Hybrid Hierarchical-Discrete Event Life Cycle Cost Model

UTC for Computational Engineering

Wong J.S., Prof. Scanlan, J.P., Dr. Eres, M. H: CEDG School of Engineering Sciences

UNIVERSITY OF Southampton

School of Engineering Sciences

Introduction

This research is part of the Integrated Products and Services (IPAS) project. Its inception was due to the shift in many industries from the selling of products to the provision of services. IPAS aims to link three very different worlds: new service design, new product design and the operation of existing products and services in the field. To do this, IPAS will look to integrate and improve a number of disparate generic technologies and apply them to real life scenarios. One of the objectives of IPAS is to develop a Life Cycle Cost (LCC) modelling tool kit.

Designing for Life Cycle Cost

Life Cycle Cost (LCC) is defined as the sum of all the costs a system incurs throughout all phases of its life from conception all the way to disposal. And with the introduction of programs like TotalCare®, LCC has become even more important. Conventional wisdom tells us that by the time full-scale development has been reached, 75% of LCC will already have been committed. So it is critical that the necessary tools are made available to the designer to design for low LCC. Perhaps a familiar conundrum to a designer would be whether to choose a design which is cheap to manufacture but expensive to maintain or vice versa. This research, to date, has centred on modelling operating costs with particular emphasis on maintenance.

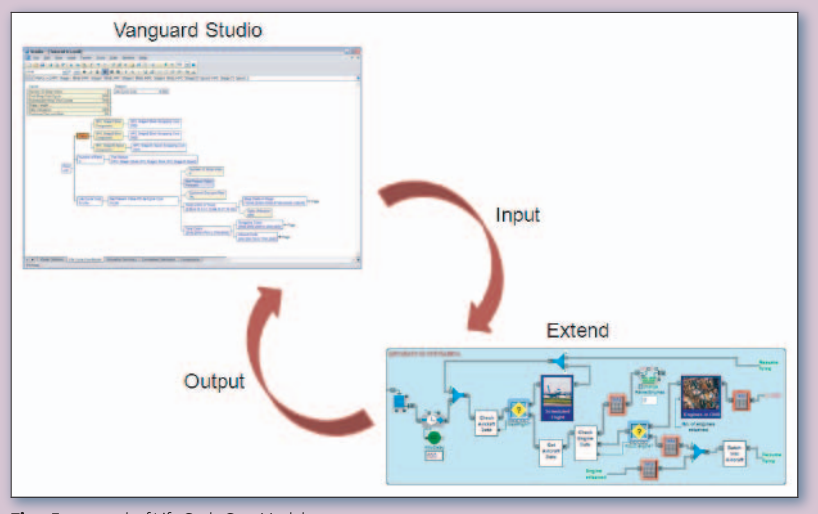


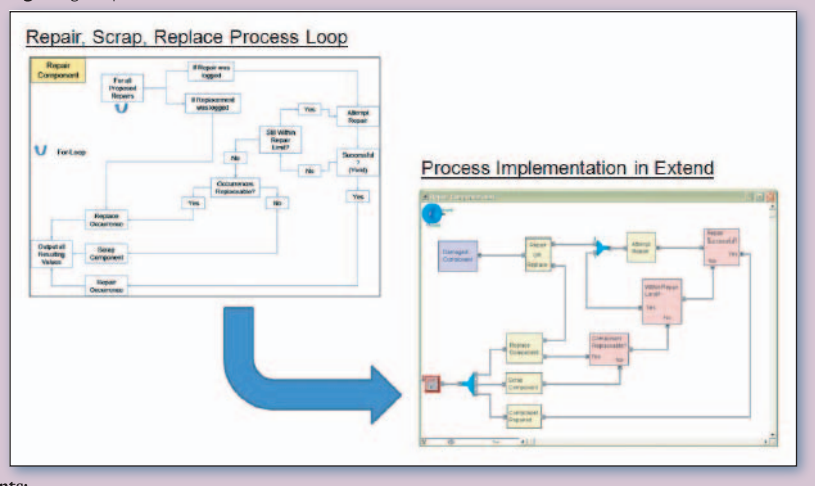
Fig.1 Framework of Life Cycle Cost Model.

