10

SM5: Architecting/Designing System Solutions

Sir Christopher Wren
Said, 'I am going to dine with some men.
If anyone calls
Say I am designing St Paul's'

Edmund Clerihew Bentley, 1875—1956

Approach

System design is sometimes viewed as an esoteric, even arcane, practice: so much so, that many teachers, references and books no longer refer to system design, choosing instead to talk about 'architecting,' suggesting, perhaps, that design and architecting are substantially the same thing, which may not be entirely correct.

System design imbues a product, the design set, with the essence of systems engineering (see 'The essence of systems engineering on page 120). We have seen, too, how the Generic Reference Model is able to provide a design template, such that requisite emergent properties may be forthcoming (see also 'Does the GRM capture emergence?' on page 141).

To manifest particular emergent properties, capabilities and behaviors in a particular solution system, select and bring together particular functional parts and cause them to act and interact in a cooperative and coordinated manner, invoking requisite synergies. Instantiating the GRM for a particular solution system creates the 'internal' functional architecture, or platform, to select, activate and coordinate the many and various mission functions—see Figure 10.1.

If this seems obscure, then consider the analogy of a man crossing a busy road: his mission is to cross — in one piece. He uses his eyes and ears (sensors) to collect information, chooses (decides) to dodge between traffic (strategizes) rather than use the pedestrian crossing, waits for a gap in traffic (plans) and safely walks across the road (execution) with the tacit cooperation of the motorists — who do not actively try to run him down. The *execution* of the plan requires that

Systems Engineering: A 21st Century Systems Methodology Derek K. Hitchins
 ISBN: 978-0-470-0585-5

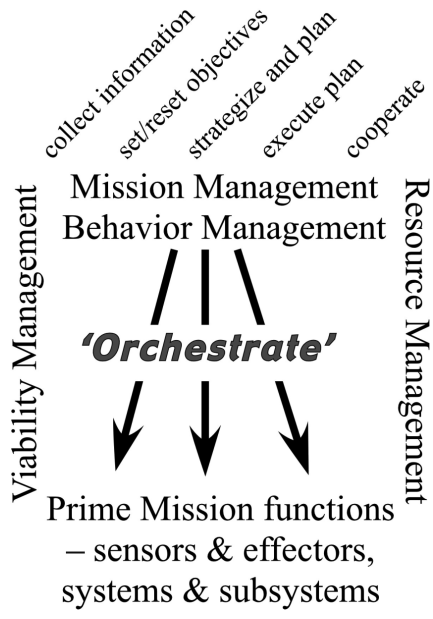

Figure 10.1 Function management (i.e., mission, behavior and resource management) 'orchestrates' prime mission functions, activating them in sequence and in parallel in accordance with the plan and in accordance with the CONOPS.

the brain scan for a gap in traffic, initiate locomotion, control limb movement, maintain balance, and continually scan sensors (primary mission functions); this is orchestration of prime mission functions, which are activated in series/parallel according to the plan (see Models of Systems Architecture on page 108 for functional architecture in the brain).

By analogy, the GRM Function Management corresponds to the cognitive brain and the central nervous system; it is the foundation, the managing platform, enabling the properties, capabilities and behaviors of the whole to emerge. Prime Mission Functions (PMFs) correspond to the sensor sweeps, and to capabilities for movement in an appropriate direction, at the right moment, the ability to dodge, etc., all of which enable the man to cross the road safely, under the direction of his intellect, and in accord with his concept of operations. Function Management activates, correlates and coordinates these PMFs.

A somewhat different analogy might be with a symphony orchestra, with strings, brass, woodwind, timpani and percussion, etc. An orchestra performs a musical piece by following a score, with each of the players having their respective musical part appropriate to their instrument in front of them. The actions of all the players may be synchronized and coordinated by the conductor, but the orchestration is predetermined, so none of the players is directly controlled in his playing by the conductor. Nonetheless, beautiful music may emerge from the whole.

