UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERING AND THE ENVIRONMENT

School of Engineering Sciences

Integrating supply chain simulation, component geometry and unit cost estimation

By

Stuart Jinks

Thesis for the degree of Doctor of Engineering

July 2012

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND THE ENVIRONMENT

SCHOOL OF ENGINEERING SCIENCES

DOCTOR OF ENGINEERING

INTEGRATING SUPPLY CHAIN SIMULATION, COMPONENT GEOMETRY AND UNIT COST ESTIMATION

By Stuart Jinks

This thesis shows how utilising dynamic simulation to estimate unit costs and manufacturing resources, can aid design decisions. A framework specification is introduced that integrates Computer Aided Design (CAD), Discrete Event Simulation (DES) and Activity Based Cost (ABC) methodology. The framework aids a design team in understanding the consequences of design decisions in terms of unit cost and manufacturing resources, by returning aggregated unit cost and manufacturing based data, directly to the design team, within the design environment.

Dynamic Resource Estimation System (DRES) has been developed to implement the framework and conduct two case studies based on representative aerospace components. The purpose of the first case study is to determine the benefits and applications of integrating a dynamic supply chain simulation and unit cost estimation. The second case study is used to show that the framework is capable of handling significantly different components and to highlight the effort required to implement a new component within the framework.

This thesis concludes that there are three primary benefits provided by the framework, which are: firstly, the framework can accurately predict required resources to fulfil a supply chain for a specific production rate, which can be utilised by manufacturing engineers to aid production planning; secondly, the framework increases refinement of a component unit cost estimate, by including manufacturing time and dynamically determined resource requirements into an ABC cost model; and thirdly, the framework has the ability to compare multiple supply chain options and different supply chain types at the same time from component geometry.

Contents

List of figures

List of tables

Declaration of authorship

I, Stuart Jinks, declare that the thesis entitled *"Integrating supply chain simulation component geometry and unit cost estimation"* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as:

Journal papers that have been submitted and awaiting peer review

- Jinks, S. and J. P. Scanlan (?). "Integrating unit cost, dynamic supply chain modelling and component geometry." International Journal of Advanced Manufacturing Technology **?**(?): ?

Journal papers

- Jinks, S. (2011). "Integrating simulation and geometry to determine cost." The Journal of the Association of Cost Engineers **49**(3): 4.

Conference Papers:

- Jinks, S., J. P. Scanlan, et al. (2010). Utilising Dynamic Factory Simulation To Improve Unit Cost Estimation and Aid Design Decisions. 2010 Winter Simulation Conference. B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yucesan. Baltimore, USA, IEEE.
- Jinks, S., J. Scanlan, et al. (2010). Improving unit cost analysis by generating dynamic factory simulation from CAD geometry. 21st International Computer Aided Process Engineering (CAPE). Edinburgh, GB.

- Jinks, S., J. P. Scanlan, et al. (2009). Predicting Manufacturing Costs by Generating a Factory Simulation From CAD Geometry. Fifteenth Post-Graduate Conference in the Materials, BioEngineering and nCATS Research Groups at the University of Southampton. Lyndhurst, GB.
- Jinks, S., J. P. Scanlan, et al. (2008). Near Net-shape Manufacturing Costs. 15th ISPE International Conference on Concurrent Engineering. R. Curran, S.-Y. Chou and A. Trappey. Belfast, Springer.

Signed:…………………………………………………………………………

Date:……………………………………………………………………………

Acknowledgements

I need to thank many people for their help and support throughout this research. Without them completing this work would have been an uphill struggle and without some of my friends I probably would never have started it.

First I wish to thank my supervisors Prof Jim Scanlan, Prof Philippa Reed and Dr Steve Wiseall. Without their advice, guidance and expertise I may have been destined to struggle up that hill of research for much longer than I did; to them I am very grateful.

A special thanks to my 'left hand' colleague Julie Cheung who I have been through this process with from the very start. She has helped me by talking through my problems and has been a sanity check for my ideas. I also need to thank the Southampton University 'Crew' who made this process fun with some banter and probably too many breaks on a Friday.

I need to thank the Computational Engineering Design Group of Southampton University, Product Cost Systems of Rolls-Royce Plc and the REMAC project (sponsored in part by the Technology Strategy Board) for their help, support and funding because without it this project would never have happened.

I need to thank my friends and family because without their encouragement I would never have started this research. Lastly, but not by least, I need to thank my lovely wife Katie who has always been there supporting me through thick and thin.

Abbreviations

ABC	Activity Based Costing
ABM	Agent Based Modelling
AFR	Automated Feature Recognition
API	Application Programming Interface
AUC	Aggregated Unit Cost
CAD	Computer Aided Design
CAM	Computer Aided Manufacture
CAPP	Computer Aided Process Planning
CER	Cost Estimation Relation
COS	Condition Of Supply
DES	Discrete Event Simulation
DRES	Dynamic Resource Estimation System
GUI	Graphical User Interface
JIT	Just In Time
HIP	Hot Iso-static Pressing
PCS	Product Cost Systems
PDP	Product Development Process
REMAC	Resource Efficient Manufacture of high performance hybrid Aerospace Components
SD	System Dynamic
WIP	Work In Process

Chapter 1

Introduction

Understanding the cost of a component can mean the difference between making a profit or not. Cost estimation, a method of understanding cost, requires knowledge, data and when these are not fully available, assumptions in the form of a probability distribution. The cost of a component could be determined from the geometry alone, however many assumptions would be required such as the material and manufacturing process, therefore the distribution surrounding the cost would be so significant as to render the estimate almost useless. If the component material, manufacturing process sequence and process times were supplied with the geometry, the reduced number of assumptions required would result in a smaller distribution around the cost estimate. The primary assumption required in this situation would be the process cost rates which, in most cost estimation methods, are based on historical data (Tammineni 2007) that is collected from similar processes.

Cost rates from historical data is collected, at a point in time, from components and processes that are based on specific parameters. Therefore the scope of relevance of historical data has a distribution around these parameters. As the parameters change the applicability of the historical data reduces. Therefore the crux of the problem is, if historical data is utilised within a cost estimation it must be for components or processes with similar parameters and therefore within scope of the historical data otherwise the cost estimation may diverge from actual cost.

A method of mitigating this problem is to reduce the use of historical data and build the cost estimate from a scientific base data. Hence instead of using a cost rate of a machine or process determined from historical data it can be calculated by determining the resource requirements from a model of the manufacturing system. However a limitation of typical industry cost estimation methods is that they are based on static models which have difficulty fully representing dynamic system (Marsh, Jonik et al. 2010).A solution that addresses this limitation is to utilise a modelling method that fully represents a dynamic system as it evolves with time; dynamic modelling allows this (Law and Kelton 1992).Discrete Event Simulation (DES) is a dynamic modelling technique which is accepted as research and industry best practice for modelling manufacturing systems (See 2.2.1.4 Discrete event)

Integrating a DES model with a cost estimation method has complications. DES models typically require skilled modellers to construct useful, flexible and well structured models (Barton, Bryan et al. 2004; Pidd and Carvalho 2006). Also DES models require substantial amounts of data (such as number and type of manufacturing processes and required resources) and knowledge (such as logic of how the supply chain works and rules to control it) to be embedded in to the model before it can be used to produce meaningful results. Therefore the data and knowledge must be gathered or determined using other tools before the dynamic model can be built.

A framework is required to integrate all the tools necessary to bring together the data and knowledge required to build the dynamic model and incorporate the dynamic results into the cost estimation in real time. This research has formalised the framework and created a system to implement it to prove the concept.

1.1 Motivation for this research

Rolls-Royce plc provides integrated power systems in the aerospace, defence, marine and energy markets. The company is the second largest manufacturer of gas turbine engines (Figure 1) with annual sales of £10.9bn, of which civil aerospace constitutes the largest area of sales at £4.9bn (45.3% of total sales) in 2010 (Rolls-Royce Plc 2010).

Figure 1: Rolls-Royce Trent 900 (®Rolls-Royce)

Product Cost Systems (PCS) is a central group within Rolls-Royce that develops and promotes cost estimation within the organisation and has supported this research throughout its duration. Cost estimation within Rolls-Royce utilises the knowledge based method (Tammineni, Scanlan et al. 2007) that combines historical cost rates with features in a visual object orientated modelling environment called Vanguard (Vanguard Software^{TM} Corporation 2011). PCS has an active role in understanding the cost of manufacturing processes early in their development so that cost drivers can be understood or developed to reduce process cost.

Due to the active role of PCS this research was sponsored in part by the Resource Efficient Manufacture of high performance hybrid Aerospace Components (REMAC) development project (Jinks, Scanlan et al. 2008). The REMAC project was developing a near net shape capability for manufacturing a component through the use of the powder Hot Iso-static Pressing (HIP) process. The powder HIP process is a batch process that produces a near net shape by consolidating powder contained in a canister into a solid component.

There are three areas of industrial motivation and direction that the author took into the research. Firstly, due to the lack of historical data for the HIP processes, the dynamic interactions of batch processes and the distribution of possible production rates of components, directed the author to investigate the integration of dynamic modelling and static cost estimation. Secondly there was a requirement to compare different manufacturing processes for a single component to ensure that optimum cost is being achieved for the component. Thirdly a requirement of the aerospace industry is to make geometry design changes for multiple reasons, such as specification change, design optimisation and design for the process. This third requirement can be described as real time decision making.

1.2 Statement of research

The statement of hypothesis is:

"Integrating supply chain simulations with design geometry can assist in design decision making"

1.3 Research aims

The aim of this research is to assist the design process by aiding decision making by conducting real time cost estimations, incorporating a dynamic aspect into unit cost estimation and allowing comparisons of manufacturing processes.

To do this a framework has been developed that integrates a dynamic model with cost estimation. This provides the modelling capability to fully represent the dynamic characteristics of the manufacturing supply chain. The results of the dynamic model are utilised in two ways, within a cost estimation model and directly to compare manufacturing methods. Also by integrating with design geometry the consequences of design decisions can be linked directly to the cost output. This will provide a design team with a real time cost estimation and holistic manufacturing prediction which is intended to lead to more informed design decisions.

1.4 Research scope

There are four general areas that cover the scope of this research as shown in Figure 2. The four areas are: cost estimation, manufacturing process technologies, dynamic modelling and components. Within each of these areas are specific topics upon which this research will focus. Unit cost will be considered within the cost estimation area. Forging, machining, electrochemical machining and powder HIP are the manufacturing process technologies that will be utilised. Discrete event simulation will be utilised in the dynamic modelling area because it is used by industry for manufacturing and supply chain simulation, as discussed in section 2.2. The framework will be applied to two significantly different aerospace component types which are an aero engine case and a blisk; these are described further in Chapter 4.

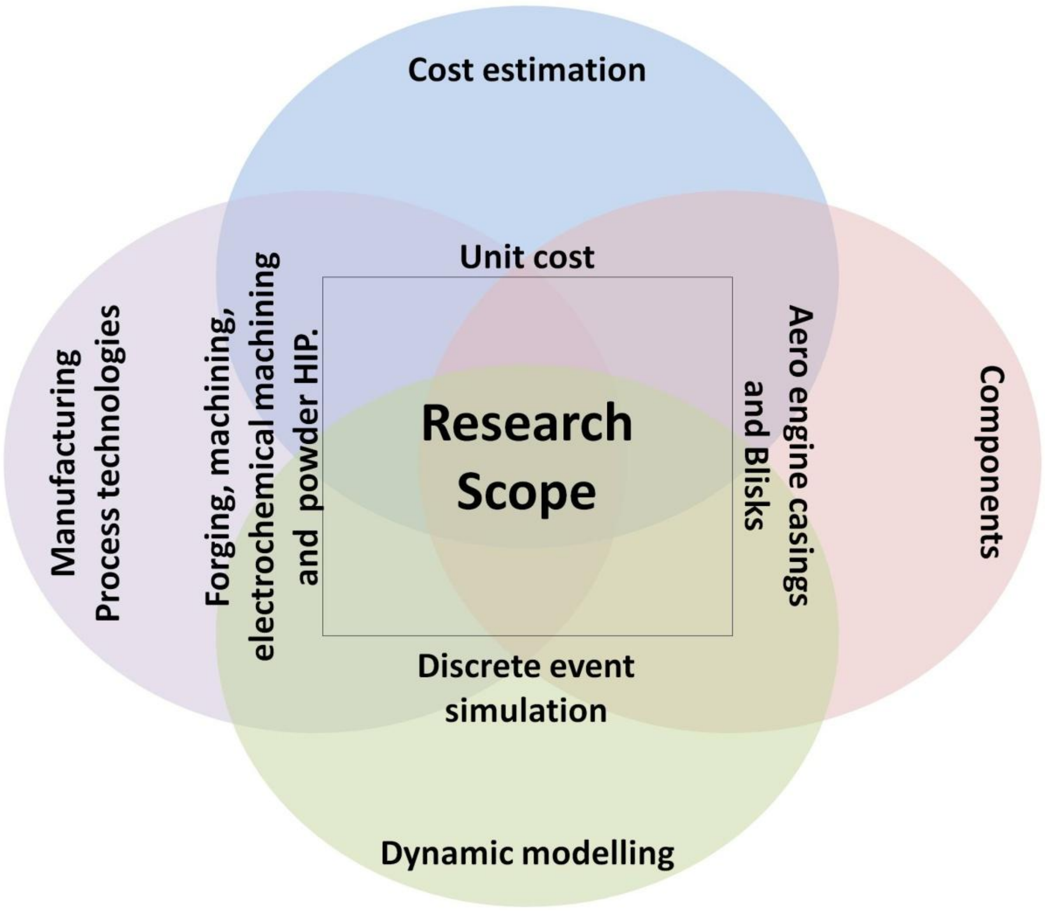

Cost estimation
Unit cost
Manufacturing Process technologies
Forging, machining, electrochemical machining and powder HIP.
Research Scope
Aero engine casings and Blisks
Components
Discrete event simulation
Dynamic modelling

Figure 2: Research scope

1.5 Layout of thesis

This thesis consists of 6 chapters laid out as shown in Figure 3. The remaining chapters include literature review, framework, case studies, discussion and conclusions and future work.

Chapter 2, the literature review, presents a foundation for the research and contains four sections. The first describes the main cost estimation methods and their primary limitation. This section ends with a recommendation of integrating a dynamic modelling capability with cost estimation to solve the limitation. The second section describes the main dynamic modelling methods, recommending discrete event simulation as a suitable option. The section finishes by discussing data driven generic modelling. Integrating geometry is the third section and discusses two methods, automated feature recognition and computer aided process planning. These methods were determined unsuitable for complex components, therefore alternative, less flexible, methods were proposed. The last section, supply chains, defines a supply chain and aspects related to modelling a supply chain.

A framework structure is proposed that integrates dynamic modelling with cost estimation in Chapter 3. The framework structure is described in detail and contains five stages, which include:

1. Geometry modification
2. Determine manufacturing process
3. Manufacturing process data generation
4. Dynamic modelling
5. Aggregated unit cost

The fourth chapter describes the results from two aerospace component case studies, which are: a combustor outer case and a blisk. The first case study has the purpose of determining if integrating a dynamic model with cost estimation provides a difference in results compared to a cost estimation that does not have an integrated dynamic model. Also it assesses, under which circumstances any difference between the different modelling approaches occur. The purpose of the second case study has two parts. The first is to show the flexibility of the framework by implementing a different component type. The second is to highlight the steps necessary to implement a component or supply chain within the framework.

The fifth chapter, discussion, contains five sections. The first discusses the findings from the case studies. The second discusses whether the framework benefits are worth the required effort to set up the framework for a component. The third discusses validation, for both of the case studies and future implementation of the framework. The last discusses framework improvements.

The last chapter presents the significant conclusions of the research, followed by key contributions to the research field. Recommendations of future research, building on the findings, are discussed before concluding remarks.

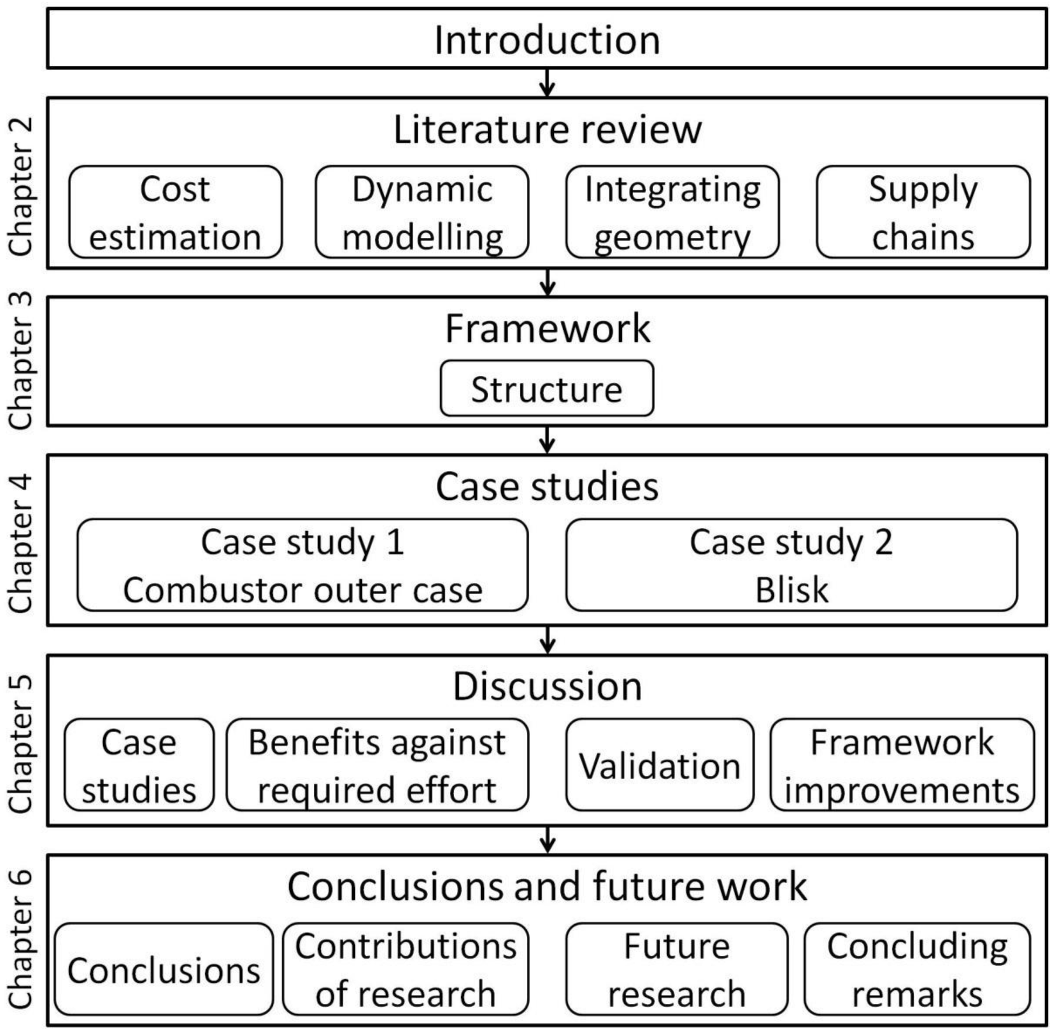


Figure 3: Layout of thesis

Chapter 2
Literature review

This chapter presents a foundation for the research in the form of a literature review of the relevant areas. The relevant areas have been categorised into four sections which are: cost estimation, dynamic modelling, integrating geometry and supply chains.

The first, cost estimation, highlights that cost estimation methods are based on static modelling techniques. An argument is put forward that static modelling techniques are unable to make sufficiently accurate predictions of dynamic systems, therefore are a limitation to cost estimation methods. Utilising dynamically derived data within cost estimation is recommended as a solution to the limitation, which leads to the integration of dynamic modelling to generate the dynamic data required.

The second section, dynamic modelling, discusses possible modelling methods. A dynamic modelling method is suggested based on the requirement to model manufacturing and supply chain systems. A generic modelling methodology is also proposed as a way to reuse the model and store the required input and output data.

The third section, integrating design geometry, discusses possible methods to aid the integration of dynamic models with design geometry. Two methods are discussed: the first automated feature recognition, the second computer aided process planning. Both methods have limitations that resulted in a direct approach being proposed as this increases automated capability, however at the loss of flexibility.

The last section, supply chain, defines a supply chain within the scope of this research. This section also defines three critical aspects of a supply chain which are utilised when creating a dynamic model, these are:

- Resource modelling
- Inventory control
- Manufacturing batch operations

The chapter ends with a summary that incorporates the main conclusions and findings from the four sections and proposes suggestions for achieving the research aims.

2.1 Cost estimation

Cost estimation, as defined by Stewart et al. (1995), is *"a process of predicting or forecasting the cost of a work activity or work output"* which can be used throughout the Product Development Process (PDP) to aid understanding of total unit cost. A PDP represents the life cycle of a product from conception through design, manufacture, operation and finally disposal (Asiedu and GU 1998; Kim, Jeong et al. 2009). A PDP typically contains a number of stages, which can be classified into categories, formal review procedures and decision gates, as shown in Figure 4. This standardised process seeks to minimise risk by systematically identifying and reducing uncertainties (Scanlan, Rao et al. 2006).

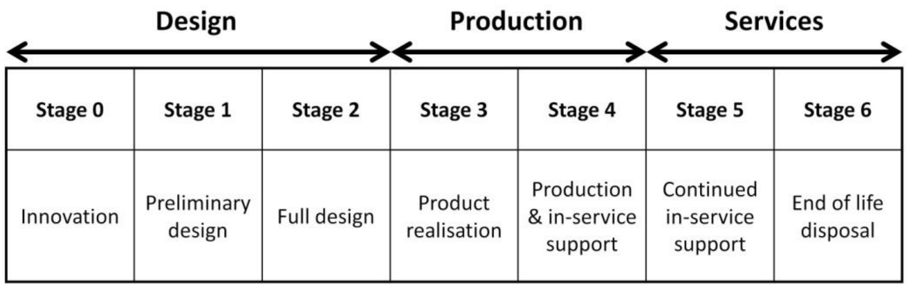


Figure 4: Product development process. Based on (Tammineni 2007)

Many authors believe that 70% - 80% of a product's total cost is controlled by early design decisions (Cooper 1990; Zeigler, Kim et al. 1999; Beck and Nowak 2000). The empirical evidence that supports this statement is questionable (Forrester 1961; Ulrich and Pearson 1993), however, it is widely accepted that design decisions, especially at early design stages, control total unit cost (Pidd 1992) therefore cost estimation is important at early design stages (Newnes, Mileham et al. 2008).

Asiedu and Gu (1998) discuss how uncertainty of cost estimation results result in the accuracy of the estimate being inversely proportional to the span of time between the estimate and the event to which it refers. The graph in Figure 5 supports this thinking by showing how total unit cost of a product is not fully understood until steady state production has been reached, as shown by cost determination. However the duration of time to reach steady state production depends on many aspects such as production rate and variability of the manufacturing processes. When steady state production has been achieved uncertainty can be removed from the data required by the cost estimate, which at that point ceases to be an estimate and becomes a calculation. There are many uncertainties when a new product design project is initiated, but as time progresses decisions are made that narrow these uncertainties. This is shown by the cost determination line in Figure 5, which shows how over time cost is determined and eventually matches costs incurred.

A third aspect shown in Figure 5 is the ease with which changes can be made to the product design as the project progresses through the PDP. At early stages of the PDP designs can be changed easily, but as decisions are made throughout the PDP the ease of making changes decreases due to implied constraints from previous decisions.

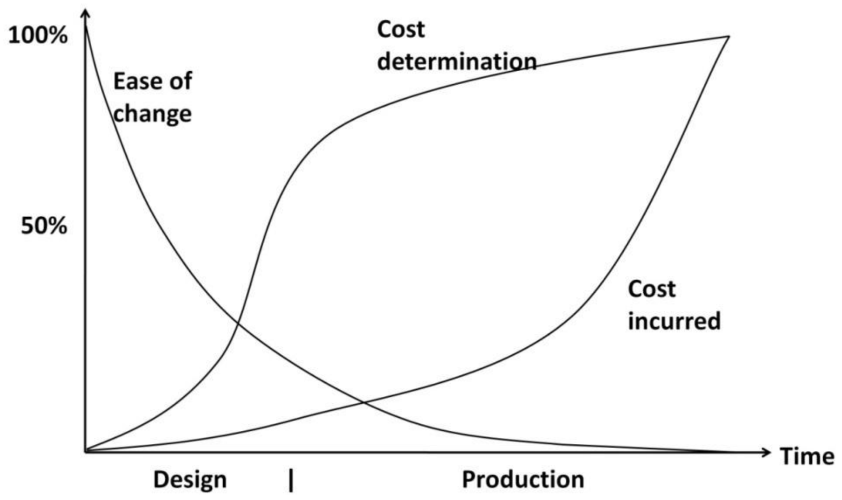


Figure 5: Cost determination, cost incurred and ease of change against time. Modified from (Dowlatshahi 1992; Miles and Swift 1998; Layer, Brinke et al. 2002)

2.1.1 Cost estimation methods

Asiedu et al (1998), Rush et al (2000), Curran et al (2004), Niazi et al (2006), Tammineni (2007) and (García-Crespo, Ruiz-Mezcua et al. 2011) have extensively reviewed cost estimation research and methods. The main cost estimating methods include: analogy, parametric, feature, activity, and knowledge based.

2.1.1.1 Analogy based costing

Analogy cost estimating is based on adjusting the cost of a similar product relative to the differences between the new and similar product (Taylor 1998).

This method requires complete historical data of similar components, and appropriate scaling parameters to be applied (Scanlan, Rao et al. 2006).

There are risks associated with this method that relate to the amount and accuracy of historical data and the scaling parameters used which require an understanding of the product and involve expert judgement (Jenab and Liu 2009). Analogy methods cannot be used for new component designs or manufacturing processes (Jaya Suteja, Prasad KDV et al. 2013) because there are no similar products to base the new component cost on. This limitation is also extended if the production volume of the new component is significantly different to the product it is based on. This is because a significant change in production volume may require changes in manufacturing process, tool and fixture design or equipment to enable the production volumes to be achieved.

2.1.1.2 Parametric based costing

A definition of parametric estimating is given by Dean (1995) as the “generation and application of equations that describe relationships between cost, schedule, and measurable attributes of systems that must be brought forth, sustained, and retired”. (Wright 1936) is one of the first published uses of parametric cost models in aerospace however it was the Rand Corporation in the 1950s that developed Cost Estimating Relations (CER) (Younossi, Arena et al. 2002). A CER is developed by determining a correlation between the dependent variable cost, and independent variables such as size. An example of a simple CER is the relationship between cost and mass of a component.

CER’s are based on historical data which lead to two limitations. First, CER’s can have a limited range, for instance, if a new piston is designed, which is the same as a previous piston except for a small change in diameter, a CER utilising mass could be used to determine the unit cost of the piston. However if the change in diameter of the new piston was large enough to require a new manufacturing process or tool, the CER may no longer be valid. Secondly, a CER cannot be used for new components or novel technology because there is no historical data to create the CER (Rush and Roy 2001; Jaya Suteja, Prasad KDV et al. 2013). These limitations are the same as the analogy based method because of the reliance on historical data.

A CER could be a simple scale factor, for example based on surface area, therefore the parametric method and the analogy method are, in a simple form, the same. In practice the parametric method can utilise multiple complex scale factors to determine the cost of a component. However the reliance on scoped historical data limits the use of this and the analogy methods.

2.1.1.3 Feature based costing

A component can be described by a number of associated features such as: hole, pocket, slot and flange. Each feature has an associated process cost therefore the component process cost is the sum of all the feature process costs (Rush and Roy 2000). For simple components, where features are independent, this method allows a designer to understand which features drive the component unit cost. However, features are not always independent; they interact with each other affecting the cost and reducing the distinction between the features. Two reasons to use features as drivers of cost as described by Wierda (1991) are:

- Cost functions can be derived for classes of similar objects that serve as key drivers of global cost estimation and are linked to the engineering domain
- The designer expects to know the causes of costs so that when linked to design features, they are able to influence committed cost directly

There are however two difficulties with the feature based method. First, there is no consensus for a standard set of features or methods to create them (Taylor 1998). For instance a feature could be described by using either manufacturing or design parameters; see section 2.3.1. Companies are therefore required to create their own set of feature definitions. Secondly, linter-linked features which are connected or share parameters can cause complexity when calculating the cost (Srikantappa and Crawford 1994). For instance, if an interacting pocket, slot and hole were manufactured by machining, feature based costing could estimate the processing cost of each feature individually. However the order of manufacture would affect each feature cost because they are interacting and this could affect the total process cost.

True feature based cost estimation, that is the ability to update the cost of the component when the component geometry is changed, requires an ability to either: assess the component features and all the different processing methods to achieve them; or requires a predefined manufacturing process for feature combinations. Both of these methods requires a connection to the component geometry which is discussed further in section 2.3 Integrating geometry. Without either of these abilities a true feature based cost approach cannot be achieved. However a feature based cost model can be created that is a static representation of the component. This static feature based cost model would require an expert modeller to determine the necessary changes to the model based on any updates to the component geometry.

2.1.1.4 Activity based costing

The theory behind Activity Based Costing (ABC) is that virtually all a company's activities exist to support the production and delivery of its products. By determining which activities and the amount of those activities the products consumes, a product unit cost can be determined (Cooper and Kaplan 1988; Cooper and Kaplan 1988; Liggett, Trevino et al. 1992; Özbayrak, Akgün et al. 2004). Activities can be classified into four categories(Cooper 1990), including:

1. Unit level – Performed each time a unit is produced
2. Batch level – Performed each time a batch of goods is produced
3. Product level – Performed as needed to support the production of each type of product
4. Facility level – Performed to sustain the factory's performance such as rent, depreciation and insurance

The unit level category of activities forms the majority of activities for most component unit costs because these activities include operations such as machining, inspection and cleaning. Batch level activities include heat treatment and HIP cycles. Product level activities may only happen infrequently such as tool specific setups. The facility level activity incorporates all the other none direct activities which would are difficult to quantify and assign to a specific product. It is this final activity level that reduces the 'hidden' factory cost, otherwise known as overhead cost, and can improve the accuracy of unit cost estimation (Asiedu and GU 1998; Mikko, Marko et al. 2007; Park and Simpson 2007; Askarany, Yazdifar et al. 2010; Wang, Du et al. 2010).

The disadvantages of ABC is that it requires substantial amounts of detailed data compared to analogy, parametric and feature based methods which results in more complex models requiring expert knowledge to complete the estimate (Spedding and Sun 1999). The requirement of less data is one reason why analogy and parametric methods are widely used. In certain instances such as single product settings there is little advantage over traditional costing systems

(Asiedu and GU 1998). In single product factories there is no need to determine which parts of the factory, equipment and labour need to be assigned to which products because it is all assigned. Therefore the total factory cost per year is divided by the number of products manufactured to determine the unit cost.

Askarany et al's (2010) paper is a supply chain management study of the adoption of ABC within different size organisations. Askarany et al give multiple reasons why the adoption of ABC can provide improvements to any organisation, with the main points being: providing a clear picture of where resources are being spent; providing an alternative to volume based product costing; identifying value added activities allowing the reduction of non-value added activities; improving the accuracy of process and product cost estimation; and a method to obtain long term profit by exercising complete control over overheads. Askarany et al is among a growing community (Farrell and Simpson 2009; Hammami, Frein et al. 2009; Wang, Du et al. 2010; Tsai, Shen et al. 2012) outside engineering that are utilising ABC to improve understanding of supply chains. For example an ABC approach was suggested by Tsai et al (2012) to solve the problem of environmental and cost evaluation. They utilised the ABC method to determined environmental cost by calculating the pollutants and energy usage per product, then converted this into a cost.

2.1.1.5 Knowledge based costing

The knowledge based system was developed by Tammineni (2007) to overcome two shortcomings of modelling environments and two limitations of cost estimation methods. The modelling environment limitations are considered to be:

- Basic visualisation of data in the modelling environment
- Little support to users with limited programming skills

The cost estimation limitations are considered to be:

- Uncertainties are applied in a black box approach and without sensitivity analysis capability
- Minimal ways to present manufacturing knowledge to the user

The system Tammineni developed utilises a generic modelling tool developed by Vanguard Software Corporation (2011) and is used by Rolls-Royce to conduct their unit cost estimation. The tool overcomes the modelling environment limitations by providing a method of building a cost model without programming and by using a visual tree structure. The tool has a web based view that allows reuse of models by none modellers allowing them to modify the model inputs and viewing the output. This means the customer is not required to have expert knowledge in cost modelling and with the tool to conduct different scenarios within the constraints of the model inputs and logic.

The cost estimation limitations were solved by allowing uncertainties to be applied to all inputs therefore allowing a Monte Carlo and sensitivity analysis to be conducted. The also tool presents the cost of the component in the same process order the component is manufactured, or by feature, therefore improving how manufacturing knowledge is presented to the user. This improves understanding of the model and consequently the unit cost by linking it to the design. The method also incorporates an object orientated approach that allows libraries of building blocks to be used in a parent child arrangement. This approach allows reuse of data, simplified maintenance and consistency of model structure. (Tammineni, Scanlan et al. 2007; Tammineni, Rao et al. 2009).

In overcoming the limitations that Tammineni discovered in his research the developed tool Vanguard can combine any of the other cost estimating methods through the use of the object orientated approach. However the knowledge based approach does not introduce a new method of determining the cost of a component. What it does do is allow a cost modeller to create a cost model using the best costing method that fits the problem, with all the benefits of applying uncertainties and object orientated features without programming skills. In Rolls-Royce Vanguard is used mostly to create top down process based cost models, however when it is appropriate analogy models and ABC models are used. All the different cost estimation models for the different components are then combined in to a whole engine model to determine a whole engine cost.

2.1.2 Static or dynamic cost estimation

A definition of a 'static system is one where output is independent of past values of input', therefore' determining the output of a static system requires no memory of the input history' (Cassandras and Lafortune 2008). A similarity between the cost estimation methods discussed in section 2.1 is that they are based on the definition of a static system and use static models to determine output values. Another definition of a static model is a model that represents a specific point in time (Law and Kelton 1992). The definition of a 'dynamic system is one where the output generally depends on past values of input' (Cassandras and Lafortune 2008) and dynamic systems evolve over time (Law and Kelton 1992; Marsh, Jonik et al. 2010). In a static system the output is always the same when the input is fixed, however the output of a dynamic system is not always the same with a fixed input. A limitation of static models, and therefore the cost estimation methods discussed, is that they are unable to fully represent dynamic systems. A potential solution for this limitation is to use dynamic data. Two sources of dynamic data are a real system or a modelled system.

Each cost estimation method discussed utilises historical data. Historical data is collected from a real system, therefore can be categorised as dynamic data. A problem is that historical data has a limited scope and is collected at a point in time. For instance, a change in a system input variable (such as production rate, operation times or number of operations) could cause changes to system outputs and the collected data. Therefore a cost estimation model using historical data of a component but requiring different parameter values would be using historical data that was out of scope. Creating a dynamic model of the system and generating the dynamic data is the second source. A dynamic model can be used to represent the system, with the required input variables, as it progresses over time, see dynamic modelling section 2.2.

Many authors have integrated ABC and dynamic modelling. Spedding et al (1999), Savory et al (2001) and Savory et al (2010) use ABC approaches within a dynamic model to address the time consuming, costly and difficult implementation of ABC. They conclude that integration provides greater detail to the cost estimate by incorporating the dynamic variations of a manufacturing system. Costa et al (2010) integrated ABC with a dynamic model to quantify the what if scenarios within the dynamic model. Beck et al (2000), Lee et al (2001) and Andersson et al (2012) use dynamic modelling output within a ABC model to improve accuracy. This is because resource requirements, therefore the costs, are more accurate and arbitrary allocation is avoided. Lee et al suggests that integration gives a greater understanding of the dynamic nature of the component cost within a manufacturing system. Marsh et al (2010) created a system that coupled a DES model with a generative cost model. Marsh et al concluded that their system allowed sharing of data and results that enabled informed decision making for process planning and more accurate results for assembly costs.

Tse et al (Tse and Gong 2009) suggested a Time Driven Activity Based Costing (TDABC) model to account for idle resources in resource pools. This is an improvement over the traditional ABC models and in some simple cases may be the correct approach; however it is still a static model therefore the dynamic interactions of a system will not be taken into account. Which is why the integration of a dynamic model with a ABC model is proposed.

2.1.3 Summary

A limitation of the main cost estimation methods is that they employ static models which are unable to fully represent dynamic systems. A solution to this is to utilise dynamic data. Two sources of dynamic data are: to collect it from a real system or to generate it with a dynamic model. Improvements in ability and model accuracy have been shown with the integration of a cost model to a dynamic model within the literature. Therefore a dynamic model will be utilised, in this research which is discussed in section 2.2.

ABC allocates all costs, including overheads, within a manufacturing system to individual components. Many authors (Asiedu and GU 1998; Mikko, Marko et al. 2007; Askarany, Yazdifar et al. 2010; Costa, Montevechi et al. 2010; Savory and Williams 2010; Wang, Du et al. 2010; Andersson, Bj et al. 2012) have shown cost estimation improvements by utilising ABC over other cost estimation methods However a disadvantage of ABC is that substantial amounts of detailed data, concerning the manufacturing system and the manufacture of components that utilise it, are required to complete the estimate. The dynamic modelling method chosen also requires substantial amounts of detailed data (section 2.2.1.4), it is therefore prudent to utilise ABC as the costing estimation method, because the detailed data required for the dynamic model can also be utilised by the ABC model.

2.2 Dynamic modelling

If a system does not exist a model can be created to replicate its behaviour in order to experiment on it. There are two types of model shown in Figure 6, mathematical and physical, as described by Law and Kelton (1992). In many situations, and increasingly with computer utilisation, a mathematical model has advantages over a physical model. For instance a physical model requires manufacturing, and cannot be changed easily once created. There are two categories of mathematical model, analytical and dynamic. Analytical represents the use of mathematical methods to obtain an exact solution to a problem.

Dynamic modelling, often called simulation, usually refers to the modelling of a system, which has stochastic elements, as it progresses through time.

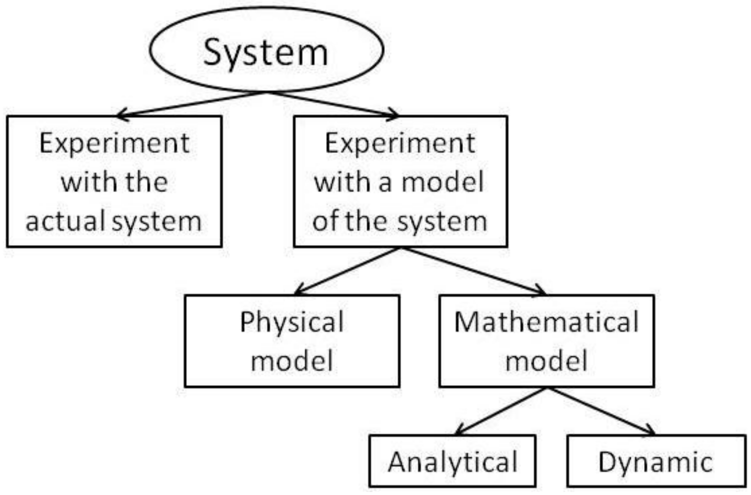

System
Experiment with the actual system
Experiment with a model of the system
Physical model
Mathematical model
Analytical
Dynamic

Figure 6: Ways to study a system (Law and Kelton 1992)

2.2.1 Forms of simulation

Simulation can be classified into two forms as shown in Figure 7; these are:

- Time-driven simulation, where the model time progresses either continuously, by fixed time points or by regular intervals, and the state variables change continuously with respect to time
- Event-driven simulation, which progresses through time by advancing from event to event resulting in variable time steps, and the state variables change only at these discrete points in time

Also shown in Figure 7 are four modelling methods, two for each classification. The modelling methods for time driven classification are: system dynamic and continuous. For event driven classification they are: agent based and discrete event. Each of these modelling methods will now be discussed.

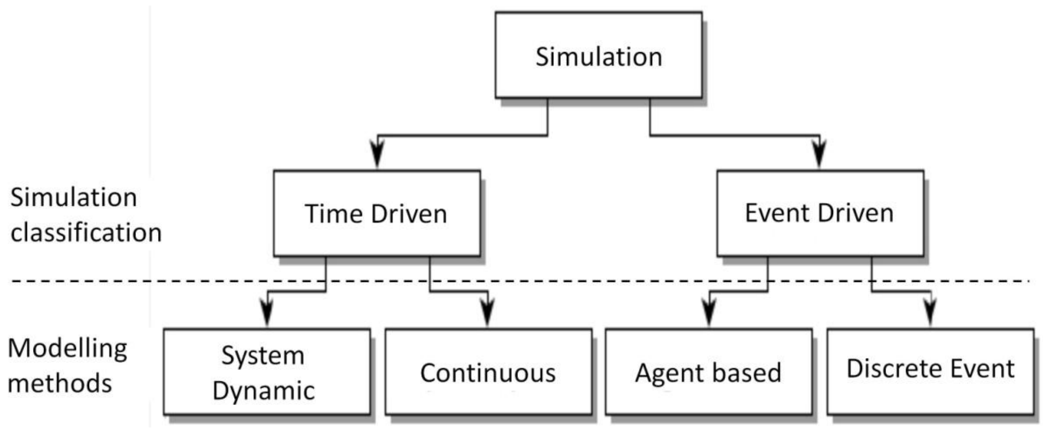


Figure 7: Simulation classification and modelling methods. Modified from (Yu 2008)

2.2.1.1 System dynamic

System Dynamic (SD) modelling was developed by Forrester (1961) and is defined as the study of information feedback to understand how organisational structure, amplification and time delays interact to influence the system. SD is typically used to model systems where abstraction is high and details are low, examples include: water resource management (Winz, Brierley et al. 2009), social epidemiology (Galea, Hall et al. 2009), ecological systems (Miller, Cable et al. 2012), energy policy modelling (Qudrat-Ullah and Seong 2010) and charging control of batteries (Huang, Hsu et al. 2010). Since the emergence of agent based modelling (section 2.2.1.3) the use of SD modelling has reduced to specific applications such as where there are flows, or the overall system dynamics is required (Norling 2007). For these reasons SD is not a modelling method normally used for manufacturing or supply chain simulation.

2.2.1.2 Continuous

Continuous modelling is used to model systems where state variables change continuously with respect to time such as the depth of water in a container (Sokolowski and Banks 2011). A computer cannot represent time in a truly continuous manner therefore it simulates time passing continuously by utilising very small fixed time steps (Yu 2008). The state variables of a continuous simulation are re-evaluated at each time step, this can result in model run times that grow with the increase in required model simulation time. Differential equations are typically used within continuous models (Wainer 2009) and many continuous models try to transfer the complex discrete parts of the model to the continuous level because differential equations allow for fast simulation times. To do this individual discrete elements are converted to dynamic flows for example parts per unit time (D'Apice, Herty et al. 2010). A result of this is that some of the fine details of the system are lost. For these reasons continuous simulation is not normally used for detailed manufacturing simulation of components passing through a series of manufacturing operations within a supply chain. Therefore continuous modelling will not be used within this thesis.

