The Design of Reconfigurable Manufacturing Systems for Product Mass Customisation

J. Padayachee, G. Bright
Discipline of Mechanical Engineering
University of KwaZulu-Natal, South Africa

Abstract
The concept of Reconfigurable Manufacturing Systems (RMSs) was formulated due to the global necessity for production systems that are able to economically evolve according to changes in markets and products. Technologies and design methods are under development to enable RMSs to exhibit transformable system layouts, reconfigurable processes, cells and machines. This paper presents a framework for a software system that will be used to optimally design RMSs for high variety and mass customisation manufacturing. The framework uses part family information to automatically generate candidate RMS configurations. The framework implements a Genetic Algorithm for the optimisation of randomly generated solutions and the evolution engine incorporates a discrete system simulator for the evaluation stage of the algorithm. The simulator is based on the DEVS, (Discrete Event System Specification), formalism and is used to assess the dynamic behaviour of a candidate configuration in response to a changing production schedule and customer configured products.

Keywords

1 INTRODUCTION
Mass Customisation Manufacturing (MCM) is an emerging competitive strategy in the global manufacturing sector [1]. MCM can be defined as the production of customised goods at prices and quantities similar to that of mass production. The realisation of MCM requires the evolution of production systems toward market sensitivity and responsiveness to changes and variations in products. Reconfigurable Manufacturing Systems (RMSs) are an emerging paradigm, which through mechanisms of machine and process reconfigurations promise to economically provide the functionality necessary for Mass Customisation Manufacturing (MCM). The following is a definition of RMSs offered by Koren [2]: "A Reconfigurable Manufacturing System (RMS) is a system that combines the advantages of Dedicated Manufacturing Systems (DMS) and Flexible Manufacturing Systems (FMS) by designing it at the outset for rapid change in structure, as well as in its machines and controls, in order to quickly adjust production capacity and functionality in response to market or product changes."

A present challenge in the design of RMSs for MCM is designing a system such that the mechanisms of reconfiguration used to adjust manufacturing functions:
- least hinder the production rate;
- do not cause buffer overflows, i.e. unmanageable spikes in Work In Process (WIP);
- do not lead to prolonged machine idle times at downstream processes;
- do not lead to late delivery of products (unreasonable make span and tardiness).

Shop floor scheduling algorithms, such as those developed by Walker et al [3] or Gershwin et al [4] may be used to implement flow control, which will reduce the reconfiguration frequency of machines and thereby diminish the time delay effects of reconfigurations. The optimal design of a RMS is necessary to ensure that any suitable scheduling algorithm is able to maintain stability in a manufacturing system more easily and fully exploit the production capacity of the available equipment. Manufacturing system design for RMSs includes equipment selection, the physical arrangement of equipment into cells, the allocation of tasks to machines (creation of operational clusters), the selection of machine fixtures and the creation of material transportation paths. The design of a complex system such as a manufacturing system is achieved by identifying well defined subsystems that may be designed individually. This paper outlines a methodology for the automated design of manufacturing subsystems in RMSs based on a customisable part family. Section 2 of this paper proceeds with a presentation of relevant literature on the design of RMSs. Section 3 presents the formulation of the design task. Section 4 presents a solution representation method for the optimisation of a design. Section 5 presents a method for simulating the performance of the design under a changing production schedule. Section 6 presents
an overview of the design system and demonstrates how individual tools will be integrated to complete the system. Section 7 concludes with a discussion on the implementation of the design system in software.

2 RELATED LITERATURE

Koren and and Shpitalni [5] presented a method for the design of a RMS. The method extends to the design of a RMS subsystem based on a part family of automotive cylinder heads. The first step in the method is a calculation of the number of machines required to complete an operation. This is based purely on known cycle times for each feature creation operation. The number of machines N, may be arranged in a variety of different configurations. Koren’s method then proceeds with the manual population of a space of potential configurations for the N machines. The selection of an optimal design configuration then proceeds by eliminating those solutions that do not meet the required production rate. Design solutions that meet the required daily production rate are then rated according to system throughput with machine reliability less than 100%, investment cost, scalability and floor space. Koren and Shpitalni did not consider the creation of a RMS for MCM where manufacturing cycle times vary according to the customisation of a product. Neither did the authors consider the design of a RMS, where the configuration of the system actively changes in response to a changing production schedule.

Abdi [6] proposed the use of the Analytic Hierarchy Process, (AHP) for the design of a suitable RMS. The AHP requires a set of RMS design alternatives and a set of design criteria as inputs to the algorithm. The algorithm merely assists the designer in ranking the suitability of the various alternative designs based on the design objectives. Abdi listed reconfigurability, cost, quality and reliability as the design objectives. The method presented by Abdi assumes that the designer already has multiple viable designs already available for ranking. Moreover, the author did not consider a MCM implementation of a RMS.

