UNIVERSITY OF Southampton
School of Engineering Sciences

Modelling Paradigms in Lifecycle Costing

UTC for Computational Engineering
Tai-Tuck Yu and J.P. Scanlan: CEDG School of Engineering Sciences | G.B. Wills: Learning Societies Lab, School of Electronics and Computer Science

Introduction

As the boundary for the design of a complex product is progressively extended to encompass less traditional engineering fields, the importance of a customer's high-level requirements becomes increasingly prominent. While it might have been acceptable in the past to design a component in isolation, or perhaps in relation to adjacent components, the customer's demands today have driven designers to consider also its economic performance over its operational life. The design goal now tends not to be expressed in narrow, technical terms but more abstractly like, "Reduce engine total operating cost by 10%."

Why use simulation?

A real, complex engineering system will certainly involve variables such as component reliability and process consistency. It is generally acknowledged that these inherent uncertainties rule out an analytic solution for the reason that combining different probability distribution functions analytically for even a simple system will soon produce a set of equations too cumbersome to manipulate. While it may be possible to simplify a model to make it manageable, it is also likely to result in a model with fidelity too low to be of practical value. Computer simulation is often the only tool available to deal with a complex stochastic problem. The flexibility offered by any of the mature COTS modelling packages enables uncertainties and discontinuities to be introduced easily at any appropriate point in a simulation model.

Simulation in lifecycle costing

Lifecycle costing of a complex engineering product like the Rolls-Royce Trent800 engine draws together various strands of input among which are the product's attributes, its operating environment, supporting infrastructure, supply chain, and the rules governing its ongoing maintenance and final disposal. In the IPAS project, the Extend6 modelling package has been used successfully to build a discrete-event model (DEM) of the Trent800 repair operation (**Fig.1**). The Extend6 tool also allows a bidirectional flow of data with external software applications like the Vanguard DPro costing package and the Excel spreadsheet. To be able to integrate loosely with other tools gives the model added utility as it expands its ability to execute multiple 'what if' scenarios and to perform post-simulation data processing.

The Extend6 tool provides a powerful, intuitive, visual drag-and-drop programming environment on a computer desktop so that it is possible for experts in specific knowledge areas to build and modify their own models. An example of a model sub-system is shown in **Fig.2**. By devolving these activities to the desktop, it is conceivable that the bottlenecks usually present because of an over-dependence on the IT department's expert modellers will cease to exist. An outcome of this will be a highly desirable increase in design productivity.

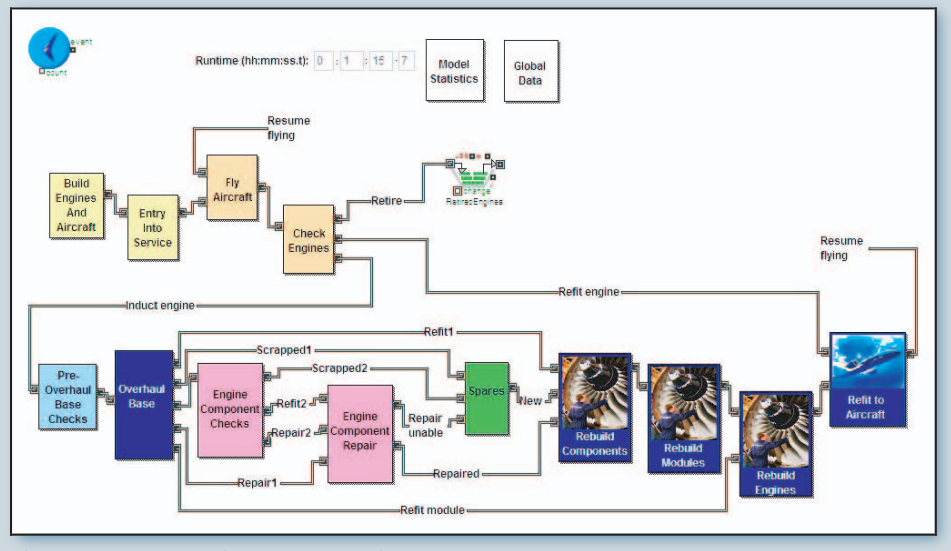

Fig.1 Discrete-event model of the Trent800 engine fleet repair and overhaul operation