2.2.1.3 Agent based

Agent Based Modelling (ABM) has no central control of the model unlike the other modelling methods. Instead control is distributed among the agents which have a well defined sphere of influence (Yu 2008). A defining characteristic of agent-based modelling is the ability of individual agents to exhibit autonomous action. This is possible because an agent is governed by rules that control its behaviour, accept inputs from its environment, learn from previous experience, adapt to future actions and to communicate with other agents (Yu 2008). A result of the interactions between agents is a behaviour which is not explicitly programmed and can be unpredictable.

In recent years ABM has seen increased use in many areas of research including: geographical system simulation (Crooks and Heppenstall 2012), social economic system simulation (Birkin and Wu 2012), transportation system simulation and virus transmission (Cheng, Qiu et al. 2012), generative social science modelling (Epstein 2011), ecosystem services modelling (Murray-Rust, Dendoncker et al. 2011) and economic modelling (Farmer and Foley 2009). In the manufacturing simulation area ABM has been utilised to develop manufacturing control systems (Anosike and Zhang 2009; Leitão 2009) and manufacturing schedules (Ouelhadj and Petrovic 2009). All these areas of research have one similarity which is that they all model, in different levels of abstraction, behaviour of systems that contain either humans or animals. The ability of ABM to model behaviours with simple agents is the primary reason it is used. This is also why it is not normally utilised to model the flow of products through a manufacturing supply chain to determine bottlenecks and optimum resource requirements. Agent based modelling is not suited to model manufacturing systems to a detailed level.

2.2.1.4 Discrete event

A discrete-event model typically describes a logical sequence of activities. These activities can, for instance, represent a process in a manufacturing system. From an abstract point of view an activity represents a time delay. Entities move through the model; in a manufacturing system entities could represent individual components. When an entity enters an activity an event occurs instantaneously in simulated time that acts on the entity changing its state, also the event can trigger other events to occur. For these reasons DES is a widely used method for studying the design and operation of manufacturing systems (Knoll and Heim 2000; Creighton and Nahavandi 2003; Venkateswaran, Young-Jun et al. 2004; Cassandras and Lafortune 2008; Sharda and Bury 2008; Persson and Araldi 2009; Robinson, Brooks et al. 2010; Fischbein and Yellig 2011; Sajadi, Seyed Esfahani et al. 2011; Wainer and Mosterman 2011; Chen, Mockus et al. 2012; Turner, Madachy et al. 2012) and is industry best practice.

The primary disadvantage of a DES model is the detail required to build the model, both in knowledge in the form of logic concerning the process flow and decisions, and the data to populate each activity within the process flow (Caro, Möller et al. 2010). However the level of data and knowledge required is similar to that required to create a ABC model.

Advantages of using DES (Law and Kelton 1992; Robinson 2004; Jahangirian, Eldabi et al. 2010) for simulating manufacturing systems, include:

- Complex systems can be described including stochastic elements
- Individual entities can be tracked, allowing performance data to be collected
- The ability to integrate individual stages of a system allows a detailed view to be taken (Kendall, Mangin et al. 1998; Denkena, Rudzio et al. 2006)

Any of the four methods of simulation could solve any problem, however each method has an area of expertise that allows it to solve a problem with less modelling complexity than the others. Pidd (2009) emphasises, throughout his book, that all models should be kept simple and within scope of the problem. It is therefore a logical step that discrete event simulation will be utilised as the modelling method within this thesis as it is most suited to simulate a long time duration logical process sequence system such as a manufacturing supply chain.

2.2.2 Data driven generic modelling

Robinson (2004) suggests separating the data and results from a model (Figure 8). Holding model data, experimental factors and results separately from each other and outside the simulation model, usually in a spreadsheet, database or data file, has various advantages, including:

- Ease of use - understanding of the simulation code is not required by the user as they are able to modify the input parameters and data from outside the model
- Version control - a record of all experimental factors associated with the results can be kept
- Further analysis - by storing the results outside the simulation, specialist software can be used for further analysis

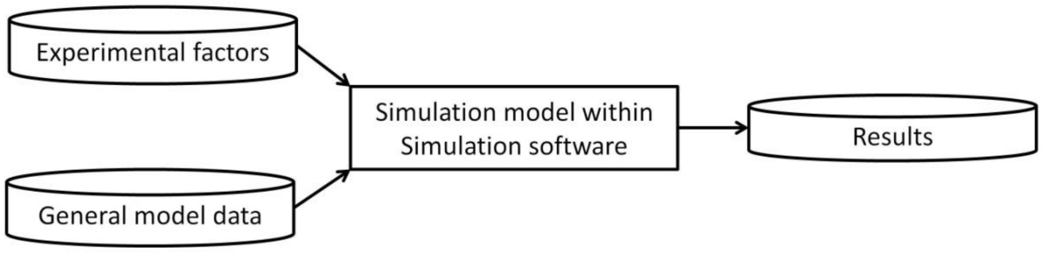


Figure 8: Separate simulation model from model input and output data (Robinson 2004)

Robinson’s suggestion is, in an abstract form, a description of a data driven generic model. Pidd (1992) defines a generic model as a model with the ability to cope with a range of structurally similar systems. Brown (2010) clarifies the definition by adding that the generic model is logic only. Pidd defines a data driven model as a model that has the ability to fully specify any instance of a system without the need for programming. Therefore a generic model must first be created then a data source can populate it for each instance required. Tannock et al (2007) defines a data driven model as a model which is constructed automatically by a model builder software program based on pre-existing user data. These two definitions lead to the same result but are different because Tannock’s definition uses data to construct the model, whereas Pidd’s utilises a generic model which is populated with data for a particular system. They both, from the user’s perspective, have some form of 'black box' elements to their operation because the user only deals with data input and output. Therefore the simulation model, either generic or created, does not have to be seen by the user.

There are examples of use for both definitions. Many authors have developed generic data driven simulation models: McLean et al (2002) developed one for small machine shops; Kibira et al (2007) developed one for an automotive supply chain; and Brown et al (Brown and Powers 2000) developed a military aircraft resource maintenance model. This research will utilise Pidd and Brown's definition by creating a generic model that is populated by a data source.

All data driven models require a data source; some sources utilised by authors include: spreadsheets (Curran, Gomis et al. 2007; Nasereddin, Mullens et al. 2007), databases (Randell and Bolmsjo 2001; Son and Wysk 2001; Son, Wysk et al. 2002; Neugebauer, Plonnigs et al. 2004; Cao, Farr et al. 2005), and CAD layouts (Moorthy 1999; Paprotny, Zhao et al. 1999; AbouRizk and Mather 2000; Kim, Jeong et al. 2009; Wy, Jeong et al. 2011). This research will utilise a database due to the different data types that will be used, ease of integrating to code and capabilities of searching and linking data.

A disadvantage of generic data driven models is the trade-off between flexibility and scope (Brown and Powers 2000). A generic model needs to be flexible to enable a user to complete necessary simulation experiments, yet a generic model could never have the scope to cover all possible experiments without becoming overly complex (Pidd 2009). Two other disadvantages of generic data driven models is that they require: access to externally available structured data; and the simulation tool must contain a code execution capability that can dynamically create and configure complex models.

All the authors using generic data driven models agree that there are advantages, including: reusability within the scope of the generic model or model builder; and, models should contain fewer bugs than traditional methods because a generic model or model builder requires validation of logic only.

Generic data driven models however are not a replacement for general purpose simulation tools (Cao, Farr et al. 2005; Tannock, Cao et al. 2007) because they require extra effort to build initially and in some cases may only be used once. Also, the payback of creating a generic data driven model requires it to be used many times to create models for a variety of solutions. The choice of which method to utilise to create a generic data driven model is, it seems, dependent on the situation, the data and the tools available.

2.2.3 Summary

Dynamic modelling is utilised because cost, time and practical considerations do not allow experimentation with the actual system, or a physical model. DES is an event driven dynamic modelling method that is typically utilised by research and industry to model manufacturing systems. The other dynamic modelling methods (SD, continuous and agent based) could be coerced into a functional model but are not suited to the specific system and problem combination within this research. For these reasons a DES model will be utilised within this research.

A generic data driven model combines the use of either a logic based model that can cope with a range of structurally similar systems populated by external data, or a model constructed from data via a model builder program. The main disadvantages of generic data driven models include the trade-off between flexibility and scope and access to structured data. The primary advantage is reusability within the scope of the model.

It is proposed that the generic data driven modelling methodology will be used to conduct supply chain simulations specified by the hypothesis. A DES methodology will be used within the generic model. It is also proposed that the generic model should be capable of optimising the input parameters and a database should be the data source.

2.3 Integrating geometry

Data required by a dynamic model of a manufacturing system includes: a manufacturing process plan (section 2.3.2) and process times for each stage of the plan. Design geometry is the start and focal point for generating the data required in a dynamic model. Integrating the design geometry should allow this data to be determined automatically.

There are two methods that could aid the integration. The first, Automated Feature Recognition (AFR) extracts design geometry in a recognisable format. The second, Computer Aided Process Planning (CAPP), determines a manufacturing process plan by, in most cases, utilising the output from AFR. Each of these methods will be discussed in further detail.

2.3.1 Automated feature recognition

Shah (1991) gives a definition of a feature as a “representation of engineering meaning of the geometry of a part or assembly”. However, an agreement on a set of features that can represent all applications, has thus far eluded the feature community (Han, Pratt et al. 2000; Marchetta and Forradellas 2010). A reason for this is that specifying a feature “requires knowledge of the context or application domain in which the geometry has a meaning and an interpreter" (Brown, McMahon et al. 1995). There are many types of features for different applications (Brown, McMahon et al. 1995) two of these are: design and manufacturing. Design features are developed for a design engineer to use. Manufacturing features represent a feature that a specific manufacturing process would create, such as a hole or a slot. A combination of manufacturing features can be used to create almost all design features. However it could be possible to create design features that may be impossible to create via a specific manufacturing process. Therefore design features should be specific to manufacturing processes and follow rules that would ensure the design features are always manufacturable. There are many challenges with creating a library of features, however without this understanding AFR would not know what a feature was (Xu, Wang et al. 2010).

The goal of AFR is to recognise features without intervention from a manufacturing or design engineer (Babic, Nesic et al. 2008; Babic, Nesic et al. 2011; Brousseau and Eldukhri 2011). This goal is in line with Han et al (2000) who believes that designers should be given flexibility to design and that AFR systems should be used to convert designs into feature models for use in other systems. Supplying recognised features in a valid format to required systems is therefore an important secondary goal for AFR systems.

There are two tasks that limit AFR's wider utilisation: interacting features, otherwise known as component complexity, limits AFR ability to recognise features (Fu, Ong et al. 2003; Gao, Zheng et al. 2004; Abouel Nasr and Kamrani 2006) (Chu, Tang et al. 2011); and the complexity and scalability of the algorithms (Babic, Nesic et al. 2008; Verma and Rajotia 2010). The inability to recognise features that are interacting limits the possible uses of AFR to geometries that contain independent features. For these reasons an overall AFR algorithm for complex component geometries with interacting features requiring multiple manufacturing processes does not exist with little evidence that one will emerge in the near future.

2.3.2 Computer aided process planning

Process planning is defined as the preparation of a set of instructions, that detail which manufacturing processes and machines should be used, in a specific sequence, to manufacture a component design specification from raw material to finished product (Marri, Gunasekaran et al. 1998; Kumar and Rajotia 2005;

Phanden, Jain et al. 2011). A process plan contains the following information (Marri, Gunasekaran et al. 1998; Kumar and Rajotia 2005):

- Operation sequence
- Material specifications
- Cutting tools
- Manufacturing methods
- Processing times
- Setup details

CAPP is defined as a system that can interpret a component design in terms of features and use knowledge bases to perform process planning tasks that can optimise cost and time (Marri, Gunasekaran et al. 1998; Kang, Han et al. 2003; Zhou, Qiu et al. 2007).

CAPP aims to provide a link between design and manufacturing by linking Computer Aided Design (CAD) and process planning activities (Marri, Gunasekaran et al. 1998; Feng 2003; Zhou, Qiu et al. 2007). The ultimate aim of CAPP is to achieve automated process planning and remove human judgment (Zhou, Qiu et al. 2007). A method to achieve this is to utilise AFR output data. CAPP has multiple benefits (Giachetti 1998; Gupta, Chen et al. 2003; Kumar and Rajotia 2005; Xu, Wang et al. 2010), such as:

- Reduced time between component design and manufacture
- Reduced clerical load of plan preparation on manufacturing engineers and skilled process planners
- Optimised process plans
- Aiding design decisions about component geometry, manufacturing processes and materials

2.3.2.1 CAPP system designs

Two classifications of CAPP system include (Marri, Gunasekaran et al. 1998; Zhou, Qiu et al. 2007; Xu, Wang et al. 2010) variant and generative. Variant approaches represent a method of retrieving data from a database. Data is categorised into component families distinguished according to their manufacturing characteristics, where standard process plans can be created for a family. This approach is limited for new components because they must be classified into families and other sub categories. Novel components that contain unknown geometry or manufacturing processes may require new classifications and data within the database.

The generative approach synthesises a process plan based on analysis of the feature geometry and other factors that may influence the manufacturing decision. Park (2003) suggests a third type, called knowledge based. However his suggestion, in an abstract form, is an extended generative approach. He suggests that a process planning framework should be controlled and customised by dynamic rules as shown in Figure 9 (b) instead of a set of rules that are structured in a program like form (Figure 9 (a)). Park also suggests that the knowledge base should be constructed from four knowledge elements:

- Facts - which cover data objects
- Constraints - correspond to technical constraints of process planning
- Ways of thinking - which imitates intelligence
- Rules - represent key parameters that control the way of thinking

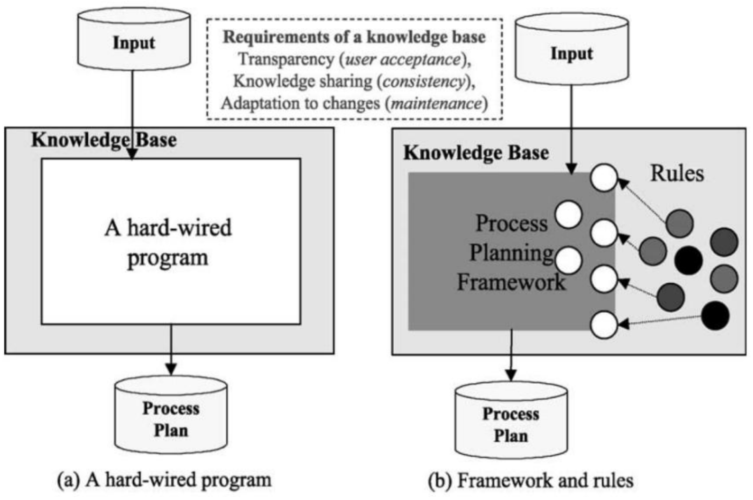

Input
Requirements of a knowledge base
Transparency (user acceptance),
Knowledge sharing (consistency),
Adaptation to changes (maintenance)
Input
Knowledge Base
A hard-wired program
Knowledge Base
Rules
Process Planning Framework
Process Plan
Process Plan
(a) A hard-wired program
(b) Framework and rules

Figure 9: Requirements of a knowledge base (Park 2003)

The trend of CAPP is towards the generative approach coupled with a knowledge base (Xu, Wang et al. 2010) because it allows a more automated process therefore reducing the need for manufacturing engineers and reducing the time to generate the process plan. Recent enhancements within the field see the introduction of technologies to enable self learning and adaptation to new data, aid integration to schedules and aid optimisation of parameters. Some of the technologies utilised include: neural networks (Guangru and Xiaoliang 2010; Deb, para-Castillo et al. 2011; Wang, Zhang et al. 2012), genetic algorithm (Salehi and Bahreininejad 2011; Wei-jun and Yu-jin 2011) and agent based simulation (Li, Zhang et al. 2010).

The limiting factor within CAPP is the inability of AFR to supply feature data of a component in a format for CAPP systems to use which is more pronounced for complex geometry containing interacting features. This is therefore a limit to the generative approach as it relies on feature data from an AFR system.

2.3.3 Summary

Extracting data from design geometry is required to generate a process plan. There are two technologies which can aid this, these are AFR and CAPP. AFR interprets geometry by recognising features without intervention from a manufacturing or design engineer. This is then supplied to a CAPP system which aims to provide a link between design and manufacturing by automating process planning activities.

There are two types of CAPP systems: variant and generative. Variant approaches represent predefined family based process plans. Generative approaches generate a process plan based on geometry, knowledge and dynamic rules. However the generative approach is limited in terms of geometry due to the limitations of AFR which has limited ability interpreting complex geometry when features interact.

It is proposed that until AFR technology can extract the necessary data in a suitable format a direct approach should be taken therefore bypassing AFR. This direct approach will link directly to the geometry. This reduces the scope of parameterised component geometry but will increase system automation by reducing complexity. The direct approach will allow extracted data to be supplied to the downstream CAPP system in a suitable format. To increase automation of the system and allow real time decision making for the user a variant based CAPP approach will be utilised.

2.4 Supply chains

Beamon (1998) defines a supply chain as “an integrated process wherein a number of various business entities (i.e., suppliers, manufacturers, distributors, and retailers) work together in an effort to: (1) acquire raw materials, (2) convert these raw materials into a specified final component, and (3) deliver the final components to retailers”. Beamon points out that although a supply chain is comprised of multiple business entities, the supply chain is considered a single entity. This definition of a supply chain allows it to be applied to multiple companies or within a single company as an internal supply chain where the suppliers are business units within the same company as within Rolls-Royce.

Beamon describes a supply chain that comprises two integrated processes: a production planning and inventory control process, and a distribution and logistics process. Min et al (2002) also combines two processes: a physical distribution process (Bowersox and Closs 1996) which is similar to Beamons' distribution process and a material management process (Johnson and Malucci 1999). A combination of these two supply chain descriptions is shown in Figure 10. Also shown in Figure 10 is the flow of components passing through the supply chain and the flow of information which moves in an opposite direction to the flow of components. A third flow is from third party suppliers, who do not supply products that are used directly in the finished component, but which are used to support its manufacture.

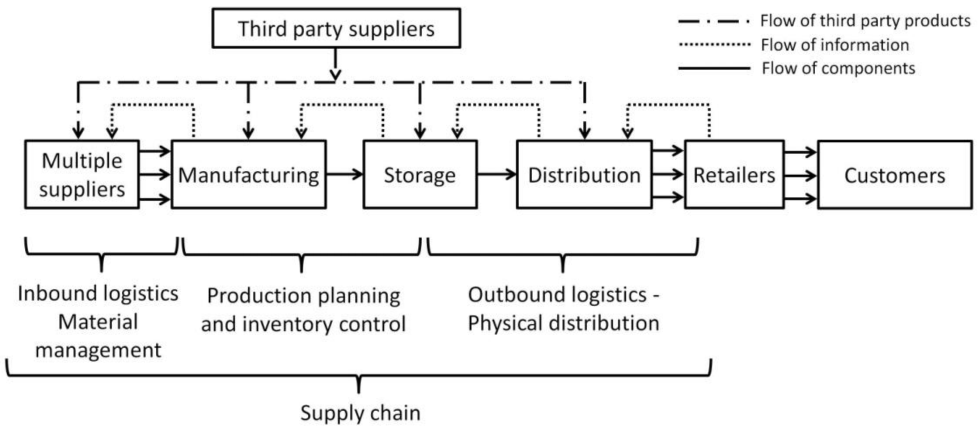

Third party suppliers
Flow of third party products
Flow of information
Flow of components
Multiple suppliers
Manufacturing
Storage
Distribution
Retailers
Customers
Inbound logistics Material management
Production planning and inventory control
Outbound logistics - Physical distribution
Supply chain

Figure 10: The supply chain process. Modified from (Beamon 1998; Min and Zhou 2002)

A building that contains a manufacturing process is called a factory. This factory can be a part of a supply chain or can be thought of as a small supply chain in itself. This small supply chain has all the characteristics of a large supply chain but does not span multiple businesses or locations. The only distinguishing aspect between a small and large supply chain is where the scope of the supply chain ends. For instance a supply chain may encompass the manufacture of a jet engine which contains thousands of components, or it may be for a single component.

For this research a supply chain will be considered dedicated to the manufacture of a single component from supply of the raw material, through manufacture, to delivery of the finished component. Also the supply chain will not be constrained to a single location, therefore the component may require transportation between locations.

2.4.1 Resource modelling

Resources are items that are required by the supply chain to manufacture the component, some examples include: machines, equipment, fixtures, and human resources. Each resource has a utilisation maximum, where utilisation is the amount of time spent working against the total available time. Rules concerning resources within this research include:

- A finite amount of each resource is contained within a supply chain
- A manufacturing operation may use more than one resource, of the same or different type, at a time
- A manufacturing operation cannot be started unless all required resources are available
- Mean utilisation of a resource cannot exceed the utilisation maximum value

Most resources are constrained to a single location such as a factory, where the factory may manufacture many different components. Therefore resources in a factory may not be constrained to the manufacture of a single component. In this research however resources will only be utilised for the manufacture of the component being considered. This represents a significant limitation, however it is acceptable as a method to prove the concept of integrating a dynamic model into a unit cost estimation so support design decision making.

2.4.2 Inventory control

Traditional manufacturing organisations have been based on a 'push system', whereas many modern manufacturing organisations have endeavoured to become efficient and lean by removing waste, such as inventory. One method of achieving lean manufacturing is Just-In-Time (JIT) manufacturing which is a 'pull system' developed by Toyota (Taiichi 1988). In a push system a production rate is predetermined, and then materials are pushed through the system to achieve it. In a pull system customer demands drive production rate. Demand for a product from the customer pulls the necessary parts from the previous step to fulfil the requirement. The pull signal propagates down through the system in order to replenish the parts ready for the next demand signal; this is shown in Figure 10 as the flow of information.

Kanban stands for card in Japanese (Aytug and Dogan 1998) because originally Toyota used simple cards to implement a JIT manufacturing system by passing them from one process to another to represent the information flow. A kanban contains information about the type and quantity of the component it represents. Kanbans are used to limit the level of Work In Process (WIP) and coordinate the flow of information and material.

2.4.3 Manufacturing batch operations

The manufacturing stage of the supply chain is a combination of manufacturing processes such as turning, milling and heat treatment. A manufacturing process can be split up into operations, for instance a turning process entails a setup operation, multiple turning operations to form the desired shape and a set-down operation.

If a piece of equipment can conduct an operation on two or more components at a time it is classed as a batch operation. Batch operations usually wait until there are enough components available to fill the equipment by holding them in a storage area until required. It is not operationally necessary to fill the batch operation; however it is required for cost efficiency as it increases the utilisation of the equipment which allows the costs associated with the operation to be spread amongst the maximum number of components.

The resource requirements of a batch operation are difficult to calculate in a static model due to the interactions between components entering the batch operation, available batch operation resource and the requirement to fill the resource. Also the batch flow of components leaving the batch operation can affect the quantity of resources in the preceding operations.

2.4.4 Summary

A supply chain is an integrated process where a number of business entities work together in an effort to fully manufacture and deliver a component. Some supply chains encompass multiple components, however for this research a supply chain will be considered dedicated to the manufacture of a single component.

A resource is an item required by a manufacturing process to complete an operation. Multiple resources of the same or different type may be required by a manufacturing operation at a time and the operation cannot start until all resources are available. Also a supply chain contains a finite amount of each resource and for this research will be dedicated to a single supply chain.

Kanban is a term used to denote an implementation of a pull system within a supply chain. Kanban controls the level of WIP and the flow of information. A kanban system is proposed as a method to achieve a pull system within the dynamic model.

Batch operations are a type of operation which can process more than one component at a time. The resource requirements of a batch operation are difficult to calculate in a static model due to interactions between components entering the batch operation, available batch operation resource and the requirement to fill the resource.

2.5 Chapter summary

Four areas have been reviewed these are: cost estimation, dynamic modelling, integrating geometry and supply chains. The main conclusions from these reviews were:

- Static models used within cost estimation have difficulty in fully representing dynamic systems. Mitigation of this limitation is to integrate a dynamic modelling capability that can supply dynamic data to a cost estimation model. ABC is proposed as the cost estimation method to utilise because it has the ability to utilise the detailed data from the dynamic model therefore providing benefit over the other cost estimation methods reviewed. Also the primary disadvantage of ABC, which is substantial data requirement, would be lessened as the integration of a dynamic modelling capability which also requires substantial data would utilise the data.
- A generic data driven DES model is proposed for two reasons. First a generic data driven model can be reused within the scope of the model therefore reducing complexity of creating a model for each different scenario. Second a DES model is recognised as best practice for modelling manufacturing systems.
- Integrating geometry into an automated system requires extraction of data. AFR is a technology that has aims in-line with this; however it has limited ability to achieve it. CAPP systems are used to generate a manufacturing process plan from geometry; however they also have limited capabilities. Therefore a direct approach of integrating component

geometry to extract necessary data, and a variant based CAPP approach of utilising predefined family based process plans is proposed.

- Some supply chains encompass multiple components, however for this research a supply chain will be considered dedicated to the manufacture of a single component consistent with high production rates. Also resources required by the supply chain will be considered dedicated to the supply chain. A kanban system is proposed as a method to achieve a pull system within the dynamic model.

The literature review has shown that there has been extensive research in cost estimation and dynamic modelling with some authors linking both areas. Other authors have used geometry to automate the process of cost estimation and to aid dynamic model generation. However no research has linked all three areas. By integrating all three areas component geometry can be used in real time to drive the creation and optimisation of a dynamic model to aid unit cost estimation and therefore real time decision making.

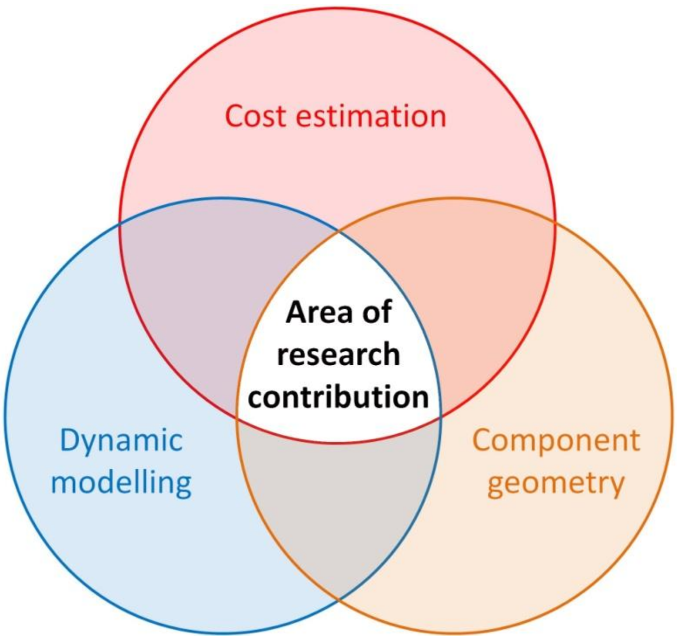


Figure 11: Area of research contribution

Chapter 3

Framework

This section describes a working proof of concept framework that integrates a dynamic model with an ABC based cost model for multiple supply chain options from CAD geometry of a component. The framework is classed as semi-automatic because user interaction is required to create the geometry, supply production rate data and select supply chain options. The framework generates an aggregated unit cost (AUC) and manufacturing data for each supply chain option chosen by the user. The output from the framework can be used to aid design decisions and compare supply chain options of a component. Reference to parts of the integration code held in the appendix is given to clarify necessary sections. The whole integration code is included on a CD because it contains 15,000 lines.

3.1 Structure

There are five stages within the framework, as shown in Figure 12, these are:

1. Geometry modification – allows the user to modify parameterised geometry which the system extracts via the geometry engine
2. Determine manufacturing process – extracts production rate and supply chain options from the user
3. Manufacturing process data generation – selects resources and generates operation data
4. Dynamic modelling – conducts the dynamic simulations and optimises the inputs
5. Aggregated unit cost – calculates the cost using an ABC based cost model and outputs AUC and manufacturing data to the user

These five stages form the framework which sits within a design iteration loop. This loop allows geometry parameters to be change until all specification parameters are achieved, of which the framework can provide two: unit cost and manufacturing data.

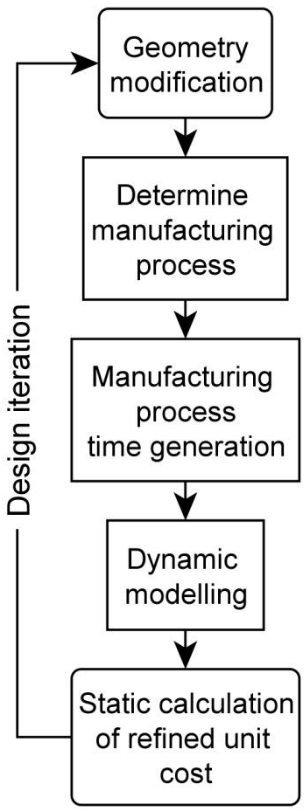

Geometry modification
Determine manufacturing process
Manufacturing process time generation
Dynamic modelling
Static calculation of refined unit cost
Design iteration

Figure 12: Framework stages

The framework schematic in Figure 13 shows how the user and database interact with the five stages that make up the core of the framework. The blue arrows show data entering the core and the red arrows show data leaving the core. Stages 3, 4 and 5 have an iteration loop which allows each supply chain option chosen by the user to be processed. Each stage of the framework will now be described including, where appropriate, reference to the code used. The database will be discussed in its own section as it, along with the integration code, forms a backbone to the framework.

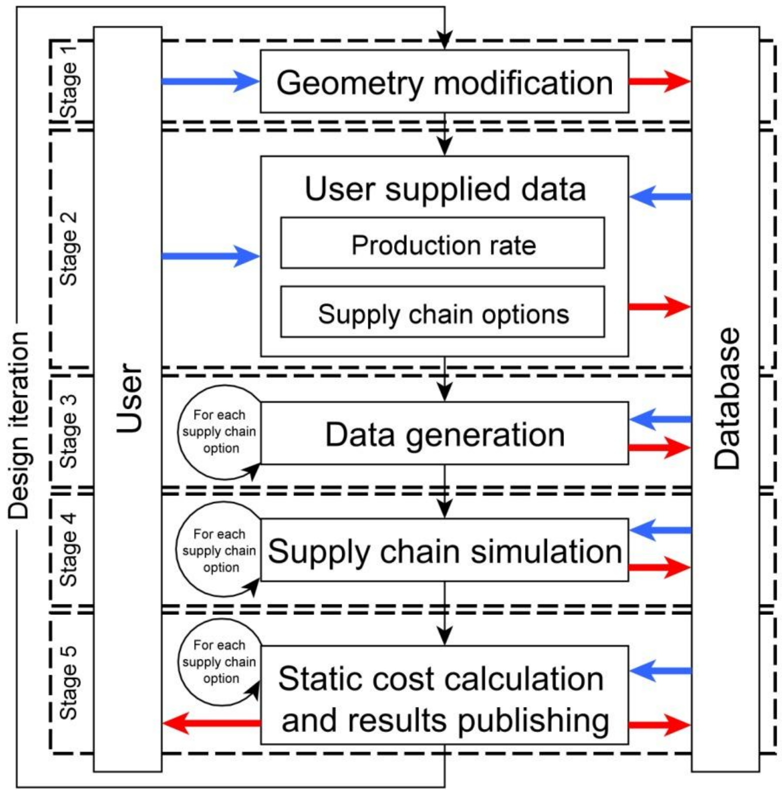


Figure 13: Framework schematic

3.1.1 Stage 1 – Geometry modification

The aim of this stage is to extract geometry data from the geometry engine into the system database so the system can utilise it. To do this, stage one contains three sub-stages as shown in Figure 14.

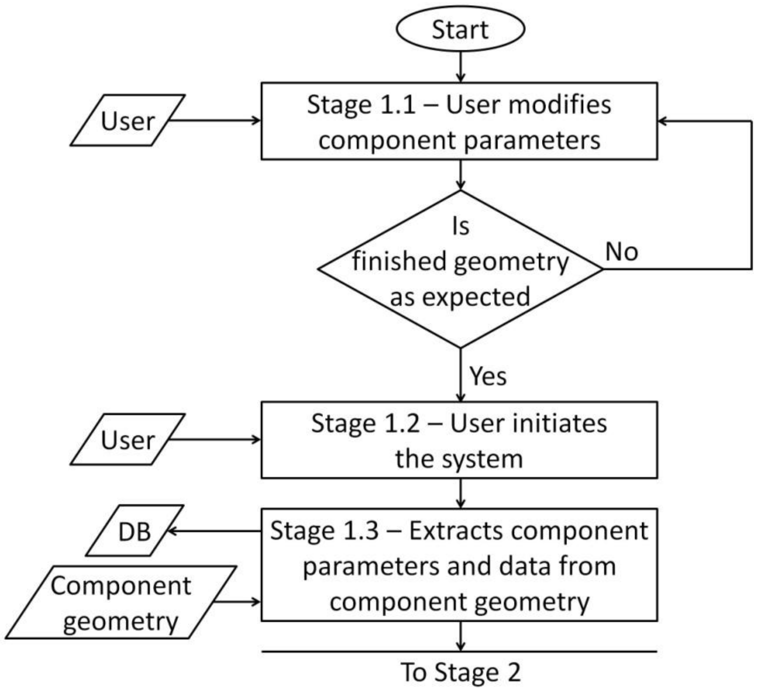

Start
User
Stage 1.1 – User modifies component parameters
Is finished geometry as expected
No
Yes
User
Stage 1.2 – User initiates the system
DB
Component geometry
Stage 1.3 – Extracts component parameters and data from component geometry
To Stage 2

Figure 14: Framework stage 1 flow chart

In the first sub-stage, 1.1, parameterised solid model geometry, based on the family type of the component, is modified by the user, from within the CAD engine. The CAD tool used was Siemens NX6 as this is used by Rolls-Royce, however any CAD engine that can create parameterised geometry and contains an Application Programming Interface (API) that allows extraction of the geometry could be utilised. Linked to the parameterised component geometry, are a series of parameterised state geometries that build up depending on rules from one to the next to form a parameterised geometry that represents the Condition Of Supply (COS). The COS represents the shape of raw material at the start of manufacture. The state geometries represent states that the component must pass through during manufacture from initial COS geometry to finished geometry. Figure 15 shows three states of a component (S1, S2 and S3), between each of these states is a transformation stage (T1 and T2), which represent the multiple manufacturing options to transfer from one state to the next. Figure 16 shows screen shots of case study one geometry at three different states as described in Figure 15.

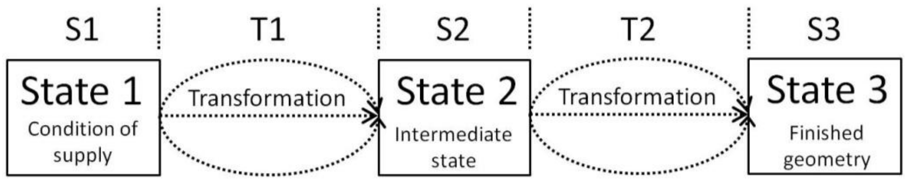


Figure 15: State geometries

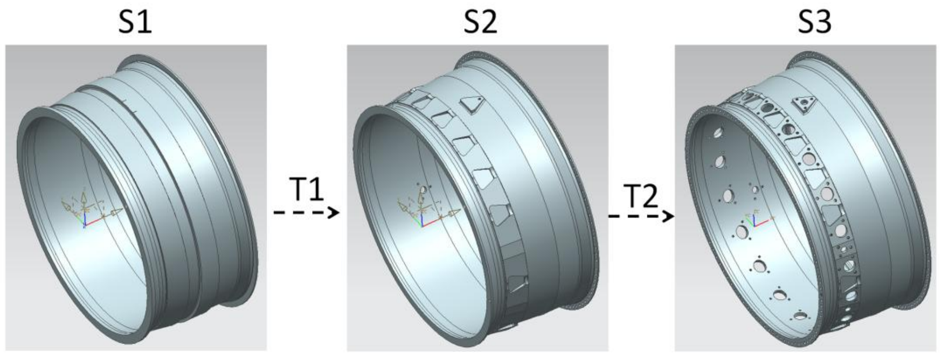


Figure 16: Screen shots of case study one component at three different states

Different methods of manufacture, called supply chain types, have different COS and intermediate state geometries. Therefore each supply chain type requires its own COS and intermediate state geometries which are linked together in an assembly. A schematic of the CAD implementation is shown in Figure 17 and a screen shot of the implementation from case study one is shown in Figure 18 which also contains overlays to show the different aspects of the schematic on the screen shot . The user interacts with the supply chain collection, which holds each supply chain type assembly; in Figure 17 there are two: powder Hot Iso-static Pressing (HIP) (see section 4.1 for information on HIP) and Forging. Contained within these linked assemblies are geometry states for each supply chain type; HIP has three and forging has five. The user only interacts with the component state through the component parameters via the supply chain collection. When the geometry is updated the parameters are sent to the lowest level, the component, then each level is updated in turn based on rules and the previous level. This functionality is fully implemented within the geometry engine, no additional code or capability was required.

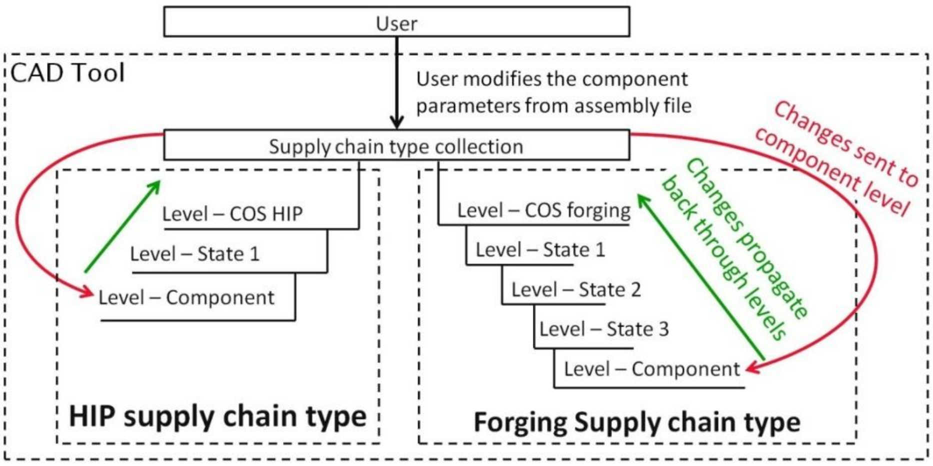
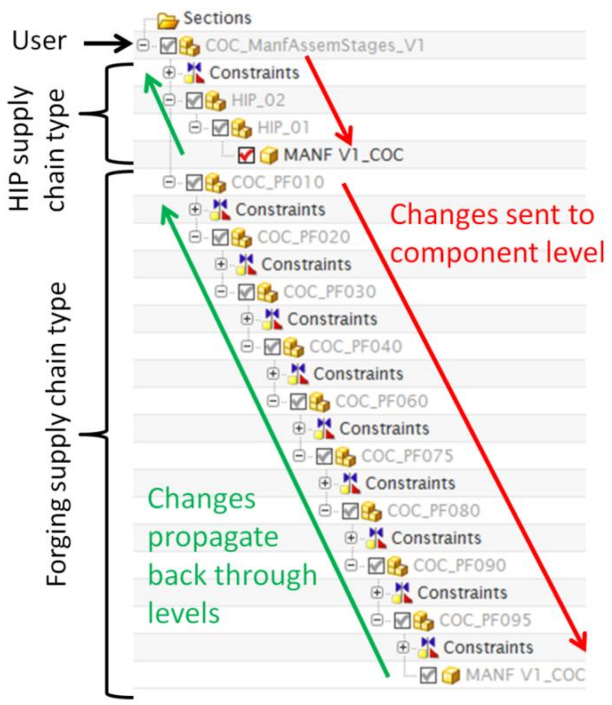

User
CAD Tool
User modifies the component parameters from assembly file
Supply chain type collection
Changes sent to component level
Changes propagate back through levels
Level – COS HIP
Level – State 1
Level – Component
Level – COS forging
Level – State 1
Level – State 2
Level – State 3
Level – Component
HIP supply chain type
Forging Supply chain type

Figure 17: Schematic of CAD implementation


User
HIP supply chain type
Forging supply chain type
Changes sent to component level
Changes propagate back through levels

Sections
COC_ManfAssemStages_V1
Constraints
HIP_02
HIP_01
MANF V1_COC
COC_PF010
Constraints
COC_PF020
Constraints
COC_PF030
Constraints
COC_PF040
Constraints
COC_PF060
Constraints
COC_PF075
Constraints
COC_PF080
Constraints
COC_PF090
Constraints
COC_PF095
Constraints
MANF V1_COC

Figure 18: Screen shot from case study one of implementation of schematic in Figure 17 with overlays to shows different aspects of the screen shot.

When the user has finished modifying the finished component geometry the user executes sub-stage 1.2 by initiating the remainder of the framework from a menu button within the geometry engine. The Framework executes automatically from this point until the end of stage 5. In a production version of the framework the vision is that the user would not see any change on their screen, except for dialog boxes asking further questions from stage 2, until the framework had finished executing.

Sub-stage 1.3 extracts all parameters from the geometry via the geometry engines API. In NX6 the API allows any .net coding language, C++ or Java. The author completed all coding work in C# utilising Microsoft Visual Studio 2008. The code in Appendix A.1 forms part of the integration code and is used to extract the parameters from the geometry engine utilising the API. The geometry extraction code is hard coded to the geometry, therefore any fundamental changes to the geometry, such as re-drawing aspects of the geometry, will likely break the hard code links due to changes in how the code identifies geometry features.

There are three categories of extracted parameters, these are: shape design characteristics (lengths, radii, volumes, areas and number of features), manufacturing grades (dimensional tolerance and surface finish), and component data (family type, unique identification and material type). All manufacturing grades and component data are checked for compliance against predefined acceptable values contained within the database. In the proof of concept specific geometry to an operation is extracted when the operation time is being calculated as shown in the code in Appendix A.2. It must be noted that sub-stage 1.1 and 1.3 as described in this thesis represent a method that developed due to the use of NX. Other methods may exist, currently all extracted data is required, resulting in all state geometries being required, however if a CAM tool can be integrated into the process instead of stage 3 'manufacturing process data generation' the required data to extract would be reduced.

3.1.2 Stage 2 – Determine manufacturing process

The aim of the second stage is to determine the manufacturing process. There are two tasks in stage 2, as shown in Figure 19, which both involve the user, extracted geometry data and knowledge from the database.