Life Cycle Cost Model

There are several reasons for the development of a hybrid hierarchical-discrete event model; but chief among it is that it is unreasonable to find a design tool which possesses all the capabilities that are required. It is more practical to have a suite of tools which can interact with each other to solve the task at hand. The developed approach links two different commercial software packages. The first is a graphical modelling tool named Vanguard Studio which generates hierarchical models. The philosophy behind Vanguard Studio is that to solve a problem, it can be broken down into simpler components. And to solve these component problems, they should be broken down into still finer elements. The result of which is a hierarchical tree. The other software package is a discrete event simulation tool named Extend which models dynamic processes or systems. Its strength lies in its graphical interface which allows complex processes to be displayed graphically. The benefits of this are great especially considering that the alternative would be writing a significant amount of code.

Fig. 1 shows how the two programs interact. Vanguard Studio will collate all the necessary input data and format them before passing the data on to Extend. Extend runs the dynamic simulation and passes the results back to Vanguard Studio, which has superior statistical analysis functions, for post-processing. Vanguard Studio in essence, acts as a wrapper for the Extend model. In other words, for each simulation the user will not have to interact with the Extend software. The Extend model will only have to be modified if the maintenance logic has to be changed.

Fig.2 Logic Implementation in Extend.



Maintenance Logic Representation

One of the drawbacks with previous tools for this kind of application is the representation of the underlying logic behind LCC calculations. Often in previous LCC tools, especially spreadsheet based programs, critical logic statements are hidden behind lines of code. The result is that the program becomes difficult to maintain, debug and modify. These programs also lack transparency and so it is practically impossible to trace if there are any errors in the logic or if the model actually does what it is supposed to do. This is where a discrete event simulation tool, such as Extend, has an advantage. It allows a process to be modelled visually according to its original form. In **Fig. 2**, the box on the left shows the repair logic that a component is subjected to in a maintenance routine. The box on the right shows the process modelled in Extend.

Results

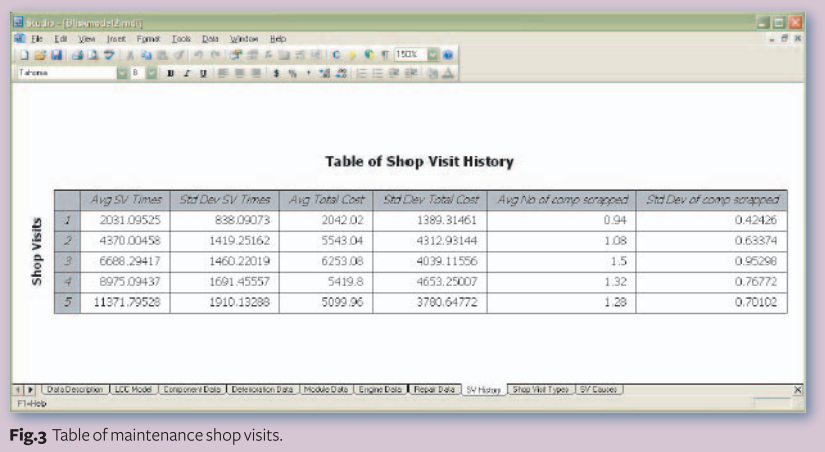


Table of Shop Visit History

Shop Visits	Avg SV Times	Std Dev SV Times	Avg Total Cost	Std Dev Total Cost	Avg No of comp scrapped	Std Dev of comp scrapped
1	2081.09525	838.09073	2042.02	1389.31461	0.94	0.42426
2	4370.00458	1419.25162	5543.04	4312.93144	1.08	0.63374
3	6688.29417	1460.22019	6253.08	4039.11556	1.5	0.95298
4	8975.09437	1691.45557	5419.8	4653.25007	1.32	0.76772
5	11371.79528	1910.13288	5099.96	3780.64772	1.28	0.70102

Fig.3 Table of maintenance shop visits.

Extend has multiple run functionality and this allows Monte Carlo simulations to be performed. With the statistical analysis tools available in Vanguard Studio, the simulation results can be displayed comprehensively. **Fig. 3** and **4** were taken from a LCC model of a set of intermediate pressure compressor blades. The model was run for fifty Monte Carlo simulations. **Fig. 3** shows a table of maintenance shop visits and **Fig. 4** shows the percentage break down of shop visit causes per shop visit.

Fig.4 Graph of shop visit causes.

Acknowledgements:
This work is supported by the IPAS project (DTI Grant TP/2/IC/6/I/10292).

www.soton.ac.uk/ses/research/ced/posters.html | email: jsw1e06@soton.ac.uk

Computational Engineering & Design Group, University of Southampton, SO17 1BJ, U.K.