自動運転でも、車の設計とルールの設計がトレードオフなんです。すごく単純化すると、自動運転車を運転する場合に運転免許を再度取得し直すというルールにしたとすると、ある部分の設計は楽になります。レベル3のような自動運転車をドライバーに戻します、と言われたときに、どんな人が運転しても問題が起こらないように設計するのは非常に大変なことです。しかし、免許を取り直すことを前提に設計すると、車の設計は非常に楽になるんです。

松本　その通りかもしれませんね。

白坂　車がどういう原理で動いているかを知らずに乗っている人はたくさんいます。でも、その人にポンと返すときに、その原理を知っているか知らないかで、当初の設計は全く変わる。今は自分が運転しているので、状況が分かっているわけです。なぜこういう操作をやっていて、その結果何が起きているかをすべて把握しながら運転している。すごく単純に言うと、我々の世代は、例えばハンドブレーキはレバーを引っ張り上げたり、ペダルを踏んだりするから物理的にブレーキをかけているというイメージがあります。でも、ボタン一つでハンドブレーキをかけることもできますよね。

松本　できますね。

白坂　そういう設計にすると、ブレーキという概念が分からない人が出てくるわけです。車の原理を知らないときに、自動運転側に何か起きて人に戻されたとき、今こういう状況ですということを、原理を知らない人に教えるのは大変なんです。原理を知っていれば、今ここのブレーキのこういう押し付けができなくなっていますとか、そういうのが分かれば、実は返し方が楽です。同じようなことはほかにもたくさんあって、例えば自動運転車を止めていて、バイクがぶつかり、そのためセンサのアライメントがずれたので直したとします。この修理で再び安全に走るという証明は誰がしますか、というのもその一つです。ルールとしてその車は必ずディーラーに戻し、ディーラーでしか安全は保証できません、ということにすればディーラーに安全を検証する装置や人を置いておけば良い。そうではなくて、普通に町工場で修理すれば良いだけとすると、町工場の整備士は多分微細なアライメントも含めた検証をすることはできないので、安全かどうかを評価する仕組みを、あらかじめ車に入れておかなければいけないことになります。これは車の中の設計と、ルールの設計との間でのトレードオフです。そういうのがたくさんあるんです。ルールは自分たちで決められないという前提で何でもやるとなると、大変なことをたくさんやらなければいけない。ルールとモノの設計の中身が完全にカップリングを起こしているのです。しかし、今までの開発はそこを同時に設計するということをやってきていない。分離してやっているので、それができるイメージをみんな持っていないわけです。MITが言っているのも、正に一緒にデザインしなければいけないということです。

どう考えて設計したか、情報を残す

松本　感じ方や能力を考慮した設計を行うとなると、人をモデル化しなければいけなくなってくるような気がします。

白坂　人の動きを忠実に再現するということはできないと思います。実際に今でも、自動運転車の設計で、人のモデル化はそこまで厳密にやっていません。人がどう動くかというのは人によって差が出る。決して画一的なモデル化はできません。ただ、それがなければ設計ができないというわけではないと思います。

松本　以前、ABSのような、ブレーキのアシストシステムの効き具合の感じ方が人によって全く違うので、ブレーキが効いていないんじゃないかと思って、それでおかしいと言って問い合わせをする人が増えたケースがあると聞きます。そういう意味で言うと、やはり人によって捉え方というのは様々なので、それを踏まえてシステムをどう設計するかはすごく難しいですね。

白坂　ポイントは、自分たちはどう考えて設計したのかということを残しておくということです。自分たちはこう考えて、こういう人たちはこう感じるだろうと思ってやりました、と。今日のお話の冒頭に説明責任と言ったのはそこなんです。なぜ自分たちはこうしたのかということがきちんと残っていて、正にこのバウンダリーのエッジの外にいる人たちが、危ないと感じるようなことが出てきたら、あなたはちょっと外側でしたね、ということが作った側で分からないといけない。どういう考え方で設計したか、誰がどう感じるように作ったかをちゃんと残して、ライフサイクルを通じて後のことを考えて設計すると同時に、後の人が当初の設計の情報を基に考えるという、行き来ができる仕組みが重要になってきます。

システム思考ができる人をどう育てるのか

松本　では、システム思考ができる人をどういうふうに育てていけば良いのか、ということになるのですが、そもそもシステム思考ができる人が持っているスキルというのは何だと考えたら良いですか。

