Systems engineering
in search of the elusive optimum

The current wave of enthusiasm for systems engineering is considered, and 'litmus tests' are presented to identify whether activities presented as systems engineering really justify the title. The fleeting nature of optimality is explored, showing that the need for flexible, adaptable solutions is increasing as social turbulence and tempo increase. The goal of systems engineering as seeking optimum solutions is explored together with the relationship between optimum and minimal entropy configurations. How it is possible to identify optimum conditions without the need for mathematics is illustrated. Finally, a process/life-cycle model of systems engineering is introduced which retains the flexibility and adaptability essential to deal with the dynamic markets in which we increasingly compete.

by Derek K. Hitchins

At a recent systems engineering conference, it was observed that, within Europe at least, systems engineering seems to appear, disappear and reappear periodically on about a 25 year cycle—coincidentally, perhaps, once every generation…

At the same conference it was noted that many papers, although interesting, did not seem to be about systems engineering *per se*. Instead, work already done was being presented *post hoc* as though originally founded in systems engineering. Formalised processes were being claimed as successful instances of SE.

Taken together, the two observations raised some interesting questions. What decides whether some papers were about systems engineering while others were not? And, could it be that the formalising of process is a recurring phenomenon which accompanies, perhaps even *causes*, the periodic demise of systems engineering?

First, what is *a system?*

'An open set of complementary, interacting parts with properties, capabilities and behaviours (PCBs) emerging both from the parts *and* from their interactions.'

From this definition, any system could clearly be made up from different kinds of parts, arranged and interconnected in different ways. Some parts and some configurations may produce overall PCBs which were more valued, i.e. optimal. Fundamentally, systems engineering is about synthesising optimal solutions to problems.

There are many ideas about the 'promise' of systems engineering, but in general terms it seems that most people expect it to create the 'optimum solution' to any problem, need or opportunity.

Within this broad statement, 'solution' could be a process or a product or both. 'Optimum' is defined by context, e.g. best balance of efficiency, effectiveness and quality, or shortest time to market consistent with cost and quality, or meeting the customers' and users' needs at minimum cost…and so on (Fig. 1).

So, the looked-for results are generally pragmatic, real-world benefits, and they tend to focus about the notion that there is indeed some

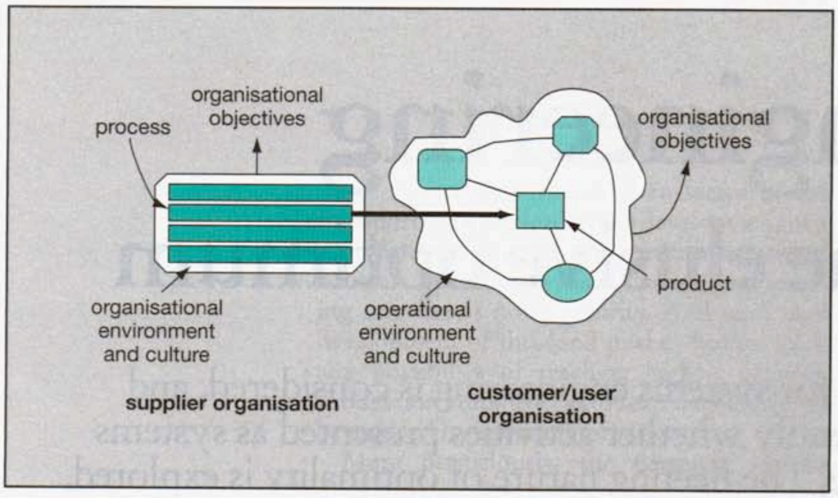


Fig. 1 Twin goals of systems engineering: there are two organisations, in two environments, with two sets of objectives—supplier's and customer's/end user's. The first goal is for the *process* of systems engineering to contribute optimally to the supplier's organisational objectives, in concert with other supplier projects and processes in the supplier-organisation's environment. The second goal is that the *product* of systems engineering should contribute optimally to the customer/end user's organisational objectives, in concert with other interacting systems in the end-user's environment

optimum arrangement, configuration, selection of 'parts' that goes to make some overall optimum 'whole'. And, since everyone is looking for an edge, whoever can find that elusive optimum has a distinct advantage, be it in business, commerce, engineering management or whatever.

Elusive optimum

Achieving the optimum system

Achieving any 'optimum' involves a balancing process—system components or elements must be selected, changed or rearranged in some way to enhance overall system properties, capabilities and behaviours.* Since elements mutually interact, changing any element or relationship may affect other elements. For instance, if the system of interest were a geosynchronous communications satellite, increasing the amount of stored thruster fuel for greater endurance might necessitate reducing the mass of other satellite parts, rearranging configurations to restore the centre of mass, and so on—and there would be practical limits to the balancing act. Any change in one part, designed to optimise the overall system, then, affects others within that whole system.

Inevitably, in such balancing processes, there are winners and losers. So, this balancing process can be tricky. As a result, organisations have developed sometimes arcane, often carefully guarded, secrets about how such-and-such an optimising process achieves its results. A mystique has, in the past, risen around systems engineering that has propagated the belief that it is difficult, and that some people can do it while others cannot—i.e. elitism.

Achieving 'optimum' does require that *the whole system* be included in the balancing process: excluded elements are unable to contribute, or adapt, to any optimising process. It is especially important to include non-adaptable items so that others may adapt around them to find the best *practical* optimum—optimisation is often about the best you can do in the circumstances…

*Properties generally refer to stable features, physical characteristics etc. Capabilities refer to potential functional achievements. Behaviours describe responses to stimuli such as sensory or information inputs.

Context shifts

In the real world, the optimum design configuration for any system is constrained by context, interacting systems, environment and foreknowledge. What if these change with the passage of time? Does the optimum condition change, too?

This question has become increasingly important as the tempo of change has increased in recent decades. The causes are many and well known:

- improved communications, transport, information handling
- consumer demand decreasing commercial product life cycles.

It is becoming increasingly evident, too, that we live in a non-linear-dynamic world. In all probability, our world is increasingly non-linear-dynamic *because* of decreasing social interaction times. Like stockmarket crashes, chaotic social behaviour can be induced by speeding-up interactions.

In many respects, societies without and within our industrial and commercial organisations exist on the so-called *edge of chaos*. The phenomenon of self-organised criticality[1] is becoming widely recognised as a potent driving force, in which interacting systems irresistibly drive themselves into a highly dynamic state. Deaths in war, crime, distances between cars on the motorway, noise in conductors, earthquakes—all seem to respond to this urge. So do directors and managers, driving themselves to a state of overworked frenzy as if, somehow, this were their optimum performance state.

In much the same way the phenomenon of auto-complexity generation occurs. It is an everyday observation that any system (process or product) gets more complex with the passage of time. The urge to add 'bells and whistles' seems irresistible. The opportunities to apply 'patches' to restore/enhance some capability seem endless. It is as though the Second Law of Thermodynamics, that disorder increases with time, were evident in our everyday organisations and systems.