Comparing the two analogies shows that the term 'orchestration,' as used in Figure 10.1, can be viewed in two ways: for the man crossing the road, the brain planned and orchestrated his sensing and movement in real time, using the central nervous system for motor control, sensory feedback, etc. In the second analogy, the orchestration was preplanned and pre-allocated to the various sections within the orchestra, so that each player was able to play, in principle, on his or her own and in isolation. The whole was then drawn together, synchronized and coordinated by the section leaders and by the conductor.

This difference in approaches to orchestrating prime mission functions is to be found in many systems and situations. In the conduct of warfare, for instance, some European military favor a method called Mission Command, or *Auftragstaktik*, in which an overall commander allocates a set of missions to various subordinate commanders, who subsequently plan and execute their respective missions without supervision or communication with the overall commander. This has the advantage of allowing communications silence, so depriving the enemy of potential intelligence. Each subordinate commander is also aware of the overall commander's intentions, and so is able to modify, or even deviate from, the allotted mission as situations develop, while still trying to achieve the commander's goal. Control is effectively delegated, trust is afforded to subordinate commanders, and they are able to develop their capabilities — and make mistakes.

An alternative form of military command employs continual real-time control of operations, with the overall commander — who is often remote from operations — controlling what is going on; in the US, this has been dubbed 'President to foxhole:' control is centralized. In this scenario, subordinate commanders have less authority and discretion, and there is a heavy dependence on communications, on which real-time control inevitably depends. The potential advantage of this approach is that potentially sensitive decisions are taken by the senior man, rather than by a subordinate commander who may not be *au fait* with political sensitivities. The obvious weaknesses are: (a) that the system breaks down if communications fail; and (b) over time, it undermines the chain of command and the authority of officers within that chain.

Instantiating the GRM requires an understanding of the CONOPS, and the developing nature of the solution system, as evinced by the prime mission functions. For example, a military platform as (part of) a solution system might expect to carry a variety of different weapons, both offensive and defensive, associated with prime mission functions, self-defense, etc. Instantiating the GRM for the platform as a solution system could then throw up a Weapons Management function under the Mission Management heading, and a Rules of Engagement function under the Behavior Selection heading. Why? Because these would be necessary platform functions to 'manage' and coordinate prime mission functions; and, it is on this coordinated interaction and cooperation between functions that emergence is founded.

Similarly, examining a range of prime mission functions for an enterprise might indicate that some are concerned with seeking and anticipating threats, or reducing risks, perhaps by situating the solution system. If so, one would expect to see a marketing, or intelligence function under the Mission Management heading in the instantiated GRM. In general, there are likely to be functions under Mission Management, Resource Management and/or Viability Management that correspond to Prime Mission functions, and which are able to orchestrate these prime mission functions.

The Functional Design Process

So far, in previous steps of the systems methodology, we have identified a range of potential CONOPS and prime mission functions (PMFs) that the solution system will need to perform. The mission features will make clear the nature of the solution system, i.e., what kind of system we are intending to design. The 'external,' or visible functions performed by the solution system find their counterparts among the internal functions of the system. We have yet to identify the many and various functions, which may be thought of as 'internal to the solution system,' that will tie the whole system together, and will cooperate and coordinate, establish and maintain system coherence and viability, etc. We know from the GRM what many of these are, generically speaking; part of design is instantiating, or giving substance to, these generics — see GRM Function Management on page 128, *et seq*.