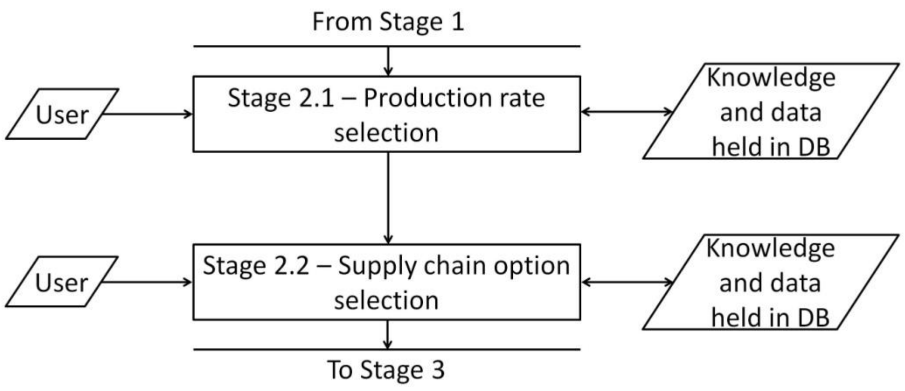

From Stage 1
User
Stage 2.1 – Production rate selection
Knowledge and data held in DB
User
Stage 2.2 – Supply chain option selection
Knowledge and data held in DB
To Stage 3

Figure 19: Framework stage 2 flow chart

In stage 2.1, production rate selection, the user is required to provide the system with a required minimum steady state production rate. This is achieved through a Graphical User Interface (GUI) coded within the integration code, shown in Figure 20, which allows the user to select previously used production rates or to add new production rates. The production rate is converted into components per minute for use within the dynamic model; see section 3.1.4 Stage 4 – Dynamic modelling for further information.

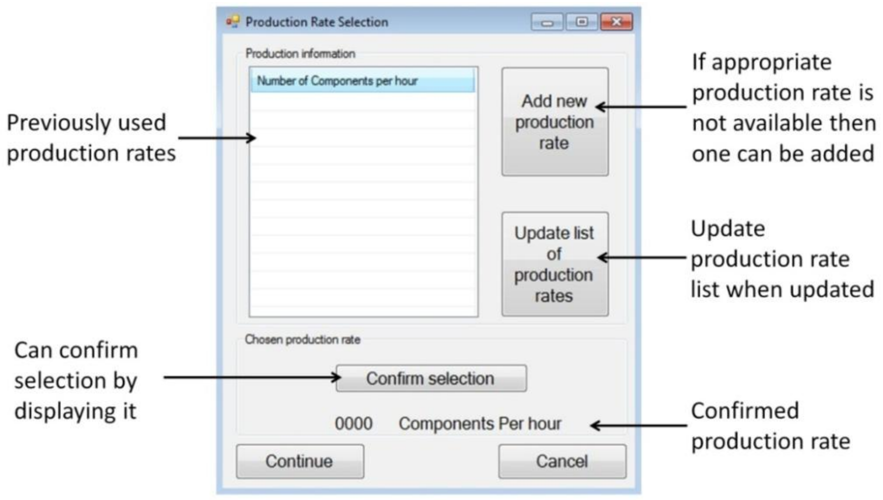

Production Rate Selection
Production information
Number of Components per hour
Add new production rate
Update list of production rates
Chosen production rate
Confirm selection
0000 Components Per hour
Continue
Cancel
Previously used production rates
If appropriate production rate is not available then one can be added
Update production rate list when updated
Can confirm selection by displaying it
Confirmed production rate

Figure 20: Production rate selection GUI

Stage 2.2, supply chain option selection, requires the user to select, from a predefined list held in the database, a supply chain option. The predefined list is populated when the component is added to the framework; see section 4.2 Case study 2 – Blisk for further information. The list is filtered based on component family type, and can be further filtered by checking primary parameters such as COS outer diameter and length against manufacturing application limitations.

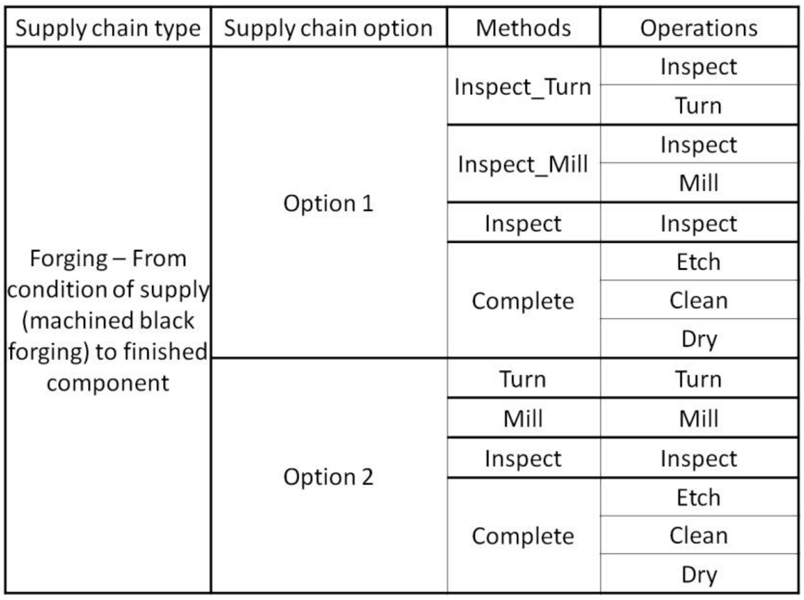
A supply chain within the system has four levels as shown by the columns in

Table 1. The first level is a supply chain type which is defined by the COS manufacturing method, in

Table 1 the example is forging. The second level is a sub section of the family type and denotes the different options within the supply chain type. In

Table 1 the supply chain option examples are option 1 and 2. Level three is a sub section of the supply chain options and splits the option up into methods, where each method represents a set up on a machine or series of operations. Level four is a sub section of the methods level and represents the individual operations that make up the methods. Each of the options produces the same finished part from the same COS, but option 1 utilises ‘Inspect_turn’ and ‘Inspect_Mill’ methods which each incorporate an extra inspection operation.

Table 1: Supply chain definition, showing four defining levels for a forging example



Supply chain type	Supply chain option	Methods	Operations
Forging – From condition of supply (machined black forging) to finished component	Option 1	Inspect_Turn	Inspect
			Turn
		Inspect_Mill	Inspect
			Mill
		Inspect	Inspect
		Complete	Etch
			Clean
			Dry
	Option 2	Turn	Turn
		Mill	Mill
		Inspect	Inspect
		Complete	Etch
			Clean
			Dry

The data is held in the database which is discussed in section 3.1.6. A GUI, shown in Figure 21, is used to select one or more supply chain options. First the user must select, from a list populated by the system based on component family type, a supply chain type. Then the user can select possible supply chain options, which are linked to the supply chain type, by sending them to the right side of the GUI. This is repeated until all options have been selected and are shown on the right side of the GUI.

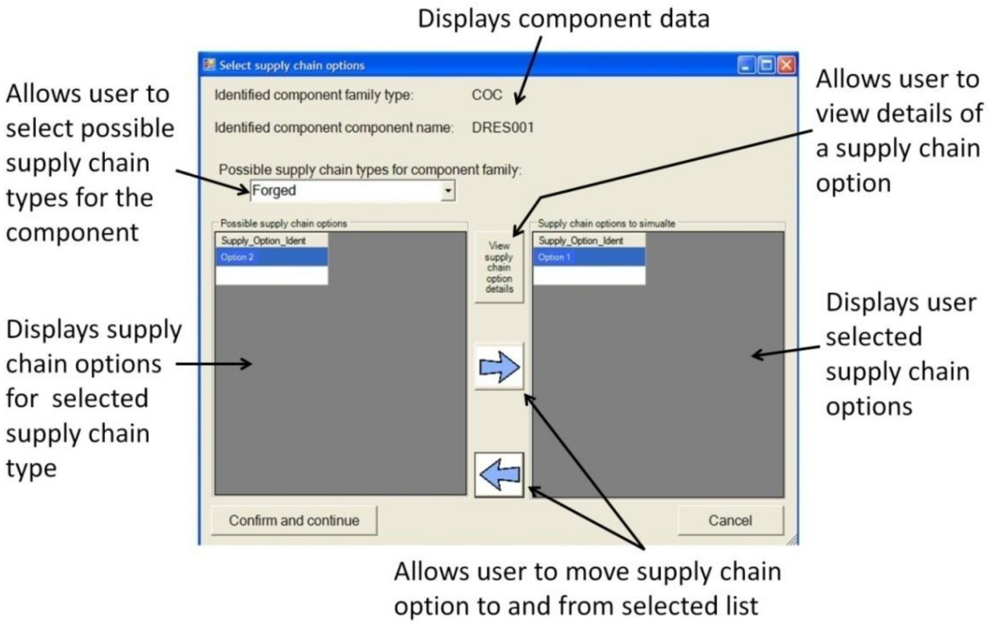

Displays component data
Allows user to select possible supply chain types for the component
Allows user to view details of a supply chain option
Displays supply chain options for selected supply chain type
Displays user selected supply chain options
Allows user to move supply chain option to and from selected list
Select supply chain options
Identified component family type: COC
Identified component component name: DRES001
Possible supply chain types for component family:
Forged
Possible supply chain options
Supply_Option_Ident
Option 2
View supply chain option details
Supply chain options to simualte
Supply_Option_Ident
Option 1
Confirm and continue
Cancel

Figure 21: Supply chain option selection GUI

Any of the supply chain options can be viewed in detail by highlighting the option and selecting the 'view supply chain option details' button. Another window, shown in Figure 22, shows the methods contained within the supply chain option to the user. Also the operations contained within individual methods can be viewed by highlighting the method and selecting the 'view method details' button.

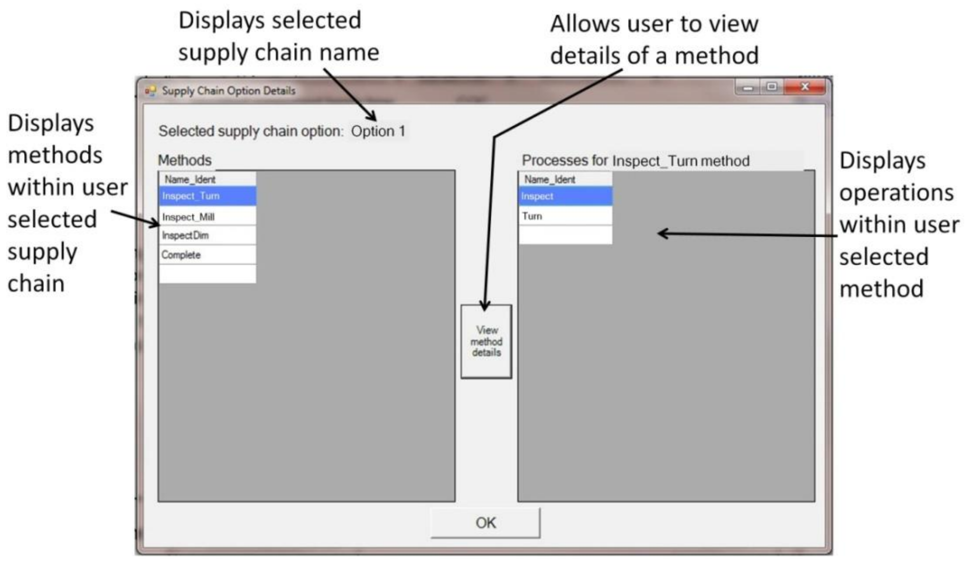

Displays selected supply chain name
Allows user to view details of a method
Displays methods within user selected supply chain
Displays operations within user selected method
Supply Chain Option Details
Selected supply chain option: Option 1
Methods
Name_Ident
Inspect_Turn
Inspect_Mill
InspectDim
Complete
Processes for Inspect_Turn method
Name_Ident
Inspect
Turn
View method details
OK

Figure 22: Supply chain option details GUI

3.1.3 Stage 3 – Manufacturing process data generation

By utilising data extracted from the geometry and user, as well as knowledge in the database and built into the integration code, stage 3 generates the manufacturing process data required for stage 4. There are four sub-stages within stage 3 as shown in Figure 23. The four sub-stages must be completed for each operation of each supply chain option selected by the user.

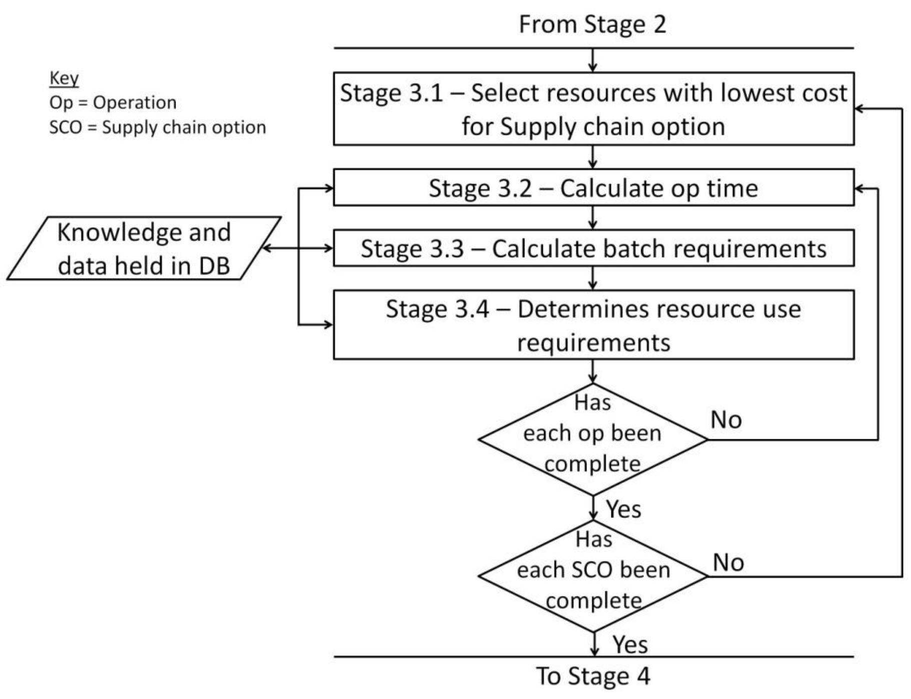

From Stage 2
Key
Op = Operation
SCO = Supply chain option
Stage 3.1 – Select resources with lowest cost for Supply chain option
Stage 3.2 – Calculate op time
Knowledge and data held in DB
Stage 3.3 – Calculate batch requirements
Stage 3.4 – Determines resource use requirements
Has each op been complete
No
Yes
Has each SCO been complete
No
Yes
To Stage 4

Figure 23: Framework stage 3 flow chart

The first sub-stage selects resources within each operation, because there is usually more than one resource that can fulfil each resource requirement. The definition of a resource in this research is given in section 2.4.1. There are three categories required for resource selection: suitability, capability and cost. Suitability represents the generic abilities for a resource to complete the operation, for example a milling machine could, in some instances, complete a turning operation but a lathe is better suited. The relation between operations and suitable resources forms part of the knowledge captured within the database described in section 3.1.6. This knowledge is captured, before the framework can be utilised, by linking one or more machines to each operation within the database. Capability represents specific capabilities of a resource to complete the operation. For example any lathe can complete a turning operation but a small lathe would not be able to turn a large component. Therefore the capabilities of individual resources are compared against the component requirements. This is completed by conducting a query on the database that compare, for example, component current diameter with lathe maximum envelope. These two categories result in a list of available resources that are suitable and capable to complete the operation. The third category, cost, is used when the first two categories result in more than one resource for a particular resource requirement. This category selects the lowest cost resource from the down selected suitable and capable list, based on data held about the resource. In the proof of concept system an overall cost rate for each resource is contained within the knowledge which is utilised to select lowest cost; see appendix A.3 Code to select and sort resources

Sub-stages 2, 3 and 4 calculate data specifically related to the supply chain operation and selected resources. Each sub-stage utilises extracted geometry data, data from the database (feeds and speeds, material properties, surface finish and dimensional tolerance manufacturing considerations and resource capabilities) and knowledge in the integration code (operation time calculations, operation batch calculations, resource use requirements).

To represent uncertainty of the setup, process and set down times of the current operation sub-stage 2 applies one of two methods. When historical data is available a probability density function is utilised, otherwise a subjective probability, such as a triangular distribution is utilised (Law and Kelton 1992; Yee Mey, Newnes et al. 2010). To do this a separate function located in the integration code which contains knowledge for a specific part of the operation generates a single time value for either: setup, process or set down. An example of a process time generation function is machine turning which is shown in Appendix A.4. Inputs into the turning process time generation function include:

- Mean diameter – Used along with surface speed to calculate the RPM
- Length of cut
- Cut type – Either: rough, medium or finishing. This affects the feed and surface speed of the cut
- Material machinability value – This depends on the material. This affects surface speed
- Tool life – This affects surface speed

The after extracting feed (by using cut type) and speed (by using material machinability, cut type and tool life) from the database the equations governing the turning process time generation function are equations 1 2 and 3:

$$RPM = \frac{Surface\ speed}{\pi * Mean\ diameter} \tag{1}$$

$$Feed\ rate = RPM * Feed \tag{2}$$

$$Cut\ time = \frac{Length}{Feed\ rate} \tag{3}$$

Each process time generation function within the proof of concept system is shown in Table 2. It can be seen in the table that each function requires a different set of inputs. Some of the functions in Table 2 are data extraction or calculation steps required by the time calculation function; drilling is an example of this. The drilling functions are an example that the system created is a proof of concept because the functions could be combined into a single function.

These time generation functions replicate one capability of a CAM tool which is to determine a time to complete an operation. As mentioned earlier the CAM tool within NX6 was unable to complete this necessary capability automatically, hence the development of the time generation functions. However any method that resulted in operation times would be an acceptable alternative.

The integration code contains knowledge that collects and calculates the inputs. When a process time has been generated a distribution is fitted to it by another function which is based on knowledge about the specific operation held in the database.

The third sub-stage is similar to sub-stage 2 but instead calculates the batch requirements for a resource. For example a heat treatment process is a batch operation, the quantity of components that can be processed by it depend on the size of the oven, the external size of the component and a packing factor. The necessary data is collected from the component or data base and the number of components that can be processed at once within the batch operation is calculated. The code for calculating the HIP vessel capacity is in Appendix A.5; the equation used within the code is equation 4.

$$HIP\ vessel\ capacity = \left\lfloor \left(\frac{Vessel\ height}{Canister\ height} \right) * Packing\ height \right\rfloor \tag{4}$$

Table 2:Table showing all process time generation functions within the proof of concept system DRES

Function type	Function name	Inputs (Data type, Input name)
Turning ops	Turning	double avgDia, double length, int cutType, int machinabilityNumber, int speedType
	Turning plunge	double avgDia, double length, string toolWidth, int machinabilityNumber
Milling	Milling	string cutterType, double toolDia, double lengthOfCut, int machinabilityNumber, int numCuts, string typeOfCut, int toolLife, double MachFactor
	Milling 2	string cutterType, double toolDia, double lengthOfCut, int machinabilityNumber, int numCuts, string typeOfCut, int toolLife, double MachFactor, int numTeeth
	Machining factor	string partRigidity, string toolRigidity, string adverseCutterForm, string surfaceCondition
Drilling	Drill traverse time	double traverseDistance
	drill type ID	string drillType
	Drill dia ID	int drillTypeID, double drillDia
	Drill dia type ID	int drillDiaID, int drillTypeID
	Drill Feed	int machinabilityNumber, int diaTypeID
	Drill speed	int machinabilityNumber, int drillTypeID, string drillManfType
	Drill number of teeth	int drillDiaID, int drillTypeID
	Drill cut time	double speed, double drillDia, double feed, int drillTeeth, double cutDepth, double leadIn
Pickel	Pickel	double depthToPickel
	Pickel run time	double depthToPickel, double pickelRate
Fill HIP canister	Fill HIP canister run time	double inputMass
	Fill HIP canister run time 2	double inputMass, double fillRate
Pressure test canister	Pressure tect canister run time	double volume
	Pressure test canister run time 2	double volume, double volumeRate
Assemble canister	Assemble canister runt time	
Weld	Weld run time	double weldLength
	Weld run time 2	double weldLength, double weldRate
Laser cut	Laser cut run time	double cutLength, string complexity, double materialThickness
HIP	HIP run time	int vesselCapacity, double componentMass, double canisterMass
	HIP setup time	int vesselCapacity
	HIP set down time	int vesselCapacity
	HIP argon mass required	int vesselCapacity, double canisterVolume, double pressureRequired, double temperatureRequired, double canisterDia, double canisterHeight
Grinding	Grinding	double area
	Grinding 2	double area, double feedAreaRate, double coverAreaMultiplier
Super finish	Super finish	double area
	Super finish 2	double area, double feedAreaRate, double coverAreaMultiplier
ECM	ECM	double length, string type

The fourth sub-stage determines when the resource will be used within the operation. For instance in Figure 24 the resource types: machine, fixture and operator type 1 are used throughout the operation. However the operator type 2 resource is only used in the setup and set down phase of the operation, therefore fewer operator type 2's are required. This is completed by assigning necessary resources to each separate operation (setup, process and set down) which is represented within the dynamic model as individual delays without queues in between them.

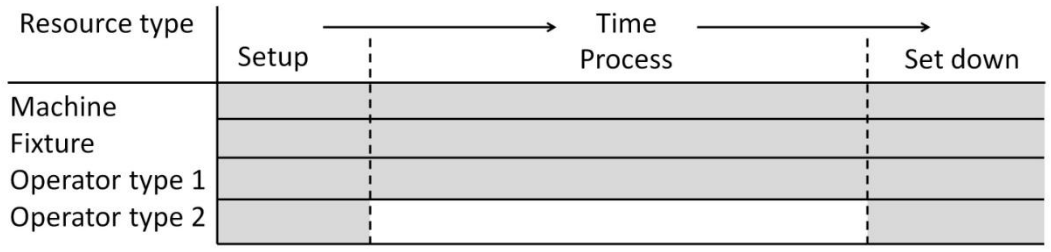


Figure 24: Resource use within an operation

3.1.4 Stage 4 – Dynamic modelling

The aim of stage 4, dynamic modelling, is to determine the optimised steady state capacity and kanban (see section 2.4.2) values required for each resource while meeting production rate and resource utilisation constraints. Figure 25 shows that there is a single sub-stage which executes the dynamic integration model for each supply chain option.

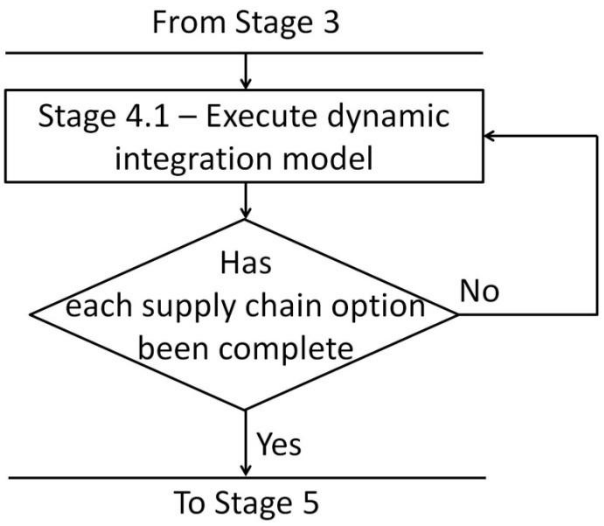

From Stage 3
Stage 4.1 – Execute dynamic integration model
Has each supply chain option been complete
No
Yes
To Stage 5

Figure 25: Framework stage 4 flow chart

A tool called Anylogic was utilised to create the dynamic integration model, however any DES software that is batch capable and linked with an optimisation tool, or has an optimiser within the tool could be used. The Anylogic tool has an internal optimiser, however the author utilised his own shown in Figure 26. The dynamic integration model was compiled into a java executable and executed via a batch file from the integration code; see Appendix A.6. A schematic of the dynamic integration model is shown in Figure 26; it contains two parts: the experiment class, which optimises the parameters; and the simulation model, which simulates the supply chain option for a particular set of parameters.

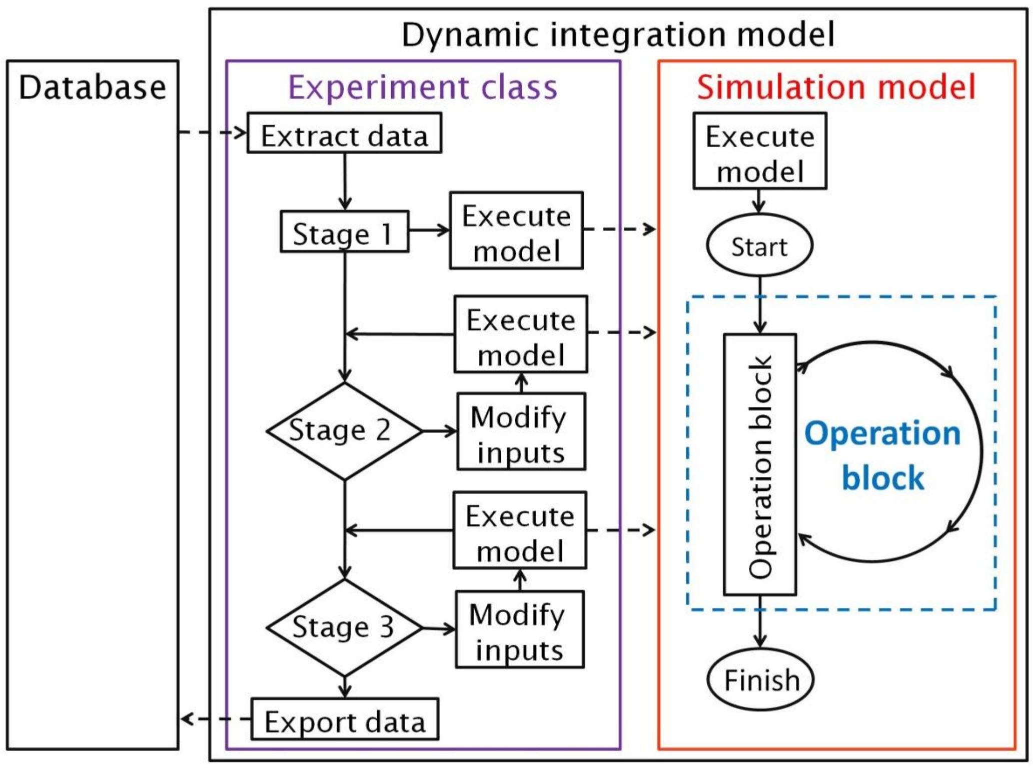


Figure 26: Schematic of the dynamic integration model

The purpose of the experiment class is to optimise the simulation model input parameters by conducting simulation experiments. Figure 27 shows a flow chart of the experiment class logic which contains three stages: initial run, reduce kanban value and determine solution.

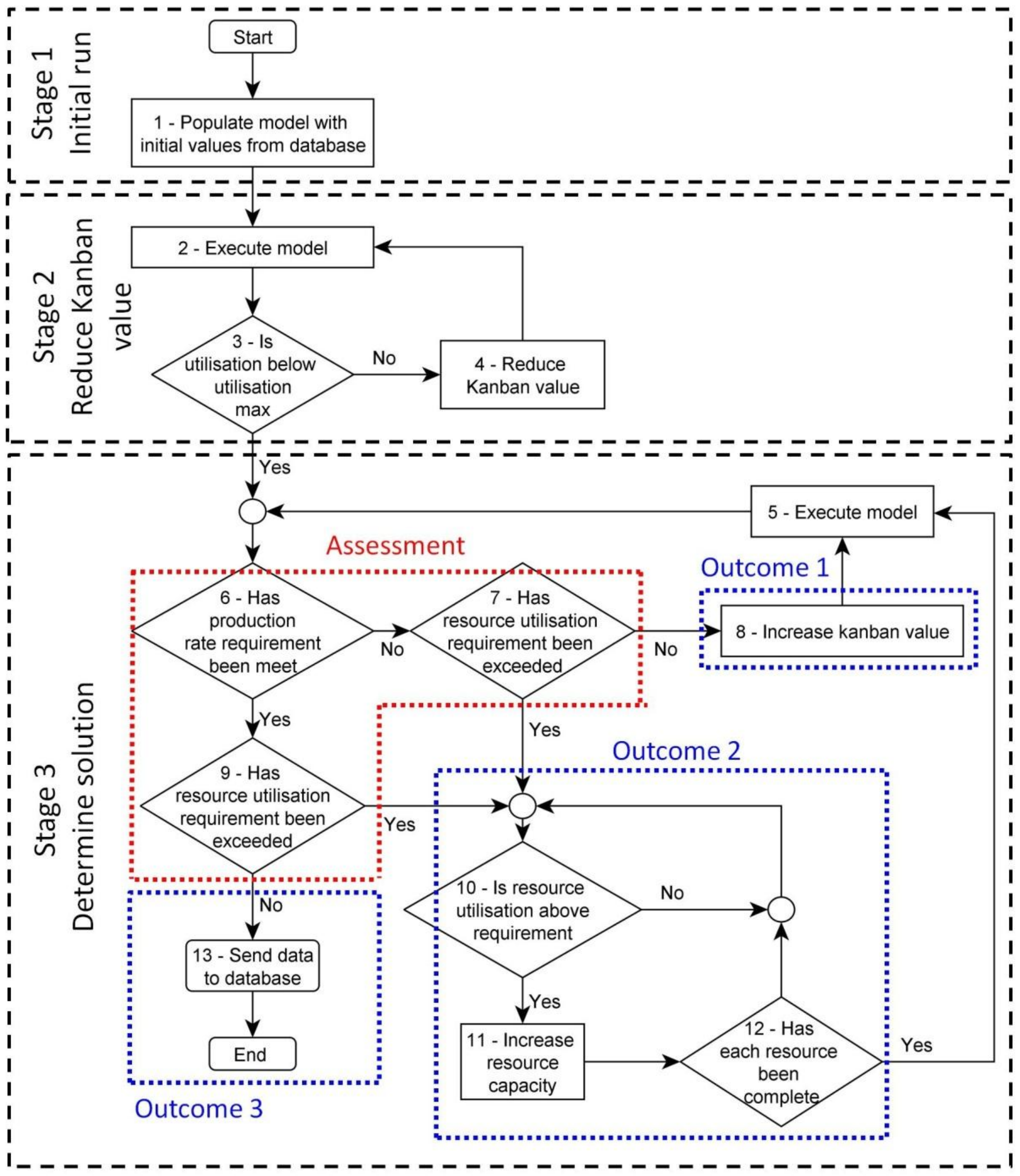

Stage 1
Initial run
Start
1 - Populate model with initial values from database
Stage 2
Reduce Kanban value
2 - Execute model
3 - Is utilisation below utilisation max
No
4 - Reduce Kanban value
Yes
Stage 3
Determine solution
5 - Execute model
Assessment
Outcome 1
6 - Has production rate requirement been meet
No
7 - Has resource utilisation requirement been exceeded
No
8 - Increase kanban value
Yes
Yes
Outcome 2
9 - Has resource utilisation requirement been exceeded
Yes
10 - Is resource utilisation above requirement
No
No
13 - Send data to database
End
Outcome 3
Yes
11 - Increase resource capacity
12 - Has each resource been complete
Yes

Figure 27: Experiment class logic

Stage 1, initial run, requires the simulation model to be executed with initial data from the database. Input parameters include:

- Minimum steady state production rate for the component
- Process plan detailing each operation of each method from COS to finished geometry
- Associated resources for each operation
- Associated with each resource:
    - Operation time with triangular distribution; if operation time is less than another resource for the same operation then a start time within the operation is required

  - Initial resource capacity shown in equation 5. See appendix A.7 for code.
  - Maximum steady state utilisation
  - If the resource is batch capable then batch size is required
- Kanban initial value shown in equation 6. See appendix A.7 for code

$$Resource\ capacity = \left\lceil \frac{\left(\frac{(Production\ rate * Op\ time)}{Reource\ utilisation}\right)}{Batch\ quantity} \right\rceil \tag{5}$$

$$Kanban\ initial\ value = \sum_{i=1}^{n} Resource\ capacity_i \tag{6}$$

Where n = number of resources in supply chain and i = the current resource.

There are two methods to achieve statistically valid output data from a simulation model (Law and Kelton 1992; Robinson 2004), either: the model needs to be executed many times (replications) with different random numbers to achieve mean output values; or a model needs to be executed with a long run time, which allows the model to reach steady state, with mean values of the output taken from the model. The long model run time method is utilised within the dynamic model to aid automated optimisation.

Production rate against simulation time for case study one with production rates of six components an hour and half a component an hour are shown in Figure 28 and Figure 29 respectively. Figure 28 represents the best case because it shows a higher production rate which achieves steady state in a shorter period of time, whereas Figure 29 represents the worst case because it shows a lower production rate which takes longer to achieve steady state. These figures show that 1.5 years is a suitable duration for model run time because both reach steady state in that time duration. The Figures also show that 0.25 years is a suitable duration for initialisation bias, described as the warm-up period by Law and Kelton (1992). The warm-up period allows the simulation model to achieve steady-state behaviour before results are collected. When model execution has completed mean utilisation of resources and production rate results are passed to the experiment class. The results from stage 1 are representative of the initial supply chain setup.

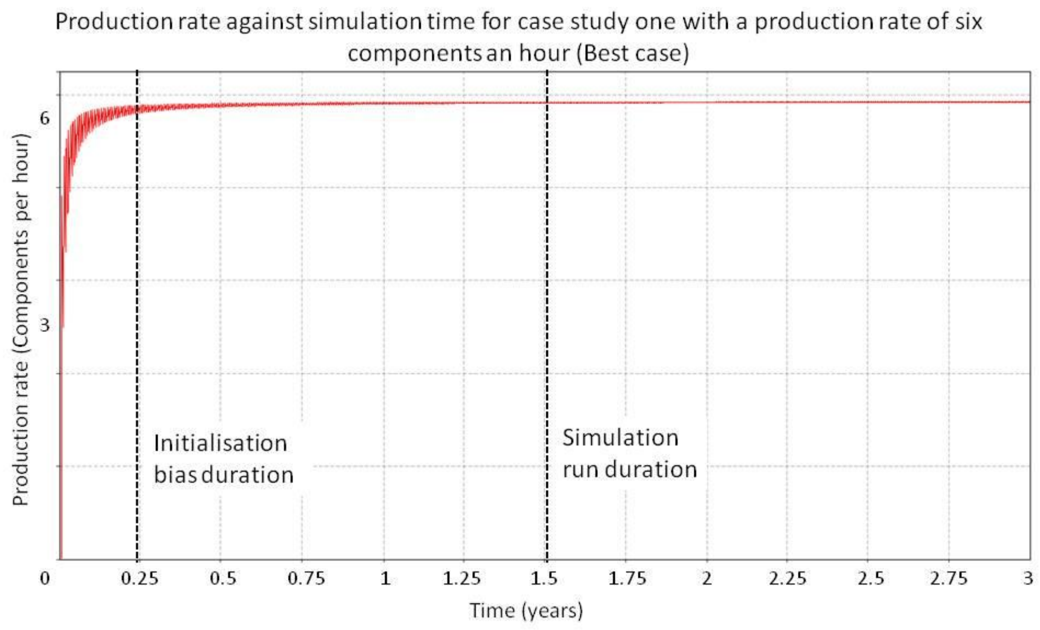


Figure 28: Production rate against simulation time for case study one with a production rate of six components an hour

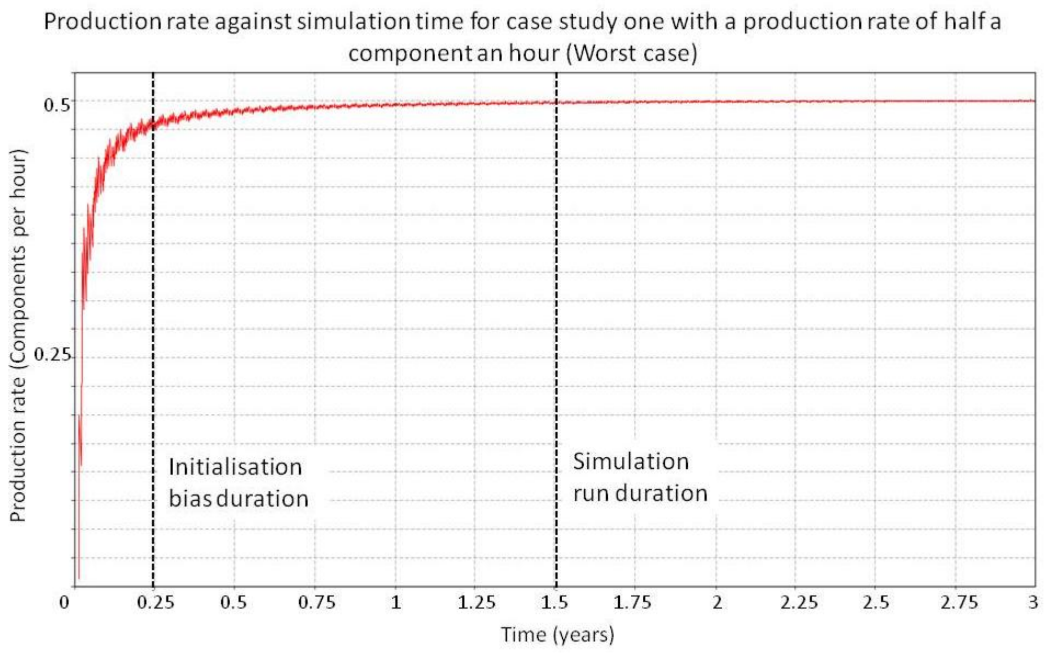


Figure 29: Production rate against simulation time for case study one with a production rate of half a component an hour

For a specific set of resource capacities increasing the kanban value increases the production rate and resource utilisation. Therefore a balance is required between meeting production rate and resource utilisation. These values are controlled through the kanban value and resource capacities. Stage 2 reduces the kanban value so that the value of each resource utilisation is below their required maximum.

Stage 3, determine solution, modifies kanban and resource capacity values individually to meet production rate and resource utilisation requirements. To do this the simulation output values are assessed. There are three outcomes of the assessment, as shown in Figure 27, these are:

- Outcome 1, increase kanban value - If both production rate and resource utilisations have not been met then the kanban value is increased. The kanban value is updated and the simulation model is executed again.
- Outcome 2, increase resource capacities - If production rate has not been met and resource utilisation has been exceeded by one or more resources, or if production rate has been met and resource utilisation has been exceeded by one or more resources, then resource capacity values will be increased for each resource where utilisations have exceeded their requirement. The resource capacity values are updated and the simulation model is executed again.
- Outcome 3, end - If production rate and resources utilisation requirements are met the experiment class updates the database with the relevant output data from the dynamic model.

The second part of the dynamic integration model is the simulation model which is a generic data driven DES model, see section 2.2.2. An analogy for a generic data driven model is a large open plan building where the logic represents the building structure. The data represents the fittings which can remodel the building's interior allowing it to serve many purposes but are within the constraints of the building. The simulation model is able to represent a supply chain with any number of operations. An operation is either batch or non-batch capable and can have any number of resources applied to it for any period of time within the total operation time, as shown in Figure 24.

To achieve this generic ability an object oriented approach was taken. Generic objects, which can be described as template blocks, were created to represent different levels within the supply chain and to complete replicated actions. The template blocks can be seen in Figure 30 which is a screen shot of the top level of the dynamic model. Figure 31 shows a schematic of dynamic model implementation with three levels: supply chain, operation and detail operation. Also Figure 31 shows three objects that complete replicated actions: route to resource, resource release and resource allocate. These objects are arrayed to represent multiple versions of the same object, therefore allowing any number of operations to be represented by a single object. Entities, which represent components, flow through the objects. If an entity reaches the end of the object it is passed on to the next object in the array which represents the next stage of the supply chain. If the entity reaches the end of the array object it is passed up one level and on to the next arrayed object. This process continues until the entity has passed through all the objects within the model. A disadvantage of this method is that there will always be some redundant arrayed object elements held in memory. However no processing power is lost cycling through these redundant elements because only active elements are cycled through.

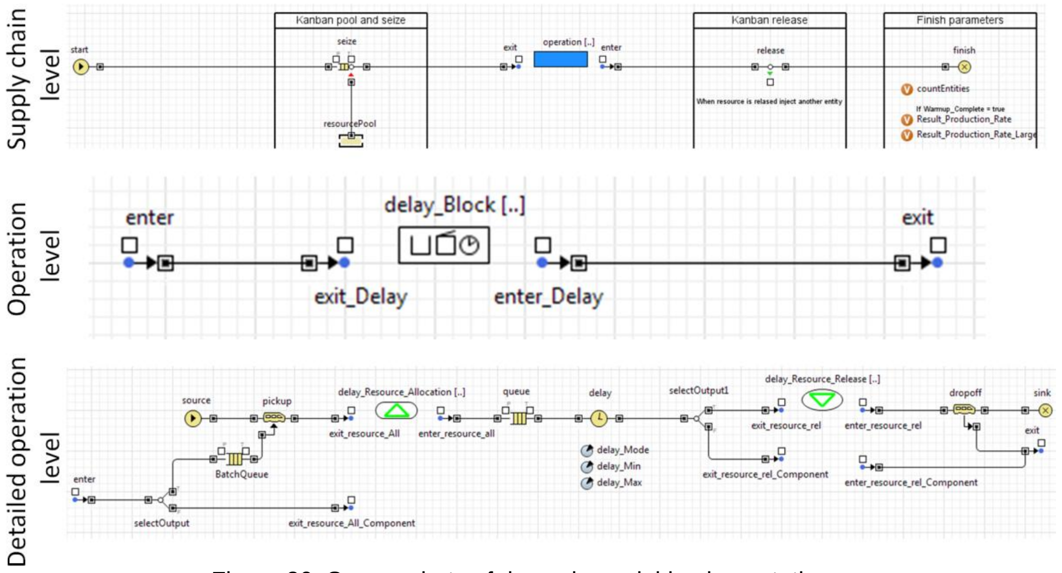


Figure 30: Screen shots of dynamic model implementation

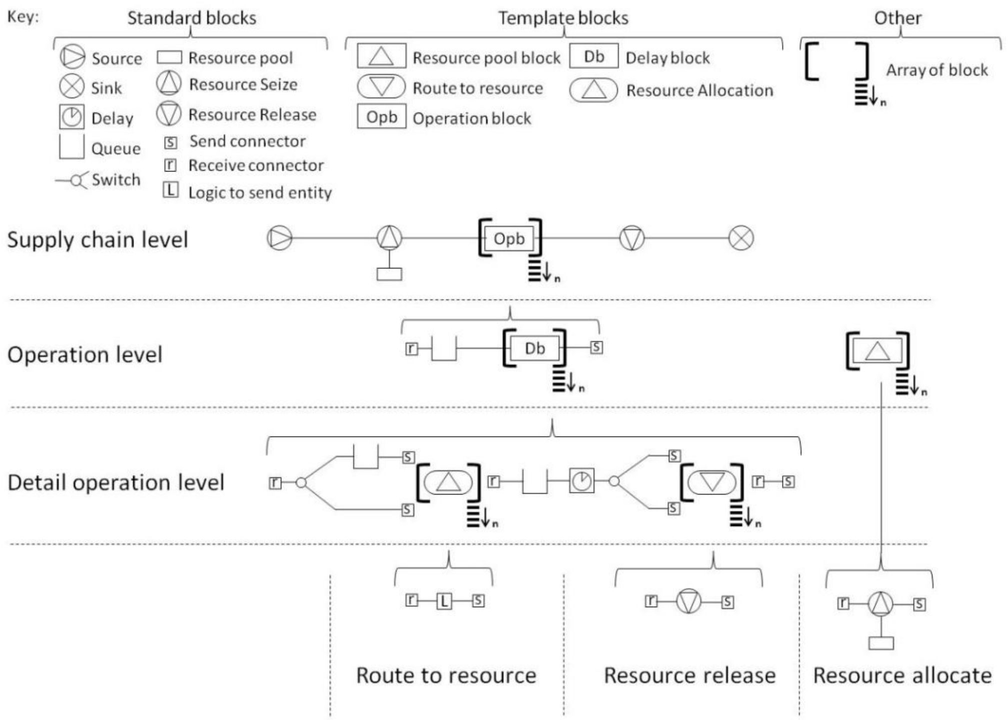


Figure 31: Schematic of dynamic model implementation

The simulation model is required to output data after execution. The data is used by the experiment class to optimise the inputs into the next iteration of the simulation model. Outputs from the simulation model include:

- Steady state production rate achieved
- Mean duration an entity stays within each operation
- Mean utilisation of each resource

Once the integration model has completed the optimisation is sends the outputs of the simulation model, plus the input values of capacities for each resource, to the database.