Tang et al [7] proposed the use of a Genetic Algorithm to design a Multiple Part Line (MPL) for RMSs. A MPL was defined as a line that consists of several serial stages with a finite size buffer between every two stages. Each stage consists of identical machines in parallel that perform the same set of tasks. The design of the MPL was based on the production of a given part family. The objective of the GA was to optimally allocate machines and tasks to the various stages of the MPL. The optimisation objective was to minimize the ratio of the total investment in machines and buffers to the total throughput of all parts. This research did not extend to the use of RMSs for MCM.

3 FORMULATING THE DESIGN TASK

3.1 Product Platform Decomposition

The first step in formulating the design task is to consider each customisable part that constitutes the product platform. Symbolically the product platform can be described by a set of parts:

\[ \text{PRODUCT} = \{P_n\} \]

Each part will be manufactured in a machine cell and parts may share a cell. The design procedures from hence forth considers the optimal formation of machine cells and the allocation of parts to cells. Each part will have a set of features that completely describes the part. Each feature \( F_{mk} \) will have associated with it a customisation function \( C_{mk}(r) \) with \( r \) being a variable in the customisation domain \( D_{mk} \):

\[ P_m = \{F_{mk}\} \]

3.2 Selection of Machinery

Equipment is selected for the design of a manufacturing cell based on the clustering of parts into that cell. Machinery must be made available for the creation of each feature \( F_{mk} \) in each part \( P_m \). The selection of the best machine and setup for the creation of each feature \( F_{mk} \) is usually a complex and specialised task that is difficult to treat in a generic manner. The automation of the design process requires that a design engineer identify the best machine and configuration for a task as an input. The design system can then use the engineer’s selection of individual machines to generate a manufacturing system configuration. The information pertaining to the creation of feature \( F_{mk} \) on a part, using machine type \( M_i \) in configuration (setup) \( j \) is encapsulated in a 5-tuple as follows:

\[ F_{mk} = (C_{mk}(r), D_{mk}, M_i, tool, fixture, program) \]

The tool, fixture and program names are the entities that constitute configuration \( j \).

3.3 Operation Precidence

The arrangement of machines in a manufacturing cell is dictated by the precedence in which features will be created. Moreover, it is desirable that features and parts be grouped into Operational Clusters (OCs) to be created by the same machine. Features that are grouped onto a machine must have equal feature creation precedence to ensure unidirectional flow within a cell. Feature creation precedence is therefore a necessary input to the design system.
For part $P_m$ the feature creation precedence may be written as an ordered set:

$$P_m = \{(F_{mk}, F_{mk+1}), F_{mk+2}, \ldots, F_{mk+k}\} \quad (4)$$

Subgroups in the above set are unordered, this indicates equal precedence. The subscript $k$ is a number that indicates manufacturing precedence of features.

### 3.4 The RMS Design Task

The RMS design task is to use the information stored in the set $\{P_1, P_2, \ldots, P_m\}$ to:

- group parts into cells;
- ensure unidirectional flow within a cell at a minimal cost of additional machines;
- optimise the cell formation such that the process of reconfiguration requires minimal time;
- optimise the cell formation such that the reconfiguration frequency for a forecast market demand is minimised;
- reduce inter-cell movement of parts;
- create material routing paths between machines;
- specify buffer sizes between machines.

### 4 GA ENCODING AND OPERATORS

#### 4.1 Solution Encoding

Traditionally production flow analysis or the Rank Order Cluster Algorithm would be used to create cells in a manufacturing system, however neither of these algorithms consider reconfiguration time or frequency during the cell formation process. Both these techniques used a binary adjacency matrix to encode a solution and block diagonals within this matrix indicate cellular groupings of machines.

The framework proposed in this paper uses a Genetic Algorithm combined with discrete system simulation to create cells for a RMS. For the purpose of a genetic algorithm, a solution is more efficiently encoded using a tree structure. Each branch in the structure, originating from the root, represents a manufacturing cell. Nodes in the branch represent parts that have been grouped into that manufacturing cell, the order in which the nodes are presented is irrelevant. The grouping of parts into cells, encoded in a tree structure is illustrated in Figure 1.