As context, interacting systems, environment and knowledge change; what constitutes the 'optimum' configuration must indeed change too. Concomitantly, the increasing social tempo of recent decades demands a fresh look at how we design and create future systems—demands a new systems engineering, one that definitely does not disappear periodically!

Significance of 'optimum'

The notion of 'optimum', then, is *not* one of some fixed balance point. The optimum path to any goal is *not* fixed—it changes as the environment changes, as new factors emerge. Unlike the technology from which they may be partly or wholly comprised, the points at which systems are optimal are for ever in a dynamic state of change induced by their mutual interactions with other, changing systems in a dynamically changing environment.†

It is this observation, more than any other, which distinguishes systems engineering from other disciplines. Systems engineers see the world as fluid and dynamic, and the optimal system as fleeting, even ephemeral. Today's challenge for systems engineering is to create effective, enduring systems which *pursue* and *maintain* optimality in this changing scene.

Optimum whole comprised of sub-optimal parts...

It follows from previous topics that choosing elements/subsystems which are themselves optimised does *not* result in an optimum overall system.[2] Common sense tells us:

- a collection of soloists does *not* make a good choir/orchestra; they find it difficult to adjust their own performances to blend in with each other
- a F1 car made from the best parts of Williams, Ferrari, Maclaren, Benetton etc., would not make the best F1 car; the parts would not interact effectively to provide a balanced whole
- a well-designed and tested artificial heart may *not* suit the transplant patient; it might be too heavy, incompatible with the work environment, place too much pressure on other weakened blood vessels...

Reason? Element/subsystems optimised in their own right and their own context are unlikely to be optimal when interacting with other parts in a different environment and context. Or, looked at another way, bringing a set of parts together will not result in an optimal system *unless* there is a whole-system optimising process. Even then, the optimum reached may fall short of the ultimate, if the parts which were selected cannot be sufficiently changed, adjusted or reconfigured.

This observation, capable of irrefutable proof, sits uncomfortably with those who simply collect a set of parts and create a system from them—so-called systems integration.

Product and process—defining ideas

Both product and process may be viewed as systems. One way to look at systems engineering is as a process which produces a product. The process is a system, interacting with other systems in some (developmental/organisational/cultural/technological) environment. The eventual product will be a system, too, interacting with other systems in some (operational/organisational/cultural/technological) environment. So, in designing a systems engineering process, there are two distinct issues:

- the *process*-as-a-system must exist, perform and survive within the organisational environment
- the process will create a viable *product*-as-a-system within the operational environment.

Looking outwards

Because systems exist within systems exist within systems...what is true for the subsystem is true for the whole system. So, the idea of optimisation by altering the component parts/subsystem while measuring changes in the whole is recursive, i.e. it occurs at every level of system-within-system.*

†For instance, how many of us have home computers stored in a cupboard, as good as new, but outdated/outmoded by the introduction of newer, better, faster, shinier, cleverer machinery.

*Systems engineering addresses this issue of potential complexity by encapsulation: each subsystem is viewed as opaque, with discrete properties, capabilities and behaviours; how these might be created may be disregarded in this context, just so long as they exist repeatably. Encapsulation is the key 'trick' for managing complexity, since it is possible to understand and manage the creation of even the most complex system using encapsulation to 'hide' the complexity below one level of decomposition. In this way the intellect is able to cope with situations that would otherwise seem too complex to comprehend.

So too, a system-to-be-created must be viewed during conception and design as though it already existed and was already operating—only in this way can the whole system be created to complement (future) interacting systems, yet achieve its purpose.

For this reason, systems engineers concern themselves with looking 'outwards' into the future environment. This often involves some form of modelling—it always involves imagination. This brings us, perhaps, to our first 'litmus test' to establish whether 'real' systems engineering is being pursued.

Systems engineering litmus test A

True systems engineering will pursue the following generic process:

- first and foremost, look *outwards*
  —into the environment
  —into other systems
- identify what effect the system-to-be-created will have on other (future) systems in the (future) environment
- identify/anticipate response of those future systems
- hence establish requisite properties, capabilities and behaviours of the dynamic, interacting system-to-be-created.

Conceptually, this process creates (future) order—a compatible, balanced, enduring effective system

Systems engineering litmus test A:
systems engineering looks outwards first to identify the dynamic problem space

The social/industrial/commercial environment is operating at or near the edge of chaos—and so it should if we want to live in a creative, dynamic world. This nonlinear dynamic world in which we increasingly live defines the second 'litmus test' for systems engineering:

Systems engineering litmus test B:
the whole system (product and/or process) is continually rebalanced to optimise properties, capabilities and behaviours throughout its life cycle

This litmus test suggests that systems engineering and optimisation continue after system creation, into system operation and finally to system replacement. Employees of airlines, nuclear power generation industries and fighting services, for instance, who re-optimise outdated processes and products are indeed systems engineering according to this litmus test. In the process they are continually restoring order, or reducing disorder that has disturbed the pattern of interacting systems and interchanges with the passage of time.

The first litmus test looked 'upwards and outwards': the second addressed the time domain. The third litmus test looks at the optimising process itself, which is designed to

Table 1 Applying systems engineering litmus tests to contemporary practices

		litmus tests				
item	candidate	*A*	*B*	*C*	*D*	comment
1	classic decomposition, progressive build-up 'V'	N	N	N	N	generally fails all litmus tests; systematic but not *systems*
2	quality function deployment	N	Y	N	N	balances contributions during specification; static only; necessary, but not sufficient
3	classic systems engineering, continually proposing variety of solutions, trading-off to find best fit…	Y	Y	Y	Y	can be hit-or-miss; needs many solutions to ensure finding optimum; D presumes system proving in dynamic simulation
4	concurrent engineering, UK-style*—integrated product design teams	Y	N	Y	N	intended to reduce rework by anticipating errors and omissions; static, one-off; *creates* organisational entropy
5	concurrent engineering, UK-style—telescoped, overlapped processes/activities	N	Y	Y	N	intended to minimise time to market; *creates* organisational entropy; fragile, high risk
6	centralised software development, intensive software systems†	N	N	N	N	increases both organisational and process entropy; not systems engineering

*As opposed to, say, Japanese automobile concurrent engineering, where 'concurrent' refers to activities synchronised along the length of the supply chain/supply circle

†The term 'software system' is an oxymoron, since software on its own cannot constitute a system