3.1.5 Stage 5 – Aggregated unit cost

The aim of this stage is to calculate the AUC and display it, along with manufacturing data, to the user via a GUI. The AUC is calculated utilising data from the dynamic model and cost knowledge about each resource. The flow chart in Figure 32 shows there are four sub-stages with two iteration loops within stage 5. The first sub-stage and iteration loop allows each operation - known as an activity in ABC - to be calculated. When all operations have been calculated a data collation process takes place that is used to populate sub-stage 3; the supply chain calculation. The second iteration loop repeats the previous steps for each supply chain option chosen by the user.

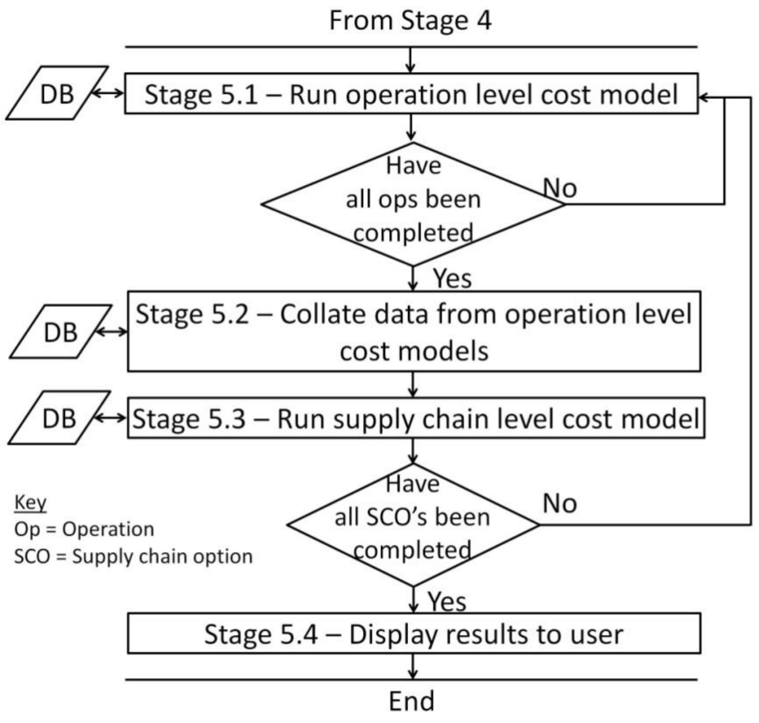
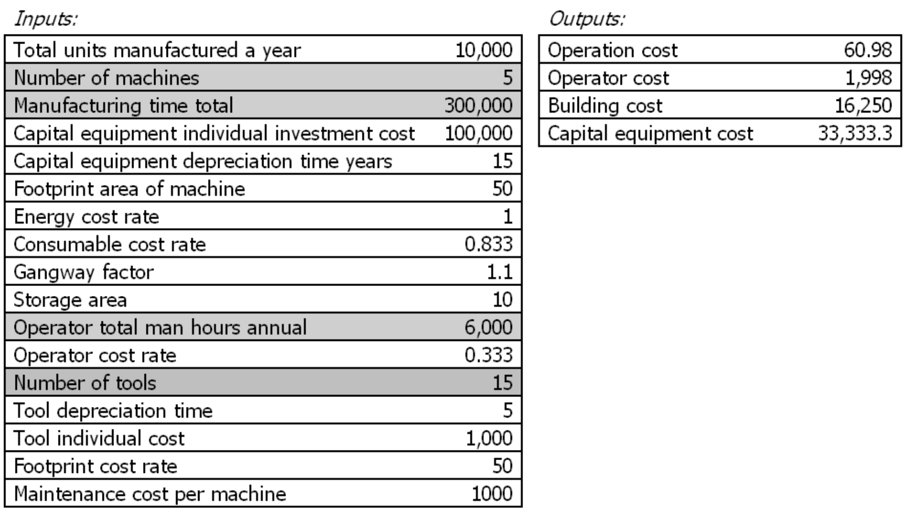

From Stage 4
DB
Stage 5.1 – Run operation level cost model
Have all ops been completed
No
Yes
DB
Stage 5.2 – Collate data from operation level cost models
DB
Stage 5.3 – Run supply chain level cost model
Key
Op = Operation
SCO = Supply chain option
Have all SCO's been completed
No
Yes
Stage 5.4 – Display results to user
End

Figure 32: Framework stage 5 flow chart

Sub-stages 5.1 and 5.3 are generic data driven ABC based cost models. The proof of concept utilises generic models built in Vanguard and accessed by the framework via web service provided by the host server. However any form of cost model which can be used in batch, therefore can receive inputs and export outputs, could be used; an example being implementation of the cost model within a function of the integration code. Sub-stage 5.1 is used to calculate the cost of an operation. Figure 33 shows the inputs and outputs of the operation level cost model. The shaded input boxes show which data comes from the dynamic model, as determined necessary to achieve the production rate, these are:

- Number of machines
- Manufacturing time total
- Operator total man hours
- Number of tools



Inputs:

Total units manufactured a year	10,000
Number of machines	5
Manufacturing time total	300,000
Capital equipment individual investment cost	100,000
Capital equipment depreciation time years	15
Footprint area of machine	50
Energy cost rate	1
Consumable cost rate	0.833
Gangway factor	1.1
Storage area	10
Operator total man hours annual	6,000
Operator cost rate	0.333
Number of tools	15
Tool depreciation time	5
Tool individual cost	1,000
Footprint cost rate	50
Maintenance cost per machine	1000

Outputs:

Operation cost	60.98
Operator cost	1,998
Building cost	16,250
Capital equipment cost	33,333.3

Figure 33: Operation level cost model – Inputs and outputs

The other inputs are gathered from the database by using the base data. The following inputs are based on the machine resource:

- Capital equipment individual investment cost
- Capital equipment depreciation time years
- Consumable cost rate
- Footprint area of machine
- Energy cost rate – cost energy used by the machine per hour
- Gangway factor – percentage of foot print area required for operators
- Storage area – required to hold tools and fixtures
- Maintenance cost per machine – per year

The remaining inputs include:

- Total units manufactured a year

- Operator cost rate
- Tool depreciation time
- Tool individual cost
- Footprint cost rate
- Scrap cost rate
- Exchange rate

The operation level cost model contains equations 7 to 17.

$$Operation\ level\ cost = \frac{Fixed\ cost + Variable\ cost}{Total\ components} \tag{7}$$

Where 'Operation cost' variables are:

$$Fixed\ cost = Building\ cost + Capital\ equipment\ cost + Maintenance\ cost\ annual \tag{8}$$

$$Variable\ cost = Labour\ cost + Equipment\ variable\ cost + Tooling\ cost + Scrap\ cost \tag{9}$$

Where 'Fixed cost' variables are:

$$Building\ cost = Footprint\ cost\ rate * Footprint\ area \tag{10}$$

$$Footprint\ area = Number\ of\ machines * * ((Footprint\ area\ of\ machine * Gangway\ factor) + (Storage\ area)) \tag{11}$$

$$Capital\ equipment\ cost = \left(\frac{Capital\ equipment\ individual\ investment\ cost}{Capital\ equipment\ depreciation\ time\ years}\right) * Number\ of\ machines \tag{12}$$

$$Maintenance\ cost\ annual: = Maintenance\ cost\ per\ machine * Number\ of\ machines \tag{13}$$

Where 'Variable cost' variables are:

$$Labour\ cost = Operator\ cost + ME\ Cost$$

$$Operator\ cost = Operator\ total\ man\ hours\ annual \\ * Operator\ cost\ rate \tag{14}$$

$$ME\ Cost = Operator\ cost * 0.15$$

$$Equipment\ variable\ cost = Consumable\ cost + Energy\ cost$$

$$Consumable\ cost \\ = Consumable\ cost\ rate \\ * Manufacturing\ time\ total \tag{15}$$

$$Energy\ cost = Energy\ cost\ rate * Manufacturing\ time\ total$$

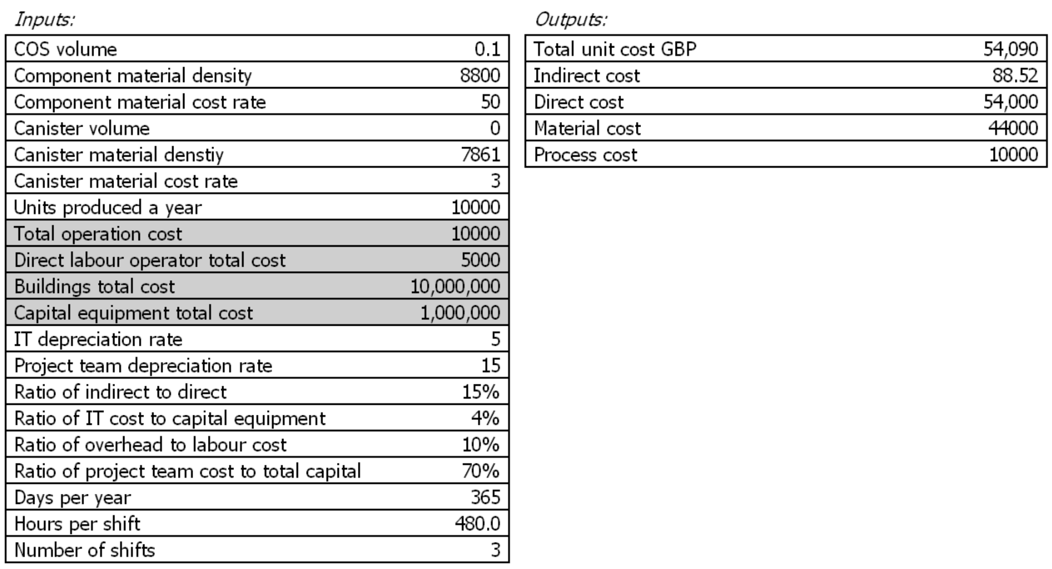
$$Tooling\ cost = \frac{(Number\ of\ tools * Tool\ individual\ cost)}{Tool\ depreciation\ time} \tag{16}$$

$$Scrap\ cost = Scrap\ cost\ rate * Manufacturing\ time\ total \tag{17}$$

The input ‘Foot print cost rate’ is a fixed value (per meter squared) in the current implementation of the framework but should be linked to worldwide location of the operation; this is discussed further in section 5.4.

It can be seen from the inputs that the operation cost model can only cope with three resources: machine, tool and an operator. This implementation of the framework was suitable for the case studies and the proof of concept, but for the framework to be fully flexible a resource cost model is required; this is discussed further in section 5.4.

In sub-stage 5.3 the output data from all the operation level models of the supply chain option, is collated. The summed outputs from each operation model become the inputs for the supply chain level cost model which is shown in Figure 34, where the shaded areas are inputs from operation level cost model. For example Equation 18 shows that the sum of each operation cost becomes the total operation cost which is the unit process cost at the supply chain level; this is implemented with a for loop.



Inputs:

COS volume	0.1
Component material density	8800
Component material cost rate	50
Canister volume	0
Canister material denstiy	7861
Canister material cost rate	3
Units produced a year	10000
Total operation cost	10000
Direct labour operator total cost	5000
Buildings total cost	10,000,000
Capital equipment total cost	1,000,000
IT depreciation rate	5
Project team depreciation rate	15
Ratio of indirect to direct	15%
Ratio of IT cost to capital equipment	4%
Ratio of overhead to labour cost	10%
Ratio of project team cost to total capital	70%
Days per year	365
Hours per shift	480.0
Number of shifts	3

Outputs:

Total unit cost GBP	54,090
Indirect cost	88.52
Direct cost	54,000
Material cost	44000
Process cost	10000

Figure 34: Supply chain level cost model – Inputs and outputs

$$Total\ operation\ cost = \sum operation\ cost_n \tag{18}$$

The supply chain level cost model contains equations 19 to 25.

$$Supply\ chain\ level\ cost = Indirect\ cost + Direct\ cost \tag{19}$$

Where 'Supply chain level cost' variables are:

$$Indirect\ cost = \frac{\begin{gathered}(Annual\ indirect\ labour\ cost\\ +Annual\ other\ overhead\ cost\\ +Annual\ amortisation\ cost)\end{gathered}}{Units\ produced\ a\ year} \tag{20}$$

$$Direct\ cost = Material\ cost + Total\ operation\ cost \tag{21}$$

Where 'Indirect cost' variables are:

$$\begin{aligned}&Annual\ indirect\ labour\ cost\\ &\qquad = Ratio\ of\ indirect\ to\ direct\\ &\qquad * Annual\ direct\ labour\ cost\end{aligned} \tag{22}$$

$$\begin{aligned}&Annual\ other\ overhead\ cost\\ &\qquad = Ratio\ of\ overhead\ to\ labour\ cost\\ &\qquad * (Annual\ direct\ labour\ cost\\ &\qquad + Annual\ indirect\ labour\ cost)\end{aligned} \tag{23}$$

$$\begin{aligned}
&\mathit{Annual\ amortisation\ cost}: \\
&\qquad = \mathit{IT\ amortisation\ costs} \\
&\qquad + \mathit{Project\ team\ amortisation\ cost}
\end{aligned}$$

$$\mathit{IT\ amortisation\ costs} := \frac{\mathit{Ratio\ of\ IT\ cost\ to\ capital\ equipment} \\ * \mathit{Capital\ equipment\ total\ cost}}{\mathit{IT\ depreciation\ rate}}$$

$$\begin{aligned}
&\mathit{Project\ team\ amortisation\ cost}: \\
&\qquad = \frac{\mathit{Project\ team\ total\ costs}}{\mathit{Project\ team\ depreciation\ rate}}
\end{aligned} \tag{24}$$

$$\begin{aligned}
&\mathit{Project\ team\ total\ costs} \\
&\qquad = \mathit{Ratio\ of\ project\ team\ cost\ to\ total\ capital} * (1 \\
&\qquad + \mathit{Ratio\ of\ project\ team\ cost\ to\ total\ capital}) \\
&\qquad * (\mathit{Buildings\ total\ cost} \\
&\qquad + \mathit{Capital\ equipment\ total\ cost} + \mathit{IT\ total\ costs})
\end{aligned}$$

Where 'Direct cost' variables are:

$$\begin{aligned}
\mathit{Material\ cost} :=\ & (\mathit{COS\ mass} * \mathit{Component\ material\ cost\ rate}) \\
& + (\mathit{Canister\ mass} * \mathit{Canister\ material\ cost\ rate})
\end{aligned} \tag{25}$$

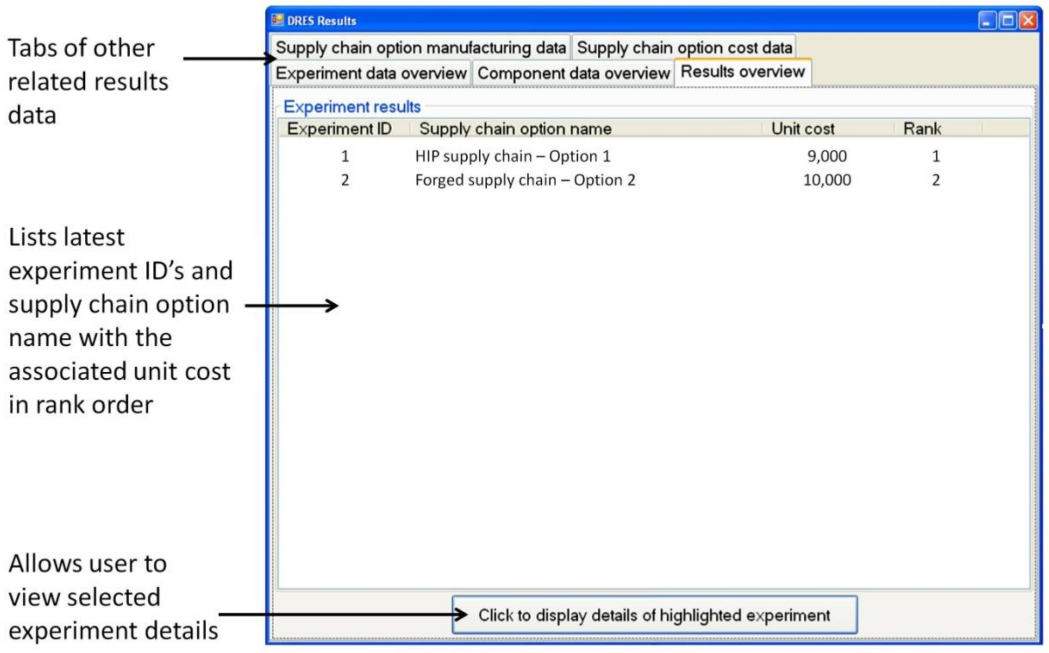
Sub-stage 5.4 displays the manufacturing and cost results to the user. The cost results are represented by outputs from each cost model. The output from the dynamic model represents manufacturing data, such as:

- Type, quantity and utilisations of machines and their resources (labour and fixtures for example)
- Number of components produced and production rate achieved
- Mean time a component spends waiting within the supply chain

Presentation of the results is completed by using a GUI, built into the integration code, with five tabs (Figure 35 to Figure 39), which are:

- Results overview
- Experiment data overview
- Component data overview
- Supply chain manufacturing data
- Supply chain option cost data

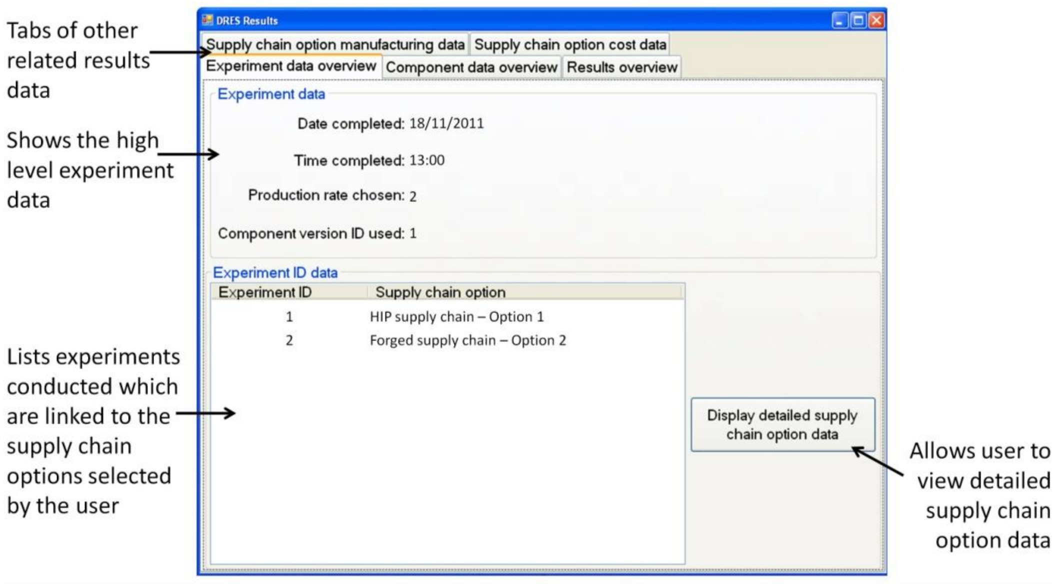
The results overview tab, shown in Figure 35, displays to the user a list of completed experiments from the last run of the system. Associated with the experiment ID is the supply chain option and calculated unit cost. The system ranks the experiments in terms of unit cost. By highlighting an experiment and selecting the button at the bottom, manufacturing and cost details for the selected experiment are loaded into the other tabs for interrogation by the user.


Tabs of other related results data
Lists latest experiment ID's and supply chain option name with the associated unit cost in rank order
Allows user to view selected experiment details
DRES Results
Supply chain option manufacturing data
Supply chain option cost data
Experiment data overview
Component data overview
Results overview
Experiment results
Click to display details of highlighted experiment

Experiment ID	Supply chain option name	Unit cost	Rank
1	HIP supply chain – Option 1	9,000	1
2	Forged supply chain – Option 2	10,000	2

Figure 35: Results form – Results overview tab

The experiment data overview tab, shown in Figure 36, contains two sections. The first, shown at the top of the form, shows the high level parameters associated to all the experiments conducted in the latest system execution. The second section, shown at the bottom of the form, displays the experiment ID and the associated supply chain option. A button allows the user to load the highlighted experiment ID into the other tabs.


Tabs of other related results data
Shows the high level experiment data
Lists experiments conducted which are linked to the supply chain options selected by the user
Allows user to view detailed supply chain option data
DRES Results
Supply chain option manufacturing data
Supply chain option cost data
Experiment data overview
Component data overview
Results overview
Display detailed supply chain option data

Experiment data

Date completed: 18/11/2011

Time completed: 13:00

Production rate chosen: 2

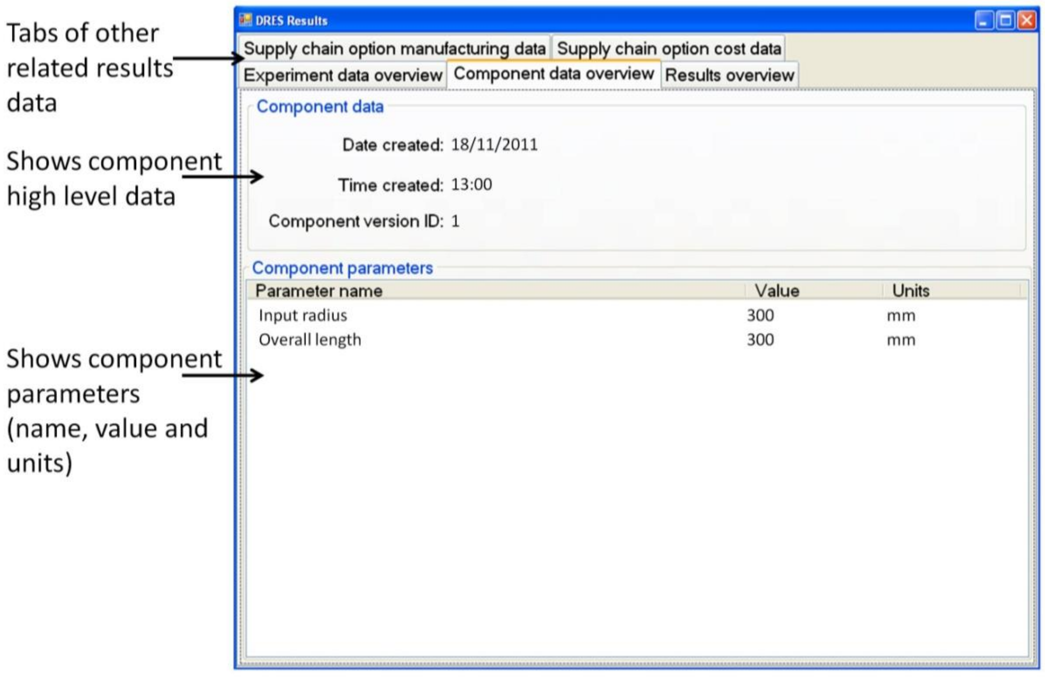
Component version ID used: 1

Experiment ID data

Experiment ID	Supply chain option
1	HIP supply chain – Option 1
2	Forged supply chain – Option 2

Figure 36: Results form – Experiment data overview tab

The component data overview table, shown in Figure 37, contains two sections with the first showing high level component data. The second lists all the component parameters by name with their associated value and unit.


Tabs of other related results data
Shows component high level data
Shows component parameters (name, value and units)
DRES Results
Supply chain option manufacturing data
Supply chain option cost data
Experiment data overview
Component data overview
Results overview

Component data

Date created: 18/11/2011

Time created: 13:00

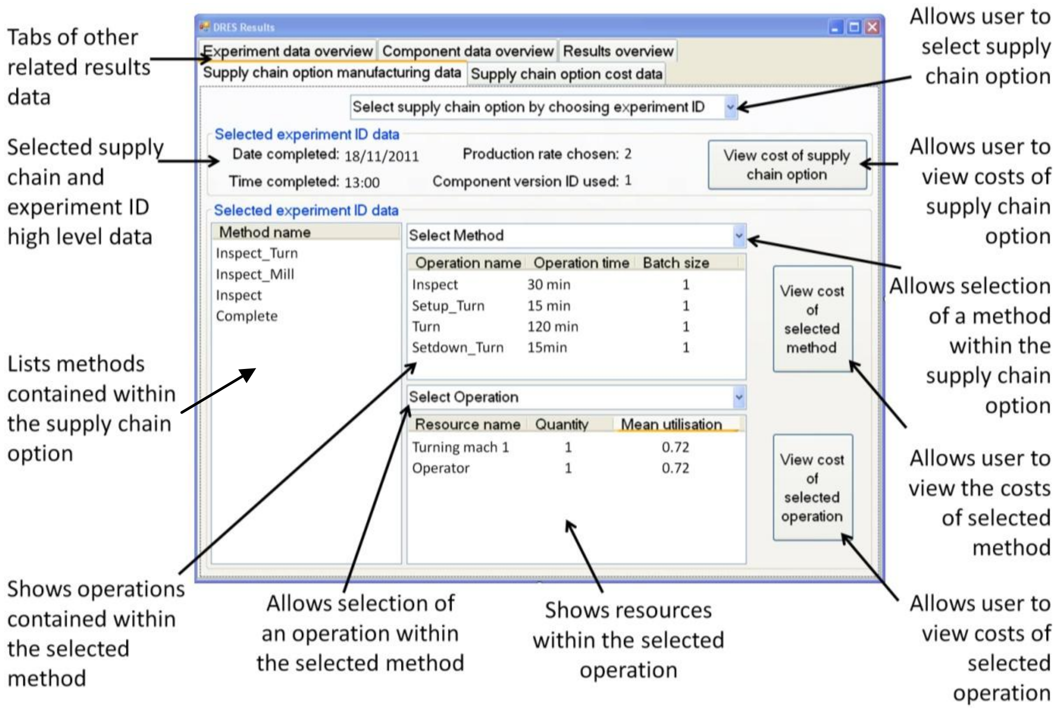
Component version ID: 1

Component parameters

Parameter name	Value	Units
Input radius	300	mm
Overall length	300	mm

Figure 37: Results form – Component data overview tab

The supply chain option manufacturing data tab, shown in Figure 38, first requires the user to select an experiment ID unless one has been loaded by pressing a button on another tab. The first section, at the top of the form, contains high level data for the selected experiment ID and a button to allow the user to view cost data for it. The second section at the bottom of the form contains three viewing areas, two drop down boxes and two buttons. The first viewing area on the left lists the methods contained within the experiment ID. The user can select a method, via the first drop down box at the top of the section, that list the operations contained within the method through the second viewing area. The list of operations contains associated operation time and, if appropriate, batch size. The user can also view cost details of the selected method by selecting a button on the right. When a method has been selected the second drop down box is populated with the operations it contains allowing the user to select one and view the resources for the operation in the third viewing area. The resource list displays: name of the resource, quantity required and mean utilisation achieved within the dynamic model. The second button allows the user to view costs for the selected operation.


DRES Results
Experiment data overview
Component data overview
Results overview
Supply chain option manufacturing data
Supply chain option cost data
Select supply chain option by choosing experiment ID
View cost of supply chain option
Select Method
View cost of selected method
Select Operation
View cost of selected operation
Tabs of other related results data
Selected supply chain and experiment ID high level data
Lists methods contained within the supply chain option
Shows operations contained within the selected method
Allows selection of an operation within the selected method
Shows resources within the selected operation
Allows user to select supply chain option
Allows user to view costs of supply chain option
Allows selection of a method within the supply chain option
Allows user to view the costs of selected method
Allows user to view costs of selected operation

Selected experiment ID data

Date completed: 18/11/2011   Production rate chosen: 2

Time completed: 13:00   Component version ID used: 1

Selected experiment ID data

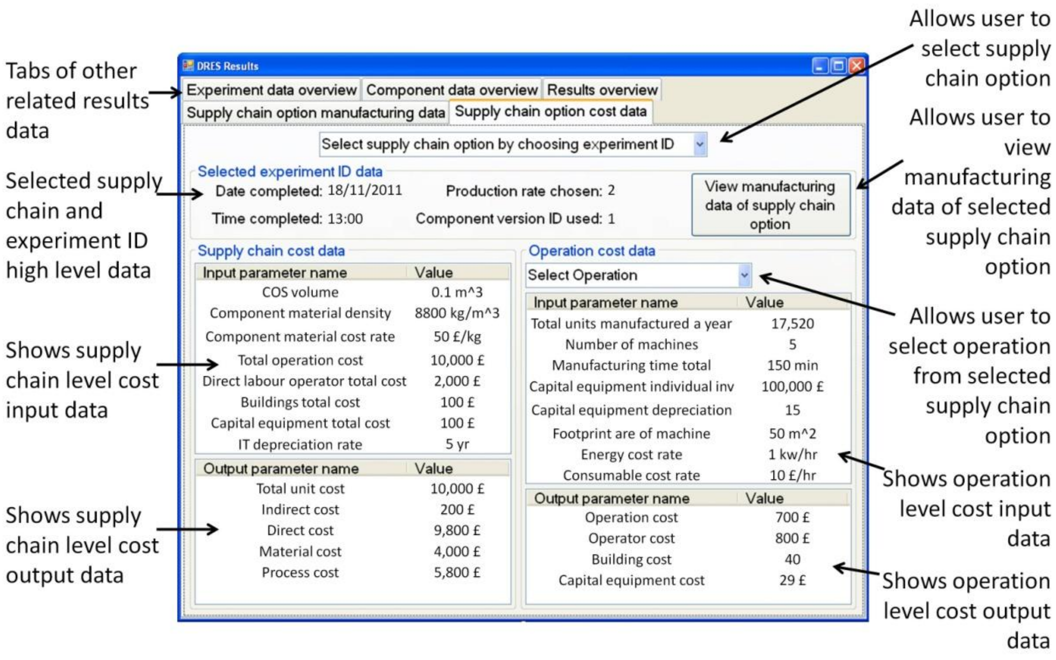
Method name
Inspect_Turn
Inspect_Mill
Inspect
Complete

Operation name	Operation time	Batch size
Inspect	30 min	1
Setup_Turn	15 min	1
Turn	120 min	1
Setdown_Turn	15min	1

Resource name	Quantity	Mean utilisation
Turning mach 1	1	0.72
Operator	1	0.72

Figure 38: Results form – Supply chain option manufacturing data tab

The supply chain option cost data tab, shown in Figure 39, requires the user to select an experiment ID, if it has not been loaded previously by pressing a button in another tab that loads the experiment ID. The first section, at the top of the form, displays high level data and a button to view manufacturing data of the selected experiment ID. The second section, bottom left of the form, displays the input and output cost parameters with associated values for the supply chain level. The third section, bottom right, requires the user to select an operation, via a drop down box, unless it has been previously populated by another tab. The input and output parameters are listed with their values in two separate areas.


Tabs of other related results data
Selected supply chain and experiment ID high level data
Shows supply chain level cost input data
Shows supply chain level cost output data
Allows user to select supply chain option
Allows user to view manufacturing data of selected supply chain option
Allows user to select operation from selected supply chain option
Shows operation level cost input data
Shows operation level cost output data
DRES Results
Experiment data overview
Component data overview
Results overview
Supply chain option manufacturing data
Supply chain option cost data
Select supply chain option by choosing experiment ID
View manufacturing data of supply chain option
Select Operation

Selected experiment ID data

Date completed: 18/11/2011 Production rate chosen: 2

Time completed: 13:00 Component version ID used: 1

Supply chain cost data

Input parameter name	Value
COS volume	0.1 m^3
Component material density	8800 kg/m^3
Component material cost rate	50 £/kg
Total operation cost	10,000 £
Direct labour operator total cost	2,000 £
Buildings total cost	100 £
Capital equipment total cost	100 £
IT depreciation rate	5 yr

Output parameter name	Value
Total unit cost	10,000 £
Indirect cost	200 £
Direct cost	9,800 £
Material cost	4,000 £
Process cost	5,800 £

Operation cost data

Input parameter name	Value
Total units manufactured a year	17,520
Number of machines	5
Manufacturing time total	150 min
Capital equipment individual inv	100,000 £
Capital equipment depreciation	15
Footprint are of machine	50 m^2
Energy cost rate	1 kw/hr
Consumable cost rate	10 £/hr

Output parameter name	Value
Operation cost	700 £
Operator cost	800 £
Building cost	40
Capital equipment cost	29 £

Figure 39: Results form – Supply chain option cost data

3.1.6 Database

A relational database was utilised due to findings in section 2.2.2 which discussed the benefit of separating the data from the models. In this proof of concept Microsoft Access has been utilised, but any database software could be used. The layout and relationships of data and tables within the database were created during the development of the framework, however there are many layouts and relationships that could be implemented to gain the same result.

The database holds data which can be classified into two categories: data that is specific to an *experiment* and base data. Experiment specific data represents data that is directly required by an experiment, such as: component geometry data, material type, operation times, production rate, supply chain options chosen, experiment setup details, and experiment results. This data is collected or calculated during execution and stored for later use. Base data represents generic data that does not change often, such as: manufacturing speeds and feeds that are associated with material machinability, possible supply chains, material data, and resource data.

There are two databases within the framework: a manufacturing database that holds base manufacturing data and a framework database that holds all the other data which is connected to the component and experiment. The two databases have been split into seven sections, shown in Figure 40 to Figure 46, which are:

- Links to experiment ID
- Component version data
- Supply chain and associated data
- Simulation data
- Cost data
- Material and machining data
- Surface finish, dimensional tolerance, laser cutting and powder HIP vessel data

The experiment table, shown in Figure 40, forms the core of the framework database where all other data tables' link to it. The experiment ID is a unique identifier contained within the experiment table that links together:

- A component version
- Simulation data
- A production rate
- Cost data
- A specific supply chain option

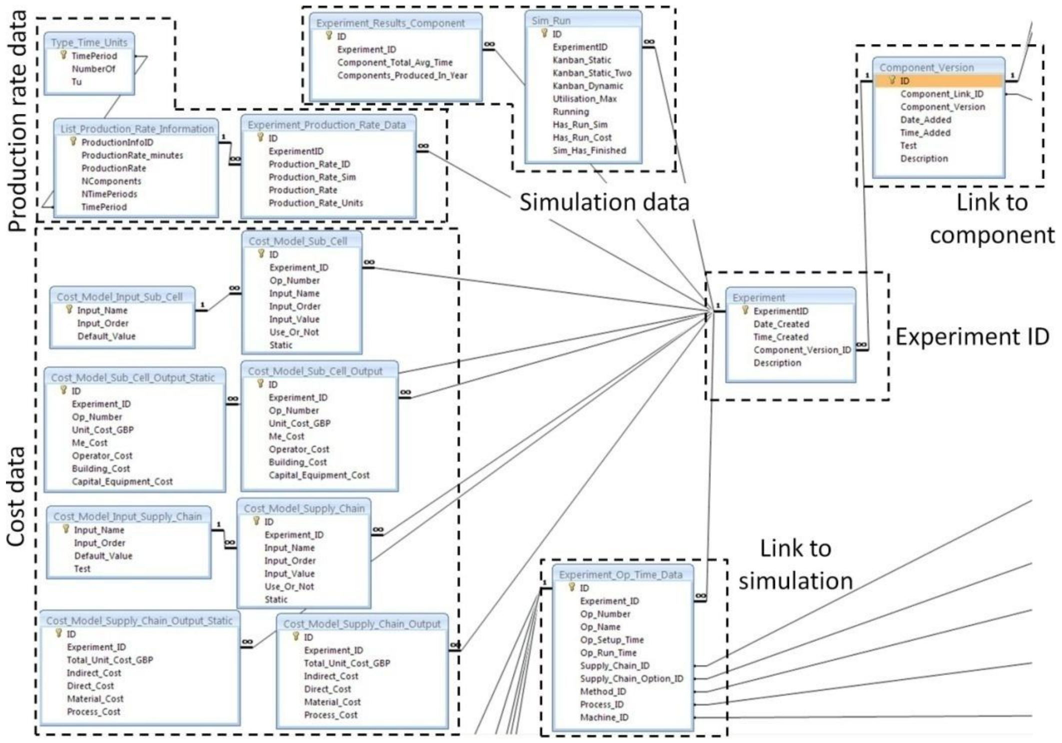


Figure 40: Framework database – Data linked to the experiment ID table

When the framework is executed it checks attributes of previously completed experiments against the new attributes, if they are different it creates a new experiment ID, otherwise it informs the user of the match. As an experiment ID is related to a specific supply chain option the check is completed for each supply chain option chosen by the user.

The component version table contains a component version ID which forms the centre of the component data, as shown in Figure 41. The component version ID is a unique identifier which links together all the component parameters and attributes. The framework performs checks to maintain the data with the first check concerning the component parameters. If a new component is required a new component version ID is created.

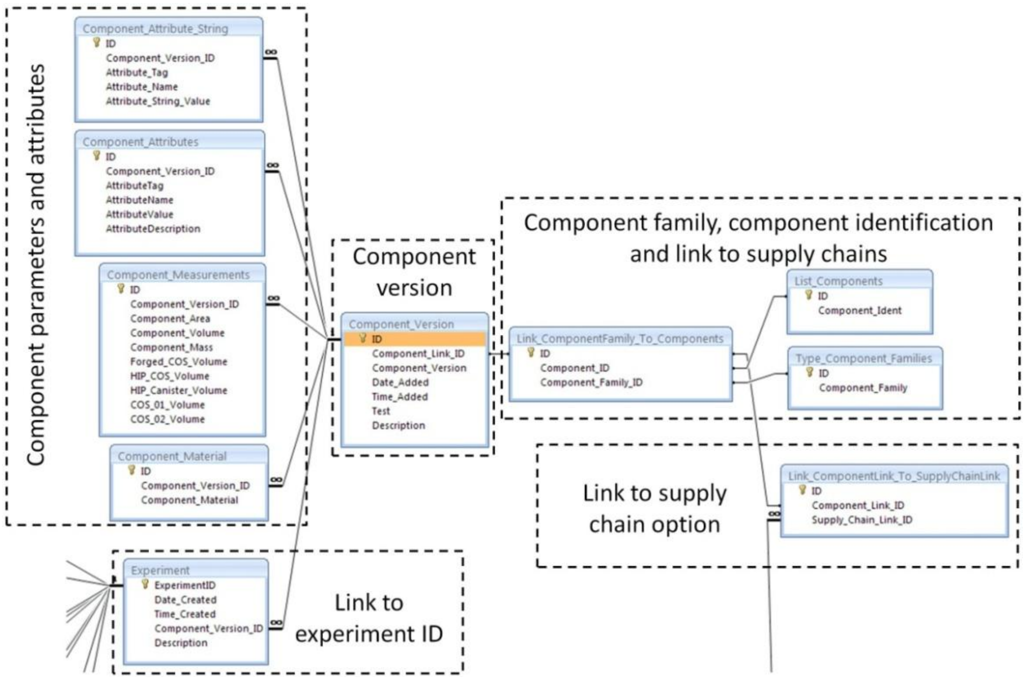


Figure 41: Framework database – Component version data tables

The supply chain data is shown in the centre of Figure 42. Four tables contain lists of supply chain types, options, methods and processes, which are linked together with four link tables. This structure allows any supply chain to be created. Machine, fixture and resource data is linked to the appropriate manufacturing operation. The supply chain options are linked to the rest of the database via the component and simulation link tables.

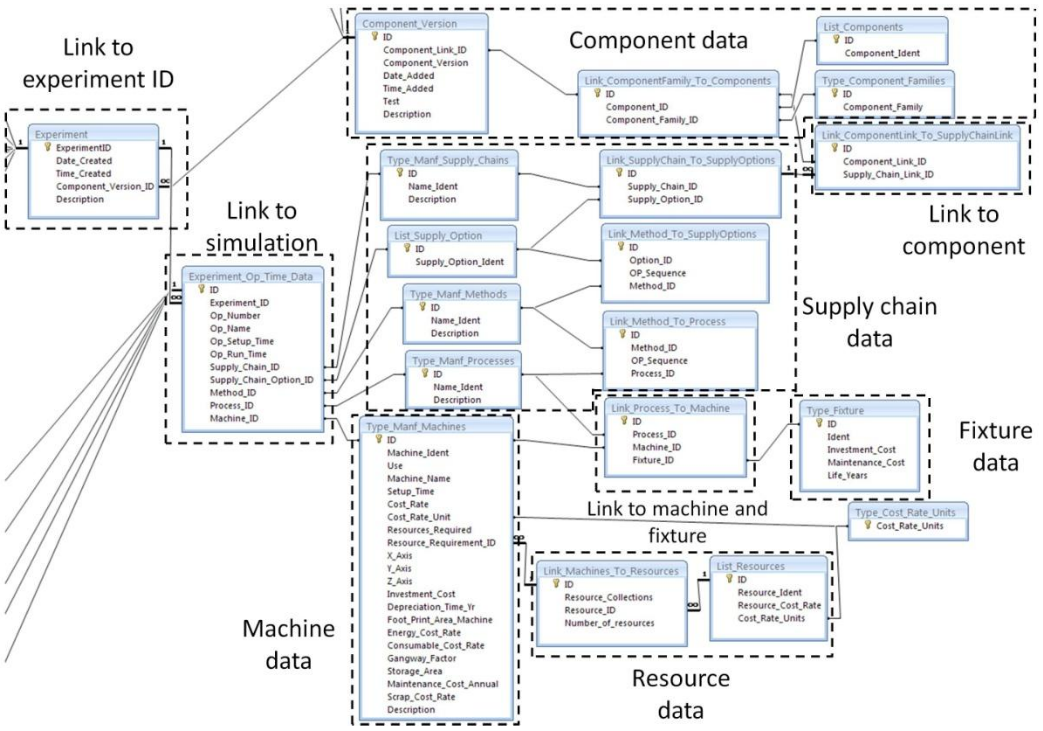


Figure 42: Framework database – Supply chain, fixture, machine and resource data tables

The simulation data tables, Figure 43, store input and output data for the dynamic model. Also the tables have the ability to store the required data for the dynamic model optimisation. For instance the ‘Sim_Resource_Allocation’ table contains columns that store the value for static and dynamic resource capacities. The static value represents the initial data and the dynamic value represents the dynamically generated data which is used to store the data during the dynamic model optimisation and results.