白坂　我々もよく議論しますが、やはり必ず出てくるのは、抽象化力です。人間が一度に理解できる範囲に限りがあるとすると、幅広い範囲を見ようと思えば思うほど、大量の詳細な要素をそのまま扱うのではなく、抽象化して少ない要素にして全体を考えるようにしなければ全体をつかめないと思います。抽象化して、これは要するにこういうこととこういうこと、この組み合わせなのか、みたいなことがぱっと分かるようにならないと、全体の把握ができない。抽象化するのと具体化するのを、我々は抽象度のコントロールと言いますけれども、これができないと多分うまくいかない。

松本　抽象化力とは具体的にはどういうスキルですか。

白坂　抽象化力は目的志向性を持った帰納化能力と言えると思います。色々な色の車があったときに、色の観点で帰納的にまとめていくのか、車の観点で帰納的にまとめていくのかは、目的によって変わってきます。色々な視点から物事を見て、ぱっと考えることができて、その中のこのポイントが今の目的に対しては適合しているから、この視点で考えよう、ということが考えられないといけません。

松本　ある意味で言うと、論理的な思考ができるということでしょうか。

白坂　論理思考は前段階として必要です。ベースとして論理的な思考ができるというのは、おそらくあると思います。

松本　そういう人材はどのように育てていけば良いのでしょうか。

白坂　論理的な思考は訓練でできるようになります。抽象化力は、難しいのですが、訓練すれば身に付きます。ただ、100人に教えて100人全員が完璧にできるようになりますか、と言われると難しいでしょう。

松本　それは個人に依存するということでしょうか。

白坂　その人にとっての向き、不向きはあると思います。それが興味なのか、心理学的なものなのかまでは分析していないのですが、明らかに、ぱっとうまくできる人と、なかなかできない人がいます。

松本　システム思考ができる人間をどういう場で育てていくのが良いのか、大学できっちり理論的な研究、学習で教えていくのが良いのか、それとも現場で実践して身に付けていくのが良いのか、どうお考えですか。

白坂　両方必要だと思います。現場でやっていることには、システム思考以外のものがたくさんあります。何がシステム思考の範囲で、何がそうではないのか、現場だけではそこが分からなくなります。一方で、理論だけでは使えない。教育理論で言うと、行動主義と言われる知識を問うものがあります。これは知識なんです。使うかどうかは関係ない。やる過程が合っているかどうかを見るのは認知主義的なアプローチです。ワークショップなどで、ここはこうやったほうが良いですよ、ああやったほうが良いですよと教えていくようなものですね。しかし、これができても学んだことが自分の課題に使えるかと言うと、ダイレクトにはできないです。やったことをいったん自分のものにして、違うものに適用する能力というのは、もう一段上の能力で、これを教育するのは構造主義的アプローチです。実際に使うことを通じて、知識を一度昇華させる。学んだこれは、こういうふうにやることなんだ、こういう考え方をすることなんだ、と自分で受け取れると、違う分野に持っていけるようになります。実際に習ったことを使いながら自分のものにしていく作業をやらなければいけないので、プロジェクトベースラーニングと言われています。“やって、学んで、やる”――これがベストです。学んだ後は、やらないと駄目なんです。過去の経験を学んだことにマッピングするだけでは、なぜそれで良いかが分かりません。マッピングすれば、これはこういうことなのかと分かるのですが、新しいところに持っていったときに、結局自分のやったものでしか持っていけなくなります。違うものをやったときに分からなくなる。学ぶ場合は汎用的なものとなりますが、実際に使うためには、その分野に落とし込む必要があります。この落とし込みのときに、こう考えたから、これは実はやらなくても良いんじゃないかと、重要なポイントが違うのをちゃんと自分で理解して、今やろうとしているシステムはこういう特徴だから、こういうアプローチでやらなければいけないんだと、学んだ後にやるということことはすごく大きいのです。最も良いのは、システム開発をした経験がある人が、リカレント教育で学んで、更に実際に使うということです。ぜいたくですがそういうことなのです。

松本　いったん学び直して、きちんと体系化され、普遍化されたものにして、それを別の領域に適用していくということですね。

白坂　はい。その通りです。別の領域でやる必要がないのであったら、誰かが作ったものの通りにすれば良いだけです。しかし、そんなことはこれから先あまりないでしょう。新たな人材を育成するとなると、実際にものづくりを経験したことのある人たちが、一度きちんと学んで、自分たちの領域で使っていくという、ここをいかに早くやるかだと思います。日本では社会人が学ぶことはMBA以外でほどんどありません。この教育の弱さがシステム的な人材育成の弱さにつながっている可能性はあります。海外では、社会人がもう一度学ぶことは当たり前です。