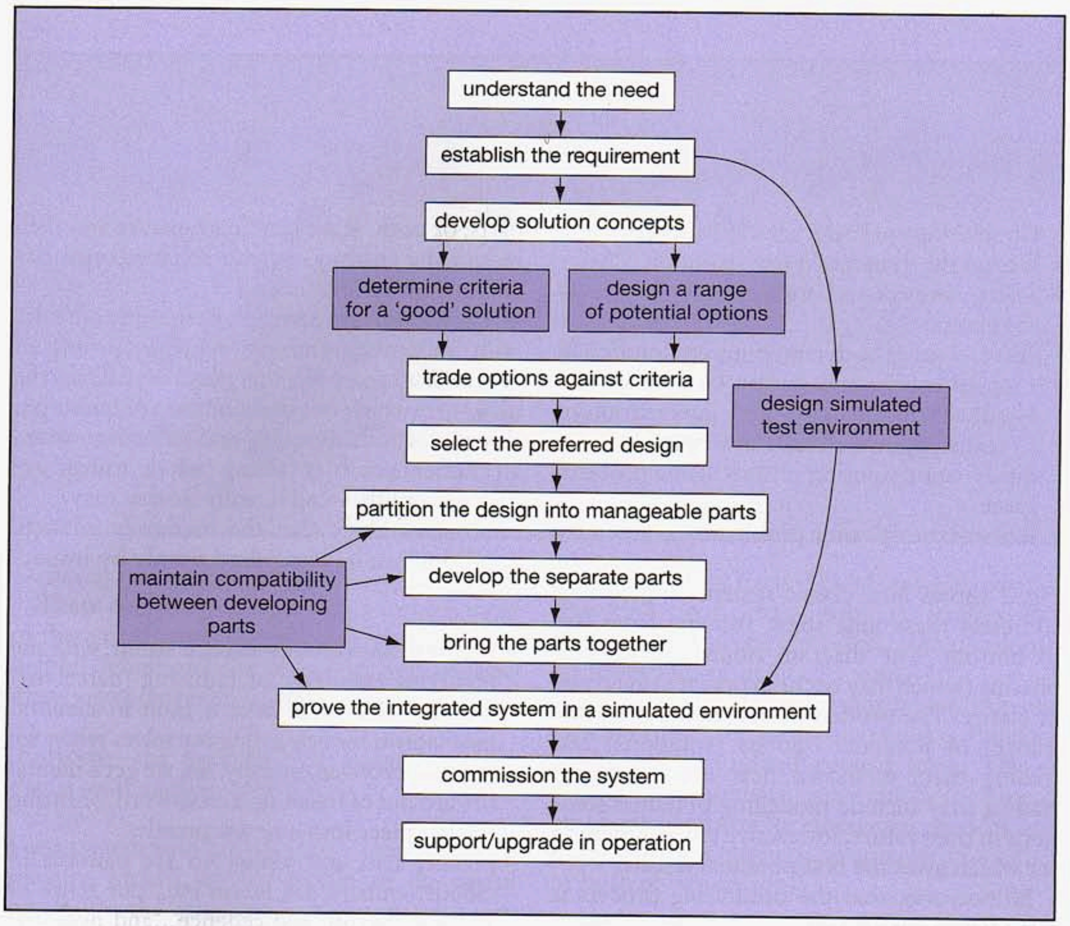


Fig. 2 Classic systems engineering process

reduce entropy within the whole system-of-interest:

Systems engineering litmus test C: the whole system (process and/or product) is balanced by adjusting internal parts/ relationships to optimise whole-system properties, capabilities and behaviour

Finally, the last litmus test recognises that the *dynamics* of interactions has a major impact on the point of optimality:

Systems engineering litmus test D: the condition for optimum system properties, capabilities and behaviours is identified while the whole system is interacting with the other systems in its operating environment

It may seem that this is a return to systems engineering litmus test A, above. In once sense, it is, since it requires that we look outwards into the operating environment. However, the essence of the fourth litmus test is concerned with dynamics. The optimum point can be truly identified only when systems are interacting *dynamically*. Often it is *rates* or *surges* or *frequencies* of flow emanating from one system that prove inadequate or overwhelming to another, interacting system.

In many ways, litmus test D is the most difficult for the reductionist to accept: it would be so much simpler if we could just bring pieces of technology together to create a system-in-isolation and then launch it into its operational environment. And, of course, you can. But the results will not only be far from optimal, they will not meet your customers' requirements. They may even make matters worse and they will incur a high risk of failure. Just think of the fiascos associated with the introduction of major information systems for management, stock exchange dealing etc.

Applying the litmus tests

Table 1 shows the application of the four systems engineering litmus tests to some everyday industrial practices. The conclusion to be drawn from the Table is that many practices which present themselves as systems engineering are either inappropriately titled or are, perhaps, only part of a much richer story. Even classic systems engineering is seen to have some limitations in practice.

Systems engineering process—the steps

Essentially, systems engineering is a problem-solving methodology for addressing complex issues. In that context, it can be viewed as a matched sequence of activities:

1 Explore/bound the dynamic problem space.
2 Synthesise a dynamic solution.
3 Develop viable solution concepts.

4 Choose the optimum solution concept.
5 Design the dynamic system solution.
6 Select, connect and configure parts to meet the design.
7 Test and tune the dynamic system solution in a representative dynamic problem space
   — adjust/modify parts and interactions to realise/optimise PCBs of whole.
8 Fit dynamic solution into dynamic problem space.
9 Resolve the dynamic problem.

Fig. 2 shows how classic systems engineering addresses these nine steps, running from top to bottom. The diagram omits rework and phasing (which may occur in practice) in favour of clarity. The process of generating a variety (range) of potential options (solutions) and trading them is shown near the beginning; trading may include modelling potential solutions in their future, interactive environment to see which gives the best potential results.

Notice, too, that the optimising process is intended to continue throughout development as the various subsystems are kept in balance. In practice, requirements change, environments change, understanding increases and development wanders from the specified path. Maintaining compatibility between the developing parts can prove a lively task, although it can also reduce to simple budgeting in some organisations.

One factor not addressed by the Figure is the essential process of partitioning. The way in which an overall system design is partitioned into parts turns out to be crucial. There are many options and many influences. But is there an optimum?

Optimum and entropy

Entropy—the silver bullet!

Research indicates that the notion of 'optimum' for any system is associated with minimising entropy—the degree of disorder in any system. Entropy presents itself in many forms, most commonly either as a thermodynamic measure, or a measure of the number of different ways items can be arranged. These different measures are mathematically equivalent.

The essential notion of 'system' is one of order; the more ordered, the more we perceive a well-organised, well-structured system. For our purposes, we are particularly interested in configuration entropy, the degree of disorder in the pattern of process activities, or product parts, or both. If we can first measure and then reduce the entropy, we can enhance organisation and structure.

The notion that a system optimum coincides with minimum entropy requires proof, of course. But, accepting that proof is possible, the idea of searching out the condition of minimum disorder is both appealing and seductive: we are all rather good at seeing when things are ordered or tidy—can it really be that easy?

It seems likely that the minimum entropy condition can be recognised simply by inspection:

- Our human cortex evolved, it seems, with the powerful capability of reducing (perceived) entropy. We even have a built-in cerebral mechanism for rewarding ourselves when we reduce perceived entropy. So, we get a mental reward out of finishing a crossword, or fitting the last piece into a jigsaw puzzle.
- Hence, also, our ability to see patterns in clouds, embers, tea leaves etc., our sense of musical rhythm and cadence…and potential order in a jumble of activities and parts.