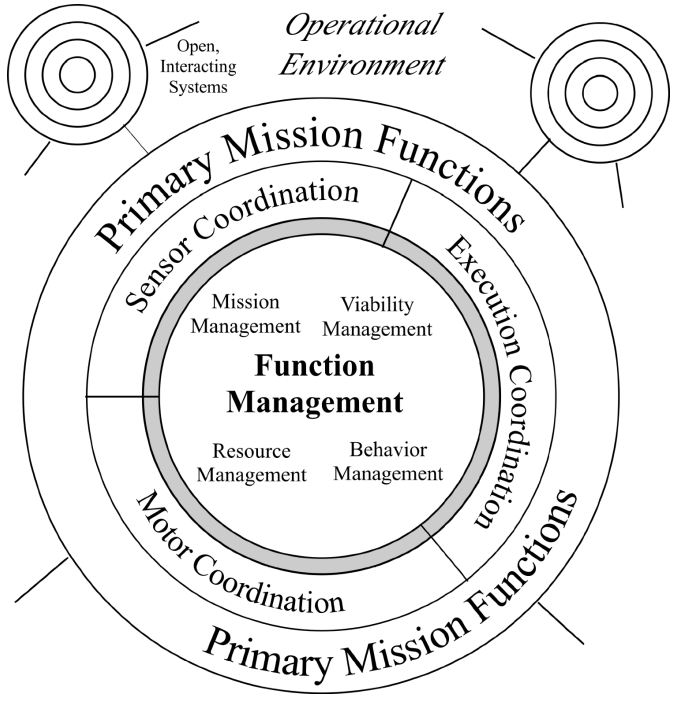

Figure 10.2 Functional architecture concept. Executive commands from, e.g., mission management may be elaborated into a set of activities, a routine, and progressed by 'execution coordination.' Similarly, routine sensor and motor coordination can be delegated, such that primary mission functions can be orchestrated without overloading mission management with detail.

A useful design starting point is to develop a specific version of the GRM, initially in tabular form — see page 133. This process generates all of the elements required for mission management, resource management and viability management. Similarly, it generates elements for cognition management, behavior selection management and behavior stimulation management. The elements are functional subsystems, and will be given names consistent with their function.

So, a functional subsystem that manages internal and external communications might be called a 'communications center' in one context, something else in another; a functional subsystem that senses remote activity might be named 'sensor suite;' a functional subsystem that detects and anticipates competitors activities might be called 'intelligence subsystem;' and so on.

These functional subsystems can be analogous to the 'mirror neuron templates' in the brain, which can initiate, coordinate and interpret series of sensor and motor activities 'subliminally,' so relieving the central, conscious brain from routine processing (see page 110 *et seq*.) This, then, presages the development of functional architecture, with a central functional intent (prosecuting the mission) activating surrounding functional subsystems, which, in their turn, activate and coordinate 'peripheral' Prime Mission Functions. Such a 'tiered' functional architecture minimizes mission management effort, allowing many different missions to be carried out at the same time, if need be, see Figure 10.2.

The overall functional design process is shown in Figure 10.3. In addition to the sequence of functions already discussed above, the figure also shows the tools and the products or 'deliverables'

