INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 05 MELBOURNE, AUGUST 15 – 18, 2005

IDENTIFYING AND EVALUATING PARALLEL DESIGN ACTIVITIES USING THE DESIGN STRUCTURE MATRIX

R.I. Whitfield, A.H.B. Duffy, L Kortabarria Gartzia-Etxabe

Abstract

This paper describes an approach based upon the Design Structure Matrix (DSM) for identifying, evaluating and optimising one aspect of CE: activity parallelism. Concurrent Engineering (CE) has placed emphasis on the management of the product development process and one of its major benefits is the reduction in lead-time and product cost [1]. One approach that CE promotes for the reduction of lead-time is the simultaneous enactment of activities otherwise known as Simultaneous Engineering. Whilst activity parallelism may contribute to the reduction in lead-time and product cost, the effect of iteration is also recognised as a contributing factor on lead-time, and hence was also combined within the investigation. The paper describes how parallel activities may be identified within the DSM, before detailing how a process may be evaluated with respect to parallelism and iteration using the DSM. An optimisation algorithm is then utilised to establish a near-optimal sequence for the activities with respect to parallelism and iteration. DSM-based processes from previously published research are used to describe the development of the approach.

Keywords: Concurrent engineering, parallelism, iteration, Design Structure Matrix, digraph, optimisation.

1 Introduction

Yassine and Braha [2] describe the overall CE philosophy as: "resting on a single, but powerful, principle that promotes the incorporation of downstream concerns into the upstream phases of the development process." The approaches used for implementing this CE philosophy focus towards: "the timely availability of critical design information to all development participants." Yassine and Braha identified four principles as underpinning successful CE: iteration, overlapping, decomposition, and stability. The first three principles are regarded as being static in nature and are applied with respect to the planning of the product development process. Iteration relates to the amount of potential rework that would be created as a result of either the design failing to meet established criteria, or new information being obtained following a prior iteration [3]. The management of iteration aims to ensure that unnecessary activity is not undertaken as a result of the necessary iteration. Yassine and Braha consider overlapping as being synonymous with parallelism and consider it with reference to the development stages of the product development process and not of individual activities. Improvements in lead-time may be realised through the manipulation of the overlapping which is affected by the relationships between the activities within the product development process. Yassine and Braha manage communication statically within the planning stage by investigating the normal paths that communication occurs between team members, and using decomposition to structure sub-teams to facilitate this communication. The stability principle defines the state of the system with respect to the convergence towards a particular "equilibrium state" and refers to the bounding associated with the number and dynamic creation of design problems during the lifetime of the process.

Loch and Terwiesch [4] identified four managerial issues for CE: task definition, time concurrence, information concurrence, and organisational structure as well as defining CE as: "integrating the new product development process to allow participants making upstream decisions to consider downstream and external requirements." The task definition and time concurrence issues are similar in nature to the decomposition and overlapping principles identified by Yassine and Braha and represent two of the fundamental research areas for CE [1, 2, 5-10]. Loch and Terwiesch considered co-ordination as being essential in ensuring an effective exchange of information within the concurrent product development process. The need for co-ordination to facilitate CE was an issue that was previously identified by Duffy *et al.* [11] who stated that: "while the primary objective of concurrent engineering would seem to be directed at considering aspects of design simultaneously, design co-ordination provides the means of integrating and controlling disparate activities, i.e. design co-ordination is a vehicle for the realisation of concurrent engineering."

The Design Structure Matrix (DSM) has seen considerable use for the modelling of engineering design processes as well as applications associated with parametric analysis and modular design [12]. Steward [13] originally developed the technique for application within the process-modelling domain, and has since seen considerable application for the modelling and management of the product development process [2, 3, 9, 14-27].

Whilst the DSM has been used to aid in the planning of CE projects, relatively little research has been undertaken in order to establish which of the activities within the DSM may be undertaken in parallel. Eppinger identified that parallel activities are independent and provided examples of groups of parallel activities within matrices [20]. The concept of "partitioning" the matrix was derived by Steward [13], and has been used to expose the tasks that may be undertaken both sequentially and in parallel [2]. The procedure of partitioning may be undertaken by "trial and error" to "minimise the number of informational feedbacks above the diagonal". However Eppinger identified the need for "a systematic approach involving the use of computer-based algorithms" for more complex processes [20]. This systematic-approach had however already been resolved by both Rogers and McCulley [28], and Scott [29] with the use of Genetic Algorithms to re-sequence the activities within the matrix with the aim of minimising the number of feedback dependencies.

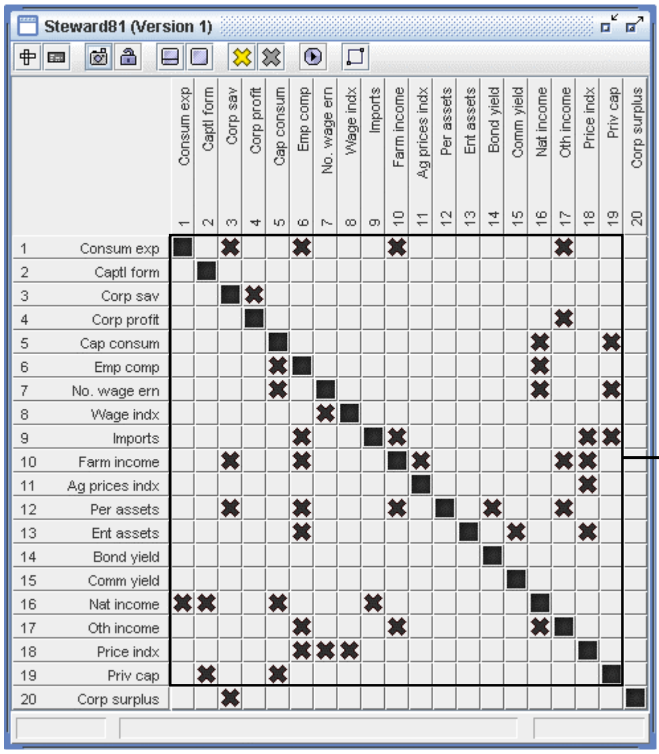
Where Scott considered the importance of "concurrency" when developing the "Scott Partitioning Procedure", the approach failed to consider the fundamental nature of the process that enables activities to be undertaken in parallel. The approach was primarily aimed at managing the process from a matrix perspective, rather than managing the process from a process perspective. That is, the concurrency within the process was considered through the objective of re-sequencing the activities within the matrix in order to get the dependencies as close to the bottom-left-hand corner of the matrix. Consideration is not given within this matrix-oriented approach for the distinguishing feature of parallel activities irrespective of the modelling approach adopted: independence.