So, for some, yes! it really can be that simple—except that we have been taught as engineers not to trust our instincts. From the pyramids to Gothic cathedrals, great architecture was conceived, designed and constructed using the architect's feeling for proportion. Renaissance artists created paintings of enduring greatness using the proportions of the Golden Section discovered by the Ancient Greeks.

Only in recent times have we supplanted the instinctive artistry with the calculated science. And only in very recent times have we started to realise that the feeling for sound proportions might actually derive from the brain's capacity to find a minimum entropy configuration in the cortex. Just as the human cortex discovered musical proportion before science discovered music's mathematical basis, so it appears that the human cortex can perceive system optimal configurations.

We all have this capacity. For many of us, does not the Second World War Spitfire or the post-war Lightning simply 'look right'? When we look at a new bridge, do we not instinctively feel whether or not it is 'right'? When a new model of car hits the showrooms, do we not get an immediate feeling for it? When we see a new system design, do we not get an instinctive feeling for its completeness, its balance, its structure, its flow…?

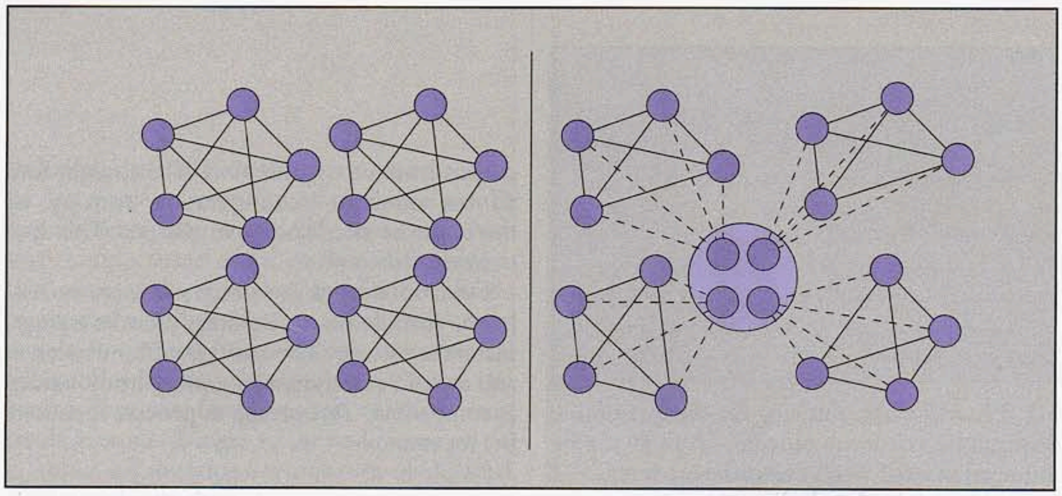


Fig. 3 Partitioning the system

Defining entropy

There are many definitions of entropy, most of them mathematical. However, a particularly helpful definition from an engineering management viewpoint is as follows:

'Entropy—the measure of a system's energy that is *unavailable for work!*'

For a system with high entropy, internal energy remains locked inside, unable to do useful work. On the other hand, a system with low entropy can direct most of its energy towards useful work.

Low (configuration) entropy, or minimum disorder, is associated in open systems with both efficiency and effectiveness explaining why systems engineering seeks to reduce entropy in process *and* product. It also explains why, in many cases, 'minimum entropy' corresponds to 'optimum'.

Eine, kleine mathmusik...

This subsection introduces some mathematically-based notions. Readers who are nervous of mathematics, and many of us are, please keep reading. The result of the mathematics may support what you already know—that systems engineering is largely common sense. After all, as Descartes noted, common sense must be one of the most universal attributes, since so many people seem to think they have it...!

Finding the minimum entropy configuration need not be difficult. Often, all we have to do is compare several situations using very simple, common-sense ideas.

Consider Fig. 3. At left, are four groups; the dots might represent individuals of different disciplines, while the lines represent mutual relationships. Each group is concerned with developing its own viable subsystem: the four viable subsystems are designed to be developed and proved in their own right without relating to the others—prior, that is, to final integration and overall test. Early NASA projects were of this kind, with lifters, orbiters, landers etc., being complementary projects, each developed by its respective, dedicated team.

At right is a similar arrangement, but one element from each project has been withdrawn to form a centralised project team. This is typical of centralised software development, area control centres for emergency services, and many others: it might be so organised for either or both of two reasons:

(i) to improve control and to enhance standards across all four parallel projects
(ii) to reduce manpower through economy of scale.

The effect of this centralisation on the four teams is sometimes overlooked—effectively, they have had a key discipline, and team member, removed. Further, since each team can no longer work towards completion, they can lose impetus and motivation. The removed member will find themselves working on (e.g.) software for any or all of the subsystems, especially where economy of scale was the driver. So, the centralised team risks losing identification, perspective and motivation.

So there are (at least) two ways of partitioning the system. Can entropy help us to compare them objectively? Entropy is related to the number of different ways in which N entities can be arranged. N objects can be combined one at a time, two at a time, three at a time, and so on. The sum of all the combinations is given, surprisingly perhaps, by the simple formula $2^N–1$.

Each of the small independent teams at the left of Fig. 3 could be arranged in $2^5–1 = 31$ different ways. The four teams can also be arranged in $2^4–1 = 15$ different ways. That makes a total of $4 \times 31 + 15 = 139$ different arrangements for the four independent teams.

The one large centralised team at the right of

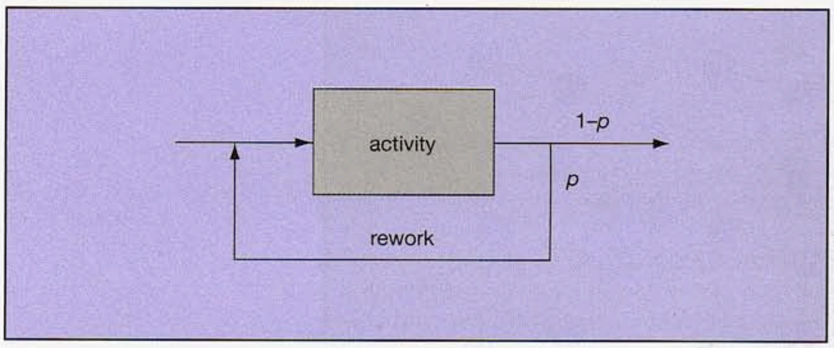


Fig. 4 Entropy and rework

Fig. 3 has 19 parts, allowing for the loss of one position for economy of scale. These 19 can be arranged in $2^{19}-1 = 525\,287$ different ways.

So, the right hand diagram represents a potential for much greater disorder. That does not mean that such an organisation *will* be disordered, but that it has the potential to be, and will become so unless measures are taken. These will normally amount to some form of regulation or control, involving effort and eating into any benefits expected from economy of scale.