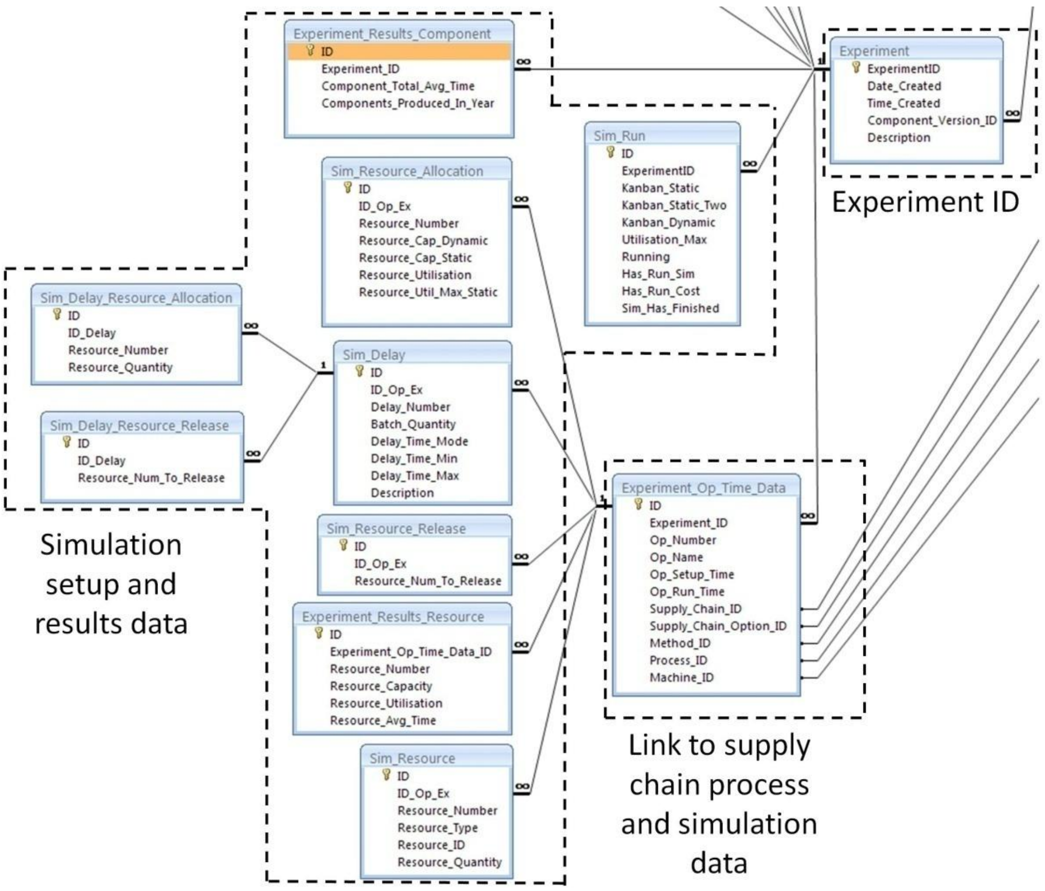


Figure 43: Framework database – Simulation data tables

The cost data tables are split into two sections, as shown in Figure 44, the operation level and the supply chain level tables. Each section contains the cost model parameters with default values, framework generated input values, and cost model calculated output values. The output values are separated into two tables: static and dynamic. The static and dynamic cost model inputs are both contained within a single table. This method of splitting the outputs into different tables was utilised for ease of results publishing.

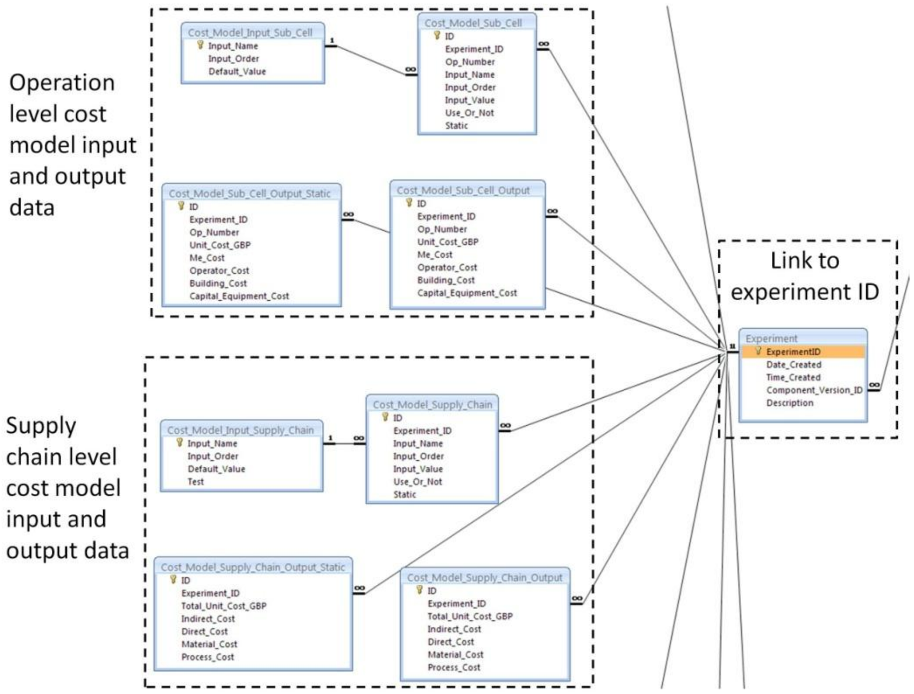

Operation level cost model input and output data
Cost_Model_Input_Sub_Cell
Cost_Model_Sub_Cell
Cost_Model_Sub_Cell_Output_Static
Cost_Model_Sub_Cell_Output
Link to experiment ID
Experiment
Supply chain level cost model input and output data
Cost_Model_Input_Supply_Chain
Cost_Model_Supply_Chain
Cost_Model_Supply_Chain_Output_Static
Cost_Model_Supply_Chain_Output

Figure 44: Framework database – Cost data tables

The majority of the data held in the manufacturing database is related to component material and machining, as shown in Figure 45. This is because the supply chains implemented within the framework all contain machining and is a significant part of the manufacturing process. There are three machining sections (turning, drilling and milling) which are linked to component material via a material machinability value. These data tables contain feed and speed data which are utilised by specific time generation functions.

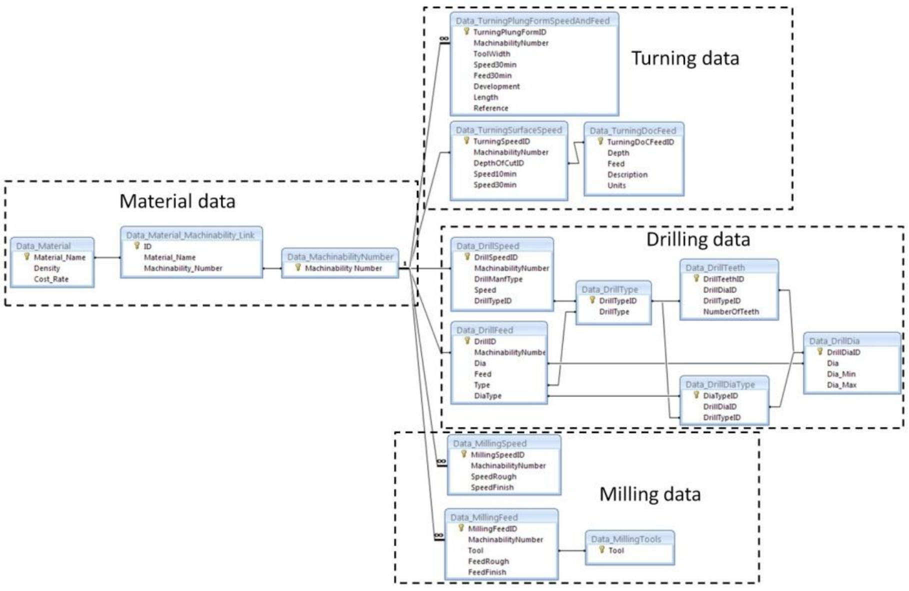


Figure 45: Manufacturing database – Material and machining data tables

The remainder of the manufacturing database contains four sections, shown in Figure 46, which are: surface finish, dimensional tolerance, laser cutting and powder HIP vessel. The surface finish and dimensional tolerance data tables supply a manufacturing method to the framework which is then used to determine a time, via a time generation function, to complete a specific operation. The laser cutting section supplies a feed rate and the powder HIP vessel section supplies vessel size data to calculate the batch size for a HIP operation based on other vessel requirements.

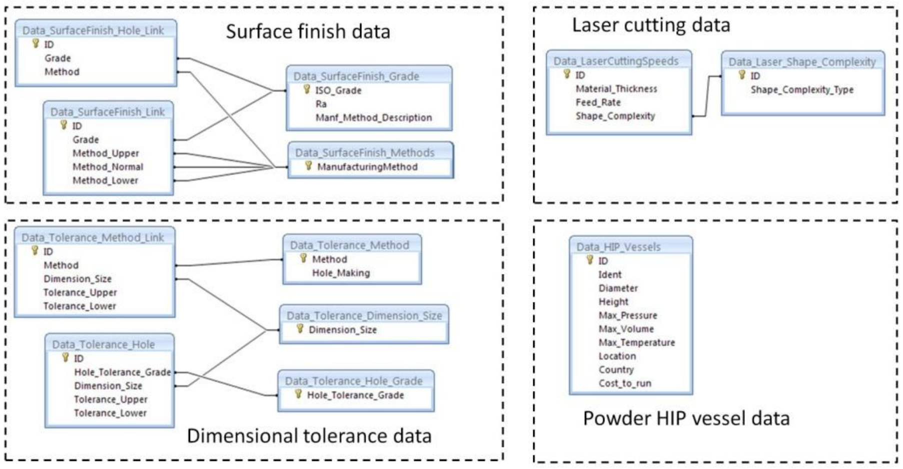


Figure 46: Manufacturing database – Surface finish, dimensional tolerance, laser cutting and powder HIP vessel data

3.2 Chapter Summary

The framework contains five stages that combine the tools necessary to integrate a dynamic modelling capability with unit cost. The first stage allows the user to modify the component geometry via a parameterised solid geometry model. The parameters from the geometry are extracted via the geometry engines API. The second stage extracts production rate and predefined supply chain option information from the user via GUIs. The third stage calculates or determines data for each supply chain operation, including: resource selection; setup, process and set-down time generation; and batch requirements. The fourth stage executes and optimises a dynamic supply chain model to generate dynamic data to be utilised in the cost model. The fifth stage calculates unit cost for each supply chain option selected by the user by utilising an ABC methodology. The last part of stage five, and of the framework, displays all results to the user via a GUI with multiple tabs.

Two databases form a significant part of the framework because they contain base manufacturing data that enables the framework to generate new manufacturing data for specific aspects of the supply chain. Also all data generated by the framework is stored within the database.

The software tools utilised within this proof of concept have been used to create a working proof of concept system. However other software could be used and the relevant characteristics required by the software to fulfil the requirements of the framework have been discussed.

Chapter 4
Case studies

Design Resource Estimation System (DRES) is a system that has been created by the author to implement the framework and conduct case studies as a proof of concept. The case studies utilise representative aerospace component geometry to show the framework can cope with complex geometry, however the unit cost results outputted by DRES are not directly validated for the following reasons:

- A representative geometry was created for the case studies because: the original geometry is confidential to Rolls-Royce; was not fully parameterised; and contains un-required attributes for Rolls-Royce internal processes.
- COS geometry is not representative. A COS is created for each supply chain type connected to the geometry by using knowledge based rules. The full utilisation of the necessary rules to create a validated COS was determined to be out of scope for this research.
- Supply chains containing methods and operations that focus on the primary manufacturing stages are utilised as a proof of concept case study. Representing the full manufacturing process for each supply chain type for both case studies was deemed outside the scope of this research.
- Each resource contained within the database has data associated with it which is required to populate and execute the dynamic supply chain model and the AUC. However this represents a time consuming data collection exercise that was out of scope of this research. Therefore when resource data was unavailable representative data was created which was based on similar resources.
- The component used within the first case study is manufactured with a production rate of hundreds of components a year compared to 4,000 to 17,000 components a year within the framework case studies. This difference in production rate results in non-similar characteristics between the component manufacturing requirements; therefore they should not be directly compared.

Unit cost values are not directly validated due to the reasons above, however by using the same base data for static and dynamic costs a comparison between the two can be made. To allow the comparison an extra cost calculation step was added to the framework. This step was placed directly before the dynamic simulation and utilises the initial base data, which is destined for the dynamic model, to calculate the static costs using the same cost models as the dynamic cost. Any difference will be due to changes in the number of resources required to achieve the production rate or because of the distributions applied throughout the framework. The percentage difference of dynamic cost data from the static cost data is utilised in the results. Equation 26 shows the how the percentage difference is calculated.

$$1 - \left(\frac{static\ cost}{dynamic\ cost}\right) * 100 \tag{26}$$

Material cost is based on static data therefore it has the same value in the static results as the dynamic results. This causes a bias towards the static results based on the percentage that the material cost makes towards the total unit cost. The overhead and process costs are unaffected by material cost bias, therefore these are used to calculate the percentage difference. Also as a form of validation against input error and anomalies within the proof of concept each experiment was completed at least two times, with the mean value of the experiments used in the results.

4.1 Case study 1 – Combustor outer case

Case study one is based on a representative large civil aerospace gas turbine combustor case, shown in Figure 47. The purpose of this case study is to show the difference between static and dynamic costs for different supply chain types, materials and component parameter changes.

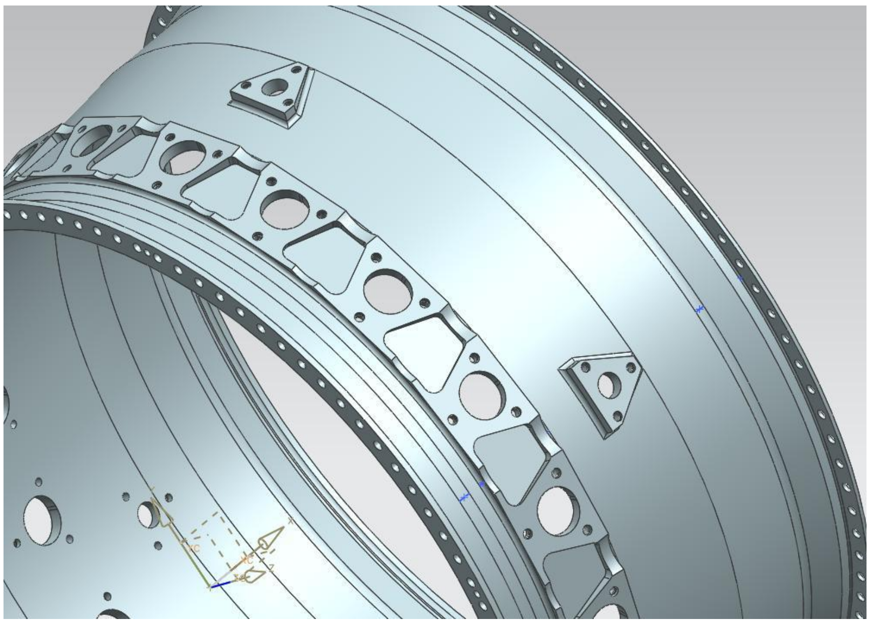


Figure 47: Case study 1 component based on representative aerospace gas turbine combustor case

For the purpose of the case study two supply chain types will be used, these are Forging and powder HIP which are shown in Figure 48. Forging represents a supply chain that machines a ring rolled forged COS to the finished component shape and then conducts finishing operations such as cleaning and dimensional inspection methods.

Loh (1992) described the powder HIP process as "the simultaneous application of iso-static pressure and elevated temperature to a work piece, which results in the work piece (usually powder) becoming consolidated”. In this case study the powder HIP supply chain type produces a near net shape COS by manufacturing a canister to hold powder material in the required shape during the HIPing process. The COS is removed from the canister by a combination of machining and pickling. The near net COS is machined to its finished shape then finishing operations, similar to those in the Forge supply chain type, are completed as shown in Figure 48. Table 3 shows the operations for both supply chains.

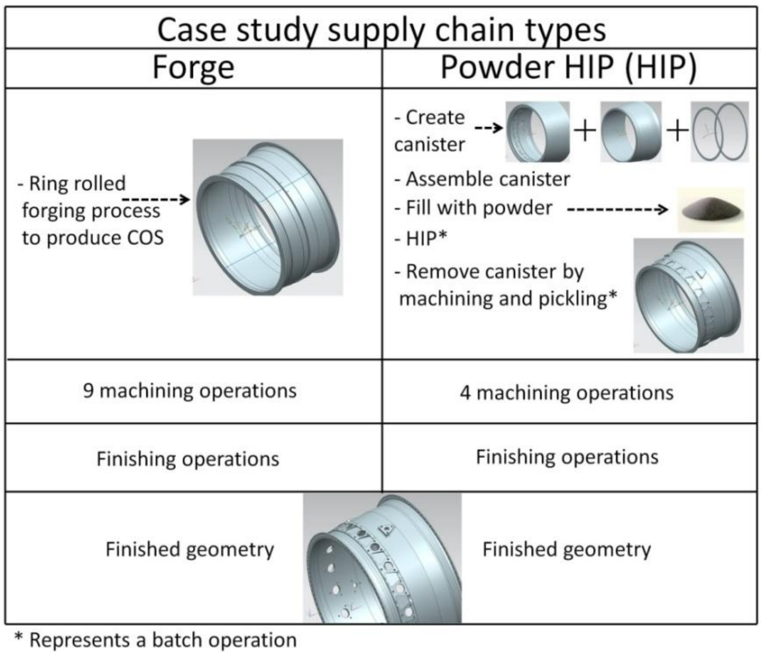


Figure 48: Case study supply chain options

Table 3: Table showing operations for both Forge and HIP supply chain types with operation times from two case study experiments

OP sequence	Supply chain option	Method	Process	Op Time
1	Forge	Turn_PF020	Turn small end	29
2		Turn_PF030	Turn large end	436
3		Turn_PF040	Turn external	225
4		Mill_PF060	Mill bosses	765
5		Turn_PF075	Turn small end	19
6		Turn_PF080	Turn lage end	250
7		Turn_PF090	Turn internal	116
8		Drill_PF095	Drill large end flange holes	43
9		DrillMill_PF100	Mill bosses and drill small end flange holes and bosses	429
10		Finishing operations	Etch	12
11			Clean	16
1	HIP	Machine canister large Plate	Laser cut	5
2			Drill holes	8
3		Machine canister small plate	Laser cut	5
4		Machine canister internal	Turn small end	64
5			Turn large end	19
6		Machine canister external	Turn small end	70
7			Turn large end	21
8			Mill external surfaces	228
9		Assemble canister	Assemble and weld	332
10		Pressure test canister	Pressure test	444
11		Fill canister	Fill with powder	113
12		HIP	HIP	447
13		Machine excess of canister	Machine excess internal	75
14			Machine excess external	444
15		Pickel canister	Pickel canister	80
16		Finish machine component	Turn internal	266
17			Turn external	43
18			Mill	20
19			Drill holes	18
20		Finishing operations	Etch	12
21			Clean	16

Three materials will be used in the case study. Two Nickel based alloys which are representative of the material used for the real component, and mild steel, which is very different and is used here to show a difference in results and system capability. The material called Nickel 1 has a higher cost rate and machinability factor than Nickel 2. The geometry contains 30 parameters but for the purpose of the case study only the parameters shown in the specific experiments will be modified and when not in use are set to a default value as shown in Table 4.

Table 4: Case study 1 – default values of changing parameters

Parameters	Default value
Internal radius	300 mm
Production rate	2 components per hour
Triangular boss large hole surface finish	N12
Triangular boss large hole dimensional tolerance	D12
Triangular boss top flat surface finish	N12
Material	Nickel 1

Five experiments have been completed within the first case study. The first two experiments vary geometrical design parameters, material type and supply chain type to mimic how a user might utilise the system. An example of this is how material selection might affect the supply chain and unit cost. The last three experiments vary supply chain parameters such as available resources, production rate and batch operation characteristics to understand how these affect the results. These parameters may not be directly changeable by the user but an understanding of their effect is required to determine under which circumstances the framework provides maximum benefit.

4.1.1 Results

The first experiment varies the internal radius, material type and supply chain type. The internal radius is varied from 275mm to 400mm which represents the minimum and maximum values outside of which other geometric parameters must be changed to maintain component geometry feasibility. Three materials and two supply chain types, discussed above, are used.

Figure 49 and Figure 50 show graphs of the percentage difference of dynamic process and overhead costs compared with static process and overhead costs respectively, against the geometric parameter internal radius for the powder HIP supply chain type and three materials.

The graph in Figure 49 shows that the process cost for the HIP supply chain type at 400mm internal radius has a percentage difference of 4.1%, 3.5% and 4.2% for Nickel 1, Nickel 2 and Mild steel respectively. As the internal radius reduces the percentage difference increases to 9.9%, 9.2% and 12.4% for the same materials respectively. The same trend is replicated for the overhead cost in Figure 50 with a mean value lower than the process cost percentage difference of 0.7%, 2.1%, 1.2% and 0.5% for 275mm, 300mm, 350mm and 400mm respectively.

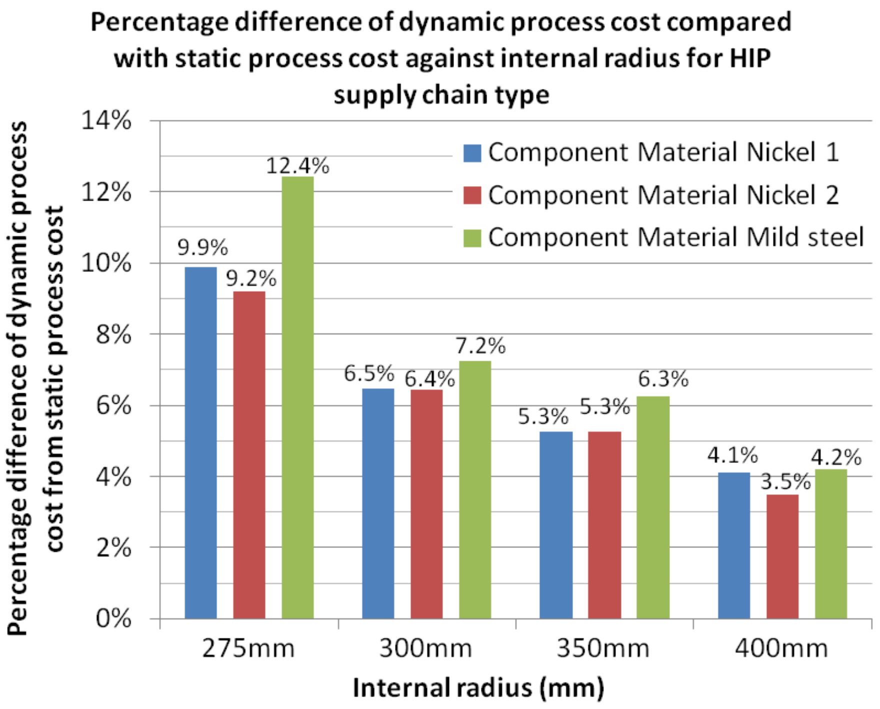


Figure 49: Graph showing percentage difference of dynamic process cost compared with static process cost against internal radius parameters for HIP supply chain type

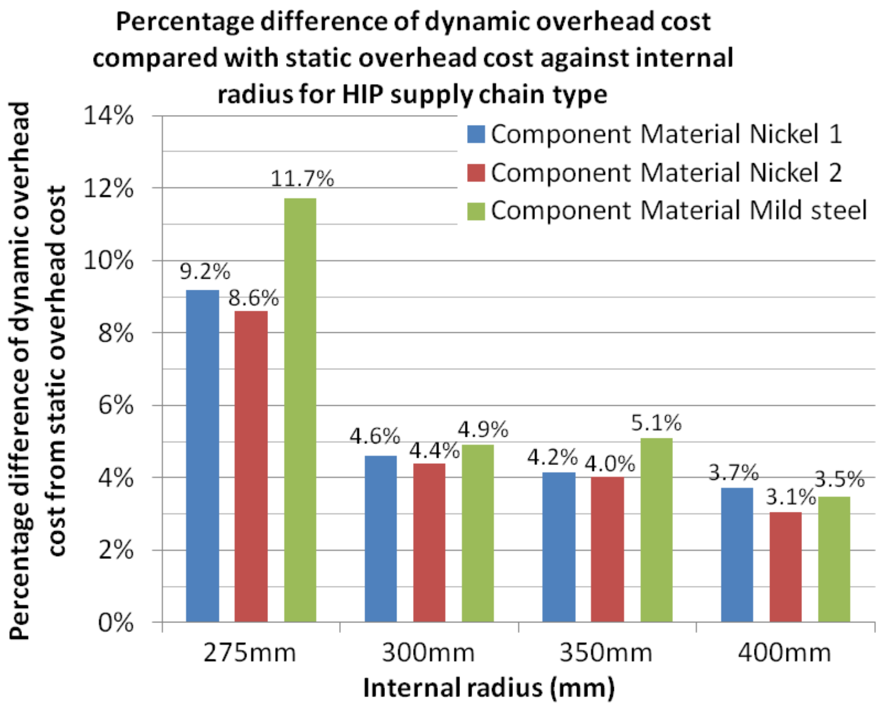


Figure 50: Graph showing percentage difference of dynamic overhead cost compared with static overhead cost against internal radius parameter for HIP supply chain type

Figure 51 and Figure 52 show the same graphs as Figure 49 and Figure 50 except they are for the forged supply chain type. The graphs show that for the forged supply chain type the difference is below 1.4% for both process and overhead costs.

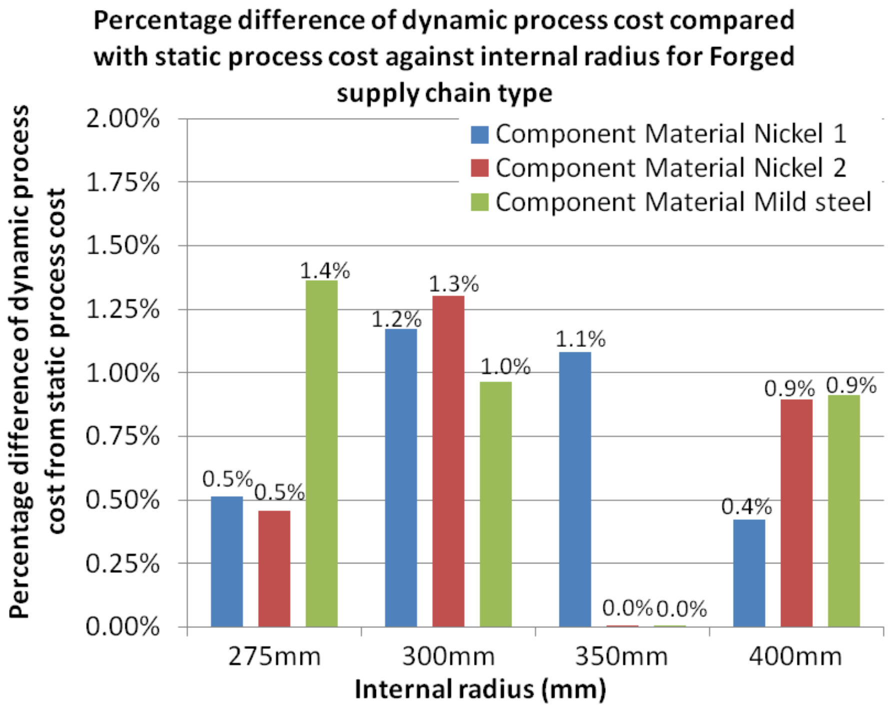


Figure 51: Graph showing percentage difference of dynamic process cost compared with static process cost against internal radius parameters for Forged supply chain type

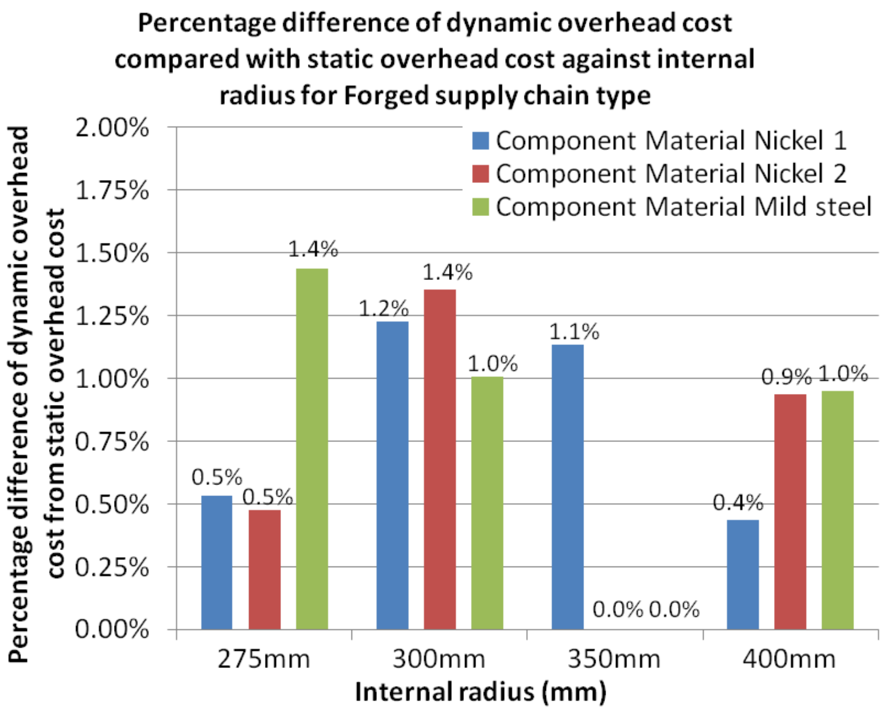


Figure 52: Graph showing percentage difference of dynamic overhead cost compared with static overhead cost against internal radius parameters for Forged supply chain type

The higher percentage difference seen for the HIP supply chain type compared to the forged supply chain type is due to two outcomes of the dynamic model. The first is the dynamic model increasing the capacity values for the batch operations within the HIP supply chain type during the resource and kanban optimisation. The second is an increase in the total manufacturing time, which is due to an increase in waiting time for resources of operations to become available, which is higher for batch operations.

The decrease in the difference of the HIP supply chain as the internal radius increases is also attributed to the reasons above. However there is a bias towards the fixed costs of the batch operation compared to the other processes which are dependent on the geometry. The batch operations are affected by component size but in this case study the change in internal radius does not affect the number of components per batch, therefore the cost for the HIP operation is fixed. However the other processes, such as machining, are dependent on the volume of raw material required to be removed which varies as internal radius changes. Therefore a bias towards the fixed batch operation costs results in a higher percentage difference for a component with smaller internal radius compared to a component with a larger internal radius.

Experiment two varies surface finish and hole diameter tolerances. Three individual parameters with two combinations of these parameters have been modelled. The three parameters are: triangular boss top flat surface finish (TB SF) which was set to grade N2, from default value N12; triangular boss large hole surface finish (TB LgHole SF) which was set to grade N7, from default value N12; and triangular boss large hole tolerance (TB LgHole Tol) which was set to grade H7, from default value D12. The two combinations were TB SF with TB LgHole SF and all three parameters together. More details on surface finish and dimensional tolerance grades are contained in Appendix B.

There is no specific equation that calculates the affect that a change in tolerance or surface finish has. Instead knowledge is and logic is utilised to select a method of manufacture based on the tables and graphs in Appendix B that can achieve the required tolerance and surface finish. The geometry data is then used by the selected method of manufacture to generate an operation time.

The graph in Figure 53 shows the percentage difference of dynamic cost compared with static costs against different surface finish and tolerance parameters associated with the triangular boss features as discussed above. The graph shows that there is between 5.3% - 6.5% difference for the process cost and between 3.1% - 4.5% difference for the overhead cost.

The graph shows there is no noticeable change in the percentage difference as more surface finish and tolerances are applied. There are two reason for this. The first is because there is little change in the operation times between the different method of manufactures required to achieve the tolerance and surface finishes. The second is that any changes are insignificant compared to the total cost and therefore do not considerable affect the percentage difference.

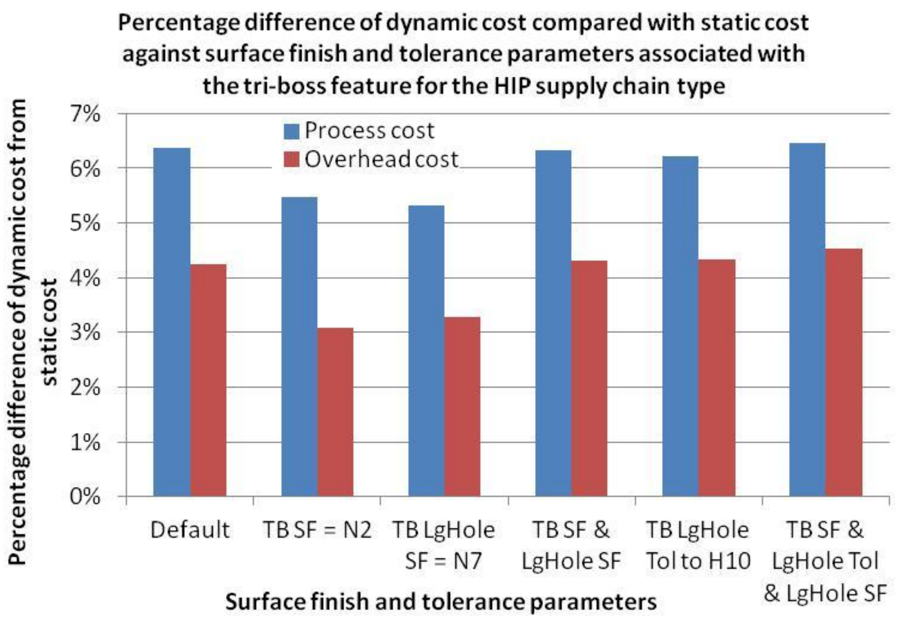


Figure 53: Graph showing percentage difference of dynamic cost compared with static cost against surface finish and tolerance parameters associated to the tri-boss features for the HIP supply chain type

Experiment three shows the effect of having different available resources for the system to select when varying the internal radius. The graphs in Figure 54 and Figure 55 show how available resources affect the difference between static and dynamic costs. In this example the experiments represented by the red bars only had access to a single large diameter HIP vessel which is the default vessel used in all the other experiments and has a diameter of 1.1m.

It is explained in experiment 1 why the percentage difference is larger when the component is smaller and reduces with increasing internal radius which is also seen here. The experiments represented by the blue bars had access to three different diameter sized HIP vessels, but with the same vessel length, that had running and investment costs relative to their size as shown in Table 5. The system selects an appropriate vessel for the component size; therefore the small vessel is chosen for the 275mm internal radius, the medium vessel for the 300mm internal radius, and the large vessel for the 400mm radius. It must be noted that the number of components each vessel can hold is the same for all the vessels, as this is based on the vessel length which is the same for each vessel.

Table 5: HIP vessels used in experiment 3

Machine Name	Setup Time	X Axis	Y Axis	Z Axis	Investment Cost	Depreciation Time Yr	Foot Print Area	Energy Cost Rate	Consumable Cost Rate	Gangway Factor
HIP vessel small	60	0.8	0.8	3	5000000	15	200	80	80	1.1
HIP vessel medium	75	0.9	0.9	3	7500000	15	200	90	90	1.1
HIP vessel Large	90	1.1	1.1	3	10000000	15	200	100	100	1.1

It can be seen in Figure 54 that a weighting towards the batch operations for smaller components is still present with the same trend as previous experiments (percentage difference decreases as internal radius increases), but less significant because the capital costs of the smaller vessels are lower than the larger vessel. However, in Figure 55 the overhead cost difference, of the three vessel study, increases with increasing internal radius instead of decreases as with a single vessel. This is because the overhead cost is based on capital cost which is less for the smaller vessel.

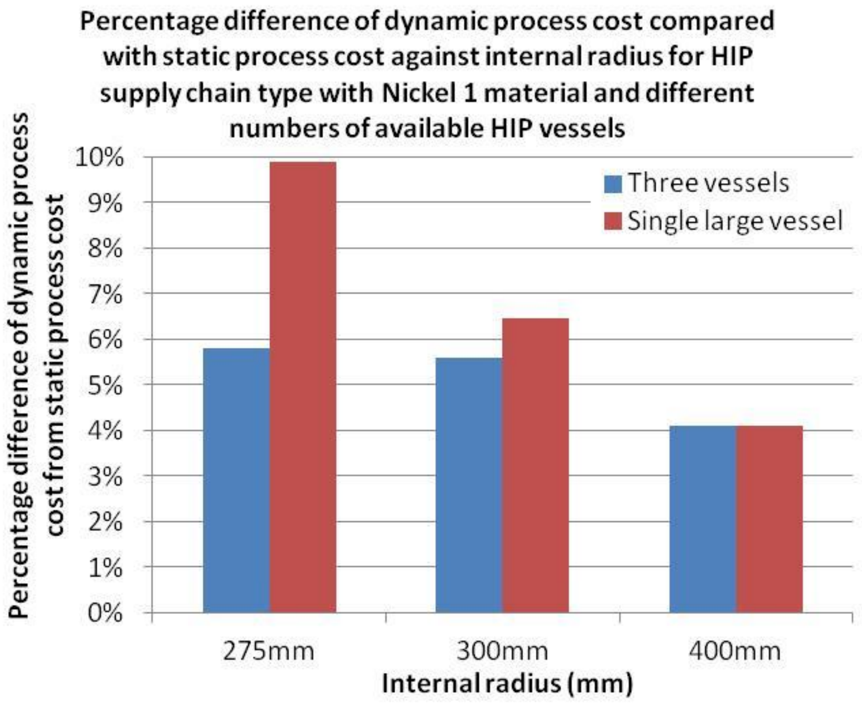


Figure 54: Graph showing percentage difference of dynamic process cost compared with static process cost against internal radius for the HIP supply chain type with Nickel 1 as the material and different numbers of available HIP vessels

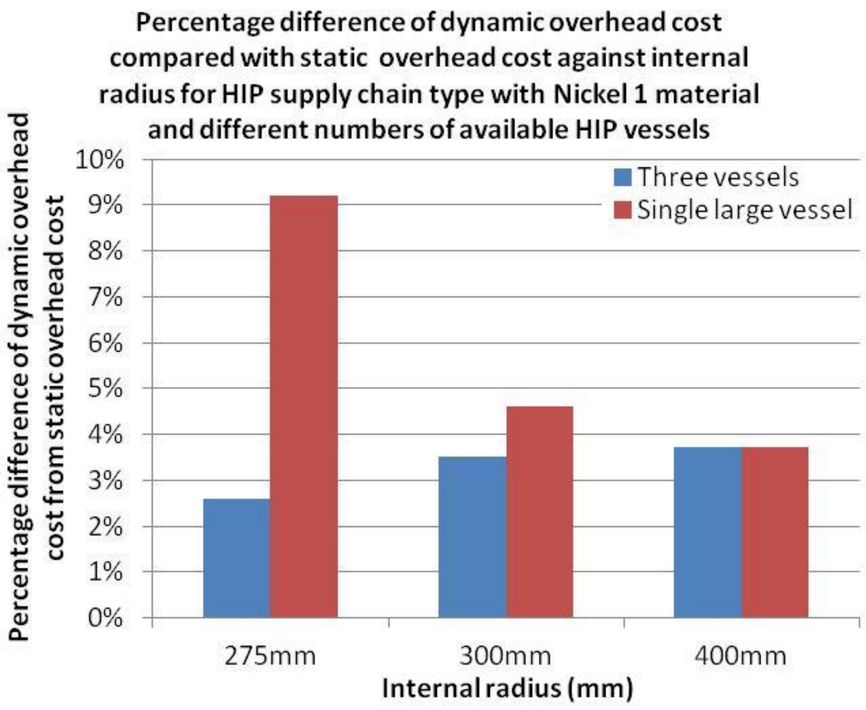


Figure 55: Graph showing percentage difference of dynamic overhead cost compared with static overhead cost against internal radius for the HIP supply chain type with Nickel 1 as the material and different numbers of available HIP vessels

Experiment four shows the effect of modifying production rate for both supply chain types. The graphs in Figure 56 and Figure 57 show the percentage difference of dynamic costs compared with respective static costs against production rate for both supply chain types for a fixed internal radius of 300mm and material type of Nickel 1. It can be seen that the percentage difference for the forged supply chain type is stable to within 1.2% for both process and overhead costs, which is expected due to there being no batch operations within the supply chain.

However for the HIP supply chain type the process cost percentage difference decays following an exponential curve from 79.7% to 1.3% as production rate increases from 365 to 52,560 components a year. The reason for this is that as production rate increases the utilisations of individual resources achieve their maximum value. Therefore the cost of the resource is spread amongst the optimum number of components, reducing unit process cost.

For the overhead cost in Figure 57 the overhead cost difference is seen to be between 0% and 3.75% up to 2950 components a year. At 4380 components per year it jumps to 11.5% then reduces following a non-linear decay curve to 1.4% at 52560 components per year. The decay curve is a result of overhead cost being a percentage of capital cost divided by the number of components per year. For example if a new resource is required to meet production rate requirements the overhead cost will increase, falling as the resource meets its maximum utilisation. However this effect is reduced as production rate increases because the addition of a single resource becomes insignificant compared to the total capital cost.