Based on this encoding method, the size of the solution space is calculated from:

$$z = \sum_{i=1}^{i=p(n)} \frac{nl!}{(1!)^{n_1}(2!)^{n_2} \ldots (n!)^{n_n} \lambda_1 \lambda_2 \ldots \lambda_n} \quad (5)$$

Where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are integer partition indices, these indices are generated by an integer partitioning algorithm; $p(n)$ is the number of sets of indices that the algorithm will generate. The total number of partitions that an integer partitioning algorithm will generate for $n$ parts is calculated by:

$$p(n) = \sum_{m=1}^{n} p_m^n \quad (7)$$

Where $p_m^n$ is calculated recursively:

$$p_m^n = p_m^1 + p_m^2 + \ldots + p_m^{n-1} \quad (8)$$

And:

$$p_m^1 = p_m^n = 1 \quad (9)$$

#### 4.2 Genetic Crossover

Two types of Genetic Crossover are proposed. The first type of crossover is performed by randomly selecting a branch from the first parent $(R_1)$ and inserting it into the second parent $(R_2)$, indicated by the black shaded nodes. Duplicate part allocations are deleted, indicated by the grey shaded nodes in Figure 2. The second type of crossover is performed by selecting a branch from the first parent $(R_1)$ and appending it to a branch of $(R_2)$, and duplicate part allocations are deleted.

#### 4.3 Genetic Mutation

Two types of Genetic Mutation are proposed. The first involves the movement of a node from one branch to another branch, the second type of mutation involves the creation of a new branch by randomly selecting and splitting a branch in the parent chromosome. Consider the mutation of solution $R_4$, into two new solutions as illustrated in Figure 3.

**Figure 1** - Tree Structure Encoding

**Figure 2** - Crossover: Type 1 (Left), Type 2 (Right)

**Figure 3** - Mutation: Type 1 (Left), Type 2 (Right)
He DEVS (Discrete Event System Specification) formalism satisfies the modularity criteria.

The DEVS formalism is modular and hierarchal, consisting consists of Atomic Models and Coupled Models [8]. The modularity provided by Atomic Models makes it possible for these models to be coupled in varying configurations. An atomic DEVS model will be used to model individual units of manufacturing equipment. An Atomic DEVS model $M$ is a 7-tuple defined by:

$$ M = \langle X, Y, S, d_{ext}, d_{int}, \lambda, ta \rangle $$  

Where:

- $X$: is a set of input events;
- $Y$: is a set of output events;
- $S$: is a set of sequential states;
- $d_{ext}$: is a set of subcomponent names;
- $d_{int}$: a set of sequential states;
- $\lambda$: $S \rightarrow Y$, an internal transition function;
- $ta$: $S \rightarrow Real$, the time advance function.

A Coupled Model is an assembly of several Atomic Models or smaller Coupled Models. The process of coupling several smaller models is used to build complex systems such as models of manufacturing cells and complete factories from elementary equipment models. A Coupled DEVS model $N$, is an 8-tuple defined by:

$$ N = \langle X, Y, D, \{M\}, C_{xx}, C_{yx}, C_{yy}, Select \rangle $$  

Where:

- $X$: a set of input events;
- $Y$: a set of output events;
- $D$: is a set of subcomponent names;
- $\{M\}$: the set of sub-components where for each $j \in D, M_j$ can be either an atomic DEVS model or a smaller coupled DEVS model;
- $C_{xx} \subseteq X \times UX_j$: is the set of external input couplings;
- $C_{yx} \subseteq UX_i \times Y$: is the set of external output couplings;
- $C_{yy} \subseteq UX_i \times UX_j$: is the set of internal couplings;
- $Select$: $2^D \rightarrow D$: is the tie-breaking selector.

5 SIMULATING A SOLUTION

A candidate solution is to be evaluated and optimised within the framework of a Genetic Algorithm. Because the RMS solution is to be implemented for MCM, the performance of the solution must be investigated under dynamic conditions. This investigation requires discrete system simulation. The choice of a discrete simulation modelling formalism must satisfy the necessity for a modular modelling approach with a structured, hierarchical method of assembling and closing combined models. A formalism like this is necessary because candidate solutions will be continuously modified within the framework of the Genetic Algorithm. The DEVS (Discrete Event System Specification) formalism satisfies the modularity criteria.

6 FITNESS EVALUATION

6.1 Integrated Simulation and GA

An overview of the manufacturing system design and optimisation process is illustrated in Figure 2. Initially a generation of candidate solutions is created in the Configuration Generator by randomly populating multiple solution trees from the set of parts $\{P_1, P_2...P_m\}$. Coupled DEVS models, which represent manufacturing cells, are assembled from
a library of precompiled Atomic models according to the Phenotype interpretation of a candidate solution. Candidate solutions are then simulated using a benchmark production schedule. Simulation results are then used to evaluate the fitness of a candidate solution. Multiple candidate solutions are then selected in Genotype format, according to their performance for the breeding of a subsequent, superior generation of solutions using the operators described in Section 4.2 and 4.3.