Now, just looking at Fig. 3 could have told us that the four independent teams offered much simpler, tighter, ordered organisation than the single arrangement with the centralised facility. We really didn't need the mathematics, we could trust our instincts.

Yet, the calculation should make us think. Are we so easily seduced by specious arguments about 'economy of scale' and 'gaining better control' that we do not pause to think in terms of increasing the organisational-and-process entropy with its concomitant increases in risk, complexity-through-interdependency and timescale—not to mention increased management overheads and tighter control, leading inevitably to reduced motivation?

What does entropy look like?

Turbulent flow in process

It is useful to view an ideal organisation as affording a laminar flow of substance/information through sequential processes, e.g. the value-added chain. From this viewpoint, process is a series of sequential/parallel activities, organised in straight, parallel lines from entry into an organisation through to output from an organisation. The straight line affords minimum configuration entropy as there can be no disorder in pattern. This is a *streamlined* view.

Deviations from the straight line are turbulence. Turbulence causes drag, absorbs energy, increases entropy. The analogy with turbulence and drag in aerodynamics and hydrodynamics is compelling. Turbulence in process is caused by, for example:

- inadequate training/practice/tools, which make the line wobble as the inexperienced or ill-equipped hunt back and forth instead of pursuing the shortest route
- rework such that complete sections of the line need to be travelled again, and perhaps again; rework causes visible turbulence loops on an activity chart and N^2 chart
- hierarchy introducing delays while decisions are made. Hierarchy also produces visible turbulence loops on an activity chart and on an N^2 chart.

Fig. 4 shows a single activity, perhaps one of many making up a process, with rework occurring with probability p. If rework occurs, then the Figure represents a swirl, analogous to those in streams as water runs past a rock. And, in the same way, rework may go around the circle more than once.

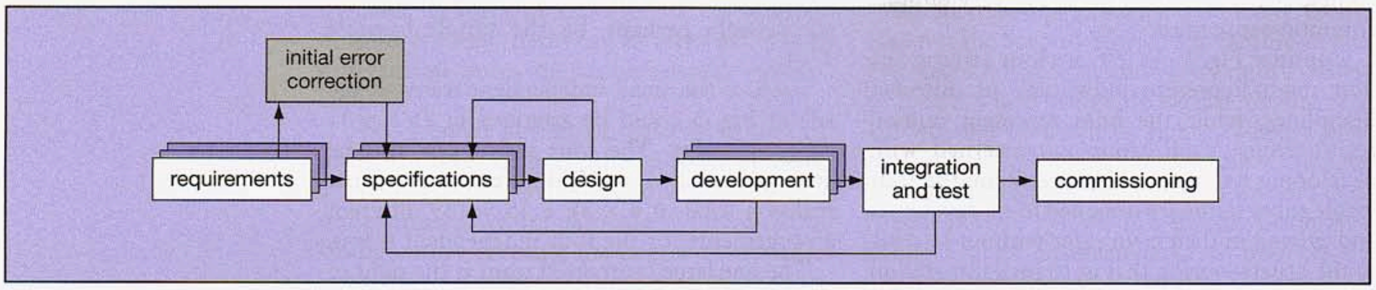
The probability of *no* rework is then $1-p$. If the activity were to take time T without rework, then it is simple to show that it would take time $T' = T/(1-p)$ with rework. Why are we interested? With $p = 0$ there is no turbulence, the process is streamlined, entropy is at its minimum, and the process would be optimum, at least in terms of efficiency and effectiveness. Rearrangement of the time relationship gives:

$$p = 1-(\text{direct path time}/\text{total time}) \qquad (1)$$

where direct path time is the time without rework and total time is the time with rework.

Fig. 5 shows a typical process model,

Fig. 5 Typical process model showing feedback turbulence



including the feedback that most models omit. At left projects typically start with a set of requirements, often numbering thousands. Not surprisingly these often contain inconsistencies, omissions, errors and ambiguities. Error correction usually follows, resulting hopefully in a set of matched system specifications for the optimum solution to the customer's and the business's needs. Design comes next and may result in some changes to the specifications or, in some organisations, a new set of detailed design specification.

Development commences along parallel paths and further problems may be found in the specifications as development reveals practical issues. These may cause revised specifications and design. Integration and test can similarly reveal problems requiring a return to the specification or design stage.

The whole Figure can be viewed as a series of swirls with work, rework and rework-upon-rework. Clearly, the greater the degree of rework, the more intense the swirls, the more turbulent the flow and the greater the entropy. It is possible for any one activity to be executed many times. Indeed it is possible for the complete process never to terminate. Is there some optimum entropy in such situations?

In an ideal world, there would be no errors, so no rework and the Figure would show no feedback. Work would move smoothly from requirements to commissioning. Fig. 6 shows typical consequences for various degrees of initial error eradication (IEE).

The Figure, of commissioning activity, shows the minimum time to complete the whole process as the 100% IEE line. Other lines for different degrees of IEE take longer. Reading from the graph, it takes 2·6 units of time to complete the 100% line (line 4) and 3·3 units to complete the 90% IEE line (line 3).

From eqn. 1, $p = (1-2\cdot6/3\cdot3) = 0\cdot30$. This means that the average level of rework, caused in this example by 10% of uncorrected errors in the specifications, is 30% throughout the project.

Once again, we should not be surprised by the mathematics. Common sense would tell us that allowing 10% of requirements errors to pass through into the rest of the systems engineering process is asking for real trouble. We would be bound to face rework at every turn, even rework of items already reworked. So, common sense would tell us that 10% errors in our specifications was a really high figure. Notice from the graph that the 70% and 80% IEE lines did not return to zero. Further analysis shows both lines continuing towards infinity, suggesting that the project would never terminate with these levels of IEE.