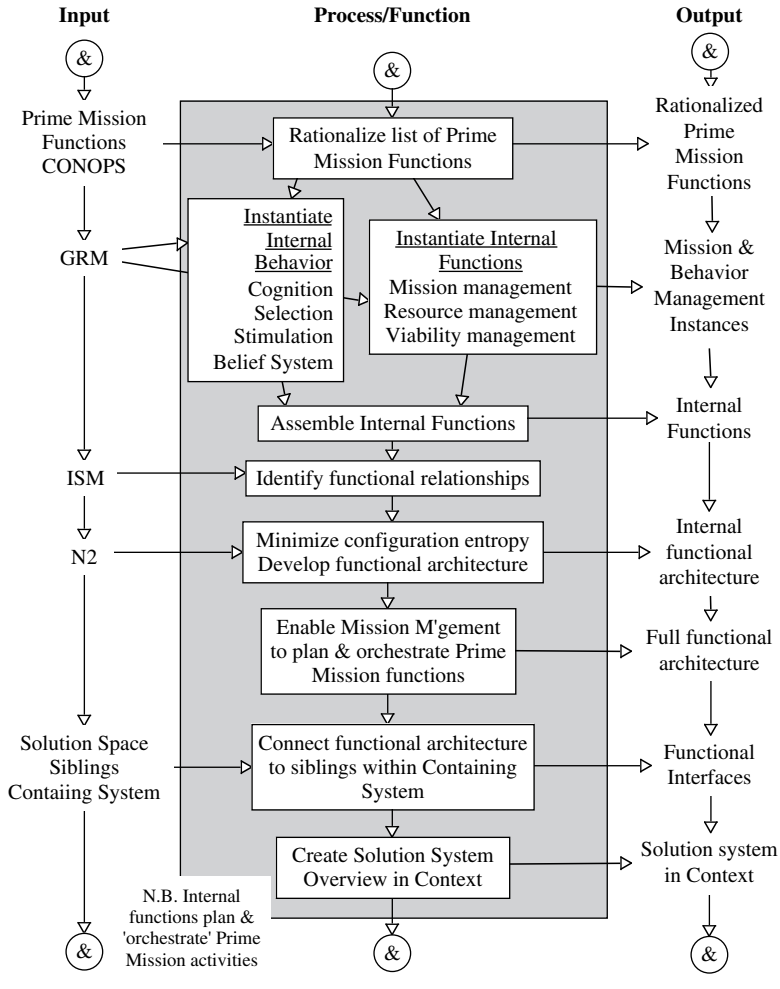

Figure 10.3 Functional design.

that form part of any behavior diagram. As the inputs at left indicate, the various methods and tools employed in design may be 'handraulic,' i.e., developed on paper or on a computer screen, by hand—as, indeed was Figure 10.3.

However, there can be significant advantage, particularly with the design of large or complex solution systems, in using computer-assisted tools. John Warfield's ISM, for instance may be enhanced in use with computer support, and the ubiquitous N-squared (N^2) chart may be automated to significant advantage. In the latter case, the use of clustering algorithms, hill climbing routines and/or cumulative selection methods enable the configuration entropy of a disordered N^2 chart to be minimized: in the process, errors and omissions are highlighted, and clusters are revealed which form the basis of architecture (see Hitchins, 1992, 2003).

Contrary to conventional wisdom, functional architecture and, as we shall see, physical architecture, do not have to be separately designed, or — as some preconceived framework — superimposed on the solution system. Architecture emerges naturally from the problem space, the solution system concept, CONOPS and prime mission functions, and almost as a byproduct of the design process. That is not to say that architectures cannot, or should not, be superimposed, say, for reasons of interoperability, compatibility, damage tolerance, etc., or to reconcile solution space constraints. It is to suggest, however, that superimposed architectures need not be optimal in the context of solution system performance and effectiveness.

The Physical Design Process

The physical design process, outlined in Figure 10.4, involves mapping, or folding the functional architecture, with its subsystems and interactions, on to/into some physical substrate. The solution system is to be manifested as a contained whole, body, organization, enterprise, process, etc: in