The approach described within this paper identifies parallel activities from the perspective of the process through an examination of the relationships between activities – Section 2. The procedure for identifying parallel activities is then used to produce an algorithm called the "Kortabarria Parallelism Criterion" (KPC) that enables the performance of the process to be evaluated with respect to parallel activities. A Genetic Algorithm is then utilised to optimise the performance of the process with respect to parallel activities – Section 3. Consideration is also given towards the minimisation of feedback dependencies affecting rework. The approach identifies parallel activities, which may be undertaken concurrently providing that effective co-ordination is provided [11]. The assumption being that it is co-ordination that is required in order that parallel activities may be undertaken concurrently. The use of the term "activity" and not "task" is intentional – see [30] for a distinction between the two terms.

2 Identifying parallel activities

The objective of this investigation is to provide an approach and implementation that will enable the evaluation of the process performance with the aim of reducing lead-time. In order to achieve this, two characteristics of the process are considered: activity parallelism, and iteration. This section describes an approach for the identification of parallel activities within a process.

The main thrust of investigations undertaken to date with respect process improvement using the DSM has been towards the identification and evaluation of iteration. Rework arises through iteration where activities within the process require information to be estimated that would normally be generated within a later stage. The activities within the iterative loop may require re-enactment where the initial estimation differs significantly from the information that is eventually generated. Iteration is clearly presented within the DSM as a dependency between two activities above the diagonal and can be seen within Figure 1 between activities 1 and 17 for example. The further the dependency is above the diagonal, the greater number of activities that may be performed within the iterative cycle.



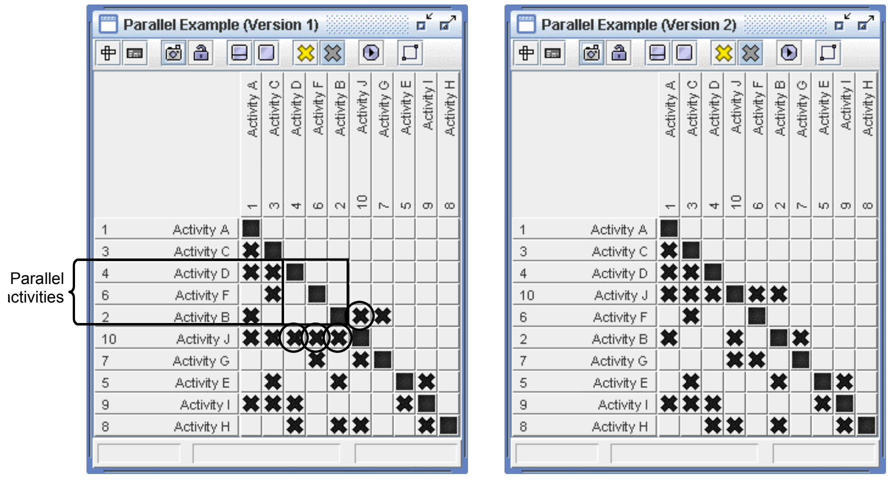
		Consum exp 1	Captl form 2	Corp sav 3	Corp profit 4	Cap consum 5	Emp comp 6	No. wage ern 7	Wage indx 8	Imports 9	Farm income 10	Ag prices indx 11	Per assets 12	Ent assets 13	Bond yield 14	Comm yield 15	Nat income 16	Oth income 17	Price indx 18	Priv cap 19	Corp surplus 20
1	Consum exp	■		✖			✖				✖							✖			
2	Captl form		■																		
3	Corp sav			■	✖																
4	Corp profit				■													✖			
5	Cap consum					■											✖			✖	
6	Emp comp					✖	■										✖				
7	No. wage ern					✖		■									✖			✖	
8	Wage indx							✖	■												
9	Imports						✖			■	✖								✖	✖	
10	Farm income			✖			✖				■	✖						✖	✖		
11	Ag prices indx											■							✖		
12	Per assets			✖			✖				✖		■		✖			✖			
13	Ent assets						✖							■		✖			✖		
14	Bond yield														■						
15	Comm yield															■					
16	Nat income	✖	✖			✖				✖							■				
17	Oth income						✖				✖						✖	■			
18	Price indx						✖	✖	✖										■		
19	Priv cap		✖			✖														■	
20	Corp surplus			✖																	■

Figure 1. Example process modelled within DSM format.

This ease of identification of iteration within the DSM has resulted with a great deal of interest in the application of the DSM for process improvement with respect to iteration with the development of "partitioning" and "tearing" approaches as well as algorithms to evaluate the process performance [2, 3, 9, 14-27]. The objective of partitioning is therefore to reduce the amount of iteration within the process through re-sequencing the activities to either eliminate the dependencies above the diagonal, or move the dependencies as close as possible to the diagonal.

Whilst Yassine and Braha have claimed that this partitioning procedure enables parallel activities to be identified [2], they do not state whether the partitioning procedure actively considers parallelism – with the aim of increasing it, or if the parallel activities result as a side-affect of the procedure. Scott produced an algorithm for measuring concurrency based on the DSM and used optimisation to re-sequence the DSM with the aim of improving the concurrency [29]. The algorithm was described as, "The sum of all of the data-dependencies of the matrix, each multiplied by a value which represents, for each matrix position, the scaled distance of the data dependency from the bottom-left-hand corner of the matrix for data-dependencies below the diagonal and from the leading-diagonal for data-dependencies above the leading diagonal". Optimisation using this algorithm would re-sequence the activities within the process such that: "their associated data dependencies are re-positioned in those matrix positions close to the bottom left-hand corner of the matrix, or alternatively, in those positions directly above the leading diagonal." The outcome of Scott's approach was the production of processes that were optimised entirely on the basis of the position of dependencies within the matrix, without any consideration of what the dependency represented from a process perspective.