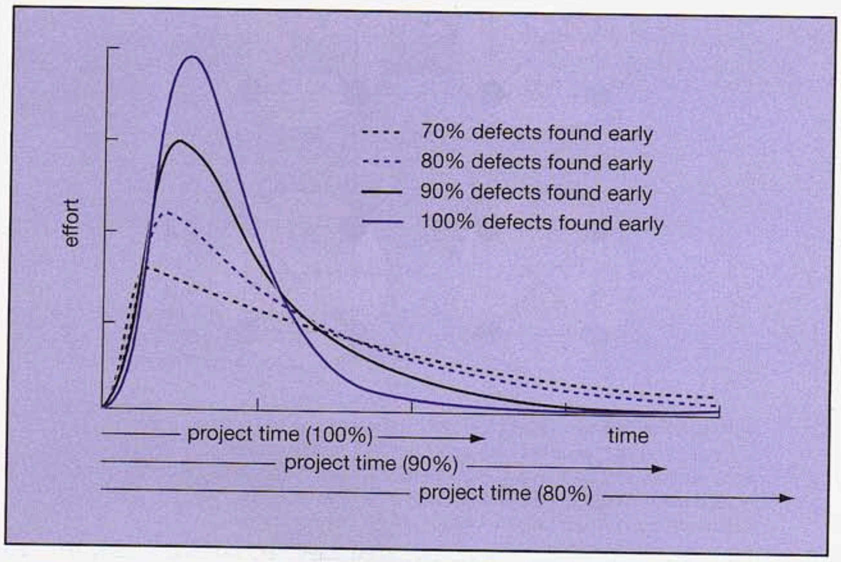


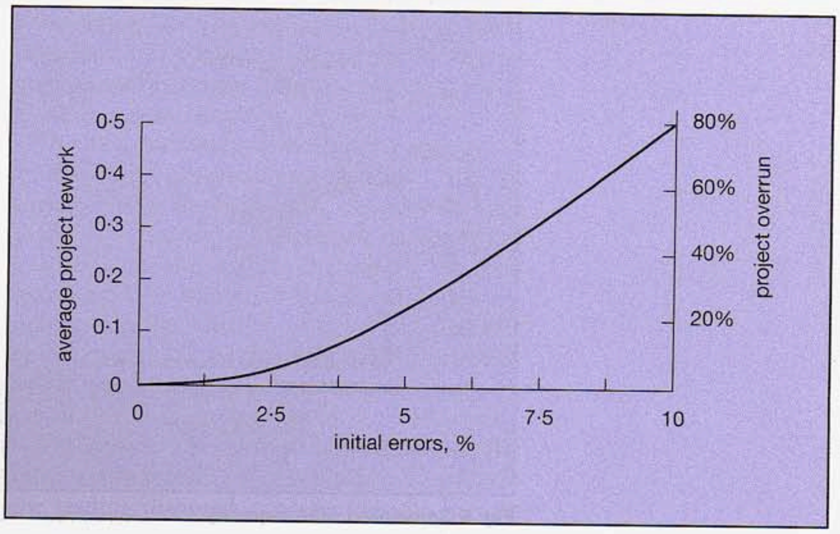
Fig. 6 Time to project completion for different error levels

Fig. 7 shows the effect that poor initial error eradication has on projects. For convenience, the graph plots errors remaining in specifications as the independent variable, and shows both project rework and project overruns as percentages above the zero initial error condition. From the Figure, it is apparent that:

- rework is significant for specification error rates of less than around 2·5%
- there is a high gearing between rework and project overruns.

Despite such findings, it is still common practice to curtail the requirements phase to 'get on' with the project. Curtailment invariably leaves requirement and design specifications with errors and omissions. Such projects may appear to be 'getting on', when it might be more

Fig. 7 Impact of initial errors on rework and project overruns



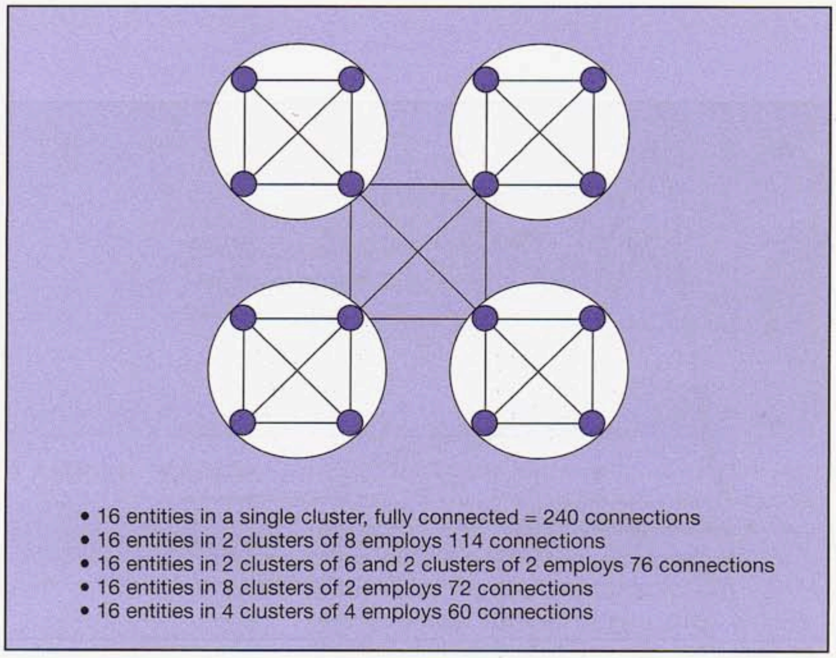


Fig. 8 Symmetry and infrastructure

appropriate to suggest that they are storing up major, self-induced problems.

Further analysis of this nature suggests that the very idea of phasing is suspect. Project phases, regarded as *de rigeur* in many quarters, divide a homogeneous project into sequential parts which may be virtually independent. This has the beneficial effect of reducing project entropy, but unfortunately it also prevents the whole process from being optimised as one system. So, management is simplified, but the optimum process is prejudiced. The example from the previous paragraph is one of many such de-optimisations incurred by phasing.

Perhaps, in the end, the optimum is to be found *only* at the minimum process entropy condition—no swirls, zero feedback. The Japanese automobile industry has tackled the zero rework problem using *Kaizen*, the philosophy of continuous improvement (see Reference 3). They accept no errors. Errors and defects are vigorously eradicated at source. It works very well for them!

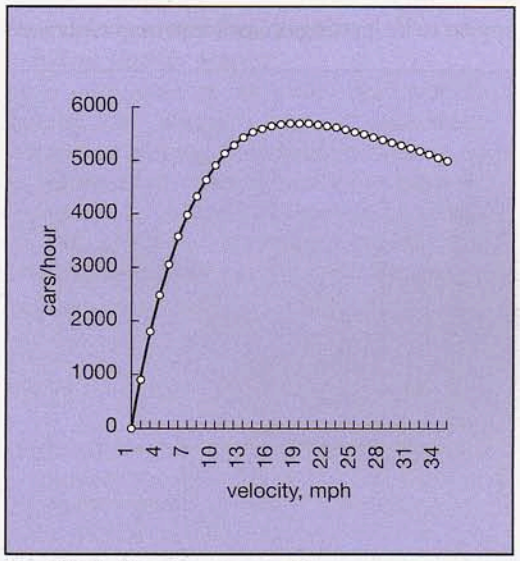


Fig. 9 Calculated road capacity

Finding the optimum entropy

There are many ways of seeking the condition of optimum entropy; this section looks at four archetypal approaches.

(i) Cumulative selection

In general:

- measure vital parameter(s) of the whole system while it interacts with other systems in their operational environment
- vary internal parts, observe effects on the whole
- change each internal part in turn to slightly improve the whole—this is Nature's cumulative selection; choose to vary whichever part makes the most difference, i.e. has the highest gearing (this is the method used to optimise the matrix in Fig. 10)
- continue until no more improvement is possible
- test to detect and avoid local optima.

These are all best done using some kind of model, rather than experimenting on a live system! Once the optimum condition is identified:

- set as a goal
- develop a strategy to achieve that goal.