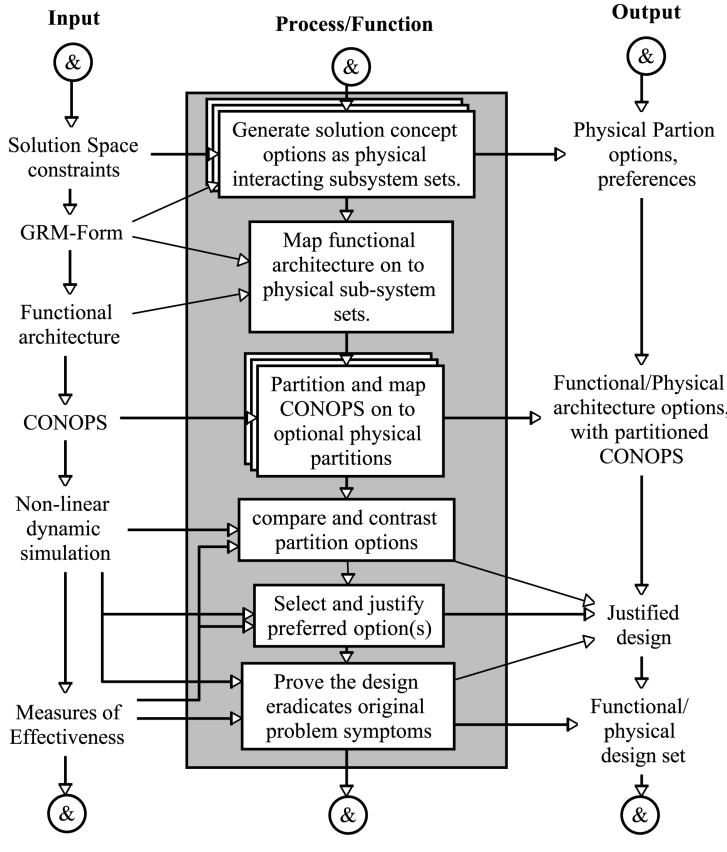

Figure 10.4 Functional/physical design — developing a functional/physical design set.

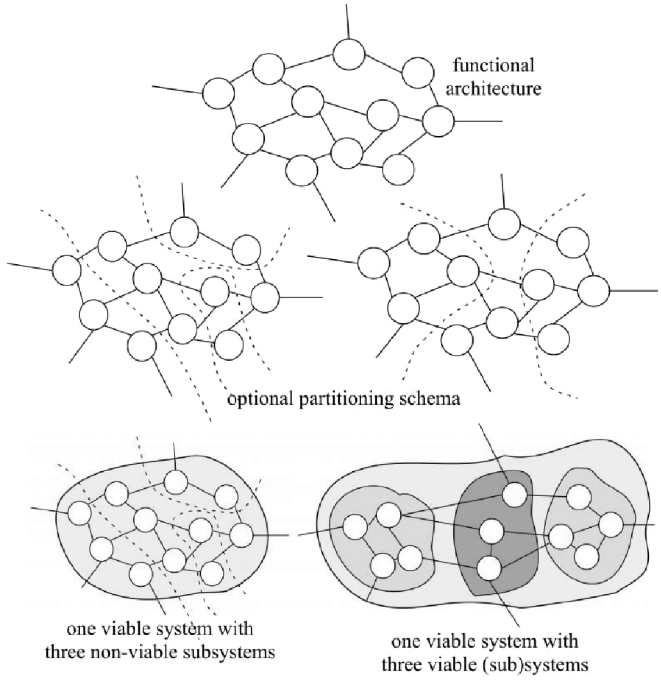

Figure 10.5 Alternative partitioning schema. Complex systems may be partitioned without prejudice to their overall functional behavior, with a view to their future organization or creation. There can be many, different partitioning schema for a complex system. Partitions may contain nonviable subsystems or may, on the other hand, be realized as discrete open, interacting viable systems within a Containing System, which will interact, with other Containing Systems, not shown: this is the development of hierarchy.

the general case, however, a variety of separated parts may be joined to each other by interaction pathways to facilitate behavior as a unified whole.

As Figure 10.5 shows, there may be many different ways to partition a system — in principle the more complex a system, the more ways. To some extent, systems partition themselves, with some functions clustering together owing to more connections between members of the cluster, and fewer with member of other clusters; such functional clusters are candidates for physical association, i.e., for constituting a partition, where closeness facilitates the higher degree of mutual interaction in the cluster. There may be other reasons for partitioning: the overall system may be large and distributed, so that partitions become more manageable; some groups of functions may be quite different in nature from others, suggesting quite different approaches to realization; and so on; and the constraints of the solution space may dictate that the whole system be created as interconnected parts, each of which is able to fit within local constraints.