A number of different types of relationships between activities were identified by Prasad [31]: dependent, semi-independent, independent, and inter-dependent. The semi-independent and independent activity relationships were defined as representing "pseudo-parallel" and parallel activities respectively. Whilst the process represented within Figure 1 includes a number of dependent and inter-dependent activities, it is apparent that there are no independent activities and it is difficult to determine if any of the activities are semi-dependent. Eppinger identified parallel (independent) activities with respect to the DSM - Figure 2(a). However such parallel activities were clearly identifiable when the activities were already grouped together.



		Activity A 1	Activity C 3	Activity D 4	Activity F 6	Activity B 2	Activity J 10	Activity G 7	Activity E 5	Activity I 9	Activity H 8
1	Activity A	■									
3	Activity C	✖	■								
4	Activity D	✖	✖	■							
6	Activity F		✖		■						
2	Activity B	✖				■	✖	✖			
10	Activity J	✖	✖	✖	✖	✖	■				
7	Activity G				✖		✖	■			
5	Activity E		✖			✖			■	✖	
9	Activity I	✖	✖	✖					✖	■	
8	Activity H			✖		✖	✖			✖	■

(a)

		Activity A 1	Activity C 3	Activity D 4	Activity J 10	Activity F 6	Activity B 2	Activity G 7	Activity E 5	Activity I 9	Activity H 8
1	Activity A	■									
3	Activity C	✖	■								
4	Activity D	✖	✖	■							
10	Activity J	✖	✖	✖	■	✖	✖				
6	Activity F		✖			■					
2	Activity B	✖			✖		■	✖			
7	Activity G				✖	✖		■			
5	Activity E		✖				✖		■	✖	
9	Activity I	✖	✖	✖					✖	■	
8	Activity H			✖	✖		✖			✖	■

(b)

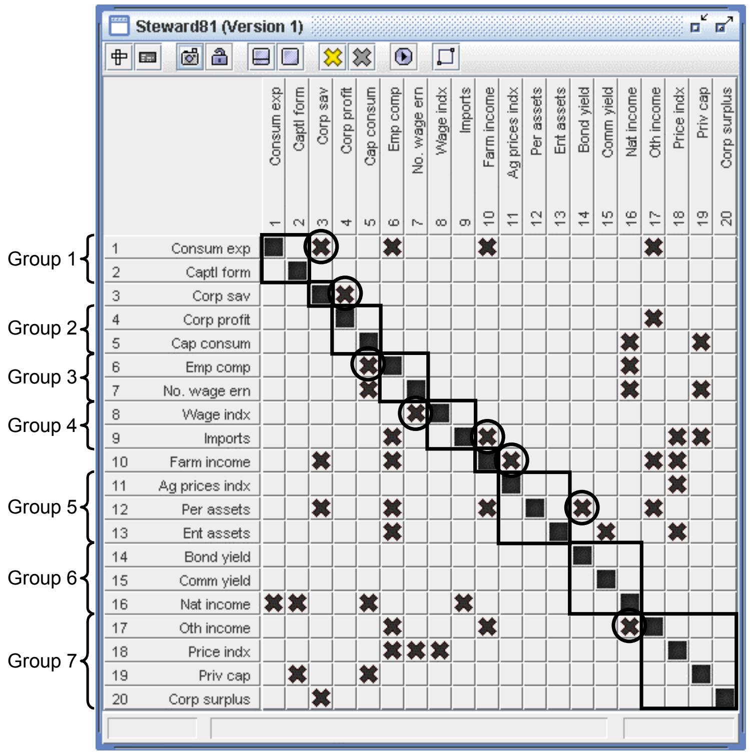
Figure 2. Affect of re-sequencing on the identification of parallel activities.

When the parallel activities are re-sequenced, within the DSM for example, they are no longer easily identifiable - Figure 2(b). In addition, Eppinger gave no consideration for semi-independent activities. Activities D and F within Figure 2(b) may still in principle be undertaken in parallel due to the nature of the dependencies between the activities. However the sequence represented within Figure 2(b) would suggest that there are no activities that may be undertaken in parallel. The semi-independent, and inter-dependent activities therefore require further effort to be identified. Since the focus of this investigation is the identification of parallel activities, only those that are parallel and "pseudo-parallel" are considered further.

Considering the matrix within Figure 2(a), it is possible to define a number of characteristics that may be used to identify parallel activities within the DSM:

- The highlighted square boxes represent the groups of parallel activities that contain no feed-forward or feedback dependencies and hence represent activities that are independent of each other.
- A new group of parallel activities is established when a dependency appears either under or to the right of the activities within the currently considered group – indicated by the circled dependencies within Figure 2(a). In addition a single dependency is sufficient to result with the creation of a new group of parallel activities, e.g. the dependency between activity J and D.
- The nature of the dependencies between groups indicates how groups may be overlapped.

Applying the above characteristics to the original sequence within Figure 1, results with the identification of seven groups of parallel activities and two sequential activities - Figure 3.



		Consum exp 1	Captl form 2	Corp sav 3	Corp profit 4	Cap consum 5	Emp comp 6	No. wage ern 7	Wage indx 8	Imports 9	Farm income 10	Ag prices indx 11	Per assets 12	Ent assets 13	Bond yield 14	Comm yield 15	Nat income 16	Oth income 17	Price indx 18	Priv cap 19	Corp surplus 20
1	Consum exp	■		✖			✖				✖							✖			
2	Captl form		■																		
3	Corp sav			■	✖																
4	Corp profit				■													✖			
5	Cap consum					■											✖			✖	
6	Emp comp					✖	■										✖				
7	No. wage ern					✖		■									✖			✖	
8	Wage indx							✖	■												
9	Imports						✖			■	✖								✖	✖	
10	Farm income			✖			✖				■	✖						✖	✖		
11	Ag prices indx											■							✖		
12	Per assets			✖			✖				✖		■		✖			✖			
13	Ent assets						✖							■		✖			✖		
14	Bond yield														■						
15	Comm yield															■					
16	Nat income	✖	✖			✖				✖							■				
17	Oth income						✖				✖						✖	■			
18	Price indx						✖	✖	✖										■		
19	Priv cap		✖			✖														■	
20	Corp surplus			✖																	■

Figure 3. Parallel activity groups identified using characteristics.