(ii) Symmetry and optima

The process of partitioning a total-system design creates both parts and interconnections between those parts. There are many ways to divide any given system. One useful guide is to invoke symmetry in the process.

For instance, overall infrastructure tends to be minimal for symmetrical partitions. Fig. 8 shows 16 entities, each connected to their nearest neighbour. As shown, the 16 are grouped into clusters of four. However, it is possible to cluster the 16 in many different ways, as detailed in the Figure. Infrastructure is recorded as the number of connections between the 16 entities. The lowest number of connections coincides with the symmetrical arrangement of four clusters of four.

Happily, we humans seem to be born with a natural preference for symmetry observed, for instance, in our perception of beauty which is based on bilateral symmetry. So, our instincts for 'balanced' designs once again are likely to lead us naturally towards an optimum configuration.

(iii) Looking at the whole

The need to look at the whole system while seeking an optimum can be illustrated by looking at traffic jams. Using figures simply available from the Highway Code for braking distances at various speeds, it is possible to develop the graph in Fig. 9.

The graph represents the capacity of a three-lane highway in cars per hour passing some fixed point for various average road speeds. When traffic is stationary, the road capacity is zero, since no cars pass a fixed point. As cars pick up speed, the separation between them increases—each driver allows a greater breaking distance for higher speed. The result is a mathematical optimum at about 20 mph.

Knowledge of this theoretical optimum condition might be useful in several ways. For instance, by sensing the average speed, it is possible to gauge what proportion of maximum capacity is being employed at any instant. This leads to notions of anticipating and preventing jams.

For instance, if the mean traffic speed reduced to, say, 25 mph, it would be reasonable to anticipate the onset of jams. To prevent this, place traffic lights on the approach road and prevent further traffic from joining the highway until the mean speed picks up. After all, what is the point of letting more traffic onto a highway which is already on the critical edge of jamming?

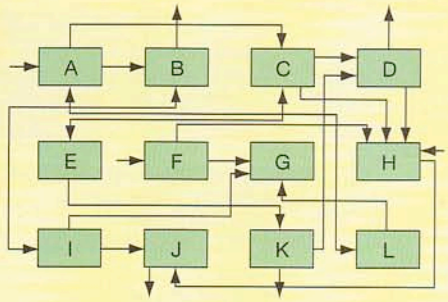
Well, you will have already guessed the snag. Preventing traffic from entering the highway means progressive tailbacks from the approach-road traffic lights, leading to off-highway snarl-ups and very likely to traffic being unable to *leave* the highway—gridlock!

It really is important to optimise the *whole* system.

(iv) Untangling connections

A major airline's base storage depot has 12 groups of spares and consumables arranged within a large rectangular hangar. Items are 'picked' from groups as needed. Individual 'pickers' visit one, two or more groups in sequence, according to the 'pick list'.

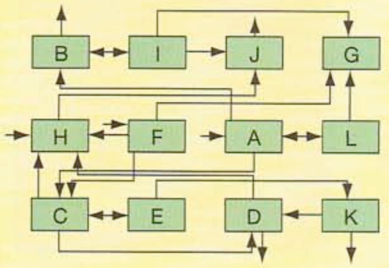
The pattern of 'pick movements' around the hangar is uneven, some types occurring more frequently than others. It is becoming apparent that a considerable amount of unnecessary travelling is occurring around the hangar, taking time and effort, and potentially causing a safety hazard. The rectangular hangar is the only suitable space available. Can anything be done to improve the situation?



- Figure shows rectangular room with 12 pick groups, A–L, and arrows showing principal movement paths

```
A 1 1                 3
  B           2
    C 1 1     3
      D       2
    3   E           3
          F 1 1
            G
              H   1
  1         2   I 2
                  J
      3             K
1           2         L
```

- Matrix represents path-length between pick-groups A–L. Numbers represent path utilisation e.g. 1 = low, 2 = moderate, 3 = heavy
- Travel index = Σ_i (path-length$_i$* utilisation) for i = 1 to 12
- Travel index from matrix = 160



- Figure shows pick groups rearranged to maintain original movement logic, but reduce overall travel index
- Paths form 'waterfall'

```
B 2
1 I 2 2
    J
      G
      2 L 1
1       3 A       1
      1     F 1
    1         H
              3 C 1 1
                3 E   3
              2     D
                    3 K
```

- Matrix rearranged to reduce overall value of travel index from 160 to 56, a real reduction of 65% in the work of travelling between pick groups
- Some separations increased, e.g. A to B, but overall path-length reduced from 79 to 36, i.e. by 54%
- Matrix score = f(entropy)

Fig. 10 Entropy, efficiency and effectiveness

Fig. 10 shows one approach to identifying the optimum configuration for this type of problem, which might equally arise when considering the layout of a factory floor, organisation of hospital departments, airport layouts, cells within command and control, gates on a logic board etc.

In the upper part of the Figure, the layout of the various logistic piles is shown at left, together with the pattern of journeys that are made between them. These are mapped into a matrix at the right. The matrix is then rearranged to minimise the travel index by rearranging the piles. Moving any pile makes it closer to another, but further from its previous neighbours. There are winners and losers—a classic optimisation problem.

The dilemma is resolved by rearranging the numbers so that they lie alongside the letters on the leading diagonal, representing the logistic