With so many drivers, it is not surprising that there may be many different partitioning schemes. As Figure 10.5 shows too, there may be two quite different results from partitioning. In one case, the partitions exist within the one system and consist of nonviable subsystems, i.e., subsystems that could not exist, flourish or survive on their own. An example of this might be the human body which has many closely coupled subsystems: cardiovascular, central nervous system, pulmonary system,

immune system, etc., etc. Another example in the same vein (!) might be a multi-band analog double super-heterodyne radio receiver, with modular power supplies, radio and intermediate frequency amplifiers and detectors, frequency synthesizers, frequency control loops, etc. In both instances, the many and varied, complementary subsystem parts are closely coupled, mutually interdependent and nonviable, i.e., they cannot function independently.

Alternatively, the result from partitioning might be a number of interconnected, viable subsystems, each capable of existing and operating on its own. An example of this approach was to be seen in Apollo, where the various subsystems were the various craft: rocket, command module, lunar excursion module, etc. While they all fitted together and interacted, each was a viable system, able to exist and function on its own; however, all were needed to fulfill the CONOPS, and, indeed, each of them was able to fulfill part of the overall CONOPS on its own. In effect, the overall CONOPS was partitioned, too, so that each physical partition was allocated a phase of the CONOPS.

A second example of partitioning into interconnected viable subsystems might be a large corporation comprised of a headquarters and a number of divisions, each operating as a separate business with its own marketing, production, sale, etc. Each division may manufacture different parts and products, say, for the aerospace, or the transport industries, such that the various product outputs from the divisions are complementary, with few overlaps. For vehicles, one division might make engines and gearboxes, another transmission and suspension, another bodies and interiors, and so on.

Importantly, the emergent properties, capabilities and behaviors of the whole, are (should be) unaffected by the partitioning into viable subsystems, since both prime mission functions, and the functional architecture mapping, should be unchanged — see Figure 10.4. On the other hand, in creating a number of viable subsystems, i.e., systems in their own right, the whole can now be seen as a Containing System, with its own complementary variety of viable subsystems. Failure of one viable subsystem does not cause immediate failure of another, since they each have their own function, behavior and form management capabilities, and indeed their own CONOPS.

When partitioning into viable subsystem, then, an opportunity presents itself to consider each of these subsystems, with its respective CONOPS (part of the overall CONOPS) as a discrete, open, conceptual remedial solution system, potentially operating in a future environment consisting largely of the other viable subsystems. It may be sensible to iterate the systems methodology, from SM3: Focusing System Purpose; through SM4: Develop CONOPS; and up to SM5: Design; for every viable subsystem, each of which will merit its own instantiated GRM — see Figure 10.6.

Indeed, for a very large and/or very complex system, it may be appropriate to iterate the systems methodology at several levels of this emerging hierarchy of systems within systems within systems . . . and this is what happened, effectively with Apollo and with many other major defense and aerospace projects. This works in practice by creating a 'top-level' systems engineering team, concerned with the whole mission system. Additionally, each of the major subsystems, which may be realized as independent projects running in parallel, has its own systems engineering team, members of which also participate in, or subscribe to, the top-level systems engineering team activities, designs and decisions.

This ensures that any changes, developments, etc. arising at the top level can be cascaded down the hierarchy, and *vice versa* — any unavoidable changes in any one of the major subsystems can be elevated to the top level to see if they would affect interactions and/or the whole. Changes in any subsystem which did not affect its emergent properties, capabilities and behaviors may prove unimportant; changes that did affect subsystem emergent properties, capabilities and behaviors, on the other hand, might lead to major redesign, rebalancing, etc., and could potentially affect the whole. So, each level in the hierarchy had its systems engineering teams; the corresponding systems