3 Evaluating activity parallelism and iteration using the DSM

DSM software designed within the CAD Centre of the University of Strathclyde was used as a basis upon which to construct an algorithm for identifying and evaluating parallel activities. Matrices may be created, with activities added, copied, cut and pasted between matrices. The software enables multiple activities to be selected by the user and dragged to various positions; changing the sequence, with the dependencies changing automatically, and the included evaluation criteria re-calculated. This mechanism enables the user to rapidly change the sequence of activities within the process and determine the impact of the changes on the performance of the process. The manual modification of the sequence does however produce some un-intuitive behaviour, especially when multiple activities are selected and dragged, due to the nature of the interactions between the rows and columns (the column is dragged automatically when the user drags a row and vice-versa). For this reason a genetic algorithm was implemented within the DSM software that enables the process to be optimised with respect to any number of performance criteria. Further information relating to the application and use of the Genetic Algorithm with respect to optimising processes modelled within the DSM can be found in [32].

3.1 Consideration of parallelism and iteration

As with the investigation undertaken by Scott [29], the focus of this research is in the production of an algorithm that may be used to consider both parallelism and iteration. In addition, Scott's approach was aimed at the modification of the process from a modelling perspective (the DSM), whereas the approach presented here represents the modification of the process from the process perspective. Scott quantified the negative impact of dependencies within the matrix, and used this as the basis to modify the matrix using an optimisation approach. The approach presented here has provided a number of rules in order to identify groups of parallel activities within a process irrespective of the modelling approach. In order to consider both parallelism and iteration, it is necessary to consider the impact of the feedback dependencies that result with iteration.

Feedback dependencies have a potentially detrimental effect on the process lead-time: requiring careful consideration of the nature of the feedback information in order to make an initial estimate, necessitating further consideration of the information once it is generated in order to validate the initial estimate, and resulting with possible repetition of the activities in order to address any discrepancies. It is however naïve to assume that the process may be completely re-engineered in order to remove these feedback dependencies. They occur naturally in many engineering processes and therefore require management in order to ensure that the iteration is being effectively co-ordinated and that the information within the process is correctly propagated and consistent [13]. From a process planning perspective, the objective is to minimise the impact with respect to the number of activities that would require re-enactment as a result of iteration.

Feed-forward dependencies are considered to be beneficial unless they result in the creation of a new group of parallel activities. A process consisting of entirely feed-forward dependencies is optimum from an iteration viewpoint since the process requires no rework and provides the most likelihood of determining a reliable lead-time for the process (assuming that resource allocation isn't an issue) since there is no need to estimate any of the information within the process – it is always generated by preceding activities. However Figure 3 indicates that certain feed-forward and feedback dependencies (those that are circled) limit the size of the parallel activity groups, which is clearly detrimental from the parallelism perspective. A trade-off exists with respect to the placement of the feed-forward dependencies; any feed-forward or feedback dependency that results with the creation of a new parallel activity group has a negative effect, whereas any feed-forward dependency that does not affect the creation of a parallel activity group has a positive effect. Any dependencies that cross over groups are considered beneficial. The dependencies between subsequent groups (such as the dependency between activity 6 and 5 in Figure 3) are considered detrimental, since these are responsible for the creation of group 4 for example.

The consideration of feedback dependencies is routine: any feedback dependency has a detrimental effect that is in proportion to the number of activities that would require re-enactment as a result of it. It is therefore possible to define a set of characteristics that may be used to aid the identification of a sequence of activities that consider both parallelism and iteration:

- The first feed-forward or feedback dependency that is responsible for creating a new group of parallel activities has a negative impact - Figure 4(a).
- Any other feed-forward dependencies, irrespective of whether they would have created a new group of parallel activities, are considered to have a positive impact - Figure 4(b).
- Any feedback dependencies have a negative impact in proportion to the number of activities that would be associated with the iteration - Figure 4(c).

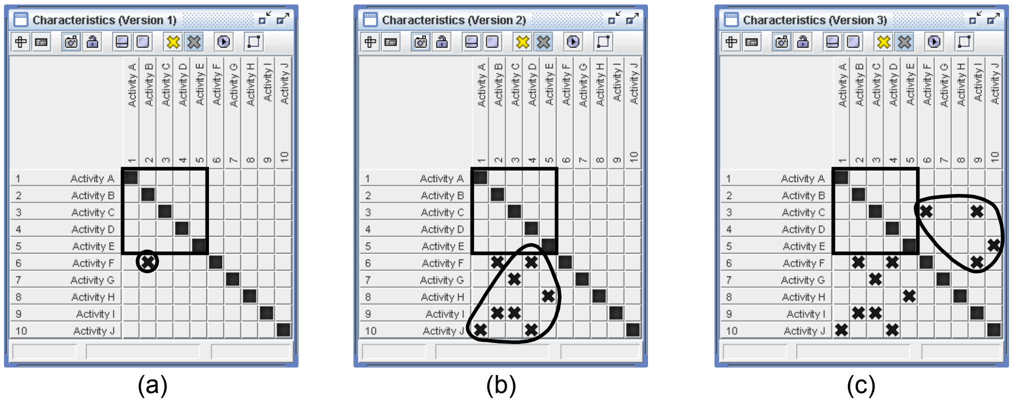


Figure 4. Dependencies affecting parallelism and iteration.

3.2 Process optimisation considering parallelism and iteration

The KPC algorithm was developed to evaluate the performance of the process using the rules defined above and was implemented for inclusion within the DSM software [33]. The algorithm considered the nature of the dependencies between the activities as defined above and allocated weightings for the various beneficial and detrimental effects. The focus when selecting the weightings was not on the absolute values, but on the relative beneficial and detrimental effects. The KPC algorithm was used within the optimisation by the GA and resulted with the sequence of activities as presented in a DSM format within Figure 5. The resulting process consists of five groups of parallel activities and three sequential activities – a reduction from nine design stages to eight compared with the original process - Figure 3.

The optimised sequence using the KPC also indicates a reduction in the number of feedback dependencies – from 22 within the original process to 10, indicating a possible reduction in the rework. The relatively small improvement in the number of parallel activities after optimisation would suggest that the original process sequence was already near optimal with respect to parallelism. The process represented within Figure 5 indicates a significant improvement in the iteration as well as a small increase in parallelism. The algorithm has produced a trade-off between the positive impact of the feed-forward dependencies and the negative impact of the feedback dependencies and the dependencies creating new parallel activity groups.

