Notes for a presentation at the inaugural SESA conference, Sydney, 6 October 1995

Systems Engineering as a Design Process

The current view of Systems Engineering (SE) is that it is (primarily) a *management* methodology. This is reflected in the NCOSE Symposia and in most textbooks on SE, and it is probably the view of most of you present here today. The objective of my presentation is to show you that SE is also a methodology for *designing* complex systems. That is, a methodology for solving complex design problems and a process for transforming requirements into solutions. To understand how SE can be both of these things, we first need to take a step back and recall the fundamental aspects of SE. We shall also need to reconfirm that design really is a process in itself, not just management of people with engineering science knowledge.

Quite generally:

- A *system* is a description of something (anything) complex in terms of interacting parts.
- *Systems Engineering* is a methodology for woeking with systems, i.e. for handling complexity, and has three main features:
  - it is step-wise (top-down)
  - it contains rules for the partitioning
  - it contains rules for how to handle the interactions

Useful analogies are:

- Numbers (describing quantity), and arithmetic as the methodology for working with numbers.
- Groups (describing symmetry or invariance), and group theory for working with groups.

Abstract concepts are only interesting if they are useful, i.e. if they can be applied so as to produce a benefit. Let us now look at the application of the system concept to engineering projects. An engineering project involves three entities:

The project,
the physical system; and
the service.

Each one of these entities becomes complex for large projects, and each one can be described as a system. The elements are, respectively:

Work packages;
equipment and/or functional elements; and
functions/characteristics.

Three points must be noted here:

a. The physical system can be *described* by a system which consists of either equipment (i.e. by what it is) or functional elements (i.e. by what it does), or a combination of both.
b. The word “system” now has two meanings; the colloquial use in “telephone system” or “solar system”, and the meaning of “functional (abstract) description”.
c. The description of the project is *always* physical, of the service *always* functional, and in the case of the system we have a choice.

The purpose of top-down design is to partition the complex design problem into a set of less complex, but interacting, design problems. Each of these can then be solved by conventional, bottom-up design. The end result of the complete design process is always a physical entity. That is, at some stage se have to make the transition from the functional into the physical

domain; the balance between top-down and bottom up design effort depends on the complexity of the problem,

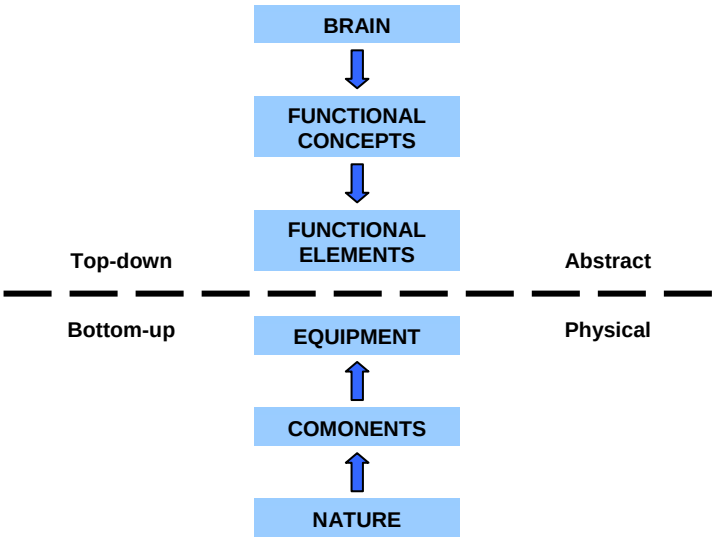
The application of SE to the project (i.e. to the work) is well known to all of you; it is reflected in such concepts as WBS, configuration management, engineering specialty integration, and concurrent engineering. Note that this application involves a breakdown (structuring) of both the work and the physical product. The application of SE to *design* is much less well developed; it is mostly limited to managing the design process. The reason why the application of the SE methodology to design (i.e. top-down design) is difficult is that it *must* take place in the *functional domain*.

The best way to understand what top-down design entails is to compare with traditional, bottom-up design. In the latter, we start out with a set of physical elements – *components* – which are then combined to provide an entity that will have the desired performance. The best combination is the one that is most cost-effective. In top-down design, we start out with an entity, the set of requirements on the performance, and subdivide this into a set of *functional elements*. Again, the best subdivision is the one that is most cost-effective; i.e. that results in functional elements that can be realised in the most cost-effective manner.

But imagine how cumbersome bottom-up design would be if you had to design each component each time you needed one. To design a bolt and nut would take a day! But that is what we are doing in top-down design; we reinvent our functional elements (models) every time.

What we need are standard functional elements

Let us push the comparison between bottom-up and top-down a little further. In the former case, the elements (components) are developed from our knowledge of the laws of nature and the properties of matter, and our understanding of the behaviour of those elements is based on that knowledge. But in the case of top-down design, on what is our understanding of functionality based? Where do the concepts that enter into defining functionality come from? They may come form the way in which our brain works (ref. N. Chomsky on the origins of grammar), and we have the following picture:



The approach to developing the functional domain is again an aplication of the SE methodology; actually, a two-fold application. On the one hand, we start with the most general (the class of all systems) and proceed downward to more specialised (but therefore smaller)

classes of systems. On the other hand we start with the most general parameters for describing the class and proceed to develop these parameters in more detail. Take, for example, the class of all (engineered) systems. Any system must have a purpose, i.e. have a functionality, which resutls in its *service*. This service must has a *value*, and with the system is associated a *cost* of providing the service. So we have, at least conceptually, identified two parameters.

But how can value be defined in an operational sense, i.e. so that we can do something with it? Well, we could define it as the price people would be willing to pay for it. But who are "people"? This line of thought leads us to the realisation that a (functional) system exists only in relation to a *user group*. This again emphasizes how top-down and bottom-up are "conjugate" methodologies, and why the top-down methodology is called "requirements driven".

We are at the very beginning of the development of the functional domain and of top-down design, as the comparison with the physical domain shown Championing this development and taking the role equivalent to the standardising bodies (e.g. IEC, CCITT, DIN, IEEE, etc.) would be a great role for INCOSE and SESA.