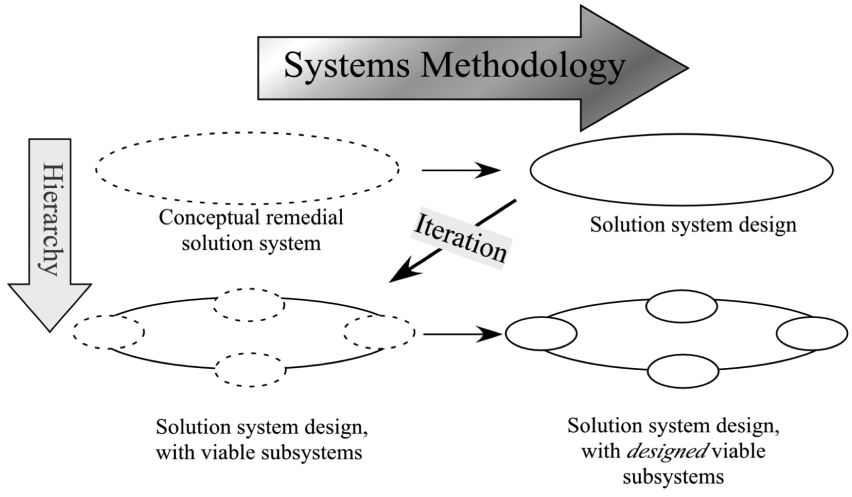

Figure 10.6 Iterating (part of) the systems methodology to accommodate developing hierarchy.

engineering teams were tasked with ensuring that the requisite emergent properties, capabilities and behaviors did, indeed, emerge on schedule, and to the satisfaction of the hierarchy level above.

Output/outcome

Whether or not iteration is appropriate, the end result of SM5: Architecting/Designing the Solution System is an arrangement, or configuration, of interconnected, functional parts. The emergent properties, capabilities and behaviors of the whole SoS, and of each of the subsystems and their interactions, may also be specified. These emerge principally from the nonlinear dynamic simulation of the partitioned SoS as an open, adaptive system interacting with other (simulated) systems in their mutual environments; this simulation is necessary to prove that the functional/physical design resolves (rather than solves) the original problem by eliminating all of the original symptoms emanating from the problem space. It is also used to compare different partitioning schemes. And, when it is proved the design fulfills its purpose, the various parameters, values and structures can be used to specify emergent properties, capabilities and behaviors.

The output from SM5, then, is architecture in the sense of pattern and paradigm, but it is more than that — it also identifies and defines purpose, function, relationship, association and separation, interaction, hierarchy, and emergence. Moreover, by virtue of its derivation, it is largely an Ideal World representation, except in one respect: it is not an optimum solution system design.

To understand why not, consider the orchestra analogy again. Members of the orchestra play their instruments according to the written score, and if their playing is coordinated, the result is music. However, it may not be the best music. To achieve excellence, the conductor has to interpret the score, and this interpretation emerges, partly, as *balance* between the sections of the orchestra. Left to their own devices, the brass may drown out the strings, or the timpani may overshadow the woodwind, and so on. The best, or optimum, performance from the whole emerges when the outputs from the various sections are balanced dynamically as well as coordinated rhythmically. So

it is with systems in general — the best performance, the most effective system, and the greatest degree of emergence, arise when the various system parts cooperate, coordinate and balance in operation. So, while the *basis* for emergence has been established by the orchestration of prime mission functions, the optimum *degree* of emergence has yet to be realized.

A second analogy may help to make the point. Consider a Formula One racing car, and suppose we wished to create a new one by bringing together the best parts from the various racing car makers. We might employ the best suspension from Ferrari, the best engine from Renault, and so on. But, would the resulting, new, racing car, automatically be the best? Not necessarily: the various major components must operate in harmony with each other, so that propulsion does not overcome traction, so that weight distribution does not prejudice suspension and cornering, and so on. Again, dynamic balance is the key to getting the best out of the system . . . And we should not forget that the car is 'tuned,' not only to give the best performance on a particular track, but also very much to suit the individual driver. The best car on the day combines the purposeful skills of the driver with the capabilities of the vehicle: a great driver may win with a less-than-perfect car; and *vice versa*.