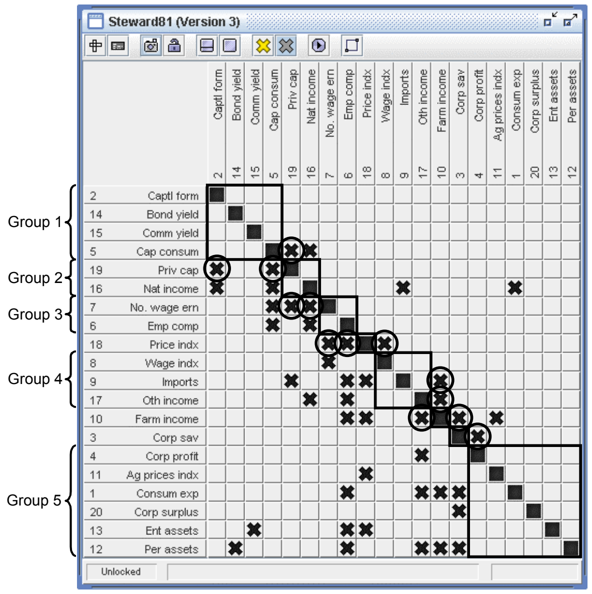


Figure 5. Process optimised for parallel activities and iteration using KPC.

Using the Scott combined concurrency and iteration procedure, it is possible to improve the result from the perspective of iteration - Figure 6. The number of feedback dependencies within Figure 6 has been further reduced to five, whereas the process has four parallel activity groups with eight sequential activities – an increase in the number of stages compared to the original process.

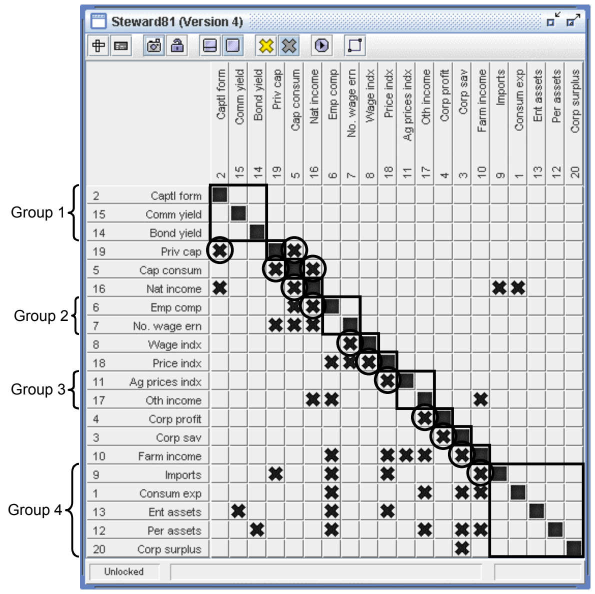


Figure 6. Process optimised using Scott Partitioning Procedure.

Despite Scott considering “concurrency” within the design of the criterion, the process resulting from optimisation using the Scott procedure indicates a reduction with respect to number of activities that may be undertaken in parallel when compared with the original process. Benefit may however be seen with respect to the reduction in the number of feedback dependencies. The optimised feedback process indicated within Figure 6 would suggest that the process within Figure 5 achieved using the KPC has traded an increase in parallelism for a slight increase in iteration.

The trade-off between the parallelism and the feedback may be modified through re-considering the relative positive and negative impacts of the dependencies. From a process management and enactment perspective, close communication is necessary between the upstream and downstream activities in order to reduce the possibility of rework within the groups having overlapping dependent activities [1, 34], as well as “analysing and optimising the flow of information between team members” [2].

3.3 Consideration of pseudo-parallelism

The matrices used in the examples so far have included dependencies that have been of a single weight and may therefore be considered as being “binary matrices”, where the activities are either dependent or independent [13]. One limitation of the binary matrix is that it could potentially become densely populated with dependencies, irrespective of whether the relationship between the activities was weak or strong. Gebala and Eppinger stated [22]: “No attempt is made to differentiate between amounts of information transferred between tasks in the matrix. It is reasonable to expect that certain dependencies will be stronger than others, or that certain transfers of information will be critical.” The binary matrix was therefore enhanced through the introduction of numerical dependencies that provide: “different measures of the relation between the tasks or the task’s relation to the entire design process.” Eppinger *et al.* considered the grouping of the weights to represent strong, medium and weak dependencies [18]. For example a dependency was considered weak if either: the heavily dependent information that was transferred between the activities was known to lie within predictable limits and was therefore relatively easy to estimate, or the slightly dependent information was known to be unpredictable. Eppinger *et al.* suggested the use of a signal-to-noise ratio, where the mean part of the ratio would correspond to the degree of dependency between the activities, and the deviation part would correspond to the predictability of the information. A scheme was then proposed for grouping the dependencies which was further established by Austin *et al.* [14] and can be seen within Figure 7. Forbes *et al.* [21] and Kortabarria [33] defined how these different classes of relationship might affect the overlapping of the associated activities.

Class A	It is essential to an activity that Class A data is available before its commencement.
Class B	It is not essential to an activity that Class B data is available before its commencement but it is preferable.
Class C	It is not essential to an activity that Class C data is available before its commencement.
Zero – Class D	No data is required.

Figure 7. Four-level dependency scheme [14].

Figure 8 shows a system for defining how the activities may be overlapped that is derived from [33] and is based on the dependency scheme developed by Austin *et al.* Class A represents activities that would be enacted sequentially, with the information that is required by Activity B being generated on completion of Activity A. Using the concepts of evolution and sensitivity defined by Krishnan *et al.* [5], the Class A relationship defined with Figure 8 could be defined as having a slow evolution of data from Activity A, and a high sensitivity to changes in data within Activity B. Class B indicates that Activity B may start at some point during the enactment of Activity A, when the information is of a suitable degree of evolution and known to vary less than the sensitivity limits for Activity B. Class B assumes that Activity B requires additional information from the completion of Activity A in order to continue enactment. Without this information Activity B could not be completed. Class C indicates that Activity A and Activity B may run in parallel with information being transferred at various points during their enactment if necessary but not affecting the progress of the activities. Activity B is assumed to be relatively insensitive to changes in the rapidly evolved information from Activity A. Class D indicates that no information is transferred between activities A and B.