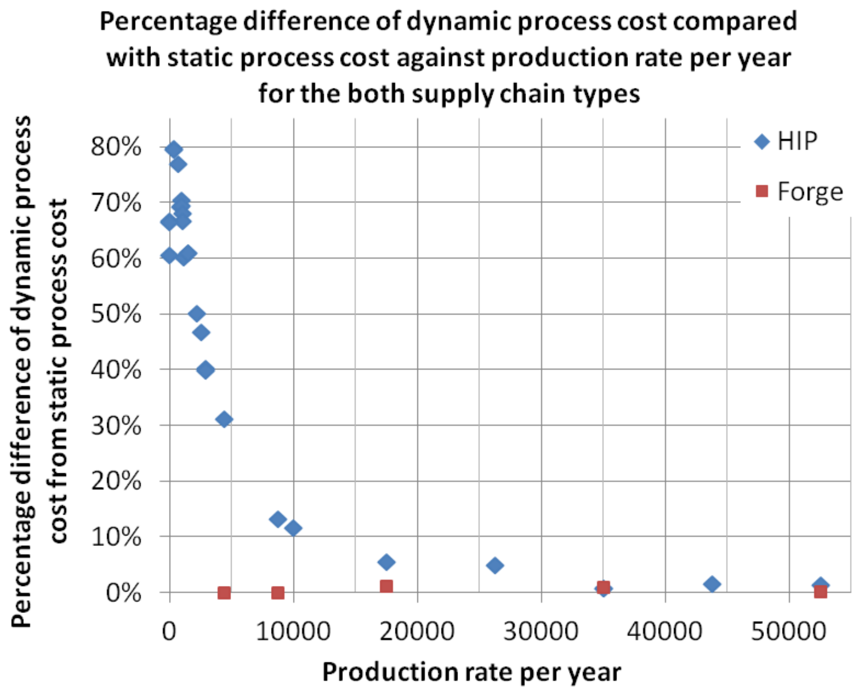


Figure 56: Graph showing percentage difference of dynamic process cost compared with static process cost against production rate changes for HIP and forged supply chain types

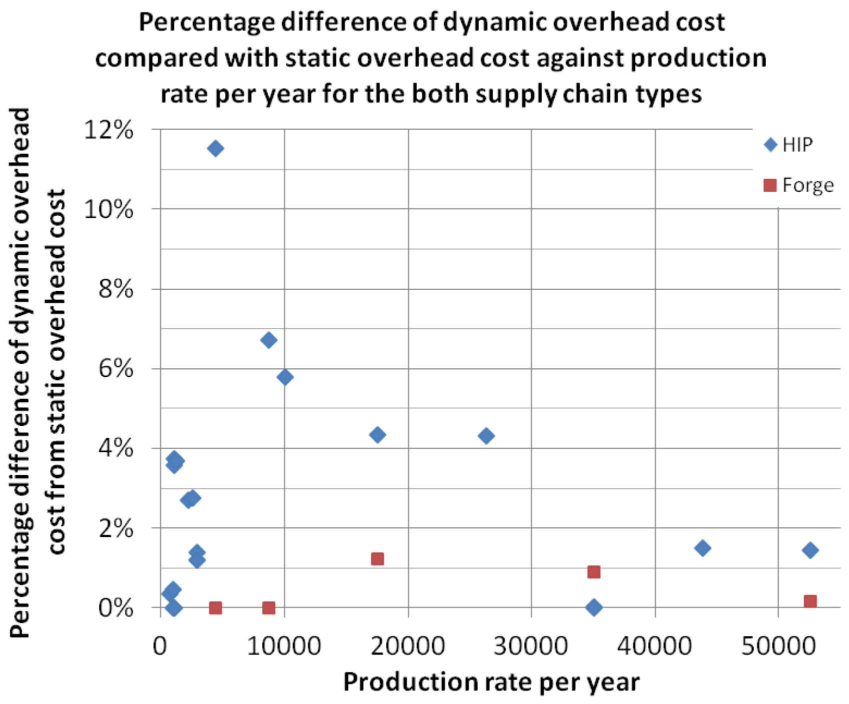


Figure 57: Graph showing percentage difference of dynamic overhead cost compared with static overhead cost against production rate per year for the HIP and forged supply chain types

Analysing the HIP supply chain type experiment data further produced two graphs, Figure 58 and Figure 59. Figure 58 shows normalised mean utilisation of all the resources within the supply chain against the production rate per year. The graph is an exponential curve up to 0.7, which is logical because the dynamic model optimises the number of resources so that the individual resource utilisations do not go above 0.7.

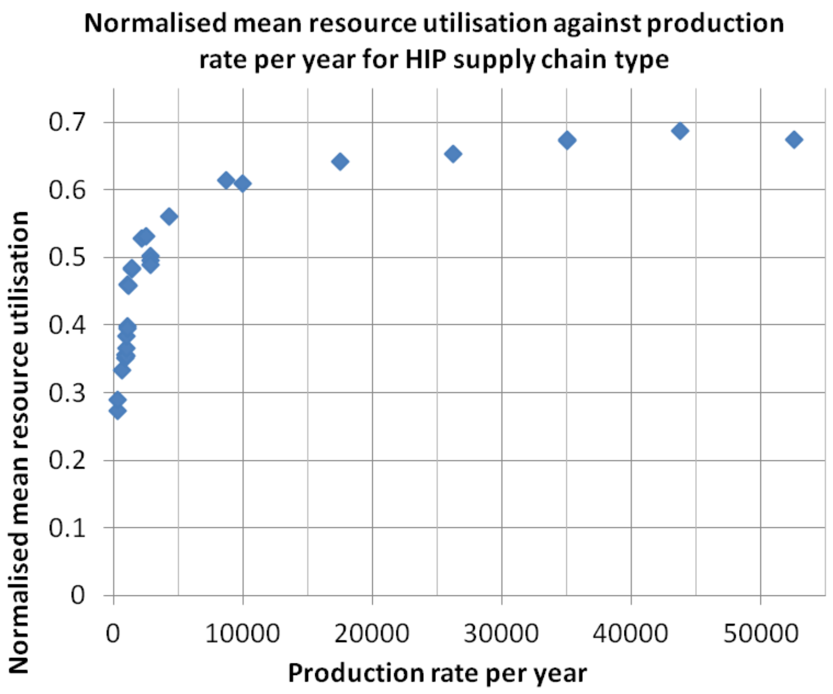


Figure 58: Graph showing normalised mean resource utilisation against production rate per year for the HIP supply chain type

The second graph, Figure 59, shows an almost linear relationship between process cost percentage difference and normalised mean utilisation. These two graphs show the relationship between production rate, resource utilisation and percentage difference of process cost. They could be used to determine when this framework is most useful.

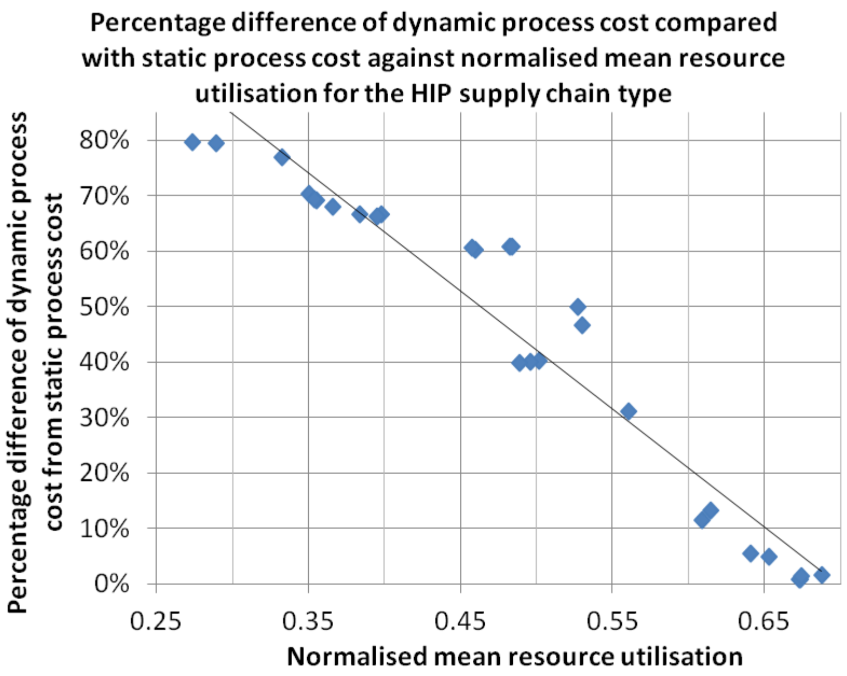
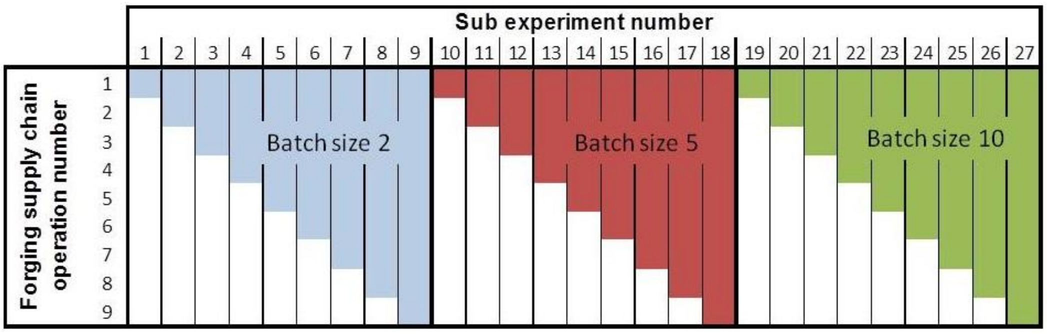


Figure 59: Graph showing percentage difference of dynamic process cost from static process cost against normalised mean resource utilisation for the HIP supply chain type

Experiment five modifies the number and size of batch operations within a supply chain. To complete this experiment the forging supply chain was utilised as a default then sub experiments were conducted by changing necessary non-batch operations to batch operations with a size using the procedure shown in Table 6. Therefore sub-experiment 1 required op1 to become a batch op of size 2, whereas sub-experiment 23 required ops1-5 to become batch ops with a size of 10.

Table 6: Case study 1 – experiment five parameter setup



Forging supply chain operation number	Sub experiment number																										
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
2		🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦		🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥		🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
3			🟦	🟦	🟦	🟦	🟦	🟦	🟦			🟥	🟥	🟥	🟥	🟥	🟥	🟥			🟩	🟩	🟩	🟩	🟩	🟩	🟩
4				🟦	🟦	🟦	🟦	🟦	🟦				🟥	🟥	🟥	🟥	🟥	🟥				🟩	🟩	🟩	🟩	🟩	🟩
5					🟦	🟦	🟦	🟦	🟦					🟥	🟥	🟥	🟥	🟥					🟩	🟩	🟩	🟩	🟩
6						🟦	🟦	🟦	🟦						🟥	🟥	🟥	🟥						🟩	🟩	🟩	🟩
7							🟦	🟦	🟦							🟥	🟥	🟥							🟩	🟩	🟩
8								🟦	🟦								🟥	🟥								🟩	🟩
9									🟦									🟥									🟩

🟦 Batch size 2 🟥 Batch size 5 🟩 Batch size 10

The graphs in Figure 60 and Figure 61 show how the number and size of batch operations affect the percentage difference of dynamic cost compared with static cost for a production rate of four components an hour. Figure 60 shows the process cost difference and Figure 61 shows the overhead cost difference. Both graphs show the difference increasing as the number of batch operations increases within the supply chain. They also show that the size of the batch operation affects the difference. The results may be exaggerated due to the batch operations being sequential within the supply chain, and the fact that some operations have short process times as shown in Table 3.

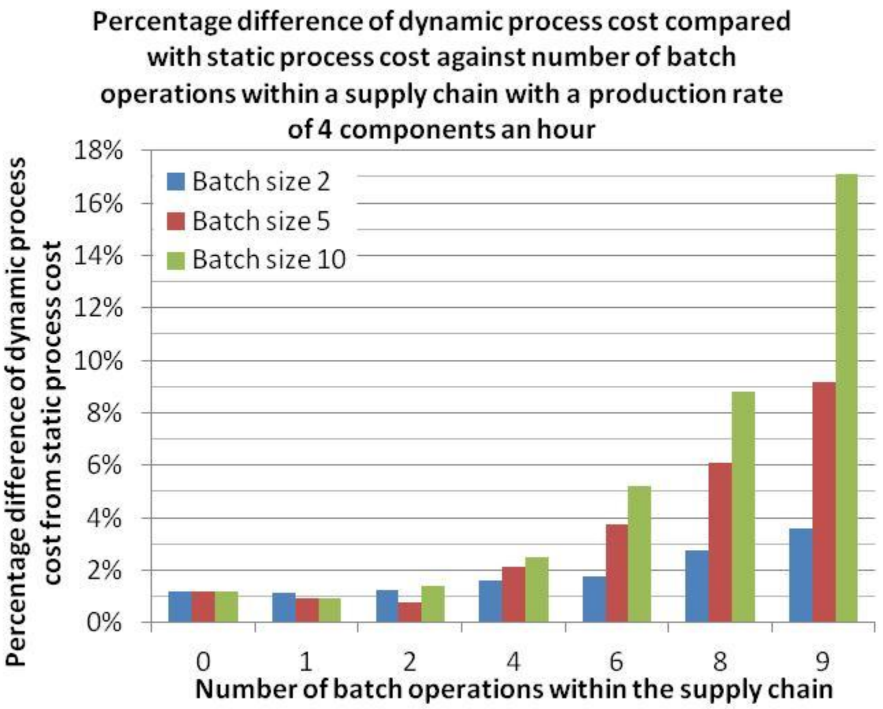


Figure 60: Graph showing percentage difference of dynamic process cost compared with static process cost against number of batch operations within a supply chain

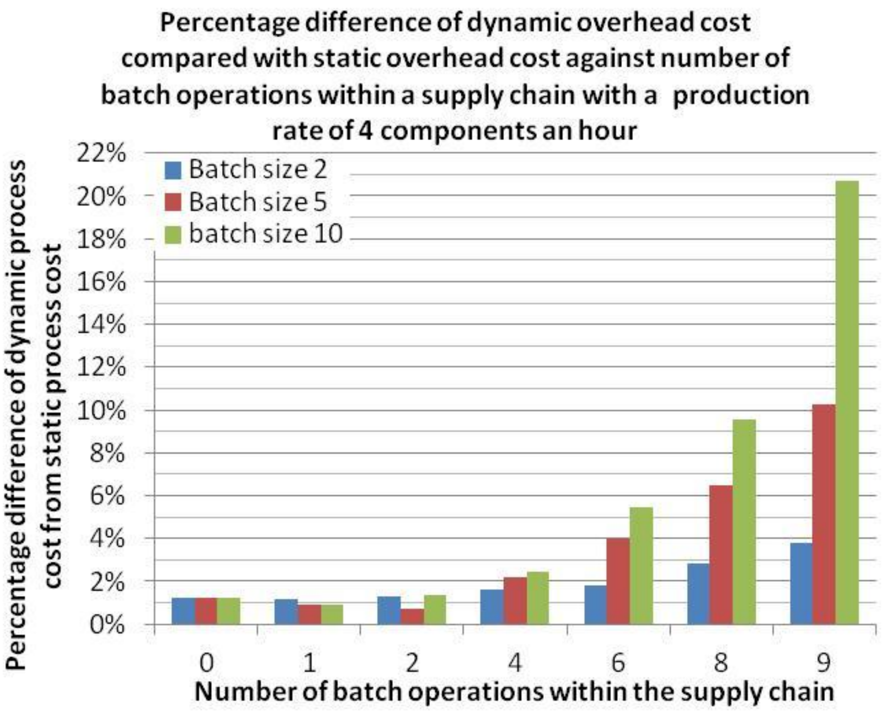


Figure 61: Graph showing percentage difference of dynamic overhead cost compared with static process cost against number of batch operations within a supply chain

4.1.2 Summary of case study 1

All the results show there is a difference between static and dynamic cost for supply chains that contain batch operations with the size of the percentage difference increasing when:

- Quantity and size of batch operations is increased
- Batch operations are more heavily weighted compared to other operations within the supply chain due to batch operation being a fixed cost and most other process costs are dependent on component geometry
- Different resources are utilised to complete an operation
- Production rate decreases which reduces resource utilisation resulting in increased process cost

4.2 Case study 2 – Blisk

The purpose of this case study is to highlight two points. Firstly, the framework is capable of being used on a wide variety of components. Secondly, to illustrate the steps necessary to implement a new component within the framework.

The component for this case study is a blisk, shown in Figure 62, which combines blades and a disk into a single component. The combination into a single component has various advantages, reduced mass being one. The reduced mass is achieved by removing the blade and disc joining mechanisms by either producing the blisk from a solid piece of metal or by welding the blades onto the disc.

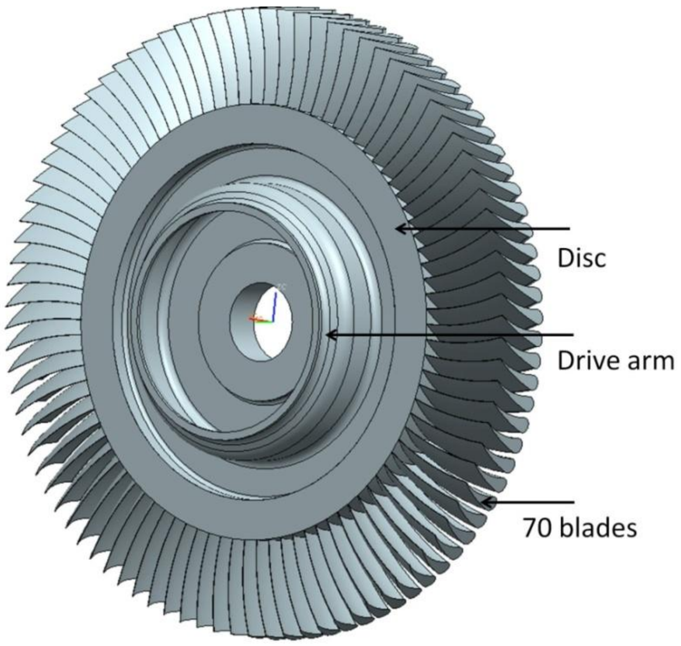


Figure 62: Case study 2 component based on a representative aerospace gas turbine blisk

Only one supply chain type has been applied to this component, which machines a solid forging but does not complete finishing operations such as grinding. For this reason the supply chain type contains three methods. The first method turns the drive arm side of the disc by completing seven separate turning operations. The second method turns the non-drive arm side by completing two operations. The third method utilises Electro-Chemical Machining (ECM) (Zhan, Zhao et al. 2000) to roughly remove the material between each blade, then shapes each blade with a finishing ECM operation.

Implementing the framework for a new component requires work in three areas: creation of the component and state geometries within the geometry engine; input of data into the database; and addition of code to cope with the new component and supply chain type. Specific steps required to be completed within the geometry engine include:

1. Create parameterised component geometry
2. Create parameterised state geometries utilising rules to build upon each state, starting at the component geometry and ending with the COS for the supply chain type
3. Create top level assemble file that holds all supply chain types required
4. Add and link component expressions to the top level assemble file including component identification, type and material

Determining the details of the parameterised component geometry and understanding the rules required to create the state geometries, represent the majority of the effort within the geometry engine. Specific steps required to be completed within the database include:

1. Add component identification and type to appropriate tables
2. Add supply chain type data including: options, methods and operations
3. Add machines, fixtures and resources when relevant data is required
4. Link new data together with link tables

Understanding the supply chain type and gathering data for required resources are the main steps to be completed within the database. Specific steps required to be completed within the integration code include:

1. Add code to retrieve measurement data for supply chain type
2. Add code for each method within the supply chain type for operation time generation, setup time generation and batch calculation
3. Add code to calculate the extent of the COS boundary limits for machine selection

Other than data collection the code required to generate the process time for each operation within a supply chain represents the majority of the effort required for all of the steps to integrate a new component into the framework. However this effort has substantially reduced compared to the first case study. This is because most of the generic functionality and data required was created for the first case study which can be reused by any subsequent case studies.

4.2.1 Results

The graph in Figure 63 shows normalised cost against number of blades for a disc diameter of 220mm, Nickel 1 as the material and a production rate of 2 components an hour. The number of blades on the blisk is modified for three different blade lengths. The results show how cost increases as the number of and length of the blade increases; this is because the ECM operations contribute the majority of process cost and are dependent on the number and length of the blades.

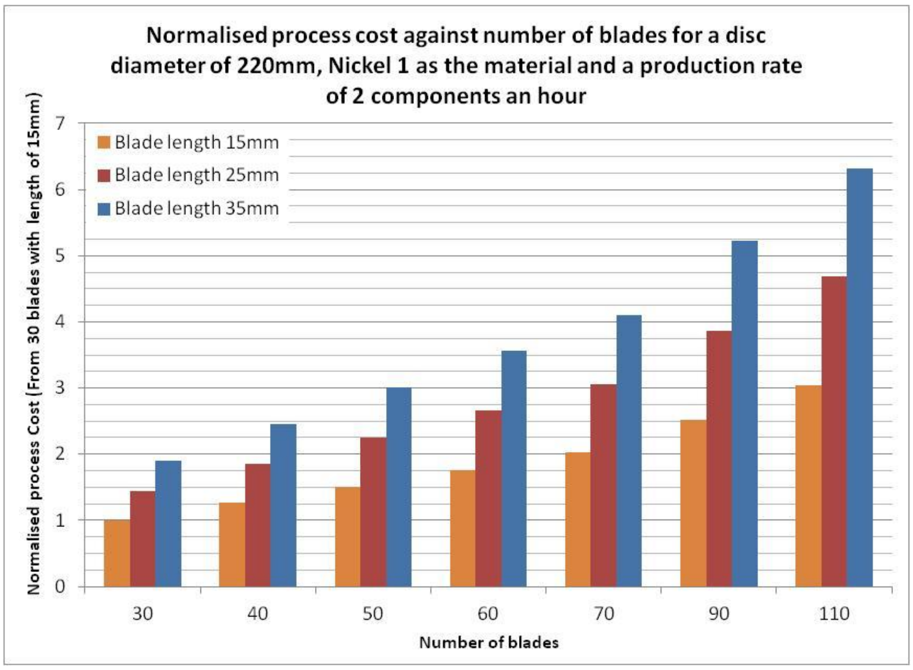


Figure 63: Graph showing normalised process cost against number of blades for a disc diameter of 220mm, Nickel 1 as the material and a production rate of 2 components an hour

4.2.2 Summary of case study 2

The second case study has shown that integrating different components into the framework is possible. It highlighted that an understanding of the geometry, supply chain type and specific operations within the methods is required. Also, other than data collection, a significant amount of effort is spent creating code to calculate process times for each operation.

4.3 Chapter Summary

The first case study has shown that there is a difference between static and dynamic process and overhead costs for components with supply chains that contain batch operations. The percentage difference depends on many factors, but increases when:

- Quantity and size of batch operations is increased
- Batch operations are more heavily weighted compared to other operations within the supply chain due to batch operation being a fixed cost and most other process costs are dependent on component geometry
- Different resources are utilised to complete an operation
- Production rate decreases which reduces resource utilisation resulting in increased process cost

It could be seen from experiment 4 that a 10% percentage difference was the result of a 65% mean resource utilisation, which equates to between 10,000 and 15,000 components per year. The percentage difference continues to increase by 10% for every 5% decrease in mean resource utilisation.

By implementing a Blisk the second case study has shown that different component types can be utilised by the framework. It also highlighted that after data and knowledge capture, which is a non-trivial task, a significant portion of the effort required to integrate a component into the framework is related to creating the time generation code for the manufacturing operations.

Chapter 5

Discussion

There are four sections in this chapter; the first discusses the findings from the case studies. The second discusses the benefits and required effort to achieve them. The third discusses validation, both case studies and future implementation of the framework. The last discusses, in two parts, improvements to the framework: first, improvements that should be implemented, and second, improvements that would add value.

5.1 Case studies

The first case study conducted five experiments, two that mimic how the user might conduct "what if" studies with different design choices and three experiments to understand the effect that production rate, available resources and batch operation characteristics have on the results. This case study had a high level purpose of determining if a percentage difference of dynamic cost compared with static cost existed, and to show if and when a benefit can be gained from the framework. The parameters affecting the results are discussed but will not be quantified with a value, because the size of the benefit depends on the interactions of all the parameters within the framework.

It can be clearly seen in the results from all experiments, that the inclusion of batch operations within a low volume supply chain, can affect the results from a static cost model due to an underestimation of the resources required. The inclusion of a dynamic model allowed the resources to be determined dynamically which resulted in an increase in unit cost, compared to the static calculation of unit cost, this is due to the increase of resources required.

Experiments one and two show how different component design parameters (internal radius, tolerances, surface finish and component material) coupled with the supply chain affect the percentage difference. Experiment one highlighted the cost bias within components, in this case how the fixed cost of the HIP cycle becomes a larger bias as the component becomes smaller, and how this can increase any unit cost error due to a error in resource calculation. In experiment one this resulted in the percentage difference increasing as internal radius decreased.

In experiment three it is shown how the availability of different resources affects the percentage difference results. By selecting the lowest cost resource that is capable of completing the desired operation the percentage difference can be reduced as shown in Figure 54 and Figure 55. It would therefore be good practice to ensure there are different resources available for each operation when populating the database during the implementation of a new component or supply chain option. Also this ability is particularly useful for manufacturing engineers to determine the effect of resources on both the unit cost and the supply chain.

In experiment four it is seen that as production rate increases, the percentage difference decreases (shown in Figure 56 Figure 57) because all resources within the supply chain reach their utilisation maximum values seen in Figure 58. Therefore the resources and overhead costs are spread across the optimum number of components, which is an assumption of the static based cost estimation methods, hence the percentage difference reduces. To gain maximum benefit from the framework a high percentage difference is required, for experiment four the graph in Figure 59 shows that a normalised mean resource utilisation of less than 0.65 is required to gain a percentage difference of greater than 10%.

This creates two problems in understanding when to implement the framework. The first is to understand how the trend in Figure 59 changes with different supply chains. If this is understood a decision of when to implement the framework can be made. The second is based on the current framework requirement of resources being dedicated to a single supply chain; which is not always the case. Therefore there is scope for future work to determine how to incorporate multiple components utilising resources.

In experiment five the number and size of batch operations within a supply chain affects the percentage difference as shown in Figure 60 and Figure 61. As the number of or the size of the batch operations increases so does the percentage difference. These results may have been artificially exaggerated because the batch operations were sequentially placed within the supply chain, this means there were no non-batch operations in-between the batch operations to smooth the flow of components through the supply chain. However this further highlights how operations dynamically interact in unpredictable ways further adding to the evidence that the inclusion of a dynamic model is beneficial.

The second case study was a blisk component; its purpose was to highlight two points: firstly to show that widely different components could be implemented within the framework; secondly to illustrate the steps necessary to implement a new component within the framework. Integrating different component types into the framework highlighted that a detailed understanding of the geometry, supply chain type and manufacturing operations is required before the integration into the framework can be complete. It also highlighted that after data collection, which is a non-trivial task, a significant portion of the effort is spent creating code to calculate process times for each operation. The amount of effort required to add a new component or supply chain to the framework should reduce as more general manufacturing functions, manufacturing data, supply chains and components are incorporated in to the framework because data and methods can be reused.

5.2 Benefits against required effort

Deciding whether to implement the framework or not requires a trade off to be conducted between the benefits that will be delivered against the effort required to achieve them. This section discusses the benefits, the requirements to achieve the benefits and under which circumstances the benefits are worth the effort.

There are three primary benefits provided by the framework, which are:

- The framework can dynamically predict required resources to fulfil a supply chain for a specific production rate. This data is used in the second benefit but can also be utilised by manufacturing engineers to aid production planning.
- An increase in refinement of component unit cost estimate. This is completed by including the manufacturing time and dynamically determined resource requirements into an ABC cost model.
- The ability to compare multiple supply chain options and different supply chain types, at the same time, directly from component geometry.

There are two categories of requirements to achieve the framework benefits, these are: initial requirements to implement the framework; and requirements for each component and supply chain to be applied to the framework. The first category involves substantial effort to create the integration code, database and the generic data driven models. The integration code forms the backbone of the framework and needs to be tailored to the tools. Also the database and generic data driven models need to be created.

The second category is concerned with the knowledge, data and effort requirements to apply a new component or supply chain to the framework, these include:

- Knowledge concerning the rules required to create the parameterised component, state and COS geometries. This knowledge must ensure the geometry parameters of the component are what the user requires and that the component geometry is able to cope with the scope of required parameter changes
- Material data with understanding of the effect on manufacturing feeds and speeds
- Operation level understanding of the manufacturing sequence
- Understanding of which resources, in what quantities are utilised for each operation
- Resource data for resource selection process and cost calculations (data required includes: maximum operating envelope, consumable requirements, auxiliary equipment, total foot print, operator requirements and special to resource fixtures and tooling)
- For each new manufacturing process, knowledge of the logic to calculate the process times and manufacturing data (speeds, feeds and capabilities) which form the scientific basis of the process time
- Effort to create and implement the manufacturing process logic into the integration code

There are requirements in effort, knowledge and data before any benefit can be achieved by the framework. However this is true of any model, therefore three comparisons can be made between the requirements of sections of the framework and an individual 1) cost estimation model, 2) dynamic model and 3) geometry model. Each of these comparison models require effort, knowledge and data to build. It could be argued that if these comparison models were built there would be no need to go to the same level of detail as the framework, for instance creating the stage models in the geometry. However if an understanding of the machining sequence was required the stage models would be necessary. Aspects of the framework that require extra effort are centred around integration of the modelling tools, however utilisation of generic data driven models reduces the modelling effort compared to the individual cost and dynamic comparison models. Extra knowledge and data is required by the framework compared to all the comparison models because the comparison models are based on a single instance, whereas the framework needs to cope with variation of component and supply chain parameters. Therefore the amount of effort, knowledge and data required is arguably similar to any collection of individual models, however the framework requires all of it to be complete before any benefit can be achieved through integration.

There are four factors that result in maximum benefit from the framework; these are:

- Components that are process cost biased, instead of material cost biased, result in higher percentage difference because the framework benefits only apply to the dynamic aspects of a unit cost which are process and overhead costs.
- Supply chains that contain more batch operations and large quantity batch operations result in a higher percentage difference. This is because the dynamic modelling can determine the correct resource quantity to achieve the production rate.
- A production rate that results in the mean utilisation of the resources within a dedicated supply chain to be below 0.65. This is based on the supply chain in case study one and should ensure that the process cost percentage difference is above 10%. This result is because the resource utilisations are not achieving their maximum values, in this case 0.7.
- When there are two or more possible resources for an operation the framework can select a resource. This reduces the percentage difference however this selection ability is not seen in most cost modelling tools.

If the component and supply chain do not contain these requirements to achieve maximum benefit there are still opportunities to gain benefit by removing unknowns. This is because the framework can determine the cost effect of different parameter changes. Two examples of this are: components where material choice may change during the design process which can affect the results, therefore comparisons need to be conducted to determine the optimal supply chain selection; and when required production rate is unknown, because production rate affects unit cost and the amount of required resources.

The framework is designed to be a tool to aid design decisions however there are two situations where this will not happen. Novel component design or processes where an understanding of time taken or equipment is unknown cannot be applied to the framework because the required knowledge and data is not available. Also geometry parameterisation may inhibit designer creativity as it is limited in flexibility and scope. In these situations other tools are required until greater understanding is gained and can be applied to the framework.

5.3 Validation

There are approximately 15,000 lines of integration code and 1,000 lines of code within the dynamic model. Validating the code fully by completing multiple test cases or an expert review, is out of scope for this research. However the results and logic within critical functions that aid process time generation within the integration code, were subject to trend evaluation that confirmed the correct working of the functions within case study parameters.

No commercially sensitive data has been utilised within the framework, where possible publically available data has been used, otherwise appropriate values for the case studies were created. This means that the data utilised within the case studies has not been fully validated against actual manufacturing data for real components. However, by utilising the same base data for both static and dynamic costs calculation, the need for fully validated case studies was avoided, which would not have been possible for the data reasons above and the reasons given in Chapter 4.

For future implementation it is recommended that a detailed validation exercise against a known component takes place for any new component and supply chain type added to the framework. This exercise would determine a known set of results for a set of parameters for a specific component supply chain combination.

5.4 Framework improvements

The framework, in its current form, has shown that there are benefits to be gained under certain circumstances. However, based on the case studies the author has identified improvements that can be made to the framework to increase its flexibility and scope. These improvements fall into two categories: improvements that in ‘hindsight’ are deemed necessary to enhance the framework and improvements that would add value.

5.4.1 Necessary improvements

The following is recommended by the author as necessary to increase the cost modelling flexibility to the same level as the dynamic model. The current cost model implementation utilises three predefined resource types within the operation cost model. To ensure the cost model is fully flexible and can cope with any number of resources a third generic data driven cost model should be created. This resource level cost model, shown in Figure 64 stage 5.1, calculates the cost of a resource. Once each resource cost has been calculated the results are collated and used within the operation level cost model, then the process continues as described in section 3.1.5. Without this change the amount of resource types is limited to the three predefined ones, utilised within the proof of concept case studies, with this change the cost model is not limited and therefore matches the dynamic model capability.

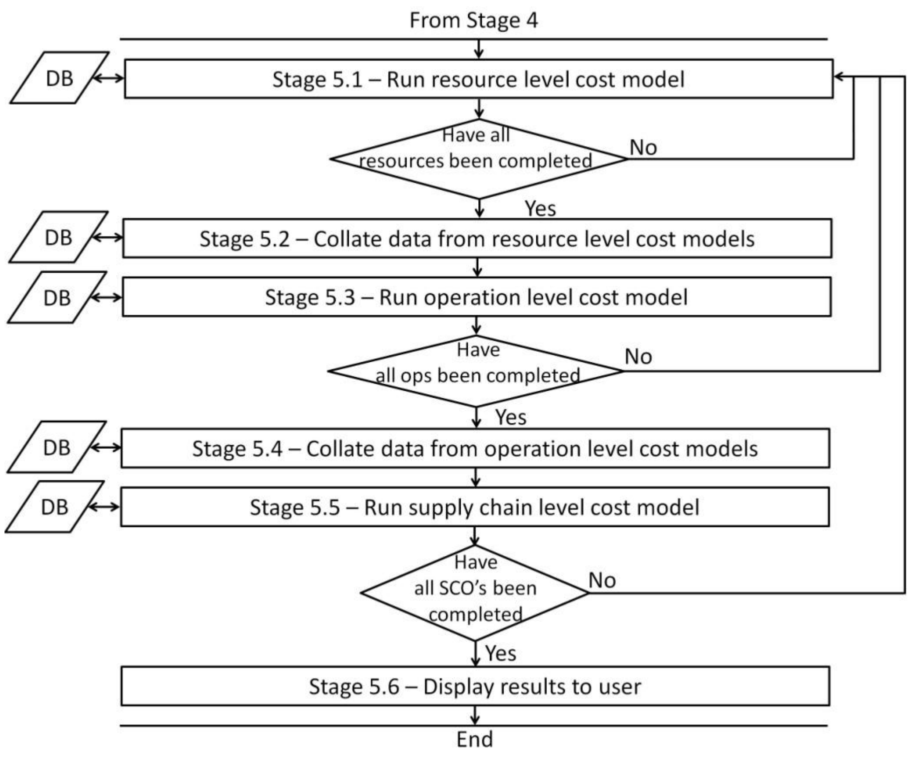

From Stage 4
DB
Stage 5.1 – Run resource level cost model
Have all resources been completed
No
Yes
DB
Stage 5.2 – Collate data from resource level cost models
DB
Stage 5.3 – Run operation level cost model
Have all ops been completed
No
Yes
DB
Stage 5.4 – Collate data from operation level cost models
DB
Stage 5.5 – Run supply chain level cost model
Have all SCO's been completed
No
Yes
Stage 5.6 – Display results to user
End

Figure 64: Recommended implementation of the framework stage 5

5.4.2 Improvements that would add value

There are a number of improvements that would add value to the framework. Five general areas within the framework that could be improved are:

- Cost
- Resource utilisation type selection
- Dynamic model

- Optimisation
- Technology integration

By implementing the cost calculation within the integration code execution time could be reduced. Currently cost is implemented within Vanguard as purpose built generic data driven cost models, which are accessed over a network. By implementing the cost equations within the integration code this would decrease cost calculation execution time because data is not transferred over a network. A disadvantage of this is an increase to the code validation process and any required update to the cost calculation may require more effort because it will be contained internally within the integration code instead of externally as in the case of the generic data driven cost models. However the reduced execution time would be beneficial when completing holistic optimisations.

Another cost based improvement would include extra data and functionality to allow more cost rates to be dynamic. For instance if a user was given a choice of worldwide locations for the supply chain, or if the choice was a part of an optimisation, different cost rates would apply to parameters such as factory foot print, consumables and wages as the location of the supply chain changes.

Each resource has a utilisation maximum value that is held in the database. This functionality can be extended to allow a user to select a type of utilisation maximum. For instance there could be three types: min, standard and max. Min would represent the minimum utilisation value that the company is willing to accept. Standard would represent what the company believes is the normal utilisation. Max represents the industry best value. The user could utilise this functionality to gain the best, normal and worst case scenarios or different utilisation types could be applied throughout the simulation to act as a learning curve.

Extra functionality could be applied to the dynamic model such as:

- Applying a distribution to each operation that represents the probability of scrap. This would be used by the scrap cost rate in the operation level cost model.
- Implementing a schedule for maintenance and a distribution for unscheduled maintenance would remove the requirement for a resource utilisation maximum to be utilised. This is because the utilisation would then become an output of the dynamic model instead of an input.

However this could only be implemented if knowledge of scheduled and un-scheduled maintenance was available as well as other necessary factors.

- The dynamic model can only represent a supply chain where the resources contained within it are utilised by the component being considered. A method to overcome this is to block out a percentage of resource capacity by reducing the individual resource utilisations. Another method is to use a two step approach discussed in the next point.
- The dynamic model determines an idealised supply chain. If data concerning resources within current supply chains and components passing through those supply chains were available, a second stage holistic simulation could be conducted. This simulation would determine if the new supply chain could be accommodated within the current capabilities of a company at the holistic level.
- The dynamic model could determine the risk associated with the supply chain. For instance there could be a choice between two resources where resource 1 requires a quantity of two to meet the production rate and resource 2 requires a quantity of one. The supply chain utilising resource 1 would results in a higher unit cost, because more resource is required, and a lower risk. Whereas the supply chain utilising resource 2 results in a lower unit cost but a higher risk because it is a single point failure within the supply chain. In that situation the supply chain utilising resource 1 may be more appropriate due to the lower risk.

An optimisation loop could be utilised to select resources. Currently suitable and capable resources are selected by resource cost attributes. However resources could be selected based on results of the supply chain as a whole and utilise other attributes such as risk.

The framework currently calculates operation process time through the utilisation of generic manufacturing time generation functions held within the integration code. A method of utilising the Computer Aided Manufacturing (CAM) tool with the CAD tool UGS NX 6 to complete this task was determined to be incapable because the CAM tool was unable to cope with the addition and removal of features automatically. However the ability of CAM tools to complete this task will change as their capabilities increase, which could result in improved time generation accuracy and aid optimisation. As the CAM tool should be able to simulate the tool paths more realistically.

Parameterisation of the component geometry has been utilised within the framework as the method of generating the required geometry. This combined with a direct link to the geometry and a variant CAPP approach allowed a process plan populated with process times and initial resource requirements to be generated. A combination of AFR and generative CAPP methods was the first choice to complete these tasks. However, as discussed in sections 2.3.1 and 2.3.2, the current capabilities of these methods are limited, but as they develop they would offer greater flexibility and extended scope to the framework.

Chapter 6
Conclusions and future work

This chapter presents the significant conclusions of this research, followed by key contributions to the research field. Recommendations of future research building on findings from this work are discussed before the concluding remarks.

6.1 Conclusions

A framework has been created that integrates a dynamic modelling capability with component geometry and unit cost estimation. A system called DRES embodies the framework by integrating three primary tools: a dynamic modelling tool, a geometry engine and cost modelling tool. The primary purpose of DRES was a proof of concept and to conduct two case studies. The first case study was to determine if and when a benefit could be gained from the framework. The second case study highlighted that different components could be implemented within the framework and the steps necessary to complete it.

Integration of the three tools gives three direct benefits. First, the integration of a dynamic modelling capability allows optimised resource requirements to be utilised within unit cost estimation, therefore producing an AUC estimate. The integration of component geometry facilitates the second and third direct benefits. The second is that the integration allows a user to understand the consequences of design changes on unit cost. The Third, integration allows real time decision making to take place.

By utilising a direct method of extracting geometry data and a variant CAPP method to determine a manufacturing process, the framework allows the user to select multiple supply chain types and multiple options within each type. This flexibility supplies the user with a benefit that allows unit cost comparison of multiple supply chains.

The goal of the dynamic model is to determine required resource of a supply chain for a component so that it can be utilised in a cost estimation model. A secondary benefit is to utilise required resource data to reduce time and effort required by manufacturing engineers to develop the supply chain for manufacture.

There are two application requirements that must be taken into consideration before a component is applied to the framework, these are: manufacturing production rate, and availability of knowledge and data. An assumption of the dynamic model, in its current form, is that all resources within a supply chain are dedicated to the manufacture of the component; see section 5.4 for possible improvements. Due to this assumption production rate affects the amount of benefit that can be gained from the system. For example if production rate is high (52,560 components a year for experiment 4 of case study 1 Figure 56) the potential benefit against a static cost estimation is low (1.3% in experiment 4 case study 1). However, as the production rate decreases (365 components a year for experiment 4 case study 1 Figure 56) the potential benefit increases (79.7% in experiment 4 case study 1). Therefore production rate needs to be considered to determine if a benefit will be gained from the framework. The second requirement to consider is availability of knowledge and data. It was discussed in section 5.2 that the amount of knowledge and data required by the framework is similar to building the necessary models individually. However the knowledge and data is all required before a component is applied to the framework because without it the framework will not deliver any output for that component. This constitutes a considerable upfront commitment to the framework because it is an all or nothing situation.

Four characteristics have been found from the cases studies that provide the most benefit from the framework. The first is delivered by components with a unit cost that is process cost biased instead of material cost biased. Optimising resource requirements only affects the dynamic aspects of unit cost – process and overhead costs – therefore components that are process cost biased will see more benefit than components which are not (see experiment 1 case study 1, section 4.1.1). The second characteristic concerns components with supply chains that contain batch operations. Static cost models have difficulty determining resource requirements for batch operations, therefore more benefit can be gained from supply chains that contain many large quantity batch operations (see experiment 5 case study 1, section 4.1.1). The third characteristic is production rate required. For a dedicated supply chain as production rate decreases resource utilisation decreases which exaggerates any errors in resource quantity calculation, therefore effecting the unit cost and, in experiment 4, the percentage difference. The fourth characteristic is selection of possible resources which the framework completes automatically.