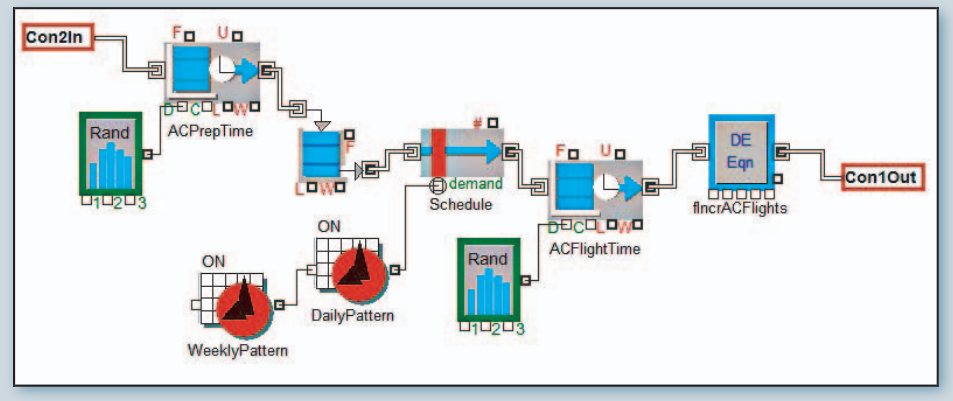

Fig.2 Detail of an Extend6 hierarchical block used for scheduling flights

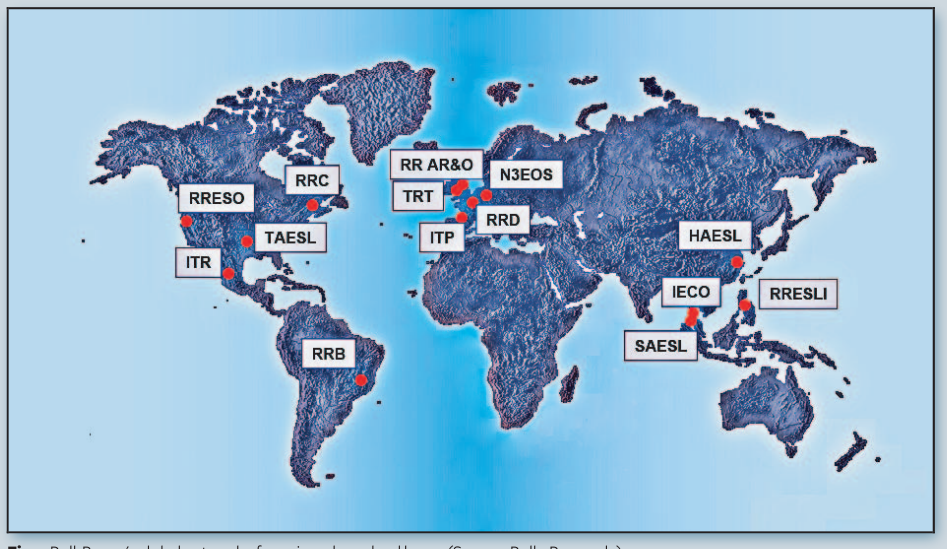

Fig.3 Roll-Royce's global network of repair and overhaul bases (Source: Rolls-Royce plc)

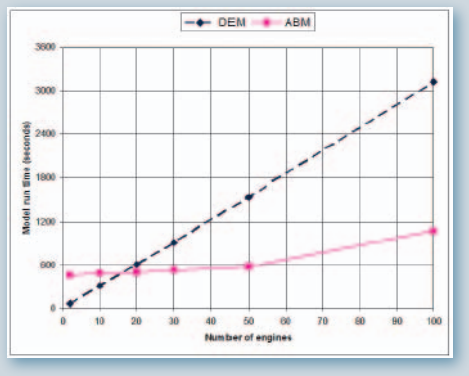

Fig.4 Variation of model runtimes against engine fleet size

Agents in modelling

The versatility of agent-based modelling (ABM) is apparent in its ability to tackle problems ranging from single-cell biology to the global ecosystem, and from a simple reactive module to complex deliberative systems. Although some tend to think of an agent as merely an active software object, the distinguishing feature of agenthood is its capacity for autonomous action. A logically consistent set of 'if-then-else' rules embedded in an agent define how it should behave in response to an external demand. Because of its ability for self-determination, there is always the potential for unintended or unprogrammed behaviour to emerge from a community of interacting agents. In the social sciences such emergent group intelligence may be desirable but it is not necessarily so in the engineering sciences where predictability is more important.

Subsystem Metric	A	B	C	D	E
Lines of non-comment code	24	48	17	15	28
Number of methods or procedures	28	68	22	32	84
Number of classes or blocks	22	30	8	27	8
Weighted method per class	81	188	76	57	266

Table.1 Ratio of DEM to ABM code metrics

ABM vs DEM

To carry out an objective, quantitative comparison, an AB model and a DE model were built to the same level of detail using a common functional specification based on the Trent800 worldwide repair and overhaul operation (**Fig.3**). The models were capable of tracking more than 3,000 selected unique parts for each engine over a simulated lifetime of 40 years. Some of the model code metrics which are commonly used to indicate the effort needed for software maintenance are shown in **Table 1**. They describe five clearly delineated and equivalent subsystems in both models. The values, adjusted for programming paradigm and modelling environment, are all much greater than unity thus showing that a DEM will be more difficult to understand, modify, and test. It is also clear from the model runtimes (**Fig.4**) that ABM performs better than the DEM when the engine fleet gets larger. From the comparison study, two findings stand out

The almost total absence of COTS ABM tools which continues to be a huge obstacle in the commercial adoption of that modelling paradigm.The improved flexibility of control in an AB model as the structure is established at runtime and not before.

By adhering to a structured software design methodology and using the Extend6 tool, a DE model can be made to resemble an AB model while retaining its process-centricity – a natural metaphor to traditionally trained engineers. This is achieved by a three-layer architecture which sandwiches a programmable control layer between a bottom layer of Dprocess models and a top layer consisting of agent-like control rules.

www.soton.ac.uk/ses/research/ced/posters.html | email: j.p.scanlan@soton.ac.uk
Computational Engineering & Design Group, University of Southampton, SO17 1BJ, U.K.