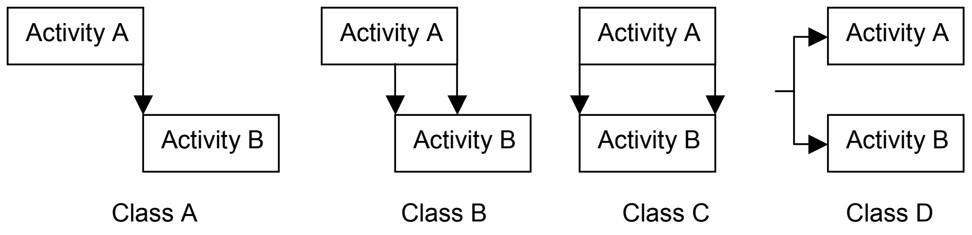


Figure 8. Activity overlapping states for four-level dependency scheme [33].

Given the dependency scheme defined by Austin *et al.* [14] and the overlapping states defined by Forbes *et al.* [21] and Kortabarria [33], it is apparent that it is not only Class D relationships that may be enacted in parallel, but also Class C, and Class B to an extent that is defined by the nature and context of the activities.

Since Class C activity relationships indicate that the associated activities may be enacted in parallel, a caveat is included to the rules used to define how new parallel activity groups are identified enabling consideration of pseudo-parallel activities:

- A new group of parallel activities is established when a dependency appears either under or to the right of the activities within the currently considered group, *unless it is a Class C dependency*.

Within the present investigation, Class A and Class B dependencies were considered to result with the creation of a new parallel activity group. Since the nature and context of the interaction between activities of Class B is managed during the enactment of the activities and is an information concurrence or co-ordination issue, this type of dependency was not considered within the current investigation.

Austin *et al.* [14] provided a building design management process consisting of 51 activities inter-related with Class A, B and C dependencies - Figure 9. The process consists of 13 parallel activity groups accounting for 32 of the activities, with 19 sequential activities giving 32 design stages. The process has 63 feedback dependencies, 25 Class A, 23 Class B and 15 Class C. Optimising the sequence of activities within a process of this size is not a trivial task: 51 activities yields 51! or $15*10^{66}$ possible sequences. Whitfield *et al.* [32] describe an investigation regarding the efficient use of Genetic Algorithms for application to DSM process optimisation. The GA settings used for the optimisation of the process represented within Figure 9 were derived from [32].

The process was optimised using the KPC as the objective function and the results of the optimisation can be seen within Figure 10. The optimised process consists of 16 parallel activity groups accounting for 46 of the activities within the process, with the remaining five activities requiring sequential enactment, and represents 21 design stages. The optimised process has a 43% increase in the number of identified parallel activities. A considerable reduction in the number of feedback dependencies can also be seen with 8 Class A, 15 Class B, and 18 Class C dependencies, giving 41 feedback dependencies in total. The most significant impact has been made on the number and position of the critical Class A feedback dependencies. The number of Class A feedback dependencies was reduced by 68%, whilst these dependencies were all placed adjacent to the diagonal where the impact of the dependency is minimised. It would of course be preferable to have no Class A, B or C feedback dependencies, however placing them adjacent or close to the diagonal reduces the number of activities that would require re-enactment in the event that an unsuitable estimate is made for the input. It can also be seen that pseudo-parallelism has been accounted for with the placement of three Class C dependencies within the parallel groups 2, 9 and 15. The results from the optimisation would again indicate that the algorithm has produced a trade-off between the number of parallel activities within the process and the amount of rework generated through iterative feedback – it would be possible to reduce the feedback further at the expense of reducing the number of parallel activities.

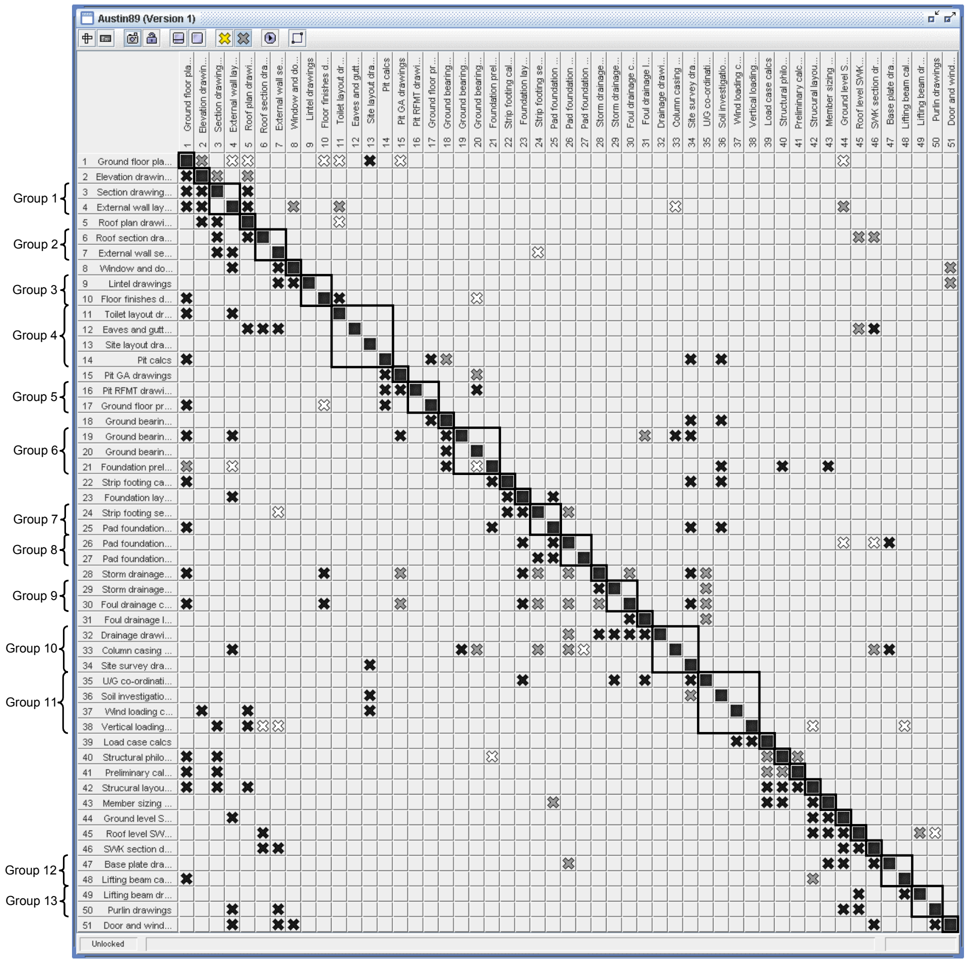


Figure 9. Building design process [14].