During the design process there are situations when some design parameters and decisions are either unknown or have multiple possible choices. These design decisions can affect many aspects of the design including the manufacturing process and unit cost. An example of this is component material choice which could affect supply chain selection. A second example is when required production rate is unknown or is different to known historical data; this can affect supply chain selection and unit cost. The framework can be utilised to understand different situations to aid design decision making, by allowing a design team to investigate the affects of deign decisions.

The framework can deliver benefits as discussed above, however there are disadvantages. The majority of effort required to implement the initial framework is up front, due to two reasons:

- Integration of the five stages and necessary tools for the framework. The integration process is a considerable task especially because the user does not see the integration as each tool is linked within the framework to work automatically. All aspects of tool integration, user interaction through the GUI's and data manipulation must be handled by the integration code through a series of error checking routines which further increase the complexity of the integration.
- Generic ability of the framework to cope with different components and supply chain types. Creating a robust parametric geometry is a technical challenge, the framework however takes this a step further by requiring that the generic ability is represented throughout each stage of the framework including: data handling, manufacturing process time generation, dynamic modelling and cost modelling.

Once the initial effort of creating the framework integration has been completed substantial data and knowledge is required to implement a component and supply chain type within the framework as discussed in Case study 2 – Blisk section 4.2.

Novel components, a new component design not seen before, or processes, new manufacturing process, cannot be applied to the framework in its current form. This is because the framework is based on a generic parameterised component geometry that is linked to multiple known manufacturing processes that is linked to data and knowledge about every aspect of its manufacture. A novel design or process, by definition, does not have all the necessary data or knowledge to populate the database or equations to determine the process operation times. The topic of including novel design or manufacturing process is discussed in future research.

The hypothesis for this research was “Integrating supply chain simulations with design geometry can assist in design decision making”. The conclusions from this research prove that the hypothesis is correct when the component and supply chain characteristics that produce maximum benefit are partly achieved. DRES the proof of concept implementation of the framework has achieved the aim of this research, which was to assist the design process by aiding decision making by conducting real time cost estimation, incorporating a dynamic aspect into unit cost estimation and allowing comparisons of manufacturing processes.

6.2 Contributions of research

It was determined in section 2.1, cost estimation, of the literature review that cost estimation methods were based on static models that were unable to fully represent dynamic systems. This research has shown that there are benefits of utilising dynamic modelling to provide dynamic data to cost estimation methods. This research also discovered characteristics when the most benefit can be achieved; these are:

- A process cost biased component
- Supply chains that contain batch operations
- Production rates that result in mean resource utilisation below 0.65 for dedicated supply chains
- Multiple possible resources for operations.

The literature review also showed that some researchers have integrated dynamic modelling and cost estimation, section **Error! Reference source not found.**, some have integrated geometry and cost estimation, section **Error! Reference source not found.**, and some have integrated dynamic modelling and geometry, section 2.2.2. However no research has focused on the integration of the three areas for the purpose of aiding design decisions. This research has integrated all three areas, as shown in Figure 65, within a framework and has implemented the framework with a system called DRES.

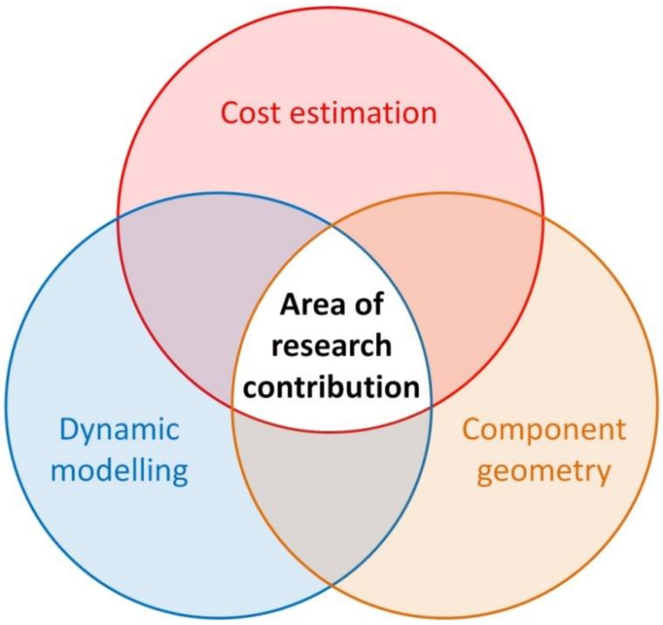


Figure 65: Area of research contribution

6.3 Future research

Implementing the recommendations in section 5.4, framework improvements, provide value but are not research questions. The rest of this section discusses those research questions that have come to light and areas of research that are not fully understood and would be required to achieve the vision the author has for the framework.

Fully understanding the benefits of integrating a dynamic modelling capability into cost estimation and under which criteria the benefits occur, will provide a comprehensive guide of when to apply the framework. Determining this will require experimentations with different supply chain types to understand the dynamic interaction of operations by varying:

- Size of operation process times and the sequence of operations within the supply chain
- Sequence of batch and non-batch operations as well as batch sizes
- The amount of different resources per operation and the quantity of those resources

Understanding these interactions will provide a greater understanding of when to implement the framework for components, which is a necessity to create a business case due to the investment that this framework requires before use in a production design environment.

Understanding and quantifying the risk of different supply chain options is a research area that will allow a user to make an informed decision about the selection of a supply chain option. For instance when comparing two supply chain options one may result in a lower unit cost but may present a higher risk, therefore the second supply chain option may be a better choice. To understand risk the following areas need investigation:

- What constitutes a risk within the supply chain, an example of this is the utilisation of a single resource in conjunction with un-planned maintenance and the fact that this represents a single point of failure
- How risk is quantified and presented to the user
- Is extra data or capability required within the dynamic model logic to cope with determining risk

The research in to quantifying risk of different supply chain options is required to aid design decisions for people who do not fully understand the manufacturing process and the implications one supply chain has over another.

Optimisation that combines this framework with other analysis systems such as finite element analysis and computational fluid dynamics could be used to optimise the geometry directly. The combination of these systems would allow a multi-objective optimisation for a component or set of components such as a gas turbine engine or subsystem. This research would try to understand how the analysis outputs would interact, and how these affect geometry parameter values. This would allow the framework to truly achieve its aim by being fully incorporated into the design optimisation.

The framework determines the idealised supply chain requirements for the component and the selected supply chain. However this could be the first stage of a two stage optimisation. For instance the second stage could determine if the idealised supply chain can be incorporated into a company's current supply chain capabilities, and if not, could determine what extra resources are required. This would also force an understanding of all the supply chain capabilities within a company as all the data and knowledge would be required to complete the second stage optimisation. Creating a generic data driven dynamic model that can represent multiple supply chains and multiple components where resources are not dedicated to a single supply chain would be required for this work. Also how the results would be viewed by the user needs consideration.

If a company was to use this two stage optimisation, to determine what extra capacity is required, would be the ultimate aim of the framework. This is because this would force the company to completely model its manufacturing capacity which could be used to optimise each year's load and capacity incorporating the new component volumes.

The use of novel components or processes within the framework is difficult because not all data or knowledge is available. However, an understanding of what data and knowledge is required, what makes a component ‘novel’, and how the lack of data can be overcome would aid the inclusion of novel components and processes within the framework. Without this understanding the framework will be limited to families of components and known processes, which is a limitation for aiding design decisions.

6.4 Concluding remarks

The integration of multiple systems is becoming more common place as multi-objective problems are been solved. The inclusion of dynamic modelling in these integrations is also becoming more common, especially with increasing computer capability that allows dynamic models to execute in a short time.

The capability of creating generic data driven dynamic models is allowing researchers to utilise the models to solve and understand problems instead of building models. Commercial dynamic modelling tools are now capable of combining multiple dynamic modelling methods (DES, continuous, system dynamic and ABC) into a single model. This will aid researchers to apply different dynamic modelling methods to different parts of a single problem, allowing them to focus on finding a solution.

The next step of cost estimation is encompassed partly in this research. It is not about supplying a cost service to a design team or even a tool that requires extra effort and time on their part. It is about supplying a design team with a tool that aids them to understand cost in terms of their design decisions in real time. To do that it needs: to be automated, to include the geometry, to include the manufacturing process, to be optimised at all levels, and to be based on scientific data.

Appendix A

There are approximately 16,000 lines of code within the framework, which has many repeated sections therefore this appendix contains extracts from the framework integration code that was directly discussed within the thesis. The code extracts are:

- Extracting parameters using the geometry engine API
- Generating a process time for turning
- Executing the generic data driven dynamic model

The code in A.1 is specific to the geometry engine which is Siemens Unigraphics NX 6.0. All the code has been written in C# using MS Visual Studio.

A.1 Code to extract parameters using geometry API

Below is the code required to extract parameters from the top level assemble file for the component geometry. The geometry engine calls the parameters expressions, hence the use of the word expressions throughout the code. Each parameter name, value and unique tag number are extracted and put into a data table for storage until required later in the framework. As some parameters are strings, for example N7 as a tolerance, and some are number values they represent different types of data, therefore they need to be handled and stored differently within the code.

```
/// <summary>
/// <para>Retrieves the expressions data from the NX work part (eg the assembly)</para>
/// <para>Returns DataTable</para>
/// </summary>
private DataTable retrieveComponentExpressions()
{
   Session theSession = Session.GetSession(); // Retrieves the current sessions
   Part workPart = theSession.Parts.Work; // Retrievs the current work part
   PartLoadStatus partLoadStatus1;
   NXOpen.Assemblies.Component nullAssemblies_Component = null; // Loads assemble
   theSession.Parts.SetWorkComponent(nullAssemblies_Component, out          partLoadStatus1); // Sets the work
component to the assemble

   workPart = theSession.Parts.Work; // Retrievs the new work part
   partLoadStatus1.Dispose();

   DataTable dt = new DataTable(); // Initiates a new datatable
   dt.TableName = "Component_Expressions";
   dt.Columns.Add("Tag", typeof(int)); // Creates new column in dt
   dt.Columns.Add("Value", typeof(double)); // Creates new column in dt
   dt.Columns.Add("Value_String", typeof(string)); // Creates new column in dt
   dt.Columns.Add("Name", typeof(string)); // Creates new column in dt
   dt.Columns.Add("Description", typeof(string)); // Creates new column in dt

   // Loads all expressions in work part and iterates through each
   foreach (Expression expression in workPart.Expressions)
   {
      if (!expression.Name.StartsWith("p"))//Ignores expressions starting with p
      {
         double value = 0;
         string value_string = "";
         if (expression.Type.Equals("Number"))// If expression is a number
```

```
        {
           value = expression.Value;// Assigns expression value to variable
        }
        if (expression.Type.Equals("String")) // If expression is a string
        {
                                // Removes unecessary characters from the string
           value = 0;
           int start = expression.Equation.IndexOf("=\"") + 2;
           int end = expression.Equation.Length - 1;
                                // Assigns expression string to variable
           value_string = expression.Equation.Substring(start, end - start);
        }
                    // Applies extracted data to a datatable
        dt.Rows.Add(expression.Tag, value, value_string, expression.Name,                        expression.Description);
     }
  }
  return dt;
}
```

A.2 Code to generate an operation time

Below is the code that extracts geometry data related to operation then calculates the operation time based on the geometry information and cutting parameters based on speed and feed data from the database.

```
      #region Turn_PF020 (Uses PF010) - Complete - PF010 to PF020 - Hold sm end - machining          large flange end, face
and internal profile
      /// <summary>
      /// <para>Returns the time taken to machine the PF010 stage to the PF020 stage uses   PF010</para>
      /// <para>PF010 to PF020 - machining large flange end, face and internal     profile</para>
      /// </summary>
      /// <returns></returns>
      public double machine_PF010_to_PF020()
      {
         double timeTotalForPF010_TO_PP020 = 0;
         int location = 0;
         double SubLocation = 0;
         double SubSubLocation = 0;
         try
         {
            #region Extracts Depth of cut, Length of face, length of internal profile
            location = 1;
            SubLocation = 1;
            Session theSession = Session.GetSession();
            Part workPart = theSession.Parts.Work;
            Part displayPart = theSession.Parts.Display;

            NXObject nullNXObject = null;
            MeasureDistanceBuilder measureDistanceBuilder1;
            measureDistanceBuilder1 =
              workPart.MeasureManager.CreateMeasureDistanceBuilder(nullNXObject);
            measureDistanceBuilder1.Mtype =   NXOpen.MeasureDistanceBuilder.MeasureType.Minimum;

            NXOpen.Assemblies.Component component1 =
                              (NXOpen.Assemblies.Component)displayPart.ComponentAssembly.RootComponent.FindObject("
              COMPONENT COC_PF010 1");
            Unit nullUnit = null;
            DisplayableObject[] objects1 = new DisplayableObject[1];
            MeasureLength measureLength1;

            // ********* Depth of cut ***********
            SubLocation = 2;
            Line line1 = (Line)component1.FindObject("PROTO#.Sketches|SKETCH_007|Curve              Line43");  //  Depth  of
cut line
            objects1[0] = line1;
            measureLength1 = workPart.MeasureManager.NewLength(nullUnit, objects1);
            double depthOfCut = measureLength1.Value;

            // ********* length of face ************
            SubLocation = 3;
            Line line2 = (Line)component1.FindObject("PROTO#.Sketches|SKETCH_007|Curve                         Line41");
            objects1[0] = line2;
            measureLength1 = workPart.MeasureManager.NewLength(nullUnit, objects1);
            double lengthOfFace = measureLength1.Value;

            // ********* AvgDia of face ***************
            SubLocation = 4;
            Edge edge1 = (Edge)component1.FindObject("PROTO#.Features|LINKED_BODY(1)|EDGE                 *        784
REVOLVED(6) [CURVE 2 0] {(299.4999999999999,299.0385719267667,-
  172.65)(299.4999999999999,0,345.3)(299.4999999999999,-299.0385719267666,-
  172.6500000000002) LINKED_BODY(1)}");
DisplayableObject objects2 = edge1;
MeasureDistance measureDistance1;
measureDistance1 = workPart.MeasureManager.NewDistance(nullUnit,objects2);
double ExternalRadOfFace = measureDistance1.Value;

SubLocation = 5;
edge1 = (Edge)component1.FindObject("PROTO#.Features|REVOLVED(6)|EDGE *                    [CURVE  0
0] * [CURVE 2 0] {(299.4999999999999,279.2931927204815,-
  161.2499999999999)(299.4999999999999,0,322.4999999999999)(299.4999999999999,-
  279.2931927204814,-161.2500000000001) LINKED_BODY(1)}");
objects2 = edge1;
measureDistance1 = workPart.MeasureManager.NewDistance(nullUnit, objects2);
double InternalRadOfFace = measureDistance1.Value;

double avgDiaOfFace = (((ExternalRadOfFace - InternalRadOfFace)/2) +                    InternalRadOfFace)   *
2;

// ********* Length of internal profile stage 2 *********
SubLocation = 6;
SubSubLocation = 1;
Line line5 = (Line)component1.FindObject("PROTO#.Sketches|SKETCH_007|Curve                    Line43"); //
Short line at top which represents the bulk material sticking                    out from the flange face
SubSubLocation = 2;
objects1[0] = line5;
SubSubLocation = 3;
measureLength1 = workPart.MeasureManager.NewLength(nullUnit, objects1);
double lengthOfInternalProfile2 = measureLength1.Value;
SubSubLocation = 0;

// ********* Length of internal profile stage 1 ***********
SubLocation = 7;
SubSubLocation = 1;
Array.Resize(ref objects1, 4);
Line line3 =                    (Line)component1.FindObject("PROTO#.Features|INTERSECTION_CURVES(3)|CURVE
54 {3                    (295.6918708717893,0,323.5)}");
objects1[0] = line3;
SubSubLocation = 2;
Arc arc1 =                    (Arc)component1.FindObject("PROTO#.Features|INTERSECTION_CURVES(3)|CURVE  57
{5                    (290.8205505282297,-0,324.2466210044534)}");
SubSubLocation = 2.1;
objects1[1] = arc1;
SubSubLocation = 3;
Line line4 =                    (Line)component1.FindObject("PROTO#.Features|INTERSECTION_CURVES(3)|CURVE
58 {3                    (287.9933302293619,0,324.5)}");
objects1[2] = line4;
SubSubLocation = 4;
Arc arc2 =                    (Arc)component1.FindObject("PROTO#.Features|INTERSECTION_CURVES(3)|CURVE  60
{5                    (286.07869500536,0,324.4151502696375)}");
objects1[3] = arc2;
SubSubLocation = 5;
measureLength1 = workPart.MeasureManager.NewLength(nullUnit, objects1);
SubSubLocation = 6;
double lengthOfInternalProfile1 = measureLength1.Value;

// ************ End of data collection ****************
SubLocation = 8;
measureLength1.Dispose();
measureDistance1.Dispose();
measureDistanceBuilder1.LengthObjects.Clear();
measureDistanceBuilder1.Destroy();
displayPart.FacetedBodies.DeleteTemporaryFacesAndEdges();

#endregion

#region Calculation of time
location = 2;
SubLocation = 1;
clsMachiningTimeGen machTime = new clsMachiningTimeGen();
int speedType = 30;

// Turning data for face op
double timeFace = machTime.turning(avgDiaOfFace, lengthOfFace, 3,                    machinabilityNumber,
speedType);
// Turning data for internal profile op
double timeInternalProfile = machTime.turning(InternalRadOfFace * 2,                    lengthOfInternalProfile1          +
lengthOfInternalProfile2, 3, machinabilityNumber,                    speedType);

#endregion

#region Calculation of tool changes and totals
location = 3;
SubLocation = 1;
int toolChangesFace = (int)Math.Ceiling(timeFace / speedType);
int toolChangesInternalProfile = (int)Math.Ceiling(timeInternalProfile /                    speedType);
```

```
            // **** Totals
            double toolChangeTotal = toolChangesFace + toolChangesInternalProfile + 1; //                  Changes        durning
machining and initial

            double timeTotalForToolChanges = toolChangeTotal * toolChangeTime;

            timeTotalForPF010_TO_PP020 = timeFace + timeInternalProfile +
              timeTotalForToolChanges;

            #endregion

            #region Message
            location = 4;
            string message = "";
            message += "***** Data for PF010 to PF 020 *****";
            message += "\nDepth of cut (should be 1mm) = " + depthOfCut;
            message += "\nFace edge length (should be 22.8) = " + lengthOfFace;
            message += "\nExternal Rad of face (should be 354.3) = " + ExternalRadOfFace;
            message += "\nInternal Rad of face (should be 322.5) = " + InternalRadOfFace;
            message += "\nAvgDia of face = " + avgDiaOfFace;
            message += "\nLength of internal profile stage 1 (should be 13.85) = " +
              lengthOfInternalProfile1;
            message += "\nLength of internal profile stage 2 (should be 6.288) = " +
              lengthOfInternalProfile2;
            message += "\n\n ***** Times *****";
            message += "\nTime of face op = " + timeFace;
            message += "\nTime of internal profile op = " + timeInternalProfile;
            message += "\nTool changes total = " + toolChangeTotal;
            message += "\nTime total = " + timeTotalForPF010_TO_PP020;

            //MessageBox.Show(message, "Extraction data for 'machine_PF010_To_PF020()'");

            MachiningDataString += message;
            #endregion
        }
        #region catch
        catch (Exception e)
        {
            string message = "There has been an error in                    machine_PF010_to_PF020()\nLocation: " + location
              + "\nSublocation: "+ SubLocation+"\nSubSubLocation: " + SubSubLocation +                    "\n\n";
            MessageBox.Show(message, "Error");
            writeToFile("log", message, "\n");
            throw new Exception("Error occurred", e);
        }
        #endregion
        return timeTotalForPF010_TO_PP020;
    }
    #endregion
```

A.3 Code to select and sort resources

This function determines the maximum size of the component at the current time, the component size changes with each op therefore needs checking, then it extracts all the resources related to the current operation that have large enough working envolpes, then it sorts them based on reource cost rate.

```
        /// <summary>
    /// <para>This function uses a process ID to select usable machines and orders them in a datatable
        which is returns</para>
    /// <para>****************** Needs checking *******************</para>
    /// <para> Created on 14/09/11 - Last modified on 15/09/11 - last check on 15/09/11</para>
    /// </summary>
    /// <param name="processID"></param>
    /// <param name="supplyChainOption"></param>
    /// <returns></returns>
    private DataTable selectMachine(int processID, string supplyChainOption)
    {
        string sql, message;
        double xAxis = 0, yAxis = 0, zAxis = 0, location = 0, subLocation = 0 ;
        DataTable dtMachine = new DataTable();
        try
        {
            #region NX stuff
            location = 1;
            NXOpen.Assemblies.Component nullAssemblies_Component = null;
            Session theSession = Session.GetSession();
            PartLoadStatus partLoadStatus1;
            theSession.Parts.SetWorkComponent(nullAssemblies_Component, out
              partLoadStatus1);
        Part workPart = theSession.Parts.Work;
        Part displayPart = theSession.Parts.Display;

        Unit nullUnit = null;
        DisplayableObject[] objects1 = new DisplayableObject[1];
        DisplayableObject objects2;
        DisplayableObject objects4;
        IBody[] objects3 = new IBody[1];

        MeasureDistance measureDistance1;
        MeasureLength measureLength1;
        Edge edge1;

        #endregion

        if (supplyChainOption.StartsWith("Forged"))
        {
          #region Forged x,y,z
          NXOpen.Assemblies.Component component2 =
         (NXOpen.Assemblies.Component)workPart.ComponentAssembly.RootComponent.FindObject("COM
         PONENT COC_PF010 1");
          theSession.Parts.SetWorkComponent(component2, out partLoadStatus1);
          location = 2;
          subLocation = 1;
          edge1 = (Edge)component2.FindObject("PROTO#.Features|LINKED_BODY(1)|EDGE                    *        784
REVOLVED(6) [CURVE 2 0] {(301.4999999999999,384.7750869014261,-
         222.1499999999999)(301.4999999999999,0,444.3)(301.4999999999999,-
         384.7750869014259,-222.1500000000002) LINKED_BODY(1)}");
          objects2 = edge1;
          measureDistance1 = workPart.MeasureManager.NewDistance(nullUnit,
         objects2);
          xAxis = ((measureDistance1.Value) * 2) / 1000; // External canister dia
          yAxis = xAxis; // Outer radius of forged component when internal rad =                    400      is
444.3

          // Length = 301.5
          subLocation = 2;
          Array.Resize(ref objects1, 2);
          Face face1 =
         (Face)component2.FindObject("PROTO#.Features|LINKED_BODY(1)|FACE                    795                   {(-
         3,0,0.0000000000001) LINKED_BODY(1)}");
          objects2 = face1;
          Face face2 =            (Face)component2.FindObject("PROTO#.Features|REVOLVED(6)|FACE [CURVE 2 0]");
          objects4 = face2;
          measureDistance1 = workPart.MeasureManager.NewDistance(nullUnit,                        objects2, objects4);
          zAxis = (measureDistance1.Value) / 1000; // Canister length
          subLocation = 2.6;

          //MessageBox.Show("xAxis is(444.3*2): " + xAxis + " YAxis is(444.3*2):" +                   yAxis   +   "   ZAxis
is(301.5):" + zAxis);
          #endregion Forged x,y,z
        }
        else if (supplyChainOption.StartsWith("HIP"))
        {
          #region HIP x,y,z
          NXOpen.Assemblies.Component component1 =
         (NXOpen.Assemblies.Component)workPart.ComponentAssembly.RootComponent.FindObject("COM
         PONENT HIP_02 1");
          location = 3;
          theSession.Parts.SetWorkComponent(component1, out partLoadStatus1);
          workPart = theSession.Parts.Work;
          subLocation = 1;
          // ***** External rad
          edge1 = (Edge)component1.FindObject("PROTO#.Features|REVOLVED(20)|EDGE *                  [CURVE  8
0] * [CURVE 9 0] {(357.17,366.6890123687919,-
         211.7079999999999)(357.17,0,423.416)(357.17,-366.6890123687918,-211.7080000000002)
         REVOLVED(20)}");
          objects2 = edge1;
          measureDistance1 = workPart.MeasureManager.NewDistance(nullUnit,
         objects2);
          double canisterDia = ((measureDistance1.Value) * 2) / 1000; // External                   canister
dia
          xAxis = canisterDia;
          yAxis = canisterDia;

          subLocation = 2;
          // ***** Canister length
          Line line1 =
         (Line)component1.FindObject("PROTO#.Sketches|SKETCH_001|Curve Line18");
          objects1[0] = line1;
          measureLength1 = workPart.MeasureManager.NewLength(nullUnit, objects1);
          zAxis = (measureLength1.Value) / 1000; // Canister length

          subLocation = 3;
          //***** Resets the workpart back to the top assembly
```

```
                theSession.Parts.SetWorkComponent(nullAssemblies_Component, out
               partLoadStatus1);
                partLoadStatus1.Dispose();
                #endregion HIP x,y,z
            }
            else if (supplyChainOption.StartsWith("Blisk"))
            {
                #region Forged x,y,z
                NXOpen.Assemblies.Component component1 =
               (NXOpen.Assemblies.Component)workPart.ComponentAssembly.RootComponent.FindObject("COM
               PONENT Blisk1_COS 1");
                location = 4;
                subLocation = 1;
                theSession.Parts.SetWorkComponent(component1, out partLoadStatus1);
                double cosBaseWidth = 0, cosExternalWidth = 0, cosExternalRadius = 0;
                foreach (Expression expression1 in workPart.Expressions)
                {
                    if (!expression1.Name.StartsWith("p"))
                    {
                        if (expression1.Name.Equals("COS_Base_Width"))
                            cosBaseWidth = expression1.Value;
                        if (expression1.Name.Equals("COS_External_Width"))
                            cosExternalWidth = expression1.Value;
                        if (expression1.Name.Equals("COS_External_Radius"))
                            cosExternalRadius = expression1.Value;
                    }
                }
                xAxis = cosBaseWidth + cosExternalWidth; // Lath bed length therefore the                width of the cos
                zAxis =cosExternalRadius; // the dia of the cos
                yAxis = zAxis; // dia of the cos
                #endregion Forged x,y,z
            }
            else
            {
                MessageBox.Show("Error in selectmachine() the supply chain does not start                with Forged or HIP");
                throw new Exception();
            }
            theSession.Parts.SetWorkComponent(nullAssemblies_Component, out
              partLoadStatus1);
            workPart = theSession.Parts.Work;
            partLoadStatus1.Dispose();
            location = 5;
            sql = "SELECT * FROM Type_Manf_Machines WHERE ID IN "
                + "(SELECT Machine_ID FROM Link_Process_To_Machine WHERE Process_ID = " +                processID
                + ") AND X_Axis > " + xAxis + " AND Y_Axis > " + yAxis + " AND Z_Axis >"                + zAxis + " AND Use
= true "
                + " ORDER BY Cost_Rate ASC, Investment_Cost ASC, Foot_Print_Area_Machine                ASC,
Maintenance_Cost_Annual ASC"; // before order need AND X_Axis > dim AND                Y_Axis > dim AND Z_Axis > dim
            dtMachine = completeSQL(sql);
            if (dtMachine.Rows.Count < 1) // Checks if the sql has returned somthing
            {
                MessageBox.Show(message = "There is no machine avaiable in the database                that can be used to
complete this selectMachine() function.\nThe process ID                is: " + processID + "\nThe last sql was: " + sql +
"\nPlease update the data                base 'Type_Manf_Machines' table.\n\nThanks", "Error");
                writeToFile("log", message, "\n");
                throw new Exception();
            }
        }
        #region catch
        catch (Exception)
        {
            message = "There has been an error in selectMachine() function.\nLocation is:                " + location +
"\nSublocation is: " + subLocation;
            MessageBox.Show(message, "Error");
            writeToFile("log", message, "\n");
            throw new Exception(message);
        }
        #endregion
        return dtMachine;
    }
```

A.4 Code to generate the process time for turning

Below is the code that generates the operation process time for a turning operation. The output is given as a double value and the input parameters are:

- avgDia – the median diameter for the turning cut
- length – the length of the cut in total

- cutType – the type of cut with rough, medium or finish to be inputted as 1, 2, 3 respectively
- machinabilityNumber – the machinability of the material
- speedType – the duration in min of tool life, either 10 for 10 min or 30 as 30 min

```
/// <summary>
/// <para>Calculates turning cut time in min.</para>
/// <para>CutType: 1 = Rough, 2 = Medium, 3 = Finish, Default = Rough.</para>
/// <para>MachinabilityNumber: 1 to 24, Default = 22.</para>
/// <para>SpeedType: 10 = 10 min tool life, 30 = 30 min tool life, Default =
30.</para>
/// </summary>
/// <param name="avgDia"></param>
/// <param name="length"></param>
/// <param name="cutType"></param>
/// <param name="machinabilityNumber"></param>
/// <param name="speedType"></param>
/// <returns></returns>
public double turning(double avgDia, double length, int cutType, int
machinabilityNumber, int speedType)
{
    string speedTypeString, sql;
    double speed, RPM, feedRate, cutTime;

    // Sets the speedTypeString to given value otherwise to default
    if (speedType == 10)
        speedTypeString = "Speed10min";
    else if (speedType == 30)
        speedTypeString = "Speed30min";
    else
    {
        speedTypeString = "Speed30min";
    }

    // Set the default machinability number to 22, if non specified
    if (machinabilityNumber < 1 || machinabilityNumber > 24)
        machinabilityNumber = 22;

    // sets the cuttype default, if non specified
    if (cutType < 1 || cutType > 3)
        cutType = 1;

    //Selects the feed from the database
    sql = "SELECT Feed FROM Data_TurningDocFeed WHERE TurningDoCFeedID = " +
        cutType;
    DataTable dtFeed = completeSQL(sql);

    // selects the speed for the given machinability number and cut type
    sql = "SELECT Speed10min, Speed30min FROM Data_TurningSurfaceSpeed WHERE
        MachinabilityNumber = "
        + machinabilityNumber + " AND DepthOfCutID = " + cutType;
    DataTable dtSpeed = completeSQL(sql);

    // Calculates data for final cut time
    speed = 1000 * Convert.ToDouble(dtSpeed.Rows[0][speedTypeString]);
    RPM = speed / (Math.PI * avgDia);
    feedRate = RPM * Convert.ToDouble(dtFeed.Rows[0]["Feed"]);
    cutTime = length / feedRate;

    return cutTime;
}
```

A.5 Code to generate the batch size

This function generates the batch size for batch the batch operations Pickel_001 and HIP_001.

```
/// <summary>
/// <para>When passed a process name this function will calculate and return the batch size for the process, otherwise it
                     returns a default value of 1</para>
/// </summary>
/// <param name="process"></param>
/// <param name="machineID"></param>
/// <returns></returns>
private int batchSize(string process, int machineID)
{
    messageMain = "Entered batchSize()";
    double location = 0;
    string sql = "SELECT * FROM Type_Manf_Machines WHERE ID = " + machineID + " AND Use = true";
    DataTable dtmachine = completeSQL(sql);

    //MessageBox.Show("Entered batchsize()");
    #region NX stuff
    NXOpen.Assemblies.Component nullAssemblies_Component = null;
    Session theSession = Session.GetSession();
    PartLoadStatus partLoadStatus1;
    theSession.Parts.SetWorkComponent(nullAssemblies_Component, out partLoadStatus1);

    Part workPart = theSession.Parts.Work;
    Part displayPart = theSession.Parts.Display;

    Unit nullUnit = null;
    DisplayableObject[] objects1 = new DisplayableObject[1];
    DisplayableObject objects2;
    IBody[] objects3 = new IBody[1];

    MeasureDistance measureDistance1;
    MeasureLength measureLength1;

    #endregion
    clsMachiningTimeGen machTime = new clsMachiningTimeGen();
    int batchSize = 0;

    sql = "SELECT * FROM test"; // Experimenting purposes
    DataTable dt1 = completeSQL(sql); // Experimenting purposes
    int size = (int)dt1.Rows[0]["BatchSize"]; // Experimenting purposes

    try
    {
        #region Switch
        switch (process)
        {
            case "Pickel_001":
                #region calculation of batch size

                // ***** External rad
                NXOpen.Assemblies.Component component1 =
          (NXOpen.Assemblies.Component)workPart.ComponentAssembly.RootComponent.FindObject("COMPONENT
                     HIP_02 1");
                theSession.Parts.SetWorkComponent(component1, out partLoadStatus1);
                Edge edge1 = (Edge)component1.FindObject("PROTO#.Features|REVOLVED(20)|EDGE * [CURVE 8 0] *
                     [CURVE 9 0] {(357.17,366.6890123687919,-211.7079999999999)(357.17,0,423.416)(357.17,-
                     366.6890123687918,-211.7080000000002) REVOLVED(20)}");
                objects2 = edge1;
                measureDistance1 = workPart.MeasureManager.NewDistance(nullUnit, objects2);
                double canisterDia = ((measureDistance1.Value) * 2) / 1000; // External canister dia in meters

                //***** Resets the workpart back to the top assembly
                theSession.Parts.SetWorkComponent(nullAssemblies_Component, out partLoadStatus1);
                partLoadStatus1.Dispose();

                //batchSize = machTime.pickel_Capacity(canisterDia);
                //MessageBox.Show("pickel cpacity = " + batchSize);

                double pickelEnvolope_X = (double)dtmachine.Rows[0]["X_Axis"];
                double pickelEnvolope_Y = (double)dtmachine.Rows[0]["Y_Axis"];

                double canisterArea = Math.PI * (Math.Pow(canisterDia / 2, 2)); //1.13
                double pickelEnvolope_Area = pickelEnvolope_X * pickelEnvolope_Y; // 16

                double allowanceFactor = 0.9;
                int capacity = (int)Math.Floor((pickelEnvolope_Area / canisterArea) * allowanceFactor);
                batchSize = capacity;
                break;
                #endregion
            case "HIP_001":
                #region calculation of batch size
                int vesselCapacity = 0;
                try
                {
                    location = 1;
                    // ***** External rad
                NXOpen.Assemblies.Component component2 =
        (NXOpen.Assemblies.Component)workPart.ComponentAssembly.RootComponent.FindObject("COMPONENT
                    HIP_02 1");
                theSession.Parts.SetWorkComponent(component2, out partLoadStatus1);
                workPart = theSession.Parts.Work;
                edge1 = (Edge)component2.FindObject("PROTO#.Features|REVOLVED(20)|EDGE * [CURVE 8 0] *
                    [CURVE 9 0] {(357.17,366.6890123687919,-211.7079999999999)(357.17,0,423.416)(357.17,-
                    366.6890123687918,-211.7080000000002) REVOLVED(20)}");
                objects2 = edge1;
                measureDistance1 = workPart.MeasureManager.NewDistance(nullUnit, objects2);
                canisterDia = ((measureDistance1.Value) * 2) / 1000; // External canister dia

                location = 2;
                // ***** Canister length
                Line line1 = (Line)component2.FindObject("PROTO#.Sketches|SKETCH_001|Curve Line18");
                objects1[0] = line1;
                measureLength1 = workPart.MeasureManager.NewLength(nullUnit, objects1);
                double canisterHeight = (measureLength1.Value) / 1000; // Canister length

                location = 3;
                //***** Resets the workpart back to the top assembly
                theSession.Parts.SetWorkComponent(nullAssemblies_Component, out partLoadStatus1);
                partLoadStatus1.Dispose();

                //double pressureRequired = 138000000; // The temperature is material and canister dependant not vessel
                    dependant;  // Not used any more but keep
                //double temperatureRequired = 1173; // The temperature is material and canister dependant not vessel
                    dependant // Not used any more but keep

               // vesselCapacity = machTime.HIP_VesselCapacity(canisterDia, canisterHeight, pressureRequired,
                    temperatureRequired); // Old way of doing it
                double vesselHeight = 0;
                double packingFactor = 0.8;
                vesselHeight = (double)dtmachine.Rows[0]["Z_Axis"];

                vesselCapacity = (int)Math.Floor((vesselHeight / canisterHeight) * packingFactor);
            }
            #region catch
            catch (Exception e)
            {
                string message = "There has been an error in batchSize() - HIP_001 switch statement.\n\nLocation: " +
                    location;
                MessageBox.Show(message, "Error");
                writeToFile("log", message, "\n");
                throw new Exception("Error occurred", e);
            }
            #endregion

            batchSize = vesselCapacity;
            break;
            #endregion
        default:
            batchSize = 1;
            break;
    }
    #endregion
}
#region catch
catch (Exception e)
{
    string message = "There has been an error in batchSize() switch statement\nThe process name that has entered this
                    function is: " + process + "\n\n";
    MessageBox.Show(message, "Error");
    writeToFile("log", message, "\n");
    throw new Exception("Error occurred", e);
}
#endregion
//MessageBox.Show("batch size = " + batchSize + "\nThe process is: " + process);
return batchSize;
}
```

A.6 Code to execute generic data driven dynamic model

Below is the code that executes the generic data driven dynamic model. First the integration code executes a batch file which executes the dynamic model within a Java applet viewer that has all necessary permissions to access the database. It then waits until the dynamic model has completed it optimisation after which the integration code continues.

```
public void runAnylogic()
{
    try
    {
        string name = RunSim_Uni.bat";
        string processName = "appletviewer";
        Process.Start(path + name);
        System.Threading.Thread.Sleep(500); // wait 0.5 seconds

       Process[] processes = Process.GetProcessesByName(processName);// Check for
       bat file
        if (processes.Length == 0 || processes.Length > 1)
        {
       MessageBox.Show("Function runAnylogic(), batProcess not loaded\nNumber of
       processes is: "+ processes.Length, "Error");
            Exception ex = new Exception();
        }
        System.Threading.Thread.Sleep(1000); // wait one second
       string windowName = "Applet Viewer:
       dres_v2/Search_Resource_Capacities$Applet.class"; // window name
        IntPtr hWndPtr = FindWindow(IntPtr.Zero, windowName); // find window
       handle
        ShowWindow(hWndPtr, SW_MAXIMIZE); // maximise window

        System.Threading.Thread.Sleep(500); // Wait 0.5 seconds
        windowName = "Message"; // Window name
        hWndPtr = FindWindow(IntPtr.Zero, windowName); // find window handle
        SendMessage(hWndPtr, WM_SYSCOMMAND, SC_CLOSE, 0); // close window

        int loop = 0;
        bool initialLoop = false, checking = true;

        while (checking == true)
        {
            System.Threading.Thread.Sleep(1000);
            if (initialLoop == true || loop > 5)// Wait 5 second before checking
            {
                initialLoop = true;
                processes = Process.GetProcessesByName(processName);
                System.Threading.Thread.Sleep(1000); // wait one second
                if (processes.Length == 0)
                {
                    break;
                }
            }
            loop++;
        }
    }
    #region catch
    catch (Exception e)
    {
        string message = "There has been an error in runAnylogic()\n\n";
        MessageBox.Show(message, "Error");
        writeToFile("log", message, "\n");
        throw new Exception("Error occurred", e);
    }
    #endregion
}
```

The batch file holds two lines, the first changes the directory that holds the dynamic model and the second executes the dynamic model within a Java applet viewer that contains permissions required by the dyanmic model to access the database.

```
cd C:\.....Insert directory of dynamic model here

start appletviewer.exe -J-Djava.security.policy=allpermissions.txt
DRES_V2_Model_V13.html
```