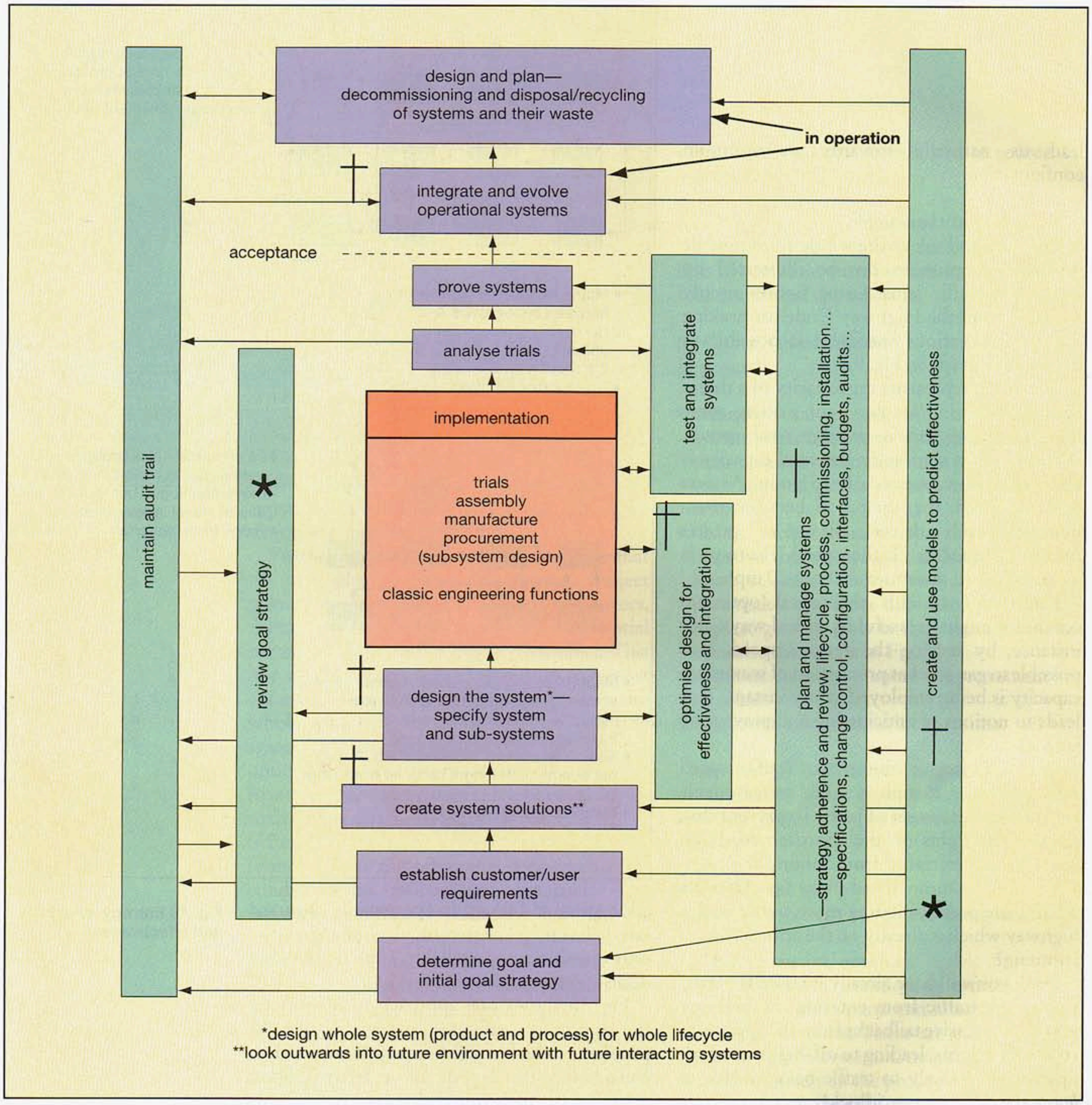


Fig. 11 Auto-adaptive life cycle systems engineering process model. Large asterisks indicate repeated strategic reviews. Daggers indicate repeated optimisation

piles. The closer the whole set of numbers gets to the leading diagonal, the shorter the overall distance travelled. Notice that the resulting matrix, bottom right, is tidier, neater and more symmetrical than the starting matrix above it.

Finally, the lower matrix is mapped back into the hangar floor, bottom left. Note the remarkable savings in wasted travelling. Note, too that there are losers. For instance, A and B were adjacent to start, but now are well separated.

Of course, this isn't the final answer—we have not considered the whole system. For a start, what about traffic congestion in the aisles—have we made it better or worse? Have we, perhaps, put things together which are safer apart? Rarely is a mathematical optimum the final answer.

Keeping systems engineering alive

We have explored the notion and value of the optimum system, and why the optimum state is short lived. We have yet to consider why, if systems engineering is so valuable, it periodically disappears. Is it, perhaps, an optimum system?

The reason may be to do with our very desire for perceived minimal entropy. If it is true that our cortexes seek to minimise perceived disorder, then it follows that we are uncomfortable with uncertainty, since that represents future disorder. One way in which we dispel perceived uncertainty is by producing detailed plans and process models for our projects. Project management is founded on GANTT and PERT charts, after all. And we like those plans to represent the future as prescribed and ordained—less than total certainty is uncomfortable. Certainly, just demonstrating that true systems engineering, with its inherent adaptability, affords the shortest time to market, the optimum combination of quality, efficiency and effectiveness etc. seems to make little impact. We are locked into a reductionist paradigm.

How, then, can we accept true systems engineering when it seeks optima that are continually changing, when it continually revises plans in the light of changing circumstances and new knowledge about, *inter alia*, other systems in some future-uncertain environment?

Surely we must cling to our faith in fixed-price contracting, prescribed value-for-money projects with carefully-calculated costs and timescales? After all, our faith has remained unshaken through countless timescale and budget overruns.

How, above all, can we maintain control? For many of us, if the truth were told, this last is the real cruncher—fear of loss of control overrides all other concerns. For systems engineering, fear of loss of control means that systems engineering must be prescribed, regulated, controlled...to the point where it is no longer systems engineering at all, but some pale, ineffectual systematic rote shadow of the real thing.

One way of reconciling the control freak's reductionist panic with true systems engineering is represented at high level in Fig. 11. At first glance, the Figure may appear just like any other life-cycle or process model. But, look more closely.

The very first activity, centre bottom, requires the determination both of the goal and of the goal strategy—the means, route or way in which the goal is to be pursued. (As we shall see, there are several archetypal strategies one of which was represented in Fig. 2.) Next, observe the column second from the left—review goal strategy. Since the vertical dimension represents time, this column indicates that the goal strategy is in continual review for much of the project's early period. Similarly, 'dagger' symbols point to repeated optimisation. So, the variability essential to true systems engineering has been built into a process model which should, as a result, continually adapt itself to optimise overall process performance. Not surprisingly, this model passes all the systems engineering litmus tests too.

Of course, there is more to it. Fig. 11 does not indicate any particular goal strategy, and there is a variety of archetypal options. These archetypes include:

- waterfall
- Sashimi[4]
- spiral
- evolutionary
- evolutionary acquisition
- regression
- chaos[5]
- concurrent
- goal-oriented[6]
- etc.

So, we have to choose the optimum strategy to achieve the goal, and that choice will remain optimum only for a period limited by the ever-changing situation about us.

Difficult? Demanding? Not what we are used to? Yes. But if we want to win, then we cannot afford to stick with our static, methodical, rigid approaches in an increasingly-turbulent world. We cannot afford to let systems engineering disappear like TQM, BPR and all the other TLAs. After all, systems engineering is not a method or a technique—it is a philosophy and a way of life. And one that commerce and industry needs to embrace wholeheartedly if it is to compete at world level.

References

1 BAK, P., and CHEN, K.: 'Self-organized criticality', *Scientific American*, 1991, **264**, (1)
2 HITCHINS, D. K.: 'Putting systems to work' (John Wiley & Sons, Chichester, 1992, pp.209–210)
3 WOMACK, J. P., JONES, D. T., and ROOS, D.: 'The machine that changed the world (Rawson Associates, New York, 1990)
4 DEGRACE, P., and STAHL, L. H.: 'Wicked problems, righteous solutions' (Yourdon Press, 1990)
5 RACCOON, L. B. S.: 'The chaos model and the chaos life cycle', *Software Engineering Notes*, 1995, **20**, (1)
6 HITCHINS, D. K.: 'Systems engineering the pyramids' (IEE Master Class, 1997)

Professor Hitchins is an IEE Fellow.