The process given within Figure 9 was again optimised using the Scott combined concurrency and iteration procedure. The resulting optimised process had a reduction in the number of groups of parallel activities compared with the original process from 13 to 12. The Scott procedure however demonstrated considerable success with respect to iteration reducing the total number of feedback dependencies from 63 to 37, with three Class A, 18 Class B, and 16 Class C dependencies. These results along with those demonstrated within Figure 6 suggest that the Scott procedure is biased towards iteration – trading off concurrency. It is of course possible to change the bias for both the KPC and the Scott procedure. A neutral weighting was however used within the Scott procedure, with the iteration and concurrency components having equal influence. Further investigation is however necessary to determine what the optimum trade-off is between iteration and concurrency with respect to lead-time. For example, is it more advantageous to undertake a greater number of activities in parallel whilst having to make a greater number of estimates (and potentially increase rework), or is it preferable to undertake a lesser number of activities in parallel (a sequential process) whilst reducing the number of estimates.

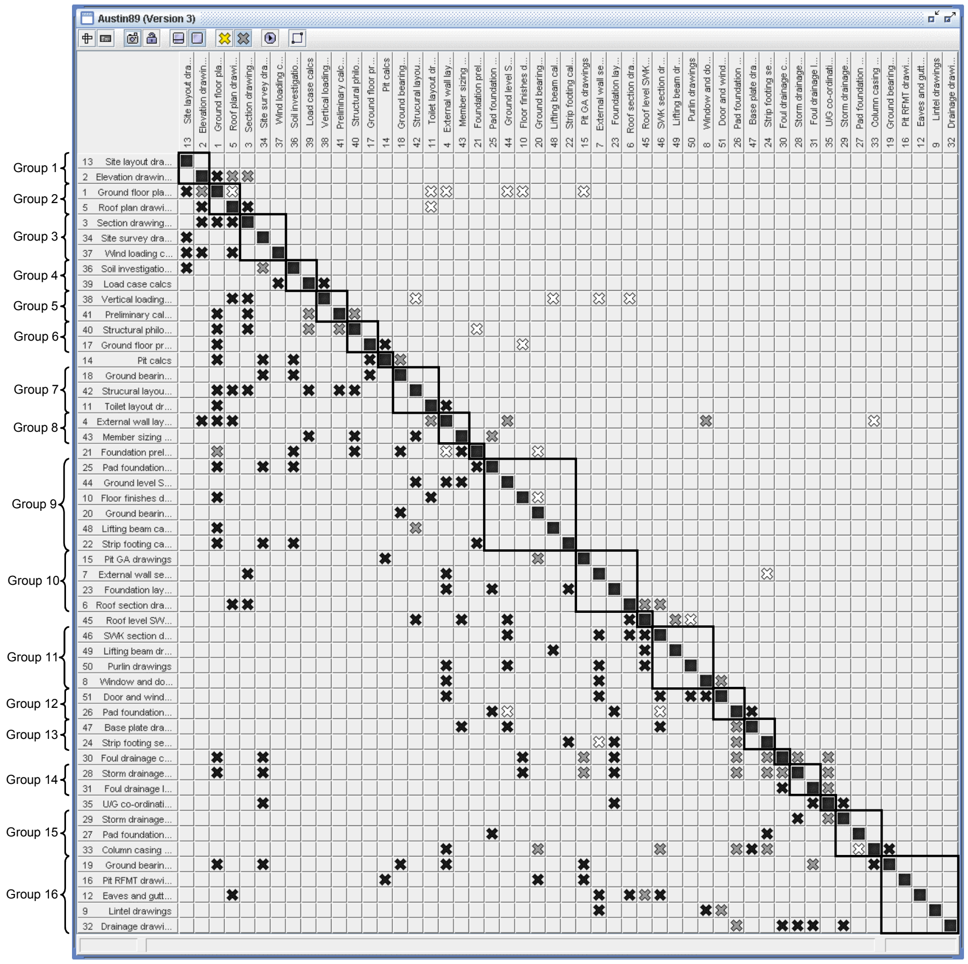


Figure 10. Building process optimised for parallel activities and iteration using KPC.

The results generated using through the optimisation of the process using the KPC algorithm compare favourably with the original results produced by Austin for the re-partitioned process. Austin's process produced 15 groups of parallel activities accounting for 71% of the activities (compared with 16 groups accounting for 90% of the activities using the KPC). In addition, similar numbers of feedback dependencies were generated by Austin – a total of 40, consisting of three Class A, 20 Class B, and 17 Class C.

4 Conclusion

An approach was described that provides rules that may be used for identifying groups of parallel activities. An algorithm was produced based upon these rules that enabled the performance of processes modelled using the DSM to be evaluated with respect to the number of activities that may be enacted in parallel as well as the amount of iteration within the process. The KPC considered the trade-off between increasing the number of parallel activities and its affect on the amount of iteration and hence rework.

The results suggested that parallel activities could be identified and evaluated within the DSM using the presented approach. The processes were optimised using a Genetic Algorithm to re-sequence the activities with the aim of maximising the KPC: resulting with an increase in the number of parallel activities in both processes presented.

The approach is used primarily as a method to determine which activities may be undertaken in parallel, in order to facilitate the generation and management of a concurrent product development process. The algorithm used to calculate the KPC does not currently consider the overlapping of groups of parallel activities, however this may be considered in the future.

The approach that has been demonstrated forms one stage that may be used to facilitate Concurrent Engineering and addresses two of the principles defined by Yassine and Braha: iteration and overlapping. It suggests which activities may be undertaken in parallel representing an aspect of strategic CE. Undertaking the activities concurrently, or operational CE, requires knowledge relating to the resource availability, deadlines, other projects, as well as effective co-ordination in order ensure "the timely availability of critical design information".

References

1. Loch, C.H., C. Terwiesch, *Communication and uncertainty in concurrent engineering.* Management Science, 1998. **44**(8): p. 1032-1048.

2. Yassine, A., D. Braha, *Complex concurrent engineering and the Design Structure Matrix method.* Concurrent Engineering Research and Applications, 2003. **11**(3): p. 165-176.