A.7 Code to collect, calculate and order all data required for the dynamic model

This code collects and calculates if necessary the data necessary for the dynamic model. It then puts it into a table within the database so that the dynamic model can use it. The Kanban equations are highlighted towards the end of the code.

```
        /// <summary>
        /// <para>Populates Experiment_Op_Time_Data, Sim_Delay, Sim_Delay_Resource_Release, Sim_Resource_Allocation, Sim_Resource_Release and Sim_Run tables in DB</para>
        /// </summary>
        private void sendDataToDBSection3()
        {
            double location = 0;
            double sublocation = 0;
            string sql = "" ;
            DataTable dt = new DataTable();
            DataTable dtMachineResourceQuantity = new DataTable();
            DataTable dtMachineResource = new DataTable("Machine_Resource");
            DataTable dtProcessFixture = new DataTable("Process_Fixture");
            int ID_Op_Ex = 0;
            int Kanban = 0;
            double Cycle_Time = 0;
            int batchQuantity = 1;
            int Kanban2 = 0;
            double productionRateSim = (double)ds.Tables["Production_Rate"].Rows[0]["Production_minutes"]; // the number of components per minute
            //displayDataTable(ds.Tables["Op_Time"], "Op_Time");
            foreach (DataRow drTime in ds.Tables["Op_Time"].Rows)
            {
                #region Experiment_Op_Time_Data table population
                try //Try 1 - For Experiment_Op_Time_Data
                {
                    sql = "INSERT INTO Experiment_Op_Time_Data (Experiment_ID, Op_Number, Op_Name, Supply_Chain_ID, Supply_Chain_Option_ID, Method_ID, Process_ID, Machine_ID) VALUES ("
                        + (int)drTime["Experiment_ID"] + " ,"
                        + (int)drTime["Op_Number"] +", \""
                        + drTime["Op_Name"].ToString() +"\", "
                        + (int)drTime["Supply_Chain_ID"] +", "
                        + (int)drTime["Supply_Chain_Option_ID"] +", "
                        + (int)drTime["Method_ID"] + ", "
                        + (int)drTime["Process_ID"] + ", "
                        + (int)drTime["Machine_ID"] +")";
                    executeNonSql(sql);
                }
                #region catch
                catch (Exception e)
                {
                    string message = "There has been an error in section 3 sendDataToDBSection3() try block 1. \nThe SQL statement is: " + sql;
                    MessageBox.Show(message, "Error");
                    writeToFile("log", message, "\n");
                    throw new Exception("Error occurred", e);
                }
                #endregion
                #endregion

                #region Extraction of ID_Op_Ex
                try //Try 1-1 - extract ID_Op_Ex
                {
                    sql = "SELECT ID FROM Experiment_Op_Time_Data WHERE Experiment_ID = " + (int)drTime["Experiment_ID"] + " AND Op_Number = " + (int)drTime["Op_Number"] + " ORDER BY ID ASC";
                    dt = completeSQL(sql);
                    ID_Op_Ex = (int)dt.Rows[0]["ID"];
                }
                #region catch
                catch (Exception e)
                {
                    string message = "There has been an error in section 3 sendDataToDBSection3() try block 1-1 extract ID_Op_Ex. \nThe SQL statement is: " + sql;
                    MessageBox.Show(message, "Error");
                    writeToFile("log", message, "\n");
                    throw new Exception("Error occurred", e);
                }
                #endregion
                #endregion
            #region Extracts the number of resources connected to the machine and puts in to ds table
            try // Try block 1-2
            {
               sql = "SELECT * FROM List_Resources WHERE ID IN "
                  + "(SELECT Resource_ID FROM Link_Machines_To_Resources WHERE ID IN "
                  + "(SELECT Resource_Requirement_ID FROM Type_Manf_Machines WHERE Use = true AND ID = " +
drTime["Machine_ID"].ToString() + " ORDER BY Cost_Rate ASC ))";
               dtMachineResource = completeSQL(sql);
               ds.Tables.Add(dtMachineResource);
            }
            #region catch
            catch (Exception e)
            {
               string message = "There has been an error in section 3 sendDataToDBSection3() try block 1-2 extract Extracts the
number of resources connected to the machine. \nThe SQL statement is: " + sql;
               MessageBox.Show(message, "Error");
               writeToFile("log", message, "\n");
               throw new Exception("Error occurred", e);
            }
            #endregion
            #endregion

            int numResourcesConnectedToMachine = dtMachineResource.Rows.Count; // the number of resources connected to
the chosen machine

            #region Extracts the number of fixtures connected to the process and puts in to ds table
            try
            {
               sql = "SELECT * FROM Type_Fixture WHERE ID IN "
                  + "(SELECT Fixture_ID FROM Link_Process_To_Machine WHERE Process_ID = " +
drTime["Process_ID"].ToString() + " AND Machine_ID = " + drTime["Machine_ID"].ToString() + ")";
               dtProcessFixture = completeSQL(sql);
               ds.Tables.Add(dtProcessFixture);
            }
            #region catch
            catch (Exception e)
            {
               string message = "There has been an error in section 3 sendDataToDBSection3() try block 1-3 Extracts the
number of fixtures connected to the process. \nThe SQL statement is: " + sql;
               MessageBox.Show(message, "Error");
               writeToFile("log", message, "\n");
               throw new Exception("Error occurred", e);
            }
            #endregion
            #endregion

            int numProcessFixture = dtProcessFixture.Rows.Count; // the number of fixtures for process

            int NumberOfResources = 1 + numResourcesConnectedToMachine + numProcessFixture; // total number of
resources for the OP, Machine, resources connected to machine, fixtures connected to process

            #region Sim_Resource table population
            try // For Sim_Resource_Allocation
            {
               location = 0;
               sublocation = 0;
               for (int r = 1; r <= NumberOfResources; r++)// This need to be a foreach loop of the resources required for this OP
               {
                  string Resource_Type = "";
                  int Resource_ID = 0;
                  int Resource_Quantity = 0;
                  if (r == 1)
                  {
                     location = 1;
                     sublocation = 0;
                     Resource_Type = "Machine";
                     Resource_ID = (int)drTime["Machine_ID"];
                     Resource_Quantity = 1;
                  }
                  else if (r > 1 & r <= (numResourcesConnectedToMachine + 1))
                  {
                     location = 2;
                     sublocation = 0;
                     Resource_Type = "Machine_Resource";
                     Resource_ID = (int)dtMachineResource.Rows[r - 2]["ID"];

                     sql = "SELECT * FROM Link_Machines_To_Resources WHERE ID IN "
                        + "(SELECT Resource_Requirement_ID FROM Type_Manf_Machines WHERE Use = true AND ID = " +
drTime["Machine_ID"].ToString() + " )";
                     sublocation = 1;
                     dtMachineResourceQuantity = completeSQL(sql);
                     sublocation = 2;
                     /*RQ = (int)dtMachineResourceQuantity.Rows[0]["Number_of_resources"];
                     sublocation = 3;*/
                     //displayDataTable(dtMachineResourceQuantity, "number of resources for machine resources");
```

```
                    sublocation = 4;

                    Resource_Quantity = Convert.ToInt32(dtMachineResourceQuantity.Rows[0]["Number_of_resources"]);//
stupid program is not recognising the column name
                    //MessageBox.Show(Resource_Quantity.ToString());

                }
                else if (r > (numResourcesConnectedToMachine + 1))
                {
                    location = 3;
                    sublocation = 0;
                    Resource_Type = "Fixture";
                    Resource_ID = (int)dtProcessFixture.Rows[r - (numResourcesConnectedToMachine + 2)]["ID"];
                    Resource_Quantity = 1;
                }

                location = 4;
                sublocation = 0;
                sql = "Insert INTO Sim_Resource (ID_Op_Ex, Resource_Number, Resource_Type, Resource_ID,
Resource_Quantity) VALUES ("
                    + ID_Op_Ex + ", "
                    + r + ", \""
                    + Resource_Type + "\", "
                    + Resource_ID + ", "
                    + Resource_Quantity + ")";
                executeNonSql(sql);
            }
        }
        #region catch
        catch (Exception e)
        {
            string message = "There has been an error in section 3 sendDataToDBSection3() - For Sim_Resource. \nThe last
SQL statement used was: " + sql;
            message += "\nlocation: " + location + "\nSubloaction: " + sublocation;
            MessageBox.Show(message, "Error");
            writeToFile("log", message, "\n");
            throw new Exception("Error occurred", e);
        }
        #endregion
        #endregion

        double OpTime = 0;
        int NumberOfDelays = 3; //**** this is where the number of delays per op can be controled - currently set at 3 for
setup, run, setdown
        for (int delay = 1; delay <= NumberOfDelays; delay++) // ************** Need to populate the for num of loops in the for
block
        {
            // *********************************
            // This is where the code would need to be if i want to have different delay bits to allow release and allocation of
resources
            // The little bit of code below does it for setup and run currently
            // *********************************

            #region Determineation of time for delay
            double timeMode = 0;
            string delayType = "";
            if (delay == 1)
            {
                timeMode = (double)drTime["Op_Setup_Time"];
                OpTime += timeMode;
                delayType = "Setup";
            }
            if (delay == 2)
            {
                timeMode = (double)drTime["Op_Run_Time"];
                OpTime += timeMode;
                delayType = "Run";
            }
            if (delay == 3)
            {
                timeMode = (double)drTime["Op_Setup_Time"];
                OpTime += timeMode;
                delayType = "Set down";
            }

            batchQuantity = (int)drTime["OP_Batch_Size"]; // ************** Need to check this
            double timeMinChange = 0.95; // ************** Need to populate this
            double timeMin = timeMode * timeMinChange;

            double timeMaxChange = 1.05; // ************** Need to populate this
            double timeMax = timeMode * timeMaxChange;
            #endregion

            #region Sim_Delay table population
            try //Try 2 - For Sim_Delay
            {
                    sql = "INSERT INTO Sim_Delay (ID_Op_Ex, Delay_Number, Batch_Quantity, Delay_Time_Mode,
Delay_Time_Min, Delay_Time_Max, Description) VALUES ("
                        + ID_Op_Ex + ", " // ID
                        + delay + ", " // Delay num
                        + batchQuantity + ", " // batch quantity
                        + timeMode + ", " // time mode
                        + timeMin + ", " // time min
                        + timeMax + ", \"" // time max
                        + delayType + "\")";
                    executeNonSql(sql);
                }
                #region catch
                catch (Exception e)
                {
                    string message = "There has been an error in section 3 sendDataToDBSection3() try block 2 - For Sim_Delay.
\nThe SQL statement is: " + sql;
                    MessageBox.Show(message, "Error");
                    writeToFile("log", message, "\n");
                    throw new Exception("Error occurred", e);
                }
                #endregion
                #endregion

                #region Sim_Delay_Resource_Release and sim_delay_resource_allocation tables population
                bool anyResourceToRelease = true; //**************** This need to be determined automaticllay
                if (anyResourceToRelease)
                {
                    location = 0;
                    sublocation = 0;
                    //****
                    // There needs to be a for block here so that more that one resource can be released
                    // Also this is the location where resources can be set here
                    // ****
                    try //Try 3 - For Sim_Delay_Resource_Release and Sim_Delay_Resource_Allocation
                    {
                        location = 1;
                        DataTable dtSimDelay = new DataTable();
                        sql = "SELECT ID FROM Sim_Delay WHERE ID_Op_Ex = " + ID_Op_Ex + " AND Delay_Number = " +
delay;
                        dtSimDelay = completeSQL(sql);
                        sublocation = 1;
                        int ID = Convert.ToInt32(dtSimDelay.Rows[0]["ID"]);
                        sublocation = 1.1;
                        // Sim delay resource allocation
                        for (int r = 1; r <= NumberOfResources; r++)// This need to be a foreach loop of the resources required for
this OP
                        {
                            sublocation = 1.2;
                            // need function here to determine the quantity of resource to apply
                            // resource number and delay ID ( and maybe ID_OP_EX) would be the inputs
                            int rQuantity = resourceQuantity(ID_Op_Ex, r);
                            sublocation = 1.3;
                            sql = "INSERT INTO Sim_Delay_Resource_Allocation (ID_Delay, Resource_Number, Resource_Quantity)
VALUES ("
                                + ID + ", " // ID
                                + r + ", " // Resource number
                                + rQuantity + ")"; // Resource quantity
                            sublocation = 1.4;
                            executeNonSql(sql);
                        }
                        location = 2;
                        // sim delay resource release
                        for (int r = 1; r <= NumberOfResources; r++)// This need to be a foreach loop of the resources required for
this OP
                        {
                            sublocation = 2.1;
                            // need function here to determine if the resource needs releaseing at this point.
                            // the resource number and the delay ID would be the inputs to the function
                            int resourceNumToRelease = (int)dtSimDelay.Rows[0]["ID"];
                            if (resourceNumToRelease > 0)
                            {
                                sql = "INSERT INTO Sim_Delay_Resource_Release (ID_Delay, Resource_Num_To_Release) VALUES
("
                                    + (int)dtSimDelay.Rows[0]["ID"] + ", " // ID
                                    + r + ")"; // Resource num to release
                                executeNonSql(sql);
                            }
                        }
                        location = 3;
                    }
                    #region catch
                    catch (Exception e)
                    {
                        string message = "There has been an error in section 3 sendDataToDBSection3() try block 3 - For
Sim_Delay_Resource_Release. \nThe SQL statement is: " + sql;
                        message += "\nLocation: " + location + "\nSublocation: " + sublocation;
                MessageBox.Show(message, "Error");
                writeToFile("log", message, "\n");
                throw new Exception("Error occurred", e);
            }
            #endregion
        }
        #endregion
    }

    #region Sim_Resource_Allocation
    try //Try 4 - For Sim_Resource_Allocation
    {
        double utilMaxStatic = 0.7; //************** Need to populate this

        int capStatic = (int)(Math.Ceiling(productionRateSim * OpTime) / utilMaxStatic / batchQuantity); // Replaced
BatchQuantity
        if (capStatic < 1)
        {
            capStatic = 1;
        }
        Kanban += capStatic
        Cycle_Time += OpTime;

        for (int r = 1; r <= NumberOfResources; r++)// This need to be a foreach loop of the resources required for this OP
        {
            capStatic = capStatic * resourceQuantity(ID_Op_Ex, r);

            sql = "INSERT INTO Sim_Resource_Allocation (ID_Op_Ex, Resource_Number, Resource_Cap_Dynamic,
Resource_Cap_Static, Resource_Utilisation, Resource_Util_Max_Static) VALUES ("
                + ID_Op_Ex + ", " // ID
                + r + ", " // Resource number
                + 0 + ", " // Cap dynamic iniate to zero
                + capStatic + ", " // Cap static
                + 0 + ", " // Util dynamic, iniate to zero
                + utilMaxStatic + ")"; // Util max static
            executeNonSql(sql);
        }
    }
    #region catch
    catch (Exception e)
    {
        string message = "There has been an error in section 3 sendDataToDBSection3() try block 4 - For
Sim_Resource_Allocation. \nThe last SQL statement used was: " + sql;
        MessageBox.Show(message, "Error");
        writeToFile("log", message, "\n");
        throw new Exception("Error occurred", e);
    }
    #endregion
    #endregion

    #region Sim_Resource_Release table population
    try //Try 5 - For Sim_Resource_Release
    {
        for (int r = 1; r <= NumberOfResources; r++)// This need to be a foreach loop of the resources required for this OP
        {
            sql = "INSERT INTO Sim_Resource_Release (ID_Op_Ex, Resource_Num_To_Release) VALUES ("
                + ID_Op_Ex + ", " // ID
                + r +")"; // Resource num to release
            executeNonSql(sql);
        }
    }
    #region catch
    catch (Exception e)
    {
        string message = "There has been an error in section 3 sendDataToDBSection3() try block 5 - For
Sim_Resource_Release. \nThe SQL statement is: " + sql;
        MessageBox.Show(message, "Error");
        writeToFile("log", message, "\n");
        throw new Exception("Error occurred", e);
    }
    #endregion
    #endregion
}

#region Sim_Run table population
try //Try 6 - Sim Run Kanban value
{
    Kanban2 = (int)Math.Ceiling(Cycle_Time / (1 / productionRateSim));
    //MessageBox.Show("cycle time is: " + Cycle_Time + "\nKanban value2 is: " + Kanban2);
    sql = "UPDATE Sim_Run SET Kanban_Static = " + Kanban +", Kanban_Static_Two = " + Kanban2;
    executeNonSql(sql);
}
#region catch
catch (Exception e)
{
```

```
                string message = "There has been an error in section 3 sendDataToDBSection3() try block 6 - Sim Run Kanban value. \nThe SQL statement is: " + sql;
                MessageBox.Show(message, "Error");
                writeToFile("log", message, "\n");
                throw new Exception("Error occurred", e);
            }
            #endregion
            #endregion
        }
```

Appendix B

This appendix contains data for surface finish and hole tolerance grades and the associated manufacturing methods that can achieve the grades. All the data contained in this appendix section has been collected from public sources.

B.1 Surface finish

Table 7 contains a list of surface finish grades and Table 8 contains a list of manufacturing processes and the associated surface finish grades that they can achieve. The green area represents normally achievable surface grades and the gray areas represent surface grades that can be achieved under the correct circumstances. All this data is contained within the manufacturing database.

Table 7: Surface finish grade with description

Surface finish Grade	Description	
N1	Small	Tight / fine surface finish
N2		
N3		
N4		
N5		
N6		
N7	Mid	
N8		
N9		
N10		
N11		
N12	Large	Loose / Rough surface finish

Table 8: Manufacturing methods associated to achievable surface finish grades

Manufacturing process	Surface finish (Micro m / Grade)											
	50 N12	25 N11	12.5 N10	6.3 N9	3.2 N8	1.6 N7	0.8 N6	0.4 N5	0.2 N4	0.1 N3	0.05 N2	0.025 N1
Sawing	▨	🟩	🟩	🟩	🟩	▨						
Planning		▨	🟩	🟩	🟩	▨	▨					
Drilling			▨	🟩	🟩	▨						
Chemical milling			▨	🟩	🟩	▨						
Electrical discharge machine			▨	▨	🟩	🟩	▨					
Milling		▨	▨	🟩	🟩	🟩	▨	▨				
Broaching				▨	🟩	🟩	▨					
Reaming				▨	🟩	🟩	▨					
Electron Beam				🟩	🟩	🟩	▨	▨				
Laser				🟩	🟩	🟩	▨	▨				
Electro-chemical			▨	▨	🟩	🟩	🟩	🟩	▨	▨		
Boring		▨	▨	🟩	🟩	🟩	🟩	▨	▨	▨	▨	
Turning		▨	▨	🟩	🟩	🟩	🟩	▨	▨	▨	▨	
Grinding				▨	▨	🟩	🟩	🟩	🟩	▨	▨	
Honing						▨	🟩	🟩	🟩	▨	▨	
Electro-polish						▨	🟩	🟩	🟩	▨	▨	▨
Polishing							▨	🟩	🟩	▨	▨	▨
Lapping							▨	🟩	🟩	🟩	▨	▨
Super finishing							▨	▨	🟩	🟩	🟩	▨
Sand casting	▨	🟩	▨									
Hot rolling	▨	🟩	▨									
Forging		▨	🟩	🟩	▨							
Investment casting				▨	🟩	▨	▨					
Extruding			▨	▨	🟩	🟩	▨					
Cold rolling				▨	🟩	🟩	▨	▨				
Drawing				▨	🟩	🟩	▨	▨				
Die casting					▨	🟩	▨					

B.2 Hole tolerance

Table 9 contains a list of hole tolerance grades. A graph of tolerance (mm) against dimension size (mm) for different manufacturing methods and tolerance grades is shown in Figure 66. All this data is contained within the manufacturing database.

Table 9: Hole tolerance grades with description

Hole tolerance grade	Description	
H6	Small	Tight tolerance
H7		
H8		
H9		
H10	Mid	
E9		
D9		
D10	Large	Loose tolerance

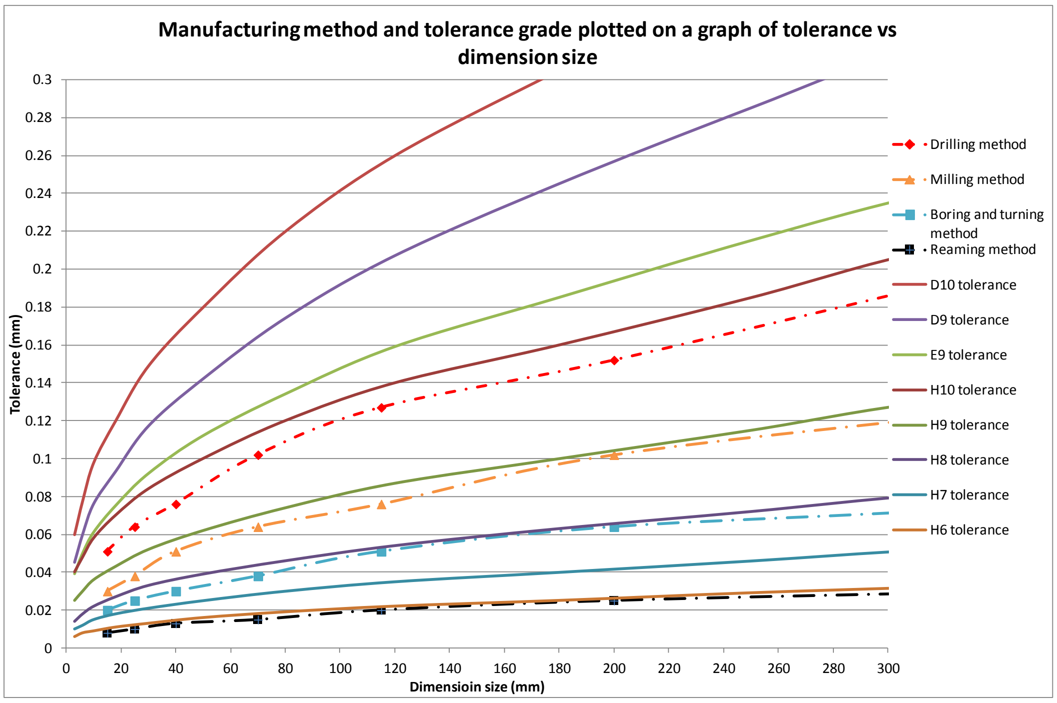


Figure 66: Graph of tolerance against dimension size for different manufacturing methods and tolerance grades

References

Abouel Nasr, E. S. and Kamrani, A. K. (2006). "A new methodology for extracting manufacturing features from CAD system." Computers & Industrial Engineering **51**(3): 389-415.

AbouRizk, S. and Mather, K. (2000). "Simplifying Simulation Modelling through Integration with 3D CAD." Journal of Construction Engineering and Management **126**(6): 475-483.

Andersson, J., Bj, et al. (2012). Framework for ecolabeling using discrete event simulation. Proceedings of the 2012 Symposium on Emerging Applications of M&S in Industry and Academia Symposium. Orlando, Florida, Society for Computer Simulation International**:** 1-8.

Anosike, A. I. and Zhang, D. Z. (2009). "An agent-based approach for integrating manufacturing operations." International Journal of Production Economics **121**(2): 333-352.

Asiedu, Y. and GU, P. (1998). "Product life cycle cost analysis: state of the art review." International Journal of Production Research **36**(4): 883-908.

Askarany, D., Yazdifar, H., et al. (2010). "Supply chain management, activity-based costing and organisational factors." International Journal of Production Economics **127**(2): 238-248.

Aytug, H. and Dogan, C. A. (1998). "A framework and a simulation generator for kanban-controlled manufacturing systems." Computers & Industrial Engineering **34**(2): 337-350.

Babic, B., Nesic, N., et al. (2008). "A review of automated feature recognition with rule-based pattern recognition." Computers in Industry **59**(4): 321-337.

Babic, B. R., Nesic, N., et al. (2011). "Automatic feature recognition using artificial neural networks to integrate design and manufacturing: Review of automatic feature recognition systems." Artificial Intelligence for Engineering Design, Analysis and Manufacturing **25**: 289-304.

Barton, P., Bryan, S., et al. (2004). "Modelling in the economic evaluation of health care: selecting the appropriate approach." Journal of Health Services Research & Policy **9**(2): 110-118.

Beamon, B. M. (1998). "Supply chain design and analysis:: Models and methods." International Journal of Production Economics **55**(3): 281-294.

Beck, U. v. and Nowak, J., W (2000). The merger of discrete event simulation with activity based costing for cost estimation in manufacturing environments.

Birkin, M. and Wu, B. (2012). A Review of Microsimulation and Hybrid Agent-Based Approaches. Agent-Based Models of Geographical Systems. A. J. Heppenstall, A. T. Crooks, L. M. See and M. Batty, Springer Netherlands**:** 51-68.

Bowersox, D. J. and Closs, D. J. (1996). Logistical management: The integrated supply chain process. New York, McGraw-Hill.

Brousseau, E. and Eldukhri, E. (2011). "Recent advances on key technologies for innovative manufacturing." Journal of Intelligent Manufacturing **22**(5): 675-691.

Brown, K. N., McMahon, C. A., et al. (1995). "Features, aka the semantics of a formal language of manufacturing." Research in Engineering Design **7**(3): 151-172.

Brown, N. and Powers, S. (2000). Simulation in a box (a generic reusable maintenance model). Winter Simulation Conference 2000. Orlando USA. **1:** 1050-1056 vol.1051.

Brown, N. A. (2010). Model flexibility: Development of a generic data-driven simulation. Winter Simulation Conference 2010. Baltimore USA**:** 10.

Cao, B., Farr, R., et al. (2005). Data-driven Simulation of the Extended Enterprise. 18th International Conference on Production Research.

Caro, J. J., Möller, J., et al. (2010). "Discrete Event Simulation: The Preferred Technique for Health Economic Evaluations?" Value in Health **13**(8): 1056-1060.

Cassandras, C. G. and Lafortune, S. (2008). Introduction to Discrete Event Systems - Second edition, Springer.

Chen, Y., Mockus, L., et al. (2012). "Simulation-optimization approach to clinical trial supply chain management with demand scenario forecast." Computers &amp; Chemical Engineering **40**(0): 82-96.

Cheng, Z., Qiu, X., et al. (2012). An Agent-Based Artificial Transportation System Framework for H1N1 Transmission Simulation. System Simulation and Scientific Computing. T. Xiao, L. Zhang and S. Ma, Springer Berlin Heidelberg**:** 313-321.

Chu, X., Tang, C., et al. (2011). "Identification of machining features based on available resources of cutting tools." International Journal of Production Research **50**(15): 4141-4157.

Cooper, R. (1990). "Cost classification in unit based activity based manufatcuring cost systems." Journal of Cost Management for the Manufacturing Industry.

Cooper, R. and Kaplan, R., S (1988). "How cost accounting distorts product costs " Management Accounting **69**(10): 20-27.

Cooper, R. and Kaplan, R., S (1988). Measure costs right: Make the right decisions. Harvard Business Review**:** 97-98.

Costa, R. F. d. S., Montevechi, J. A. B., et al. (2010). Discrete Event Simulation and Activity Based Costing to aid the decision making process in a manufacturing cell. The Internacional Workshop on Applied Modeling & Simulation. Rio de Janeiro.

Creighton, D. and Nahavandi, S. (2003). "Application of discrete event simulation for robust system design of a melt facility." Journal of Robotics and computer integrated manufacturing **19**: 469 - 477.

Crooks, A. and Heppenstall, A. (2012). Introduction to Agent-Based Modelling. Agent-Based Models of Geographical Systems. A. J. Heppenstall, A. T. Crooks, L. M. See and M. Batty, Springer Netherlands**:** 85-105.

Curran, R., Gomis, G., et al. (2007). "Integrated digital design for manufacture for reduced life cycle cost." International Journal of Production Economics **109**(1-2): 27-40.

Curran, R., Raghunathan, S., et al. (2004). "Review of aerospace engineering cost modelling: The genetic causal approach." Progress in Aerospace Sciences **40**(8): 487-534.

D'Apice, C., Herty, M., et al. (2010). Modeling, Simulation, and Optimization of Supply Chains: A Continuous Approach, SIAM.

Dean, E., B (1995). Parametric Cost Deployment. Proceedings of the Seventh Symposium on Quality Function Deployment, MI, USA.

Deb, S., para-Castillo, J. R., et al. (2011). "An Integrated and Intelligent Computer-Aided Process Planning Methodology for Machined Rotationally Symmetrical Parts." International Journal of Advanced Manufacturing Systems **13**(1).

Denkena, B., Rudzio, H., et al. (2006). "Methodology for Dimensioning Technological Interfaces of Manufacturing Process Chains." CIRP Annals - Manufacturing Technology **55**(1): 497-500.

Dowlatshahi, s. (1992). "Product design in a concurrent engineering enviroment:an optimization approach." Journal of Production Research **30**(8): 1803-1818.

Epstein, J. M. (2011). Generative Social Science: Studies in Agent-Based Computational Modeling, Princeton University Press.

Farmer, J. D. and Foley, D. (2009). "The economy needs agent-based modelling." Nature **460**(7256): 685-686.

Farrell, R. S. and Simpson, T. W. (2009). "Improving cost effectiveness in an existing product line using component product platforms1." International Journal of Production Research **48**(11): 3299-3317.

Feng, S. C. (2003). "A machining process planning activity model for systems integration." Journal of Intelligent Manufacturing **14**(6): 527-539.

Fischbein, S. A. and Yellig, E. (2011). Why Is It So Hard to Build and Validate Discrete Event Simulation Models of Manufacturing Facilities?

Planning Production and Inventories in the Extended Enterprise. K. G. Kempf, P. Keskinocak and R. Uzsoy, Springer New York. **152:** 271-288.

Forrester, J., W (1961). Industrial Dynamics. Cambridge, Massachusetts, The MIT Press.

Fu, M. W., Ong, S. K., et al. (2003). "An approach to identify design and manufacturing features from a data exchanged part model." Computer-Aided Design **35**(11): 979-993.

Galea, S., Hall, C., et al. (2009). "Social epidemiology and complex system dynamic modelling as applied to health behaviour and drug use research." International Journal of Drug Policy **20**(3): 209-216.

Gao, J., Zheng, D., et al. (2004). "Mathematical representation of feature conversion for CAD/CAM system integration." Journal of Robotics and Computer-Integrated Manufacturing **20**(5): 457-467.

García-Crespo, Á., Ruiz-Mezcua, B., et al. (2011). "A review of conventional and knowledge based systems for machining price quotation." Journal of Intelligent Manufacturing **22**(6): 823-841.

Giachetti, R. E. (1998). "A decision support system for material and manufacturing process selection." Journal of Intelligent Manufacturing **9**(3): 265-276.

Guangru, H. and Xiaoliang, F. (2010). An Intelligent Approach of Obtaining Feasible Machining Processes and Their Selection Priorities for Features Based on Neural Network. Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on.

Gupta, S. K., Chen, Y., et al. (2003). "A system for generating process and material selection advice during embodiment design of mechanical components." Journal of Manufacturing Systems **22**(1): 28-45.

Hammami, R., Frein, Y., et al. (2009). "A strategic-tactical model for the supply chain design in the delocalization context: Mathematical formulation and a case study." International Journal of Production Economics **122**(1): 351-365.

Han, J., Pratt, M., et al. (2000). "Manufacturing feature recognition from solid models: a status report." Robotics and Automation, IEEE Transactions on **16**(6): 782-796.

Huang, B. J., Hsu, P. C., et al. (2010). "System dynamic model and charging control of lead-acid battery for stand-alone solar PV system." Solar Energy **84**(5): 822-830.

Jahangirian, M., Eldabi, T., et al. (2010). "Simulation in manufacturing and business: A review." European Journal of Operational Research **203**(1): 1-13.

Jaya Suteja, T., Prasad KDV, Y., et al. (2013). "A Framework for Life Cycle Cost Estimation of a Product Family at the Early Stage of Product Development." Advanced Materials Research.

Jenab, K. and Liu, D. (2009). "A graph-based model for manufacturing complexity." International Journal of Production Research **48**(11): 3383-3392.

Jinks, S., Scanlan, J. P., et al. (2008). Near Net-shape Manufacturing Costs. 15th ISPE International Conference on Concurrent Engineering. R. Curran, S.-Y. Chou and A. Trappey. Belfast, Springer.

Johnson, G. A. and Malucci, L. (1999). Shift to supply chain reflects more strategic approach. APICS - The performance advantage. **October:** 28-31.

Kang, M., Han, J., et al. (2003). "An approach for interlinking design and process planning." Journal of Materials Processing Technology **139**(1-3): 589-595.

Kendall, K., Mangin, C., et al. (1998). "Discrete event simulation and cost analysis for manufacturing optimization of an automotive LCM component." Composites, Part A **29(A)**(7): 711-720.

Kibira, D. and McLean, C. R. (2007). Generic simulation of automotive assembly for interoperability testing. Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come. Washington D.C., IEEE Press**:** 1035-1043.

Kim, B.-I., Jeong, S., et al. (2009). "A Layout- and Data-Driven Generic Simulation Model for Semiconductor Fabs." Semiconductor Manufacturing, IEEE Transactions on **22**(2): 225-231.

Knoll, J. M. and Heim, J. A. (2000). Ensuring the successful adoption of discrete event simulation in a manufacturing environment. Winter Simulation Conference.

Kumar, M. and Rajotia, S. (2005). "Development of a generative CAPP system for axisymmetric components for a job shop environment." The International Journal of Advanced Manufacturing Technology **27**(1): 136-144.

Law, A., M and Kelton, W., David (1992). Simulation Modeling and Analysis. Singapore, Mc Graw Hill.

Layer, A., Brinke, E. T., et al. (2002). "Recent and future trends in cost estimation." International Journal of Computer Integrated Manufacturing **15**: 499-510.

Lee, T.-R. and Kao, J.-S. (2001). "Application of simulation technique to activity-based costing of agricultural systems: a case study." Agricultural Systems **67**(2): 71-82.

Leitão, P. (2009). "Agent-based distributed manufacturing control: A state-of-the-art survey." Engineering Applications of Artificial Intelligence **22**(7): 979-991.

Li, X., Zhang, C., et al. (2010). "An agent-based approach for integrated process planning and scheduling." Expert Systems with Applications **37**(2): 1256-1264.

Liggett, P., H, R, Trevino , J., et al. (1992). "Activity-based cost management systems in an advanced manufacturing enviroment." Economic and Financial Justification of Advanced Manufacturing Technologies.

Loh, N. L. and Sia, K. Y. (1992). "An overview of hot isostatic pressing." Journal of Materials Processing Technology **30**(1): 45-65.

Marchetta, M. G. and Forradellas, R. Q. (2010). "An artificial intelligence planning approach to manufacturing feature recognition." Computer-Aided Design **42**(3): 248-256.

Marri, H. B., Gunasekaran, A., et al. (1998). "Computer-aided process planning: A state of art." The International Journal of Advanced Manufacturing Technology **14**(4): 261-268.

Marsh, R., Jonik, M., et al. (2010). "Modelling an assembly process using a close coupled generative cost model and a discrete event simulation." International Journal of Computer Integrated Manufacturing **23**(3): 257-269.

McLean, C., Jones, A., et al. (2002). An architecture for a generic data-driven machine shop simulator. **2:** 1108-1116.

Mikko, V., Marko, S., et al. (2007). "Detailed cost modelling: a case study in warehouse logistics." International Journal of Physical Distribution and Logistics Management **37**(3).

Miles, B. L. and Swift, K. (1998). "Design for Manufacture and Assembly." Manufacturing Engineering: 221-224.

Miller, G. R., Cable, J. M., et al. (2012). "Understanding ecohydrological connectivity in savannas: a system dynamics modelling approach." Ecohydrology **5**(2): 200-220.

Min, H. and Zhou, G. (2002). "Supply chain modeling: past, present and future." Computers & Industrial Engineering **43**(1-2): 231-249.

Moorthy, S. (1999). Integrating the CAD model with dynamic simulation: Simulation Data Exchange. Winter Simulation Conference.

Murray-Rust, D., Dendoncker, N., et al. (2011). "Conceptualising the analysis of socio-ecological systems through ecosystem services and agent-based modelling." Journal of Land Use Science **6**(2-3): 83-99.

Nasereddin, M., Mullens, M. A., et al. (2007). "Automated simulator development: A strategy for modelling modular housing production." Journal of Automation in Construction **16**(2): 212-223.

Neugebauer, M., Plonnigs, J., et al. (2004). Automated modelling of LonWorks building automation networks. IEEE International Workshop on Factory Communication Systems.

Newnes, L. B., Mileham, A. R., et al. (2008). "Predicting the whole-life cost of a product at the conceptual design stage." Journal of Engineering Design **19**(2): 99-112.

Niazi, A., Dia, J. S., et al. (2006). "Product cost estimation: Technique classification and methodology review." Journal of Manufacturing Science and Engineering **128**: 563-575.

Norling, E. (2007). Contrasting a System Dynamics Model and an Agent-Based Model of Food Web Evolution. Multi-Agent-Based Simulation VII. L. Antunes and K. Takadama, Springer Berlin Heidelberg. **4442:** 57-68.

Ouelhadj, D. and Petrovic, S. (2009). "A survey of dynamic scheduling in manufacturing systems." Journal of Scheduling **12**(4): 417-431.

Özbayrak, M., Akgün, M., et al. (2004). "Activity-based cost estimation in a push/pull advanced manufacturing system." International Journal of Production Economics **87**(1): 49-65.

Paprotny, I., Zhao, W., et al. (1999). Reducing model creation cycle time by automated conversion of a CAD AHMS layout design. Winter Simulation Conference.

Park, J. and Simpson, T. W. (2007). "Toward an activity-based costing system for product families and product platforms in the early stages of development." International Journal of Production Research **46**(1): 99-130.

Park, S. C. (2003). "Knowledge capturing methodology in process planning." Computer-Aided Design **35**(12): 1109-1117.

Persson, F. and Araldi, M. (2009). "The development of a dynamic supply chain analysis tool—Integration of SCOR and discrete event simulation." International Journal of Production Economics **121**(2): 574-583.

Phanden, R. K., Jain, A., et al. (2011). "Integration of process planning and scheduling: a state-of-the-art review." International Journal of Computer Integrated Manufacturing **24**(6): 517-534.

Pidd, M. (1992). "Guidelines for the design of data driven generic simulators for specific domains." Simulation **59**(4): 237-243.

Pidd, M. (2009). Tools for thinking: modelling in management science. Chichester, John Wiley.

Pidd, M. and Carvalho, A. (2006). "Simulation software: not the same yesterday, today or forever." Journal of simulation **1**(1).

Qudrat-Ullah, H. and Seong, B. S. (2010). "How to do structural validity of a system dynamics type simulation model: The case of an energy policy model." Energy Policy **38**(5): 2216-2224.

Randell, L. G. and Bolmsjo, G. S. (2001). Database driven factory simulation: a proof-of-concept demonstrator. Winter Simulation Conference.

Robinson, S. (2004). Simulation: The practice of model development and use. Chichester, John Wiley and Sons, Ltd.

Robinson, S., Brooks, R., et al. (2010). Conceptual Modeling for Discrete-Event Simulation, CRC Press, Inc.

Rolls-Royce Plc (2010). Annual report 2010 - Teamwork and Technology. London, Rolls-Royce Group Plc.

Rush, C. and Roy, R. (2000). Analysis of cost estimation processes used within a concurrent engineering environment throughout a product life cycle. Seventh ISPE International Conference on Concurrent Engineering: Research and Application, Lyon, France, Technomic.

Rush, C. and Roy, R. (2001). "Expert judgement in cost estimating: modelling the reasoning process." Concurrent Engineering Research Application **9**(4).

Sajadi, S., Seyed Esfahani, M., et al. (2011). "Production control in a failure-prone manufacturing network using discrete event simulation and automated response surface methodology." The International Journal of Advanced Manufacturing Technology **53**(1): 35-46.

Salehi, M. and Bahreininejad, A. (2011). "Optimization process planning using hybrid genetic algorithm and intelligent search for job shop machining." Journal of Intelligent Manufacturing **22**(4): 643-652.

Savory, P. and Williams, R. (2010). "Estimation of cellular manufacturing cost components using simulation and activity-based costing." Journal of Industrial Engineering and Management **3**(1).

Savory, P. A., Williams, R. E., et al. (2001). "Combining Activity-Based Costing with the Simulation of a Cellular Manufacturing System." Journal of Design and Manufacturing Automation **1**(3): 221-229.

Scanlan, J., Rao, A., et al. (2006). "DATUM project: cost estimating environment for support of aerospace design decision making." Journal of Aircraft **43**(4): 1022-1028.

Shah, J. J. (1991). "Conceptual development of form features and feature modelers." Research in Engineering Design **2**: 93-108.

Sharda, B. and Bury, S. J. (2008). A discrete event simulation model for reliability modeling of a chemical plant. Simulation Conference, 2008. WSC 2008. Winter.

Sokolowski, J. A. and Banks, C. M. (2011). Principles of Modeling and Simulation: A Multidisciplinary Approach, John Wiley & Sons.

Son, Y. J. and Wysk, R. A. (2001). "Automatic simulation model generation for simulation-based, real-time shop floor control." Journal of Computers in Industry **45**(3): 291-308.

Son, Y. J., Wysk, R. A., et al. (2002). "Simulation-based shop floor control: formal model, model generation and control interface." Journal of IIE Transactions **35**: 20.

Spedding, T. A. and Sun, G. Q. (1999). "Application of discrete event simulation to the activity based costing of manufacturing systems." International Journal of Production Economics **58**(3): 289-301.

Srikantappa, A., B and Crawford, R., H (1994). "Automatic part coding based on inter-feature relationship." 215-237.

Stewart, R. D., Wyskida, R. M., et al., Eds. (1995). Cost Estimator's Reference Manual. Johan Wiley & sons Inc. New York.

Taiichi, O. (1988). Toyota production system - beyond large scale production, Productivity Press.

Tammineni, S. V. (2007). Designer driven cost modelling Doctor of Philosophy Ph.D., University of Southampton.

Tammineni, S. V., Rao, A. R., et al. (2009). "A knowledge-based system for cost modelling of aircraft gas turbines." Journal of Engineering Design **20**(3): 289 - 305.

Tammineni, S. V., Scanlan, J. P., et al. (2007). Knowledge based system for cost modelling. 7th AIAA Aviation Technology, Integration and Operations Conference (ATIO), Belfast, Northern Ireland, AIAA.

Tannock, J., Cao, B., et al. (2007). "Data-driven simulation of the supply-chain--Insights from the aerospace sector." International Journal of Production Economics **110**(1-2): 70-84.

Taylor, I. (1998). Cost engineering: A feature based approach. 85th Meeting of the AGARD Structures and Material Panel, Aalborg,Denmark.

Tsai, W.-H., Shen, Y.-S., et al. (2012). "Integrating information about the cost of carbon through activity-based costing." Journal of Cleaner Production **36**(0): 102-111.

Tse, M. and Gong, M. (2009). "Recognition of idle resources in time-driven activity-based costing and resource consumption accounting models." Journal of applied management accounting research **7**(2): 13.

Turner, R., Madachy, R., et al. (2012). Modeling kanban processes in systems engineering. Software and System Process (ICSSP), 2012 International Conference on.

Ulrich, K., T and Pearson, S., A (1993). Does product design really determine 80% of manufacturing cost?, Cambridge, MA : Alfred P. Sloan School of Management, Massachusetts Institute of Technology. .

Vanguard Software^{TM} Corporation. (2011). "Vanguard Software." Retrieved 1st December 2011, from www.vanguardsw.com.

Venkateswaran, J., Young-Jun, S., et al. (2004). Hierarchical production planning using a hybrid system dynamic-discrete event simulation architecture. Winter Simulation Conference.

Verma, A. K. and Rajotia, S. (2010). "A review of machining feature recognition methodologies." International Journal of Computer Integrated Manufacturing **23**(4): 353-368.

Wainer, G. A. (2009). Discrete-Event Modeling and Simulation: A Practitioner's Approach, CRC Press.

Wainer, G. A. and Mosterman, P. J. (2011). Discrete event modeling and simulation - Theory and applications, Taylor and Francis Group.

Wang, J., Zhang, H., et al. (2012). "Manufacturing Knowledge Modeling Based on Artificial Neural Network for Intelligent CAPP." Applied Mechanics and Materials **127**.

Wang, P., Du, F., et al. (2010). "The Choice of Cost Drivers in Activity-Based Costing: Application at a Chinese Oil Well Cementing Company." International Journal of Management Reviews **27**(2).

Wei-jun, H. and Yu-jin, H. (2011). Hybrid evolutionary multi-objective approaches to process planning global optimization for complex parts. Industrial Engineering and Engineering Management (IE&EM), 2011 IEEE 18Th International Conference on.

Wierda, L., S (1991). "Linking design, process planning and cost information by feature-based modelling." Journal of Engineering Design **2**(1): 3-19.

Winz, I., Brierley, G., et al. (2009). "The Use of System Dynamics Simulation in Water Resources Management." Water Resources Management **23**(7): 1301-1323.

Wright, T. P. (1936). "Factors affecting the cost of airplanes." Journal of Aeronautical Sciences **3**: 122-128.

Wy, J., Jeong, S., et al. (2011). "A data-driven generic simulation model for logistics-embedded assembly manufacturing lines." Computers & Industrial Engineering **60**(1): 138-147.

Xu, X., Wang, L., et al. (2010). "Computer-aided process planning – A critical review of recent developments and future trends." International Journal of Computer Integrated Manufacturing **24**(1): 1-31.

Yee Mey, G., Newnes, L. B., et al. (2010). "Uncertainty in Through-Life Costing - Review and Perspectives." Engineering Management, IEEE Transactions on **57**(4): 689-701.

Younossi, O., Arena, M., V, et al. (2002). Military Jet Engine Acquisition: Technology Basics and Cost-Estimating Methodology. U. S. A. Force, RAND**:** 153.

Yu, T.-T. (2008). The Development of a Hybrid Simulation Modelling Approach Based on Agents and Discrete-Event Modelling. PhD Thesis, University of Southampton.

Zeigler, B. P., Kim, D., et al. (1999). Distributed supply chain simulation in a DEVS/CORBA execution environment. Proceedings of the 31st conference on Winter simulation: Simulation---a bridge to the future - Volume 2. Phoenix, Arizona, United States, ACM**:** 1333-1340.

Zhan, H., Zhao, W., et al. (2000). "Manufacturing turbine blisks." Aircraft Engineering and Aerospace Technology: An International Journal **72**(3): 247-252.

Zhou, X., Qiu, Y., et al. (2007). "A feasible approach to the integration of CAD and CAPP." Computer-Aided Design **39**(4): 324-338.