松本　システム的に見ていくという能力がどこまで付いたのか、この評価については、どうお考えですか。

白坂　簡単ではないですね。例えばシステムズエンジニアリングで言うと、スキル標準的なものは世界中で作られようとしています。もともとイギリスが作ったものがあるのですが、それをブラッシュアップしようとしています。難しいのは、それで評価できるのが知識なのか、ということです。資格というのは大体において知識を問うものです。しかし、簡単に分かる通り「知識がある」=「できる」ではありません。前提になるかもしれないけれど、ここのギャップを埋めていくときにどう評価するかというのは、やっぱり単純ではありません。確かに、できる人とできない人というのは話していて分かります。ではそれを客観的に評価する仕組みは何だろうというのは、実は今悩んでいるところです。

学び直す機会を作る

松本　システムを設計するときに、いわゆる良いシステムと悪いシステムと言ったら良いのか、先ほどの色々なシステム、俯瞰的に見て考えられたシステムと、そうではなくて積み上げで局所最適の合計でできたようなシステムと、どのように評価すれば良いのでしょうか。

白坂　難しいのですが、システムの評価について今あるアプローチは、結局は目的適合性だと思います。何で評価するか自体もシステムごとに異なるというのは、基本的にシステムの分野で言われていることです。そのシステムにとっての重要なポイント、

低コストが重要なのであれば、低コスト性が主な評価軸でしょうし、何らかの品質が重要なシステムであれば、それがポイントになるでしょう。もちろん一つではなくて複数になるのですが、それによって変わるというのが今の世界的な流れだと思います。

松本　その優先度が分かるというのがシステム思考に優れた人間なのかもしれないですね。目的が幾つかあるときに、どの目的がメインなのかということが分かる。

白坂　確かにそこを間違えてしまうと、機能がたくさんあるから良いと考えてしまったりしますね。でも機能がたくさんあると結局、使いづらいこともあります。

松本　そうですね。コストにもつながってきますし、保守性の良し悪しにもつながってきます。やはり本来の目的、メインの目的は何かということをきちんと見定めることができる人間、そういう人をどうやって育てていくのかということですね。先ほどのリカレント教育によって、きちんと学び直す機会を増やしていくことの重要性を感じます。

白坂　同感です。

部分をシステムとして捉えて考える

松本　学び直すときの、学の受け入れ体制についてはどうお考えですか？

白坂　日本の教育には大きく2つ問題があります。一つは社会人が学べる環境です。これは内容を問わず、そもそも少ない。ビジネススクール以外ほとんどないのが現状で、働きながら学べる環境をもっと整備しなければいけないと思います。

もう一つは教える側として、システム的なものを教える人の確保が難しいことです。上述した通り、学んだ後に、それを実際の開発に適応して初めて分かるという意味では、ずっと学にいる人ではない人を学に連れてきたいわけです。アメリカでは普通に行われていることです。企業出身の人が、今度は学に行って教えて、そこで学んだ人が実務に適応して学んで、また学に来る人がいて、というようになっています。

松本　そもそも今の学は、どちらかと言うと要素技術偏重で、AIや量子コンピューターやデバイスなどには力を入れるけれども、システムなど目に見えないモデルのようなものを教えることへの取り組みが弱いですね。

白坂　その理由の一つが、論文になりにくいことだと思っています。システムの開発は、違う人が同じプロセスでやっても同じ結果にはなりません。再現性がないため、サイエンスではないと考えられてしまうんです。

松本　確かにそうかもしれないですね。システム思考でやらなければいけないということを、日本の産業界にもっと根付かせていきたいと思うのですが、日本の産業界というのは色々な問題を抱えていて、例えば産業構造の問題があります。ソフトウェアの開発も多重下請け構造で、末端になると自分が何のソフトを開発しているか分からないんです。本当に個別のモジュールでしかない。それをシステムで見ろと言われても見られないのではないでしょうか。しかも全体をシステマティックに見られる人間は発注側にもいません。

白坂　おっしゃる通りですが、とは言えきちんと作っている人たちがいます。システムは階層性を持つので、必ずしも全体を見る人だけがシステムを見ているかと言うと、決してそうではありません。私も「こうのとり」をやっているときは、「こうのとり」の全部を見ていたわけではありませんでした。その中の電気モジュールという一モジュールを見ていたんです。ここを見ている人が全体をシステム的に考えなければいけないのかと言うと、そうではなくて、部分をシステムとして捉えて考えるというのもあります。メンテナンスや試験の容易性を考えて、どうやってここの範囲を設計しようか、ということは考えられるはずです。自分のやる範囲の中でシステム的に捉えてやるという訓練はできるので、そこからでも良いと思うんです。