3. Smith, R.P., Eppinger, S.D., *Identifying controlling features of engineering design iteration.* Management Science, 1997. **43**(3): p. 276-293.

4. Loch, C.H., C. Terwiesch, *Product development and Concurrent Engineering*, in *Encyclopaedia of Production and Manufacturing Management*, P.M. Swamidass, Editor. 2000, Kluwer Academic Publishing. p. 567-575.

5. Krishnan, V., S.D. Eppinger, D.E. Whitney, *A model-based framework to overlap product development activities.* Management Science, 1997. **43**(4): p. 437-451.

6. Terwiesch, C., C.H. Loch, *Measuring the effectiveness of overlapping development activities.* Management Science, 1999. **45**(4): p. 455-465.

7. Yan, J.H., C. Wu, *Scheduling approach for concurrent product development processes.* Computers in Industry, 2001. **46**(2): p. 139-147.

8. Browning, T.R., S.D. Eppinger, *Modeling impacts of process architecture on cost and schedule risk in product development.* IEEE Transactions on Engineering Management, 2002. **49**(4): p. 428-442.

9. Carrascosa, M., S.D. Eppinger, D.E. Whitney. *Using the Design Structure Matrix to estimate product development time*. in *1998 ASME Design Engineering Technical Conferences*. 1998. Atlanta, Georgia: ASME.

10. Joglekar, N.R., A.A. Yassine, S.D. Eppinger, D.E. Whitney, *Performance of coupled product development activities with a deadline.* Management Science, 2001. **47**(12): p. 1605-1620.

11. Duffy, A.H.B., M.M. Andreasen, K.J. MacCallum, L.N. Reijers, *Design coordination for Concurrent Engineering.* Journal of Engineering Design, 1993. **4**(4): p. 251-265.

12. Whitfield, R.I., J.S. Smith, A.H.B. Duffy. *Identifying component modules*. in *Artifical Intelligence in Design*. 2002. Cambridge, UK.

13. Steward, D.V., *The design structure system: a method for managing the design of complex systems.* IEEE Transactions on Engineering Management, 1981. **EM-28**(3): p. 71-74.

14. Austin, S., A. Baldwin, A. Newton, *A data flow model to plan and manage the building design process.* Journal of Engineering Design, 1996. **7**(1): p. 3-25.

15. Browning, T.R., *Applying the Design Structure Matrix to system decomposition and integration problems: a review and new directions.* IEEE Transactions on Engineering Management, 2001. **48**(3): p. 292-306.

16. Choo, H.J., J. Hammond, I.D. Tommelein, S.A. Austin, G. Ballard, *DePlan: a tool for integrated design management.* Automation in Construction, 2004. **13**: p. 313-326.

17. Denker, S., D. Steward, T. Browning, *Planning concurrency and managing iteration in projects.* Centre for Quality of Management Journal, 1999. **8**(2): p. 55-62.

18. Eppinger, S.D., D. E. Whitney, R. P. Smith, D. A. Gebala, *A model-based method for organising tasks in product development.* Research in Engineering Design, 1994. **6**: p. 1-13.

19. Eppinger, S.D. *A planning method for integration of large-scale engineering systems*. in *International Conference on Engineering Design*. 1997. Tampere, Finland.

20. Eppinger, S.D., *Innovation at the speed of information.* Harvard Business Review, 2001: p. 1-11.

21. Forbes, G.A., D.A. Flemming, A.H.B. Duffy, P.D. Ball. *The optimisation of a strategic business process*. in *20th International Manufacturing Conference IMC-20*. 2003. Cork, Ireland.

22. Gebala, D.A., S.D. Eppinger. *Methods for analysing design procedures*. in *ASME Conference on Design Theory and Methodology*. 1991. Miami, USA: ASME.

23. Kusiak, A., J. Wang, *Efficient organising of design activities.* International Journal of Production Research, 1993. **31**(4): p. 753-769.

24. Kusiak, A., J. Wang. *Concurrent Engineering: simplification of the design process*. in *Computer Applications in Production and Engineering: Integration Aspects*. 1991. Bordeaux, France.

25. Rogers, J.L., C.L. Bloabaum. *Ordering design tasks based on coupling strengths*. in *5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimisation*. 1994. Panama City, Florida.

26. Tang, D., L. Zheng, Z. Li, D. Li, S. Zhang, *Re-engineering of the design process for concurrent engineering.* Computers & Industrial Engineering, 2000. **38**: p. 479-491.

27. Yassine, A.A., D.E. Whitney, T. Zambito. *Assessment of rework probabilities for simulating product development processes using the Design Structure Matrix*. in *ASME 2001 International Design Engineering Technical Conferences - Computers and Information in Engineering Conference*. 2001. Pittsburgh, USA: ASME.

28. Rogers, J.L., C.M. McCulley, *Integrating a genetic algorithm into a knowledge-based system for ordering complex design processes*. 1996, National Aeronautics and Space Administration: Virginia. p. 13.

29. Scott, J.A., *A strategy for modelling the design-development phase of a product*, in *Department of Marine Engineering*. 1998, University of Newcastle upon Tyne: Newcastle upon Tyne. p. 234.

30. Duffy, A.H.B. *Designing Design*. in *3rd International Workshop on Engineering Design and Integrated Product Development - Design Methods that Work*. 2002. Lagow, Poland.

31. Prasad, B., *Concurrent Engineering Fundamentals: Integrated Product and Process Organisation*. Vol. 1. 1995: Prenctice Hall. 502.

32. Whitfield, R.I., A.H.B. Duffy, G. Coates, W. Hills, *Efficient process optimisation.* Concurrent Engineering Research and Applications, 2003. **11**(2): p. 83-92.

33. Kortabarria Gartzia-Etxabe, L., *Design process optimisation*, in *Design Manufacture and Engineering Management*. 2004, University of Strathclyde: Glasgow. p. 79.

34. Ha, A.Y., E.L. Porteus, *Optimal review frequency in concurrent engineering.* Management Science, 1995. **41**(9): p. 1431-1447.

Robert Ian Whitfield

CAD Centre, DMEM, University of Strathclyde, 75 Montrose Street, Glasgow G1 1XJ, UK
Phone: +44-141-548 3020 Fax: +44-141-552 7896 Email: ianw@cad.strath.ac.uk