The last function/process on Figure 10.3 requires justification of the selected partitioning option. This would be achieved ideally by further dynamic simulation modeling, to show that the solution system design, if realized, would neutralize all of the original problem symptoms.

Summary

Chapter 10 has addressed the conceptually difficult issue of systems design, so often overlooked in texts, papers and handbooks as 'too difficult.' Undoubtedly, design of a complex solution system can be a significant task, but it can be undertaken systematically and comprehensively, using the GRM as a design template. In the process, system design does, indeed develop functional and physical architectures, but as an outcome from the processes rather than as an arbitrary input. Architecture is, at its most fundamental, clustering and linking: clusters form during the design process by closer association between some functions (binding) than others; links between clusters (coupling) then appear as a natural outcome. As linkages are instantiated in the design, functional architectures appear without conscious effort. Physical architectures, on the other hand, may be imposed in part by constraints within the solution space, and by other systems with which the solution system will interact. In this text, the term functional/physical architecture is preferred, to plain physical architecture. It is vital, in mapping functional architecture on to physical architecture, to maintain the linkages and interactions between all functions such that full and effective orchestration of the prime mission functions is maintained — else, requisite emergent properties, capabilities and behaviors will not be evinced, or worse, counterintuitive and unwelcome properties, capabilities and behaviors may emerge. Something as simple as inadequate capacity of, or noise on, a seemingly minor communication link, can radically change the behavior of a whole solution system. Similarly, the smallest timing error can prevent coordinated behavior, decoding, synchronized synergistic actions, etc.

Partitioning can lead to the creation of hierarchy, with viable (sub)systems within viable systems. A viable subsystem is one with its own function, behavior and form management. Partitioning an overall system into viable subsystems should not, in principle, affect the functional behavior of the whole. On the other hand, it can mean that failure of one viable subsystem does not materially affect other subsystems, although of course it may impair overall functional behavior and capability. Where partitioning results in a set of open, interacting, viable subsystems within a

Containing System, it may be appropriate to repeat the systems methodology for each and every viable subsystem, considering it to be operating in an environment consisting largely of its sibling, viable subsystems.

N.B. Understanding the practice of system design may prove difficult when it is presented in the abstract form of behavior diagrams, no matter how informative they attempt to be. The best way to understand the processes and methods is to see them at work in examples. Case Study D: Architecting a Defense Capability, is an exemplar of the systems methodology in operation.

Assignment

You undertake the system design for a new, inner city school for children aged 11–18 years, as follows:

- Propose a CONOPS for the school: how is it going to operate, from where will it attract pupils, what will be their age range and ability, what subjects will be taught, what style of teaching will be employed, what will be the students' qualifications on leaving, etc.
- Establish a Prime Directive for the school, and employ the TRIAD Building System, or other means, to propose a set of Prime Mission Functions for the school, including PMFs to address risks and threats to the school, and its achievement of purpose. Limit the number of PMFs to seven.
- Instantiate a Function Management Model (Mission, Viability and Resource) and a Behavior Management Model for the school, using the list form of the models. Include within the lists, instances of functional subsystems to orchestrate the execution of PMFs.
- Using ISM, or similar, identify relationships and interactions between the instantiated Function and Behavior Management subsystems, and develop the resulting subsystem-interface information as an N^2 chart.
- Configure the N^2 chart to minimize entropy, i.e., reveal clusters, links and functional architecture
- Develop the architecture from the N^2 chart as a functional block diagram, connecting relevant subsystems to their corresponding Prime Mission functions.
- Connect the whole to other systems (education administration, social systems, etc.) to form an overview of the school within its future environment.
- Hence, create a design for the new school.
- Penultimately, consider whether, or not, the design that you have created is simply a reflection-in-memory of your own school, or if there are new, and perhaps unexpected, elements in your design.
- Finally, propose measures of effectiveness for the new school, and justify your design.