松本　最近は将来の機能拡張に容易に対応できるような仕組みにしておくとか、なるべくモジュールの構成にしておいて、取り替えられるようにしていく、そういうことを少しずつ考えるようになってきましたね。

白坂　例えば変化対応をどうしようか、何か変わったときに、これはどうやって変えやすくしておこうか、というような考えがあるのだと思います。システム・オブ・システムズの全体を設計しているわけではなくても、たとえセンサの中の組込みソフトだったとしても、セキュリティも考えなければいけない場面があるかもしれません。うちはここだけだから、と言い切るよりは、ここだけでもシステムとして考えておく、ということをやるべきでしょう。

その辺りが変わっていくとすごく良いと思います。日本の場合は技術力がある。システムズエンジニアリングは、結局はシステムなので、最後は技術になるわけです。技術というか、モノになります。ここがちゃんとできる人が日本はもともと多い。だから、システムズエンジニアリングが強くなれば鬼に金棒なんです。海外では、システムズエンジニアリングのようなシステムとして捉えるやり方を、工学部ではみんな学びます。とくにアメリカはそうですね。ヨーロッパも、ドイツはインダストリ4.0のもとで今どんどんその分野を強めていますが、結局そこを学んでいるかいないかの差がすごく大きい。日本にも学べ

ばできる人はたくさんいます。単に環境として学ぶチャンスが少ないから、できる人が少なくて、活用する人が少ないだけだと思います。

産業界で活躍している人からもっと学ぶ

松本　もう少し産と学が密に連携するようにしていかないといけないでしょうね。先ほどの学び直しのようなチャンスを増やしていくなど、学のほうも産と一緒にやって、製品化のところまで考えていくような、そういうプロジェクトを作っていくことも必要になってくるかもしれないですね。国レベルのテストベッドのようなものを立ち上げることも必要ではないかと個人的には思っています。昔アポロ計画でシステムズエンジニアリングが大躍進した、あのようなものです。

白坂　日本人は昔から人に思いをはせて作ることが得意なはずです。いわゆるシステムズエンジニアリング的な、要求を決めてがちっとやるというだけの話よりも、もう少し人の気持ちや感じ方に踏み込んでいって、例えば先ほどの新しいシステムの社会受容性を上げる設計のような体系ができるはずなんです。それを産業の幾つかのパターンでやって、日本としてどこを強めていけば良いのかということを考えたほうが良いと思います。申し上げた通り、やって学んで、学んでやって、使ってみればできるようになる人はたくさんいて、それを教えられるような人もたくさん出て、そこを起点にして次の世代を生んでいくというような、その最初の世代を生むのはプロジェクトベースでやらざるを得ないんです。そこは国が支援しても良いような気がします。

松本　とくに社会受容性のようなところの検証実験を考えていくべきなのかもしれないですね。単に技術の検証だけだったら、産業界だけでもできる話かもしれません。

白坂　そこを分離してやるんじゃなくて一緒にやる、一緒にどうデザインしていくのか、どう設計していくのかをやっていくようなパイロットにすると良いと思うんです。社会受容性だけを検討しますというグループと、技術をやりますというグループを分けて作ってしまうと、結局はばらばらで、全体を設計する人たちにならないので、全体を設計するような人たちのグループを作って試していく。

松本　正にシステム思考ですね。最後に、今後こういう考え方を進めていく上での課題について、どうお考えですか。

白坂　教えることができる人の絶対数の少なさです。最初のひと回し、最初の教えられる人たちを一度作り上げるというところをどこかで回さない限りは、本当に少しずつしか増えていかない気がしています。1年2年で数人鍛えても広がらないでしょう。

松本　そうですね。

白坂　どこかで数年かけて一回大きく回すということをやれば、その後はそこから生まれた人たちが回していくことができます。教え始め、学んでしまえば、できる人たちは絶対できるというのが今までやってきた私の感触です。

松本　その場合、本当にトップのいわゆるスーパーマン的な人材を育てていくのか、それとも全体をもう少し底上げしていくのか、どちらが良いとお考えですか。

白坂　両方必要だと考えています。教えられる人が増えないと、結局絶対数が増えませんからね。その一方で、これだけではまだまだ足りない、広がらないということも感じています。

松本　そうですね。これだけ急激に変化する時代においては、システム思考をどんどん広めていかなければいけないと思います。ぜひ今後ともお力添えをお願いします。本日は貴重なお話をありがとうございました。