Simulation Based Design – the basics.

Simulation has been around for a long time and modelling for much longer – they are staple activities for scientists and engineers over all disciplines. The advantages of simulation are well known – here are some of them:

- The discipline of constructing a model of the system or object of interest is in itself informative; the understanding and insight yields better designs even if the simulation is never used!
- Simulation allows a more detailed system description than analytical methods so that nonlinear phenomena, mode changes and second-order effects can be included.
- Predictions can be made of how a system reacts outside its normal operating envelope, which may be too dangerous or expensive to do in practice.
- Simulation allows rapid and relatively cheap design iteration leading to optimum selection of system parameters.

Simulation also has some drawbacks:

- There is additional effort involved in deriving a system model.
- It can produce a great deal of data which then has to be interpreted.
- The results are only as good as the model.

Nevertheless, simulation has become a standard tool that is used every day throughout industry and commerce.

So what is simulation-based design? Well, it's not the principle of simulation that has changed but the way that it is becoming a ubiquitous part of the systems engineering process.

Some years ago, companies working on large, complex and multi-disciplinary projects (aerospace and defence are the archetypal examples) started to find that problems occurred during the very late stages of development. Some of them were severe and very expensive to rectify. These problems could often be traced to erroneous statements made right back at the early requirements stage. Traditionally, the supplier of a sub-system would be given a textual specification (possibly a very long one) for the required system and this would eventually be delivered at the systems integration stage. Provided the original specification was complete and unambiguous and had been followed diligently then integration should proceed smoothly – but this was not always the case. Perhaps the specification was not complete, allowing the sub-system supplier a degree of discretion in what was provided or maybe ambiguity in interpreting the text caused deviation from what was really intended. The field of *systems engineering* emerged in an effort to eliminate these problems.

Two features of systems engineering particularly encouraged the increased use of simulation:

- The need to have better interfaces between sub-system suppliers and the systems integrator;
- The need to begin systems integration from an early stage in the project.

The increasing trend is to start with a <u>managed</u> set of textual requirements which are then encapsulated as an executable specification, i.e. essentially a form of computer simulation. This means that feedback about how the overall system should work and how the sub-systems interact is already happening at a very early stage in the project. A consequence of this way of working is that sub-system suppliers must be able to understand the model (specification) in order to understand the requirements. Furthermore, they will be asked to provide the systems integrator with progressively refined models of their own sub-system as the project proceeds. In return, the systems integrator will provide increasingly detailed models of the overall system, against which the sub-system supplier can test their model. The improved interface between supply chain and systems integrator allows discrepancies or misunderstandings to be picked up promptly. In effect, systems integration is largely done in a 'virtual' workspace rather than being delayed to the final stages when the engineering hardware becomes available. So the whole project lifecycle becomes dependent on computer modelling and simulation – hence the term *simulation-based design* (SBD).

Another factor that motivates use of the SBD methodology is improved physical systems modelling, allowing models from different physical domains (hydraulic, thermal, electromagnetic, mechanical etc) to be linked together. The simulation generates a holistic and time-varying account of how the target system functions and interacts with the environment. For instance, consider a metal component which has been designed to work over certain ranges of motion and temperature variation, typically fixed at the start of the project according to some ballpark assumptions. Subsequent simulation of the proposed component within a realistic scenario, provided by more accurate environmental and operational models, may reveal unacceptable stresses in the metal. A re-design can take place before any metal is cut. This procedure is often known as *co-simulation*.

Many subsystems will require the delivery of software for monitoring, display, signal processing or control. Much of this will be real-time software where it is critical to test the timing and communication protocols at the interfaces as embedded code emerges from the design phase. A simulation centric approach helps here by executing the code on general purpose computers in simulated time, and replacing the actual electronic units and physical components with their simulated counterpart. This forms a sort of 'Virtual Systems Integration Laboratory' which, as well as alleviating systems integration problems, makes it possible to begin training personnel at an early stage. The transition to real-time simulation is made as the expensive target hardware gradually becomes available later on in the project.

Software support tools are essential for managing this kind of complexity. The *Unified Modelling Language* (UML) is the most widely used visual modelling language used by software engineers but the systems engineering function is now supported by an off-shoot known as the *Systems Modelling Language* (SysML). Both languages rely on the use of diagramming techniques to create models of software and systems and are standards regulated by the *Object Management Group* (OMG). A number of companies sell implementations of UML and SysML as part of an integrated tool suite to underpin teams of modellers and systems and software engineers working together. The required infrastructure is not easy to achieve because it must include features such as remote working, multiple language support, configuration management, data repository and security measures to protect IPR embedded within the sub-system models contributed by participating companies.

The implication for sub-system suppliers is that they must be an integral part of the systems engineering process. In addition to supplying the end-product, this means being able to use and to contribute computer simulation models at all stages of the project lifecycle. At present, simulation based design is prevalent in the aerospace and defence industries but these are by no means unique in their need to handle large, complex and interrelated systems; it is likely that the methodology will extend to other industries too.