The benchmark simulation schedule is a worst case production schedule that is formulated by selecting the the extreme values of each customisable and optional feature in the part family. All candidate solutions are evaluated against the benchmark schedule for an objective comparison of their performance under a changing production schedule. The aim of creating the worst case schedule is to force the candidate manufacturing system to reconfigure each time there is a change in the product configuration.

6.2 Fitness Evaluation

The Genetic Algorithm that will be implemented as part of the design system will be a multi-objective algorithm (NSGA II). The fitness evaluation will be based on the cost of implementing a solution and the Reconfiguration Sensitivity of a solution. The cost of implementing a solution is a summation of machinery costs ($CM_i$), labour costs ($CL_i$) and material handling costs ($CH_i$). Labour costs are directly related to the number of machines of type $N_i$ required by the RMS solution while material handling costs are directly related to the number of inter-cell moves $N_p$ necessitated by the cell formation in the RMS:

$$\text{COST} = \sum_{i=1}^{j} N_i (CM_i + CL_i) + \sum_{p=1}^{q} N_p CH$$

The second fitness criterion, which is Reconfiguration Sensitivity, requires simulation. Reconfiguration Sensitivity may be defined as the sensitivity of a manufacturing system to the adverse effects of reconfigurations. These adverse effects include drastic increases in buffer levels, prolonged machine idle time and late delivery of components. All of these adverse effects are directly related to high WIP levels in a system for prolonged periods of time. Reconfiguration Sensitivity is therefore mathematically defined as:

$$CS_i = \sum_{t=t_i}^{t_s} B_i(t) - B_i(t_r - t_s)$$

$$RS = \sum_{i=1}^{n} CS_i$$

Where:
- $CS_i$ is Cellular Reconfiguration Sensitivity;
- $RS$: is System Reconfiguration Sensitivity;
- $B_i$: is the level of a buffer that feeds the manufacturing cell;
- $t$: is the time at which the buffer level changed to $B_i$;
- $t_r$: is the time elapsed since the last change in buffer level;
- $t_s$: is the time at which a reconfiguration of the system was started;
- $t_e$: is the time at which the buffer level $B_e$ stabilises to level $B_o$ after reconfiguration and ramp-up.

Mathematically, Cellular Reconfiguration Sensitivity is effectively the increased area under a buffer level versus time plot, like the one illustrated in the example of Figure 3.

![Buffer Level vs Time](image)

**Figure 5 - Example: Response of Candidate Manufacturing Systems to Reconfiguration.**

System Reconfiguration Sensitivity is effectively a measure of the elevation in WIP over the reconfiguration and system ramp-up period. Factors that will influence the sensitivity of a solution include the time required for new configurations to be instantiated in the system. Some candidate solutions may require more configuration changes than others to begin processing a new custom product. These candidates will experience longer reconfiguration delays. Other factors will include the ability of some candidates to manufacture certain product customisations faster than others.
7 CONCLUSIONS
The RMS design system in this paper requires significant software development for the full implementation of the methods presented in Sections 3 to 6. Significant components of the software development include the creation of:

- a graphical user interface for the input of the design task;
- a library of atomic DEVS models for various types of manufacturing equipment;
- an automatic RMS design generator;
- the integration of the RMS design system with a DEVS simulation engine,
- the integration of the RMS design system with an evolutionary computation platform.

The parallel DEVS simulation engine developed in C++ by Nutaro [9] has been identified for implementation in the RMS design system. The ParadisEO-MOEO evolutionary computation platform developed in C++ by Liefooghe et al. [10] has been identified as a suitable platform for the optimisation of RMS Designs. This platform supports the Non-Dominated Sorting Genetic Algorithm (NSGA) and NSGA II, which will be the first algorithms that will be tested in the design system. The platform also supports other evolutionary computation algorithms such as the Strength Pareto Evolutionary Algorithm (SPEA2), Indicator Based Evolutionary Algorithms (IBEA) and the Steady Elimination Evolutionary Algorithm (SEEA) [11]. The comparative testing of these algorithms also constitutes future work.

8 ACKNOWLEDGEMENTS
The authors would like to gratefully thank TIA, DST and UKZN for the provision of project funding and resources during this research.

9 REFERENCES

10 BIOGRAPHY
Jared Padaychee holds a BSc and MSc Engineering degree (UKZN). He is currently involved in reconfigurable manufacturing research at a PhD level at UKZN.

Glen Bright holds BSc, MSc, MBA and PhD Engineering degrees (UKZN). He is the current professor of Mechatronics and Robotics at UKZN.